EDF R&D



Fluid Dynamics, Power Generation and Environment Department
Single Phase Thermal-Hydraulics Group

6, quai Watier
F-78401 Chatou Cedex

Tel: 33 1 30 87 75 40
Fax: 33 1 30 87 79 16

APRIL 2026

code_saturne documentation

code_saturne 8.1 Theory Guide

contact: saturne-support@edf.fr



code_saturne

ABSTRACT

code_saturne solves the Navier-Stokes equations for 2D, 2D axisymmetric, or 3D, steady or unsteady, laminar or turbulent, incompressible or dilatable flows, with or without heat transfer, and with possible scalar fluctuations. The code also includes a Lagrangian module, a semi-transparent radiation module, a gas combustion module, a coal combustion module, an electric module (Joule effect and electric arc) and a compressible module. In the present document, the "gas combustion", "coal combustion", "electric" and "compressible" capabilities of the code will be referred to as "particular physics". The code uses a finite volume discretization. A wide range of unstructured meshes, either hybrid (containing elements of different types) and/or non-conform, can be used.

This document constitutes the theory guide associated with the kernel of code_saturne. The system of equations considered consists of the Navier-Stokes equations, with turbulence and passive scalars. Firstly, the continuous equations for mass, momentum, turbulence and passive scalars are presented. Secondly, information related to the time scheme is supplied. Thirdly, the spatial discretisation is detailed: it is based on a co-located[1] finite volume scheme for unstructured meshes. Fourthly, the different source terms are described. Fithly, boundary conditions are detailed. And finally, some algebrae such as how to solve a non-linear convection diffusion equation and some linear algebrae algorithms are presented.

In a second part, advanced modellings such as Combustion, electric and compressible flows are presented with their particular treatments.

To make the documentation suitable to the developers' needs, the appendix has been organized into sub-sections corresponding to the major steps of the algorithm and to some important subroutines of the code.

During the development process of the code, the documentation is naturally updated as and when required by the evolution of the source code itself. Suggestions for improvement are **more than** welcome. In particular, it will be necessary to deal with some transverse subjects (such as parallelism, periodicity) which were voluntarily left out of the first versions, to focus on the algorithms and their implementation.

To make it easier for the developers to keep the documentation up to date during the development process, the choice is made to not base this document on the implementation (except in the appendix) but to keep as much as possible a general formulation. For developers who are interested in the way theory is implemented, please refer to the `doxygen` documentation. A special effort will be made to link this theory guide to the `doxygen` documentation.

code_saturne is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. code_saturne is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

[1] All the variables are located at the centres of the cells.

Table of contents

1 Introduction 7

1.1 Aims of the document 8

I Generic solver capabilities 9

2 Governing equations 10

2.1 Continuous mass and momentum equations 11

2.2 Thermal equations 14

2.3 Equations for scalars 16

3 Time stepping 19

3.1 Time discretisation of a transport equation 20

3.2 Pressure-based velocity-pressure solver 22

4 Space discretization 26

4.1 Introduction 27

4.2 Convective term 29

4.3 Diffusive term 31

4.4 Gradient calculation 32

4.5 Compatible Discrete Operator (CDO) schemes 41

4.6 Advanced topics 42

5 Boundary conditions 48

5.1 Introduction 49

5.2 Standard user boundary conditions 50

5.3 Internal coding of the boundary conditions – Discretization 51

5.4 Wall boundary conditions 53

6 Algebrae 65

6.1 Iterative process 66

6.2 Linear algebrae 66

II Advanced modelling 67

7 Turbulence modelling 68
7.1 Eddy viscosity Models (*EVM*) 69
7.2 Differential Reynolds Stress Models (*DRSM*) 75
7.3 Large-Eddy Simulation (*LES*) 76
7.4 Turbulence models for velocity – scalar correlations 78

8 Compressible flows 81
8.1 Pressure-based solver 82

9 Combustion 83
9.1 Introduction 84
9.2 Thermodynamics 87
9.3 Gas combustion 88
9.4 Coal, Biomass, Heavy Oil combustion 97

10 Groundwater flows 113
10.1 Introduction 114
10.2 Groundwater flows 114
10.3 Soil-water relationships 115
10.4 Solving of the Richards equation 116
10.5 Groundwater Transfers 119
10.6 Alternative models to treat the soil-water partition of solutes 122

11 Magneto-Hydro Dynamics 124

12 Lagrangian particle tracking 125
12.1 Trajectography 126

13 Atmospheric flow modelling 129
13.1 Atmospheric flow modelling 130
13.2 Method of imbrication in code_saturne Atmospheric physics (MICA) . . . 132
13.3 Radiation parameterizations for atmosphere 132
13.4 Warm cloud parameterization 140
13.5 Earth-atmosphere interactions 145

14 Arbitrary Lagrangian Eulerian 151

15 Cavitation modelling 152
15.1 System equations 153
15.2 Vaporization source term 153
15.3 Time stepping 154

III Appendices 155

IV Base module 158

A- bilsc2 routine 160

B- cs_turbulence_ke routine 170

C- clptur routine 176

D- clptrg routine 192

E- clsyvt routine 202

F- codits routine 210

G- condli routine 216

H- gradrc routine 224

I- inimas routine 226

J- itrmas/itrgrp routine 230

K- matrix routine 232

L- cs_solve_navier_stokes routine 238

M- cs_velocity_prediction routine 248

N- resopv routine 260

O- turrij routine 270

P- viscfa routine 284

Q- visort routine 286

R- visecv routine 290

V Compressible module 293

A- cfbl** routine 294

B- cfener routine 312

C- cfmsvl routine 318

D- cfqdmv routine 324

E- cfxtcl routine 328

VI Electric Arcs 348

A- cs_elec_model routine 350

VII Mesh Handling 368

B Mesh Algorithms 369

C Mesh Quality 386

D Extended neigbourhood 389

D.1 Extended cell neighbourhood 390

D.2 Optimized (heuristics) neighbourhood 390

VIII Appendices 393

D.1 Tensorial and index notations 394

D.2 Differential operators and standard relationships 394

IX References 396

Nomenclature

Greek symbols

α_1	mass fraction of the continuous phase	
$\alpha_{2,\,i}$	mass fraction of the particle class i	
$\underline{\underline{\varepsilon}}$	turbulent kinetic energy dissipation tensor	$m^2.s^{-3}$
ε	turbulent kinetic energy dissipation	$m^2.s^{-3}$
Γ	mass source term	
μ	dynamic viscosity	$kg.m^{-1}.s^{-1}$
μ_l	dynamic molecular viscosity	$kg.m^{-1}.s^{-1}$
V_c	the cell c	
$\lvert V_c \rvert$	volume of the cell c	m^3
$\underline{\underline{\Phi}}$	pressure-velocity correlation tensor	$kg.s^{-3}$
ρ	density field	$kg.m^{-3}$
ρ_m	bulk density	kg/m^3
$\underline{\underline{\sigma}}$	total stress tensor	Pa
$\underline{\underline{\tau}}$	viscous stress tensor, which is the deviatoric part of the stress tensor	Pa

Operators

$\left(\underline{\underline{.}}\right)^D$	deviatoric part of a tensor
:	double dot product
$\left(\underline{\underline{.}}\right)^S$	symmetric part of a tensor
$tr\left(\underline{\underline{.}}\right)$	trace of a tensor

Roman symbols

$\mathcal{G}$	turbulent kinetic energy buoyancy term	$kg.m^{-1}.s^{-3}$
G_ε	turbulent buoyancy term for dissipation	
$\underline{\underline{\mathcal{G}}}$	turbulent buoyancy production tensor	$kg.m^{-1}.s^{-3}$
C_{ε_1}	constant of the standard $k-\varepsilon$ model	
C_{ε_2}	constant of the standard $k-\varepsilon$ model	

C_{ε_3}	constant of the standard $k-\varepsilon$ model depending on the buoyancy term	
C_μ	eddy viscosity constant	
f	interior or boundary cell face	
$\mathcal{F}_c$	group of all faces of the cell c	
$\underline{x}_f$	center of the face $f_{c\|\overline{c}}$ between cells c and $\overline{c}$	
H_m	bulk enthalpy	J/kg
$\underline{x}_c$	centre of V_c	
k	turbulent kinetic energy	$m^2.s^{-2}$
$\underline{\underline{K}}$	tensor of the velocity head loss	s^{-1}
P	pressure field	Pa
$P_{f_{c\|\overline{c}}}$	average of the pressure field on the interface between the neighbouring cells c and $\overline{c}$	Pa
P_m	pressure of the bulk	Pa
$\mathcal{P}$	turbulent kinetic energy production	$kg.m^{-1}.s^{-3}$
$\underline{\underline{\mathcal{P}}}$	turbulent production tensor	$kg.m^{-1}.s^{-3}$
R_{ij}	componant ij of the Reynolds stress tensor	$m^2.s^{-2}$
$\underline{r}$	velocity density correlation vector $\overline{\rho'\underline{u}'}$, generally modelled by a Generalized Gradient Diffusion Hypothesis (GGDH): $\frac{3}{2}\frac{C_\mu}{\sigma_t}\frac{k}{\varepsilon}\underline{\underline{R}}\cdot\underline{\nabla}\rho$	
$\underline{\underline{R}}$	Reynolds stress tensor	$m^2.s^{-2}$
$\underline{S}_{c>\overline{c}}$	outward normal vector of the face f of the cell c, normalized by the surface $\|\underline{S}\|$	
$\underline{\underline{S}}$	strain rate tensor	s^{-1}
ST_ε	additional turbulent dissipation source term	$kg.m^{-1}.s^{-4}$
ST_k	additional turbulent kinetic energy source term	$kg.m^{-1}.s^{-3}$
$\underline{ST}_{\underline{u}}$	explicit additional momentum source terms	$kg.m^{-2}.s^{-2}$
t	time [s]	
$\underline{u}_m$	bulk velocity	m/s
$\underline{u}$	velocity field	$m.s^{-1}$
x_1	mass fraction of continuous phase	
x_2	mass fraction of the particle phase	
$x_{2,i}$	mass fraction of the particle class i	

Chapter 1

Introduction

Disclaimer

code_saturne is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

code_saturne is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.[1]

1.1 Aims of the document

This chapter constitutes an introduction to the theory guide associated with the kernel of code_saturne. The system of equations considered consists of the Navier-Stokes equations, with turbulence and passive scalars. Firstly, the continuous equations for mass, momentum, turbulence and passive scalars are presented. Secondly, information related to the time scheme is supplied. Thirdly, the spatial discretisation is detailed: it is based on a co-located[2] finite volume scheme for unstructured meshes. Fourthly, the different source terms are described. Fifthly, boundary conditions are detailed. And finally, some algebrae such as how to solve a non-linear convection diffusion equation and some linear algebrae algorithms are presented.

In a second part, advanced modellings are presented with their particular treatments.

To make the documentation suitable to the developers' needs, the appendix has been organized into sub-sections corresponding to the major steps of the algorithm and to some important subroutines of the code.

During the development process of the code, the documentation is naturally updated as and when required by the evolution of the source code itself. Suggestions for improvement are **more than** welcome. In particular, it will be necessary to deal with some transverse subjects (such as parallelism, periodicity) which were voluntarily left out of the first versions, to focus on the algorithms and their implementation.

To make it easier for the developers to keep the documentation up to date during the development process, the choice is made not to base this document on the implementation (except in the appendix) but to keep as much as possible a general formulation. For developers who are interested in the way theory is implemented, please refer to the `doxygen` documentation (see local html documentation). A special effort will be made to link this theory guide to the `doxygen` documentation.

[1] You should have received a copy of the GNU General Public License along with code_saturne; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

[2] All the variables are located at the centres of the cells.

Part I

Generic solver capabilities

Chapter 2

Governing equations

2.1 Continuous mass and momentum equations

This section presents the continuous equations. It is no substitute for the specific sub-sections of this documentation: the purpose here is mainly to provide an overview before more detailed reading.

Balance methodology: The continuous equations can be obtained applying budget on the mass, momentum, or again on mass of a scalar. A useful theorem, the so-called Leibniz theorem, states that the variation of the integral of a given field A over a moving domain Ω reads:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_V A\mathrm{d}V\right) = \int_V \frac{\partial A}{\partial t}\mathrm{d}V + \int_{\partial V} A\underline{v}\cdot\mathrm{d}\underline{S}, \tag{I.2.1}$$

where $\underline{v}$ is the velocity of the boundary of Ω and $\partial\Omega$ is the boundary of Ω with a outward surface element $\mathrm{d}\underline{S}$.

2.1.1 Laminar flows

Mass equation: Let now apply (I.2.1) to a fluid volume[1], so $\partial\Omega$ moves with the fluid velocity denoted by $\underline{u}$, and to the field $A=\rho$, where ρ denotes the density:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_V \rho\mathrm{d}V\right) &= \int_V \frac{\partial\rho}{\partial t}\mathrm{d}V + \int_{\partial V}\rho\underline{u}\cdot\mathrm{d}\underline{S},\\ &= \int_V\left(\frac{\partial\rho}{\partial t}+\mathrm{div}\,(\rho\underline{u})\right)\mathrm{d}V,\end{aligned} \tag{I.2.2}$$

the second line is obtained using Green relation. In (I.2.2), the term $\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_V\rho\mathrm{d}V\right)$ is zero because[2] it is the variation of the mass of a fluid volume. This equality is true for any fluid volume, so if the density field and the velocity field are sufficiently regular then the **continuity** equation holds:

$$\frac{\partial\rho}{\partial t}+\mathrm{div}\,(\rho\underline{u})=0. \tag{I.2.3}$$

Equation (I.2.3) could be slightly generalized to cases where a mass source term Γ exists:

$$\frac{\partial\rho}{\partial t}+\mathrm{div}\,(\rho\underline{u})=\Gamma, \tag{I.2.4}$$

but Γ is generally taken to 0.

Momentum equation: The same procedure on the momentum gives:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_V \rho\underline{u}\mathrm{d}V\right) &= \int_V \frac{\partial(\rho\underline{u})}{\partial t}\mathrm{d}V + \int_{\partial V}\underline{u}\otimes(\rho\underline{u})\cdot\mathrm{d}\underline{S},\\ &= \int_V \frac{\partial(\rho\underline{u})}{\partial t}+\underline{\mathrm{div}}\,(\underline{u}\otimes\rho\underline{u})\,\mathrm{d}V,\end{aligned} \tag{I.2.5}$$

once again, the Green relation has been used to obtain the second line. One then invokes Newton's second law stating that the variation of momentum is equal to the external forces:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_V \rho\underline{u}\mathrm{d}V\right) &= \underbrace{\int_{\partial V}\underline{\underline{\sigma}}\cdot\mathrm{d}\underline{S}}_{\text{boundary force}} + \underbrace{\int_V\rho\underline{g}\mathrm{d}V}_{\text{volume force}},\\ &= \int_V \underline{\mathrm{div}}\,(\underline{\underline{\sigma}})+\rho\underline{g}\mathrm{d}V,\end{aligned} \tag{I.2.6}$$

[1] A fluid volume consists of fluid particles, that is to say it moves with the fluid velocity.

[2] It can be non-zero in some rare cases when fluid is created by a chemical reaction for instance.

where $\underline{\underline{\sigma}}$ is the Cauchy stress tensor[3], $\underline{g}$ is the gravity field. Other source of momentum can be added in particular case, such as head losses or Coriolis forces for instance.

Finally, bringing (I.2.5) and (I.2.6) all together the **momentum** equation is obtained:

$$\frac{\partial}{\partial t}(\rho \underline{u}) + \underline{\mathrm{div}}\ (\underline{u} \otimes \rho \underline{u}) = \underline{\mathrm{div}}\ \left(\underline{\underline{\sigma}}\right) + \rho \underline{g} + \underline{ST}_{\underline{u}} - \rho \underline{\underline{K}}\, \underline{u} + \Gamma \underline{u}^{in}, \tag{I.2.7}$$

where $\underline{ST}_{\underline{u}}$ and $\rho \underline{\underline{K}}\, \underline{u}$ stand for an additional momentum Source Terms (head loss, treated as implicit by default) which may be prescribed by the user (head loss, $\Gamma \underline{u}^{in}$ contribution associated with a user-prescribed mass source term...). Note that $\underline{\underline{K}}$ is a symmetric positive tensor, by definition , so that its contribution to the kinetic energy balance is negative.

In order to make the set of Equations (I.2.4) and (I.2.7) closed, the Newtonian state law linking the deviatoric part of the stress tensor $\underline{\underline{\sigma}}$ to the velocity field (more precisely to the rate of strain tensor $\underline{\underline{S}}$) is introduced:

$$\underline{\underline{\tau}} = 2\mu \underline{\underline{S}}^{D} = 2\mu \underline{\underline{S}} - \frac{2}{3}\mu tr\left(\underline{\underline{S}}\right) \underline{\underline{1}}, \tag{I.2.8}$$

where $\mu = \mu_l$ is called the dynamic molecular viscosity, whereas $\underline{\underline{\tau}}$ is the viscous stress tensor and the pressure field are defined as:

$$\left\{ \begin{array}{rcl} P & = & -\dfrac{1}{3} tr\left(\underline{\underline{\sigma}}\right), \\ \underline{\underline{\sigma}} & = & \underline{\underline{\tau}} - P \underline{\underline{1}}. \end{array} \right. \tag{I.2.9}$$

and $\underline{\underline{S}}$, the strain rate tensor, as:

$$\underline{\underline{S}} = \frac{1}{2}\left(\underline{\underline{\nabla}}\, \underline{u} + \underline{\underline{\nabla}}\, \underline{u}^{T}\right). \tag{I.2.10}$$

Note that a fluid for which (I.2.8) holds, is called a Newtonian fluid, it is generally the case for water or air, but not the case for a paint because the stresses do not depend linearly on the strain rate.

Navier-Stokes equations: Injecting Equation (I.2.8) into the momentum Equation (I.2.7) and combining it with the continuity Equation (I.2.4) give the Navier-Stokes equations:

$$\left\{ \begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \mathrm{div}\ (\rho \underline{u}) & = & \Gamma, \\ \dfrac{\partial}{\partial t}(\rho \underline{u}) + \underline{\mathrm{div}}\ (\underline{u} \otimes \rho \underline{u}) & = & -\underline{\nabla} P + \underline{\mathrm{div}}\left(\mu\left[\underline{\underline{\nabla}}\, \underline{u} + \underline{\underline{\nabla}}\, \underline{u}^{T} - \dfrac{2}{3} tr\left(\underline{\underline{\nabla}}\, \underline{u}\right) \underline{\underline{Id}}\right]\right) + \rho \underline{g} + \underline{ST}_{\underline{u}} - \rho \underline{\underline{K}}\, \underline{u} + \Gamma \underline{u}^{in}, \end{array} \right. \tag{I.2.11}$$

The left hand side of the momentum part of Equation (I.2.11) can be rewritten using the continuity Equation (I.2.4):

$$\frac{\partial}{\partial t}(\rho \underline{u}) + \underline{\mathrm{div}}\ (\underline{u} \otimes \rho \underline{u}) = \rho \frac{\partial \underline{u}}{\partial t} + \underbrace{\frac{\partial \rho}{\partial t} \underline{u}}_{[\Gamma - \mathrm{div}\,(\rho \underline{u})]\underline{u}} + \mathrm{div}\ (\rho \underline{u})\, \underline{u} + \underbrace{\underline{\underline{\nabla}}\, \underline{u} \cdot (\rho \underline{u})}_{\text{convection}}. \tag{I.2.12}$$

Then the Navier-Stokes equations read in non-conservative form:

$$\left\{ \begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \mathrm{div}\ (\rho \underline{u}) & = & \Gamma, \\ \rho \dfrac{\partial \underline{u}}{\partial t} + \underline{\underline{\nabla}}\, \underline{u} \cdot (\rho \underline{u}) & = & -\underline{\nabla} P + \underline{\mathrm{div}}\left(\mu\left[\underline{\underline{\nabla}}\, \underline{u} + \underline{\underline{\nabla}}\, \underline{u}^{T} - \dfrac{2}{3} tr\left(\underline{\underline{\nabla}}\, \underline{u}\right) \underline{\underline{Id}}\right]\right) + \rho \underline{g} + \underline{ST}_{\underline{u}} - \rho \underline{\underline{K}}\, \underline{u} + \Gamma\left(\underline{u}^{in} - \underline{u}\right), \end{array} \right. \tag{I.2.13}$$

This formulation will be used in the following. Note that the convective term is nothing else but $\underline{\underline{\nabla}}\, \underline{u} \cdot (\rho \underline{u}) = \underline{\mathrm{div}}\ (\underline{u} \otimes \rho \underline{u}) - \mathrm{div}\ (\rho \underline{u})\, \underline{u}$, this relationship should be conserved by the space-discretized scheme (see Chapter 4).

[3] $\underline{\underline{\sigma}} \cdot \mathrm{d}\underline{S}$ represents the forces exerted on the surface element $\mathrm{d}\underline{S}$ by the exterior of the domain Ω.

2.1.2 Turbulent flows with a Reynolds-Averaged Navier-Stokes approach ($RANS$):

When the flow becomes turbulent, the $RANS$ approach is to consider the velocity field $\underline{u}$ as stochastic and then splat into a mean field denoted by $\overline{\underline{u}}$ and a fluctuating field $\underline{u}'$:

$$\underline{u} = \overline{\underline{u}} + \underline{u}'. \tag{I.2.14}$$

The Reynolds average operator $\overline{(\cdot)}$ is applied to Navier-Stokes Equation (I.2.11):

$$\left\{\begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \mathrm{div}\left(\rho\overline{\underline{u}}\right) & = & \Gamma, \\ \rho\dfrac{\partial \overline{\underline{u}}}{\partial t} + \underline{\underline{\nabla}}\,\overline{\underline{u}}\cdot\left(\rho\overline{\underline{u}}\right) & = & -\underline{\nabla}\overline{P} + \underline{\mathrm{div}}\left(\mu\left[\underline{\underline{\nabla}}\,\overline{\underline{u}} + \underline{\underline{\nabla}}\,\overline{\underline{u}}^{T} - \dfrac{2}{3}tr\left(\underline{\underline{\nabla}}\,\overline{\underline{u}}\right)\underline{\underline{Id}}\right]\right) + \rho\underline{g} - \underline{\mathrm{div}}\left(\rho\underline{\underline{R}}\right) \\ & + & \underline{ST}_{\underline{u}} - \rho\underline{\underline{K}}\,\overline{\underline{u}} + \Gamma\left(\overline{\underline{u}}^{in} - \overline{\underline{u}}\right), \end{array}\right. \tag{I.2.15}$$

Only the mean fields $\overline{\underline{u}}$ and $\overline{P}$ are computed. An additional term $\underline{\underline{R}}$ appears in the Reynolds Equations (I.2.15) which is by definition the covariance tensor of the fluctuating velocity field and called the Reynolds stress tensor:

$$\underline{\underline{R}} \equiv \overline{\underline{u}' \otimes \underline{u}'}. \tag{I.2.16}$$

the latter requires a closure modelling which depends on the turbulence model adopted. Two major types of modelling exist:

i/ Eddy Viscosity Models (EVM) which assume that the Reynolds stress tensor is aligned with the strain rate tensor of the mean flow ($\overline{\underline{\underline{S}}} \equiv \frac{1}{2}\left(\underline{\underline{\nabla}}\,\underline{u} + \underline{\underline{\nabla}}\,\underline{u}^{T}\right)$):

$$\rho\underline{\underline{R}} = \frac{2}{3}\rho k\underline{\underline{1}} - 2\mu_T\overline{\underline{\underline{S}}}^{D}, \tag{I.2.17}$$

where the turbulent kinetic energy k is defined by:

$$k \equiv \frac{1}{2}tr\left(\underline{\underline{R}}\right), \tag{I.2.18}$$

and μ_T is called the dynamic turbulent viscosity and must be modelled. Note that the viscous part $\mu_T\overline{\underline{\underline{S}}}^{D}$ of the Reynolds stresses is simply added to the viscous part of the stress tensor $\mu_l\overline{\underline{\underline{S}}}^{D}$ so that the momentum equation for the mean velocity is similar to the one of a laminar flow with a variable viscosity $\mu = \mu_l + \mu_T$. Five EVM are available in code_saturne: $k-\varepsilon$, $k-\varepsilon$ with Linear Production (LP), $k-\omega$ SST, Spalart Allmaras, and an Elliptic Blending model (EB-EVM) $Bl-v^2-k$ ([Billard and Laurence, 2012]).

ii/ Differential Reynolds Stress Models ($DRSM$) which solve a transport equation on the components of the Reynolds stress tensor $\underline{\underline{R}}$ during the simulation, and are readily available for the momentum equation (I.2.15). Three $DRSM$ models are available in code_saturne: $R_{ij}-\varepsilon$ proposed by Launder Reece and Rodi (LRR) in [Launder et al., 1975], $R_{ij}-\varepsilon$ proposed by Speziale, Sarkar and Gatski (SSG) in [Speziale et al., 1991] and an Elliptic Blending version EB-RSM (see [Manceau and Hanjalić, 2002]).

2.1.3 Large Eddy Simulation (LES):

The LES approach consists in spatially filtering the $\underline{u}$ field using an operator denoted by $\widetilde{(\cdot)}$. Applying the latter filter to the Navier-Stokes Equations (I.2.22) gives:

$$\left\{\begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \mathrm{div}\left(\rho\widetilde{\underline{u}}\right) & = & \Gamma, \\ \rho\dfrac{\partial \widetilde{\underline{u}}}{\partial t} + \underline{\underline{\nabla}}\,\widetilde{\underline{u}}\cdot\left(\rho\widetilde{\underline{u}}\right) & = & -\underline{\nabla}\widetilde{P} + \underline{\mathrm{div}}\left(2\mu\widetilde{\underline{\underline{S}}}^{D}\right) + \rho\underline{g} - \underline{\mathrm{div}}\left(\rho\widetilde{\underline{u}'\otimes\underline{u}'}\right) + \underline{ST}_{\underline{u}} - \rho\underline{\underline{K}}\,\widetilde{\underline{u}} + \Gamma\left(\widetilde{\underline{u}}^{in} - \widetilde{\underline{u}}\right), \end{array}\right. \tag{I.2.19}$$

where $\underline{u}'$ are non-filtered fluctuations. An eddy viscosity hypothesis is made on the additional resulting tensor:

$$\rho\widetilde{\underline{u}'\otimes\underline{u}'} = \frac{2}{3}\rho k\underline{\underline{Id}} - 2\mu_T\underline{\underline{\widetilde{S}}}^D, \tag{I.2.20}$$

where the above turbulent viscosity μ_T now accounts only for sub-grid effects.

2.1.4 Formulation for laminar, RANS or LES calculation:

For the sake of simplicity, in all cases, the computed velocity field will be denoted by $\underline{u}$ even if it is about *RANS* velocity field $\overline{\underline{u}}$ or *LES* velocity field $\widetilde{\underline{u}}$.

Moreover, a manipulation on the right hand side of the momentum is performed to change the meaning of the pressure field. let $P^\star$ be the dynamic pressure field defined by:

$$P^\star = P - \rho_0\underline{g}\cdot\underline{x} + \frac{2}{3}\rho k, \tag{I.2.21}$$

where ρ_0 is a reference constant density field. Then the continuity and the momentum equations read:

$$\left\{\begin{array}{rcl} \dfrac{\partial\rho}{\partial t} + \mathrm{div}\left(\rho\underline{u}\right) &=& \Gamma, \\ \rho\dfrac{\partial\underline{u}}{\partial t} + \underline{\underline{\nabla}}\,\underline{u}\cdot\left(\rho\underline{u}\right) &=& -\underline{\nabla}P^\star + \underline{\mathrm{div}}\left(2\left(\mu_l+\mu_T\right)\underline{\underline{S}}^D\right) - \underline{\mathrm{div}}\left(\rho\underline{\underline{R}}^D + 2\mu_T\underline{\underline{S}}^D\right) + \left(\rho-\rho_0\right)\underline{g} \\ &+& \underline{ST}_{\underline{u}} - \rho\underline{\underline{K}}\,\underline{u} + \Gamma\left(\underline{u}^{in}-\underline{u}\right). \end{array}\right. \tag{I.2.22}$$

2.2 Thermal equations

2.2.1 Energy equation

The energy equation reads:

$$\rho\frac{\mathrm{d}e}{\mathrm{d}t} = -\mathrm{div}\left(\underline{q''}\right) + q''' - P\mathrm{div}\left(\underline{u}\right) + \mu S^2, \tag{I.2.23}$$

where e is the specific internal energy, $\underline{q''}$ is the heat flux vector, q''' is the dissipation rate or rate of internal heat generation and S^2 is scalar strain rate defined by

$$S^2 = 2\underline{\underline{S}}^D : \underline{\underline{S}}^D. \tag{I.2.24}$$

The Fourier law of heat conduction gives:

$$\underline{q''} = -\lambda\underline{\nabla}T, \tag{I.2.25}$$

where λ is the thermal conductivity and T is the temperature field.

2.2.2 Enthalpy equation

Thermodynamics definition of enthalpy gives:

$$h \equiv e + \frac{1}{\rho}P. \tag{I.2.26}$$

Applying the Lagrangian derivative $\dfrac{\mathrm{d}}{\mathrm{d}t}$ to h:

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{\mathrm{d}e}{\mathrm{d}t} + \frac{1}{\rho}\frac{\mathrm{d}P}{\mathrm{d}t} - \frac{P}{\rho^2}\frac{\mathrm{d}\rho}{\mathrm{d}t}, \tag{I.2.27}$$

then

$$\begin{aligned}\rho\frac{\mathrm{d}h}{\mathrm{d}t} &= \mathrm{div}\,(\lambda\underline{\nabla}T) + q''' - P\mathrm{div}\,\underline{u} + \mu S^2 + \frac{\mathrm{d}P}{\mathrm{d}t} - \tfrac{P}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}t},\\ &= \mathrm{div}\,(\lambda\underline{\nabla}T) + q''' + \mu S^2 + \frac{\mathrm{d}P}{\mathrm{d}t} - \tfrac{P}{\rho}\left(\underbrace{\frac{\mathrm{d}\rho}{\mathrm{d}t} + \rho\mathrm{div}\,\underline{u}}_{=0}\right),\\ &= \mathrm{div}\,(\lambda\underline{\nabla}T) + q''' + \mu S^2 + \frac{\mathrm{d}P}{\mathrm{d}t}.\end{aligned} \tag{I.2.28}$$

To express (I.2.28) only in terms of h and not T, some thermodynamics relationships can be used. For a pure substance, Maxwell's relations give:

$$\mathrm{d}h = C_p\mathrm{d}T + \frac{1}{\rho}\left(1-\beta T\right)\mathrm{d}P, \tag{I.2.29}$$

where β is the thermal expansion coefficient defined by:

$$\beta = -\frac{1}{\rho}\left.\frac{\partial\rho}{\partial T}\right|_P. \tag{I.2.30}$$

The equation (I.2.28) then becomes:

$$\rho\frac{\mathrm{d}h}{\mathrm{d}t} = \mathrm{div}\left(\frac{\lambda}{C_p}\left(\underline{\nabla}h - \frac{1-\beta T}{\rho}\underline{\nabla}P\right)\right) + q''' + \mu S^2 + \frac{\mathrm{d}P}{\mathrm{d}t}. \tag{I.2.31}$$

Remark 2.1 *Note that for incompressible flows, βT is negligible compared to 1. Moreover, for ideal gas, $\beta = 1/T$ so the following relationship holds:*

$$\mathrm{d}h = C_p\mathrm{d}T. \tag{I.2.32}$$

2.2.3 Temperature equation

In order to rearrange the enthalpy Equation (I.2.28) in terms of temperature (I.2.29) is used:

$$\left.\frac{\partial s}{\partial P}\right|_T = -\left.\frac{\partial(1/\rho)}{\partial T}\right|_P = \frac{1}{\rho^2}\left.\frac{\partial\rho}{\partial T}\right|_P = -\frac{\beta}{\rho}, \tag{I.2.33}$$

and also:

$$\frac{\lambda}{C_p}\left(\underline{\nabla}h - \frac{1-\beta T}{\rho}\underline{\nabla}P\right) = \lambda\underline{\nabla}T, \tag{I.2.34}$$

and Equation (I.2.31) becomes:

$$\rho C_p\frac{\mathrm{d}T}{\mathrm{d}t} = \mathrm{div}\,(\lambda\underline{\nabla}T) + \beta T\frac{\mathrm{d}P}{\mathrm{d}t} + q''' + \mu S^2. \tag{I.2.35}$$

The Eq. (I.2.35) can be reduced using some hypothesis, for example:

- If the fluid is an ideal gas, $\beta = \dfrac{1}{T}$ and it becomes:

$$\rho C_p\frac{\mathrm{d}T}{\mathrm{d}t} = \mathrm{div}\,(\lambda\underline{\nabla}T) + \frac{\mathrm{d}P}{\mathrm{d}t} + q''' + \mu S^2. \tag{I.2.36}$$

- If the fluid is incompressible, $\beta = 0$, $q''' = 0$ and we generally neglect μS^2 so that it becomes:

$$\rho C_p\frac{\mathrm{d}T}{\mathrm{d}t} = \mathrm{div}\,(\lambda\underline{\nabla}T)\,. \tag{I.2.37}$$

2.3 Equations for scalars

Two types of transport equations are considered:

i/ convection of a scalar with additional source terms:

$$\frac{\partial(\rho a)}{\partial t} + \underbrace{\mathrm{div}\,(a\,\rho\underline{u})}_{\text{advection}} - \underbrace{\mathrm{div}\,(K\underline{\nabla} a)}_{\text{diffusion}} = ST_a + \Gamma\, a^{in}, \tag{I.2.38}$$

ii/ convection of the variance $\widetilde{a"^2}$ with additional source terms:

$$\begin{aligned}\frac{\partial\left(\rho\widetilde{a"^2}\right)}{\partial t} + \underbrace{\mathrm{div}\left(\widetilde{a"^2}\,\rho\,\underline{u}\right)}_{\text{advection}} - \underbrace{\mathrm{div}\left(K\,\underline{\nabla}\widetilde{a"^2}\right)}_{\text{diffusion}} &= ST_{\widetilde{a"^2}} + \Gamma\,\widetilde{a"^2}^{in} \\ &\quad + \underbrace{2\,\frac{\mu_t}{\sigma_t}\,(\underline{\nabla}\widetilde{a})^2 - \frac{\rho\,\varepsilon}{R_f k}\,\widetilde{a"^2}}_{\text{production and dissipation}},\end{aligned} \tag{I.2.39}$$

The two previous equations can be unified formally as:

$$\frac{\partial(\rho Y)}{\partial t} + \mathrm{div}\,(\rho\underline{u}Y) - \mathrm{div}\,(K\underline{\nabla}Y) = ST_Y + \Gamma\, Y^{in} + \mathcal{P}_Y - \epsilon_Y \tag{I.2.40}$$

with:

$$\mathcal{P}_Y - \epsilon_Y = \begin{cases} 0 & \text{for } Y = a\ , \\ 2\frac{\mu_t}{\sigma_t}(\underline{\nabla}\widetilde{a})^2 - \frac{\rho\,\varepsilon}{R_f k}\,\widetilde{a"^2} & \text{for } Y = \widetilde{a"^2}. \end{cases} \tag{I.2.41}$$

ST_Y represents the additional source terms that may be prescribed by the user.

2.3.1 Equations for scalars with a drift

The Diffusion-Inertia model is available in code_saturne; it aims at modelling aerosol transport, and was originally proposed by Zaichik *et al.* [Zaichik et al., 2004]. Details on the theoretical work and implementation of the model in the framework of code_saturne can be found in technical note H-I81-2013-02277-EN.

Aerosol transport numerical model

The so-called diffusion-inertia model has been first proposed by Zaichik *et al.* [Zaichik et al., 2004]. It is based on the principle that the main characteristics of the aerosol transport in turbulent flows can be described by solving a single transport equation on the particle mass concentration, which reads (using the notations of P. Nérisson [Nerisson, 2009]):

$$\begin{aligned}\frac{\partial C}{\partial t} + \frac{\partial}{\partial x_i}\left(\left[U_{f,i} + \tau_p g_i - \tau_p\left(\frac{\partial U_{f,i}}{\partial t} + U_{f,k}\frac{\partial U_{f,i}}{\partial x_k}\right) - \frac{\partial}{\partial x_k}\left(D_b\delta_{ik} + \frac{\Omega}{1+\Omega}D^T_{ik}\right)\right] C\right) = \\ \frac{\partial}{\partial x_i}\left(\left(D_b\delta_{ik} + D^T_{p,ik}\right)\frac{\partial C}{\partial x_k}\right)\end{aligned} \tag{I.2.42}$$

In this equation:

- C represents the particle mass concentration;

- $U_{f,i}$ is the i component of the fluid velocity;
- τ_p is the particle relaxation time;
- g_i is the i component of the gravity acceleration;
- D_b and D_{ij}^T are respectively the coefficient of Brownian diffusion and the tensor of turbulent diffusivity.

A physical interpretation of the different terms involved in the transport equation of the aerosols follows:

- $\tau_p g_i$ represents the transport due to gravity;
- $-\tau_p \left(\dfrac{\partial U_{f,i}}{\partial t} + U_{f,k} \dfrac{\partial U_{f,i}}{\partial x_k} \right)$ represents the deviation of the aerosol trajectory with respect to the fluid (zero-inertia) particle due to particle inertia (which may be loosely referred to as "centrifugal" effect);
- $\dfrac{\partial}{\partial x_k} D_b \delta_{ik}$ is the transport of particles due to the gradient of temperature (the so-called thermophoresis);
- $\dfrac{\partial}{\partial x_i} \left(\dfrac{\Omega}{\Omega+1} D_{ik}^T \right)$ is the transport of particles due to the gradient of kinetic energy (the so-called "turbophoresis", or "turbophoretic effect").

If the particulate Reynolds number is sufficiently small, the particle relaxation time τ_p can be defined as

$$\tau_p = \frac{\rho_p d_p^2}{18 \mu_f} \tag{I.2.43}$$

with ρ_p the particle density, μ_f the fluid dynamic viscosity and d_p the particle diameter. It should be underlined that to take advantage of the classical transport equation of the species in code_saturne, Eq. (I.2.42) is reformulated by considering the variable $Y \equiv C/\rho_f$ (with ρ_f considered as a good enough approximation of the density of the particle-laden flow) and actually solving an equation on this variable. With a vectorial notation, this equation reads[4]:

$$\rho \frac{\partial Y}{\partial t} + \mathrm{div}\left([\rho \underline{u}_Y] \, Y \right) - \mathrm{div}\left(\rho \underline{u}_Y \right) Y + \mathrm{div}\left(\rho \underline{u}_Y - \rho \underline{u} \right) Y = \mathrm{div}\left(\left[\rho D_b \underline{\underline{1}} + \rho \underline{\underline{D}}_p^T \right] \underline{\nabla} Y \right) \tag{I.2.44}$$

where the additional convective flux is:

$$\left(\rho \underline{u}_Y - \rho \underline{u} \right) = \tau_p \rho \underline{g} - \tau_p \rho \frac{\mathrm{d}\underline{u}}{\mathrm{d}t} - \underline{\mathrm{div}} \left(\rho D_b \underline{\underline{1}} + \rho \frac{\Omega}{1+\Omega} \underline{\underline{D}}_p^t \right) \tag{I.2.45}$$

Brownian diffusion

Let us detail Eq. (I.2.42) in case all terms are canceled except the diffusion one:

$$\rho \frac{\partial Y}{\partial t} = \mathrm{div}\left(\left[\rho D_b \underline{\underline{1}} + \rho \underline{\underline{D}}_p^t \right] \underline{\nabla} Y \right) \tag{I.2.46}$$

The coefficient D_b is theoretically given by the Stokes-Einstein relation:

$$D_b = \frac{k_B T}{6 \pi \mu_f \frac{d_p}{2}} \tag{I.2.47}$$

with k_B the Boltzmann constant equal to 1.38×10^{-23} J.K^{-1}.

[4] This equation is not exact. In the second member, density has been extracted from the gradient operator, and in the additional convective flux density was integrated inside the divergent operator, for compatibility reason with code_saturne construction. These approximations do not have major impact.

Sedimentation terms

Let us now focus on the term simulating transport by the gravity acceleration, in case all terms that model particle transport and diffusion are set to zero except gravity and the fluid velocity, the scalar speed $\underline{u}_Y$ reduces to:

$$\rho \underline{u}_Y = \rho \underline{u} + \tau_p \rho \underline{g} \tag{I.2.48}$$

Turbophoretic transport

Cancelling all but turbophoresis transport terms (no gravity, no turbulent diffusion, etc.), the scalar associated velocity $\underline{u}_Y$ becomes:

$$\rho \underline{u}_Y = \rho \underline{u} - \underline{\mathrm{div}} \left(\rho \frac{\Omega}{1+\Omega} \underline{\underline{D}}_p^T \right) \tag{I.2.49}$$

Turbophoresis should move the particles from the zones with higher turbulent kinetic energy to the lower one. The fluid turbulent diffusion tensor can be expressed as:

$$\underline{\underline{D}}_p^T = \tau_T \langle \underline{u}' \otimes \underline{u}' \rangle \tag{I.2.50}$$

With a Eddy Viscosity turbulence Model (EVM), one has

$$\langle \underline{u}' \otimes \underline{u}' \rangle = \frac{2}{3} k \underline{\underline{1}} - \nu_T \underline{\underline{S}} \tag{I.2.51}$$

Also, for the $k-\varepsilon$ model:

$$\tau_T = \frac{3}{2} \frac{C_\mu}{\sigma_T} \frac{k}{\varepsilon} \tag{I.2.52}$$

where $C_\mu = 0.09$ is a constant and σ_T the turbulent Schmidt, and Ω is defined by:

$$\Omega = \frac{\tau_p}{\tau_{f\ p}^T} \tag{I.2.53}$$

with $\tau_{f\ p}^T = \tau_T = \dfrac{3}{2} \dfrac{C_\mu}{\sigma} \dfrac{k}{\varepsilon}$.

Chapter 3

Time stepping

3.1 Time discretisation of a transport equation

The time-stepping is described in [Ferrand, 2022] and [Amino et al., 2022].

At first, the physical properties of the flow are computed (density, viscosity, specific heat *etc.*): indeed, they may depend upon the variables (such as the temperature for example).

The time scheme is a θ-scheme:

$$\begin{cases} \theta = 1 & \text{for an implicit first order Euler scheme,} \\ \theta = \frac{1}{2} & \text{for second order Crank-Nicolson scheme.} \end{cases} \tag{I.3.1}$$

For the second order scheme, the time step is assumed to be constant.

If required, the equations for the turbulent variables are solved (turbulent kinetic energy and dissipation or Reynolds stresses and dissipation), using a θ-scheme again. For the $k-\varepsilon$ model, an additional step is carried out to couple the source terms. For the Reynolds stress model, the variables (turbulent stresses and dissipation) are solved sequentially, without coupling.

Next, the equations for the *scalars* (enthalpy, temperature, tracers, concentrations, mass fractions...) are solved, also with a θ-scheme.

Finally, all the variables are updated and another time step may start.

The general equation for advection (valid for the velocity components, the turbulent variables and the scalars) is re-written as follows in a condensed form; the mass equation ($\frac{\partial \rho}{\partial t} + \text{div}\,(\rho \underline{u}) = \Gamma$ see Equation (I.2.4)) has been used to split the time derivative:

$$\rho \frac{\partial Y}{\partial t} + \underline{\nabla} Y \cdot (\rho \underline{u}) - \text{div}\,(K \underline{\nabla} Y) = S_i\,(\Phi,\, \varphi)\, Y + S_e\,(\Phi, \varphi) + \Gamma \left(Y^{in} - Y\right). \tag{I.3.2}$$

In Equation (I.3.2), Φ represents the physical properties such as $(\rho, K, \mu_t, ...)$, φ represents the variables of the problem such as $(\underline{u}, k, \epsilon, ...)$, $S_i\,(\Phi,\, \varphi)\, Y$ is the linear part of the source terms and $S_e\,(\Phi,\, \varphi)$ includes all other source terms.

Therefore, four different time steppings are used, they all define the time at which the quantities are evaluated:

i/ θ is the time stepping applied to the variable $Y^{n+\theta} \equiv \theta Y^{n+1} + (1-\theta)\, Y^n$,

ii/ θ_Φ is the time stepping applied to the physical properties,

iii/ θ_F is the time stepping applied to the mass flux,

iv/ θ_S is the time stepping applied to the source terms,

If $\theta = 1/2$, or if an extrapolation is used, the time step Δt is constant in time and uniform in space.

3.1.1 Physical properties Φ

The physical properties of the flow (density, viscosity, specific heat...) are:

- either explicit, defined at the time step n.
- or extrapolated at $n + \theta_\Phi$ using the Adam-Bashforth time scheme (in this case, the time step is assumed to be constant).

Under a more general form, this reads:

$$\Phi^{n+\theta_\Phi} \equiv (1+\theta_\Phi)\,\Phi^n - \theta_\Phi\,\Phi^{n-1}, \tag{I.3.3}$$

$$\begin{cases} \theta_\Phi = 0 & \text{standard explicit formulation,} \\ \theta_\Phi = 1/2 & \text{second order extrapolation at } n+1/2, \\ \theta_\Phi = 1 & \text{first order extrapolation at } n+1. \end{cases} \tag{I.3.4}$$

3.1.2 Mass flux

For the mass flux, three time schemes are available. The mass flux may be:

- explicit, taken at time step n for the momentum equations and updated with its value at time step $n+1$ for the equations for turbulence and scalars (standard scheme).
- explicit, taken at time step n for the momentum equations and also for the equations for turbulence and scalars.
- taken at $n+\theta_F$ (second order if $\theta_F = 1/2$). To solve the momentum equations, $(\rho\,\underline{u})^{n-2+\theta_F}$ and $(\rho\,\underline{u})^{n-1+\theta_F}$ are known. Hence, the value at $n+\theta_F$ is obtained as a result of the following extrapolation:

  $$(\rho\underline{u})^{n+\theta_F} = 2\,(\rho\underline{u})^{n-1+\theta_F} - (\rho\underline{u})^{n-2+\theta_F}\,. \tag{I.3.5}$$

  At the end of this phase (after the pressure correction step), $(\rho\,\underline{u})^{n+1}$ is known and the following interpolation is used to determine the mass flux at $n+\theta_F$ that will be adopted for the equations for turbulence and scalars:

  $$(\rho\,\underline{u})^{n+\theta_F} = \frac{1}{2-\theta_F}\,(\rho\underline{u})^{n+1} + \frac{1-\theta_F}{2-\theta_F}\,(\rho\underline{u})^{n-1+\theta_F}\,. \tag{I.3.6}$$

See the programmers reference of the dedicated subroutine for further details on the computation of the mass flux.

3.1.3 Source terms

As for the physical properties, the **explicit** source terms are:

- explicit:

  $$[S_e\,(\Phi,\,\varphi)]^n = S_e\left(\Phi^{n+\theta_\Phi},\,\varphi^n\right), \tag{I.3.7}$$

- extrapolated at $n+\theta_S$ using the Adams-Bashforth scheme:

  $$[S_e\,(\Phi,\,\varphi)]^{n+\theta_S} = (1+\theta_S)\;S_e\,(\Phi^n,\,\varphi^n) - \theta_S\,S_e\left(\Phi^{n-1},\,\varphi^{n-1}\right). \tag{I.3.8}$$

By default, to be consistent and preserve the order of convergence in time, the implicit source terms are discretized with the same scheme as that is used for convection-diffusion of the unknown considered, *i.e.* taken at $n+\theta$:

$$[S_i\,(\Phi,\,\varphi)\,Y]^{n+\theta} = S_i(\Phi^{n+\theta_\Phi},\,\varphi^n)\;\left[\theta Y^{n+1} + (1-\theta)\,Y^n\right]. \tag{I.3.9}$$

Remark 3.1 *The **implicit** source terms taken also at $n+\theta$ for $\theta_S \neq 0$, while for $\theta_S = 0$, the implicit source terms are taken at $n+1$, this to enhance stability.*

3.1.4 General time discretized form

For the sake of clarity, it is assumed hereafter that, unless otherwise explicitly stated, the mass flux is taken at $n+\theta_F$ and the physical properties are taken at $n+\theta_\Phi$, with θ_F and θ_Φ dependent upon the specific schemes selected for the mass flux and the physical properties respectively and all θs are denoted by θ.

Under a general form, the discrete counterpart of Equation (I.3.2) at $n+\theta$ reads:

$$\frac{\rho}{\Delta t}\left(Y^{n+1}-Y^n\right)+\underline{\underline{\nabla}}Y^{n+\theta}\cdot(\rho\underline{u})-\mathrm{div}\left(K\underline{\underline{\nabla}}Y^{n+\theta}\right)=\left[S_i(\Phi,\,\varphi)Y\right]^{n+\theta}+\left[S_e(\Phi,\,\varphi)\right]^{n+\theta}. \tag{I.3.10}$$

Using the standard θ-scheme $Y^{n+\theta}=\theta Y^{n+1}+(1-\theta)\,Y^n$, the equation reads:

$$\begin{aligned}\frac{\rho}{\Delta t}\left(Y^{n+1}-Y^n\right)+\theta\left[\underline{\underline{\nabla}}Y^{n+1}\cdot(\rho\underline{u})-\mathrm{div}\left(K\underline{\underline{\nabla}}Y^{n+1}\right)\right]&=-(1-\theta)\left[\underline{\underline{\nabla}}Y^n\cdot(\rho\underline{u})-\mathrm{div}\left(K\underline{\underline{\nabla}}Y^n\right)\right]\\&\quad+S_i\left(\Phi,\,\varphi^n\right)\left[\theta Y^{n+1}+(1-\theta)\,Y^n\right]+\left[S_e(\Phi,\,\varphi)\right]^{n+\theta}.\end{aligned} \tag{I.3.11}$$

For numerical reasons, the system is solved in an iterative and incremental manner, with the help of the series $\delta Y^{n+1}_{k+1}=Y^{n+1}_{k+1}-Y^{n+1}_k$ (with, by definition, $Y^{n+1}_0=Y^n$). More theoretical details of such an iterative process are given in §6.1.

3.2 Pressure-based velocity-pressure solver

The aim of this section is to describe how Navier Stokes equations are solved for an incompressible or weakly compressible (dilatable or Low Mach algorithm) combined with an implicit Euler time stepping or a second order Crank Nicolson time stepping.

The set of equations to be solved is

$$\begin{cases}\rho\dfrac{\partial\underline{u}}{\partial t}+\underline{\underline{\nabla}}\,\underline{u}\cdot(\rho\underline{u})=\underline{\mathrm{div}}\,(\underline{\underline{\sigma}})-\underline{\mathrm{div}}\,(\rho\underline{\underline{R}})+\underline{ST}_u-\underline{\underline{K}}\,\underline{u}+\rho\underline{g}+\Gamma\left(\underline{u}^{in}-\underline{u}\right),\\ \dfrac{\partial\rho}{\partial t}+\mathrm{div}\,(\rho\underline{u})=\Gamma,\end{cases} \tag{I.3.12}$$

where ρ is the density field, $\underline{u}$ is the velocity field to be solved, $[\,\underline{ST}_u-\underline{\underline{K}}\,\underline{u}+\rho\underline{g}\,]$ are source terms (note that $\underline{\underline{K}}$ is expected to be a positive definite tensor), $\underline{\underline{\sigma}}$ is the stress tensor, composed of the viscous stress tensor $\underline{\underline{\tau}}$ and of the pressure field as follows

$$\begin{cases}\underline{\underline{\sigma}} &=& \underline{\underline{\tau}}-P\underline{\underline{Id}},\\ \underline{\underline{\tau}} &=& 2\mu\underline{\underline{S}}+\left(\kappa-\dfrac{2}{3}\mu\right)tr\left(\underline{\underline{S}}\right)\underline{\underline{1}},\\ \underline{\underline{S}} &=& \dfrac{1}{2}\left(\underline{\underline{\nabla}}\,\underline{u}+\underline{\underline{\nabla}}\,\underline{u}^T\right).\end{cases} \tag{I.3.13}$$

where μ is the dynamic molecular viscosity, κ the volume viscosity (also called the second viscosity, usually neglected in the code, excepted for compressible flows). $\underline{\underline{S}}$ is called the strain rate tensor and Γ is a possible mass source term.

3.2.1 Segregated solver: SIMPLEC

A fractional step scheme is used to solve the mass and momentum equations (see Chorin [Chorin, 1968]).

The first step (predictor step) provides predicted velocity components: they are determined in a coupled way solving a $3N_{cell}\times 3N_{cell}$ system. The mass equation is taken into account during the

second step (corrector step): a pressure Poisson equation is solved and the mass fluxes at the cell faces are updated.

See the programmers reference of the dedicated subroutine for further details on the SIMPLEC solver.

Prediction step

In this section, a predicted velocity field $\widetilde{\underline{u}}$ is obtained by solving the momentum equation of (I.3.12)

$$\begin{aligned}\rho\frac{\widetilde{\underline{u}}-\underline{u}^n}{\Delta t}+\underbrace{\underline{\underline{\nabla}}\,\widetilde{\underline{u}}^{n+\theta}\cdot(\rho\underline{u})}_{\text{convection}} &= \underbrace{\underline{\text{div}}\left[(\mu+\mu_t)\left(\underline{\underline{\nabla}}\,\widetilde{\underline{u}}^{n+\theta}+\left(\underline{\underline{\nabla}}\,\widetilde{\underline{u}}^{n+\theta}\right)^T-\frac{2}{3}tr\left(\underline{\underline{\nabla}}\,\widetilde{\underline{u}}^{n+\theta}\right)\right)\right]}_{\text{viscous term}} \\ &- \underline{\nabla}\left[(P^*)^n\right]-\underline{\text{div}}\left(\rho\underline{\underline{R}}^D+2\mu_T\underline{\underline{S}}^D\right)+(\rho-\rho_0)\,\underline{g} \\ &+ \underbrace{\Gamma\left(\underline{u}^{in}-\widetilde{\underline{u}}\right)}_{\text{Mass source term}}-\underbrace{\rho\underline{\underline{K}}\,\widetilde{\underline{u}}}_{\text{Head loss}}+\underbrace{\underline{\underline{ST}}_u^{exp}+\underline{\underline{ST}}_u^{imp}\,\widetilde{\underline{u}}}_{\text{user source terms}}\end{aligned} \tag{I.3.14}$$

For more details, see § L and M. The programmers reference of the dedicated subroutine can also be helpful.

Correction step

The predicted velocity has *a priori* non-zero divergence. The second step corrects the pressure by imposing the nullity[1] of the stationary constraint for the velocity computed at time instant t^{n+1}. We then solve:

$$\begin{cases}\dfrac{(\rho\underline{u})^{n+1}-(\rho\widetilde{\underline{u}})^{n+1}}{\Delta t} &= -\underline{\nabla}\delta P^{n+\theta}, \\ \text{div}\,(\rho\underline{u})^{n+1}=\Gamma,\end{cases} \tag{I.3.15}$$

where the pressure increment $\delta P^{n+\theta}$ is defined as:

$$\delta P^{n+\theta}=P^{n+\theta}-P^{n-1+\theta}. \tag{I.3.16}$$

Remark 3.2 *The ρ and μ quantities remain constant over the course of both steps. If there has been any variation in the interval, their values will be modified at the start of the next time step, after the scalars (temperature, mass fraction,...) have been updated.*

For more details, see § L and N. The programmers reference of the dedicated subroutine can also be helpful.

3.2.2 Variable density conservative solvers

Some flows such as those encountered in fire are natively unsteady. In addition, density varies with mixture and temperature. Thus, the temporal variation of density in Navier-Stokes equation has to be considered to get unsteady phenomena such as flame puffings. The SIMPLEC algorithm detailed before rewrites for a variable density field:

- the prediction step solves the momentum equation with an explicit pressure:

$$\frac{\rho^n\widetilde{\underline{u}}-\rho^{n-1}\underline{u}^n}{\Delta t}+\underline{\text{div}}\,[\widetilde{\underline{u}}\otimes(\rho\underline{u})^n]=-\underline{\nabla}p^n+\underline{\text{div}}\,\underline{\underline{\tau}}+\rho^n\underline{g}. \tag{I.3.17}$$

[1] or the density time-variation is the corresponding algorithm is chosen.

- the correction step computes the pressure increment $\delta p^{n+1} = p^{n+1} - p^n$ used to correct the velocity respecting mass balance equation:

$$\left\{ \begin{array}{rcl} \dfrac{\rho^n \underline{u}^{n+1} - \rho^n \underline{\widetilde{u}}}{\Delta t} & = & -\underline{\nabla}\delta p^{n+1} \\ \dfrac{\rho^n - \rho^{n-1}}{\Delta t} + \mathrm{div}\left(\rho \underline{u}\right)^{n+1} & = & \Gamma \end{array} \right. \tag{I.3.18}$$

Combining the two equations of (I.3.18) leads to a Poisson equation on the pressure increment:

$$\mathrm{div}\left(-\Delta t \underline{\nabla}\delta p^{n+1}\right) = \Gamma - \frac{\rho^n - \rho^{n-1}}{\Delta t} - \mathrm{div}\left(\rho \underline{\widetilde{u}}\right) \tag{I.3.19}$$

3.2.3 Variable density semi-analytical solvers for fire modelling

The algorithm detailed in this section is not conservative, and is based on the state law for density. The idea is to use an analytical expression for the total derivative of density, expressed in terms of the solved variables involved in the state law and the total derivatives of these scalars, which are derived from their non conservative balance equations ($\rho \dfrac{\mathrm{d}Y}{\mathrm{d}t} = \mathrm{div}\left(K \underline{\nabla} Y\right) + S_Y$):

$$\frac{1}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}t} = \frac{1}{\rho}\sum_{p=1}^{N} \frac{\partial \rho}{\partial Y_p}\frac{\mathrm{d}Y_p}{\mathrm{d}t} = \frac{1}{\rho^2}\sum_{p=1}^{N} \frac{\partial \rho}{\partial Y_p}\left[\mathrm{div}\left(K \underline{\nabla} Y_p\right) + S_{Y_p}\right]. \tag{I.3.20}$$

These scalars have to be independent to avoid counting several times their effect on the mass accumulation. One has also to bear in mind that the total derivatives are dependent of the state law for density.

Therefore, the correction system (I.3.18) written in a non conservative form reads:

$$\left\{ \begin{array}{l} \dfrac{\left(\rho \underline{u}\right)^{n+1} - \left(\rho \underline{\widetilde{u}}\right)}{\Delta t} = -\underline{\nabla}\delta p^{n+1} \\ \dfrac{\mathrm{d}\rho}{\mathrm{d}t} + \mathrm{div}\left(\rho \underline{u}\right)^{n+1} - \underline{u}^{n+1} \cdot \underline{\nabla}\rho^n = \Gamma, \end{array} \right. \tag{I.3.21}$$

and leads to a new equation for the pressure increment:

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \mathrm{div}\left(\rho \underline{\widetilde{u}}\right) + \mathrm{div}\left(-\Delta t \underline{\nabla}\delta p^{n+1}\right) - \underline{u}^{n+1} \cdot \underline{\nabla}\rho^n = \Gamma. \tag{I.3.22}$$

Using the expression for the corrected velocity:

$$\underline{u}^{n+1} = \underline{\widetilde{u}} - \frac{\Delta t}{\rho^n}\underline{\nabla}\delta p^{n+1}, \tag{I.3.23}$$

in equation (I.3.22) leads to:

$$\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n \cdot \underline{\nabla}\delta p^{n+1} - \mathrm{div}\left(\Delta t \underline{\nabla}\delta p^{n+1}\right) = \Gamma + \underline{\widetilde{u}} \cdot \underline{\nabla}\rho^n - \frac{\mathrm{d}\rho}{\mathrm{d}t} - \mathrm{div}\left(\rho \underline{\widetilde{u}}\right). \tag{I.3.24}$$

Using some relationships between differential operators one obtains:

$$\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n \cdot \underline{\nabla}\delta p^{n+1} = \mathrm{div}\left(\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n \delta p^{n+1}\right) - \delta p^{n+1}\mathrm{div}\left(\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n\right), \tag{I.3.25}$$

leads to a steady convection diffusion equation for the pressure increment:

$$\underbrace{\mathrm{div}\left(\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n \delta p^{n+1}\right) - \mathrm{div}\left(\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n\right)\delta p^{n+1}}_{\text{convection}} - \underbrace{\mathrm{div}\left(\Delta t \underline{\nabla}\delta p^{n+1}\right)}_{\text{diffusion}} = \Gamma - \frac{\mathrm{d}\rho}{\mathrm{d}t} - \rho^n \mathrm{div}\,\underline{\widetilde{u}} \tag{I.3.26}$$

This equation is solved in two steps with $\delta p = \delta p_1 + \delta p_2$ to split it into a pure diffusive step (which is the only remaining step when the density is constant) and a convective/diffusive step:

1. the first step solves the following equation which is closed to the equation solved in the standard algorithm,

$$-\mathrm{div}\left(\Delta t\underline{\nabla}\delta p_1^{n+1}\right) = \Gamma - \frac{\mathrm{d}\rho}{\mathrm{d}t} - \rho^n \mathrm{div}\,\widetilde{\underline{u}} \tag{I.3.27}$$

2. the second step solves the following steady convection diffusion equation (see §6.1):

$$\begin{aligned} -\mathrm{div}\left(\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n\right)\delta p_2^{n+1} + \mathrm{div}\left(\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n\delta p_2^{n+1}\right) - \mathrm{div}\left(\Delta t\underline{\nabla}\delta p_2^{n+1}\right) = \\ \mathrm{div}\left(\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n\right)\delta p_1^{n+1} - \mathrm{div}\left(\frac{\Delta t}{\rho^n}\underline{\nabla}\rho^n\delta p_1^{n+1}\right) \end{aligned} \tag{I.3.28}$$

Chapter 4

Space discretization

4.1 Introduction

4.1.1 Definition and notations

Within the framework of the finite volume approach, the equations are integrated over each cell of the mesh (or *control volume* V_c). This section is limited to a brief description of the way 0^{th}-order, convection, diffusion and gradient terms appearing in the equations are integrated using the budget methodology. Specific attention is devoted to the calculation of gradients, since it is a major characteristic of the co-located finite volume method (all the variables are associated with the same point, namely the cell centre[1]).

Let N_{cell} be the number of cells, then each discretized field Y has N_{cell} degrees of freedom, which are denoted by Y_c, $c \in [1, \cdots, N_{cell}]$ given by definition by:

$$Y_c \equiv \frac{1}{|V_c|} \int_{V_c} Y \mathrm{d}V. \tag{I.4.1}$$

As each discretized field Y is supposed to be linear in every single cell, Y_c can be identified by the value of the field in $\underline{x}_c$, the cell center of V_c:

$$Y_{\underline{x}_c} = Y_c. \tag{I.4.2}$$

0^{th}-order terms: Then, terms of **order** 0 (*i.e.* terms that are not space derivatives) are integrated to introduce their average over the cell. For example, $\rho \underline{g}$ becomes $|V_c|\, \rho_c \underline{g}$. In this expression, $|V_c|$ is the measure of cell volume V_c and ρ_c denotes the average of ρ over the control volume (the cell) V_c applying (I.4.1).

Divergence operator–conservative gradient terms: The **divergence** terms (or *flux* terms, or again *conservative*8 terms) are integrated using the Green relation to introduce cell faces values so that *fluxes* appear naturally. For example, a term such as $\underline{\mathrm{div}}\left(Y \underline{\underline{1}}\right)$ becomes[2]

$$\int_{V_c} \underline{\mathrm{div}}\left(Y \underline{\underline{1}}\right) \mathrm{d}V = \sum_{f \in \mathcal{F}_c} Y_f \underline{S}_{c>\overline{c}}. \tag{I.4.3}$$

In expression (I.4.3), face values of the field Y appear. They are defined as:

$$Y_f \equiv \frac{1}{|\underline{S}|_f} \int_f Y \mathrm{d}S, \tag{I.4.4}$$

so that the relationship (I.4.3) is exact. As the field Y is linear over the face f, Y_f can be associated to the face centre $\underline{x}_f$:

$$Y_{\underline{x}_f} = Y_f. \tag{I.4.5}$$

In the following sections, faces $\mathcal{F}_c$ are usually split into two categories: the interior faces $f_{c|\overline{c}} \in \mathcal{F}_c^{int}$ separating two neighbouring cells c and $\overline{c}$; and the boundary faces $f_b \in \mathcal{F}_c^{ext}$. Outward (with respect to the cell c) normals are respectively denoted $\underline{S}_{c>\overline{c}}$ and $\underline{S}_{c>f_b}$, which means that $\underline{S}_{c>\overline{c}}$ is oriented from c toward f.

Then Y_f is expressed as an average of the degree of freedom, which are for the interface $f_{c|\overline{c}}$ the value of Y_c and $Y_{\overline{c}}$ but also the gradients in these cells. The use of gradients to reach an higher order in space is called *reconstruction* in the following sections. The detailed computation of $\int_{V_c} \underline{\mathrm{div}}\left(Y \underline{\underline{1}}\right) \mathrm{d}\Omega$ is given in § 4.4.1.

[1] The centre of a cell is a geometric point associated with the cell and located preferably inside the cell. Nevertheless, the word *centre* shall not be taken literally, especially in the case of polyhedral cells that do not have a regular shape.

[2] in $\underline{\mathrm{div}}\left(Y \underline{\underline{1}}\right)$, Y might be the pressure field P, this term then corresponds to the pressure gradient in the momentum equation.

Convection operator–mass flux terms: Let us now focus on the convective term div $(Y\rho\underline{u})$. This term and the unsteady term $\dfrac{\partial(\rho Y)}{\partial t}$ will be treated together. As a matter of fact, if the field Y is transported by the convective field $\rho\underline{u}$, the balance of the quantity ρY over a cell c is written using Leibniz theorem as:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{V_c}\rho Y\,\mathrm{d}V\right) &= \int_{V_c}\frac{\partial\rho Y}{\partial t}\mathrm{d}V+\int_{\partial V_c}Y\rho\underline{u}\cdot\mathrm{d}\underline{S},\\ &= \int_{V_c}\frac{\partial\rho Y}{\partial t}+\mathrm{div}\,(Y\rho\underline{u})\,\mathrm{d}V,\end{aligned}\tag{I.4.6}$$

the second line is obtained using Green's relation.

Moreover, the unsteady and convection terms are usually written in a *non-conservative* form that is in continuous formalism:

$$\frac{\partial(\rho Y)}{\partial t}+\mathrm{div}\,(Y\rho\underline{u})=\rho\frac{\partial Y}{\partial t}+\underline{\nabla}Y\cdot(\rho\underline{u})+\Gamma Y.\tag{I.4.7}$$

Note that for (I.4.7) to hold, the continuity equation (I.2.4) must be fulfilled. If (I.4.7) is required even for discrete volumes, the convection term must be defined as:

$$\begin{aligned}\int_{V_c}\underline{\nabla}Y\cdot(\rho\underline{u})\,\mathrm{d}V &\equiv \int_{V_c}\mathrm{div}\,(Y\rho\underline{u})\,\mathrm{d}V-Y_c\int_{V_c}\mathrm{div}\,(\rho\underline{u})\,\mathrm{d}V,\\ &= \int_{\partial V_c}Y\rho\underline{u}\cdot\mathrm{d}\underline{S}-Y_c\int_{\partial V_c}\rho\underline{u}\cdot\mathrm{d}\underline{S},\\ &= \sum_{f\in\mathcal{F}_c}(Y_f-Y_c)\,(\rho\underline{u})_f\cdot\underline{S}_{c>\overline{c}},\end{aligned}\tag{I.4.8}$$

the second line is obtained using once again Green's formula. In formula (I.4.8), one still has to express the face value Y_f and also the value of the mass flux $(\rho\underline{u})_f\cdot\underline{S}_{c>\overline{c}}$: all the available convective schemes (*upwind, centred, SOLU, etc.*) are presented in § 4.2. Let $\dot{m}_{c>\overline{c}}$ be the outgoing mass flux from cell c through the face f:

$$\dot{m}_{c>\overline{c}}\equiv(\rho\underline{u})_f\cdot\underline{S}_{c>\overline{c}},\tag{I.4.9}$$

note that this convective flux is naturally defined at cell faces and thus is stored over there in the code. In the following, the convection term is denoted as follows:

$$\int_{V_c}\underline{\nabla}Y\cdot(\rho\underline{u})\,\mathrm{d}V=\sum_{f\in\mathcal{F}_c}C_{c>\overline{c}}\left(\dot{m}_{c>\overline{c}},\,Y\right),\tag{I.4.10}$$

where $C_{c>\overline{c}}\left(\dot{m}_{c>\overline{c}},\,Y\right)$ is defined by:

$$C_{c>\overline{c}}\left(\dot{m}_f,\,Y\right)\equiv(Y_f-Y_c)\,\dot{m}_{c>\overline{c}}.\tag{I.4.11}$$

Laplacian operator–diffusive terms: Let us discretize the diffusive term div $(K\underline{\nabla}Y)$:

$$\int_{V_c}\mathrm{div}\,(K\underline{\nabla}Y)\,\mathrm{d}V\equiv\sum_{f\in\mathcal{F}_c}K_f\underline{\nabla}_fY\cdot\underline{S}_{c>\overline{c}},\tag{I.4.12}$$

where K_f is the face diffusivity, and $\underline{\nabla}_fY$ is the face gradient, their computation will be detailed in § 4.3. In the following, the diffusive term is denoted as follows:

$$\int_{V_c}\mathrm{div}\,(K\underline{\nabla}Y)\,\mathrm{d}V=\sum_{f\in\mathcal{F}_c}D_{c>\overline{c}}\left(K_f\,Y\right),\tag{I.4.13}$$

where the diffusive flux $D_{c>\overline{c}}\left(K_f\,Y\right)$ over the face f is defined by:

$$D_{c>\overline{c}}\left(K_f,\,Y\right)\equiv K_f\underline{\nabla}_fY\cdot\underline{S}_{c>\overline{c}}.\tag{I.4.14}$$

Note that the diffusive flux $D_{c>\overline{c}}\left(K_{f_{c|\overline{c}}}, Y\right)$ over the interior face $f_{c|\overline{c}}$ lost by the cell c is gained by $\overline{c}$, in other words:

$$D_{c>\overline{c}}\left(K_{f_{c|\overline{c}}}, Y\right) = -D_{\overline{c}>c}\left(K_{f_{c|\overline{c}}}, Y\right). \tag{I.4.15}$$

Remark 4.1 *The diffusion operator can be extended to anisotropic tensor diffusivity* $\underline{\underline{K}}$.

More geometrical quantities: To end up the general description of the discretized operators, let us introduce some geometrical quantities which will be used during the approximation process of linking face fluxes to cell centred quantities. For consistency and to reach a higher order in space, the values of the variables at points $\underline{x}'_c$ and $\underline{x}'_{\overline{c}}$ are used. These points are respectively the projection of the centres $\underline{x}_c$ and $\underline{x}_{\overline{c}}$ along the orthogonal line to the interior face $f_{c|\overline{c}}$ passing through $\underline{x}_f$. When considering a boundary face f_b, $\underline{x}'_c$ is defined as the projection of $\underline{x}_c$ on the normal to the boundary face f_b passing through $\underline{x}_f$. All the geometrical definitions are recalled in Figure I.4.1. Using Taylor series from the values at $\underline{x}_c$ and $\underline{x}_{\overline{c}}$ and from the *cell gradient* in the respective cells, one can write for any field Y:

$$\left\{\begin{array}{rcccl} Y_{\underline{x}'_c} & \simeq & Y_{\underline{x}_c} + \underline{\nabla}_c Y \cdot (\underline{x}'_c - \underline{x}_c) & = & Y_c + \underline{\nabla}_c Y \cdot (\underline{x}'_c - \underline{x}_c), \\ Y_{\underline{x}'_{\overline{c}}} & \simeq & Y_{\underline{x}_{\overline{c}}} + \underline{\nabla}_{\overline{c}} Y \cdot (\underline{x}'_{\overline{c}} - \underline{x}_{\overline{c}}) & = & Y_{\overline{c}} + \underline{\nabla}_{\overline{c}} Y \cdot (\underline{x}'_{\overline{c}} - \underline{x}_{\overline{c}}). \end{array}\right. \tag{I.4.16}$$

Note that for orthogonal meshes (where $\underline{x}'_c = \underline{x}_c$ for all faces of all cells), no *reconstruction* (I.4.16) is needed, and therefore the distance $|\underline{x}'_c - \underline{x}_c|$ measures the *non-orthogonality* of the mesh. The computation of $\underline{\nabla}_c Y$ is presented in § 4.4.1 and § H.

Furthermore, the intersection between $(\underline{x}_{\overline{c}} - \underline{x}_c)$ and the corresponding interior face $f_{c|\overline{c}}$ is denoted by $\underline{x}_o$. The distance $|\underline{x}_f - \underline{x}_o|$ measures the *offset* of the mesh.

Eventually, a weighting factor $\alpha_{c>\overline{c}}$ is defined to measure the distance of the cell center $\underline{x}_c$ to the face $f_{c|\overline{c}}$ relatively to the other cell center $\underline{x}_{\overline{c}}$:

$$\alpha_{c>\overline{c}} = \frac{|\underline{x}'_{\overline{c}} - \underline{x}_f|}{|\underline{x}'_{\overline{c}} - \underline{x}'_c|}. \tag{I.4.17}$$

Note that the distances $|\underline{x}'_{\overline{c}} - \underline{x}'_c|$ and $|\underline{x}'_{\overline{c}} - \underline{x}_f|$ are defined algebraically, that is:

$$\begin{array}{rcl} |\underline{x}'_{\overline{c}} - \underline{x}'_c| & \equiv & \dfrac{(\underline{x}'_{\overline{c}} - \underline{x}'_c) \cdot \underline{S}_{c>\overline{c}}}{|\underline{S}_{c>\overline{c}}|}, \\ |\underline{x}'_{\overline{c}} - \underline{x}_f| & \equiv & \dfrac{(\underline{x}'_{\overline{c}} - \underline{x}_f) \cdot \underline{S}_{c>\overline{c}}}{|\underline{S}_{c>\overline{c}}|}, \end{array} \tag{I.4.18}$$

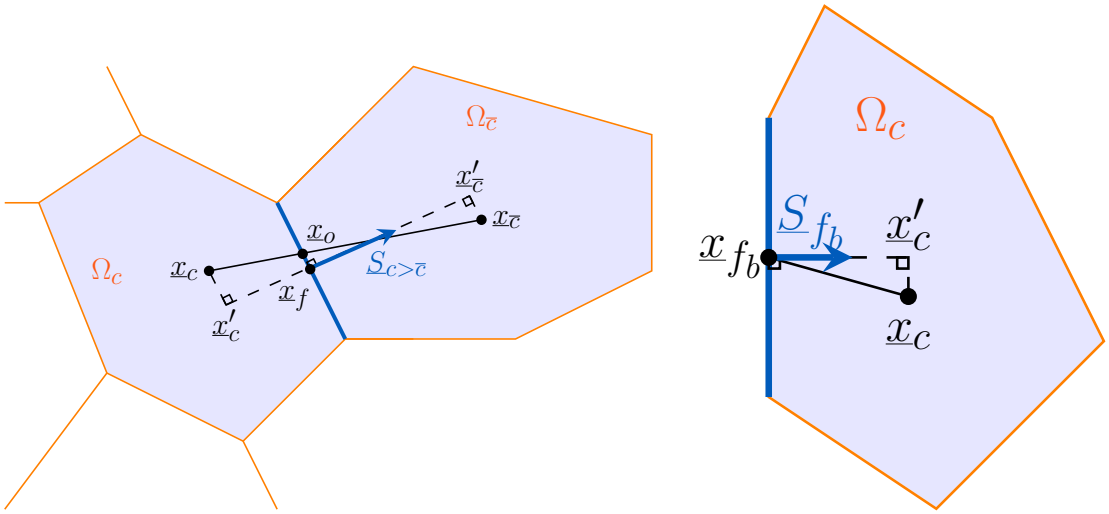
and are supposed to be positive if the mesh is *star-shaped*. Note also that $\alpha_{c>\overline{c}}$ is oriented from c to $\overline{c}$ and

$$\alpha_{c>\overline{c}} + \alpha_{\overline{c}>c} = 1. \tag{I.4.19}$$

See the programmers reference of the dedicated subroutine for more informations on the convective and diffusive terms in code_saturne.

4.2 Convective term

Using the notations adopted in § 4.1.1, the explicit budget corresponding to the integration over a cell V_c of the convective part $\underline{\nabla}_c Y \cdot (\rho \underline{u})$ has been written as a sum of the numerical fluxes $C_{c>\overline{c}}\left(\dot{m}_{c>\overline{c}}, Y\right)$ calculated at the interior faces, and the numerical fluxes $C_{c>f_b}\left(\dot{m}_{c>f_b}, Y\right)$ calculated at the boundary faces of the computational domain Ω defined by Equation (I.4.11).



(a) Internal face (b) Boundary face

Figure I.4.1: Sketch of geometric entities.

Note that $C_{c>f_b}\left(\dot{m}_{c>f_b}, Y\right)$ involves the boundary conditions of the field Y and are described in detail in Chapter 5. The value of Y_{f_b} is expressed as follows:

$$Y_{f_b} \equiv A^g_{f_b} + B^g_{f_b} Y_{\underline{x}'_c}. \tag{I.4.20}$$

The value of the convective flux $C_{c>\bar{c}}\left(\dot{m}_{f_{c|\bar{c}}}, Y\right)$ depends on the numerical scheme. Three different types of convection schemes are available in code_saturne.

4.2.1 Upwind

For a 1^{st}-order upwind scheme, the convective flux reads:

$$C^{upwind}_{c>\bar{c}}\left(\dot{m}_{c>\bar{c}}, Y\right) \equiv \left(Y^{upwind}_{f_{c|\bar{c}}} - Y_c\right)\dot{m}_{c>\bar{c}}, \tag{I.4.21}$$

with

$$Y^{upwind}_{f_{c|\bar{c}}} = \begin{cases} Y_c \text{ if } \dot{m}_{c>\bar{c}} \geqslant 0, \\ Y_{\bar{c}} \text{ if } \dot{m}_{c>\bar{c}} < 0. \end{cases} \tag{I.4.22}$$

4.2.2 Centred

For a centred scheme, the convective flux reads:

$$C^{centred}_{c>\bar{c}}\left(\dot{m}_{c>\bar{c}}, Y\right) \equiv \left(Y^{centred}_{f_{c|\bar{c}}} - Y_c\right)\dot{m}_{c>\bar{c}}, \tag{I.4.23}$$

with

$$Y^{centred}_{f_{c|\bar{c}}} = \alpha_{c>\bar{c}} Y_{\underline{x}'_c} + \left(1-\alpha_{c>\bar{c}}\right) Y_{\underline{x}'_{\bar{c}}}. \tag{I.4.24}$$

Remark 4.2 *We actually write*

$$Y^{centred}_{f_{c|\bar{c}}} = \alpha_{c>\bar{c}} Y_c + \left(1-\alpha_{c>\bar{c}}\right) Y_{\bar{c}} + \frac{1}{2}\left[\underline{\nabla}_c Y + \underline{\nabla}_{\bar{c}} Y\right]\cdot\left(\underline{x}_f - \underline{x}_o\right), \tag{I.4.25}$$

which ensures the second-order discretization in space for Y. The factor $\frac{1}{2}$ is used for numerical stability reasons.

4.2.3 Second Order Linear Upwind (SOLU)

For a 2^{nd}-order linear upwind scheme, the convective flux reads:

$$C_{c>\overline{c}}^{SOLU}\left(\dot{m}_{c>\overline{c}},\, Y\right) \equiv \left(Y_{f_{c|\overline{c}}}^{SOLU} - Y_c\right) \dot{m}_{c>\overline{c}}, \tag{I.4.26}$$

with

$$Y_{f_{c|\overline{c}}}^{SOLU} = \begin{cases} Y_c + \underline{\nabla}_c Y \cdot \left(\underline{x}_f - \underline{x}_c\right) & \text{if } \dot{m}_{c>\overline{c}} \geqslant 0, \\ Y_{\overline{c}} + \underline{\nabla}_{\overline{c}} Y \cdot \left(\underline{x}_f - \underline{x}_{\overline{c}}\right) & \text{if } \dot{m}_{c>\overline{c}} < 0. \end{cases} \tag{I.4.27}$$

The boundary value of $C_{c>f_b}^{SOLU}$ is calculated as:

$$Y_{f_b}^{SOLU} = \begin{cases} Y_c + \underline{\nabla}_c Y \cdot \left(\underline{x}_f - \underline{x}_c\right) & \text{if } \dot{m}_{c>f_b} \geqslant 0, \\ A_{f_b}^g + B_{f_b}^g Y_{\underline{x}'_c} & \text{if } \dot{m}_{c>f_b} < 0. \end{cases} \tag{I.4.28}$$

Remark 4.3 *A slope test (which may introduce non-linearities in the convection operator) allows to switch from the centred or SOLU scheme to the first-order upwind scheme (without blending). Additionally, the default option to deal with $Y_{f_{c|\overline{c}}}$ is computed as a weighted average between the upstream value and the centred value (blending), according to users' choice.*

4.3 Diffusive term

Using the notations adopted in § 4.1.1, the explicit budget corresponding to the integration over a cell V_c of the diffusive term div $(K\underline{\nabla}Y)$ has been written as a sum of the numerical fluxes $D_{c>\overline{c}}\left(K_{f_{c|\overline{c}}},\, Y\right)$ calculated at the internal faces, and the numerical fluxes $D_{c>f_b}\left(K_{f_b},\, Y\right)$ calculated at the boundary faces of the computational domain Ω defined by Equation (I.4.14).

Note that $D_{c>f_b}\left(K_{f_b},\, Y\right)$ includes the **diffusion** boundary conditions of the field Y and are described in detail in Chapter 5. The value of the flux $D_{c>f_b}$ of Y_{f_b} is expressed as follows:

$$\frac{D_{c>f_b}}{|\underline{S}|_{f_b}} \equiv -\left(A_{c>f_b}^f + B_{c>f_b}^f Y_{\underline{x}'_c}\right). \tag{I.4.29}$$

The value of the diffusive flux $D_{c>\overline{c}}$ depends on the *reconstruction* of the field Y and also on the interpolation at the face of the diffusivity K from the cell values. Two interpolations are available:

i/ a *harmonic* interpolation which reads:

$$K_{f_{c|\overline{c}}}^{harmonic} \equiv \frac{K_c K_{\overline{c}}}{\alpha_{c>\overline{c}} K_c + \left(1 - \alpha_{c>\overline{c}}\right) K_{\overline{c}}} \tag{I.4.30}$$

ii/ an *arithmetic* interpolation which reads:

$$K_{f_{c|\overline{c}}}^{arithmetic} \equiv \frac{1}{2}\left(K_c + K_{\overline{c}}\right) \tag{I.4.31}$$

Note that to ensure flux continuity at the internal faces $f_{c|\overline{c}}$, one should use the *harmonic* mean, whereas the *arithmetic* mean is set as the default option because it has been proven to be more robust numerically.

[2] Extrapolation of the upwind value at the faces centre.

4.3.1 Without reconstruction

For a *non-reconstructed* field, the diffusive flux reads:

$$D_{c>\overline{c}}^{NRec}\left(K_{f_{c|\overline{c}}},Y\right)=-\frac{K_{f_{c|\overline{c}}}\left|\underline{S}_{c>\overline{c}}\right|}{\left|\underline{x}'_{\overline{c}}-\underline{x}'_{c}\right|}\left(Y_c-Y_{\overline{c}}\right). \tag{I.4.32}$$

4.3.2 Reconstructed

For a *reconstructed* field, the diffusive flux reads:

$$D_{c>\overline{c}}^{Rec}\left(K_{f_{c|\overline{c}}},Y\right)=-\frac{K_{f_{c|\overline{c}}}\left|\underline{S}_{c>\overline{c}}\right|}{\left|\underline{x}'_{\overline{c}}-\underline{x}'_{c}\right|}\left(Y_{\underline{x}'_c}-Y_{\underline{x}'_{\overline{c}}}\right). \tag{I.4.33}$$

Remark 4.4 *In fact, it is actually written as*

$$\begin{aligned} D_{c>\overline{c}}^{Rec}\left(K_{f_{c|\overline{c}}},Y\right) &= -\frac{K_{f_{c|\overline{c}}}\left|\underline{S}_{c>\overline{c}}\right|}{\left|\underline{x}'_{\overline{c}}-\underline{x}'_{c}\right|}\left(Y_c-Y_{\overline{c}}\right) \\ &- \frac{K_{f_{c|\overline{c}}}\left|\underline{S}_{c>\overline{c}}\right|}{\left|\underline{x}'_{\overline{c}}-\underline{x}'_{c}\right|}\frac{1}{2}\left(\underline{\nabla}_c Y+\underline{\nabla}_{\overline{c}}Y\right)\cdot\left(\left(\underline{x}'_c-\underline{x}_c\right)-\left(\underline{x}'_{\overline{c}}-\underline{x}_{\overline{c}}\right)\right), \end{aligned} \tag{I.4.34}$$

which ensures the second-order discretization in space for Y. The factor $\frac{1}{2}$ is used for numerical stability reasons.

4.3.3 Anisotropic diffusion

The algorithm of anisotropic diffusion can be found in [Ferrand et al., 2014].

4.4 Gradient calculation

The aim of the present section is to describe the algorithms available in code_saturne to compute the cell gradient for scalar or vector fields. The first one uses an iterative process to handle with non-orthogonalities. It is robust but requires computational effort. The second one, the least squares method, minimizes a function. It is faster, but less accurate. On irregular meshes, it leads to smoother gradients,

For both methods, the adaptation to gradients of vectorial fields is also presented.

Please refer to the programmers reference of the dedicated subroutine for further details.

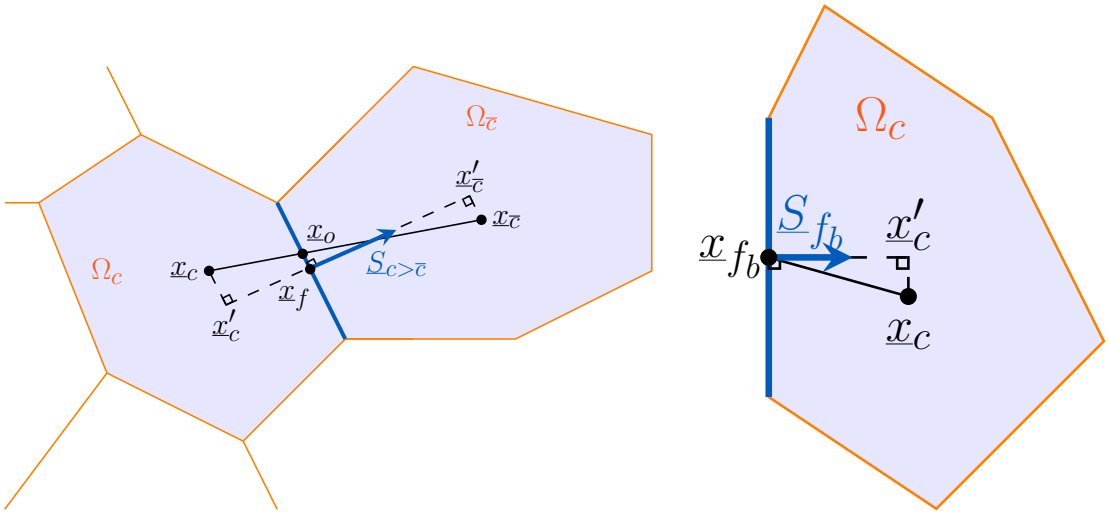
4.4.1 Standard method: iterative process

General description

Notations of the geometrical quantities are recalled in Figure I.4.2. To compute the cell gradient $\underline{\nabla}_c Y$ of the scalar field Y let us start by its definition:

$$\left|V_c\right|\underline{\nabla}_c Y\equiv\int_{V_c}\underline{\nabla}Y\,\mathrm{d}V=\int_{\partial V_c}Y\mathrm{d}\underline{S}. \tag{I.4.35}$$

In order to take the mesh non-orthogonality into account, a Taylor series (1^{st}-order) of $\underline{\nabla}_c Y$ is used as follows:



(a) Internal face (b) Boundary face

Figure I.4.2: Sketch of geometrical quantities.

$$
\begin{aligned}
|V_c|\,\underline{\nabla}_c Y \;&\equiv\; \int_{V_c} \underline{\nabla} Y \,\mathrm{d}V = \sum_{f_{c|\bar{c}}\in\mathcal{F}_c^{int}} Y_{f_{c|\bar{c}}}\,\underline{S}_{c>\bar{c}} + \sum_{f_b\in\mathcal{F}_c^{ext}} Y_{f_b}\,\underline{S}_{c>f_b},\\
&= \sum_{f_{c|\bar{c}}\in\mathcal{F}_c^{int}} Y_{\underline{x}_f}\,\underline{S}_{c>\bar{c}} + \sum_{f_b\in\mathcal{F}_c^{ext}} Y_{\underline{x}_f}\,\underline{S}_{c>f_b},\\
&\simeq \sum_{f_{c|\bar{c}}\in\mathcal{F}_c^{int}} \left[Y_{\underline{x}_o} + \underline{\nabla}_{\underline{x}_o} Y\cdot\left(\underline{x}_f-\underline{x}_o\right)\right]\underline{S}_{c>\bar{c}} + \sum_{f_b\in\mathcal{F}_c^{ext}} \left[\epsilon_{\delta Y} A_{f_b} + B_{f_b} Y_{I'}\right]\underline{S}_{c>f_b},\\
&= \sum_{f_{c|\bar{c}}\in\mathcal{F}_c^{int}} \left[\left(\alpha_{c>\bar{c}} Y_{\underline{x}_c} + (1-\alpha_{c>\bar{c}}) Y_{\underline{x}_{\bar{c}}}\right)\right]\underline{S}_{c>\bar{c}} + \sum_{f_{c|\bar{c}}\in\mathcal{F}_i^{int}} \left[\underline{\nabla}_{f_{c|\bar{c}}} Y\cdot\left(\underline{x}_f-\underline{x}_o\right)\right]\underline{S}_{c>\bar{c}}\\
&+ \sum_{f_b\in\mathcal{F}_c^{ext}} \left[\epsilon_{\delta Y} A_{f_b} + B_{f_b} Y_{I'}\right]\underline{S}_{c>f_b}.
\end{aligned}
\tag{I.4.36}
$$

The variable $\epsilon_{\delta Y}$ is set to 0 for an increment of a variable[3], to 1 for the variable itself in order to take correctly the boundary condition into account.

Using the following 1^{st}-order in space approximation

$$
\left\{
\begin{aligned}
\underline{\nabla}_{f_{c|\bar{c}}} Y \;&=\; \frac{1}{2}\left[\underline{\nabla}_{\underline{x}_c} Y + \underline{\nabla}_{\underline{x}_{\bar{c}}} Y\right],\\
Y_{\underline{x}'_c} \;&=\; Y_{\underline{x}_c} + \underline{\nabla}_{\underline{x}_c} Y\cdot\left(\underline{x}'_c-\underline{x}_c\right).
\end{aligned}
\right.
$$

Equation (I.4.36) becomes:

$$
\begin{aligned}
|V_c|\,\underline{\nabla}_c Y \;&=\; \sum_{f_{c|\bar{c}}\in\mathcal{F}_c^{int}} \left[\alpha_{c>\bar{c}} Y_c + (1-\alpha_{c>\bar{c}}) Y_{\bar{c}} + \frac{1}{2}\left(\underline{\nabla}_c Y + \underline{\nabla}_{\bar{c}} Y\right)\cdot\left(\underline{x}_f-\underline{x}_o\right)\right]\underline{S}_{c>\bar{c}}\\
&+ \sum_{f_b\in\mathcal{F}_c^{ext}} \left[\epsilon_{\delta Y} A_{f_b} + B_{f_b} Y_c + B_{f_b}\underline{\nabla}_c Y\cdot\left(\underline{x}'_c-\underline{x}_c\right)\right]\underline{S}_{c>f_b}.
\end{aligned}
$$

[3]Then a homogeneous condition has to be imposed.

Bringing $\underline{\nabla}_c Y$ terms all together on the left hand side, we have:

$$\begin{aligned}|V_c|\,\underline{\nabla}_c Y - \sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} \frac{1}{2}\underline{\nabla}_c Y\cdot\left(\left(\underline{x}_f-\underline{x}_o\right)\otimes\underline{S}_{c>\overline{c}}\right) - \sum_{f_b\in\mathcal{F}_c^{ext}} B_{f_b}\underline{\nabla}_c Y\cdot\left(\left(\underline{x}'_c-\underline{x}_c\right)\otimes\underline{S}_{c>f_b}\right) &= \sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} [(\alpha_{c>\overline{c}}Y_c + (1-\alpha \\ &+ \sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} \frac{1}{2}\underline{\nabla}_{\overline{c}} Y\cdot\left(\left(\underline{x}_f-\underline{x}\right.\right. \\ &+ \sum_{f_b\in\mathcal{F}_c^{ext}} [\epsilon_{\delta Y}A_{f_b}+B_{f_b}Y_c]\,\underline{S}\end{aligned} \tag{I.4.37}$$

Without reconstruction

On an orthogonal mesh, or if this option is chosen, only 0^{th}-order contributions are considered. Everything is as if $\left(\underline{x}'_c-\underline{x}_c\right)=\underline{0}$ and $\left(\underline{x}_f-\underline{x}_o\right)=\underline{0}$ in the previous calculation:

$$\begin{aligned}|V_c|\,\underline{\nabla}_c Y &\equiv \int_{V_c}\underline{\nabla} Y\,\mathrm{d}V = \sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} Y_{f_{c|\overline{c}}}\underline{S}_{c>\overline{c}} + \sum_{f_b\in\mathcal{F}_c^{ext}} Y_{f_b}\underline{S}_{c>f_b},\\ &= \sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} \left[\alpha_{c>\overline{c}}Y_{\underline{x}_c}+(1-\alpha_{c>\overline{c}})Y_{\underline{x}_{\overline{c}}}\right]\underline{S}_{c>\overline{c}} + \sum_{f_b\in\mathcal{F}_c^{ext}}\left[\epsilon_{\delta Y}A_{f_b}+B_{f_b}Y_{\underline{x}_c}\right]\underline{S}_{c>f_b},\end{aligned}$$

hence

$$\underline{\nabla}_c^{NRec} Y = \frac{1}{|V_c|}\left[\sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}}\left[\alpha_{c>\overline{c}}Y_{\underline{x}_c}+(1-\alpha_{c>\overline{c}})Y_{\underline{x}_{\overline{c}}})\right]\underline{S}_{c>\overline{c}} + \sum_{f_b\in\mathcal{F}_c^{ext}}(\epsilon_{\delta Y}A_{f_b}+B_{f_b}Y_{\underline{x}_c})\underline{S}_{c>f_b}\right]. \tag{I.4.38}$$

Remark 4.5 *The non-reconstructed gradient is denoted by $\underline{\nabla}_c^{NRec}Y$, and is then very easy to compute thanks to the Equation (I.4.38). However, it is neither accurate nor consistent on a non-orthogonal mesh.*

Handling with reconstruction: iterative process

In order to solve system (I.4.37), all terms containing $\underline{\nabla}_c Y$ are implicit, whereas all terms with $\underline{\nabla}_{\overline{c}} Y$ are explicit, we then use the series $\left(\delta\underline{\nabla}_c^k Y\right)_{k\in\mathbb{N}}$ defined by:

$$\left\{\begin{array}{rcl}\delta\underline{\nabla}_c^0 Y &=& \underline{\nabla}_c^{NRec}Y,\\ \delta\underline{\nabla}_c^{k+1} Y &=& \underline{\nabla}_c^{k+1}Y - \underline{\nabla}_c^k Y,\end{array}\right. \tag{I.4.39}$$

and the associated system is:

$$\begin{aligned}\underline{\nabla}_c^{k+1}Y\cdot\left[|V_c|\,\underline{\underline{1}} - \sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}}\frac{1}{2}\left(\underline{x}_f-\underline{x}_o\right)\otimes\underline{S}_{c>\overline{c}} - \sum_{f_b\in\mathcal{F}_c^{ext}} B_{f_b}\left(\underline{x}'_c-\underline{x}_c\right)\otimes\underline{S}_{c>f_b}\right] &= \sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} [(\alpha_{c>\overline{c}}Y_c+(1-\alpha_{c>\overline{c}})Y_{\overline{c}}) \\ &+ \sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} \frac{1}{2}\underline{\nabla}_{\overline{c}}^k Y\cdot\left(\left(\underline{x}_f-\underline{x}_o\right)\otimes\underline{S}_{c}\right. \\ &+ \sum_{f_b\in\mathcal{F}_c^{ext}}[\epsilon_{\delta Y}A_{f_b}+B_{f_b}Y_c]\,\underline{S}_{c>f_b},\end{aligned} \tag{I.4.40}$$

or, as the following relationship stands:

$$\underline{\nabla}_c^{k+1} Y = \underline{\nabla}_c^{k} Y + \delta\underline{\nabla}_c^{k+1} Y,$$

$$\begin{aligned}
\delta\underline{\nabla}_c^{k+1} Y \cdot \left[|V_c| \underline{\underline{1}} - \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \frac{1}{2} \left(\underline{x}_f - \underline{x}_o \right) \otimes \underline{S}_{c>\overline{c}} - \sum_{f_b \in \mathcal{F}_c^{ext}} B_{f_b} \left(\underline{x}'_c - \underline{x}_c \right) \otimes \underline{S}_{c>f_b} \right] &= - |V_c| \underline{\nabla}_c^{k} Y + \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} [(\alpha_{c>\overline{c}} Y_c - \\
&+ \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \frac{1}{2} \left(\underline{\nabla}_c^{k} Y + \underline{\nabla}_{\overline{c}}^{k} Y \right) \cdot \left((\underline{x}_f \right. \\
&+ \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\epsilon_{\delta Y} A_{f_b} + B_{f_b} \left(Y_c + \underline{\nabla}_c^{k} \right. \right.
\end{aligned} \tag{I.4.41}$$

The Equation (I.4.41) is a local 3×3 matrix which unknowns are each of the three components of the vector $\delta\underline{\nabla}_c^{k+1} Y$. Finally, for each cell c we get:

$$\underbrace{\begin{bmatrix} \delta\underline{\nabla}_{c,x}^{k+1} Y \\ \delta\underline{\nabla}_{c,y}^{k+1} Y \\ \delta\underline{\nabla}_{c,z}^{k+1} Y \end{bmatrix}}_{\delta\underline{\nabla}_c^{k+1} Y} \cdot \underbrace{\begin{bmatrix} C_{c,xx} & C_{c,xy} & C_{c,xz} \\ C_{c,yx} & C_{c,yy} & C_{c,yz} \\ C_{c,zx} & C_{c,zy} & C_{c,zz} \end{bmatrix}}_{\underline{\underline{C}}_c} = \underbrace{\begin{bmatrix} R_{c,x}^{k+1} \\ R_{c,y}^{k+1} \\ R_{c,z}^{k+1} \end{bmatrix}}_{\underline{R}_c^{k+1}}, \tag{I.4.42}$$

with:

$$\left\{ \begin{aligned}
\underline{\underline{C}}_c &= |V_c| \underline{\underline{1}} - \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \frac{1}{2} \left(\underline{x}_f - \underline{x}_o \right) \otimes \underline{S}_{c>\overline{c}} - \sum_{f_b \in \mathcal{F}_c^{ext}} B_{f_b} \left(\underline{x}'_c - \underline{x}_c \right) \otimes \underline{S}_{c>f_b}, \\
\underline{R}_c^{k+1} &= - |V_c| \underline{\nabla}_c^{k} Y + \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \left[(\alpha_{c>\overline{c}} Y_c + (1 - \alpha_{c>\overline{c}}) Y_{\overline{c}}) \right] \underline{S}_{c>\overline{c}} \\
&+ \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \frac{1}{2} \left(\underline{\nabla}_c^{k} Y + \underline{\nabla}_{\overline{c}}^{k} Y \right) \cdot \left((\underline{x}_f - \underline{x}_o) \otimes \underline{S}_{c>\overline{c}} \right) \\
&+ \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\epsilon_{\delta Y} A_{f_b} + B_{f_b} \left(Y_c + \underline{\nabla}_c^{k} Y \cdot (\underline{x}'_c - \underline{x}_c) \right) \right] \underline{S}_{c>f_b}.
\end{aligned} \right. \tag{I.4.43}$$

The inverse of the matrix $\underline{\underline{C}}_c$ is used to compute $\left(\delta\underline{\nabla}_c^{k+1} Y \right)$ and so $\left(\underline{\nabla}_c^{k+1} Y \right)$. The iterative process stops as soon as the Euclidean norm of the right-hand-side $\underline{R}_c^{k+1}$ becomes smaller than a chosen threshold (*i.e.* when the Euclidean norm of $\left(\delta\underline{\nabla}_c^{k} Y \right)$ falls under a threshold) or when the number of iterations reaches the maximal number of iterations.

4.4.2 Standard method: iterative process for vectorial fields

In this section, the adaptation of the calculation presented in § 4.4.1 is adapted to vectorial fields. Some minor modifications are required, especially for the boundary condition treatment, but the core of the formulae are the very similar. The notations of the geometrical quantities are recalled in Figure I.4.2.

The definition of $\underline{\underline{\nabla}}_c \underline{v}$ reads:

$$|V_c| \underline{\underline{\nabla}}_c \underline{v} \equiv \int_{V_c} \underline{\underline{\nabla}} \underline{v} \, \mathrm{d}V = \int_{\partial V_c} \underline{v} \otimes \mathrm{d}\underline{S}. \tag{I.4.44}$$

The same Taylor series as (I.4.36) of $\underline{\underline{\nabla}}_c \underline{v}$ is used:

$$
\begin{aligned}
|V_c|\,\underline{\underline{\nabla}}_c \underline{v} &\equiv \int_{V_c} \underline{\underline{\nabla}} \underline{v}\,\mathrm{d}V = \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \underline{v}_{f_{c|\overline{c}}} \otimes \underline{S}_{c>\overline{c}} + \sum_{f_b \in \mathcal{F}_c^{ext}} \underline{v}_{f_b} \otimes \underline{S}_{c>f_b}, \\
&= \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \underline{v}_{\underline{x}_f} \otimes \underline{S}_{c>\overline{c}} + \sum_{f_b \in \mathcal{F}_c^{ext}} \underline{v}_{\underline{x}_f} \otimes \underline{S}_{c>f_b}, \\
&\simeq \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \left[\underline{v}_{\underline{x}_o} + \underline{\underline{\nabla}}_{\underline{x}_o} \underline{v} \cdot (\underline{x}_f - \underline{x}_o) \right] \otimes \underline{S}_{c>\overline{c}} + \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\epsilon_{\delta \underline{v}} \underline{A}_{f_b} + \underline{\underline{B}}_{f_b} \cdot \underline{v}_{I'} \right] \otimes \underline{S}_{c>f_b}, \\
&= \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \left[\left(\alpha_{c>\overline{c}} \underline{v}_{\underline{x}_c} + (1-\alpha_{c>\overline{c}}) \underline{v}_{\underline{x}_{\overline{c}}} \right) \right] \otimes \underline{S}_{c>\overline{c}} + \sum_{f_{c|\overline{c}} \in \mathcal{F}_i^{int}} \left[\underline{\underline{\nabla}}_{f_{c|\overline{c}}} \underline{v} \cdot (\underline{x}_f - \underline{x}_o) \right] \otimes \underline{S}_{c>\overline{c}} \\
&+ \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\epsilon_{\delta \underline{v}} \underline{A}_{f_b} + \underline{\underline{B}}_{f_b} \cdot \underline{v}_{I'} \right] \otimes \underline{S}_{c>f_b}.
\end{aligned}
\tag{I.4.45}
$$

Once again, the variable $\epsilon_{\delta \underline{v}}$ is set to 0 for an increment of a variable, to 1 for the variable itself in order to correctly account for the boundary condition.

The same 1^{st}-order in space approximation as in the scalar gradient calculation is used:

$$
\left\{
\begin{aligned}
\underline{\underline{\nabla}}_{f_{c|\overline{c}}} \underline{v} &= \frac{1}{2} \left[\underline{\underline{\nabla}}_{\underline{x}_c} \underline{v} + \underline{\underline{\nabla}}_{\underline{x}_{\overline{c}}} \underline{v} \right], \\
\underline{v}_{I'} &= \underline{v}_{\underline{x}_c} + \underline{\underline{\nabla}}_{\underline{x}_c} \underline{v} \cdot (\underline{x}'_c - \underline{x}_c).
\end{aligned}
\right.
$$

Equation (I.4.45) becomes:

$$
\begin{aligned}
|V_c|\,\underline{\underline{\nabla}}_c \underline{v} &= \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \left[\alpha_{c>\overline{c}} \underline{v}_c + (1-\alpha_{c>\overline{c}}) \underline{v}_{\overline{c}} + \frac{1}{2} \left(\underline{\underline{\nabla}}_c \underline{v} + \underline{\underline{\nabla}}_{\overline{c}} \underline{v} \right) \cdot (\underline{x}_f - \underline{x}_o) \right] \otimes \underline{S}_{c>\overline{c}} \\
&+ \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\epsilon_{\delta \underline{v}} \underline{A}_{f_b} + \underline{\underline{B}}_{f_b} \cdot \underline{v}_c + \underline{\underline{B}}_{f_b} \cdot \left(\underline{\underline{\nabla}}_c \underline{v} \cdot (\underline{x}'_c - \underline{x}_c) \right) \right] \otimes \underline{S}_{c>f_b}.
\end{aligned}
$$

Note that, there is no simple possibility here to bring $\underline{\underline{\nabla}}_c \underline{v}$ terms all together on the left hand side, because the term $\underline{\underline{B}}_{f_b} \cdot \left(\underline{\underline{\nabla}}_c \underline{v} \right) \cdot \left((\underline{x}'_c - \underline{x}_c) \otimes \underline{S}_{c>f_b} \right)$ cannot be factorised easily, and thus will be explicit:

$$
\begin{aligned}
|V_c|\,\underline{\underline{\nabla}}_c \underline{v} - \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \frac{1}{2} \underline{\underline{\nabla}}_c \underline{v} \cdot \left((\underline{x}_f - \underline{x}_o) \otimes \underline{S}_{c>\overline{c}} \right) &= \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \left[(\alpha_{c>\overline{c}} \underline{v}_c + (1-\alpha_{c>\overline{c}}) \underline{v}_{\overline{c}}) \right] \otimes \underline{S}_{c>\overline{c}} \\
&+ \sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \frac{1}{2} \underline{\underline{\nabla}}_{\overline{c}} \underline{v} \cdot \left((\underline{x}_f - \underline{x}_o) \otimes \underline{S}_{c>\overline{c}} \right) \\
&+ \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\epsilon_{\delta \underline{v}} \underline{A}_{f_b} + \underline{\underline{B}}_{f_b} \cdot \underline{v}_c \right] \otimes \underline{S}_{c>f_b} + \sum_{f_b \in \mathcal{F}_c^{ext}} \underline{\underline{B}}_{f_b} \underline{\underline{\nabla}}_c \underline{v} \cdot \left((\underline{x}'_c - \underline{x}_c) \otimes \right.
\end{aligned}
\tag{I.4.46}
$$

Without reconstruction

Without reconstruction, the vectorial gradient reads:

$$
\underline{\underline{\nabla}}_c^{NRec} \underline{v} = \frac{1}{|V_c|} \left[\sum_{f_{c|\overline{c}} \in \mathcal{F}_c^{int}} \left[\alpha_{c>\overline{c}} \underline{v}_{\underline{x}_c} + (1-\alpha_{c>\overline{c}}) \underline{v}_{\underline{x}_{\overline{c}}} \right] \otimes \underline{S}_{c>\overline{c}} + \sum_{f_b \in \mathcal{F}_c^{ext}} (\epsilon_{\delta \underline{v}} \underline{A}_{f_b} + \underline{\underline{B}}_{f_b} \cdot \underline{v}_{\underline{x}_c}) \otimes \underline{S}_{c>f_b} \right].
\tag{I.4.47}
$$

Handling with reconstruction: iterative process

The series $\left(\delta\underline{\underline{\nabla}}_c^k \underline{v}\right)_{k\in\mathbb{N}}$ is defined by:

$$\left\{\begin{array}{rcl} \delta\underline{\underline{\nabla}}_c^0 \underline{v} & = & \underline{\underline{\nabla}}_c^{NRec} \underline{v}, \\ \delta\underline{\underline{\nabla}}_c^{k+1} \underline{v} & = & \underline{\underline{\nabla}}_c^{k+1} \underline{v} - \underline{\underline{\nabla}}_c^k \underline{v}, \end{array}\right. \tag{I.4.48}$$

A system similar to Equation (I.4.41) is obtained for each cell c

$$\delta\underline{\underline{\nabla}}_c^{k+1} \underline{v} \cdot \underline{\underline{C}}_c = \underline{R}_c^{k+1} \tag{I.4.49}$$

with:

$$\left\{\begin{array}{rcl} \underline{\underline{C}}_c & = & \underline{\underline{1}} - \dfrac{1}{|V_c|} \displaystyle\sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} \dfrac{1}{2}\left(\underline{x}_f - \underline{x}_o\right) \otimes \underline{S}_{c>\overline{c}}, \\ \underline{R}_c^{k+1} & = & -\underline{\underline{\nabla}}_c^k \underline{v} + \dfrac{1}{|V_c|} \displaystyle\sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} \left[(\alpha_{c>\overline{c}} \underline{v}_c + (1-\alpha_{c>\overline{c}})\underline{v}_{\overline{c}})\right] \otimes \underline{S}_{c>\overline{c}} \\ & + & \dfrac{1}{|V_c|} \displaystyle\sum_{f_{c|\overline{c}}\in\mathcal{F}_c^{int}} \dfrac{1}{2}\left(\underline{\underline{\nabla}}_c^k \underline{v} + \underline{\underline{\nabla}}_{\overline{c}}^k \underline{v}\right) \cdot \left(\left(\underline{x}_f - \underline{x}_o\right) \otimes \underline{S}_{c>\overline{c}}\right) \\ & + & \dfrac{1}{|V_c|} \displaystyle\sum_{f_b\in\mathcal{F}_c^{ext}} \left[\epsilon_{\delta\underline{v}} \underline{A}_{f_b} + \underline{\underline{B}}_{f_b}\left(\underline{v}_c + \underline{\underline{\nabla}}_c^k \underline{v} \cdot (\underline{x}'_c - \underline{x}_c)\right)\right] \otimes \underline{S}_{c>f_b}. \end{array}\right. \tag{I.4.50}$$

Remark 4.6 *Note that the matrix $\underline{\underline{C}}_c$ in (I.4.50) is not the same as in (I.4.43). First of all, there is no boundary term and thus its inverse does not need to be recomputed at each iteration (except if the mesh is modified). This matrix thus only measures the quality of the mesh (if the mesh is orthogonal, $\underline{\underline{C}}_c = \underline{\underline{1}}$ for all cells). Secondly, this matrix is dimensionless, whereas in (I.4.43) $\underline{\underline{C}}_c$ has the dimension of a volume. This choice has been motivated to minimize truncation errors.*

4.4.3 Least-squares method

Notations of the geometrical quantities are recalled in Figure I.4.2. The aim of the present algorithm is to compute the cell gradient $\underline{\nabla}_c Y$ of the scalar field Y using a least squares method. The idea is to evaluate the gradient of the variable at the cell faces using the value of the gradient at the cell centres. The method is not expected to be as robust as the *iterative* process presented in § 4.4.1, but much more efficient.

Let us introduce $\underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}}$ an estimation at the internal face $f_{c|\overline{c}}$ of the gradient projected in the direction $\underline{d}_{c>\overline{c}}$ (which will be chosen afterwards). Let us also define the analogous quantity for boundary faces f_b: $\underline{\nabla}_{f_b} Y \cdot \underline{d}_{c>f_b}$ ($\underline{d}_{c>f_b}$ will also be chosen afterwards).

The goal would be to find $\underline{\nabla}_c Y$ such that, for all faces the following relationships hold:

$$\left\{\begin{array}{rcl} \underline{\nabla}_c Y \cdot \underline{d}_{c>\overline{c}} & = & \underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}}, \\ \underline{\nabla}_c Y \cdot \underline{d}_{c>f_b} & = & \underline{\nabla}_{f_b} Y \cdot \underline{d}_{c>f_b}. \end{array}\right. \tag{I.4.51}$$

The previous equality is generally not reachable for all the faces, so the problem is reformulated as the minimisation of the $\mathcal{F}_c$ function:

$$\mathcal{F}_c\left(\underline{v}\right) = \frac{1}{2} \sum_{\overline{c}\in Neigh(c)} \left[\underline{v} \cdot \underline{d}_{c>\overline{c}} - \underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}}\right]^2 + \frac{1}{2} \sum_{f_b\in\mathcal{F}_c^{ext}} \left[\underline{v} \cdot \underline{d}_{c>f_b} - \underline{\nabla}_{f_b} Y \cdot \underline{d}_{c>f_b}\right]^2, \tag{I.4.52}$$

where $\overline{c} \in Neigh\,(c)$ is the neighbourhood of the cell c. By default, the neighbourhood is composed of cells which share at least a face with c. But *extended* neighbouring can be used.

To minimize $\mathcal{F}_c$, derivatives with respect to the components of the vector $\underline{v}$ are computed, the resulting system is solved and $\underline{\nabla}_c Y$ is defined as $\underline{v}_{\min}$ such that $\mathcal{F}_c\,(\underline{v}_{\min})$ is minimum.

In order to solve the systems for each cell c separately from one to an other, vectors $\underline{d}_{c>\overline{c}}$ and $\underline{d}_{c>f_b}$ are chosen so that the quantities $\underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}}$ and $\underline{\nabla}_{f_b} Y \cdot \underline{d}_{c>f_b}$ do not depend on neighbour cell gradients $\underline{\nabla}_{\overline{c}} Y$. The following choice makes it possible:

$$\begin{aligned} \underline{d}_{c>\overline{c}} &= \frac{\underline{x}_{\overline{c}} - \underline{x}_c}{\left|\underline{x}_{\overline{c}} - \underline{x}_c\right|}, \\ \underline{d}_{c>f_b} &= \frac{\underline{x}_f - \underline{x}'_c}{\left|\underline{x}_f - \underline{x}'_c\right|} = \underline{n}_{c>f_b}. \end{aligned} \tag{I.4.53}$$

Thus, for internal faces $f_{c|\overline{c}}$, $\underline{d}_{c>\overline{c}}$ is the normalized vector joining the centres $\underline{x}_c$ and $\underline{x}_{\overline{c}}$ oriented from cell c to $\overline{c}$. The quantity $\underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}}$ is given by:

$$\underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}} = \frac{Y_{\overline{c}} - Y_c}{\left|\underline{x}_{\overline{c}} - \underline{x}_c\right|}. \tag{I.4.54}$$

For boundary faces, the choice $\underline{d}_{c>f_b}$ to be the outward normal implies:

$$\underline{\nabla}_{f_b} Y \cdot \underline{d}_{c>f_b} = \frac{Y_{f_b} - Y_{\underline{x}'_c}}{\left|\underline{x}_f - \underline{x}'_c\right|}, \tag{I.4.55}$$

where Y_{f_b} is expressed thanks to the boundary conditions (see Chapter 5) and the value $Y_{\underline{x}'_c}$ is given by formula (I.4.16) recalled hereafter:

$$\left\{ \begin{aligned} Y_{\underline{x}'_c} &= Y_c + \underline{\nabla}_c Y \cdot (\underline{x}'_c - \underline{x}_c) \\ Y_{f_b} &= A^g_{f_b} + B^g_{f_b} Y_{\underline{x}'_c} = A^g_{f_b} + B^g_{f_b}\,(Y_c + \underline{\nabla}_c Y \cdot (\underline{x}'_c - \underline{x}_c)) \end{aligned} \right. \tag{I.4.56}$$

Eventually we get:

$$\underline{\nabla}_{f_b} Y \cdot \underline{d}_{c>f_b} = \frac{A^g_{f_b} + \left(B^g_{f_b} - 1\right)(Y_c + \underline{\nabla}_c Y \cdot (\underline{x}'_c - \underline{x}_c))}{\left|\underline{x}_f - \underline{x}'_c\right|}, \tag{I.4.57}$$

Equation (I.4.57) contains a term in $\underline{\nabla}_c Y$ and thus should be injected into Equation (I.4.52) before deriving it. Thus (I.4.52) becomes:

$$\begin{aligned} \mathcal{F}_c\,(\underline{v}) &= \frac{1}{2} \sum_{\overline{c} \in Neigh(c)} \left[\underline{v} \cdot \underline{d}_{c>\overline{c}} - \underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}}\right]^2 \\ &+ \frac{1}{2} \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\underline{v} \cdot \left(\underline{d}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|}\,(\underline{x}'_c - \underline{x}_c)\right) - \frac{A^g_{f_b} + \left(B^g_{f_b} - 1\right) Y_c}{\left|\underline{x}_f - \underline{x}'_c\right|}\right]^2. \end{aligned} \tag{I.4.58}$$

Then we cancel the derivatives of $\mathcal{F}_c\,(\underline{v})$ with respect to the $\underline{v}$ components:

$$\begin{aligned} \frac{\partial \mathcal{F}_c}{\partial \underline{v}}\,(\underline{v}) &= \sum_{\overline{c} \in Neigh(c)} \left[(\underline{v} \cdot \underline{d}_{-c>f})\,\underline{d}_{c>\overline{c}} - \left(\underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}}\right) \underline{d}_{c>\overline{c}}\right] \\ &+ \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\left(\underline{v} \cdot \left(\underline{d}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|}\,(\underline{x}'_c - \underline{x}_c)\right)\right)\left(\underline{d}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|}\,(\underline{x}'_c - \underline{x}_c)\right) - \frac{A^g_{f_b} + \left(B^g_{f_b} - 1\right) Y_c}{\left|\underline{x}_f - \underline{x}'_c\right|}\left(\underline{d}_c\right. \end{aligned} \tag{I.4.59}$$

A 3×3 system for each cell c is obtained by writing $\dfrac{\partial \mathcal{F}_c}{\partial \underline{v}} \left(\underline{\nabla}_c Y\right) = \underline{0}$:

$$\underline{\nabla}_c Y \cdot \underline{\underline{C}}_c = \underline{R}_c, \tag{I.4.60}$$

with

$$\left\{ \begin{array}{rcl} \underline{\underline{C}}_c & = & \displaystyle\sum_{\overline{c} \in Neigh(c)} \underline{d}_{c>\overline{c}} \otimes \underline{d}_{c>\overline{c}} + \sum_{f_b \in \mathcal{F}_c^{ext}} \left(\underline{d}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|} \left(\underline{x}'_c - \underline{x}_c\right) \right) \otimes \left(\underline{d}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|} \left(\underline{x}'_c - \underline{x}_c\right) \right), \\ \underline{R}_c & = & \displaystyle\sum_{\overline{c} \in Neigh(c)} \left(\underline{\nabla}_{f_{c|\overline{c}}} Y \cdot \underline{d}_{c>\overline{c}} \right) \underline{d}_{c>\overline{c}} + \sum_{f_b \in \mathcal{F}_c^{ext}} \frac{A^g_{f_b} + \left(B^g_{f_b} - 1\right) Y_c}{\left|\underline{x}_f - \underline{x}'_c\right|} \left(\underline{d}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|} \left(\underline{x}'_c - \underline{x}_c\right) \right), \end{array} \right. \tag{I.4.61}$$

using (I.4.53) this gives:

$$\left\{ \begin{array}{rcl} \underline{\underline{C}}_c & = & \displaystyle\sum_{\overline{c} \in Neigh(c)} \frac{\left(\underline{x}_{\overline{c}} - \underline{x}_c\right) \otimes \left(\underline{x}_{\overline{c}} - \underline{x}_c\right)}{\left|\underline{x}_{\overline{c}} - \underline{x}_c\right|^2} + \sum_{f_b \in \mathcal{F}_c^{ext}} \left(\underline{n}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|} \left(\underline{x}'_c - \underline{x}_c\right) \right) \otimes \left(\underline{n}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|} \left(\underline{x}'_c - \underline{x}_c\right) \right), \\ \underline{R}_c & = & \displaystyle\sum_{\overline{c} \in Neigh(c)} \left(Y_{\overline{c}} - Y_c\right) \frac{\left(\underline{x}_{\overline{c}} - \underline{x}_c\right)}{\left|\underline{x}_{\overline{c}} - \underline{x}_c\right|^2} + \sum_{f_b \in \mathcal{F}_c^{ext}} \frac{A^g_{f_b} + \left(B^g_{f_b} - 1\right) Y_c}{\left|\underline{x}_f - \underline{x}'_c\right|} \left(\underline{n}_{c>f_b} - \frac{B^g_{f_b} - 1}{\left|\underline{x}_f - \underline{x}'_c\right|} \left(\underline{x}'_c - \underline{x}_c\right) \right). \end{array} \right. \tag{I.4.62}$$

Remark 4.7 *i/ Note that the 3×3 $\underline{\underline{C}}_c$ tensor is symmetric.*

ii/ For cells c having at least a boundary face f_b, the tensor $\underline{\underline{C}}_c$ must be recomputed at each time step, for the other, the tensor $\underline{\underline{C}}_c$ only needs to be recomputed when the mesh is updated (in ALE for instance).

iii/ If the user chooses not to reconstruct the gradients (which introduces a lack of consistency on non-orthogonal meshes), then the gradient is computed thanks to formula (I.4.38).

iv/ For highly non-orthogonal meshes, an extended stencil[4] can be used (see different support in Figure I.4.3) and can drastically improve the results when using tetrahedral meshes.

4.4.4 Least-squares method for vectorial fields

In this section, the adaptation of the calculation presented in § 4.4.3 is adapted to vectorial fields. Some minor modifications are required, especially for the boundary condition treatment, but the core of the formulae are the very similar. The notations of the geometrical quantities are recalled in Figure I.4.2.

The functional to be minimized is defined by

$$\mathcal{F}_c\left(\underline{\underline{t}}\right) = \frac{1}{2} \sum_{\overline{c} \in Neigh(c)} \left| \underline{\underline{t}} \cdot \underline{d}_{c>\overline{c}} - \underline{\underline{\nabla}}_{f_{c|\overline{c}}} \underline{v} \cdot \underline{d}_{c>\overline{c}} \right|^2 + \frac{1}{2} \sum_{f_b \in \mathcal{F}_c^{ext}} \left| \underline{\underline{t}} \cdot \underline{d}_{c>f_b} - \underline{\underline{\nabla}}_{f_b} \underline{v} \cdot \underline{d}_{c>f_b} \right|^2, \tag{I.4.63}$$

To minimize $\mathcal{F}_c$, derivatives with respect to the components of the vector $\underline{\underline{t}}$ are computed, the resulting system is solved and $\underline{\underline{\nabla}}_c \underline{v}$ is defined as $\underline{\underline{t}}_{\min}$ such that $\mathcal{F}_c\left(\underline{\underline{t}}_{\min}\right)$ is minimum.

In order to obtain a simple system, the same choice as in (I.4.53) is made on $\underline{d}_{c>\overline{c}}$. Concerning $\underline{d}_{c>f_b}$, an other choice ensuring that $\underline{\nabla}_{f_b} Y \cdot \underline{d}_{c>f_b}$ does not depend on $\underline{\nabla}_c Y$ is made:

$$\begin{array}{rcl} \underline{d}_{c>\overline{c}} & = & \dfrac{\underline{x}_{\overline{c}} - \underline{x}_c}{\left|\underline{x}_{\overline{c}} - \underline{x}_c\right|}, \\ \underline{d}_{c>f_b} & = & \dfrac{\underline{x}_f - \underline{x}_c}{\left|\underline{x}_f - \underline{x}_c\right|}. \end{array} \tag{I.4.64}$$

[4] the stencil is the set of neighbour cells used in the cell gradient computation.

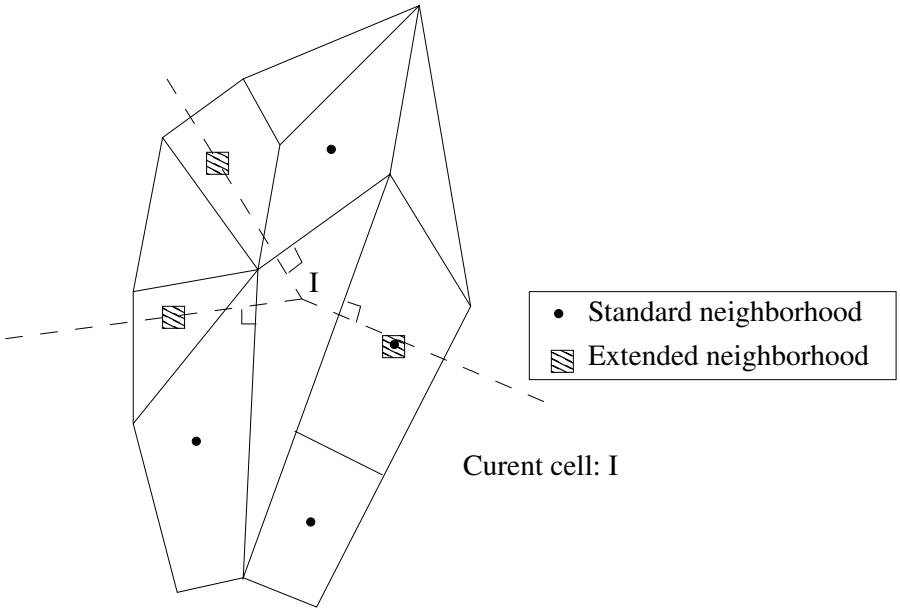


Figure I.4.3: Available stencils for computing the gradient with the least squares method.

Thus, for internal faces $f_{c|\overline{c}}$, $\underline{d}_{c>\overline{c}}$ is still the normalized vector joining the centres $\underline{x}_c$ and $\underline{x}_{\overline{c}}$ oriented from cell c to $\overline{c}$. The quantity $\underline{\underline{\nabla}}_{f_{c|\overline{c}}}\underline{v}\cdot\underline{d}_{c>\overline{c}}$ is given by:

$$\underline{\underline{\nabla}}_{f_{c|\overline{c}}}\underline{v}\cdot\underline{d}_{c>\overline{c}}=\frac{\underline{v}_{\overline{c}}-\underline{v}_c}{|\underline{x}_{\overline{c}}-\underline{x}_c|}. \tag{I.4.65}$$

For boundary faces, the choice $\underline{d}_{c>f_b}$ implies:

$$\underline{\underline{\nabla}}_{f_b}\underline{v}\cdot\underline{d}_{c>f_b}=\frac{\underline{v}_{f_b}-\underline{v}_c}{|\underline{x}_f-\underline{x}_c|}, \tag{I.4.66}$$

where $\underline{v}_{f_b}$ is expressed thanks to the boundary conditions (see Chapter 5).

Then we cancel the derivatives of $\mathcal{F}_c\left(\underline{\underline{t}}\right)$ with respect to the $\underline{\underline{t}}$ components:

A 3×3 system for each cell c is obtained by writing $\dfrac{\partial\mathcal{F}_c}{\partial\underline{\underline{t}}}\left(\underline{\underline{\nabla}}_c\underline{v}\right)=\underline{\underline{0}}$:

$$\underline{\underline{\nabla}}_c\underline{v}\cdot\underline{\underline{C}}_c=\underline{\underline{R}}_c, \tag{I.4.67}$$

with

$$\begin{cases}\underline{\underline{C}}_c &= \displaystyle\sum_{\overline{c}\in Neigh(c)}\underline{d}_{c>\overline{c}}\otimes\underline{d}_{c>\overline{c}}+\sum_{f_b\in\mathcal{F}_c^{ext}}\underline{d}_{c>f_b}\otimes\underline{d}_{c>f_b},\\ \underline{\underline{R}}_c &= \displaystyle\sum_{\overline{c}\in Neigh(c)}\underline{\underline{\nabla}}_{f_{c|\overline{c}}}\underline{v}\cdot(\underline{d}_{c>\overline{c}}\otimes\underline{d}_{c>\overline{c}})+\sum_{f_b\in\mathcal{F}_c^{ext}}\frac{\underline{A}^g_{f_b}+\left(\underline{\underline{B}}^g_{f_b}-\underline{\underline{1}}\right)\cdot\underline{v}_c}{|\underline{x}_f-\underline{x}_c|}\otimes\underline{d}_{c>f_b}.\end{cases} \tag{I.4.68}$$

To handle Neumann BC components, we use a fixed-point scheme to compute face values based on formula (I.5.1):

- Initialize the value at $\underline{v}_{f_b}$ using the value at $\underline{v}_c$ in place of $\underline{v}_{\underline{x}'_c}$.
- Compute a first estimate of $\underline{\underline{\nabla}}_c\underline{v}$based on these boundary values.

- Update $\underline{v}_{c'}$ based on this first gradient estimate and recompute $\underline{v}_{f_b}$.
- Compute a new estimate of $\underline{\underline{\nabla}}_c Y$ based on these updated boundary values.
- Iterate until convergence is reached.

Remark 4.8 *Note that the* 3×3 $\underline{\underline{C}}_c$ *tensor is slightly different from* (I.4.60) *and does not depend on boundary condition coefficients (and thus does not require re-computation if the mesh is not modified).*

4.5 Compatible Discrete Operator (CDO) schemes

4.5.1 Introduction

Compatible Discrete Operator schemes have been recently introduced in code_saturne to handle distorted and/or polyhedral meshes in a robust way. CDO schemes gather several space discretization schemes according to the localization of the degrees of freedom (DoFs). Namely, *vertex-based* schemes when DoFs are located at mesh vertices, *vertex+cell-based* schemes when DoFs are both located at mesh cells and mesh vertices, *face-based* schemes when DoFs are located at mesh faces, *edge-based* schemes when DoFs are located at mesh edges (one scalar by edge in the case of a vector-valued unknows since the circulation of vector field is the DoF) and *cell-based* schemes when DoFs are located at mesh cells (for the potential) and mesh faces (for the flux). The latter corresponds to a mixed formulation and is only available up to now for scalar-valued steady diffusion equation. All these schemes belong to the broad family of *compatible* or *mimetic* or *structure-preserving* schemes.

CDO schemes have been devised and mathematically analyzed during three PhD thesis. Part of these works is available in code_saturne:

- Bonelle's Phd [Bonelle, 2014] for vertex-based, face-based and cell-based schemes in the case of steady anisotropic diffusion equation and an edge-based scheme for a steady *curl-curl* equation;
- Cantin's Phd [Cantin, 2016] for the scalar-valued unsteady convection/diffusion/reaction equation with vertex-based schemes and vertex+cell-based schemes;
- Milani's PhD [Milani, 2020] for the discretization of the steady/unsteady Stokes and Navier–Stokes equations using face-based schemes. Two velocity-pressure coupling algorithms are investigated and analyzed: (1) a fully coupled (or *monolithic*) algorithm and (2) an *artificial compressibility* algorithm.

4.5.2 Diffusion operator

The discretization of the diffusion is detailed in several reports and articles. The EDF's report [Bonelle, 2012] (in French) is a good introduction. The link with other space discretizations and the mathematical analysis (stability, convergence proof and *a priori* error estimates) is available in [Bonelle and Ern, 2014]. The key operator is the discrete Hodge operator. This operator can be built in several ways. An example of the influence of the choice of the discrete Hodge operator on the quality of the solution is available in [Bonelle et al., 2015].

An detailed explanation on the way to enforce Dirichlet boundary conditions in vertex-based scheme along with the description of vertex+cell-based schemes for scalar-valued diffusion equation is available in [Cantin, 2016].

4.5.3 Convection operator

Several advection schemes are detailed in [Cantin, 2016] for scalar-valued vertex-based schemes. The classical upwind and centered discretizations are discussed along with a *Péclet-robust* scheme which automatically applies an upwind weighting according to the value of the local Péclet number.

An upwind and centered schemes are detailed in [Milani, 2020] in the case of scalar- and vector-valued face-based schemes.

4.5.4 Stokes and Navier–Stokes equations

The discretization of the Stokes or Navier–Stokes equations in the case of a laminar flow with a constant mass density and viscosity is detailed in [Milani, 2020]. With these assumptions, the viscous term is simply a vector-valued Laplacian operator.

Two types of velocity/pressure coupling algorithms are available:

- A *monolithic* algorithm: a fully coupled algorithm between the velocity components and the pressure DoFs. This yields to a linear system which is a saddle-point problem. A specific algorithm to solve steady-state problem is also available.
- An *artificial compressibility* algorithm [Guermond and Minev, 2015] up to a second-order accuracy [Milani et al., 2022]. Only unsteady problems are tackled with this algorithm. This is an experimental feature.

A short introduction to this discretization describing the main operators in the steady case with the monolithic approach is available in [Bonelle et al., 2020].

4.6 Advanced topics

4.6.1 Rhie & Chow filter

Please refer to the programmers reference of the dedicated keyword `arak`.

4.6.2 Handling of the hydrostatic pressure

This section aims at describing the option `iphydr` corresponding to an improved interpolation scheme, activated by default in the user interface (set to 1). It imposes the equilibrium of the static part of the pressure with any external force, head losses included (it can also be set to 2 in user routines to compute the hydrostatic pressure with an apriori momentum equation to obtain a hydrostatic pressure taking into account the imbalance between the pressure gradient and the gravity source term). In the following we describe option `iphydr`=1. When density effects are important, it improves the interpolation of the pressure and corrects non-physical velocities in highly stratifed areas, near horizontal walls or also near high head losses zones.

The pressure is split into its hydrostatic and dynamic parts following the relation:

$$p(\underline{x},\, t) = p^h(\underline{x},\, t) + p^\star(\underline{x},\, t). \tag{I.4.69}$$

Gradient from iterative process

Gradients are computed thanks to the relation described in Section 4.4.1 which involves values of the considered field at faces that require interpolation from cell values.

Internal faces. In the following, only the dynamic part is interpolated, while the hydrostatic is assumed in equilibrium with the source term $\underline{f} = \rho \underline{g}$ (hence $\underline{\nabla} p^h = \underline{f}$). The dynamic pressure $p_f^\star$ at each face is defined by using a linear interpolation between the discrete pressures p_c and $p_{\overline{c}}$ with an additional correction for non orthogonalities:

$$p_f^\star = \alpha_f p_c^\star + (1-\alpha_f)\, p_{\overline{c}}^\star + \epsilon \frac{1}{2} \left(\underline{x}_f - \underline{x}_o\right) \cdot \left(\underline{\nabla}_c p^\star + \underline{\nabla}_{\overline{c}} p^\star\right) \tag{I.4.70}$$

ϵ is set to 1 if non orthogonalities are reconstructed, 1 otherwise. Hence the pressure writes

$$\begin{aligned} p_f = & \quad p_f^h + p_f^\star \\ = & \quad p_f^h + \alpha_f p_c^\star + (1-\alpha_f)\, p_{\overline{c}}^\star + \frac{1}{2}\epsilon\left(\underline{x}_f - \underline{x}_o\right)\cdot\left(\underline{\nabla}_c p^\star + \underline{\nabla}_{\overline{c}} p^\star\right) \\ = & \quad p_f^h + \alpha_f p_c + (1-\alpha_f)\, p_{\overline{c}} - \alpha_f p_c^h - (1-\alpha_f)\, p_{\overline{c}}^h + \epsilon\frac{1}{2}\left(\underline{x}_f - \underline{x}_o\right)\cdot\left(\underline{\nabla}_c p - \underline{f}_{_c} + \underline{\nabla}_{\overline{c}} p - \underline{f}_{_\overline{c}}\right) \end{aligned}$$

As

$$p_c^h = p_f^h + \left(\underline{x}_c - \underline{x}_f\right)\cdot\underline{f}_{_c} \text{ and } p_{\overline{c}}^h = p_f^h + \left(\underline{x}_{\overline{c}} - \underline{x}_f\right)\cdot\underline{f}_{_\overline{c}} \tag{I.4.71}$$

One gets

$$\begin{aligned} p_f = & \quad \alpha_f p_c + (1-\alpha_f)\, p_{\overline{c}} + \frac{1}{2}\epsilon\left(\underline{x}_f - \underline{x}_o\right)\cdot\left(\underline{\nabla}_c p - \underline{f}_{_c} + \underline{\nabla}_{\overline{c}} p - \underline{f}_{_\overline{c}}\right) \\ & -\alpha_f\left(\underline{x}_c - \underline{x}_f\right)\cdot\underline{f}_{_c} - (1-\alpha_f)\left(\underline{x}_{\overline{c}} - \underline{x}_f\right)\cdot\underline{f}_{_\overline{c}} \end{aligned}$$

Hence

$$\begin{aligned} p_f = & \quad \alpha_f\left(p_c - \left(\underline{x}_c - \underline{x}_f\right)\cdot\underline{f}_{_c}\right) + (1-\alpha_f)\left(p_{\overline{c}} - \left(\underline{x}_{\overline{c}} - \underline{x}_f\right)\cdot\underline{f}_{_\overline{c}}\right) \\ & +\frac{1}{2}\epsilon\left(\underline{x}_f - \underline{x}_o\right)\cdot\left(\underline{\nabla}_c p - \underline{f}_{_c} + \underline{\nabla}_{\overline{c}} p - \underline{f}_{_\overline{c}}\right) \end{aligned}$$

So that

$$\begin{aligned} p_f - \left(p_c - \left(\underline{x}_c - \underline{x}_f\right)\cdot\underline{f}_{_c}\right) = & \quad (1-\alpha_f)\left(p_{\overline{c}} - \left(\underline{x}_{\overline{c}} - \underline{x}_f\right)\cdot\underline{f}_{_\overline{c}} - p_c + \left(\underline{x}_c - \underline{x}_f\right)\cdot\underline{f}_{_c}\right) \\ & +\frac{1}{2}\epsilon\left(\underline{x}_f - \underline{x}_o\right)\cdot\left(\underline{\nabla}_c p - \underline{f}_{_c} + \underline{\nabla}_{\overline{c}} p - \underline{f}_{_\overline{c}}\right) \end{aligned}$$

The classical formula corresponding to the first two lines is therefore incremented of additional terms.

Boundary faces. If a Dirichlet is imposed, no special treatment is required. In other cases, there is usually a zero-flux condition. In this approach, this condition is applied only on the dynamic part of the pressure

$$p_{f_b} = p_{f_b}^h + p_{f_b}^\star = p_{f_b}^h + p_{c'}^\star = p_{f_b}^h + p_{c'} - p_{c'}^h \tag{I.4.72}$$

At first order, one gets

$$p_{c'}^h = p_{f_b}^h + \left(\underline{x}_c' - \underline{x}_{f_b}\right)\cdot\underline{\nabla}_c p^h = p_{f_b}^h + \left(\underline{x}_c' - \underline{x}_{f_b}\right)\cdot\underline{f}_{_c} \tag{I.4.73}$$

Hence

$$p_{f_b} = p_{c'} - \left(\underline{x}_c' - \underline{x}_{f_b}\right)\cdot\underline{f}_{_c} \tag{I.4.74}$$

So for general boundary contributions, one gets

$$p_{f_b} = inc \times A_{f_b}^g + B_{f_b}^g\left(p_c + \left(\underline{x}_c' - \underline{x}_c\right)\cdot\underline{\nabla}_c p - \left(\underline{x}_c' - \underline{x}_{f_b}\right)\cdot\underline{f}_{_c}\right) \tag{I.4.75}$$

Where $A_{f_b}^g$ corresponds to the Dirichlet coefficient and $B_{f_b}^g$ the Neumann coefficient.

By summing over the faces and using the properties $\sum_f \underline{S}_f = \underline{0}$ and $\sum_f \underline{x}_f \otimes \underline{S}_f = V_c\underline{\underline{I}}$, one gets:

$$\begin{aligned} |V_c|\underline{\nabla}_c p = & \quad |V_c|\underline{f}_{_c} + \sum_f\left[(1-\alpha_f)\left(p_{\overline{c}} - \left(\underline{x}_{\overline{c}} - \underline{x}_f\right)\cdot\underline{f}_{_\overline{c}} - p_c + \left(\underline{x}_c - \underline{x}_f\right)\cdot\underline{f}_{_c}\right)\right. \\ & \left. +\frac{1}{2}\epsilon\left(\underline{x}_f - \underline{x}_o\right)\cdot\left(\underline{\nabla}_c p - \underline{f}_{_c} + \underline{\nabla}_{\overline{c}} p - \underline{f}_{_\overline{c}}\right)\right]\underline{S}_f \\ & +\sum_{f_b}\left[inc \times A_{f_b}^g + \left(B_{f_b}^g - 1\right)\left(p_c - \left(\underline{x}_c - \underline{x}_{f_b}\right)\cdot\underline{f}_{_c}\right) + B_{f_b}^g\left(\underline{x}_c' - \underline{x}_c\right)\cdot\left(\underline{\nabla}_c p - \underline{f}_{_c}\right)\right]\underline{S}_{f_b} \end{aligned}$$

Gradient from least squares method

As in Section 4.4.3, the idea is to compute the gradient of the variable at the cell faces using the value of the gradient at the cell centers. One aims at finding $\underline{\nabla}_c Y$ such that for all faces

$$\begin{cases} \underline{\nabla}_c p^\star \cdot \underline{d}_{c>\overline{c}} = \underline{\nabla}_{f_{c|\overline{c}}} p^\star \cdot \underline{d}_{c>\overline{c}} \\ \underline{\nabla}_c p^\star \cdot \underline{d}_{c>f_b} = \underline{\nabla}_{f_b} p^\star \cdot \underline{d}_{c>f_b} \end{cases} \tag{I.4.76}$$

where $\underline{d}_{c>\overline{c}} = \frac{\underline{x}_{\overline{c}} - \underline{x}_c}{|\underline{x}_{\overline{c}} - \underline{x}_c|}$ and $\underline{d}_{c>f_b} = \frac{\underline{x}_{f_b} - \underline{x}'_c}{|\underline{x}_{f_b} - \underline{x}'_c|}$. Pressure is split into its hydrostatic and dynamic parts. The hydrostatic part gives $\underline{\nabla}_c p^h = \underline{f}_c$. For the dynamic part, the problem is formulated as the minimization of the function:

$$\mathcal{F}_c\left(\underline{\nabla}_c p^\star\right) = \frac{1}{2} \sum_{\overline{c} \in Neigh(c)} \left[\underline{\nabla}_c p^\star \cdot \underline{d}_{c>\overline{c}} - \underline{\nabla}_{f_{c|\overline{c}}} p^\star \cdot \underline{d}_{c>\overline{c}}\right]^2 + \frac{1}{2} \sum_{f_b \in \mathcal{F}_c^{ext}} \left[\underline{\nabla}_c p^\star \cdot \underline{d}_{c>f_b} - \underline{\nabla}_{f_b} p^\star \cdot \underline{d}_{c>f_b}\right]^2 \tag{I.4.77}$$

Internal faces. One can write:

$$\begin{aligned} & \underline{\nabla}_{f_{c|\overline{c}}} p^\star \cdot \underline{d}_{c>\overline{c}} \\ = \; & \frac{1}{|\underline{x}_{\overline{c}} - \underline{x}_c|} \left(p^\star_{\overline{c}} - p^\star_c\right) \\ = \; & \frac{1}{|\underline{x}_{\overline{c}} - \underline{x}_c|} \left(p_{\overline{c}} - p_c - p^h_{\overline{c}} + p^h_c\right) \\ = \; & \frac{1}{|\underline{x}_{\overline{c}} - \underline{x}_c|} \left(p_{\overline{c}} - p_c - p^h_f - \left(\underline{x}_{\overline{c}} - \underline{x}_f\right) \cdot \underline{f}_{\overline{c}} + p^h_f + \left(\underline{x}_c - \underline{x}_f\right) \cdot \underline{f}_c\right) \\ = \; & \frac{1}{|\underline{x}_{\overline{c}} - \underline{x}_c|} \left(p_{\overline{c}} - p_c + \left(\underline{x}_c - \underline{x}_f\right) \cdot \underline{f}_c - \left(\underline{x}_{\overline{c}} - \underline{x}_f\right) \cdot \underline{f}_{\overline{c}}\right) \end{aligned}$$

Boundary faces. Considering only Dirichlet and Neumann conditions, the general condition writes:

$$\underline{\nabla}_{f_b} p^\star \cdot \underline{d}_{c>f_b} = \frac{A^g_{f_b} + \left(B^g_{f_b} - 1\right) p_c}{|\underline{x}_{f_b} - \underline{x}'_c|} + \frac{B^g_{f_b} - 1}{|\underline{x}_{f_b} - \underline{x}'_c|} \left(\underline{x}'_c - \underline{x}_c\right) \cdot \underline{\nabla}_c p^\star + \frac{B^g_{f_b} - 1}{|\underline{x}_{f_b} - \underline{x}'_c|} \left(\underline{x}_{f_b} - \underline{x}_c\right) \cdot \underline{f}_c \tag{I.4.78}$$

For an homogeneous Neumann condition $A^g_{f_b} = 0$ and $B^g_{f_b} = 0$ so that $\underline{\nabla}_{f_b} p^\star \cdot \underline{d}_{c>f_b} = 0$. For a Dirichlet condition $B^g_{f_b} = 0$ and one gets:

$$\begin{aligned} & \underline{\nabla}_{f_b} p^\star \cdot \underline{d}_{c>f_b} \\ = \; & \frac{1}{|\underline{x}_{f_b} - \underline{x}'_c|} \left(p^\star_{f_b} - p^\star_{c'}\right) \\ = \; & \frac{1}{|\underline{x}_{f_b} - \underline{x}'_c|} \left(p_{f_b} - p_{c'} - p^h_{f_b} + p^h_{c'}\right) \\ = \; & \frac{1}{|\underline{x}_{f_b} - \underline{x}'_c|} \left(A^g_{f_b} - p_c - \left(\underline{x}'_c - \underline{x}_c\right) \cdot \underline{\nabla}_c p - \left(\underline{x}_{f_b} - \underline{x}'_c\right) \cdot \underline{f}_c\right) \\ = \; & \frac{1}{|\underline{x}_{f_b} - \underline{x}'_c|} \left(A^g_{f_b} - p_c - \left(\underline{x}'_c - \underline{x}_c\right) \underline{\nabla}_c p^\star - \left(\underline{x}'_c - \underline{x}_c\right) \cdot \underline{f}_c - \left(\underline{x}_{f_b} - \underline{x}'_c\right) \cdot \underline{f}_c\right) \\ = \; & \frac{1}{|\underline{x}_f - \underline{x}'_c|} \left(A^g_{f_b} - p_c - \left(\underline{x}'_c - \underline{x}_c\right) \cdot \underline{\nabla}_c p^\star - \left(\underline{x}_{f_b} - \underline{x}_c\right) \cdot \underline{f}_c\right) \end{aligned}$$

Hence, one is minimizing

$$\begin{aligned}\mathcal{F}_c\left(\underline{\nabla}_c p^\star\right) =& \frac{1}{2}\sum_{\bar{c}\in Neigh(c)}\left[\underline{\nabla}_c p^\star\cdot\underline{d}_{c>\bar{c}}-\frac{1}{|\underline{x}_{\bar{c}}-\underline{x}_c|}\left(p_{\bar{c}}-p_c+\left(\underline{x}_c-\underline{x}_f\right)\cdot\underline{f}_c-\left(\underline{x}_{\bar{c}}-\underline{x}_f\right)\cdot\underline{f}_{\bar{c}}\right)\right]^2\\ &+\frac{1}{2}\sum_{f_b\in\mathcal{F}_c^{ext}}\left[\underline{\nabla}_c p^\star\cdot\underline{d}_{c>f_b}-\frac{A^g_{f_b}+\left(B^g_{f_b}-1\right)p_c}{|\underline{x}_{f_b}-\underline{x}'_c|}-\frac{B^g_{f_b}-1}{|\underline{x}_{f_b}-\underline{x}'_c|}\left(\underline{x}'_c-\underline{x}_{f_b}\right)\cdot\underline{\nabla}_c p^\star\right.\\ &\left.-\frac{B^g_{f_b}-1}{|\underline{x}_{f_b}-\underline{x}'_c|}\left(\underline{x}_{f_b}-\underline{x}_c\right)\cdot\underline{f}_c\right]^2\end{aligned}$$

Let us write

$$\underline{d}^\star_{c>f_b}=\underline{d}_{c>f_b}-\frac{B^g_{f_b}-1}{|\underline{x}_{f_b}-\underline{x}'_c|}\left(\underline{x}'_c-\underline{x}_c\right) \tag{I.4.79}$$

One gets

$$\begin{aligned}\mathcal{F}_c\left(\underline{\nabla}_c p^\star\right) =& \frac{1}{2}\sum_{\bar{c}\in Neigh(c)}\left[\underline{\nabla}_c p^\star\cdot\underline{d}_{c>\bar{c}}-\frac{1}{|\underline{x}_{\bar{c}}-\underline{x}_c|}\left(p_{\bar{c}}-p_c+\left(\underline{x}_c-\underline{x}_f\right)\cdot\underline{f}_c-\left(\underline{x}_{\bar{c}}-\underline{x}_f\right)\cdot\underline{f}_{\bar{c}}\right)\right]^2\\ &+\frac{1}{2}\sum_{f_b\in\mathcal{F}_c^{ext}}\left[\underline{\nabla}_c p^\star\cdot\underline{d}^\star_{c>f_b}-\frac{A^g_{f_b}+\left(B^g_{f_b}-1\right)p_c}{|\underline{x}_{f_b}-\underline{x}'_c|}-\frac{B^g_{f_b}-1}{|\underline{x}_{f_b}-\underline{x}'_c|}\left(\underline{x}_{f_b}-\underline{x}_c\right)\cdot\underline{f}_c\right]^2\end{aligned}$$

As in Section 4.4.3, by differentiating with respect to the three components of the gradient and taking the results equal to zero, three equations can then be derived and solved as a system to get the pressure gradient.

Please refer to the programmers reference of the dedicated keyword `iphydr`.

4.6.3 Staggered 1D scheme

Finite volume methods can use either staggered-grid or co-located-grid discretizations. If the former generally result in greater stability and robustness, it proves to be more difficult to implement for complex grid and be more resource-consuming. However, in a one-dimensional framework, it appears as a natural discretization to avoid handling drawbacks of co-localization such as checkerboard issues. We detail in the following a 1D staggered approach for code_saturne: while pressure remain located at cell centers, the velocities will be defined at cell faces.

Description

We consider a simplified 1D formulation of the Navier Stokes equations without diffusive nor convective effects. The physics one wants to model are taken into account through an isotropic head loss term. Therefore we do not need to solve the prediction step and we can perform the following resolution only:

$$\begin{cases}\dfrac{\left(\rho\underline{u}\right)^{n+1}-\left(\rho\underline{u}\right)^n}{\Delta t}+\rho^{n+1}K\left(\underline{u}^n\right)\underline{u}^{n+1}+\underline{\nabla}P^{n+1}=\rho^n\underline{g}+\underline{TS}^n\\ \text{div}\left(\left(\rho\underline{u}\right)^{n+1}\right)=0\end{cases} \tag{I.4.80}$$

Where $\underline{TS}$ denotes a source term. We integrate the gravity contribution in the pressure gradient by writing $p=P-\rho_0\underline{g}\cdot\underline{x}$ and decomposing the density into $\rho^n=\rho_0+\delta\rho^n$. The first equation can be written:

$$\left(\frac{1}{\Delta t}+K\left(\underline{u}^n\right)\right)\left(\rho\underline{u}\right)^{n+1}-\frac{1}{\Delta t}\left(\rho\underline{u}\right)^n=-\underline{\nabla}p^{n+1}+\delta\rho^n\underline{g}+\underline{TS}^n \tag{I.4.81}$$

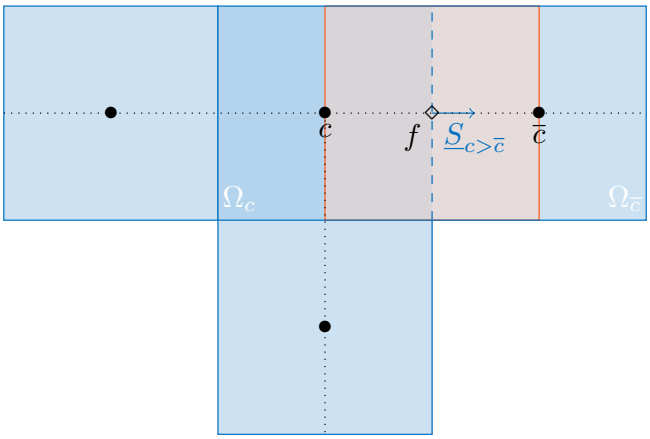


Figure I.4.4: Pseudo-1D tee configuration and notations.

Let us denote the term:

$$\tau^n = \left(\frac{1}{\Delta t} + K\left(\underline{u}^n\right) \right)^{-1} \tag{I.4.82}$$

The relation becomes:

$$\left(\rho \underline{u}\right)_f^{n+1} - \frac{\tau^n}{\Delta t}\left(\rho \underline{u}\right)_f^n = -\tau^n \underline{\nabla}_f p^{n+1} + \tau^n \left(\delta\rho^n \underline{g} + \underline{TS}_f^n\right) \tag{I.4.83}$$

In the following we will consider that we are working at constant densities:

$$\left(\rho \underline{u}\right)_f^{n+1} - \frac{\tau^n}{\Delta t}\left(\rho \underline{u}\right)_f^n = -\tau^n \underline{\nabla}_f p^{n+1} + \tau^n \underline{TS}_f^n \tag{I.4.84}$$

One can then take the divergence and use the divergence-free condition to write:

$$\mathrm{div}_c\left(\tau^n \underline{\nabla} p^{n+1}\right) = \mathrm{div}_c\left(\frac{\tau^n}{\Delta t}\left(\rho \underline{u}\right)^n\right) + \mathrm{div}_c\left(\tau^n \underline{TS}_f^n\right) \tag{I.4.85}$$

At the inlet boundary with an imposed velocity (we assume there is no source term in this zone), one can set a Neumann condition on the pressure:

$$\tau^n \frac{\partial p}{\partial n} = \frac{\tau^n}{\Delta t}\left(\rho \underline{u}\right)^n \cdot \underline{n} - \left(\rho \underline{u}\right)^{n+1} \cdot \underline{n} \tag{I.4.86}$$

Implementation

Boundary condition The inlet boundary condition has to be consistent with the particular handling of the pressure resolution. For an inlet with an imposed velocity, one can set a Neumann condition on the pressure:

$$\tau^n \frac{\partial p}{\partial n} = \frac{\tau^n}{\Delta t} \underbrace{\left(\rho \underline{u}\right)^n \cdot \underline{n}}_{mass_flux_prev} - \underbrace{\left(\rho \underline{u}\right)^{n+1} \cdot \underline{n}}_{mass_flux_new} \tag{I.4.87}$$

Velocity The new velocity, which is not used in the computation (can be seen as post-processing), is computed from mass fluxes (`irevmc=1` option):

$$\underline{u}_c^{n+1} = \frac{1}{\rho_c \Omega_c} \sum_{f_{c|\overline{c}}} \left(\underline{x}_{f_{c|\overline{c}}} - \underline{x}_c\right)\left(\rho \underline{u}\right)_{f_{c|\overline{c}}}^{n+1} \cdot \underline{S}_{ij} \tag{I.4.88}$$

With porosity

One can include the porosity ϵ in the derivation:

$$\begin{cases} \dfrac{(\epsilon\rho\underline{u})^{n+1} - (\epsilon\rho\underline{u})^{n}}{\Delta t} + \epsilon\rho^{n+1} K\left(\underline{u}^{n}\right)\underline{u}^{n+1} + \epsilon\underline{\nabla} P^{n+1} = \epsilon\rho^{n}\underline{g} + \epsilon\underline{TS}^{n} \\ \text{div}\left((\epsilon\rho\underline{u})^{n+1}\right) = 0 \end{cases} \tag{I.4.89}$$

We integrate the gravity contribution in the pressure gradient by writing $p = P - \rho_0 \underline{g}\cdot\underline{x}$ and decomposing the density into $\rho^n = \rho_0 + \delta\rho^n$. The first equation can be written:

$$\left(\frac{1}{\Delta t} + K\left(\underline{u}^{n}\right)\right)(\epsilon\rho\underline{u})^{n+1} - \frac{1}{\Delta t}(\epsilon\rho\underline{u})^{n} = -\epsilon\underline{\nabla} p^{n+1} + \epsilon\delta\rho^{n}\underline{g} + \epsilon\underline{TS}^{n} \tag{I.4.90}$$

Let us denote the term:

$$\tau^{n} = \left(\frac{1}{\Delta t} + K\left(\underline{u}^{n}\right)\right)^{-1} \tag{I.4.91}$$

The relation becomes:

$$(\epsilon\rho\underline{u})_{f}^{n+1} - \frac{\tau^{n}}{\Delta t}(\epsilon\rho\underline{u})_{f}^{n} = -\epsilon\tau^{n}\underline{\nabla}_{f} p^{n+1} + \tau^{n}\left(\epsilon\delta\rho^{n}\underline{g} + \epsilon\underline{TS}_{f}^{n}\right) \tag{I.4.92}$$

In the following we will consider that we are working at constant densities:

$$(\epsilon\rho\underline{u})_{f}^{n+1} - \frac{\tau^{n}}{\Delta t}(\epsilon\rho\underline{u})_{f}^{n} = -\epsilon\tau^{n}\underline{\nabla}_{f} p^{n+1} + \epsilon\tau^{n}\underline{TS}_{f}^{n} \tag{I.4.93}$$

One can then take the divergence and use the divergence-free condition to write:

$$\text{div}_{c}\left(\epsilon\tau^{n}\underline{\nabla} p^{n+1}\right) = \text{div}_{c}\left(\epsilon\frac{\tau^{n}}{\Delta t}(\rho\underline{u})^{n}\right) + \text{div}_{c}\left(\epsilon\tau^{n}\underline{TS}_{f}^{n}\right) \tag{I.4.94}$$

At the inlet boundary with an imposed velocity (we assume there is no source term in this zone), one can set a Neumann condition on the pressure:

$$\epsilon\tau^{n}\frac{\partial p}{\partial n} = \frac{\tau^{n}}{\Delta t}(\epsilon\rho\underline{u})^{n}\cdot\underline{n} - (\epsilon\rho\underline{u})^{n+1}\cdot\underline{n} \tag{I.4.95}$$

Please refer to the programmers reference of the dedicated keyword `staggered`.

Chapter 5

Boundary conditions

5.1 Introduction

Boundary conditions are required in at least three main cases:

- calculation of the convection terms (first order derivative in space) at the boundary: the code uses a mass flux at the boundary and requires the value of the convected variable when the flow is entering into the domain (or more general wave relations in the sense of the characteristic curves of the system entering the domain);
- calculation of the diffusion terms (second order derivative in space): the code needs a method to determine the value of the first order spatial derivatives at the boundary, these define *e.g.* the stresses or the thermal fluxes at the wall;
- calculation of the cell gradients: the variables at the boundary faces allow the code to define the gradient inside the cell connected to the boundary (*e.g.* the pressure gradient or the transpose gradient terms in the stress-strain relation).

These considerations only concern the computational field variables (velocity, pressure, Reynolds tensor, scalars solution of a advection-diffusion equations *etc.*). For these variables [1], the user has to define the boundary conditions at every boundary face.

The boundary conditions could be of Neumann type (when the flux is imposed) or Dirichlet type (when the value of the field variable is prescribed), or mixed type, also called Robin type (when a combination linking the field variable to its derivative flux is imposed).

The code (see the programmers reference of the dedicated subroutine) transforms the boundary conditions provided by the user into two internal formats of representation of the boundary conditions.

A particular treatment is performed on walls: wall functions are used to model the boundary layer flow in the vicinity of the wall when the mesh is too coarse to correctly capture the sharp variations of the fields with a linear profile in the near wall cell. This will be detailed in the next sections.

A particular treatment on symmetry boundaries is also performed for vectors and tensors whereas a symmetry boundary is equivalent to an homogeneous Neumann condition (zero normal gradient) for scalar fields.

The physics model that the user wishes to apply needs to be translated into pairs of coefficients entering the linear system of equations that the code will solve. For any variable Y for every boundary faces f_b these coefficients are:

- $\left(A^g_{f_b},\, B^g_{f_b}\right)$ used by the gradient operator and by the advection operator. The value at the boundary face f_b of the variable Y is then defined as:

$$Y_{f_b} \equiv A^g_{f_b} + B^g_{f_b} Y(\underline{x}'_c). \tag{I.5.1}$$

- $\left(A^f_{c>f_b},\, B^f_{c>f_b}\right)$ used by the diffusion operator. The value at the boundary face f_b of the diffusive flux $q_{c>f_b}$ of the variable Y is then defined as (see Equation (I.4.29)):

$$q_{c>f_b} \equiv \frac{D_{c>f_b}\left(K_{f_b},\, Y\right)}{|\underline{S}|_{f_b}} = -\left(A^f_{c>f_b} + B^f_{c>f_b} Y(\underline{x}'_c)\right). \tag{I.5.2}$$

  Note that the diffusive boundary coefficients are oriented, which means that they are such that if $D_{c>f_b}\left(K_{f_b},\, Y\right)$ is positive, this flux is gained by Y_c.

The definitions are recalled on Figure I.5.1.

[1] The other variables (the physical properties for instance) have a different treatment which will not be detailed here (for instance, for the density, the user defines directly the values at the boundary. This information is then stored; one is referred to `cs_user_physical_properties` (see the programmers reference of the dedicated subroutine) or `phyvar` for more information).

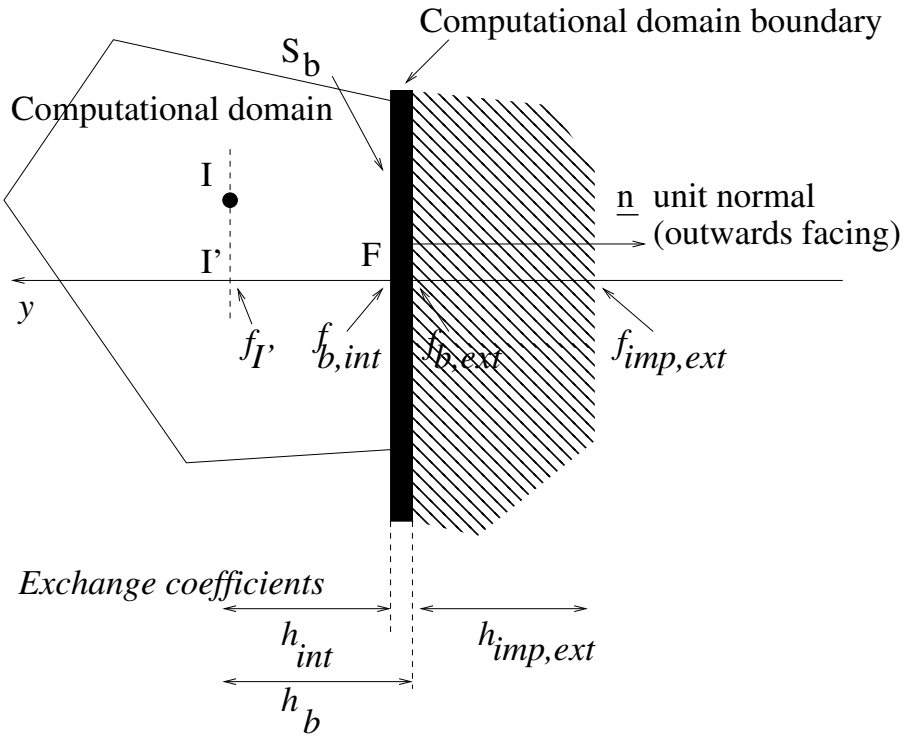


Figure I.5.1: Boundary cell.

Example 5.1 *The heat flux q_w from a solid wall to a laminar flow is defined and discretised as*

$$q_w = \lambda \underline{\nabla} T \cdot n_{c>f_b} = -\lambda \frac{T(\underline{x}'_c) - T(\underline{x}_f)}{\left|\underline{x}_f - \underline{x}'_c\right|} = -h_{int}\left(T(\underline{x}'_c) - T(\underline{x}_f)\right), \tag{I.5.3}$$

where $h_{int}/\left|\underline{x}_f - \underline{x}'_c\right|$ is the exchange coefficient between the wall and the fluid point $\underline{x}'_c$.

- *If the wall temperature T_w is known, $T(\underline{x}_f) = T_w$ is set as Dirichlet condition, q_w will be a result of the simulation, and the gradient and diffusive flux coefficients are as follows:*

$$\left\{\begin{array}{rcl} A^g_{f_b} &=& T_w, \\ B^g_{f_b} &=& 0, \end{array}\right. \qquad \left\{\begin{array}{rcl} A^f_{c>f_b} &=& -h_{int} T_w, \\ B^f_{c>f_b} &=& h_{int}. \end{array}\right. \tag{I.5.4}$$

- *If the wall heat flux q_w is known, this is a Neumann condition and $T(\underline{x}_f) = T_w$ will be a result of the simulation and the gradient and diffusive flux coefficients are as follows:*

$$\left\{\begin{array}{rcl} A^g_{f_b} &=& -\dfrac{q_w}{h_{int}}, \\ B^g_{f_b} &=& 1, \end{array}\right. \qquad \left\{\begin{array}{rcl} A^f_{c>f_b} &=& q_w, \\ B^f_{c>f_b} &=& 0. \end{array}\right. \tag{I.5.5}$$

5.2 Standard user boundary conditions

The user generally gives standard boundary conditions, which are:

Inlet: it corresponds to a Dirichlet boundary condition on all the transported variables (and should therefore be given by the user) and to a homogeneous Neumann on the pressure field.

Outlet: it correspond to a homogeneous Neumann boundary condition on all the transported variables. For the pressure field, a Dirichlet boundary condition which is expected to mimic $\frac{\partial^2 P}{\partial n \partial \tau} = 0$ for any vector $\underline{\tau}$ parallel to the outlet face (see §5.4.4 for more details). This condition means that the pressure profile does not vary in the normal direction of the outlet. Warning: if the outgoing mass-flux is negative, *i.e.* if the outlet becomes an inlet, then the mass-flux is clipped to zero. Moreover, since the pressure field is defined up to a constant, it is fixed to a reference pressure P_0 at an arbitrary chosen outlet boundary face. The user can choose an other desired face where a Dirichlet on pressure is prescribed.

Free inlet/outlet: it corresponds to the standard outlet when the flow is outgoing (see §5.4.4), but to a free inlet when the flow is ingoing. The Bernoulli relationship is used to derive a boundary condition on the pressure increment to couple velocity and pressure at these free inlet faces. Note that no clipping on the velocity is imposed. The same boundary conditions as for outlet on the other variables is imposed. For more details please refer to §5.4.5.

Walls: This particular treatment will be detailed in the following sections. For the velocity, the aim is to transform the Dirichlet boundary condition (the velocity at the wall is equal to zero, or the velocity of a moving wall) into a Neumann boundary condition where the wall shear stress is imposed function of the local flow velocity and the turbulence characteristics. A similar treatment using wall functions is done on every transported variable if this variable is prescribed. The boundary condition on the pressure field is a homogeneous Neumann by default, or alternatively an extrapolation of the gradient.

Symmetries: This condition corresponds to a homogeneous Neumann for the scalar fields (*e.g.* the pressure field or the temperature field). For vectors, such as the velocity, it corresponds to impose a zero Dirichlet on the component normal to the boundary, and a homogeneous Neumann on the tangential components. Thus, this condition couples the vector components if the symmetry faces are not aligned with the reference frame. The boundary condition for tensors, such as the Reynolds stresses, will be detailed in the following sections.

See the programmers reference of the dedicated subroutine for further details.

5.3 Internal coding of the boundary conditions – Discretization

As already mentioned, the boundary conditions set by the user for the variable Y are translated into two pairs of coefficients $\left(A^g_{f_b}, B^g_{f_b}\right)$ which are used by the gradient operator and by the advection operator, and $\left(A^f_{c>f_b}, B^f_{c>f_b}\right)$ for use by the diffusion operator, this for all the boundary faces f_b.

Let us first recall the general form of the transport equation of a variable Y, which could be a scalar, a vector or a tensor:

$$C\rho\frac{\partial Y}{\partial t} + C\underline{\nabla} Y \cdot (\rho \underline{u}) = \text{div}\,(K\,\underline{\nabla} Y) + ST_Y. \tag{I.5.6}$$

In the Equation (I.5.6) ρ is the density of the fluid, $(\rho\underline{u})$ the convective mass flux of the variable Y, K its conductivity or diffusivity and S any additional source terms. Note that K is the sum of molecular and turbulent diffusivity in case of RANS modelling with an eddy viscosity model. The dimension of K for different variables is displayed in Table 5.1. The value of C is 1 for all the variables except for the temperature where C is the specific heat C_p. If the variable Y is the variance of another scalar, then its diffusivity is deduced from the scalar itself.

5.3.1 Basic Dirichlet boundary conditions

Imposing a basic Dirichlet condition $Y^{imp}_{f_b}$ on a boundary face f_b is translated into:

Y			K		
symbol	name	units	symbol	name	units
u_i	velocity	$m.s^{-1}$	μ or $\mu+\mu_t$	dynamic viscosity	$kg.m^{-1}.s^{-1}$
P	pressure	$kg.m^{-1}.s^{-2}$	Δt	time step	s
T	temperature	K	λ	thermal conductivity	$kg.m.s^{-3}.K^{-1}$ $= W.m^{-1}.K^{-1}$
h	enthalpy	$m^2.s^{-2}$ $= J.kg^{-1}$	λ/C_p	thermal conductivity over specific heat	$kg.m^{-1}.s^{-1}$
Y	variable	unit of (Y)	K	conductivity or diffusivity	$kg.m^{-1}.s^{-1}$

Table 5.1: Definitions and units of K for the most common equations.

$$\left\{\begin{array}{lcl} A^g_{f_b} &=& Y^{imp}_{f_b}, \\ B^g_{f_b} &=& 0, \end{array}\right. \qquad \left\{\begin{array}{lcl} A^f_{c>f_b} &=& -h_{int} Y^{imp}_{f_b}, \\ B^f_{c>f_b} &=& h_{int}. \end{array}\right. \tag{I.5.7}$$

The term h_{int} is a internal coding coefficient (automatically provided by the code) similar to an exchange coefficient. Its value for particular variables is given in Table 5.2.

Remark 5.1 *$Y^{imp}_{f_b}$ must be specified by the user. The boundary type code is* 1 *(see Table 5.3).*

5.3.2 Neumann boundary conditions

Imposing a Neumann condition $D^{imp}_{c>f_b}$ on a boundary face f_b is translated into

$$\left\{\begin{array}{lcl} A^g_{f_b} &=& -\dfrac{D^{imp}_{c>f_b}}{h_{int}}, \\ B^g_{f_b} &=& 1, \end{array}\right. \qquad \left\{\begin{array}{lcl} A^f_{c>f_b} &=& D^{imp}_{c>f_b}, \\ B^f_{c>f_b} &=& 0. \end{array}\right. \tag{I.5.8}$$

Remark 5.2 *$D^{imp}_{c>f_b}$ must be specified by the user. The boundary type code is* 3 *(see Table 5.3).*

5.3.3 Mixed or Robin boundary conditions

As explained in the introduction 5.1, the simple case of a heat flux q_w from a solid wall reads: $q_w = \lambda \underline{\nabla} T \cdot \underline{n}_{c>f_b} = -\lambda \dfrac{T(\underline{x}'_c) - T(\underline{x}_f)}{|\underline{x}_f - \underline{x}'_c|} = -h_{int}\left(T(\underline{x}'_c) - T(\underline{x}_f)\right)$ where $h_{int} = \dfrac{\lambda}{|\underline{x}_f - \underline{x}'_c|}$ is displayed in Table 5.2. As presented in § 5.3.1, Dirichlet condition $T(\underline{x}_f) = T_w$ simply leads to $A^f = -h_{int} T_w$ and $B^f = h_{int}$. In some cases, heat transfer outside the flow domain is described by a one-dimensional conduction equation $q = -h_{ext}\left(T_{REF} - T_w\right)$ in which some reference temperature T_{REF} is given and the coefficient h_{ext} is known from an analytical or experimental correlation. Equaling this to the heat flux computed in the first fluid cell at the boundary gives $h_{ext}\left(T_{REF} - T_w\right) = h_{int}\left(T(\underline{x}'_c) - T(\underline{x}_f)\right)$. This relation between the variable and its gradient at the boundary is called a Robin or mixed boundary condition.

More generally, to impose an external Dirichlet $Y^{imp,\,ext}$ at some distance outside the boundary face,

this is done by a user prescribed external coefficient h_{ext} (see Figure I.5.1), and the boundary condition coefficients then read:

$$\left\{\begin{array}{lcl} A^g_{f_b} &=& \dfrac{h_{ext}}{h_{int}+h_{ext}} Y^{imp,\,ext}, \\ B^g_{f_b} &=& \dfrac{h_{int}}{h_{int}+h_{ext}}, \end{array}\right. \qquad \left\{\begin{array}{lcl} A^f_{c>f_b} &=& -h_{eq} Y^{imp,\,ext}, \\ B^f_{c>f_b} &=& h_{eq}, \end{array}\right. \tag{I.5.9}$$

where h_{eq} is defined by $h_{eq} = \dfrac{h_{int} h_{ext}}{h_{int}+h_{ext}}$. The harmonic mean (as in (I.4.30)) comes from summing of resistances instead of conductances. Note that this case reduces to Dirichlet condition if h_{ext} tends to the infinity.

Remark 5.3 *Both $Y^{imp,\,ext}$ and h_{ext} must be specified by the user. The boundary code is* 1 *(see Table 5.3). Take care that an outgoing flux is counted positively.*

5.3.4 Convective outlet boundary conditions

If the user wishes to impose a convective outlet (also called radiative outlet) condition which reads:

$$\frac{\partial Y}{\partial t} + C \frac{\partial Y}{\partial n} = 0, \tag{I.5.10}$$

where C denotes the convective celerity, then the internal coding reads:

$$\left\{\begin{array}{lcl} A^g_{f_b} &=& \dfrac{1}{1+CFL} Y^n_{f_b}, \\ B^g_{f_b} &=& \dfrac{CFL}{1+CFL}, \end{array}\right. \qquad \left\{\begin{array}{lcl} A^f_{c>f_b} &=& -\dfrac{h_{int}}{1+CFL} Y^n_{f_b}, \\ B^f_{c>f_b} &=& \dfrac{h_{int}}{1+CFL}, \end{array}\right. \tag{I.5.11}$$

where $CFL \equiv \dfrac{C\Delta t}{\left|\underline{x}_f - \underline{x}'_c\right|}$.

Remark 5.4 *Both C and $Y^n_{f_b}$ must be specified by the user, the boundary code is* 2 *(see Table 5.3).*

5.4 Wall boundary conditions

This section is dedicated to wall boundary conditions for the velocity, and any transported scalar (such as the temperature, the enthalpy, *etc.*) when a wall boundary value is prescribed and a wall function is set by the user. The dedicated boundary conditions of the turbulent variables (k, ε, R_{ij}) are detailed in Chapter 7. Here is a link to the programmers reference of the dedicated subroutine.

Simplified balance analysis proves that the fluid velocity profile or the Temperature profile are highly non-linear for turbulent flows in the vicinity of walls.

The aim of wall functions is to model the vicinity of the wall by substituting the Dirichlet condition on the fluid velocity and on the scalars such as the temperature ($\underline{u}_{f_b} = \underline{v}_{wall}$, $T_{f_b} = T_{wall}$ where $\underline{v}_{wall}$ is the velocity of the wall and T_{wall} is the temperature of the wall) by a Robin-type boundary condition linking the wall shear stress to the velocity of the fluid, and the thermal wall flux to the temperature within the first cell.

5.4.1 Velocity boundary condition for smooth walls and rough walls

We assume it is projected onto the tangent plane to the wall (if it is not, then the code projects it). The fluid velocity in the reference frame attached to the wall ("relative" velocity) projected to the wall writes $\underline{u}^r(\underline{x}'_c) \equiv \left(\underline{\underline{1}} - \underline{n} \otimes \underline{n}\right) \cdot \left(\underline{u}(\underline{x}'_c) - \underline{v}_{wall}\right)$.

Y			h_{int}	
symbol	name	units	homogeneous to	units
$\underline{u}$	velocity	$m.s^{-1}$	$\frac{\mu+\mu_t}{\underline{x}'_c\underline{x}_f}$	$kg.m^{-2}.s^{-1}$
P	pressure	$kg.m^{-1}.s^{-2}$	$\frac{\Delta t}{\underline{x}'_c\underline{x}_f}$	$s.m^{-1}$
T	temperature	K	$\frac{\lambda+C_p\mu_t/\sigma_t}{\underline{x}'_c\underline{x}_f}$	$kg.s^{-3}.K^{-1}$
				$W.m^{-2}.K^{-1}$
h	enthalpy	$m^2.s^{-2}$	$\frac{\lambda+C_p\mu_t/\sigma_t}{\underline{x}'_c\underline{x}_f}$	$kg.m^{-2}.s^{-1}$
		$J.kg^{-1}$		
Y	scalar	unit of(Y)	$\frac{K}{\underline{x}'_c\underline{x}_f}$	$kg.m^{-2}.s^{-1}$

Y			D^{imp}	
symbol	name	units	homogeneous to	units
$\underline{u}$	velocity	$m.s^{-1}$	$((\mu+\mu_t)\,\underline{\nabla u})\cdot\underline{n}$	$kg.m^{-1}.s^{-2}$
p	pressure	$kg.m^{-1}.s^{-2}$	$(\Delta t\underline{\nabla} P)\cdot\underline{n}$	$kg.m^{-2}.s^{-1}$
T	temperature	K	$((\lambda+C_p\mu_t/\sigma_t)\underline{\nabla} T)\cdot\underline{n}$	$kg.s^{-3}$
				$W.m^{-2}$
h	enthalpy	$m^2.s^{-2}$	$(\lambda/C_p+\mu_t/\sigma_t)\underline{\nabla} H)\cdot\underline{n}$	$kg.s^{-3}$
		$J.kg^{-1}$		$W.m^{-2}$
Y	scalar	unit of (Y)	$K\,\underline{\nabla} Y\cdot\underline{n}$	$kg.m^{-2}.s^{-1}$. unit of (Y)

Table 5.2: Definitions and units of h_{int} and D^{imp} for the most common equations.

The orthonormal coordinate system attached to the wall writes $\hat{\mathcal{R}} = (\underline{t}, -\underline{n}, \underline{b})$:

- $\underline{t} = \dfrac{\underline{u}^r_{\underline{x}'_c}}{\left|\underline{u}^r_{\underline{x}'_c}\right|}$ is the unit vector parallel to the projection of the relative velocity at $\underline{x}'_c$, $\underline{u}^r_{\underline{x}'_c}$, in the plane tangent to the wall (*i.e.* orthogonal to $\underline{n}$),
- $-\underline{n}$ is the unit vector orthogonal to the wall and directed towards the interior of the computational domain,
- $\underline{b}$ is the unit vector which completes the positively oriented coordinate system.

In this reference-frame the wall distance of the cell centre to the wall $\left|\underline{x}_f - \underline{x}'_c\right|$ is denoted by y.

The objective of wall functions is to increase the exchange coefficient to reflect the higher level of mixing effectively taking place in the near wall cell due to turbulence and due to effectively much sharper gradients of the computed variable than the linear profile assumed inside each cell by Finite Volumes discretisation. The end result (Cf (I.5.27)) will be to correct the laminar coefficient, $h_{int} = \lambda/y$, by a dimensionless analytical function (a "wall function") u^+ that represents the realistic non-linear profile. The correction is moreover proportional to the cell centre to wall dimensionless distance, y^+, over which a linear profile assumption may or may not be appropriate. These functions u^+ and y^+ that depends on the level of turbulent kinetic energy, are made dimensionless by:

- either the wall shear stress τ_{wall} only (this is called "one scale friction velocity" wall function),
- or by the wall shear stress τ_{wall} and the turbulent kinetic energy (this is called "two friction velocity scales").

When the mesh is made fine enough to capture the sharp variations of the variables near the wall, wall functions are no longer needed, variables are simply given Dirichlet values at the wall and the laminar exchange coefficient is correct. However the more basic versions of the turbulence models ("high Reynolds" $k-\varepsilon$ or Reynolds stress transport) will not be able to automatically "relaminarise" (*i.e.* reduce the turbulent mixing effect to zero at the correct rate of decay as y tends toward 0) and erroneous predictions will result (generally too much friction or heat transfer). When using a refined near wall mesh, "down to the wall" or "low Reynolds" versions of the turbulence models must be selected (*e.g.* v^2-f or elliptic blending models). However in this case one must make sure the mesh is fine enough.

To allow use of standard high Reynolds models whatever the mesh density, "scalable wall functions" can be activated. This consists in shifting slightly the wall outside of the mesh if the first cell lies in the viscous sub-layer. "Scalable wall function" only works with the "two friction velocity scales" version.

The wall functions are usually derived from Eddy Viscosity Models such as $k-\varepsilon$, thus the coming sections assume that $\underline{\underline{R}} = \dfrac{2}{3}k\underline{\underline{1}} - 2\nu_T\underline{\underline{S}}^D$.

One friction velocity scale

The simplified momentum balance in the first boundary cell reads:

$$(\mu + \mu_T)\,\frac{\partial u}{\partial y} = \tau_{wall}. \tag{I.5.12}$$

Let $u^\star$ be the friction velocity defined by:

$$u^\star \equiv \sqrt{\frac{\tau_{wall}}{\rho}}, \tag{I.5.13}$$

the equation (I.5.12) can be rewritten:

$$\left(1+\frac{\mu_T}{\mu}\right)\frac{\partial u^+}{\partial y^+}=1, \tag{I.5.14}$$

where $u^+ \equiv \left|\underline{u}^r_{\underline{x}'_c}\right| / u^\star$ and $y^+ \equiv y u^\star/\nu$.

The mixing length model states $\nu_T = L_m^2 S$, where the stain rate S reduces to $\frac{\partial u^r_{\underline{x}'_c}}{\partial y}$ in the wall vicinity. The mixing length is proportional to the wall distance according to the Prandtl model

$$L_m = \kappa y, \tag{I.5.15}$$

where $\kappa = 0.42$ is the Karman constant.

Two areas are therefore defined, one where $\mu_T/\mu \ll 1$ called the viscous sub-layer where μ_T/μ is neglected in (I.5.14) the velocity profile is found to be linear:

$$u^+ = y^+, \tag{I.5.16}$$

the other where $\frac{\mu_T}{\mu} \gg 1$ and the velocity profile becomes logarithmic:

$$u^+ = \frac{1}{\kappa}\ln\left(y^+\right) + C_{log}, \tag{I.5.17}$$

where $C_{log} = 5.2$.

(I.5.16) is valid for $y^+ < 5$ and (I.5.17) for $y^+ > 30$ so there is a gap (corresponding to the so called buffer layer) which more sophisticated models can cover, but they are not detailed here. Instead we introduce a dimensionless limit distance which crudely separates the viscous sub-layer from the logarithmic region and writes y^+_{lim}. Its value is $1/\kappa$ in general (to ensure the continuity of the velocity gradient) and 10.88 in LES (to ensure the continuity of the velocity).

The $u^\star$ is computed by iteratively solving(I.5.16) or (I.5.17):

$$\begin{array}{rcll} u^\star & = & \sqrt{\dfrac{\left|\underline{u}^r_{\underline{x}'_c}\right|\nu}{y}} & \text{if} \quad \dfrac{\left|y\,\underline{u}^r_{\underline{x}'_c}\right|}{\nu} < y^{+\,2}_{lim}, \\ \left\{\begin{array}{rcl} (u^\star)^0 & = & \exp\left(-\kappa C_{log}\right)\dfrac{\nu}{y} \\ (u^\star)^{q+1} & = & \dfrac{\kappa\left|\underline{u}^r_{\underline{x}'_c}\right| + (u^\star)^q}{\ln\left(\dfrac{y\,(u^\star)^q}{\nu}\right) + \kappa C_{log} + 1}. \end{array}\right. & \text{otherwise} \quad \text{solve iteratively in } q \end{array} \tag{I.5.18}$$

Therefore, the value of $\frac{y^+}{u^+}$ in the two layers reads:

$$\left\{\begin{array}{rcll} \dfrac{y^+}{u^+} & = & 1 & \text{if } y^+ < y^+_{lim}, \\ \dfrac{y^+}{u^+} & = & \dfrac{y^+}{\dfrac{1}{\kappa}\ln\left(y^+\right) + C_{log}} & \text{otherwise.} \end{array}\right. \tag{I.5.19}$$

Two friction velocity scales

The simplified momentum balance (I.5.12) still holds, but is non-dimensioned not only by the wall shear stress but also the turbulent kinetic energy. More precisely, let $u_k = \sqrt{\sqrt{C_\mu}k}$ be the friction velocity based on the turbulent kinetic energy in the first cell.

This definition is in fact valid only for high-y^+ values and a blending is performed in case of the intensity of turbulence is very low:

$$u_k \equiv \sqrt{g\frac{\nu\left|\underline{u}^r_{\underline{x}'_c}\right|}{y} + (1-g)\sqrt{C_\mu}k}, \tag{I.5.20}$$

where g is a blending factor defined by $g = \exp\left(-\frac{\sqrt{k}y}{11\nu}\right)$ (see [**?**]).

The friction velocity $u^\star$ is now defined by:

$$u^\star \equiv \frac{\tau_{wall}}{\rho u_k}. \tag{I.5.21}$$

Equation (I.5.12) can now be rewritten:

$$\left(1+\frac{\mu_T}{\mu}\right)\frac{\partial u^+}{\partial y_k^+} = 1, \tag{I.5.22}$$

where $u^+ \equiv \frac{\left|\underline{u}^r_{\underline{x}'_c}\right|}{u^\star}$ but $y_k^+ \equiv \frac{y u_k}{\nu}$.

Let us now remark that $\nu_T = C_\mu^{\frac{1}{4}}\sqrt{k}C_\mu^{\frac{3}{4}}\frac{k^{\frac{3}{2}}}{\epsilon} = u_k L_m$ where $L_m = C_\mu^{\frac{3}{4}}\frac{k^{\frac{3}{2}}}{\epsilon}$ is the integral length scale. Again using the Prandtl model (I.5.15), the following equation is obtained:

$$\left(1+\kappa y_k^+\right)\frac{\partial u^+}{\partial y_k^+} = 1. \tag{I.5.23}$$

The two areas defined in the previous section (I.5.19) that are the viscous sub-layer and the logarithmic layer still hold, but with y_k^+ instead of y^+:

$$\begin{cases} \dfrac{y_k^+}{u^+} = 1 & \text{if } y_k^+ < y_{lim}^+, \\ \dfrac{y_k^+}{u^+} = \dfrac{y_k^+}{\frac{1}{\kappa}\ln\left(y_k^+\right)+C_{log}} & \text{otherwise.} \end{cases} \tag{I.5.24}$$

Rough wall functions

When specifying a rough wall function with z_0 as roughness, the value of $\frac{y^+}{u^+}$ reads:

$$\frac{y^+}{u^+} = \frac{y^+}{\frac{1}{\kappa}\ln\left(\frac{y+z_0}{z_0}\right)}. \tag{I.5.25}$$

Wall shear stress with wall functions

The previous sections can be summarized as follows:

$$\underline{\underline{\tau}}\cdot\underline{n} = \frac{\mu}{\left|\underline{x}_f - \underline{x}'_c\right|}\frac{y^+}{u^+}\left(\underline{\underline{1}} - \underline{n}\otimes\underline{n}\right)\cdot\left(\underline{u}(\underline{x}'_c) - \underline{u}_{wall}\right) \tag{I.5.26}$$

where the rescaling factor $\frac{y^+}{u^+}$ depends on the wall function and is given in (I.5.19) or (I.5.24). Note that the wall shear stress is therefore parallel to the wall and fully implicit if the θ-scheme on the velocity is implicit.

The internal pair of boundary condition coefficients for the diffusive term for the velocity are

$$\begin{cases} \underline{A}^f_{c>f_b} &= -h_{fluid}\left(\underline{\underline{1}} - \underline{n}\otimes\underline{n}\right)\cdot\underline{u}_{wall} - h_{int}\left(\underline{n}\cdot\underline{u}_{wall}\right)\underline{n}, \\ \underline{\underline{B}}^f_{c>f_b} &= h_{fluid}\left(\underline{\underline{1}} - \underline{n}\otimes\underline{n}\right) + h_{int}\underline{n}\otimes\underline{n}, \end{cases} \tag{I.5.27}$$

where

$$h_{fluid} = \frac{\mu}{\left|\underline{x}_f - \underline{x}'_c\right|}\frac{y^+}{u^+}. \tag{I.5.28}$$

Velocity gradient boundary condition with wall functions

The gradient of the fluid velocity is of great importance for turbulence models as it appears in the production term. Moreover the profile of dissipation is even less linear than u^+ (it is hyperbolic) and would also require a wall function. A common correction in open literature is to directly modify these source term in the near wall cell volume integral. However this deeply modifies the code structure and the linear system solver. In order to implement this while still remaining within the standard boundary condition structure, ther current code version follows the suggestion of [Boucker and Mattéi, 2000] which consists in prescribing a slip velocity at the wall so that the velocity gradient evaluated inside the cell by the standard Finite Volumes linear profile is more consistent with wall function.

A dedicated boundary condition for the velocity gradient is therefore derived in presence of wall function to be consistent with the logarithmic law giving the ideal tangential velocity profile.

On the first hand, the theoretical gradient is:

$$\left(\frac{\partial u^r}{\partial y}\right)^{\text{theo}}_{I'} = \frac{u^\star}{\kappa\, y} \tag{I.5.29}$$

On the other hand, the normal finite volumes gradient of the tangential velocity reduces in the case of a regular orthogonal mesh (see notations in Figure I.5.2):

$$\left(\frac{\partial u^r}{\partial y}\right)^{\text{calc}}_{\underline{x}'_c} = \frac{u^r_G - u^r_F}{2y} = \frac{\dfrac{u^r_{\underline{x}_c} + u^r_{\underline{x}_{\overline{c}}}}{2} - u^r_{\underline{x}_f}}{2y} \tag{I.5.30}$$

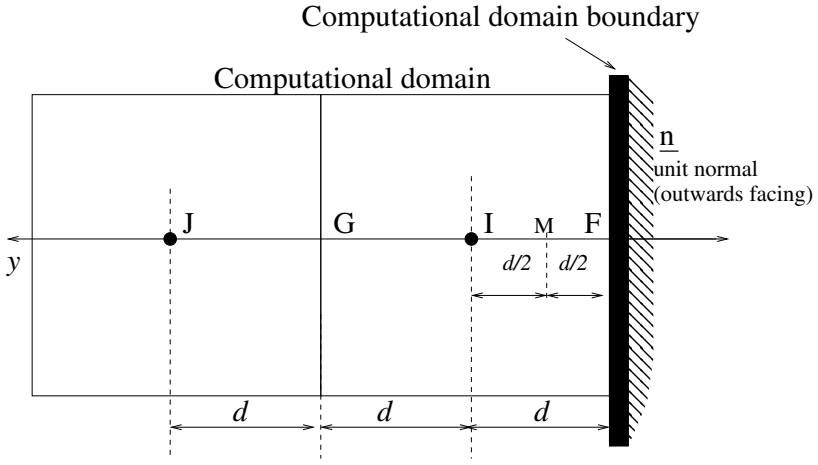


Figure I.5.2: Boundary cell - Orthogonal mesh.

We then assume that $u^r_{\underline{x}_{\overline{c}}}$ can be obtained from $u^r_{\underline{x}_c}$ and from the gradient of u^r calculated in G from the logarithmic law $u^r_{\underline{x}_{\overline{c}}} = u^r_{\underline{x}_c} + 2y\dfrac{u^\star}{\kappa\, 2y}$ and we thus obtain equalizing (I.5.29) and (I.5.30) (the formula is extended without any precaution to non-orthogonal meshes):

$$u^r_{\underline{x}_f} = u^r_{\underline{x}'_c} - \frac{3u^\star}{2\kappa} \tag{I.5.31}$$

The normal derivative at the wall is prescribed to zero. If the value obtained for y^+ at the wall is lower than y^+_{lim}, a no-slip condition is prescribed. Finally, one can also make explicit the velocity of the wall in the final expression: The internal pair of boundary condition coefficients for the diffusive term for the velocity are

$$\left\{\begin{array}{rcl} \underline{A}^g_{c>f_b} & = & (1-cofimp)\left(\underline{\underline{1}} - \underline{n}\otimes\underline{n}\right)\cdot\underline{u}_{wall} + \left(\underline{u}_{wall}\cdot\underline{n}\right)\underline{n}, \\ \underline{\underline{B}}^g_{c>f_b} & = & cofimp\left(\underline{\underline{1}} - \underline{n}\otimes\underline{n}\right), \end{array}\right. \tag{I.5.32}$$

where $cofimp = 1 - \dfrac{3}{2\kappa u^+}$ if $y^+ > y^+_{lim}$, and $cofimp = 0$ otherwise.

5.4.2 Scalar boundary condition for smooth walls

In this section, the wall functions applied to transported scalars Y (such as the temperature T for instance) are described. The reference "Convection Heat Transfer", Vedat S. Arpaci and Poul S. Larsen, Prentice-Hall, Inc was used. The approach is really similar to what is performed on the fluid velocity:

- if the wall value Y_{wall} is imposed, the Dirichlet boundary condition is substituted by a Robin-type one where the diffusive flux is function of the cell centre scalar value,
- if a Neumann value is imposed (that is to say that the user has prescribed the flux of Y), then the wall function is used to evaluate the corresponding wall value Y_{wall} which is displayed by the code.

Let us recall that $q_{c>f_b}\left(K_{f_b}, Y\right)$ is the wall diffusive flux for the scalar Y, and the simplified scalar balance reads:

$$q_{c>f_b}\left(K_{f_b}, Y\right) = -\left(K_{f_b} + C\,\frac{\mu_T}{\sigma_T}\right)\frac{\partial Y}{\partial y} \tag{I.5.33}$$

Remark 5.5 *the flux is positive if it enters the fluid domain, as shown by the orientation of the y axis*

The following considerations are presented using the general notations. In particular, the Prandtl-Schmidt number writes $\sigma = \dfrac{\mu\, C}{K}$. When the considered scalar Y is the temperature T, we have

- $C = C_p$ (specific heat capacity),
- $K = \lambda$ (molecular conductivity),
- hence, $\sigma = \dfrac{\mu\, C_p}{\lambda} = Pr$ (Prandtl number),
- $\sigma_T = Pr_T$ (turbulent Prandtl number),
- $q_{c>f_b} = \dfrac{D_{c>f_b}\left(\lambda_{f_b}, T\right)}{|\underline{S}|_{f_b}} = \left(\lambda + \dfrac{C_p\mu_T}{\sigma_T}\right)\dfrac{\partial T}{\partial y}$ (flux in $W.m^{-2}$).

In order to make (I.5.33) dimensionless, we introduce $Y^\star$ defined using the flux at the boundary $q_{c>f_b}$:

$$Y^\star \equiv -\frac{q_{c>f_b}}{\rho\, C\, u_k} \tag{I.5.34}$$

For the temperature, we thus have:

$$T^\star \equiv -\frac{q_{c>f_b}}{\rho\, C_p\, u_k} \tag{I.5.35}$$

We then divide both sides of equation (I.5.33) by $q_{c>f_b}$ and after some algebrae it comes:

$$1 = \left(\frac{1}{\sigma} + \frac{1}{\sigma_T}\frac{\nu_T}{\nu}\right)\frac{\partial Y^+}{\partial y^+} \tag{I.5.36}$$

where

$$\nu = \frac{\mu}{\rho} \qquad \nu_T = \frac{\mu_T}{\rho} \qquad y^+ = \frac{y\, u_k}{\nu} \qquad Y^+ = \frac{Y - Y_{wall}}{Y^\star} \tag{I.5.37}$$

5.4.3 The three layers wall function of Arpaci and Larsen

In order to compute the theoretical scalar profile, we integrate equation (I.5.36) to obtain $Y^+_{\underline{x}'_c}$. The only difficulty then consists in prescribing a variation law of $\mathcal{K} = \frac{1}{\sigma} + \frac{1}{\sigma_T}\frac{\nu_T}{\nu}$.

In the fully developed turbulent region (far enough from the wall, for $y^+ \geqslant y_2^+$), a mixing length hypothesis models the variations of ν_T:

$$\nu_T = \kappa\, y\, u_k \tag{I.5.38}$$

Additionally, the molecular diffusion of Y (or the conduction when Y represents the temperature) is negligible compared to its turbulent diffusion: therefore we neglect $\frac{1}{\sigma}$ compared to $\frac{1}{\sigma_T}\frac{\nu_T}{\nu}$. Finally we have:

$$\mathcal{K} = \frac{\kappa\, y^+}{\sigma_T} \tag{I.5.39}$$

On the contrary, in the near-wall region (for $y^+ < y_1^+$) the turbulent contribution becomes negligible compared to the molecular contribution and we thus neglect $\frac{1}{\sigma_T}\frac{\nu_T}{\nu}$ compared to $\frac{1}{\sigma}$.

It would be possible to restrict ourselves to these two regions, but Arpaci and Larsen suggest the model can be improved by introducing an intermediate region ($y_1^+ \leqslant y^+ < y_2^+$) in which the following hypothesis is made:

$$\frac{\nu_T}{\nu} = a_1 \left(y^+\right)^3 \tag{I.5.40}$$

where a_1 is a constant whose value is obtained from experimental correlations:

$$a_1 = \frac{\sigma_T}{1000} \tag{I.5.41}$$

Thus the following model is used for $\mathcal{K}$ (see a sketch in Figure I.5.3):

$$\mathcal{K} = \begin{cases} \dfrac{1}{\sigma} & \text{if } y^+ < y_1^+ \\ \dfrac{1}{\sigma} + \dfrac{a_1 (y^+)^3}{\sigma_T} & \text{if } y_1^+ \leqslant y^+ < y_2^+ \\ \dfrac{\kappa\, y^+}{\sigma_T} & \text{if } y_2^+ \leqslant y^+ \end{cases} \tag{I.5.42}$$

The values of y_1^+ and y_2^+ are obtained by calculating the intersection points of the variations laws used for $\mathcal{K}$.

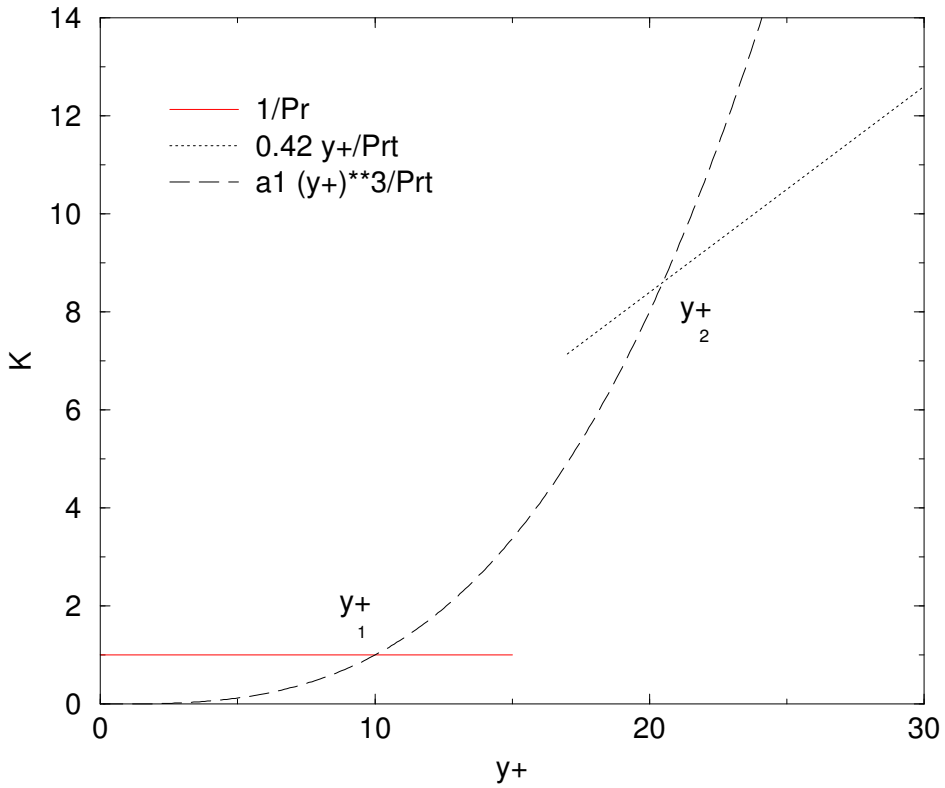


Figure I.5.3: $\frac{1}{\sigma} + \frac{1}{\sigma_T}\frac{\nu_T}{\nu}$ as a function of y^+ obtained for $\sigma = 1$ and $\sigma_T = 1$.

The existence of an intermediate region depends upon the values of σ. Let's first consider the case where σ cannot be neglected compared to 1. In practise we consider $\sigma > 0,1$ (this is the common case when scalar Y represents the air or the water temperature in normal temperature and pressure conditions). It is assumed that $\dfrac{1}{\sigma}$ can be neglected compared to $\dfrac{a_1 (y^+)^3}{\sigma_T}$ in the intermediate region. We thus obtain:

$$y_1^+ = \left(\frac{1000}{\sigma}\right)^{\frac{1}{3}} \qquad y_2^+ = \sqrt{\frac{1000\kappa}{\sigma_T}} \tag{I.5.43}$$

The dimensionless equation (I.5.36) is integrated under the same hypothesis and we obtain the law of $\dfrac{y^+}{Y^+}$:

$$\begin{cases} \dfrac{y^+}{Y^+} = \dfrac{1}{\sigma} & \text{if } y^+ < y_1^+ \\ \dfrac{y^+}{Y^+} = \dfrac{y^+}{a_2 - \dfrac{\sigma_T}{2\,a_1\,(y^+)^2}} & \text{if } y_1^+ \leqslant y^+ < y_2^+ \\ \dfrac{y^+}{Y^+} = \dfrac{y^+}{\dfrac{\sigma_T}{\kappa}\,\ln(y^+) + a_3} & \text{if } y_2^+ \leqslant y^+ \end{cases} \tag{I.5.44}$$

where a_2 and a_3 are integration constants, which have been chosen to obtain a continuous profile of Y^+:

$$a_2 = 15\sigma^{\frac{2}{3}} \qquad a_3 = 15\sigma^{\frac{2}{3}} - \frac{\sigma_T}{2\kappa}\left(1 + \ln\left(\frac{1000\kappa}{\sigma_T}\right)\right) \tag{I.5.45}$$

Let's now study the case where σ is much smaller than 1. In practise it is assumed that $\sigma \leqslant 0.1$ (this is for instance the case for liquid metals whose thermal conductivity is very large, and who have Prandtl

number of values of the order of 0.01). The intermediate region then disappears and the coordinate of the interface between the law used in the near-wall region and the one used away from the wall is given by:

$$y_0^+ = \frac{\sigma_T}{\kappa\sigma} \tag{I.5.46}$$

The dimensionless equation (I.5.36) is then integrated under the same hypothesis, and the law of Y^+ is obtained:

$$\begin{cases} \dfrac{y^+}{Y^+} = \dfrac{1}{\sigma} & \text{if } y^+ \leqslant y_0^+ \\ \dfrac{y^+}{Y^+} = \dfrac{y^+}{\dfrac{\sigma_t}{\kappa}\ln\left(\dfrac{y^+}{y_0^+}\right) + \sigma\, y_0^+} & \text{if } y_0^+ < y^+ \end{cases} \tag{I.5.47}$$

Wall diffusive flux with wall functions

The previous sections can be summarized as follows:

$$q_{c>f_b} = -\frac{C_\mu}{|\underline{x}_f - \underline{x}'_c|}\frac{y^+}{Y^+}\left(Y(\underline{x}'_c) - Y_{wall}\right) \tag{I.5.48}$$

where the rescaling factor $\frac{y^+}{Y^+}$ depends on the wall function and is given in (I.5.44) if $\sigma > 0.1$ or (I.5.47) if $\sigma \leq 0.1$.

The internal pair of boundary condition coefficient for the diffusive term for the variable Y are

$$\begin{cases} A^f_{c>f_b} &= -h_{fluid} Y_{wall}, \\ B^f_{c>f_b} &= h_{fluid}, \end{cases} \tag{I.5.49}$$

where $h_{fluid} = \dfrac{C_\mu}{|\underline{x}_f - \underline{x}'_c|}\dfrac{y^+}{Y^+}$.

5.4.4 Outlet boundary condition on the pressure

In this section the boundary condition on the pressure at the outlet is detailed. Some assumptions are made to derive this boundary condition which consists of a Dirichlet (combined with a homogeneous Neumann on the velocity) based on the pressure field at the previous time step. On basic configurations such as a channel or a pipe where the outlet is orthogonal to the flow, the shape of the pressure profile in a surface parallel to the outlet is approximately the shape of the pressure profile at the outlet. This hypothesis is valid for established flows, far from any perturbation. In this configuration one can write:

$$\frac{\partial^2 P}{\partial n \partial \tau} = 0,$$

where $\underline{n}$ is the outward normal vector and $\underline{\tau}$ is any vector in the boundary face.

Then remark that the value at the boundary face f_b is linked to the pressure value in $\underline{x}'_c$ by the relationship:

$$P_{f_b} = P_{\underline{x}'_c} + \underline{\nabla}_c P \cdot \underline{x}'_c\underline{x}_f.$$

If moreover we assume that the pressure gradient in the normal direction is uniform, and that all the $\underline{x}'_c$ related to the outlet faces are on a single plane parallel to the outlet, then the value of $\underline{\nabla}_c P \cdot \underline{x}'_c\underline{x}_f$ is constant for all the faces and denoted R.

Furthermore, the pressure is defined up to a constant, so the code chooses to fix the pressure at P_0 to a given outlet boundary face ${f_b}^{imp}$. Therefore the pressure field is shifted by the constant $R_0 = P_0 - P_{f_b}^{imp} = P_0 - \left(P_{\underline{x}'_c}^{imp} + R\right)$.

All that together gives

$$\begin{aligned} P_{f_b} &= P_{\underline{x}'_c} + R + R_0, \\ &= P_{\underline{x}'_c} + R + P_0 - \left(P_{\underline{x}'_c}^{imp} + R\right), \\ &= P_{\underline{x}'_c} + \underbrace{P_0 - P_{\underline{x}'_c}^{imp}}_{\text{constant denoted by } \widetilde{R}}, \\ &= P_{\underline{x}'_c} + \widetilde{R}. \end{aligned} \tag{I.5.50}$$

To conclude, the outlet boundary condition on the pressure is a Dirichlet based on the value in $\underline{x}'_c$ (at the previous time step) and shifted to impose the value P_0 at a given face f_b.

5.4.5 Free inlet/outlet boundary condition on the pressure increment

In this section the boundary condition on the pressure increment at the free inlet is detailed. For the other variables (even the pressure itself), the treatment is similar to §5.4.4.

Let us start with the Bernoulli relationship between a boundary face f linked to a point on the same stream line far away from the computational domain:

$$P_f - \rho_f \underline{g} \cdot \left(\underline{x}_f - \underline{x}_0\right) + \frac{1+K}{2}\rho_f \underline{u}_f \cdot \underline{u}_f = P_\infty - \rho_\infty \underline{g} \cdot \left(\underline{x}_\infty - \underline{x}_0\right) + \frac{1}{2}\rho_\infty \underline{u}_\infty \cdot \underline{u}_\infty, \tag{I.5.51}$$

where K is a possible head loss of the fluid between the infinity and the boundary face entrance (which the user may play with to model the non-computed domain).

Assuming that the stream line is horizontal, that the density is constant over the stream line and that the fluid velocity is zero at the infinity, the Equation (I.5.51) becomes:

$$P_f^\star = -\frac{1+K}{2}\rho_f \underline{u}_f \cdot \underline{u}_f, \tag{I.5.52}$$

where we recall that $P^\star$ is the dynamic pressure.

For stability reasons, the Equation (I.5.52) is implicit in time as follows:

$$P_f^{\star,n+1} = -\frac{1+K}{2}\rho_f \underline{u}_f^n \cdot \underline{u}_f^{n+1}. \tag{I.5.53}$$

This implicitation is crucial to make the calculations stable.

The prediction-correction velocity-pressure coupling algorithm requires boundary conditions on the pressure increment (computed in the correction step), and therefore relation (I.5.53) is derived to obtain boundary conditions on the pressure increment. Recall that the correction step reads:

$$\frac{(\rho \underline{u})^{n+1} - (\rho^n \widetilde{\underline{u}})}{\Delta t} = -\underline{\nabla} \delta P, \tag{I.5.54}$$

the pressure increment $\delta P = P^{\star,n+1} - P^{\star,n}$ at the face is consequently given by:

$$\delta P_f = -P_f^{\star,n} - \frac{1+K}{2}\rho_f \underline{u}_f^n \cdot \left(\widetilde{\underline{u}} - \frac{\Delta t}{\rho}\underline{\nabla}\delta P\right)_f. \tag{I.5.55}$$

Variable		Admissible Values						
Velocity	$\underline{u}$	1	2	3	4	5	6	9
Pressure	P	1	2	3				
Scalar turbulent variables	k, ε, φ, $\bar{f}$, ω	1	2	3		5	6	
Reynolds stresses	R_{ij}	1	2	3	4	5	6	
Y (except variances)	Y	1	2	3		5	6	
Variance of a variable Y		1	2	3				

Table 5.3: Admissible values of boundary conditions for all variables.

Neglecting the tangential velocity component at the entering faces ($(\rho\underline{u})_f^n \simeq (\rho\underline{u})_f^n \cdot \underline{n}\,\underline{n} = \dfrac{\dot{m}_f}{|S|_f}\underline{n}$) and approximating $P_f^{\star,n}$ with its cell-center value, Equation I.5.55 reads:

$$\delta P_f = -P_c^{\star,n} - \frac{1+K}{2}\frac{\dot{m}_f}{|S|_f}\left(\widetilde{\underline{u}}\cdot\underline{n} - \frac{\Delta t}{\rho}\frac{\partial \delta P}{\partial n}\right)_f. \tag{I.5.56}$$

Noting that $\left.\dfrac{\partial \delta P}{\partial n}\right|_f = \dfrac{\delta P_f - \delta P_c}{\left|\underline{x}_f - \underline{x}_c'\right|}$, (I.5.56) is rewritten as:

$$\delta P_f = A_{\delta P}^g + B_{\delta P}^g \delta P_c, \tag{I.5.57}$$

with:

$$\begin{aligned} A_{\delta P}^g &= \frac{1}{1+CFL}\left(-P_c^{\star,n} - \frac{1+K}{2}\frac{\dot{m}_f}{|S|_f}\widetilde{\underline{u}}_f\cdot\underline{n}\right), \\ B_{\delta P}^g &= \frac{CFL}{1+CFL}, \end{aligned} \tag{I.5.58}$$

where the CFL is defined as:

$$CFL = -\frac{1+K}{2}\frac{\Delta t \dot{m}_f}{\rho_f |S|_f}. \tag{I.5.59}$$

Remark 5.6 *CFL is a positive quantity for ingoing flows, which ensures stability.*

Remark 5.7 *The formulation (I.5.59) is nothing else but a convective outlet on the pressure increment (see §5.3.4).*

Checking step

Before computing the pairs of boundary condition coefficients, a step of checking is performed. Basically, the code checks that the user has given a boundary condition to all boundary faces, and that the setting between all the variables is compatible (see Table 5.3).

Chapter 6

Algebrae

6.1 Iterative process

See § F.

6.2 Linear algebrae

See § K.

Part II

Advanced modelling

Chapter 7

Turbulence modelling

7.1 Eddy viscosity Models (EVM)

In this section eddy viscosity hypothesis is made which states that the Reynolds stress tensor is aligned with the rate of strain $\underline{\underline{S}}$:

$$\rho\underline{\underline{R}} = \frac{2}{3}\rho k\underline{\underline{1}} - 2\mu_T\underline{\underline{S}}^D \tag{II.7.1}$$

7.1.1 Equations for the variables k and ε (standard $k-\varepsilon$ model)

$$\left\{\begin{array}{rcl} \rho\dfrac{\partial k}{\partial t} + \underline{\nabla}k\cdot(\rho\underline{u}) - \mathrm{div}\left[\left(\mu+\dfrac{\mu_t}{\sigma_k}\right)\underline{\nabla}k\right] &=& \mathcal{P}+\mathcal{G}-\rho\varepsilon+\Gamma k^{in}+ST_k, \\ \rho\dfrac{\partial \varepsilon}{\partial t} + \underline{\nabla}\varepsilon\cdot(\rho\underline{u}) - \mathrm{div}\left[\left(\mu+\dfrac{\mu_t}{\sigma_\varepsilon}\right)\underline{\nabla}\varepsilon\right] &=& C_{\varepsilon_1}\dfrac{\varepsilon}{k}\left[\mathcal{P}+(1-C_{\varepsilon_3})\mathcal{G}\right]-\rho C_{\varepsilon_2}\dfrac{\varepsilon^2}{k}+\Gamma\varepsilon^{in}+ST_\varepsilon, \end{array}\right. \tag{II.7.2}$$

where $\mathcal{P}$ is the production term created by mean shear:

$$\begin{array}{rcl} \mathcal{P} &=& -\rho\underline{\underline{R}}:\underline{\underline{\nabla}}\,\underline{u} = -\left[-2\mu_T\underline{\underline{S}}^D+\dfrac{2}{3}\rho k\underline{\underline{1}}\right]:\underline{\underline{S}}, \\ &=& 2\mu_T\underline{\underline{S}}^D:\underline{\underline{S}}^D-\dfrac{2}{3}\rho k tr\left(\underline{\underline{\nabla}}\,\underline{u}\right), \end{array} \tag{II.7.3}$$

and $\mathcal{G}$ is the production term created by gravity effects:

$$\mathcal{G} = \frac{1}{\rho}\frac{\mu_T}{\sigma_t}\underline{\nabla}\rho\cdot\underline{g}. \tag{II.7.4}$$

The dynamic turbulent viscosity reads:

$$\mu_T = \rho C_\mu\frac{k^2}{\varepsilon}. \tag{II.7.5}$$

ST_k and ST_ε stand for the additional source terms prescribed by the user (in rare cases only).

The constants of the model are given in the Table (7.1):

C_μ	C_{ε_1}	C_{ε_2}	σ_k	σ_ε
0.09	1.44	1.92	1.0	1.3

Table 7.1: Standard $k-\varepsilon$ model constants [Launder and Spalding, 1974].

$C_{\varepsilon_3}=0$ if $\mathcal{G}\geqslant 0$ (unstable stratification) and $C_{\varepsilon_3}=1$ if $\mathcal{G}\leqslant 0$ (stable stratification).

See the programmers reference of the dedicated subroutine for further details.

7.1.2 $k-\epsilon-\overline{v^2}/k$ elliptic blending turbulence model

The BL-$\overline{v^2}/k$ [Billard and Laurence, 2012] is a elliptic-blending based $\overline{v^2}-f$ model. It is a low Reynolds number model and as such the wall distance of the first off-wall cell centre must be of order of unity when expressed in viscous unit.

The following gives details about the model followed by some description of its implementation into code_saturne.

Model description

This eddy viscosity model solves for k and ε as turbulence variables, representing respectively the turbulent kinetic energy and its dissipation rate, as well as two non-dimensional variables, $\varphi = \overline{v^2}/k$ and α. The first of these latter two represents the ratio of wall normal Reynolds stress to turbulent kinetic energy (thus being a measure of the near-wall turbulence anisotropy) and the second is a wall proximity sensitive quantity (*i.e.* it takes the value 0 at a wall and 1 in the far field). The coefficient α is solved for via an elliptic equation (L representing the turbulence length-scale):

$$\alpha - L^2 \Delta\alpha = 1 \tag{II.7.6}$$

The φ transport equation reads:

$$\frac{\mathrm{d}\varphi}{\mathrm{d}t} = \left(1-\alpha^3\right) f_w + \alpha^3 f_h - P\frac{\varphi}{k} + \frac{2}{k}\frac{\nu_t}{\sigma_k}\underline{\nabla}\varphi\cdot\underline{\nabla}k + \mathrm{div}\left[\left(\frac{\nu}{2}+\frac{\nu_t}{\sigma_\varphi}\right)\underline{\nabla}\varphi\right] \tag{II.7.7}$$

The aim of the BL-$\overline{v^2}/k$ model is to stand as a code-friendly version of the $\overline{v^2}-f$ model of [Durbin, 1991]. In both the wall normal stress $\overline{v^2}$ is used in the ν_T definition to correctly represent the near-wall turbulence damping (T is the turbulence time-scale and $C_\mu = 0.22$ is calibrated in the logarithmic layer of a channel flow):

$$\nu_T = C_\mu \varphi k\, T \tag{II.7.8}$$

The elliptic blending approach mainly allows for an improved robustness. Indeed, the original $\overline{v^2}-f$ approach solves for the quantity $\overline{v^2}$ and a variable f derived from the wall normal pressure term[1] and defined as:

$$f = \frac{1}{k}\left[\underbrace{-\frac{2}{\rho}\overline{v\partial_y p}}_{\phi^*_{22}} - 2\nu\overline{\underline{\nabla}v\cdot\underline{\nabla}v} + \varepsilon\frac{\overline{v^2}}{k}\right] \tag{II.7.9}$$

with f being solved using an elliptic equation:

$$f - L^2\Delta f = f_h \tag{II.7.10}$$

Similarly to the α equation, this elliptic operator allows to represent the non-local effects induced by the incompressibility of turbulence. The quantity f_h is obtained by considering homogeneous modelling (*i.e.* $\Delta f = 0$) of f using for pressure strain-rate term the model of [Launder et al., 1975]. The correct asymptotic behaviour of the variable $\overline{v^2}$ is ensured by the following boundary condition:

$$\lim_{y\to 0} f = \lim_{y\to 0}\frac{-20\nu^2\overline{v^2}}{\varepsilon y^4} \tag{II.7.11}$$

This requires a balance between $O(y^4)$ terms which proves to be numerically problematic. In the BL-$\overline{v^2}/k$ model the elliptic equation is simply solved for a non-dimensional quantity with an homogeneous Dirichlet boundary condition, therefore alleviating the stiffness associated to the boundary condition of the elliptic variable. The inclusion of α in the definition of f allows a blending between the near-wall and the homogeneous form $f = (1-\alpha^3)f_w + \alpha^3 f_h$ in the φ equation. For the f_h model the proposal of [Speziale et al., 1991] is preferred for its better reproduction of the pressure term in a boundary layer.

The model also solves a k-ε system somewhat modified compared to the one generally adopted by $\overline{v^2}-f$ models. The k and ε equations adopted by the BL-$\overline{v^2}/k$ model read:

$$\frac{\mathrm{d}k}{\mathrm{d}t} = P - \varepsilon + \mathrm{div}\left[\left(\frac{\nu}{2}+\frac{\nu_t}{\sigma_k}\right)\underline{\nabla}k\right] - C_{\varepsilon 3}(1-\alpha)^3\frac{k}{\varepsilon}2\nu\nu_t\left(\partial_k\partial_j U_i\right)\left(\partial_k\partial_j U_i\right) \tag{II.7.12}$$

[1] the pressure term ϕ^*_{22} is not decomposed but modelled as a whole

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = \frac{C_{\varepsilon 1}P - C^*_{\varepsilon 2}\varepsilon}{T} + \mathrm{div}\left[\left(\frac{\nu}{2} + \frac{\nu_t}{\sigma_\varepsilon}\right)\underline{\nabla}\varepsilon\right] \tag{II.7.13}$$

The k equation includes the so-called "E term" dependent on the second velocity derivatives squared, similar to that introduced into the ε equation by [Jones and Launder, 1972], and the homogeneous part of the dissipation rate is independently accounted for (following the suggested formulation of [Jakirlic and Hanjalic, 2002]). This implies that the quantity ε resolved by the model has a different definition to that conventionally employed in k-ε schemes (*i.e.* a change of variable $\varepsilon \to \varepsilon + (1-\alpha)^3 \frac{k}{\varepsilon} E + \frac{1}{2}\nu\partial_{jj}k$). This has the beneficial effect of reducing the Reynolds number dependence of the near-wall value of the turbulence variables and of the time and length scales, T and L respectively, yielding better near-wall prediction of the blending variable and the turbulent viscosity, and hence mean flow quantities, for both low and high Reynolds numbers.

A further feature is that the coefficient $C^*_{\varepsilon 2}$ is taken as a function of the turbulent transport of k to ε ratio (as proposed by [Parneix et al., 1996]):

$$C^*_{\varepsilon 2} = C_{\varepsilon 2} - \alpha^3(0.4 - C_{\varepsilon 2})\tanh\left(\left|\frac{\mathrm{div}\left(\nu_t/\sigma_k\underline{\nabla}k\right)}{\varepsilon}\right|^{3/2}\right) \tag{II.7.14}$$

This improves the predictions of the dissipation rate in the defect layer of a channel flow (where the turbulent transport becomes significant) and yields better results in wake flows [Parneix et al., 1996]. Full details of the scheme can be found in [Billard and Laurence, 2012].

in extenso definition

Equations: Equations for the turbulence kinetic energy k, the turbulence dissipation rate ε, the non-dimensionnal wall-normal Reynolds stress component $\varphi = \overline{v^2}/k$ and the elliptic blending parameter α are given in Eq. (II.7.12), Eq. (II.7.13), Eq. (II.7.7) and Eq. (II.7.6) respectively.

Scales and constants: The definition of the turbulent viscosity is given in Eq. (II.7.15). The near-wall and far field models, f_w and f_h for the φ source term, f are expressed in Eq. (II.7.16) and Eq. (II.7.17). The definition of the variable coefficient $C^*_{\varepsilon 2}$ is given in Eq. (II.7.18)

Finally the time and length scales entering the definition of ν_t, the equation of ε and the definition of f_h as well as the equation of α are given in Eq. (II.7.19). The viscous limiter used as lower bound of the time scale has a finite wall value and therefore enables avoiding the singularity consecutive to the definition of the ε sink term if the term $-C^*_{\varepsilon 2}\frac{\varepsilon^2}{k}$ were used in place of $-C^*_{\varepsilon 2}\frac{\varepsilon}{T}$. Similarly a viscous (Kolmogorov) limiter is used for the length-scale definition L to avoid numerical problems which would raise in the α equation numerical resolution if the length-scale were to tend to zero at wall. The upper limiter T_{lim} is used to enforce the Bradshaw hypothesis (proportionality between shear stress and turbulent kinetic energy in a boundary layer $\overline{uv}/k = C(= 0.6/\sqrt{3}$ with the present approach)), and corrects, in a wider range of cases, the excessive production rate returned by the eddy viscosity formulation (*i.e.* allowing a linear rather than a quadratic dependence on S for large strain rate).

$$\nu_t = C_\mu \varphi k \ \min(T, T_{lim}) \tag{II.7.15}$$

$$f_w = -\frac{\varepsilon}{2}\frac{\varphi}{k} \tag{II.7.16}$$

$$f_h = -\frac{1}{T}\left(C_1 - 1 + C_2\frac{P}{\varepsilon}\right)\left(\varphi - \frac{2}{3}\right) \tag{II.7.17}$$

$$C^*_{\varepsilon 2} = C_{\varepsilon 2} + \alpha^3 \left(C_{\varepsilon 4} - C_{\varepsilon 2}\right) \tanh\left(\left|\frac{\text{div}\left(\nu_t/\sigma_k \underline{\nabla} k\right)}{\varepsilon}\right|^{3/2}\right) \tag{II.7.18}$$

$$\begin{cases} L = \sqrt{C_L^2 \left(\dfrac{k^3}{\varepsilon^2} + C_\eta^2 \dfrac{\nu^{3/2}}{\varepsilon^{1/2}}\right)} \\ T = \sqrt{\dfrac{k^2}{\varepsilon^2} + C_T^2 \dfrac{\nu}{\varepsilon}} \\ T_{lim} = \dfrac{0.6}{\sqrt{6} C_\mu \varphi \sqrt{\underline{\underline{S}} : \underline{\underline{S}}}} \end{cases} \tag{II.7.19}$$

Constants: Table 7.2 gives the value adopted for the constants of the model.

$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	$C_{\varepsilon 3}$	$C_{\varepsilon 4}$	σ_k	σ_ε
1.44	1.83	2.3	0.4	1	1.5

C_μ	C_T	C_L	C_η	C_1	C_2	σ_φ
0.22	4	0.164	75	1.7	0.9	1

Table 7.2: Constants of the BL-$\overline{v^2}/k$ model

Boundary conditions

The turbulent variables wall boundary conditions are given in Eq. (II.7.20) (y being the wall-distance)[2]:

$$\begin{cases} \lim\limits_{y\to 0} k = 0 \\ \lim\limits_{y\to 0} \varepsilon = \lim\limits_{y\to 0} \dfrac{\nu k}{y^2} \\ \lim\limits_{y\to 0} \varphi = 0 \\ \lim\limits_{y\to 0} \alpha = 0 \end{cases} \tag{II.7.20}$$

Remaining issues

- The actual source term $\left(\partial_k \partial_j U_i\right)\left(\partial_k \partial_j U_i\right)$ should be written as $\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\left(\partial_k \partial_j U_i\right)^2$ and not $\sum_{i=1}^{3}\left(\sum_{j=1}^{3}\sum_{k=1}^{3}\left(\partial_k \partial_j U_i\right)\right)^2$ as it is, incorrectly, in the present implementation.
- Issues regarding grid/code dependancy of this term have already been raised [Iaccarino, 2001]. Alternatives may be worth investigating, such as $\left(\partial_{jj} U_i\right)^2$.

[2]Note that the ε wall boundary condition is halved compared to what is used in the $\varphi - \overline{f}$ model (`iturb=50`) consequently to the change of variable described above

7.1.3 Spalart-Allmaras model

The Spalart-Allmaras turbulence model [Spalart and Allmaras, 1992] is an *EVM RANS* model developed in the 90's in aeronautics, and is therefore well suited for studying a flow around an air-plane wing for instance.

Model description

It consists in a transport equation of a scalar $\tilde{\nu}$ directly linked to the turbulent viscosity μ_T.

More recently, this model has been extended by Aupoix [Aupoix and Spalart, 2003] to rough wall for studying atmospheric flows. It was also successfully applied to flow in turbo-machinery where variants of this model has been developed.

The transport equation of $\tilde{\nu}$ (pseudo turbulent viscosity, which tends to it far from walls) reads[3]

$$\rho\frac{\partial\tilde{\nu}}{\partial t}+\underline{\nabla}\tilde{\nu}\cdot(\rho\underline{u}) = c_{b1}\rho\tilde{S}\tilde{\nu}-c_{w1}f_w\rho\left(\frac{\tilde{\nu}}{d}\right)^2+\frac{1}{\sigma}\left[\text{div}\left((\mu+\rho\tilde{\nu})\underline{\nabla}\tilde{\nu}\right)+c_{b2}\rho\left|\underline{\nabla}\tilde{\nu}\right|^2\right]+\Gamma\left(\tilde{\nu}^{in}-\tilde{\nu}^n\right)+ST^{imp}\tilde{\nu}+ST^{exp} \tag{II.7.21}$$

where $\tilde{\nu}^{in}$ is the injection value of $\tilde{\nu}$ in case of any mass source term, and $ST^{imp}_{\tilde{\nu}}$ and $ST^{exp}_{\tilde{\nu}}$ are respectively the implicit and explicit additional user source terms and where

$$\begin{aligned}
\mu_T &= \rho f_{v1}\tilde{\nu} \\
f_{v1} &= \frac{\chi^3}{\chi^3+c_{v1}^3} \\
\chi &= \frac{\tilde{\nu}}{\nu} \\
\tilde{S} &= \Omega+\frac{\tilde{\nu}}{\kappa^2 d^2}f_{v2} \\
f_{v2} &= 1-\frac{\tilde{\nu}}{\nu+\tilde{\nu}f_{v1}} \\
f_w &= g\left[\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right]^{\frac{1}{6}} \\
g &= r+c_{w2}\left(r^6-r\right) \\
r &= \min\left[\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2};\ 10\right]
\end{aligned} \tag{II.7.22}$$

The constants are defined in Table 7.3.

σ	c_{b1}	c_{b2}	κ	c_{w2}	c_{w3}	c_{v1}	c_{w1}
$\frac{2}{3}$	0.1355	0.622	0.41	0.3	2	7.1	$\frac{c_{b1}}{\kappa^2}+\frac{1+c_{b2}}{\sigma}$

Table 7.3: Constants of the Spalart Allmaras model.

[3] the present formulation is a simplified one presented by Aupoix [Aupoix and Spalart, 2003] where transition terms have been neglected.

Time stepping

Equation (II.7.21) can be rewritten with the θ-scheme presented in Chapter 3 as

$$\begin{aligned}
&\overbrace{\left[\frac{\rho}{\Delta t}+\max\left(c_{w1}f_w\rho\frac{\tilde{\nu}^n}{d^2}-c_{b1}\rho\tilde{S},\,0\right)-\theta T_s^{imp}+\theta\Gamma^n\right]}^{\text{implicit term}}\delta\tilde{\nu}^{n+1}\\
&+\theta\left(\underbrace{\underline{\nabla}\delta\tilde{\nu}^{n+1}\cdot(\rho\underline{u})}_{\text{implicit part of the convection}}-\underbrace{\text{div}\left[\frac{\mu+\rho\tilde{\nu}^n}{\sigma}\underline{\nabla}\delta\tilde{\nu}^{n+1}\right]}_{\text{implicit part of the diffusion}}\right)\\
&=\\
&-\underline{\nabla}\tilde{\nu}^n\cdot(\rho\underline{u})+\text{div}\left[\frac{\mu+\rho\tilde{\nu}^n}{\sigma}\underline{\nabla}\tilde{\nu}^n\right]\\
&+c_{b1}\rho\tilde{S}\tilde{\nu}^n-c_{w1}f_w\rho\left(\frac{\tilde{\nu}^n}{d}\right)^2+\frac{c_{b2}\rho}{\sigma}\left|\underline{\nabla}\tilde{\nu}^n\right|^2+\Gamma\left(\tilde{\nu}^{in}-\tilde{\nu}^n\right)+ST^{imp}\tilde{\nu}^n+ST^{exp}
\end{aligned}\tag{II.7.23}$$

where $\delta\tilde{\nu}^{n+1}\equiv\tilde{\nu}^{n+1}-\tilde{\nu}^n$.

Remark 7.1 *The term* $\left(c_{w1}f_w\rho\frac{\tilde{\nu}^n}{d^2}-c_{b1}\rho\tilde{S}\right)$ *is implicit so that* $\tilde{\nu}$ *does not require any clipping to remain positive if an* upwind *convective scheme and no flux reconstruction are chosen.*

Boundary conditions

Smooth walls: the boundary condition on $\tilde{\nu}$ is a standard zero Dirichlet boundary condition on the walls (see Chapter 5 for the encoding of standard Dirichlet conditions).

Note that the model gives a log law outside of the viscous sub-layer, *i.e.*:

$$\begin{aligned}
\tilde{\nu} &\simeq \kappa u^\star d\\
\tilde{S} &\simeq \frac{u^\star}{\kappa d}
\end{aligned}\tag{II.7.24}$$

Rough walls: In case of rough walls, let us define:

$$\begin{aligned}
\chi_{rough} &= \chi+c_{R1}\frac{h_s}{d_{rough}}\\
d_{rough} &= d+d_0\\
d_0 &= \exp\left(-8.5\kappa\right)h_s\simeq 0.03h_s
\end{aligned}\tag{II.7.25}$$

where h_s is the roughness size. The Dirichlet boundary conditions is replaced by the following Neumann boundary condition:

$$\left.\frac{\partial\tilde{\nu}}{\partial n}\right|_{f_b}=\frac{\tilde{\nu}|_{f_b}}{d_0}\tag{II.7.26}$$

A development in series in then written:

$$\tilde{\nu}_{f_b}=\tilde{\nu}_{\underline{x}'_c}-\underline{\nabla}_{f_b}\tilde{\nu}\cdot(\underline{x}_f-\underline{x}'_c)\tag{II.7.27}$$

Finally, that is a Robin type boundary condition formulated as follows in code_saturne:

$$\tilde{\nu}_{f_b}=A^g_{f_b}-B^g_{f_b}\tilde{\nu}_{\underline{x}'_c}\tag{II.7.28}$$

with $A^g_{f_b} = 0$ and

$$B^g_{f_b} = \frac{d_0}{d_0 + \left|\underline{x}_f - \underline{x}'_c\right|} \tag{II.7.29}$$

Inlet: the profile of $\tilde{\nu}$ is imposed, the value is deduced from the profiles imposed on k and ε for a $k-\varepsilon$ turbulence model assuming $\tilde{\nu} \simeq \nu_T$.

See the programmers reference of the dedicated subroutine for further details.

7.2 Differential Reynolds Stress Models ($DRSM$)

In this section, the presented models solve a differential transport equation on the Reynolds' stresses tensor.

7.2.1 Equations for the Reynolds stress tensor components R_{ij} and ε (LRR model)

$$\left\{\begin{array}{rcll} \rho\dfrac{\partial \underline{\underline{R}}}{\partial t} + \underline{\underline{\nabla}}\,\underline{R}\cdot(\rho\underline{u}) - \underline{\underline{\mathrm{div}}}\left(\mu\underline{\underline{\nabla}}\,\underline{R}\right) & = & \underline{d} + \underline{\mathcal{P}} + \underline{G} + \underline{\Phi} - \rho\underline{\varepsilon} & +\Gamma\underline{R}^{in} + \underline{\underline{ST}}_{R_{ij}}, \\ \rho\dfrac{\partial \varepsilon}{\partial t} + \underline{\nabla}\varepsilon\cdot(\rho\underline{u}) - \mathrm{div}\,(\mu\underline{\nabla}\varepsilon) & = & d + C_{\varepsilon_1}\dfrac{\varepsilon}{k}\left[\mathcal{P} + G_\varepsilon\right] - \rho C_{\varepsilon_2}\dfrac{\varepsilon^2}{k} & +\Gamma\varepsilon^{in} + ST_\varepsilon, \end{array}\right. \tag{II.7.30}$$

where $\underline{\underline{\mathcal{P}}}$ stands for the turbulence production tensor associated with mean flow strain-rate and $\underline{\underline{\mathcal{G}}}$ is stands for the production- tensor associated with buoyancy effects:

$$\begin{aligned} \underline{\underline{\mathcal{P}}} &= -\rho\left[\underline{R}\cdot\underline{\underline{\nabla}}\,\underline{u}^T + \underline{\underline{\nabla}}\,\underline{u}\cdot\underline{R}\right], \\ \underline{\underline{\mathcal{G}}} &= \left[\underline{r}\otimes\underline{g} + \underline{g}\otimes\underline{r}\right]. \end{aligned} \tag{II.7.31}$$

where $\underline{r} \equiv \overline{\rho'\underline{u}'}$ is modelled through a Generalized Gradient Diffusion Hypothesis (GGDH): $\underline{r} \simeq \frac{3}{2}\frac{C_\mu}{\sigma_t}\frac{k}{\varepsilon}\underline{\underline{R}}\cdot\underline{\nabla}\rho$ and $G_\varepsilon = \mathrm{Max}\left(0,\ \frac{1}{2}tr\underline{\underline{\mathcal{G}}}\right)$.

Remark 7.2 *Under Boussinesq assumption (ρ varies only in the buoyancy term in the momemtum equation, lineraly with respect to θ, the velocity density correlation becomes $\rho\overline{\theta'\underline{u}'}$.*

With these definitions the following relations hold:

$$\begin{aligned} k &= \tfrac{1}{2}tr\underline{\underline{R}}, \\ \mathcal{P} &= \tfrac{1}{2}tr\left(\underline{\underline{\mathcal{P}}}\right), \end{aligned} \tag{II.7.32}$$

$\underline{\underline{\Phi}}$ is the term representing pressure-velocity correlations:

$$\underline{\underline{\Phi}} = \underline{\underline{\phi}}_1 + \underline{\underline{\phi}}_2 + \underline{\underline{\phi}}_3 + \underline{\underline{\phi}}_w, \tag{II.7.33}$$

$$\begin{aligned} \underline{\underline{\phi}}_1 &= -\rho\, C_1\frac{\varepsilon}{k}\underline{\underline{R}}^D, \\ \underline{\underline{\phi}}_2 &= -C_2\underline{\underline{\mathcal{P}}}^D, \\ \underline{\underline{\phi}}_3 &= -C_3\underline{\underline{\mathcal{G}}}^D. \end{aligned} \tag{II.7.34}$$

The term $\underline{\underline{\phi}}_w$ is called *wall echo term* (by default, it is not accounted for: see html programmer's documentation of the subroutine and the appendix O).

The dissipation term, $\underline{\underline{\varepsilon}}$, is considered isotropic:

$$\underline{\underline{\varepsilon}} = \frac{2}{3}\,\varepsilon \underline{\underline{1}}. \tag{II.7.35}$$

The turbulent diffusion terms are:

$$\begin{aligned} \underline{\underline{d}} &= C_S \underline{\underline{\mathrm{div}}} \left(\rho \frac{k}{\varepsilon} \underline{\underline{R}} \cdot \underline{\underline{\nabla}}\, \underline{\underline{R}} \right), \\ d &= C_\varepsilon \mathrm{div} \left(\rho \frac{k}{\varepsilon} \underline{\underline{R}} \cdot \underline{\nabla} \varepsilon \right). \end{aligned} \tag{II.7.36}$$

In the rare event of mass sources, ΓR_{ij}^{in} and $\Gamma \varepsilon^i$ are the corresponding injection terms. $ST_{R_{ij}}$ and ST_ε are also rarely used additional source terms that can be prescribed by the user.

C_μ	C_ε	C_{ε_1}	C_{ε_2}	C_1	C_2	C_3	C_S	C_1'	C_2'
0.09	0.18	1.44	1.92	1.8	0.6	0.55	0.22	0.5	0.3

Table 7.4: Model constants of the *LRR* $R_{ij} - \varepsilon$ model [Launder et al., 1975].

7.3 Large-Eddy Simulation (LES)

7.3.1 Standard Smagorinsky model

$$\mu_t = \rho \left(C_s \overline{\Delta} \right)^2 \sqrt{2 \underline{\underline{\overline{S}}} : \underline{\underline{\overline{S}}}}, \tag{II.7.37}$$

where $\underline{\underline{\overline{S}}}$ the filtered strain rate tensor components:

$$\underline{\underline{\overline{S}}} = \underline{\underline{\overline{S}}}^S = \frac{1}{2} \left[\underline{\underline{\nabla}}\, \overline{\underline{u}} + \left(\underline{\underline{\nabla}}\, \overline{\underline{u}} \right)^T \right]. \tag{II.7.38}$$

Here, $\overline{u_i}$ stands for the i^{th} resolved velocity component [4].

C is the Smagorinsky constant. Its theoretical value is 0.18 for homogeneous isotropic turbulence, but the value 0.065 is classic for channel flow.

$\overline{\Delta}$ is the filter width associated with the finite volume formulation (implicit filtering which corresponds to the integration over a cell). The value recommended for hexahedral cells is: $\overline{\Delta} = 2\,|V_c|^{\frac{1}{3}}$ where $|V_c|$ is the volume of the cell c.

See the programmers reference of the dedicated subroutine for further details.

[4] In the case of implicit filtering, the discretization in space introduces a spectral low pass filter: only the structures larger that twice the size of the cells are accounted for. Those structures are called "the resolved scales", whereas the rest, $u_i - \overline{u_i}$, is referred to as "unresolved scales" or "sub-grid scales". The influence of the unresolved scales on the resolved scales have to be modelled.

7.3.2 Dynamic Smagorinsky model

A second filter is introduced: it is an explicit filter with a characteristic width $\widetilde{\Delta}$ superior to that of the implicit filter ($\overline{\Delta}$). If Y is a discrete variable defined over the computational domain, the variable obtained after applying the explicit filter to Y is noted $\tilde{Y}$. Moreover, with:

$$\begin{aligned} \underline{\underline{L}} &= \widetilde{\overline{\underline{u}} \otimes \overline{\underline{u}}} - \widetilde{\overline{\underline{u}}} \otimes \widetilde{\overline{\underline{u}}}, \\ \underline{\underline{\tau}} &= \overline{\underline{u} \otimes \underline{u}} - \overline{\underline{u}} \otimes \overline{\underline{u}}, \\ \underline{\underline{T}} &= \widetilde{\overline{\underline{u} \otimes \underline{u}}} - \widetilde{\overline{\underline{u}}} \otimes \widetilde{\overline{\underline{u}}}, \end{aligned} \tag{II.7.39}$$

Germano identity reads:

$$\underline{\underline{L}} = \underline{\underline{T}} - \widetilde{\underline{\underline{\tau}}}. \tag{II.7.40}$$

Both dynamic models described hereafter rely on the estimation of the tensors $\underline{\underline{T}}$ and $\underline{\underline{\tau}}$ as functions of the filter widths and of the strain rate tensor (Smagorinsky model). The following modelling is adopted[5]:

$$\begin{aligned} T_{ij} - \tfrac{1}{3} tr\underline{\underline{T}} \delta_{ij} &= -2C\widetilde{\Delta}^2 |\widetilde{\overline{D_{ij}}}| \widetilde{\overline{D_{ij}}}, \\ \tau_{ij} - \tfrac{1}{3} \underline{\underline{\tau}} \delta_{ij} &= -2C^* \overline{\Delta}^2 |\overline{D_{ij}}| \overline{D_{ij}}, \end{aligned} \tag{II.7.41}$$

where $\overline{u}$ stands for the *implicit-filtered* value of a variable u defined at the centres of the cells and $\tilde{u}$ represents the *explicit-filtered* value associated with the variable u. It follows that the numerical computation of L_{ij} is possible, since it requires the explicit filtering to be applied to implicitly filtered variables only (*i.e.* to the variables explicitly computed).

On the following assumption:

$$C = C^*, \tag{II.7.42}$$

and assuming that C^* is only slightly non-uniform, so that it can be taken out of the explicit filtering operator, the following equation is obtained:

$$\underline{\underline{L}}^D = C \left(\underline{\underline{\alpha}} - \widetilde{\underline{\underline{\beta}}} \right), \tag{II.7.43}$$

with:

$$\begin{aligned} \alpha_{ij} &= -2\widetilde{\Delta}^2 |\widetilde{\overline{D_{ij}}}| \widetilde{\overline{D_{ij}}}, \\ \beta_{ij} &= -2\overline{\Delta}^2 |\overline{D_{ij}}| \overline{D_{ij}}. \end{aligned} \tag{II.7.44}$$

Since we are left with six equations to determine one single variable, the least squares method is used[6]. With:

$$\underline{\underline{E}} = \underline{\underline{L}} - C \left(\underline{\underline{\alpha}} - \widetilde{\underline{\underline{\beta}}} \right), \tag{II.7.45}$$

the value for C is obtained by solving the following equation:

$$\frac{\partial \underline{\underline{E}} : \underline{\underline{E}}}{\partial C} = 0. \tag{II.7.46}$$

[5] δ_{ij} stands for the Kroeneker symbol.
[6] $tr\underline{\underline{L}}$ has no effect for incompressible flows.

Finally:

$$C = \frac{\underline{\underline{M}} : \underline{L}}{\underline{\underline{M}} : \underline{\underline{M}}}, \tag{II.7.47}$$

with

$$\underline{\underline{M}} = \underline{\alpha} - \underline{\widetilde{\beta}}. \tag{II.7.48}$$

This method allows to calculate the Smagorinsky "constant" dynamically at each time step and at each cell. However, the value obtained for C can be subjected to strong variations. Hence, this approach is often restricted to flows presenting one or more homogeneous directions (Homogeneous Isotropic Turbulence, 2D flows presenting an homogeneous span-wise direction...). Indeed, in such cases, the model can be (and is) stabilized by replacing C by an average value of C computed over the homogeneous direction(s).

For a general case (without any homogeneous direction), a specific averaging is introduced to stabilize the model: for any given cell of the mesh, the averaged Smagorinsky constant is obtained as an average of C over the "extended neighbouring" of the cell (the set of cells that share at least one vertex with the cell considered). More precisely, the average value (also denoted C hereafter) is calculated as indicated below:

$$C = \frac{\widetilde{\underline{\underline{M}} : \underline{L}}}{\widetilde{\underline{\underline{M}} : \underline{\underline{M}}}} \tag{II.7.49}$$

See the programmers reference of the dedicated subroutine for further details.

7.4 Turbulence models for velocity – scalar correlations

7.4.1 Simple Gradient Diffusion Hypothesis (SGDH)

The simplest models assume that turbulent fluxes $\overline{Y'\underline{u}'}$ is aligned with the mean gradient as for Fick diffusion law:

$$\rho \frac{\mathrm{d}Y}{\mathrm{d}t} = \underbrace{\mathrm{div}\left[\left(\frac{\mu}{Sc} + \frac{\mu_T}{Sc_T}\right)\underline{\nabla} Y\right]}_{d_Y} \tag{II.7.50}$$

where Sc and Sc_T are respectively the molecular and the turbulent Schmidt numbers.

7.4.2 Generalized Gradient Diffusion Hypothesis (GGDH)

With GGDH, Equation (II.7.50) becomes:

$$\rho \frac{\mathrm{d}Y}{\mathrm{d}t} = \underbrace{\mathrm{div}\left[\left(\frac{\mu}{Sc}\underline{\underline{1}} + \rho C_\theta \frac{k}{\varepsilon}\underline{\underline{R}}\right)\cdot \underline{\nabla} Y\right]}_{d_Y} \tag{II.7.51}$$

7.4.3 Scalar Variance

The variance transport equation is for a scalar Y:

$$\rho \frac{\mathrm{d}\overline{Y'^2}}{\mathrm{d}t} = \underbrace{-2\rho \underline{\nabla}\overline{Y}\cdot\overline{Y'\underline{u}'}}_{P_{Y^2}} + \mathrm{div}\left(\underbrace{\frac{\mu}{Sc}\underline{\nabla}\overline{Y'^2}}_{D^\nu_{Y^2}} \underbrace{-\rho\overline{Y'^2\underline{u}'}}_{D^t_{Y^2}}\right) \underbrace{-2\frac{\mu}{Sc}\overline{\underline{\nabla}Y'\cdot\underline{\nabla}Y'}}_{\varepsilon_{Y^2}} \tag{II.7.52}$$

where P_{Y^2}, $D^{\nu}_{Y^2}$, $D^{t}_{Y^2}$ et ε_{Y^2} are respectively the terms of production, molecular diffusion, turbulent diffusion and variance dissipation. The production and molecular diffusion terms are exact, only the turbulent diffusion and the dissipation need to be modelled. For the turbulent diffusion $D^{t}_{Y^2}$, a model GGDH can be used if the Reynolds stresses are solved. The variance transport equation becomes:

$$\rho\frac{\mathrm{d}\overline{Y'^2}}{\mathrm{d}t} = \mathrm{div}\left[\left(\frac{\mu}{Sc}\underline{\underline{1}} + C_{\theta\theta}\ \rho\underline{\underline{R}}\frac{k}{\varepsilon}\right)\cdot\underline{\nabla}\overline{Y'^2}\right] + P_{Y^2} - \rho\varepsilon_{Y^2} \tag{II.7.53}$$

Two methods exist to model ε_{Y^2}; one using a transport equation and one using the time scale ratio R_f, such as:

$$R_f = \frac{\tau_Y}{\tau_u} = \frac{\overline{Y'^2}}{\varepsilon_{Y^2}}\frac{\varepsilon}{k}$$

thus,

$$\varepsilon_{Y^2} = \frac{\overline{Y'^2}}{R_f}\frac{\varepsilon}{k} \tag{II.7.54}$$

By default, a SGDH equation is solved:

$$\rho\frac{\mathrm{d}\overline{Y'^2}}{\mathrm{d}t} = \mathrm{div}\left[\left(\frac{\mu}{Sc} + \frac{\mu_T}{Sc_T}\right)\underline{\nabla}\overline{Y'^2}\right] + P_{Y^2} - \underbrace{\rho\frac{\varepsilon\overline{Y'^2}}{kR_f}}_{\varepsilon_{Y^2}} \tag{II.7.55}$$

with $Sc_T = 1$ and $R_f = 0.8$, by default.

With R_{ij} models (SSG, LRR, EB-RSM), a Daly Harlow method is used for the turbulent diffusion term and $R_f = 0.5$.

The so called production term reads $P_{Y^2} = -2\overline{Y'\underline{u}'}\cdot\underline{\nabla}\overline{Y}$, and rewrites with SGDH: $P_{Y^2} \simeq 2\frac{\mu_T}{Sc_T}\left|\underline{\nabla}\overline{Y}\right|^2$.

7.4.4 Algebraic Flux Models (AFM)

This Algebraic model is defined by the following equation:

$$\overline{Y'\underline{u}'} = -C_\theta\tau\left[\underline{\underline{R}}\cdot\underline{\nabla}\overline{Y} + \xi\overline{Y'\underline{u}'}\cdot\underline{\underline{\nabla}}\,\underline{u} + \eta\beta\overline{Y'^2}\underline{g}\right] \tag{II.7.56}$$

where $C_\theta = C'_\theta/C_{1\theta}$; $\xi = 1 - C_{2\theta}$; $\eta = 1 - C_{3\theta}$. With: $C_\theta = 0.236$, $C'_\theta = 0.98$, $C_{1\theta} = 4.15$, $C_{2\theta} = 0.3$ and $C_{3\theta} = 0.6$.

For the part with the temperature gradient, it is treated like the GGDH. The other part with the velocity gradient and the gravity is treated in explicit.

7.4.5 Differential Flux Model (DFM)

The mean turbulent heat flux transport equations are:

$$\begin{aligned}\frac{\mathrm{d}\overline{Y'\underline{u}'}}{\mathrm{d}t} = &\underbrace{-\underline{\underline{\nabla}}\,\overline{u}\cdot\overline{Y'\underline{u}'}}_{\underline{P}^U_\theta}\underbrace{-\underline{\underline{R}}\cdot\underline{\nabla}\overline{Y}}_{\underline{P}^Y_\theta}\underbrace{-\beta\overline{Y'^2}\underline{g}}_{\underline{G}_\theta}\underbrace{-\overline{\frac{p'}{\rho}\underline{\nabla}Y'}}_{\underline{\phi}_\theta} - \underbrace{(\kappa+\nu)\overline{\underline{\underline{\nabla}}\,\underline{u}'\cdot\underline{\nabla}Y'}}_{\underline{\varepsilon}_\theta}\\ &+\underbrace{\underline{\mathrm{div}}\left(-\overline{Y'\underline{u}'\otimes\underline{u}'}\right)}_{\underline{D}^t_\theta} + \underbrace{\underline{\mathrm{div}}\left(\kappa\overline{\underline{u}'\otimes\underline{\nabla}Y'} + \nu\overline{Y'\underline{\underline{\nabla}}\,\underline{u}'}\right)}_{\underline{D}^\nu_\theta} + \underbrace{\underline{\mathrm{div}}\left(\overline{\frac{Y'p'}{\rho}}\underline{\underline{1}}\right)}_{\underline{D}^p_\theta}\end{aligned} \tag{II.7.57}$$

Only the production terms are exact, the equation becomes with diffusion and scrambling models:

$$\frac{\mathrm{d}\overline{Y'\underline{u}'}}{\mathrm{d}t} = \underline{P}_\theta + \underline{\phi}_\theta - \underline{\varepsilon}_\theta + \underline{\mathcal{D}}_\theta \tag{II.7.58}$$

where $\underline{P}_\theta = \underline{P}^U_\theta + \underline{P}^Y_\theta + \underline{G}_\theta$ et $\underline{\mathcal{D}}_\theta = \underline{\mathcal{D}}^\nu_\theta + \underline{\mathcal{D}}^t_\theta + \underline{\mathcal{D}}^p_\theta$.

$$\underline{\mathcal{D}}^\nu_\theta = \underline{\mathrm{div}}\left(\frac{\nu+\kappa}{2}\underline{\underline{\nabla}}\overline{Y'\underline{u}'}\right) \tag{II.7.59}$$

$$\underline{\mathcal{D}}^t_\theta + \underline{\mathcal{D}}^p_\theta = \underline{\mathrm{div}}\left(C_\theta \tau \underline{\underline{\nabla}}\overline{Y'\underline{u}'}\underline{\underline{R}}\right) \tag{II.7.60}$$

with $C_\theta = 0.22$.

$$\underline{\varepsilon}_\theta = 0 \tag{II.7.61}$$

$$\underline{\phi}_\theta = -C_{1\theta}\frac{1}{\tau}\overline{Y'\underline{u}'} + C_{2\theta}\underline{\underline{\nabla}}\,\overline{\underline{u}}\cdot\overline{Y'\underline{u}'} + C_{3\theta}\beta\overline{Y'^2}\underline{g} + C_{4\theta}\underline{\underline{\nabla}}\overline{Y}\cdot\underline{\underline{R}} \tag{II.7.62}$$

where $C_{1\theta} = 4.15$, $C_{2\theta} = 0.55$, $C_{3\theta} = 0.5$ and $C_{4\theta} = 0$.

Chapter 8

Compressible flows

See § A.

8.1 Pressure-based solver

A description of the algorithm can be found in [Colas et al., 2019].

Chapter 9

Combustion

9.1 Introduction

9.1.1 Use & call

From a CFD point of view, combustion is a (sometimes very) complicated way to determine ρ, the density.

Depending on the presence of a match or not, two solutions exist, known as ignited and extinguished. From a numerical point of view, it means that these two solutions have two attraction basin; the more representative the model, the more difficult the stabilisation of the combustion (may be difficult to ignite).

However, combustion models needs few extra fields of scalar with regular transport equations, some of them with a reactive or interfacial source term.

This version of code_saturne focuses on steady industrial combustion processes; propagating fires are out of the present range (but in the short coming release).

In code_saturne modelling of combustion is able to deal with gas phase combustion (diffusion, premix, partial premix), and with solid or liquid finely dispersed fuels (fixed and fluidised beds are out of range).

Combustion of condensed fuels involves one-way interfacial flux due to phenomena in the condensed phase (evaporation or pyrolisis) and reciprocal ones (heterogeneous combustion and gasification). Many of the species injected in the gas phase are afterwards involved in gas phase combustion.

That is the reason why many modules are similar for gas, coal and fuel oil combustion modelling. Obviously, the thermodynamical description of gas species is similar in every version as close as possible to the JANAF rules.

All models are developed in both adiabatic and unadiabatic (permeatic : heat loss, *e.g.* by radiation) version, beyond the standard (`-1, 0, 1`), the rule to call models in `usppmo` is:

$$\begin{aligned}
&\texttt{ippmod(index ; model)} = -1 \quad \text{unused} \\
&\texttt{ippmod(index ; model)} = 0 \quad \text{simplest adiabatic version} \\
&\texttt{ippmod(index ; model)} = 1 \quad \text{simplest permeatic version} \qquad \text{(II.9.1)}
\end{aligned}$$

and possibly:

$$\begin{aligned}
&\texttt{ippmod(index ; model)} = 2.p \quad \text{p adiabatic version} \\
&\texttt{ippmod(index ; model)} = 2.p+1 \quad \text{p permeatic version}
\end{aligned}$$

Every permeatic version involves the transport of enthalpy (one more variable).

9.1.2 Gas combustion modelling

Gas combustion is limited by disponibility (in the same fluid particle) of both fuel and oxidizer and by kinetic effects (a lot of chemical reactions involved can be described using an Arrhenius law with high activation energy). The mixing of mass (atoms) incoming with fuel and oxydizer is described by a mixture fraction (mass fraction of matter incoming with fuel), this variable is not affected by combustion.

A progress variable is used to describe the transformation of the mixture from fuel and oxydant to products (carbon dioxyde and so on). Combustion of gas is, traditionnaly, splitted in premix and diffusion regimes.

In premixed combustion process a first stage of mixing has been realised (without blast ...) and the mixture is introduced in the boiler (or combustor can). In common industrial conditions the combustion is mainly limited by the mixing of fresh gases (frozen) and burnt gases (exhausted) resulting in the inflammation of the first and their conversion to burnt ones ; so an assumption of chemistry much faster

than mixing (characteristic time for chemistry much smaller than characteristic time for turbulent mixing) induces an intermittent regime. The gas flow is constituted of totally fresh and totally burnt gases (the flame containing the gases during their transformation is extremely thin). With this previous assumptions, Spalding [Spalding and ???, 1971] established the "Eddy Break Up" model, which allows a complete description of the combustion process with only one progress variable (mixture fraction is both constant - in time - and homogeneous - in space).

In diffusion flames the fuel and the oxydant are introduced by, at least, two inlets. In ordinary industrial conditions, their mixing is the main limitation and the mixture fraction is enough to qualify a fluid particle, but in turbulent flows a *P*robability *D*ensity *F*unction of the mixture fraction is needed to qualify the thermodynamical state of the bulk. So, at least, both the mean and the variance of the mixture fraction are needed (two variables) to fit parameters of the pdf (the shape of whose is presumed).

Real world's chemistry is not so fast and, unfortunately, the mixing can not be as homogeneous as wished. The main part of industrial combustion occurs in partial premix regime. Partial premix occurs if mixing is not finished (at molecular level) when the mixture is introduced, or if air or fuel, are staggered, or if a diffusion flame is blown off. For these situations, and specifically for lean premix gas turbines [Libby and Williams, 2000] developed a model allowing a description of both mixing and chemical limitations. A collaboration between the LCD Poitiers [Ribert et al., 2004] and EDF R&D has produced a simpler version of their algorithm. Not only the mean and the variance of both mixture fraction and progress variable are needed but also their covariance (five variables).

9.1.3 Two-phase combustion modelling

Coal combustion is the main way to produce electricity in the world. Heavy fuel oil combustion have been hugely used to produce electrical energy. Biomass is a promising fuel to be used alone or in blend.

Advanced combustion process may include exhaust gases recycling, pure oxygen or steam injection, so this release of code_saturne takes into account three oxidizers (tracked by three mixture fractions).

Coal is a natural product with a very complex composition. During the industrial process of milling, the raw coal is broken in tiny particles of different sizes. After its introduction in the boiler, coal particles undergoes drying, devolatilisation (heating of coal turn it in a mixture of char and gases), heterogenous combustion (of char by oxygen in carbon monoxide), gasification (of char by carbon dioxide or by water steam in carbon monoxide), leaving ash particles.

The description of fuel evaporation is done with respect to its heaviness : after a minimum temperature is reached, the gain of enthalpy is splitted between heating and evaporation. This way the evaporation takes place on a range of temperature (which can be large). The total evaporation is common for light (domestic) oil but impossible for heavy ones : at high temperature, during the last evaporation, a craking reaction appears : so a particle similar to the char is leaved. The heterogeneous oxydation of this char particle is very similar to coal char ones.

Each of these phenomena are taken into account for some classes of particles : a solid class is caracterised by a coal (it is useful to burn mixture of coals with differents ranks or mixture of coal with biomass ...) and an initial diameter, for heavy fuel oil, liquid classes refer to initial diameter (neither possibility of blending after injection nor cofiring with oil and coal). code_saturne computes the number, the mass and the enthalpy for each class of particles by unit of mass of mixture; allowing the determination of local diameter and temperature (for each class; *e.g.* the finest will be be heated the fastest).

The main assumption is to solve only one velocity (and pressure) field: it means that the discrepancy of velocity between coal particles and gases is assumed to be negligible.

Due to the radiation, evaporation and heterogeneous combustion, temperature can be different for gas and different size particles : so the specific enthalpy of each particle class is solved.

The description of coal pyrolysis proposed by [Kobayashi and ???, 1976] is used, leading to two source

terms for light and heavy volatile matters (the moderate temperature reaction produces gases with low molecular mass, the high temperature reaction produces heavier gases and less char) represented by two passive scalars : mixture fractions. The description of the heterogeneous reaction (which produce carbon monoxide) produces a source term for the carbon: the corresponding mixture fraction is bounded far below one (the carbon can't be free, it is always in carbon monoxide form, mixed with nitrogen or other).

The retained model for the gas phase combustion is the assumption of diffusion flamelets surrounding particle (for a single paticle or a cloud), this diffusion flame establishes itself between a mixing of the previous gaseous fuels issued from fast phenomenon (pyrolysis or fuel evaporation) mixed in a local mean fuel and the mixing of oxidizers, water vapor (issued from drying) and carbon monoxide issued from slow phenomenon (heterogeneous oxydation and gasification of char). The PDF diffusion approach is used to describe the conversion of hydrocarbon to carbon monoxide (hydrocarbon conversion is assumed fast vs. mixing); the further conversion of carbon monoxide to carbon dioxyde was (in previous release, still existing for fast first evaluation of carbon dioxide useful to initialize the kinetic model) ruled by mixing or is (now recommended for better prediction of carbon monoxide at outlet and corrosion risks) kineticaly ruled with respect to the mean mass fraction and temperature (reach of equilibrium assumed slow vs. mixing). Special attention is paid to pollutant formation (conversion of H_2S to SO_2 involved in soot agglomeration, NOx formation).

9.2 Thermodynamics

9.2.1 Introduction

The description of the thermodynamical of gaseous mixture is as close as possible to the JANAF standard. The gases mixture is, often, considered as composed of some *global* species (*e.g.* oxidizer, products, fuel) each of them being a mixture (with known ratio) of *elementary* species (oxygen, nitrogen, carbon dioxide, ...).

A tabulation of the enthalpy of both elementary and global species for some temperatures is constructed (using JANAF polynoms) or read (if the user found useful to define a global specie not simply related to elementary ones; *e.g.* unspecified hydrocarbon known by C, H, O, N, S analysis and heating value). The thermodynamic properties of condensed phase are more simple: formation enthalpy is computed using properties of gaseous products of combustion with air (formation enthalpy of which is zero valued as O2 and N2 are reference state) and the lower heating value. The heat capacity of condensed phase is assumed constant and it is a data the user has to enter (in the corresponding data file dp_FCP or dp_FUE).

9.2.2 Gases enthalpy discretisation

A table of gases (both elementary species and global ones) enthalpy for some temperatures (the user choses number of points, temperature in dp_*** file) is computed (enthalpy of elementary species is computed using JANAF polynomia; enthalpy for global species are computed by weighting of elementary ones) or red (subroutine `pptbht`). Then the entahlpy is supposed to be linear vs. temperature in each temperature gap (i.e. continuous piece wise linear on the whole temperature range). As a consequence, temperature is a linear function of enthalpy; and a simple algorithm (subroutine `cothht`) allows to determine the enthalpy of a mixture of gases (for inlet conditions it is more useful to indicate temperature and mass fractions) or to determine temperature from enthalpy of the mixture and mass fractions of global species (common use in every fluid particle, at every time step).

9.2.3 Particles enthalpy discretisation

Enthalpy of condensed material is rarely known. Usually, only the low heating value and ultimate analysis are determined. So, using simple assumptions and the enthalpy of known released species (after burning with air) the formation enthalpy of coal or heavy oil can be computed. Assuming the thermal capacity is constant for every condensed material a table can be built with ... two (more is useless) temperatures, allowing the use of the same simple algorithm for temperature-enthalpy conversion. When intermediate gaseous species (volatile or vapour) are thermodynamically known, simple assumptions (*e.g.*: char is thermodynamically equivalent to pure carbon in reference state; ashes are inert) allow one to deduce enthalpy for heterogeneous reactions (these energies have not to be explicitely taken into account for the energy balance of particles).

9.3 Gas combustion

Flames with gaseous fuels can be categorized in premix, diffusion or partial-premix.

9.3.1 Premix: Eddy Break Up

The original Spalding model [Spalding and ???, 1971] was written for a situation where the whole boiler is filled with the same mixture (only one inlet); the motto of which is *"If it mixes, it burns"*. If the chemistry is fast vs. mixing, fluid particles are made of fresh gases or of burned ones. This situation is described by a progress variable (valued 0 in fresh gases and 1 in burnt ones) with a source term: the reaction one. The mixture of totally fresh or totally burnt gases, called intermittency, leads to a maximal variance of the progress variable determined by the mean value of the progress variable.

$$C"^{2}_{\max} = (C_{\text{moy}} - C_{\min}).(C_{\max} - C_{\text{moy}}) = C_{\text{moy}}.(1 - C_{\text{moy}}) \tag{II.9.2}$$

The mixing of fresh and burnt gases is the dissipation of this variance and it induces the conversion of fresh gases in burnt ones. So the source term for the (mean) progress variable is the dissipation of its (algebraic) variance. Be careful: in code_saturne the variable chosen to describe the reaction is the mass fraction of fresh gases (valued 1 in fresh and 0 in burnt), so:

$$S(Y_{\text{fg}}) = -C_{\text{ebu}}\ \frac{\epsilon}{k}\ \left[\ \rho\ Y_{\text{fg}}\left(1 - Y_{\text{fg}}\right)\right] \tag{II.9.3}$$

Where C_{ebu} is a constant, supposedly "universal", fitted around 1.6 (only advanced users may adjust this value).
Some improvements have been proposed, and largely used, for situations with mixture fraction gradient (staggering of reactant(s)) but are not theorically funded. The simplest extension is available (options 2 and 3) in code_saturne with one extra equation solved for f the mean of mixture fraction: the corresponding hypothesis is *"no variance for mixture fraction"* ... a little bit surprising in an EBU context (maximal variance for progress variable). The choice of the fresh gas mass fraction appears now to be quite relevant : the computation of species mass fraction can be done, with respect to the mean mixture fraction, both in fresh (the mass fraction of which is Y_{fg}) where species mass fraction are obvious and burnt gases (the mass fraction of which is $(1 - Y_{\text{fg}})$) among which species mass fraction come from a complete reaction assumption (as introduced hereafter for diffusion flame).

$$\begin{aligned}
Y_{\text{fuel}} &= Y_{\text{fg}}.f + (1 - Y_{\text{fg}}).\max\left(0\ ;\ \frac{f - f_{\text{s}}}{1 - f_{\text{s}}}\right) \\
Y_{\text{Ox}} &= Y_{\text{fg}}.(1 - f) + (1 - Y_{\text{fg}}).\max\left(0\ ;\ \frac{f_{\text{s}} - f}{f_{\text{s}}}\right) \\
Y_{\text{prod}} &= (1 - Y_{\text{fg}}).\left(\frac{f}{f_{\text{s}}}\ ;\ \frac{1 - f}{1 - f_{\text{s}}}\right)
\end{aligned} \tag{II.9.4}$$

Where f_{S} is the **s**toechiometric mixture **f**raction.

In adiabatic conditions the specific enthalpy of gases (in every combustion model the considered enthalpy contains formation one and heat content, but no terms for velocity or pressure) is directly related to the mixture fraction (as long as the inlet temperature for air and fuel is known). When heat losses, like radiation, are taken into account, an equation has to be solved for the mean enthalpy (such an equation is needed so when some inlets have different temperatures -partial preheat- enthalpy is then used as an extra passive scalar). In industrial processes, the aim is often to transfer the heat from burnt gases to the wall; even for heat loss the wall temperature is near to the fresh gases temperature and the main heat flux takes place between burnt gases and wall. So, in code_saturne, the specific enthalpy of the fresh gases is supposed to be related to mixture fraction and the specific enthalpy of burnt gases is locally

computed to reach the mean specific enthalpy. By this way every heat loss removed from the mean enthalpy is charged to the hottest gases.

$$\tilde{h} = Y_{\text{fg}} \,.\, h_{\text{fg}}(\tilde{f}) + (1 - Y_{\text{fg}}) \,.\, h_{\text{bg}} \quad \Leftrightarrow \quad h_{\text{bg}} = \frac{\tilde{h} - Y_{\text{fg}} \,.\, h_{\text{fg}}(\tilde{f})}{1 - Y_{\text{fg}}} \tag{II.9.5}$$

where $\tilde{f}$ is here the local mean of the mixture fraction or a constant value (in the regular EBU model).

9.3.2 Diffusion: PDF with 3 points chemistry

In diffusion model, the combustion is assumed to be limited only by mixing (between air and fuel), so the reaction is assumed to reach instantaneously its equilibrium state and the temperature and concentrations can be computed for every value of the mixture fraction. In code_saturne the implemented version uses an extra hypothesis: the reaction is complete; so if the mixture is stoechiometric, the burnt gases contains only final products (none unburnt like CO, except definition of product including a specified ratio of CO). As a consequence, every concentration is a piecewise linear function of the mixture fraction (subroutines: `d3pphy, d3pint, cpcym, fucym`).

$$0 \leq f \leq f_{\text{s}} \quad ; \quad Y_i(f) = Y_{\text{air}} + \frac{f}{f_{\text{s}}} \, (Y_{\text{S}} - Y_{\text{air}}) \tag{II.9.6}$$

$$f_{\text{s}} \leq f \leq 1 \quad ; \quad Y_i(f) = Y_{\text{S}} + \frac{f - f_{\text{s}}}{1 - f_{\text{s}}} \, (Y_{\text{fuel}} - Y_{\text{S}}) \tag{II.9.7}$$

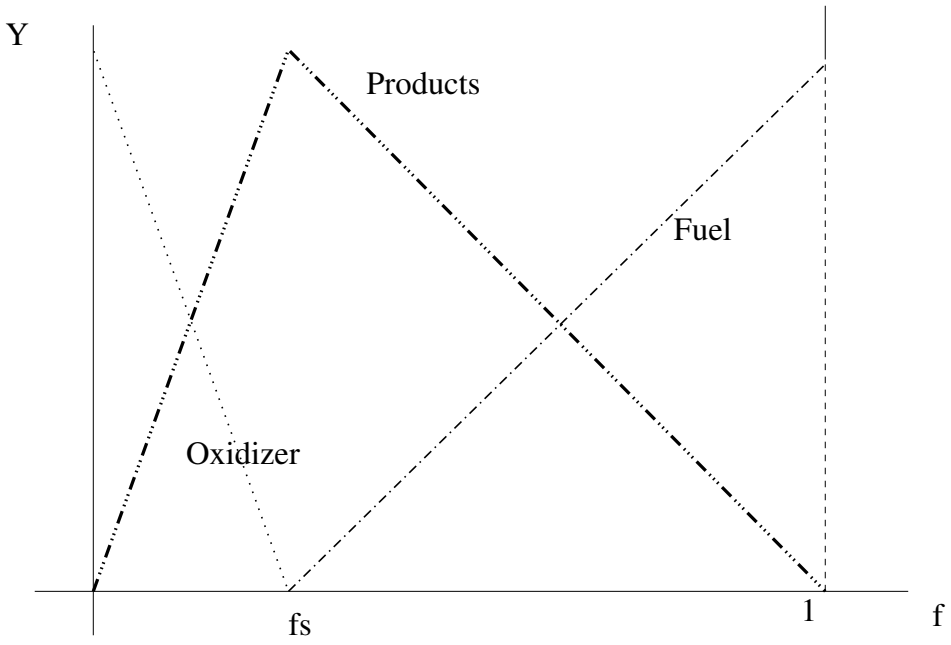


Figure II.9.1: Mass fraction of global species are piecewise linear with mixture fraction.

Where f_{s} is the stoechiometric mixture fraction, Y_s the concentrations in products of the complete reaction of a stoechiometric mixture (in such products, the chemical reaction is no more possible : inert). Beware to distinguish Y_{fuel} mass fraction of a species (depending on f) and Y_{fuel} mass fraction of species in the inlet condition for the fuel stream ($Y_{i,\text{fuel}} = Y_i(1) \quad Y_{i,\text{air}} = Y_i(0)$).
The diffusion model uses two equations for the mixture fraction and its variance. Both of them having no reaction term. The mean and the variance of the mixture fraction are used to fit parameter of a Probability Density Function, with a presumed form, for the mixture fraction. In code_saturne the

shape proposed by [Borghi and Dutoya, 1978] with a rectangle and Dirac's peak is used (subroutines `copdf, cppdf, fupdf`).

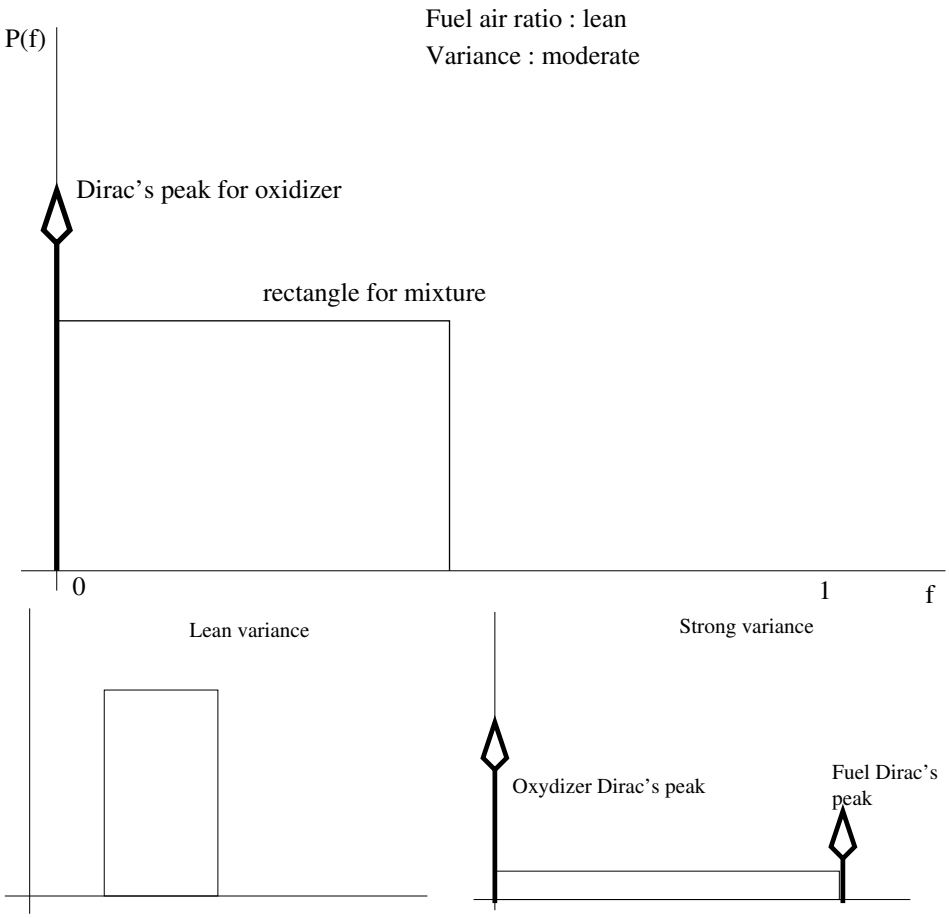


Figure II.9.2: Examples of presumed PDF: cheapest form.

The determination of the mean concentration is done by integrating the local concentration weighted by the probability density function. As a matter of fact, integrating the product of a piecewise linear function by a constant (height of the rectangle) is a very simple exercise: analytic solution is available (the original formulation [Borghi and Dutoya, 1978] which uses β function was much more computationaly expensive).
In adiabatic condition, the specific enthalpy of the mixture is a linear function of the mixture fraction (the enthalpy is not modified by the reaction). As for premixed combustion, the following assumption is done *"the hotter the gases, the worse the heat losses"*, so the enthalpy of pure oxidizer and fuel are supposed to be not modified in permeatic conditions, the enthalpy of products h_s (at the stoechiometric mixture fraction) is an unknown or auxiliary variable. The enthalpy of the mixture is supposed linear piecewise with f (like concentrations but with an unkwnon at f_s) and the resulting mean enthalpy (weighted by PDF) is linear in h_s. Fitting with the equation for the mean enthalpy (which takes in account radiation and other heat fluxes), h_s is determined and, consequently the temperature at fs and the mean temperature can be computed.

As an example of the capabilities of the simple pdf used in code_saturne, computations for the determination of the value of this auxiliary variable are detailed hereafter.

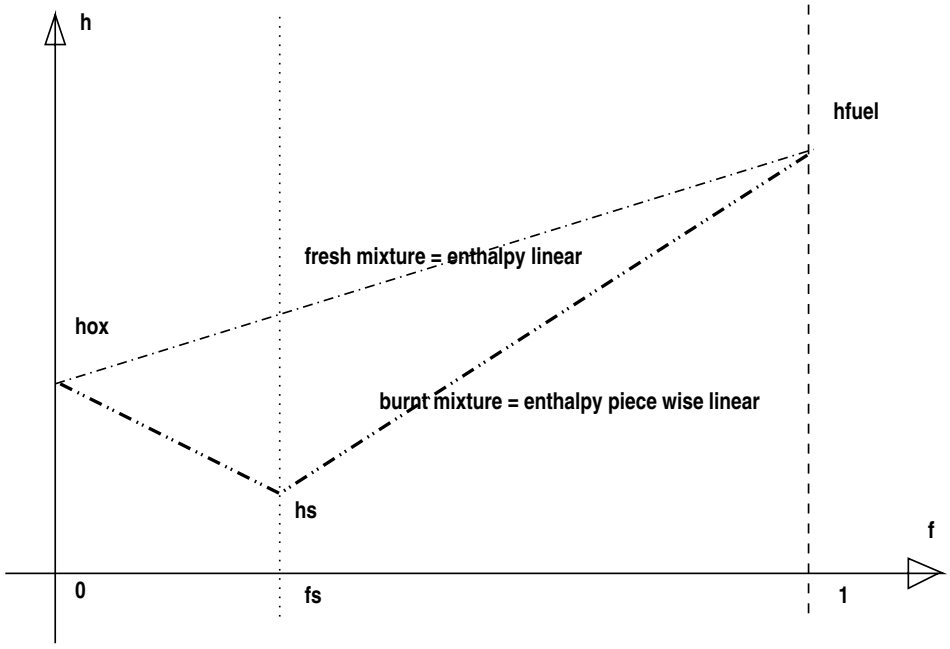


Figure II.9.3: Enthalpy of products is determined to take in account the heat losses.

$$
\begin{aligned}
0 \leq f \leq f_{\rm s} \quad &; \quad h_\ell = h_{\rm Ox} + (\ h_{\rm s} - h_{\rm Ox}\ )\ \frac{f}{f_{\rm s}} \\
f_{\rm s} \leq f \leq 1 \quad &; \quad h_{\rm s} = \frac{f_{\rm s}.\ h_{\rm fuel} - h_{\rm s}}{f_{\rm s} - 1} + (h_{\rm s} - h_{\rm fuel})\ \frac{f}{f_{\rm s} - 1}
\end{aligned}
\tag{II.9.8}
$$

$$
\begin{aligned}
\int_0^1 h(f).P(f)df &= D_0.h_{\rm Ox} + D_1.h_{\rm fuel} \\
&+ \int_{f_{1,\ell} = \min(f_1, f_{\rm s})}^{f_{2,\ell} = \min(f_{\rm s}, f_2)} h_\ell(f).H.df \\
&+ \int_{f_{1,r} = \max(f_1, f_{\rm s})}^{f_{2,r} = \max(f_{\rm s}, f_2)} h_r(f).H.df
\end{aligned}
\tag{II.9.9}
$$

$$
\begin{aligned}
\int_0^1 h(f).P(f)df &= D_0.h_{\rm Ox} + D_1.h_{\rm fuel} \\
&+ H.h_{\rm Ox}.(f_{2,\ell} - f_{1,\ell}) + H.(h_{\rm s} - h_{\rm air})\ \left\{ \frac{f_{2,\ell}^2 - f_{1,\ell}^2}{2\ f_{\rm s}} \right\} \\
&+ H.\frac{f_{\rm s}.h_{\rm fuel} - h_{\rm s}}{f_{\rm s} - 1}.(f_{2,r} - f_{1,r}) + H.(h_{\rm s} - h_{\rm fuel}) \left\{ \frac{f_{2,r}^2 - f_{1,r}^2}{2\ (f_{\rm s} - 1)} \right\}
\end{aligned}
\tag{II.9.10}
$$

$$\begin{aligned}
\int_0^1 h(f).P(f)df &= D_0.h_{\mathrm{Ox}} + D_1.h_{\mathrm{fuel}} \\
&+ \underbrace{H\ h_{\mathrm{Ox}}\ (f_{2,\ell} - f_{1,\ell}) - H\ h_{\mathrm{Ox}}\ \frac{(f_{2,\ell}^2 - f_{1,\ell}^2)}{2\ f_{\mathrm{s}}}}_{H_{Ox}\ ;\ 2 \text{ terms}} \\
&+ \underbrace{H\ \ h_{\mathrm{fuel}}\ \frac{f_{\mathrm{s}}\ (f_{2,r} - f_{1,r})}{(f_{\mathrm{s}} - 1)} - H\ h_{\mathrm{fuel}}\ \frac{(f_{2,r}^2 - f_{1,r}^2)}{2.(f_{\mathrm{s}} - 1)}}_{H_{\mathrm{fuel}}\ ;\ 2 \text{ terms}} \\
&+ \underbrace{H.h_{\mathrm{s}}\ \left\{ \frac{(f_{2,\ell}^2 - f_{1,\ell}^2)}{2\ f_{\mathrm{s}}} - \frac{(f_{2,r} - f_{1,r})}{(f_{\mathrm{s}} - 1)} + \frac{(f_{2,r}^2 - f_{1,r}^2)}{2\ (f_{\mathrm{s}} - 1)} \right\}}_{H^*\ ;\ \text{last terms}}
\end{aligned} \tag{II.9.11}$$

With h_ℓ enthalpy on the **l**ean side of f_{s}, h_r enthalpy on the **r**ich side; D_0 the Dirac's peak in pure air, D_1 Dirac's peak in pure fuel, f_1 begin of rectangle, f_2 end of rectangle, H heigth of rectangle.

This expression for enthalpy includes a last term linear in h_{s} , identification with the transported enthalpy (solved with source term for radiation and so on) allows to determine its value:

$$\int_0^1 h(f).P(f)df = \tilde{h} \quad \Leftrightarrow \quad h_s = \frac{\tilde{h} - [D_0.h_{\mathrm{air}} + D_1.h_{\mathrm{fuel}} + \{H_{\mathrm{Ox}} + H_{\mathrm{fuel}}\}]}{H^*} \tag{II.9.12}$$

9.3.3 Partial premix: Libby Williams models

code_saturne has been the test-bench for some versions of Libby-Williams model [Libby and Williams, 2000], like for the models implemented and then incremented by [Ribert et al., 2004] and [Robin and ???, 2005].

The Libby & Williams model have been developed to address the description of the combustor can of gas turbine in regime allowing a reduction of NOx production using (sometimes very) lean premix. By this way, the combustion occurs at moderate temperatures avoiding the hot spots which are favourable to thermal NOx production. Because of the moderate temperatures, the chemistry is no more so fast and the stability is questionnable. To ensure it a diffusion flame called pilot takes place in the center of the combustor. So, if the main flow is premixed, both pure fuel and pure oxidiser are introduced in the combsutor leading to continuous variation of the equivalence ratio (always the mixture fraction).This situation is clearly out of the range of both EBU and PDF models, moreover the limitation by the chemistry is needed (for stability studies).

Originally, Libby & Williams proposed a presumed PDF made of two Dirac's peaks, Ribert showed that this PDF could be determined with only the mean and the variance of the mixture fraction and a reactive variable (by now, the mass fraction of fuel is used). Then some undeterminations seem awkward and Robin *et al.* propose a four Dirac's peaks PDF whose parameters are determined with the same variables and the solved (transported) covariance (of the reactive variable and the mixture fraction) as an extra solved field. With the condition corresponding to every Dirac's peak, a global chemistry description is used (the source term for every variables is a weighting of the reaction fluxes).

With two peaks distribution, the two-variable PDF is restricted to one line, crossing the mean state and the slope of which is the ratio of variances (choice of the sign is user free, ... but relevant: expertise is needed). The correlation between variables is unity. On this line the distribution is supposed to obey a modified Curl model [Curl, 1963].

With three or four peaks distribution, the whole concentration space is available and the determination of the covariance allows evolution of the correlation (with f and Yf, it has been shown that the correlation is positive in mixing layer and can become negative across the flame: the particle nearer of

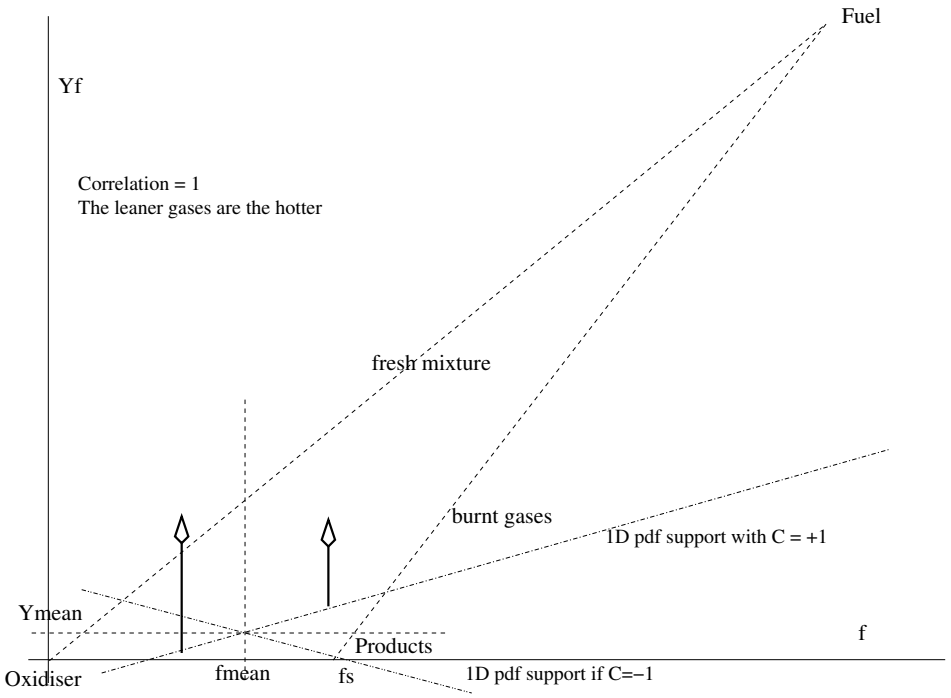


Figure II.9.4: PDF recommended by Libby & Williams: still undetermined.

stoechiometry being able to burn -then destroy Yf- faster than the particles in poor regime).

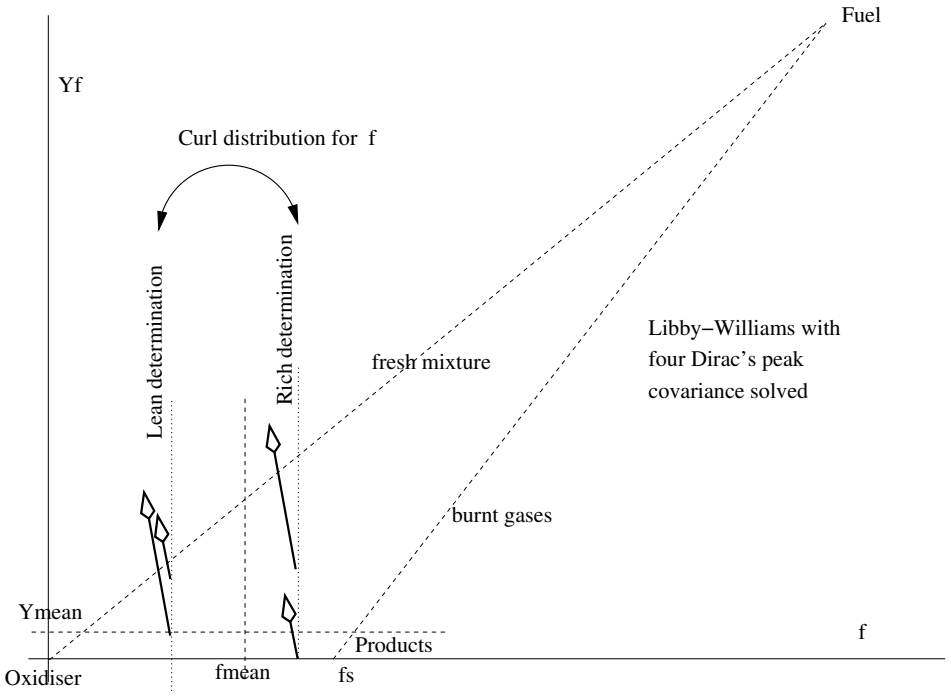


Figure II.9.5: PDF form in LWP approach: succesive modified Curl distributions.

In adiabatic conditions, the temperature can be computed for every pair $(f, Yfuel)$, allowing the determination of the kinetic constant.

As previously, with heat losses, it is assumed that the hottest gases have lost the more (of their thermal enthalpy), the enthalpy of the products at stoechiometric point $(fs, 0)$ is an auxiliary variable, the enthaply being considered as a piecewise bilinear function. Fitting mean transported enthalpy and integrated ones, allows to determine the enthalpy of stoechiometric products, then the enthalpy and temperature in the peaks conditions and, *in fine* the reactions fluxes.

9.3.4 Numerical Recipies for PDF

Some applied mathematics are involved in pdf parameters determination (rectangle and modified Curl) and for integration (species mass fraction, temperature, density and so on).

Some tests, done by [Sapa, 2011] show weak discrepancies between species mass fraction with respect to the shape of the PDF (among Beta, rectangle and peaks, Curl; for similar mean during variance dissipation).

Rectangle and Dirac's peaks probability density function

This type of pdf is used in diffusion flames both for gas combustion or dispersed fuel ones (coal and heavy fuel oil). In gas mixture, the pdf is build for an equivalence ratio for fuel (inert scalar variable) ranging on [0, 1]. For dispersed fuel, due to vaporisation, or pyrolysis, and heterogeneous combustion two or three gaseous fuels are taken in account, each of them having its own inert scalar, so the PDF is build for an inert scalar which is the total of passive scalars for volatiles matter (coal and biomass) or hydrocarbon vapor (heavy fuel oil). The algorithm for pdf parameters determination, can be written in a general form on every variable's range.

If the allowed range for the variable is $[f_{\min}; f_{\max}]$, knowing the mean and variance of the variable allow to determine first the shape (alone rectangle, rectangle and one Dirac's peak at one boundary, two Dirac's peak at boundaries and rectangle) and then, the three pertinent parameters (three conditions given by momenta $m_0 = 1, m_1 = \text{mean}, m_2 = \text{mean}^2 + \text{variance}$).

1. For a lonesome rectangle Dirac's peak intensity is null, the three parameters are: the begining and end values of the rectangle and its heigth.
2. For a rectangle with a Dirac's peak at one boundary (which is determined during the choice of shape), one of the rectangle edge is fixed at this boundary, so the three parameters are : the other rectangle edge, height of rectangle, intensity of the Dirac's peak.
3. For a two Dirac's peak distribution, both rectangle edges are at the boudaries, so the parameters are the rectangle height and the Dirac's peak intensity.

The choice between the four possible forms can be done by previous test on the variance. Defining v_1 and v_2 by:

$$\overline{f} \leq \frac{1}{2} \quad \Rightarrow \quad v1 = \frac{\overline{f}^2}{3} \quad ; \quad v2 = \frac{\overline{f}.(2 - 3.\overline{f})}{3} \tag{II.9.13}$$

$$\overline{f} \geq \frac{1}{2} \quad \Rightarrow \quad v1 = \frac{(1 - \overline{f})^2}{3} \quad ; \quad v2 = \frac{(1 - \overline{f}).(2 - 3.(1 - \overline{f}))}{3} \tag{II.9.14}$$

$$\forall \overline{f} \quad \Rightarrow \quad v1 = \min\left(\frac{\overline{f}^2}{3}; \frac{(1 - \overline{f})^2}{3}\right) \tag{II.9.15}$$

$$\forall \overline{f} \quad \Rightarrow \quad v2 = \max\left(\frac{\overline{f}.(2 - 3.\overline{f})}{3}; \frac{(1 - \overline{f}).(2 - 3.(1 - \overline{f}))}{3}\right) \tag{II.9.16}$$

Depending on the value of variance and naming D_0 and D_1, the Dirac's peak magnitude, f_2 and f_3 the begining and end of the rectangle and h its height, the determination is:

1. if the variance is lower than v_1, the pdf is only made of a rectangle (symetrical with respect to the mean)

$$D_0 = D_1 = 0 \; ; \; f_2 = \tilde{f} - \sqrt{3\widetilde{f''^2}} \; ; \; f_2 = \tilde{f} - \sqrt{3\widetilde{f''^2}} \tag{II.9.17}$$

2. if the variance is greater than v_2, the pdf is made of two Dirac's peak and a rectangle (over all range); (be careful the mean of square is neither the variance nor the square of mean; but the sum of them)

$$f_2 = 0 \; ; \; f_3 = 1 \; ; \; D_0 = 3.\widetilde{f^2} - 4.\tilde{f} + 1 \; ; \; D_1 = 3.\widetilde{f^2} - 2.\tilde{f} \tag{II.9.18}$$

3. if the variance is greater than v_1 and lower than v_2, the pdf is made of only one Dirac's peak (in 0 if the mean is lower than one half in 1 otherwise) and a rectangle.

$$\begin{aligned} \tilde{f} \leq \frac{1}{2} \quad &\Rightarrow \quad D_1 = 0 \; ; \; f_2 = 0 \; ; \; f_3 = \frac{3.\widetilde{f^2}}{2.\tilde{f}} \; ; \; D_0 = 1 - 2.\left(\frac{\tilde{f}}{f_3}\right) \\ \tilde{f} \geq \frac{1}{2} \quad &\Rightarrow \quad D_0 = 0 \; ; \; f_3 = 1 \; ; \; f_2 = \frac{3.\widetilde{f^2} - 4.\tilde{f} + 1}{2.(\tilde{f} - 1)} \; ; \; D_1 = \frac{2.\tilde{f} - 1 - f_2}{1 - f_2} \end{aligned} \tag{II.9.19}$$

4. every time, the height of the rectangle obeys the same relation :

$$h = \frac{1 - D_0 - D_1}{f_3 - f_2} \tag{II.9.20}$$

Curl distribution

The Curl distribution is only composed of two Dirac's peaks. Such a distribution needs four parameters (two positions and two amplitudes), only two moments (mean and variance) are known, the completness is the third, so an extra assumption is needed : the standard Curl distribution assumed amplitudes values (the mean for the richer, the remainder to one for the leaner), a modification of the Curl distribution is given by an extra constraint : use of recurrence relation between Beta function allows to determine the third moment (linked with skewness) and to modify amplitudes of peaks. In this code_saturne release, only regular Curl distribution is implemented and used (discrepancies in species mass fractions using the modified Curl are not worthy of this option introduction). Looking for P_1 and P_2 the amplitudes, and f_1 and f_2 the positions, with constraints from completeness, mean and variance, it comes :

$$P_1 + P_2 = 1 \Rightarrow P_2 = 1 - P_1 \tag{II.9.21}$$

$$P_1.f_1 + P_2.f_2 = \tilde{f} \Rightarrow f2 = \frac{\tilde{f} - P_1.f1}{1 - P_1} \tag{II.9.22}$$

$$P_1.f_1^2 + P_2.f_2^2 = \widetilde{f^2} = \tilde{f}^2 + \widetilde{f''^2} \Rightarrow \tag{II.9.23}$$

$$f_1 = \tilde{f} - \sqrt{\widetilde{f''^2}.\frac{1 - P_1}{P_1}} \tag{II.9.24}$$

$$f_2 = \tilde{f} + \sqrt{\widetilde{f''^2}.\frac{P_1}{1 - P_1}} \tag{II.9.25}$$

f_1 and f_2 may be inside [0, 1], the first proposal by Curl (in the context of liquid-liquid extraction, the interfacial mass flux does not modify mass of each phases) is:

$$\begin{aligned} &P_1 = 1 - \tilde{f}; P_2 = \tilde{f} \\ &f_1 = \tilde{f}.\left\{ 1 - \sqrt{\frac{\widetilde{f''^2}}{\tilde{f}.(1 - \tilde{f})}} \right\} \end{aligned} \tag{II.9.26}$$

Obviously, maximal value of variance induces Dirac's peak positionned at boundaries.

Formulae in (II.9.21) can be used to determine the peak's magnitude if any extra condition is retained. In the context of pdf, the Beta function have the fame of fairly good representation of micro-mixing, so the third momentum of a Beta distribution is easy to determine (thanks to recurrency), used as a constraint for a modified Curl model, it comes :

$$\widetilde{f^3} = \frac{\alpha+2}{\alpha+\beta+2}.\widetilde{f^2} \tag{II.9.27}$$

$$\widetilde{f''^3} = 2.\widetilde{f''^2}^2.\frac{1-2.\tilde{f}}{\tilde{f}.(1-\tilde{f})+\widetilde{f''^2}} \tag{II.9.28}$$

$$P_1^2\left\{4+\frac{\widetilde{f''^3}^2}{\widetilde{f''^2}^3}\right\}-P_1\left\{4+\frac{\widetilde{f''^3}^2}{\widetilde{f''^2}^3}\right\}+1=0 \tag{II.9.29}$$

Some numerical evaluations don't show large discrepancies (among mass fraction or temperature) so this option is not currently available.

9.4 Coal, Biomass, Heavy Oil combustion

9.4.1 Introduction

Pulverised coal combustion is described (excluding grid burning) allowing the use of mixture of coals (or of coal and biomass) and a description of granulometry (as many classes of initial diameter as wished). After a particle enters the furnace, radiation increases its temperature.

1. As particle's temperature increases, evaporation of free water (if any) begins. During evaporation, the vapor pressure gradient extract some water from the particle. The interfacial mass flux brings some mass, water and enthalpy (computed for water vapour) then latent heat is taken from the particle's enthalpy so the heating is slowed (during evaporation, the water can't reach the boiling point). Fuel oil is supposed to be dry.

2. After drying is achevied, the temperature reaches higher level, allowing the pyrolysis phase to take place. The pyrolysis is described by two competitive reactions: the first one with a moderate activation energy is able to free peripherals atoms group from skeleton leaving to light gases and a big amount of char; the second one with an higher activation enrgy is able to break links deeper in the skeleton leaving to heavier gases (or tar) and less char (more porous). So a complete description needs two sets of three parameters (two kinetics ones and a partitioning one):

$$\text{Coal} \quad = \quad (k_{01},\ T_{01}) \quad \Rightarrow \quad Y_1\ \{\text{Light Volatiles }\} \quad + \quad (1-Y_1)\ \{\text{Char}\},$$

$$\text{Coal} \quad = \quad (k_{02},\ T_{02}) \quad \Rightarrow \quad Y_2\ \{\text{Heavy Volatiles }\} \quad + \quad (1-Y_2)\ \{\text{Char}\},$$

   where Y_1, the partitionning (or selectivity) factor of the "*low temperature*" reaction is less than Y_2, the "*high temperature*" one. A practical rule is to consider that the same hydrogen can bring twice more carbon by the second reaction than by the first one. When ultimate analysis are available both for coal and for char, it is relevant to check partioning coefficient (Y_i) and composition of volatiles matters (mainly ratio of Carbon monoxide and C/H in the hydrocarbon fraction): assumptions on volatiles composition gives partitionning coefficients; assumptions on Y_i determine volatiles equivalent formulae. Pyrolisis interfacial mass flux brings energy of volatile gases (computed at the particle's temperature) in wihch the formation of enthalpy of gaseous species differs from the coal one's, as a result, the enthalpy for pyrolisis reaction (the most ofen, moderate) is taken from particle energy.

   The heavy fuel oil undertakes a set of physico-chemical transformation: light hydrocarbons can evaporate while the heaviest undergo a pyrolisis. With a few data, only a temperature range is available for mass loss of droplets: the heat flux is shared out between warming of the remaining liquid and evaporation enthalpy. At the very end of theses processes a solid particle is leaved, mainly made of a porous carbon similar to char.

3. After pyrolysis and evaporation, when every volatiles are burnt, oxygen is able to reach the surface of the char particle. So heterogenous combustion can take place: diffusion of oxygen from bulk, heterogeneous reaction (kinetically limited) and back diffusion of carbone monoxide. The heterogeneous oxidation interfacial mass flux is the difference of an incoming oxygen flux and an outcoming carbon monoxide mass flux, each of them at their own temperature. The incoming oxygen has a zero valued formation enthalpy (reference state) and the outcoming carbon monoxide has a negative formation enthalpy, as a result, the enthalpy liberated by the first oxidation of carbon is leaked in the particle energy, contributing to its heating. The heterogenous combustion is complete if all the carbon of the char particle is converted, leaving an ash particle. Unburnt carbon can leave the boiler as fly ash. The heterogeneous reaction is written as following:

$$\text{Char} + \frac{1}{2}O_2 = (k_{0,\text{het}}\ ,\ T_{0,\text{het}}) \ \Rightarrow \ CO$$

4. In the same way, after pyrolysis vanishes, gasification can take place by :

$$\begin{aligned} \text{Char} + CO_2 &= (k_{0,\text{gc}} \quad , \quad T_{0,\text{gc}}) \quad \Rightarrow \quad 2CO, \\ \text{Char} + H_2O &= (k_{0,\text{gw}} \quad , \quad T_{0,\text{gw}}) \quad \Rightarrow \quad CO + H_2. \end{aligned}$$

This version is able to deal with many class of particles $N_{classes}$, each class beeing described by an initial diameter and a constituting coal. Every coal, among `nchar` is described by a complete set of parameters: immediate and ultimate analysis, low heating value (at user choice on raw, dry or pure) and kinetic parameters (*for the two competitive pyrolysis reaction and for heterogeneous reaction*). This allows to describe the combustion of a mixture or coals or of coal and every material following the same evolution kinetics (woods chips ...). It is, obviously, possible to mix fuels with (very) different proximate analysis, like dry hard coal and wet biomass.

Subroutines allowing the user to describe inlets are dedicated to standard combustion: some inlets are for coal (eventually blend)and a gaseous media, others for oxidizers. If needed, a deeper modification allows to describe co-combustion of coal and some gases ... described as volatile matter from a coal.

The heavy fuel oil injection is described by a thermodynamic data file and granulometry, neither blend nor coal / oil mixing is possible.

9.4.2 Enhancement of diffusion turbulent reaction for two phase combustion

With a Probability Density Function and the assumption of concentrations piecewise linear vs. the variable, it is quite easy to integrate and found the mean concentrations. For gas diffusion flame this is done by `d3pphy, d3pint`, it seems more relevant to explain the algorithm in a more complicated case: for coal, biomass and heavy fuel oil, in `multiphase gas combustion five reactions` can be considered.

1. 1) gases issued from slow phenomena (*vs.* turbulent mixing) as heterogeneous combustion and gasification, both by CO_2 and H_2O are mixed with various oxidizers; if any gasification by H_2O has been undertaken, some H_2 is released. This (an industrial combustor is not a gasifier) amount is supposed to recombine as a reaction prior to mixing and main turbulent combustion.

2. 2 and 3) Coal is assumed to undergos two competitves pyrolysis reactions, the first releasing organic compound summarized as CH_{x1}, the second releasing CH_{x2} (with $x1 > x2$), both of them releasing CO. The first reaction to occur in the gas phase is the partial dehydrogenation (lowering saturation) of CH_{x1} to produce water vapor and CH_{x2}. Then the CH_{x2} (produced by pyrolisis or by CH_{x1} partial oxydation) is converted to water vapor and carbon monoxide.

3. 3 and 4) Heavy fuel oil is supposed to undergos a progressive evaporation, releasing a fuel vapor CH_x (obviously unsaturated, close to CH_2), CO, H_2S and a char particle. Two reactions are supposed to succeed. First, the conversion of CH_x to water vapor and CO. Then the oxidation of H_2S to water vapor and SO_2.

4. 5) if conversion of CO to CO_2 is assumed to be fast, this complete reaction is also ruled by the variance dissipation. (Be careful, some CO (from heterogeneous oxidation and gasification) is still mixed with the local mean oxidiser).

Both for coal and heavy fuel oil, the assumption of a diffusion flamelet surrounding the particles is done. All of the reducing gases issued from fast phenomenon are supposed mixed (to constitue a local mean fuel) and the diffusion flamelet takes place between this mixture and the mixture of oxidisers (air, oxygen, recycled flue gas) and gases issued from slow phenomenon: water vapor (from drying), carbon monoxide (from heterogeneous combustion and gasification of char if CO oxidation is not fast),

hydrogen (product of gasification by water vapor); then the "mean local oxidizer" is no more unreactive and the recombination of hydrogen have to be done first. Depending on the composition of mean local fuels and oxidizer, stoechiometries can be computed for the five (or four) successive reactions, molar composition can be easily computed and weighted by the pdf (built on [0 , 1] for the sum of fast fuels).

If carbon monoxide final oxidation is not supposed to be fast, after this turbulent reaction, the amount of carbon dioxide is null, an equilibrium value is computed (with respect to the total amount of carbon and oxygen and to the enthalpy) and the value of carbon dioxide transported is compared with the equilibrium, according to direction of the discrepancy, a relaxation term is computed with a caracteristic time (oxidation's one if the mass fraction is below the equilibrium, dissociation's one otherwise -situation encountered with exhaust gas recycling).

In this release, coal is not supposed to contain any sulphur and heavy fuel oil is supposed to release only one (unsaturated) hydrocarbon. The general subroutine is then feeded with some nil values (*e.g.* in heavy fuel computation, the tracer $f1$ dedicated to saturated hydrocarbon is not resolved, and the call of the subroutine is done with a zero scalar).

This structure is supposed to allow further developments: coal, or biomass, with sulphur, detailed mechanism for heavy fuel oil decomposition (leading to both CH_{x1} and CH_{x2}), and so on.

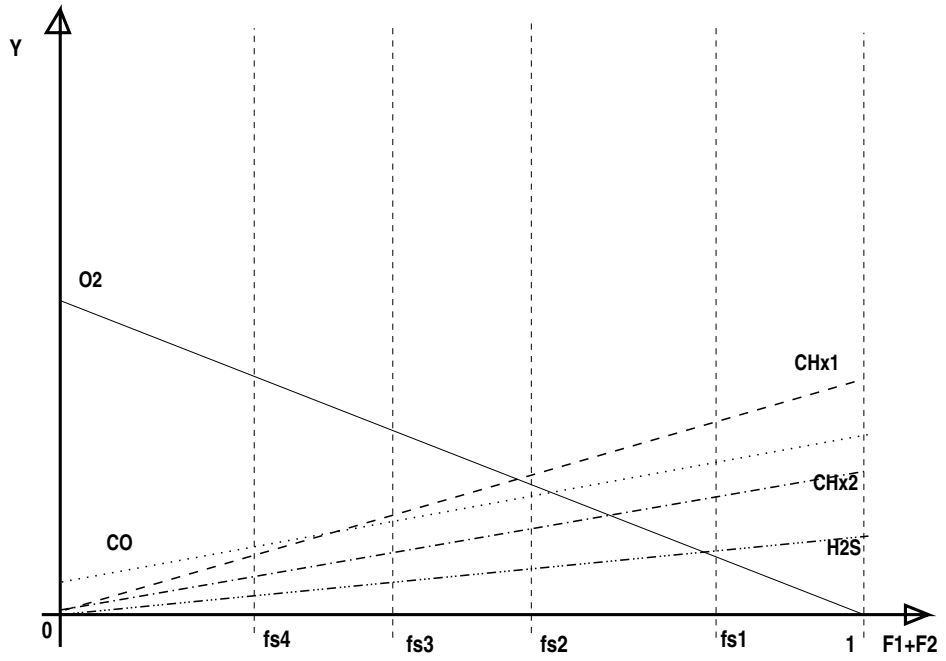


Figure II.9.6: Condensed Fuels before any gas combustion.

Before reaction between gases, only exist species coming from inlets or interfacial source term: CO in mean local fuel (*i.e.* $f_1 + f_2 = 1$) comes from devolatilisation, CO in mean local oxidizer (*i.e.* $f_1 + f_2 = 0$) comes from heterogeneous reactions of char.

During pulverised coal combustion, two kinds of volatile matters are considered and the sketch of concentrations during the three successive reactions is quite similar.

Every reaction (but, possibly the final conversion of CO to CO_2) are supposed to be fast compared to the turbulent mixing, but among these reactions some can be faster; here, a priority rule to access oxygen is established (the more eager –for oxygen– the specy, the faster the reaction).

The first reaction is a partial dehydrogenation of the light volatile CH_{x1} to form the species caracteristic of heavy volatile CH_{x2}: in fs1, all of CH_{x1} (issued from the low temperature pyrolisis reaction) is

converted, and the CH_{x2} (issued from the high temperature pyrolisis reaction) is incremented.

$$CH_{x1} + \frac{x1 - x2}{4} O_2 \Rightarrow CH_{x2} + \frac{x1 - x2}{2} H_2O.$$

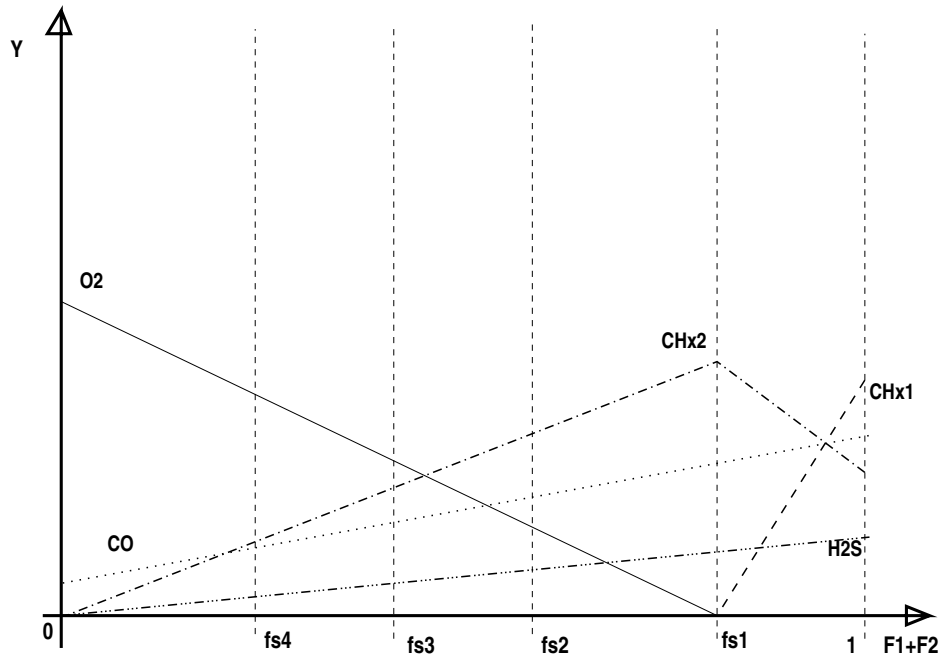


Figure II.9.7: Condensed Fuels after hydrocarbons conversion.

The oxygen and the hydrocarbon vapor have linear concentrations in $f(1+2)$ on [0, 1]. As long as the stoechiometry of the reaction is known, a simple equation allows to determine f_{s2} the stoechiometric point for the second reaction (where both oxygen and hydrocarbon vanish). The second reaction is the conversion of some hydrocarbon vapor to carbon monixide and water vapor (not plotted in an optimistic attempt to lighten the sketch).

$$CH_{x2} + \frac{2 + x2}{4} O_2 \Rightarrow CO + \frac{x}{2} H_2O.$$

Then the rich area can't undergo any reaction (no oxygen available) if the PDF(f) is not zero before Fs2, then some CH_x is unburnt.

Some H_2S can be converted to SO_2, the carbon monoxide existing between fs2 and 1 is protected from oxidation (the two first reactions have destroyed the free oxygen). Like previously, oxygen and hydrogen sulphide have concentrations linear in f on $[0, fs2]$ as long as the stoechiometry of the reaction is known, a simple equation allows to determine $fs3$ the stoechiometric point for the third reaction (where both oxygen and hydrogen sulphide vanish).

$$H_2S + \frac{3}{2} O_2 \Rightarrow SO_2 + H_2O.$$

If the final conversion of carbon monoxide to carbon dioxide is assumed fast (with respect to variance dissipation) a last reaction is taken in account:

$$CO + \frac{1}{2} O_2 \Rightarrow CO_2.$$

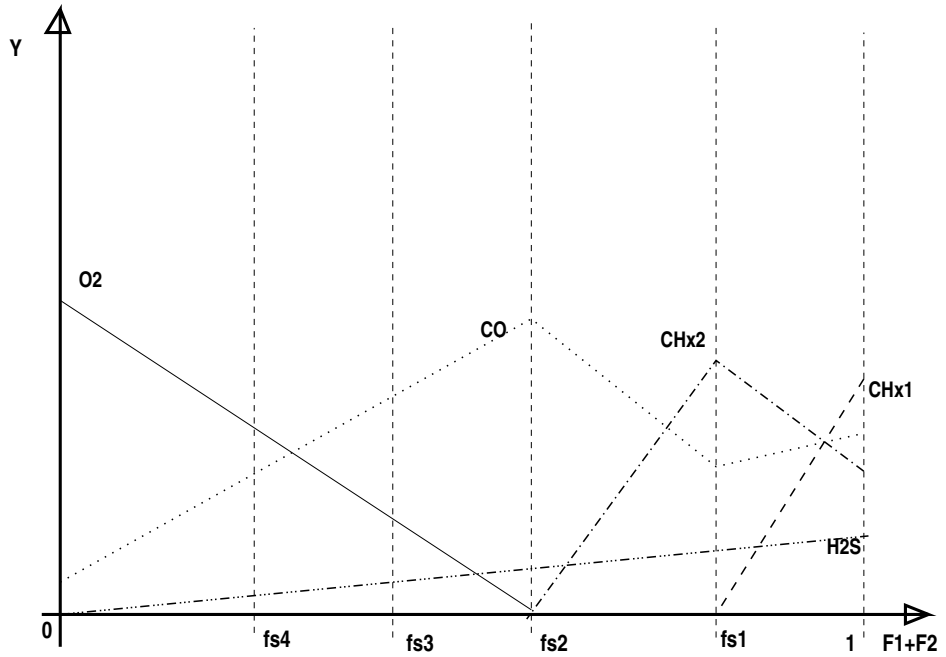


Figure II.9.8: Condensed Fuels after hydrocarbons oxidation.

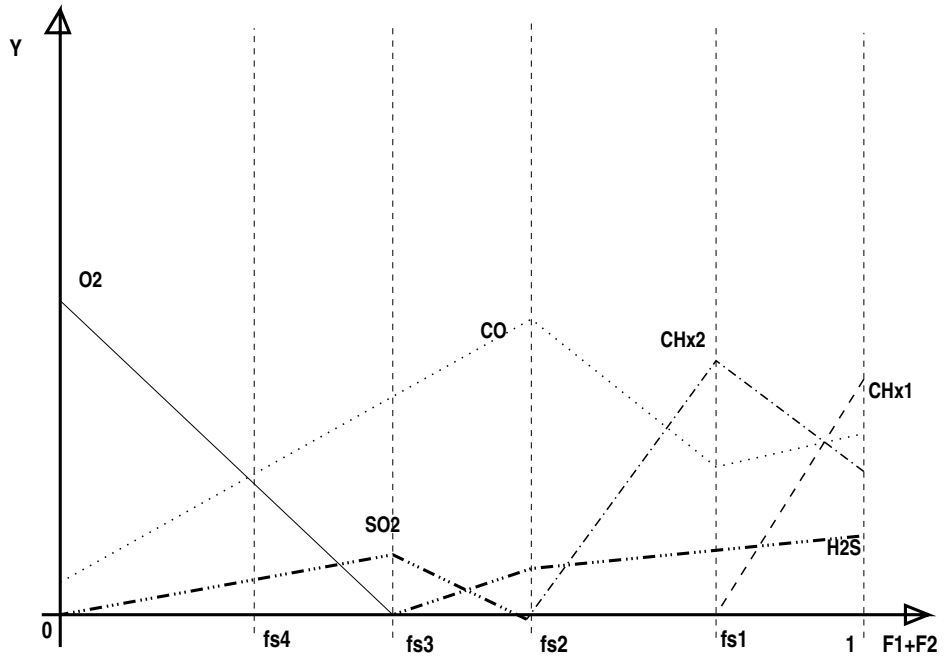


Figure II.9.9: Condensed fuels after H2S oxidation.

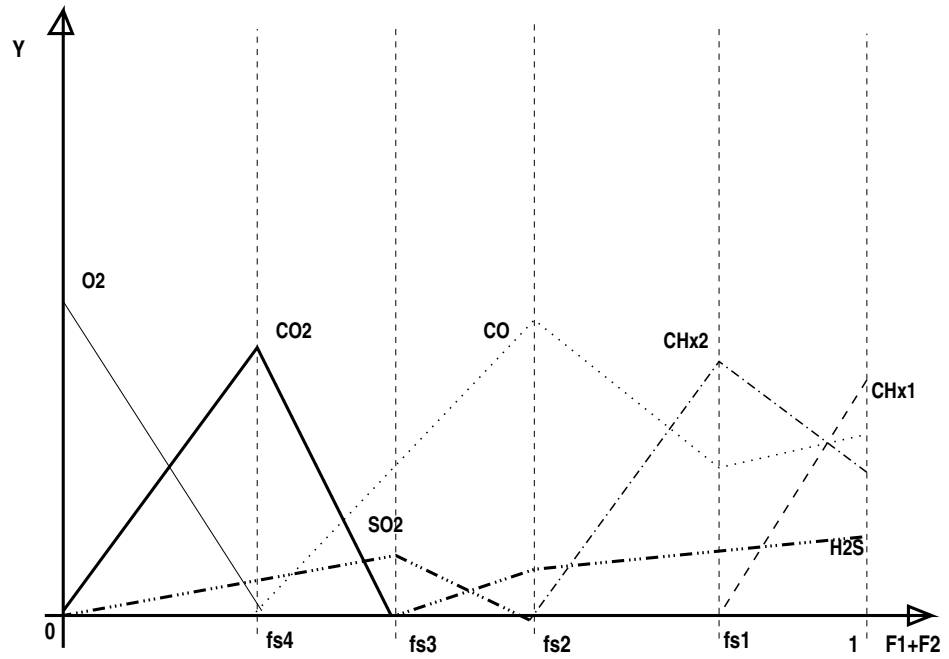


Figure II.9.10: Condensed Fuels after final oxidation of CO.

Comparisons of the PDF rectangle hedges $[f_{\text{deb}}, f_{\text{fin}}]$ and remarkable composition points $[0, f_{s4}, f_{s3}, f_{s2}, f_{s1}, 1]$ allows a simple integration: 1) Dirac's peak intensity are used to weight composition at boundaries, 2) the piece linear part is integrated with analytical formulae on each band:

1. rich range, here existing species with the higher calorific value:
   CH_x (in fuel case) or CH_{x2} (in coal case): $[\max(f_{\text{deb}}, f_{s1}); \min(f_{\text{end}}, 1)]$
2. upper-middle range CH_{x1} conversion: $[\max(f_{\text{deb}}, f_{s2}); \min(f_{\text{end}}, f_{s1})]$
3. middle range H_2S conversion: $[\max(f_{\text{deb}}, f_{s3}); \min(f_{\text{end}}, f_{s2})]$
4. working range, carbon monoxide consumption frees enthalpy : $[\max(f_{\text{deb}}, f_{s4}); \min(f_{\text{end}}, f_{s3})]$
5. poor range, only products and oxidisers: $[\min(f_{\text{deb}}, f_{s4}); \min(f_{\text{end}}, f_{s4})]$

For each band (eg. $[f_{si}, f_{sj}]$) concentrations can be written:

$$Y_e = Y_e(f_{si}) + \frac{f - f_{si}}{f_{sj} - f_{si}}.\ \left(Y_e(f_{sj}) - Y_e(f_{si})\right).$$

Integration on the band [b1 , b2] (obviously $b1 \geq f_{si}$ and $b2 \leq f_{sj}$) gives the increment:

$$Y_e := Y_e + h_{\text{REC}}\ (b2 - b1).\left[\frac{Y_e(f_{si}).\ f_{sj} - Y_e(f_{sj}).\ f_{si}}{f_{sj} - f_{si}} + \frac{Y_e(f_{sj}) - Y_e(f_{si})}{f_{sj} - f_{si}}.\ \frac{b1 + b2}{2}\right],$$

where h_{REC} is the height of the PDF's rectangle.

9.4.3 Specification of pyrolysis

Coal is currently known by its proximate and ultimate analysis. Ultimate analysis of char can be known or assumed pure carbone. The point is to determine the amount of volatile matter and their composition; the following assumptions are every time done:

1. sulphur is released as hydrogen sulphide (H_2S),
2. nitrogen is released both in HCN and NH_3, with a ratio which is an user's prescription

Three ways are available:

1. The volatile matter content determined during the proximate analysis is supposed to be representative of Y_1 selectivity in volatile of the first reaction (Kobayashi description involves two parallel reactions, the first has a low activation energy and produces ligth volatile matter, the second one has a high activation energy and produces heavy volatile matter). Oxygen is released as carbon monoxide. No water steam (linked water) among volatile matter. So the formulae for the mean hydrocarbon is determined as $x1$ in CH_{x1}. The heavy, unsaturated, volatile issued from the second reaction are caracterised by $x2$ as an half of $x1$; with the same assumption for oxygen, Y_2 can be computed. User has to check for $x1$ and $x2$ likelihood (between 1 and 4).
2. When the proximate analysis is not known, $x1$ is assumed to be equal to four (methane is a fairly good model for light volatiles) and $x2$ is assumed to be equal to 2 (ethylene and other species with double bound are good models for unsaturated species), then selectivites Y_1 and Y_2 can be deduced ... and checked (under one).
3. Large amount of oxygen appears in the ultimate analysis of biomass and low rank coal (lignite or peat) then linked or bounded water is released during pyrolisis (a chemical mechanism taking place at higher temperature than the physical drying which releases the "free" water). In this case, an extra parameter have to be determined (number of water molecules released during pyrolisis), so the user may stipulate both x (in the formulae for hydrocarbon CH_x, as previously 4 and 2 respectively) and Y (the selectivity in volatile matter, the proximate analysis set Y_1 and Y_2 is assumed from empirical criterion, *e.g.* $(1+Y_1)/2$).

Detail of computation: From ultimate analysis of coal (or heavy fuel oil) and char (if ultimate analysis of char is lacking, the pure carbon assumption is welcome), global formulae for the "monomer" (refering to one carbon, so ch...cs and kh...ks are easy to compute) can be deducted. Then, the reaction (pyrolisis and / or evaporation) transforming the original fuel in a mixture of gaseous ones and residual char can be summarized as:

$$\begin{aligned} CH_{ch}O_{co}N_{cn}S_{cs} \quad \Rightarrow \quad & a.CH_x + b.CO + c.H_2O + d.H_2S + e.HCN + f.NH_3 \\ & +(1-a-b-e).CH_{kh}O_{ko}N_{kn}S_{ks}. \end{aligned}$$

The stoechiometric coefficient for char monomer beeing deducted from Carbon conservation, five equation can be written: four for the conservation of elements (H, O, N, S) and one defines the gas selectivity.

$$\begin{aligned} \text{Hydrogen budget} \quad ch &= a.x + 2.c + 2.d + e + 3.f + (1-a-b-e).kh, \\ \text{Oxygen budget} \quad co &= b + c + (1-a-b-e).ko, \\ \text{Nitrogen budget} \quad cn &= e + f + (1-a-b-e).kn, \\ \text{Sulphur budget} \quad cs &= d + (1-a-b-e).ks, \\ \text{mass ratio of gases or selectivity} & \\ a.(12+x) + b.28 + c.18 + d.34 + e.27 + f.17 &= Y.(12 + ch + 16.co + 14.cn + 32.cs). \end{aligned}$$

But, eight unknown are involved (a, b, c, d, e, f, x, Y), so three extra conditions are needed to solve the linear system (if ax is considered as an auxiliary unknown instead of x):

1. User assumes the repartition between nitrogenated species by fixing reference numbers ei and fi. The equation $ei.f - f_i.e = 0$ can be added to the system.

2. if the proximate analysis can be used to determine the ratio of gases issued from an high rank coal or fuel, Y is know and c (number of water molecules issued from decomposition) can be assumed nil. Equations $Y = Y_i$ and $c = 0$ are added.

3. without relevant information about the selectivity, assumption have to be done on x (is the released hydrocarbon saturated or not?), and c can be again assumed nil. Equations $a.x_i - ax = 0$ and $c = 0$ are added.

4. oxygenated fuels (biomass, lignin) contains bounded water to be released during pyrolisis, so proximate analysis (for Y) and assumption about the kind of hydrocarbon (for x) are needed. Equations $a.x_i - ax = 0$ and $Y = Y_i$ are added.

Then a 8 times 8 linear system is defined and can be solved by regular algorithm.

9.4.4 Specification of granulometry

User has to choose the initial diameter of different classes and the sharing out of the inlet flow. The distribution is often known by some diameters (quantile or sieves) from which parameters of an assumed Rossin-Ramler law can be fitted (least squares). Then choosing the number of classes and flow partition allows computation of the initial diameter of each class. Obviously, the finest particles or droplets are responsible for the ignition and stability of the flame and the biggest ones are responsible for unburnt carbon in ash. So two common descritions are ten classes, each of them with a tenth of the flow, or five classes with (0.1 , 0.2 , 0.4 , 0.2 , 0.1) of the flow. The second way is nearly two times cheaper (in computer time) but includes the same extreme diameters.

By definition of the Rossin-Ramler law as used in granulometry, the mass fraction associated with particles finer than a diameter obeys:

$$P(\mathrm{d}_i) = 1 - \exp\left[-\frac{\mathrm{d}_i}{D_m}^n\right],$$

where, surprinsgly, n is not an integer (but a real) and D_m is the median diameter. When only two data are available (pass through two sieves, extreme deciles or quartiles) the determination of Rossin-Ramler law parameters is direct:

$$P(\mathrm{d}_1) = 1 - \exp\left[-\frac{\mathrm{d}_1}{D_m}^n\right], \tag{II.9.30}$$

$$P(\mathrm{d}_2) = 1 - \exp\left[-\frac{\mathrm{d}_2}{D_m}^n\right]. \tag{II.9.31}$$

The logarithm forms of (Eqs. II.9.30-II.9.31) are the following expressions:

$$\left(\frac{\mathrm{d}_1}{D_m}\right)^n = -\log(1 - P_1), \tag{II.9.32}$$

$$\left(\frac{\mathrm{d}_2}{D_m}\right)^n = -\log(1 - P_2). \tag{II.9.33}$$

The logarithm forms of (Eqs. II.9.32-II.9.33) are:

$$n.\,[\,\log(\mathrm{d}_1) - \log(D_m)\,] = \log[\,-\log(1 - P_1)\,], \tag{II.9.34}$$

$$n.\,[\,\log(\mathrm{d}_2) - \log(D_m)\,] = \log[\,-\log(1 - P_2)\,]. \tag{II.9.35}$$

Using the expressions (Eqs. II.9.34-II.9.35) we obtain:

$$n = \frac{\log\left[\dfrac{\log(1 - P_1)}{\log(1 - P_2)}\right]}{\log\left[\dfrac{\mathrm{d}_1}{\mathrm{d}_2}\right]} \tag{II.9.36}$$

and D_m is easily deduced.

When more data are available, the second logarithm relation gives a cloud of couple $(\log(\mathrm{d}_i)\ ,\ \log[\ -\log(1-P_i)\ ]\ )$ among which a linear fit is looked for $(n, n.\log(D_m))$:

$$\begin{aligned} n.\log(\mathrm{d}_i) - [\ n.\log(D_m)\ ] &= \log[\ -\log(1-P_i)\ ], \\ a.x_i + b &= y_i. \end{aligned} \tag{II.9.37}$$

Least square formulae are then used ... after a data transformation (two logarithm) relevant for the distance.

$$a = \frac{N\sum_{i=1}^{N} x_i.y_i - \sum_{i=1}^{N} x_i.\sum_{i=1}^{N} y_i}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}, \tag{II.9.38}$$

where N is the number of data.

$$n = \frac{N\sum_{i=1}^{N} \log[\mathrm{d}_i].\log[\ -\log(1-P_i)\ ] - \sum_{i=1}^{N}\log(\mathrm{d}_i).\sum_{i=1}^{N}\log[\ -\log(1-P_i)\ ]}{N\sum_{i=1}^{N}\log(\mathrm{d}_i)^2 - \left(\sum_{i=1}^{N}\log(\mathrm{d}_i)\right)^2},$$

$$b = \frac{\sum_{i=1}^{N} y_i.\sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} x_i.\sum_{i=1}^{N} x_i.y_i}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2},$$

$$-n.\log(D_m) = \frac{\sum_{i=1}^{N}\log[\ -\log(1-P_i)\ ].\sum_{i=1}^{N}\log(\mathrm{d}_i)^2 - \sum_{i=1}^{N}\log(\mathrm{d}_i).\sum_{i=1}^{N}\log(\mathrm{d}_i).\log[\ -\log(1-P_i)\ ]}{N\sum_{i=1}^{N}\log(\mathrm{d}_i)^2 - \left(\sum_{i=1}^{N}\log(\mathrm{d}_i)\right)^2}.$$

After this laborious identification of parameters (done using Excel), the determination of the mean diameter of each mass class is obtained from the definition of the rossin-Ramler law:

$$\mathrm{d}_i = D_m\ \ [-\log(1-P_i)]^{\left(\frac{1}{n}\right)}. \tag{II.9.39}$$

As an example, if the user chooses the (recommended) sharing out $[0.1, 0.2, 0.4, 0.2, 0.1]$, the corresponding diameter are deduced from the *mean cumulated mass* as:

$$\Delta_1 = 0.1 \ ;\ \Sigma_1 = 0.1 \ ;\ 1 - MCM_1 = 1 - \frac{0.1}{2} = 0.95 \ \Rightarrow\ \mathrm{d}_1 = Dm\ [\ -\log(0.95)\ ]^{\left(\frac{1}{n}\right)},$$

$$\Delta_2 = 0.2 \ ;\ \Sigma_2 = 0.3 \ ;\ 1 - MCM_2 = 1 - \frac{0.1+0.3}{2} = 0.80 \ \Rightarrow\ \mathrm{d}_2 = Dm\ [\ -\log(0.80)\ ]^{\left(\frac{1}{n}\right)},$$

$$\Delta_3 = 0.4 \ ;\ \Sigma_3 = 0.7 \ ;\ 1 - MCM_3 = 1 - \frac{0.3+0.7}{2} = 0.50 \ \Rightarrow\ \mathrm{d}_3 = Dm\ [\ -\log(0.50)\ ]^{\left(\frac{1}{n}\right)},$$

$$\Delta_4 = 0.2 \ ;\ \Sigma_4 = 0.9 \ ;\ 1 - MCM_4 = 1 - \frac{0.7+0.9}{2} = 0.20 \ \Rightarrow\ \mathrm{d}_4 = Dm\ [\ -\log(0.20)\ ]^{\left(\frac{1}{n}\right)},$$

$$\Delta_5 = 0.1 \ ;\ \Sigma_5 = 1.0 \ ;\ 1 - MCM_5 = 1 - \frac{0.9+1.0}{2} = 0.05 \ \Rightarrow\ \mathrm{d}_5 = Dm\ [\ -\log(0.05)\ ]^{\left(\frac{1}{n}\right)}.$$

With such a symetrical mass distribution, the diameter of the central class is the median diameter of the Rossin-Ramler (*i.e.* half of the mass is contained in more tiny particles).

9.4.5 Special attention paid to variance

With the gaz phase combustion model, everything is quite simple: two variables are relevant, the mean and the variance of the mixture fraction. With the two phase combustion model, a lot of mixture fractions are defined and the pdf model is constructed for the sum of the two mixture fractions related with volatiles. So the source term related with the square of the gradient of the mean can't be computed as for regular variance (the `gradcel` subroutine is called for the sum).

In code_saturne homogeneous two phase flow, only one velocity is defined, and all variables refer to the bulk (sum of gaseous and condensed phase), but the pdf has to be defined only in the gas phase. So phasic mean and variance have to be defined, with the special difficulties of variables undefined in the condensed phase (f_i is a mixture fraction in gas on mixture: kg coming from i/kg of mixture):

$$\widetilde{f} = X_1.f^* + X_2.0 \quad \Rightarrow \quad f^* = \frac{\tilde{f}}{1-X_2}, \tag{II.9.40}$$

$$\widetilde{f^2} = X_1.\left(f^{*2} + f''^{2*}\right) + X_2.0 \quad \Rightarrow \quad f^{*2} + f''^{2*} = \frac{\widetilde{f^2}}{1-X_2}, \tag{II.9.41}$$

$$f''^{2*} = \frac{\widetilde{f^2}}{1-X_2} - \left(\frac{\tilde{f}}{1-X_2}\right)^2, \tag{II.9.42}$$

$$f''^{2*} = \frac{(\tilde{f})^2 + \widetilde{f''^2}}{1-X_2} - \left(\frac{\tilde{f}}{1-X_2}\right)^2, \tag{II.9.43}$$

$$f''^{2*} = \frac{\widetilde{f''^2}}{1-X_2} - \frac{X_2.(\tilde{f})^2}{(1-X_2)^2}, \tag{II.9.44}$$

$$f''^{2*} = \frac{\widetilde{f''^2} - \dfrac{X_2}{X_1}.(\tilde{f})^2}{X_1}. \tag{II.9.45}$$

Only this phasic variance (the part of the variance in the gas phase) is able to dissipate: so, the second part of the source term for variance has to be modified.

Last but not least, mass flux crossing the interface (pyrolisis fluxes) are made of pure volatile matter and mixes with a gas at any value of mixture fraction mean. So the interfacial flux constitutes a source term for the mean and for the variance, following Escaich [Escaich, 2011], the closure would be:

$$S_{\widetilde{f''^2}} = \Gamma.\left(f_{\text{CL}} - \tilde{f}\right).\left(2.f_\Gamma - f_{\text{CL}} - \tilde{f}\right), \tag{II.9.46}$$

in which, Γ is the mass flux, f_Γ is the value of the mixture fraction in the flux (1 for pyrolisis or fuel evaporation), f_{CL} is the value of the mixture fraction in the boundary layer ... to be closed by a relevant assumption: from laminar or turbulent diffusion (to do in `coal_variance_source_term` or `fuel_variance_source_term`, not available for regular users).

With an assumption of laminar diffusion around each particle able to transport the mass flux:

$$f_{\text{CL}} = 1 - (1-\tilde{f})\exp\left[\frac{\Gamma}{2\pi D d\rho}\right].$$

When mass flux is huge, the diffusion can't stay regular around each inclusion: if the variance is maximal (intermittency assumption), the boundary layer can't be no more distinguished from the mean, if the variance vanish, the boundary layer is quiet and made of outing gases:

$$f_{\text{CL}} = f_\Gamma + (\tilde{f} - f_\Gamma)\,\frac{\widetilde{f''^2}}{\tilde{f}.(1-\tilde{f})}.$$

This special assumption allows a term source for variance easy to implicit in order to avoid overshoot

(the variance have a maximal value related to the mean):

$$S_{\gamma'^2} = \Gamma . \left(f_\Gamma - \tilde{f} \right)^2 \left\{ 1 - \left[\frac{\widetilde{f''^2}}{\tilde{f}\ (1-\tilde{f})} \right]^2 \right\} .$$

9.4.6 Nitrogen oxides (NOx)

Nitrogen oxides are a key pollutant, an accurate prediction is difficult but the relative effect of modification (of fuel, of stagging, and so on) is a reachable goal. Hereafter, the two main ways of nitrogen oxides formation are supposed to be thermal NOx (reaction between molecular nitrogen and radical oxygen activated at the higher temperature) and fuel NOx (resulting from the oxidation of nitrogen originating from fuel) are described and taken in account in the code_saturne model for diphasic combustion. The third way, resulting of reaction between the molecular nitrogen and hydrocarbon radicals is assumed negligible in diphasic combustion; its contribution is worthy of attention only for gas combustion (especially in dry low NOx combustor for gas turbine).

Thermal NOx

In the Zeldovich mechanism, the rate of the key reaction, between molecular nitrogen and radical oxygen, has a simple expression thanks to an assumption of equilibrium applied on oxygen dissociation, leading to:

$$\begin{aligned} N_2 + O_2 &\Rightarrow 2\ NO, \\ W1_{NO} &= 3.4\ 10^{12} . \exp\left[\frac{-66\,900}{RT} \right] . [N_2] . [O_2]^{1/2} . \end{aligned}$$

This production term has to be evaluated in each fluid particle because it is not only non linear (with respect to mixture fraction) but submitted to segregation: the hottest particles are near the stoechiometric point ... where oxygen is exhausted (and vice versa). As a consequence, simplest approximations, neglecting covariances, are not satisfaying:

$$\widetilde{W_1}_{NO} \neq k_0 . \exp\left[\frac{E}{R\tilde{T}} \right] . \left[\widetilde{N}_2 \right] . \left[\widetilde{O}_2 \right]^{1/2} .$$

Taking in account only the mean temperature, the contribution of hotest fuilds particles disappears and nitrogen oxide formation is under estimated.

The source term for thermal NOx has to be integrated following the example of others turbulents variables (like species mass fractions). In the turbulent oxydation model, mass fraction of species are known linear piecewise functions, the oxygen fraction is positive only between the mean local oxidizer and fs3 the stoechiometric point for the "last" reaction (H_2S conversion), the post conversion of carbon monoxide to carbon dioxide is assumed to result from a relaxation to the thermodynamical equilibrium computed with mean values. The stoechiometric point corresponding to this last reaction is no more necessary to the turbulent computation but for temperature evaluation: as, by now, the mean local oxidiser includes some carbon monoxide (originating from heterogeneous reactions involving char), it can't be no more supposed unreactive, assuming a linear profile for oxygen, and carbon monoxide, between the local mean oxidiser and the point where hydrocarbon oxidation is finished, the mass fraction in the local mean oxidizer are auxilliary unknowns the value of which is determinated by equalizing the transported value for the carbon monoxide (which involves the effect of relaxation to equilibrium) and its integrated value (taking in account the local pdf).

Assuming the entalpy piecewise linear with the enthalpy in $fs4$ considered as an auxilliairy unknown, which can be determined by equalizing the integrated enthalpy and the transported one (the tranport equation of which includes radiation losses), then the temperature piecewiselinear with respect to

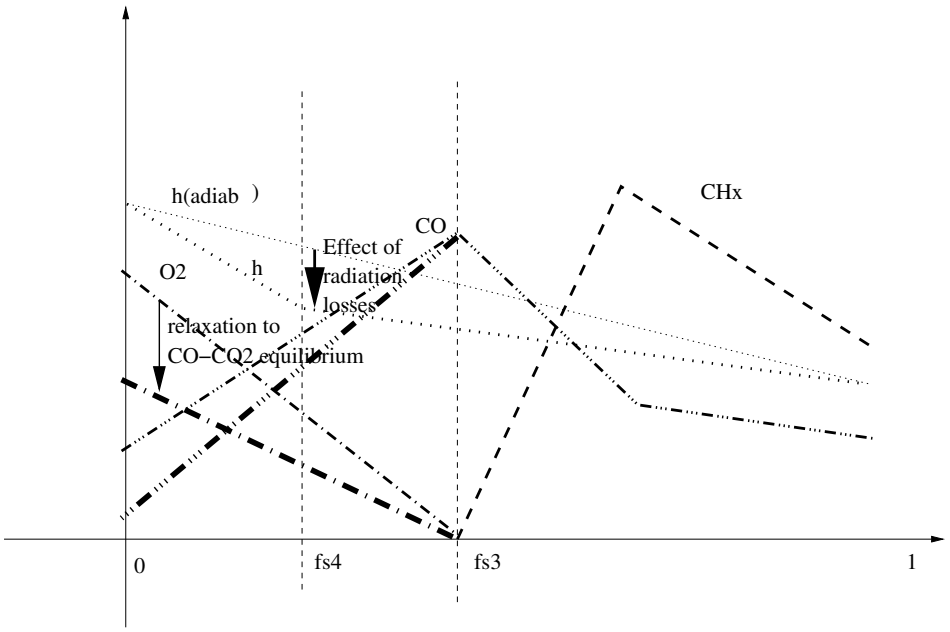


Figure II.9.11: Mass fractions of reactive species, turbulent reaction then kinetical relaxation, enthalpy with radiation losses.

the mixture fraction, the numerical integration of the source term is now available, using a regular trapezium method with 200 points (between the pdf's rectangle begining and the minimum of $fs3$ and pdf's rectangle end).

Fuel NOx

The nitrogen included in the organic part of the fuel (can be a significant part of some biomass, like agricultural residues, oil cake and so on) evolves to HCN. This reducing form of nitrogen can be oxydised either by oxygen or by nitrogen oxide:

$$\begin{aligned} HCN + \frac{5}{4}\, O_2 &\Rightarrow NO + \frac{1}{2}\, H_2O + CO, \\ W2_{NO} &= 3\,10^{12} \exp\left[\frac{-30\,000}{RT}\right] . [HCN] . [O_2]^b , \end{aligned}$$

with:

$$b = \begin{cases} 1 & \text{if} \quad [O_2] < 0.0025, \\ \dfrac{0.018 - [O_2]}{0.018 - 0.0025} & \text{if} \quad 0.0025 < [O_2] < 0.018, \\ 0 & \text{if} \quad 0.018 < [O_2] . \end{cases}$$

In the last case, NO is reduced to nitrogen: the fuel nitrogen contributes to destroy even thermal NO.

$$\begin{aligned} HCN + NO + \frac{1}{4}\, O_2 &\Rightarrow N_2 + \frac{1}{2}\, H_2O + CO, \\ W3_{NO} &= 1.2\,10^{10} . \exp\left[\frac{-33\,500}{RT}\right] . [HCN] . [NO] . \end{aligned}$$

9.4.7 Conservation Equations for two phase flow combustion

The bulk, made of gases and particles, is assumed to be modelled with only one pressure and velocity. Scalars for the bulk are:

- Bulk density

$$\rho_m = \alpha_1 \rho_1 + \sum_{i=1}^{N_{classes}} \alpha_{2,\,i} \rho_2. \tag{II.9.47}$$

- Bulk velocity

$$\underline{u}_m = \frac{\alpha_1 \rho_1 \underline{u}_1 + \sum_{i=1}^{N_{classes}} \alpha_{2,\,i} \rho_2 \underline{u}_{2,\,i}}{\rho_m}. \tag{II.9.48}$$

- Bulk enthalpy

$$H_m = \frac{\alpha_1 \rho_1 . H_1 + \sum_{i=1}^{N_{classes}} \alpha_{2,\,i}\, \rho_2 . H_{2,\,i}}{\rho_m}. \tag{II.9.49}$$

- Bulk pressure

$$P_m = P_1. \tag{II.9.50}$$

Mass fractions of gaseous medium (x_1^*) and of particles (x_2^*) are defined by:

$$\begin{aligned} x_1^* &= \frac{\alpha_1 \rho_1}{\rho_m}, \\ x_2^* &= \frac{\sum_{i=1}^{N_{classes}} \alpha_{2,\,i}\, \rho_2}{\rho_m}. \end{aligned}$$

By default, the slipping velocitiy between particles and gases is supposed negligible compared to this mean velocity, that is to say that the velocity of the continuous phase (gas phase) $\underline{u}_1$ and the velocities of the particle classes $\underline{u}_{2,\,i}$ are equal to the bulk velocity $\underline{u}_m$, then budget equations for the bulk can be written as following: (II.9.51-II.9.53):

$$\frac{\partial}{\partial t} \rho_m + \text{div}\,(\rho_m \underline{u}_m) = 0, \tag{II.9.51}$$

$$\frac{\partial}{\partial t} (\rho_m \underline{u}_m) + \underline{\text{div}}\,(\underline{u}_m \otimes \rho_m \underline{u}_m) = \underline{\text{div}} \left[\mu_T \left(\underline{\underline{\nabla}}\, \underline{u}_m + \underline{\underline{\nabla}}\, \underline{u}_m^T \right)^D - \frac{2}{3} q_m^2 \underline{\underline{1}} \right] - \underline{\nabla} P_m + \rho_m \underline{g}, \tag{II.9.52}$$

$$\frac{\partial}{\partial t} (\rho_m H_m) + \text{div}\,(\rho_m \underline{u}_m H_m) = \text{div}\,(\mu_T \underline{\nabla} H_m) + S_{m,R}, \tag{II.9.53}$$

where the tubulent dynamic viscosity of the bulk is $\mu_T = \rho_m D_m^T$.

With the (velocity) homogeneity assumption, mainly budget equation for bulk caracteristic are pertinent. So transport equation for the scalar Y_k, where k is the phase, can be written:

$$\frac{\partial}{\partial t} (\rho_m x_k^* Y_k) + \text{div}\,(\rho_m \underline{u}_m x_k^* Y_k) = \text{div}\,(\mu_T \underline{\nabla}\,(x_k^* Y_k)) + S_{Y_k} + \Gamma_{Y_k}. \tag{II.9.54}$$

If drift of the particle class to the bulk is taken into account, equation (II.9.54) reads:

$$\frac{\partial}{\partial t}\left(\rho_m x_k^* Y_k\right) + \text{div}\left(\rho_m \underline{u}_k x_k^* Y_k\right) = \text{div}\left(\mu_T \underline{\nabla}\left(x_k^* Y_k\right)\right) + S_{Y_k} + \Gamma_{Y_k}, \tag{II.9.55}$$

with $\underline{u}_k$ to be either modelled with the drop velocity or transported.

Particles enthalpy: $x_2^* H_2$

Enthalpy of droplets (J in particles/kg bulk) is the product of solid phase mass fraction (kg liq/kg bulk) by the specific enthalpy of solid (kg solid/kg bulk). So the budget equation for liquid enthalpy has six source terms:

$$\begin{aligned}\Pi_2^{'} + S_{2,\mathrm{R}} - \Gamma_{\text{evap}} H_{H_2O,\ \text{vap}}(T_2) \quad &- \quad \Gamma_{\text{devol}_1} H_{\text{MV1}}(T_2) \quad - \quad \Gamma_{\text{devol}_2} H_{\text{MV2}}(T_2) \\ &+ \quad \Gamma_{\text{het}} \left(\frac{M_{\text{O}}}{M_{\text{C}}} H_{\text{O}_2}(T_1) \ - \ \frac{M_{\text{CO}}}{M_{\text{C}}} H_{\text{CO}}(T_2) \right),\end{aligned}$$

with

- $\Pi_2^{'}$: heat flux between phases,
- $S_{2,\mathrm{R}}$: radiative source term for droplets,
- $\Gamma_{\text{evap}} H_{H_2O,\ \text{vap}}(T_2)$ the vapor flux leaves at particle temperature (H_{vap} includes latent heat),
- $\Gamma_{\text{dvol1}} H_{\text{MV1}}(T_2)$ the light volatile matter flux leaves at particle temperature (H_{vap} includes latent heat),
- $\Gamma_{\text{dvol2}} H_{\text{MV2}}(T_2)$ the heavy volatile matter flux leaves at particle temperature (H_{vap} includes latent heat),
- $\Gamma_{het}(...)$ heterogenous combustion induces reciprocal mass flux: oxygen arriving at gas temperature and carbon monoxide leaving at char particle one.

Continuous phase enthalpy $x_1 H_1$ and bulk enthalpy H_m

Budget equation for the specific enthalpy of the mixture (gas + particles) admits only one source term for radiative effects $S_{m,\mathrm{R}}$:

$$S_{m,\mathrm{R}} = S_{1,\mathrm{R}} + S_{2,\mathrm{R}}, \tag{II.9.56}$$

with contributions of each phase liable to be described by different models (*e.g.* wide band for gases, black body for particles).

In order to be conservative even with drift is taken into account, $x_1 H_1$ is transported rather than H_m, which is deduced from the identity $H_m = x_1 H_1 + x_2 H_2$.

Therefore, source terms on $x_1 H_1$ are computed using $S_{1,\mathrm{R}} - S_{x_2 H_2}$, with a specific care on the opposite source terms of $x_2 H_2$ so that it is exact even with the time stepping.

Dispersed phase mass fraction: x_2^*

In budget equation for the mass fraction of the dispersed phase (first droplets, then char particles, at last ashes) the source terms are interfacial mass fluxes (first evaporation, then net flux for heterogeneous combustion):

$$-\Gamma_{\text{evap}} - \Gamma_{\text{het}} - \Gamma_{\text{gas},\ H_2O} - \Gamma_{\text{gas},\ CO_2}. \tag{II.9.57}$$

The fuel is described by only one amount (for each diameter class) under a percentile of the initial mass (or diameter), the droplet is supposed to become a char particle. Under a more little diameter the particle is supposed to become an ash particle (inert).

$$-\Gamma_{\text{pyrol1}} - \Gamma_{\text{pyrol2}} - \Gamma_{\text{het}} - \Gamma_{\text{gas, } H_2O} - \Gamma_{\text{gas, } CO_2}. \tag{II.9.58}$$

The coal or biomass particles are described by three mass component: water (free: available for drying), reactive coal (available for pyrolisis), char (available for heterogeneous oxidation and gasification). The mass of ash is computed with respect to the number of particles, initial size and initial amount of ashes. The sum of these four component is the amount of each class (initial diameter and kind of coal).

Number of particles: N_p^*

No source term in the budget equation for number of droplets or solid particles: a droplet became a particle (eventually a tiny flying ash) but never vanish (all particles have to get out).

Mean of the passive scalar for light volatile: F_1

This scalar represents the amount of matter released by the first (low activation energy) pyrolisis reaction. It is a mass fraction of gaseous matter (in hydrocarbon form or carbon oxide one). So the source term in its budget is only the pyrolisis mass flux:

$$\Gamma_{\text{pyrol}}. \tag{II.9.59}$$

Mean of the passive scalar for heavy volatile: F_2

This scalar represents the amount of matter which has leaved the droplet as fuel vapour or the particle by the second (high activation energy reaction), whatever it happens after. It's a mass fraction of gaseous matter (in hydrocarbon form or carbon oxide ones). So the source term in its budget is only evaporation or pyrolisis mass flux:

$$\Gamma_{\text{pyrol1}} \quad \text{or} \quad \Gamma_{\text{evap}}. \tag{II.9.60}$$

Mean of the passive scalar for oxidizers: F_3 to F_5

Budget equation for the three different oxidizers taken in account don't have any source term.

Mean of the passive scalar for steam from drying: F_6

Budget equation for the water steam issued from drying of coal or biomass has one source term:

$$\Gamma_{\text{dry}}. \tag{II.9.61}$$

Mean of the passive scalar for carbon from char oxidation: F_7

Budget equation for F_7 hase for source term the mass flux due to heterogeneous combustion (mass flux of carbon monoxide minus oxygene mass flux). As for F_1, oxidation in the gaseous phase does not modifiy this *passive* scalar:

$$\Gamma_{\text{het}}. \tag{II.9.62}$$

Mean of the passive scalar for gasification by the carbon dioxide: F_8

Budget equation for F_8 has for source term the mass flux due to heterogeneous combustion (mass flux of carbon monoxide minus oxygene mass flux). As for F_1, oxidation in the gaseous phase does not modifiy this *passive* scalar:

$$\Gamma_{\text{gas, } CO_2}. \tag{II.9.63}$$

Mean of the passive scalar for gasification by steam: F_9

Budget equation for F_9 have for one source term the mass flux due to heterogeneous combustion (mass flux of carbon monoxide minus oxygene mass flux). As for F_1, oxidation in the gaseous phase does not modifiy this *passive* scalar:

$$\Gamma_{\text{gas, } H_2O}. \tag{II.9.64}$$

Droplets enthalpy: $x_2^* H_2$

Enthalpy of droplets (J in droplets/kg bulk) is the product of liquid phase mass fraction (kg liq/kg bulk) by the specific enthalpy of liquid (kg liq/kg bulk). So the budget equation for liquid enthalpy has four source terms:

$$\Pi_2^{'} + S_{2,\text{R}} - \Gamma_{\text{evap}} H_{\text{vap}}(T_2) + \Gamma_{\text{het}} \left(\frac{M_{\text{O}}}{M_{\text{C}}} H_{\text{O}_2}(T_1) - \frac{M_{\text{CO}}}{M_{\text{C}}} H_{\text{CO}}(T_2) \right) \tag{II.9.65}$$

with

- $\Pi_2^{'}$: heat flux between phases
- $S_{2,\text{R}}$: radiative source term for droplets
- $\Gamma_{\text{evap}} H_{\text{vap}}(T_2)$ the vapor flux leaves at droplet temperature (H_{vap} includes latent heat)
- $\Gamma_{\text{het}}(...)$ heterogenous combustion induces reciprocal mass flux: oxygen arriving at gas temperature and carbone monoxide leaving at char particle one.

Dispersed phase mass fraction: x_2^*

In budget equation for the mass fraction of the dispersed phase (first droplets, then char particles, at last ashes) the source terms are interfacial mass fluxes (first evaporation, then net flux for heterogeneous combustion):

$$-\Gamma_{\text{evap}} - \Gamma_{\text{het}}. \tag{II.9.66}$$

Chapter 10

Groundwater flows

10.1 Introduction

The Hydrogeology module of code_saturne is a numerical model for water flow and solute transport in continuous porous media, based on the Darcy law for flow calculation, and on the classical convection-diffusion equation for transport. It allows to simulate steady or unsteady flows, saturated or not, with scalar or tensorial permeabilities, and transport with dispersion, sorption, precipitation and radioactive decay. Any law, even unlinear, is acceptable for dependences between moisture content, permeability and hydraulic head.

For the solving of the flow, the Richards equation is used, derived from the Darcy law and the conservation of mass. In the general case, this equation is non-linear and must be solved by a Newton scheme.

From this flow, the transport equation is solved, taking into account convection and diffusion, both slightly modified to take into account the specificities of underground transport (especially partition between liquid phase and soil matrix).

Physical concepts and equations developed in this module are detailed hereafter.

See the programmers reference of the dedicated subroutine for further details.

10.2 Groundwater flows

10.2.1 Continuity Equation

The expression of the mass conservation for the water contained in a volume Ω of the subsurface, delimited by a surface boundary $\partial\Omega$, can be written:

$$\int_\Omega \rho \frac{\partial \theta}{\partial t} \mathrm{d}\Omega + \int_{\partial\Omega} \rho \underline{u} \cdot \mathrm{d}\underline{S} = \int_\Omega \rho \, Q_s \mathrm{d}\Omega \tag{II.10.1}$$

with:

- θ is the moisture content (also called saturation) $[L^3.L^{-3}]$;
- ρ is the density of water $[M.L^{-3}]$;
- $\underline{u}$ is the water velocity $[L.T^{-1}]$;
- Q_s is a volumetric source term $[L^3.T^{-1}]$.

By assuming a constant density of water ρ and using Gauss theorem, the equation simplifies to the following local expression:

$$\frac{\partial \theta}{\partial t} + \mathrm{div}\ (\underline{q}) = Q_s \tag{II.10.2}$$

As seen in section 10.3, the moisture content can be determined from the pressure head.

10.2.2 Darcy Law

The momentum conservation equation in a continuous porous medium is expressed through Darcy law, an empirical relationship which shows the proportionality between the velocity of the water $\underline{u}$ and the gradient of the soil water potential. This means that motion of water in a porous medium is due to both the gradient of water pressure and gravity. The following equation describes the pressure head h, which is equivalent to the water pressure but expressed in a length unit [L]:

$$h = \frac{p}{\rho g} + A, \tag{II.10.3}$$

with A a constant such that $h = 0$ at the atmospheric pressure, as a convention. We also introduce the hydraulic head H:

$$H = h + z. \tag{II.10.4}$$

Darcy law was initially established for mono-dimensional flows in saturated isotropic conditions. It binds darcian velocity $\underline{q}$, which is the product of the real flow velocity $\underline{u}$ and the moisture content θ of the medium, to the gradient of hydraulic head. To represent anisotropic conditions and multidimensional flows, it has to be extended to a tensorial relationship:

$$\underline{q} = \theta \underline{u} = -K \underline{\nabla} H = -K \underline{\nabla}(h + z) \tag{II.10.5}$$

where K can be scalar or tensoriel. It is called hydraulic conductivity. By abuse of langage, we also call it permeability, but in a rigorous sense the hydraulic conductivity is deduced from the permeability of the soil and the properties of the water. It varies in unsaturated conditions, and depends on the nature of the soil and the pressure head h (see section 10.3). Notice that the constant A has no importance in the Darcy equation, neither in the following of the development.

10.2.3 Richards equation

Richards equation is obtained by substitution of the velocity expression given in equation (II.10.5) directly into the continuity equation (II.10.2):

$$\frac{\partial \theta(h)}{\partial t} = \text{div } (K(h) \underline{\nabla} H) + Q_s \tag{II.10.6}$$

10.3 Soil-water relationships

To close Richards equation, two extra relationships have to be given to link the hydraulic conductivity and the moisture content to the pressure head. The relationship between the moisture content and the pressure head is usually derived from the *soil water retention* curve, which is determined experimentally. The Hydrogeology module permits to define any model of that kind. In the following we denote θ_s the porosity of the soil and θ_r the residual moisture content, which is fraction of water that cannot be removed from the soil.

In saturated conditions, we have $\theta = \theta_s$ everywhere, thus θ only depends on the nature of the soil. As for the permeability, it usually does not depend on the pressure head in this case, and is also constant for a given soil.

In unsaturated conditions, the laws used to analytically derive soil hydraulic properties are usually highly non-linear. The Richards' equation is thus a non-linear second order partial differential equation in unsaturated conditions. The method chosen to solve it involves an iterative process used to linearise the equation as described in section 10.4. Let us give the example of the Van Genuchten model with Mualem condition, wich is the most commonly used:

$$S_e = \frac{\theta - \theta_r}{\theta_s - \theta_r} = \begin{cases} [1 + |\alpha h|^n]^{-m} & \text{if } h < 0 \\ 1 & \text{if } h \geq 0 \end{cases} \tag{II.10.7}$$

with n and m two constant parameters.

$$K = \begin{cases} K_0 \; S_e^L \left(1 - \left(1 - S_e^{1/m}\right)^m\right)^2 & \text{if } h < 0 \\ K_0 & \text{if } h \geq 0 \end{cases} \tag{II.10.8}$$

with K_0 a constant not depending on the moisture content. Notice that if $h > 0$, then we have a saturated soil. That is because we chose, as a convention (and as explained in section 10.2.2), to define the pressure head such that it vanishes at the atmospheric pressure. When the soil is saturated, the permeability is equal to K_0, depending only on the soil.

10.4 Solving of the Richards equation

In the general case, the laws connecting the hydraulic properties are non-linear. We will have to implement an iterative process for solving the Richards equation. First, we define the soil capacity C:

$$C(h) = \frac{\partial \theta}{\partial h}, \tag{II.10.9}$$

which can be derived from the soil law linking θ and h. A classical way to solve Richard's equation (II.10.6) is to first transform it with the approximation:

$$\frac{\partial \theta}{\partial t} \simeq C(h) \frac{\partial h}{\partial t}, \tag{II.10.10}$$

so that it becomes:

$$C(h)\ \frac{\partial h}{\partial t} \simeq \text{div}\ (K(h)\ \underline{\nabla}(h+z)) + Q_s, \tag{II.10.11}$$

this last formulation being called the *h-based* formulation. The equation ((II.10.11)) can be written (recalling that $H = h + z$):

$$C(H-z)\ \frac{\partial H}{\partial t} \simeq \text{div}\ (K(H-z)\underline{\nabla}(H)) + Q_s, \tag{II.10.12}$$

and then discretized in time:

$$C(H^n - z)\ \frac{H^{n+1} - H^n}{\Delta t} \simeq \text{div}\ \left(K(H^{n+1} - z)\underline{\nabla}(H^{n+1})\right) + Q_s. \tag{II.10.13}$$

The complete implicitation of the right hand term is made for ensuring stability. The explicitation of the capacity C is chosen after testing different options and noticing that implicitation does not improve the results. We will now linearize the equation ((II.10.13)) and define sub-iterations to solve it. Suppose that we seek unknown variable H^{n+1} at the sub-iteration $k+1$ from its value at sub-iteration k. We write:

$$C(H^n - z)\ \frac{H^{n+1,\ k+1} - H^n}{\Delta t} \simeq \text{div}\ \left(K(H^{n+1,\ k} - z)\ \underline{\nabla}(H^{n+1,\ k+1})\right) + Q_s. \tag{II.10.14}$$

The equation ((II.10.14)), whose unknown is $H^{n+1,k+1}$, is a transport equation without convection, which can be solved by the usual routines of code_saturne.

But the approximation ((II.10.10)) does not ensure a rigorous conservation of mass after discretization, because we do **not** have:

$$C(h^n)\ \frac{h^{n+1} - h^n}{\Delta t} = \frac{\theta(h^{n+1}) - \theta(h^n)}{\Delta t}.$$

Anyway, it is still possible to define the exact residual:

$$R(h^n,\ h^{n+1}) = C(h^n)\ \frac{h^{n+1} - h^n}{\Delta t} - \frac{\theta(h^{n+1}) - \theta(h^n)}{\Delta t} \tag{II.10.15}$$

and to mix it into the discretized and linearized formulation ((II.10.14)) at the sub-iteration k, to obtain:

$$C(H^n - z)\ \frac{H^{n+1,\ k+1} - H^n}{\Delta t} \simeq \\ \text{div}\ \left(K(H^{n+1,\ k} - z)\underline{\nabla}(H^{n+1,\ k+1})\right) + Q_s + R(H^n - z,\ H^{n+1,\ k} - z). \tag{II.10.16}$$

As equation (II.10.14), this can be solved by the usual routines of code_saturne. Then the sub-iterations, if they converge, lead to the exact solution of:

$$\frac{\theta(h^{n+1}) - \theta(h^n)}{\Delta t} = \left[\text{div}\ \left(K(h^{n+1})\underline{\nabla}(H^{n+1})\right)\right]^D + Q_s, \tag{II.10.17}$$

where exponent D represents the spatial discretization of gradient - divergent operator in code_saturne. This discrete operator is rigorously conservative, thus the global volume of water:

$$\int_D \theta \mathrm{d}\Omega,$$

where D denotes the entire domain, is conserved in a discrete sense (provided that there is no physical loss at the boundaries).

10.4.1 Finite Volume method in code_saturne for the operator-diffusive terms

In code_saturne, the integral on a cell Ω of the term div $(K\underline{\nabla}Y)$ is discretized this way:

$$\int_\Omega \text{div } (K\underline{\nabla}Y)\,\mathrm{d}\Omega \simeq \sum_{f\in\mathcal{F}_c} K_f \underline{\nabla}_f Y \cdot \underline{S}_f, \tag{II.10.18}$$

where $\mathcal{F}_c$ is the set of the faces of cell c. For each face f, K_f is the face diffusivity (calculated from the diffusivities at centers of the adjacent cells, with an harmonic or arithmetic mean), $\underline{\nabla}_f Y$ is the gradient of Y at the face center, and $\underline{S}_f$ is a vector normal to the face, whose size is the surface of the face, directed towards the outside of the cell Ω. There are two ways of calculating the term $\underline{\nabla}_f Y \cdot \underline{S}_f$: a simple one (*i.e.* without reconstruction) and an accuracy one (*i.e.* with reconstruction). In the general case, two adjacent cells of code_saturne can be represented like in the picture below, where two cells c and $\bar{c}$ are separated by a face denoted $f_{c|\bar{c}}$: The variables are known at centers of the cells, *i.e.*

at points $\underline{x}_c$ and $\underline{x}_{\bar{c}}$. But to get a rigorous estimation of the normal gradient at the center of the face $f_{c|\bar{c}}$, we need values at points $\underline{x}'_c$ and $\underline{x}'_{\bar{c}}$:

$$\underline{\nabla}_{f_{c|\bar{c}}} Y \cdot \underline{S}_{ij} = \frac{Y_{\underline{x}'_{\bar{c}}} - Y_{\underline{x}'_c}}{|\underline{x}'_{\bar{c}} - \underline{x}'_c|}. \tag{II.10.19}$$

The face gradient without reconstruction is calculated by the approximation:

$$\underline{\nabla}_{f_{c|\bar{c}}} Y \cdot \underline{S}_{ij} = \frac{Y_{\underline{x}_{\bar{c}}} - Y_{\underline{x}_c}}{|\underline{x}'_{\bar{c}} - \underline{x}'_c|}. \tag{II.10.20}$$

The less the vector $\underline{x_c x_{\bar{c}}}$ is orthogonal to the face $f_{c|\bar{c}}$, the more this approximation is wrong. But it has the advantage to be easy and quick to deduce from the variables at cell centers, with a linear relation, and to depend only on the variables of the adjacent cells. The face gradient with reconstruction is calculated following the relation (II.10.19), thanks to the relations:

$$Y_{\underline{x}'_c} = Y_{\underline{x}_c} + \underline{\nabla} Y_{\underline{x}_c} \cdot (\underline{x}'_c - \underline{x}_c). \tag{II.10.21}$$

$$Y_{\underline{x}'_{\bar{c}}} = Y_{\underline{x}_{\bar{c}}} + \underline{\nabla} Y_{\underline{x}_{\bar{c}}} \cdot (\underline{x}'_{\bar{c}} - \underline{x}_{\bar{c}}). \tag{II.10.22}$$

Thus, the calculation of the face gradient with reconstruction requires a calculation of the gradients of Y at cell centers, which can be done by several methods. Depending on the choosen method, the relation between the values of Y at cell centers and the gradient at cell centers can be nonlinear and require a large stencil of cells. We will see in section 10.4.2 how the laplacian is solved in code_saturne, in order to get the solution with the accurate definition of the face gradients but to keep the simple definition for matrix inversions.

10.4.2 Solving of the linear sub-iteration in code_saturne

The sub-iteration (II.10.16) can be rewritten:

$$f_s\ \delta_H - \text{div } (\mu\underline{\nabla}\delta_H) = Q_H, \tag{II.10.23}$$

where:

- $\delta_H = H^{n+1,\ k+1} - H^{n+1,\ k}$ is the unknown;
- $f_s = \frac{C(H^n - z)}{\Delta t}$, not depending on the unknown;
- $\mu = K(H^{n+1,\ k} - z)$ is the diffusion coefficient, tensorial if the permeability is tensorial. It does not depend on the unknown;
- Q_H is the second member, not depending on the unknown.

We have:

$$Q_H = Q_s + R(H^n - z,\ H^{n+1,\ k} - z) - \text{div}\left(K(H^{n+1,\ k} - z)\ \underline{\nabla}(H^{n+1,\ k})\right). \tag{II.10.24}$$

Now, let us denote E_n the following operator, that applies on any discrete field x:

$$E_n(x) = f_s\ x - \left[\text{div}\ (\mu \underline{\nabla} x)\right]^D, \tag{II.10.25}$$

where exponent D represents the spatial discretization of gradient - divergent operator in code_saturne. This operator is linear but, when using the reconstruction of the non-orthogonalities, it is difficult to invert (see section 10.4.1). Thus, we also define EM_n, that is the equivalent of E_n but without taking into account the reconstruction of non-orthogonalities. Now, we write the discretisation of the equation (II.10.23) in the form:

$$E_n(x) = Q_H, \tag{II.10.26}$$

where x is the unknown. In order to solve it, we define the sequence $(x^m)_{m\in\mathbb{N}}$ that is calculated following these iterations:

- $EM_n(\delta x^{m+1}) = -E_n(\delta x^m) + Q_H$;
- $x^{m+1} = x^m + \delta x^{m+1}$;
- $x^0 = \textit{initial guess}$.

With that method, we only invert the simple matrix EM_n. If the iterations converge, we get the solution of (II.10.26) with an arbitrary precision, and with a precise definition of the discrete diffusive operator. This is the standard way of dealing with the diffusion problem in code_saturne. See documentation on routine *codits*, for example, for further details.

10.4.3 Determination of the velocity field

Theoretically, the darcy velocity field $\underline{q}$ of the flow just has to be calculated from the pressure head field, thanks to the Darcy relation (II.10.5). This can be done with the routine of code_saturne that calculates the gradient of a variable at cell centers from the values of this variable at cell centers. However, this simple way of getting the velocity field is only used for posttreatment purpose, and not used in the transport equation, for the reasons given below.

In code_saturne, the integral on a cell Ω of the convection term div $\left(Y\underline{q}\right)$, where $\underline{q}$ is a velocity field and Y a transported variable, is discretized this way:

$$\int_\Omega \text{div}\ \left(Y\ \underline{q}\right) \mathrm{d}\Omega \simeq \sum_{f\in\mathcal{F}_c} Y_f\ \underline{q}_f \cdot \underline{S}_f, \tag{II.10.27}$$

where $\mathcal{F}_c$ is the set of faces for cell c. For each face f, Y_f is the value of Y at the center of face $\mathcal{F}_c$ (calculated from the values of Y at centers of the adjacent cells, with an arithmetic mean), $\underline{q}_f$ is the velocity at face center, and $\underline{S}_f$ is a vector normal to the face, whose size is the surface of the face, directed towards the outside of the cell Ω. Thus the term $\underline{q}_f \cdot \underline{S}_f$ is the mass flux at face center.

These mass fluxes at face centers could be deducted from the velocity values at cell centers, but this method would not ensure the exact coherence of these mass fluxes with the continuity equation (II.10.2). To ensure this coherence, let us write the discrete contnuity equation:

$$\frac{\theta^{n+1}-\theta^{n}}{\partial t}+\left[\text{div }\left(\underline{q}\right)\right]^{D}=Q_{s}, \tag{II.10.28}$$

where exponent D corresponds to the discrete operator for convection, described above. Mixing the discrete Richards equation (II.10.17) and the discrete continuity equation (II.10.28), we want to have:

$$\left[\text{div }\left(K(h^{n+1})\underline{\nabla}(H^{n+1})\right)\right]^{D}=\left[\text{div }\left(\underline{q}\right)\right]^{D}. \tag{II.10.29}$$

Exponent D still represents discretisation of operators. Taking into account equation ((II.10.18)) and equation ((II.10.27)), this leads for each face f of the domain to:

$$K(h^{n+1})_{f}\underline{\nabla}_{f}H^{n+1}\cdot\underline{S}_{f}=\underline{q}_{f}\cdot\underline{S}_{f}. \tag{II.10.30}$$

This gives the good value for $\underline{q}_{f}\cdot\underline{S}_{f}$, available from the solving of Richards equation.

So, for the purpose of discrete coherence between flow and transport equations (which is important for the precision of the computation and coherence of the results), we deduct the mass fluxes used in the transport equation from the normal gradients of the pressure head H calculated in the solving of Richards equation, instead of deducting them from the velocity values at cell centers.

10.4.4 Convergence criterion

Two choices are available for the convergence criterion of the loop over sub-iterations k (from section 10.4). The first possibility is to continue the loop until two successive pressure head fields are close enough, *i.e.*

$$\|h^{n+1,\ k+1}-h^{n+1,\ k}\|^{L2}\leq\epsilon,$$

where the value of ϵ is given by the user. The second possibility is to impose such a condition on the velocity field of the flow, *i.e.*

$$\|u_{x}^{n+1,\ k+1}-u_{x}^{n+1,\ k}\|^{L2}+\|u_{y}^{n+1,\ k+1}-u_{y}^{n+1,\ k}\|^{L2}+\|u_{z}^{n+1,\ k+1}-u_{z}^{n+1,\ k}\|^{L2}\leq\epsilon,$$

where we denoted u_x, u_y and u_z the components of $\underline{u}$ over the three spatial directions. This last choice imposes to calculate the velocity field at the end of each sub-iteration. Both of these options are available in the module.

10.4.5 Cases without gravity

If we don't want to take into account the gravity, then the Darcy law writes:

$$\underline{u}=-K\underline{\nabla}h, \tag{II.10.31}$$

and the Richards equation becomes:

$$\frac{\partial\theta(h)}{\partial t}=\text{div }\left(K(h)\ \underline{\nabla}(h)\right)+Q_{s}, \tag{II.10.32}$$

which is solved exactly the same way, except that the solved variable is h instead of H. The user of the module must be careful to adapt the initial conditions, boundary conditions and soil-laws accordingly.

10.5 Groundwater Transfers

10.5.1 Introduction

The transport of a solute in porous media with variable saturation is treated by the Hydrogeology module, based on code_saturne transport equation solver with only few specific developments.

10.5.2 Partition between liquid and solid phases

Solutes are present in soils and aquifers in liquid phase and can be sorbed in solid matrix.

$$C_{tot} = \theta c + \rho C_S \tag{II.10.33}$$

where:

- C_{tot} is the total concentration of the solute $[M.L^{-3}_{soil}]$;
- c is the solute concentration in liquid phase $[M.L^{-3}_{water}]$;
- C_S is the solute mass fraction in solid phase (so called sorbed concentration $[M.M^{-1}_{soil}]$);
- ρ is the soil *bulk* density $[M_{soil}.L^{-3}_{soil}]$.

In most of non reactive transport models, an equilibrium between liquid and solid phases is considered. This is modeled by a soil-water partition coefficient, referred as K_d $[L^3_{water}.M^{-1}_{soil}]$:

$$C_S = K_d \times c. \tag{II.10.34}$$

Then the total concentration may be written as:

$$C_{tot} = \theta R c \tag{II.10.35}$$

where R [.] is a retardation factor representing the delay of solute transport due to its sorption in solid matrix:

$$R = 1 + \frac{\rho K_d}{\theta} \tag{II.10.36}$$

10.5.3 Advection/dispersion/diffusion/decay equation

We assume hereafter that the solute only exists in the liquid phase and is potentially sorbed on the solid matrix. We also assume that the transport phenomena in the liquid phase are advection, kinematic dispersion and molecular diffusion. The classical transport equation in variably saturated flow is:

$$\frac{\partial (R\theta c)}{\partial t} = \frac{\partial}{\partial x_j}(D_{ij}\frac{\partial c}{\partial x_i}) - \frac{\partial \underline{q}_i c}{\partial x_i} + Q_s c_r - \lambda R \theta c \tag{II.10.37}$$

where:

- R is the delay factor, representing sorption phenomena [-];
- θ is the moisture content $[L^3.L^{-3}]$;
- c is the solute concentration in the liquid phase $[M.L^{-3}]$;
- $\underline{q}$ refers to the darcian velocity, which is a result of the solving of the Richards equation (see section 10.4.3) $[L.T^{-1}]$;
- λ is a first-order decay coefficient $[M.L^{-3}.T^{-1}]$;
- Q_s refers to the volumetric flow source/sink, from the Richards equation (II.10.6) $[M^3.T^{-1}]$;
- c_r is the source/sink concentration $[M.L^{-3}]$;
- D_{ij} is the dispersion tensor. It contains both the kinematic dispersion and the molecular diffusion $[L^2.T^{-1}]$.

We note the following differences with the standard formulation of the transport equation in code_saturne:

- the presence of the delay factor R and the moisture content θ in the unsteady term;
- the tensorial diffusivity D_{ij}.

10.5.4 Kinematic dispersion

Kinematic dispersion results from the existence of a very complex and unknown velocity field which is not taken into account for advection (the average Darcy velocity is considered instead). It results in a kinematic dispersion tensor denoted D_k, whose main directions of anisotropy are the direction of the flow and two directions orthogonal to the flow. This tensor can be inferred from two parameters named longitudinal and transversal dispersion coefficients (m.s-1), denoted α_l and α_t, and from the amplitude of the velocity darcy field. In an orthonormal frame such that the first direction is the direction of the flow, this kinematic dispersion tensor writes:

$$D_k = \mid \underline{q} \mid \begin{pmatrix} \alpha_l & 0 & 0 \\ 0 & \alpha_t & 0 \\ 0 & 0 & \alpha_t \end{pmatrix} \tag{II.10.38}$$

Physically, The coefficients α_l and α_t are representative of the size of the biggest heterogeneities on the solute path. Their determination is empirical.

10.5.5 Molecular diffusion

Molecular diffusion is due to Brownian motion of solute molecules that tends to reduce the differences of concentration in different points of a continuous medium. It is already taken into account in the standard transport equation of code_saturne. In porous media, molecular diffusion occurs in the whole fluid phase but not in the solid medium. Hence, the diffusion coefficient is, in the general case, proportional to the moisture content θ. It is denoted d_m.

10.5.6 Dispersion tensor

Finally, the dispersion tensor D_{ij} is the cumulation of the kimematic dispersion tensor D_k and the molecular diffusion scalar. In a frame independant of the flow, it can be written:

$$D_{ij} = \alpha_t \mid \underline{q} \mid \delta_{ij} + (\alpha_l - \alpha_t)\frac{\underline{q}_i \underline{q}_j}{\mid \underline{q} \mid} + d_m \delta_{ij}, \tag{II.10.39}$$

where:

- δ_{ij} refers to the Kronecker symbol $[-]$;
- α_l is the longitudinal dispersivity $[L]$;
- α_t is the transversal dispersivity $[L]$;
- d_m is the water molecular diffusion $[L^2.T^{-1}]$;
- $\underline{q}_i$ refers to the darcian velocity in the direction i $[L.T^{-1}]$;
- $\mid \underline{q} \mid$ is the norm of the darcian velocity $[-]$.

Finally, the tensor is symetric (*i.e.* $D_{ij} = D_{ji}$) and can be expressed as:

- $D_{xx} = \alpha_l \frac{q_x^2}{|\underline{q}|} + \alpha_t \frac{q_y^2}{|\underline{q}|} + \alpha_t \frac{q_z^2}{|\underline{q}|} + d_m$;
- $D_{yy} = \alpha_t \frac{q_x^2}{|\underline{q}|} + \alpha_l \frac{q_y^2}{|\underline{q}|} + \alpha_t \frac{q_z^2}{|\underline{q}|} + d_m$;

- $D_{yy} = \alpha_t \frac{q_x^2}{|\underline{q}|} + \alpha_t \frac{q_y^2}{|\underline{q}|} + \alpha_l \frac{q_z^2}{|\underline{q}|} + d_m$;
- $D_{xy} = (\alpha_l - \alpha_t) \frac{q_x q_y}{|\underline{q}|}$;
- $D_{xz} = (\alpha_l - \alpha_t) \frac{q_x q_z}{|\underline{q}|}$;
- $D_{yz} = (\alpha_l - \alpha_t) \frac{q_y q_z}{|\underline{q}|}$.

10.5.7 Specificities of the groundwater transport equation in relation to the standard transport equation

The general method developed in code_saturne for the treatment of the transport equation is kept; just a few changes in the definition of the general matrix to invert and of the right hand side have been done, in order to take into account the presence of the moisture content and delay in the unsteady term, and to give to the user the opportunity to define a dispersion tensor. More specifically, as values at iterations n and $n+1$ of moisture content and delay are available for the transport calculation at iteration n, we can discretize the unsteady term this way:

$$\frac{\partial(R\theta c)}{\partial t} \simeq \frac{R^{n+1}\ \theta^{n+1}\ c^{n+1} - R^n\ \theta^n\ c^n}{\Delta t}, \tag{II.10.40}$$

which ensures global discrete mass conservation of the tracer.

10.6 Alternative models to treat the soil-water partition of solutes

10.6.1 EK model

In the previous section, an equilibrium between liquid and solid phases is considered. This assumption is valid if this equilibrium is instantaneous, linear and reversible. Hoewever, these properties are rarely verified. Indeed, sorption and desorption processes often follow different kinetics laws. EK model (equilibrium-kinetic) is an alternative, semi-empirical approach which considers two types of sorption sites (see figure 10.6.1):

- Sites denoted 1 are at equilibrium with liquid phase. As in the K_d approach, mass fraction C_{S1} is proportional to the liquid concentration:

$$C_{S1} = K_d \times c. \tag{II.10.41}$$

- Sites denoted 2 interact with liquid phase with kinetic. The time evolution of mass fraction C_{S2} is described by a partial differential equation:

$$\frac{\partial C_{S2}}{\partial t} = k^+ c - k^- C_{S2} \tag{II.10.42}$$

  where k^+ $[L^3_{water}.M^{-1}_{soil}.T^{-1}]$ and k^- $[T^{-1}]$ are experimental parameters, depending on soil properties and on the solute under consideration.

In this case the total concentration can be written as follows:

$$C_{tot} = (\theta + \rho K_d) \times c + \rho \times C_{S2} \tag{II.10.43}$$

Global transport equation (II.10.37) then becomes:

$$\frac{\partial(R\theta c)}{\partial t} + \rho \frac{\partial(Cs_2)}{\partial t} = \frac{\partial}{\partial x_j}(D_{ij} \frac{\partial c}{\partial x_i}) - \frac{\partial q_i c}{\partial x_i} + Q_s c_r - \lambda(R\theta c + \rho Cs_2). \tag{II.10.44}$$

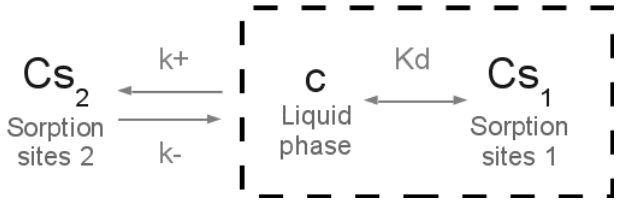


Figure II.10.1: Two-site sorption model: solid 1 at equilibrium with water phase and solid 2 with partially reversible sorption

To treat the kinetic sorption term, c is approximated by a constant and equation (II.10.42) is analytically solved :

$$C_{S2}^{n+1} = e^{-k^- \Delta t} C_{S2}^{n} - \frac{k^+}{k^-}(e^{-k^- \Delta t - 1})c^n. \tag{II.10.45}$$

Note that this approximation is valid only if the time step is not too high.

10.6.2 Precipitation

For some solutes, the concentration in liquid phase is physically bounded by a maximum value, called solubility index (or c_{sat}). When the concentration exceeds this value, the solute precipitates and a solid phase is formed. When the concentration decreases below the solubility index, the solid phase can dissolute.

This process is assumed instantaneous and is treated at the end of each time step as follows:

$$\begin{aligned} c &= \min(c + c_{precip}, c_{sat}) \\ c_{precip} &= \max(0, c + c_{precip} - c_{sat}) \end{aligned} \tag{II.10.46}$$

where c_{precip} $[M^3.L^{-3}]$ is the concentration of precipitate phase.

Chapter 11

Magneto-Hydro Dynamics

See § A and the programmers reference of the dedicated subroutine for further details.

Chapter 12

Lagrangian particle tracking

12.1 Trajectography

The Lagrangian tracking of particles within a mesh implies that both the global coordinates of each particle and its location within the mesh need to be determined. While the particle displacement is calculated according to the particle equations of motion, a specific trajectography module allows to evaluate the cell to which the particle belongs.

The main idea of the algorithm for trajectography is the following (see also Figure II.12.1): knowing the initial particle location O, its current cell and the particle location at the end of the time step D, we check if the particle displacement crosses one of the faces of the cell containing the particle. For that purpose, for each face of the current cell, we calculate if there is an intersection I between the particle displacement vector $\underline{OD}$ and each subtriangle composing the face (for instance, the face on the right-hand side in Figure II.12.1 can be decomposed into four subtriangles, here (GP_1P_2), (GP_2P_3), (GP_3P_4) and (GP_4P_1)). In this example, to determine whether the intersection with the line (OD) is inside the triangle (GP_1P_2), three conditions need to be fulfilled:

$$\frac{\underline{OD}\cdot(\underline{GP_1}\wedge\underline{GO})}{\underline{OD}\cdot(\underline{GP_2}\wedge\underline{GP_1})}<0 \tag{II.12.1a}$$

$$\frac{\underline{OD}\cdot(\underline{GP_2}\wedge\underline{GO})}{\underline{OD}\cdot(\underline{GP_2}\wedge\underline{GP_1})}>0 \tag{II.12.1b}$$

$$\frac{\underline{OD}\cdot(\underline{P_1P_2}\wedge\underline{P_1O})}{\underline{OD}\cdot(\underline{GP_2}\wedge\underline{GP_1})}<0 \tag{II.12.1c}$$

It should be noted that these three conditions are based on geometric considerations (similar to the Möller-Trumbore algorithm) which amount to checking if the intersection point I is located on the proper side of each edge of the triangle. For computational purposes, only the sign of these parameters are calculated in the calculation. This choice has been made to avoid to treat properly the case where the intersection is right on the edge (in that case, the calculated value is identical for both faces containing the edge meaning that the intersection exists only on one side of the edge).

If the line (OD) crosses a triangle, the coordinates of the intersection point I are calculated using:

$$\underline{x}_I=\underline{x}_O+t\times\underline{OD} \tag{II.12.2}$$

with $t\in[0,1[$ given by:

$$t=\frac{\underline{GO}\cdot(\underline{GP_2}\wedge\underline{GP_1})}{\underline{OD}\cdot(\underline{GP_2}\wedge\underline{GP_1})} \tag{II.12.3}$$

Moreover, Eqs. (II.12.1) actually provide information whether the whole line (OD) crosses one of the faces of the current cell. Comparing the orientation of the displacement vector to the orientation of the local normal to the triangle (oriented towards the cell center) thus gives information on the number of times the line (OD) enters (parameter *n_in*) and leaves (parameter *n_out*) the current cell. This is used to check if the particle is really inside the current cell, which amounts to the following condition: $n_{in}=n_{out}>0$.

Besides, a specific treatment has been added to deal with the case of warped faces. As depicted in Figure II.12.2, in this case, the particle displacement vector $\underline{OD}$ may cross several subtriangles of a single face. To properly treat such cases, a parameter has been added to measure how many times the displacement vector $\underline{OD}$ enters and leaves one face (in a similar way to the one used for the parameters *n_in* and *n_out*). As a result, depending on the value of this parameter, two different cases can be distinguished: either the particle leaves through the current face (positive parameter) or does not leave the cell through this face (zero value).

This procedure is repeated until the particles reaches a cell where it remains (no exit through any faces of the new cell). Warning: to avoid high computational costs, a maximum number of 100 cells can be crossed by each particle in a single time step, otherwise it is considered lost (this maximum value can be changed).

More details can be found in appendix of [Balvet et al., 2023].

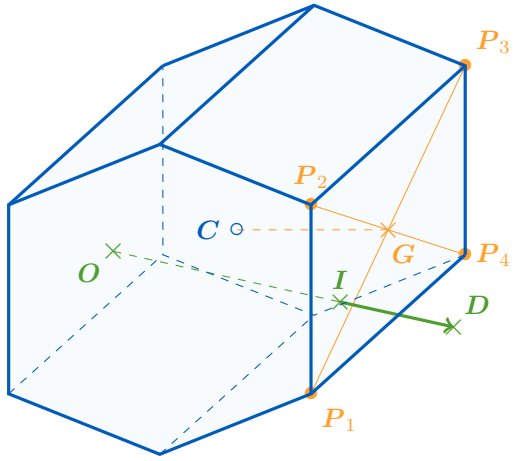


Figure II.12.1: Sketch showing a particle displacement from O to D within a cell.

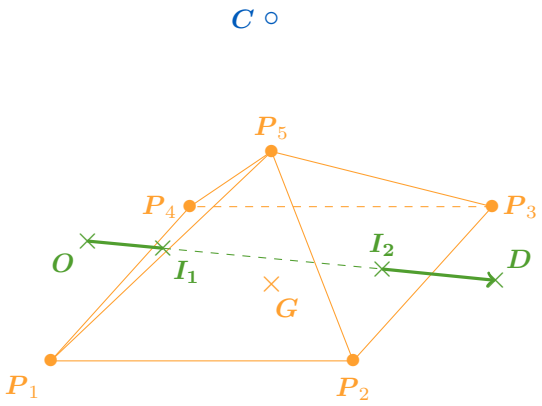


Figure II.12.2: Sketch showing a particle displacement from O to D going through a warped face.

.

Chapter 13

Atmospheric flow modelling

13.1 Atmospheric flow modelling

13.1.1 Equations of atmospheric motion

Atmospheric specific features

The following table illustrates the vertical variations observed in the standard atmosphere:

Table 13.1: vertical variation of different variables

altitude(m)	Temperature (C)	Pressure(Pa)	Density(kg/m^3)
0.	15.0	101325.	1.225
1000.	8.5	89869.	1.112
2000.	2.0	79485.	1.007
3000.	-4.5	70095.	.909
4000.	-11.0	61624.	.819
5000.	-17.5	54002.	.736
6000.	-24.0	47162.	.660
7000.	-30.5	41041.	.589
8000.	-37.0	35580.	.525
9000.	-43.5	30723.	.466
10000.	-50.0	26418.	.412
11000.	-56.5	22614.	.364

We therefore see that for motions with vertical scales that are of the order of 100m one can approximately consider the density as constant, except when thermal effects are important. Alternatively we have to take into account these variations with height as described below.

Anelastic approximation

Atmospheric flows can be both considered incompressible in the sense that their Mach number is much less than one but also flows in which the density is a function of pressure (as seen in Table 13.1).

These can be reconcilied with the anelastic approximation (see [Pielke, 1984]) in which the time derivative term $\frac{\partial \rho}{\partial t}$ is neglected in front of the other terms. The continuity equation therefore becomes:

$$\frac{\partial \rho u_i}{\partial x_i} = 0 \qquad \text{(II.13.1)}$$

This form of the continuity equation eliminates the acoustic waves, which have a very high propagation velocity, from the possible solutions.

Potential temperature

Potential temperature is derived from the temperature but is a conserved quantity through adiabatic transformation (compression/expansion). Its expression is (see [Holton, 1979] and [Stull, 1988]):

$$\theta = T\left(\frac{p_s}{p}\right)^{\left(\frac{R^\star}{C_p}\right)} \tag{II.13.2}$$

where p_s is a constant pressure conventionally taken as 1000 mb and $R^\star$ is the gas constant for dry air ($= 287\,\mathrm{Jkg^{-1}K^{-1}}$). With this variable the thermal equation becomes:

$$\rho\left(\frac{\partial\theta}{\partial t} + u_j\frac{\partial\theta}{\partial x_j}\right) \approx \left(\frac{\partial}{\partial x_j}\left(\frac{\lambda_t}{C_p}\frac{\partial\theta}{\partial x_j}\right) + \Phi\right) \tag{II.13.3}$$

Where Φ is the heating/cooling source term and λ_t is the thermal conductivity.

During their motion air parcels keep their potential temperature unless diabatic effect are present (such as condensation/evaporation of water as we will see in the micro-physics section)

Potential temperature is also used to easily characterize the local static stability of the atmosphere:

- if $\frac{\partial\theta}{\partial z}$ >0 , the thermal stratification is stable,
- if $\frac{\partial\theta}{\partial z}$ $= 0$, the thermal stratification is neutral,
- if $\frac{\partial\theta}{\partial z}$ <0 , the thermal stratification is unstable.

The distinction between the temperature and the potential temperature is important. For example in the table above the temperature is decreasing with height (by approximately 6.5 K/km) but the standard atmosphere is stably stratified with a potential temperature increasing with height.

Turbulence production by buoyancy

In standard code_saturne with the $k-\varepsilon$ turbulence model, the production or destruction rate of the turbulent kinetic energy due to buoyancy is given by:

$$G = -g\frac{\mu_t}{\sigma_t}\frac{1}{\rho}\frac{\partial\rho}{\partial z} \tag{II.13.4}$$

where σ_t is the turbulent Prandtl (Schmidt) number and μ_t is the turbulent viscosity. However for the neutral (adiabatic) atmosphere for which the buoyancy production of turbulence is zero, we have a diminution of the density with height and formula (II.13.4) is no longer valid and for atmospheric motions should be replaced by [Stull, 1988]:

$$G = -g\frac{\mu_t}{\sigma_t}\frac{1}{\theta}\frac{\partial\theta}{\partial z} \tag{II.13.5}$$

Momentum equations

The momentum equations of the standard code_saturne are not modified for small scale atmospheric flows, and in particular the Coriolis term should not be used as the Rossby number is almost always much larger than one. In very specific cases the Coriolis term and the associated pressure gradient can be introduced as user source terms.

13.2 Method of imbrication in code_saturne Atmospheric physics (MICA)

We explain in this section how meteorological profiles are used in atmospheric flow calculations for defining boundary values.

A first option is to use only one profile giving the wind (horizontal components U and V), the turbulent kinetic energy (TKE), the dissipation rate of TKE (ϵ), temperature (and optionally specific humidity and water droplet density for humid atmosphere physics) as function of the altitude. This option does not allow to take into account the horizontal gradients present in the fields given by a larger scale model, as the values given by the profile are imposed at all the boundary faces of the computational domain marked as inlet.

Generating a physically consistent marking is already a challenging task since the wind direction may vary with the altitude, justifying the introduction of the "automatic boundary conditions" flag: for each boundary face so flagged the scalar product of the wind given by the profile and the normal vector to the face is computed and that face is then deemed inlet or free outlet according to the sign of the result (the limit case 0 is considered inlet too). But if the nature (inlet or outlet) is diagnosed face by face at each time step one is not restricted to horizontally homogeneous profiles: this flag permits the use of large scale fields coming from another meteorological model , WRF (the Weather Research and Forecasting Model: www.wrf-model.org) for instance, to specify boundary values.

We describe now the method of interpolation of the large scale profiles on the "automatic boundary conditions" faces of the computational domain: the so called Cressman method.

This method has a rather long history of use in meteorological modelisation: the first publication (see [Cressman, 1959]) appeared more than 50 years ago. Its formulation is simple: in order to estimate the value of a physical variable (e.g. wind components, temperature) at a grid point from scattered data points one uses a weighted average of the values. The weights of the different data are function of the displacement between the data points and the estimation point, very often function of their distance only. In the original publication the function was a clipped rational function of the distance while the method used in code_saturne atmospheric flows uses a Gaussian kernel like in [Barnes, 1964]. If we denote the Cartesian coordinates of the scattered data points by $(x_i y_i z_i)$, by (x, y, z) the coordinates of the generic estimation point and by Vi the values of the variable of interest the estimate reads:

$$\tilde{V} = \frac{\sum_i V_i \, f(x - x_i, y - y_i, z - z_i)}{\sum_i f(x - x_i, y - y_i, z - z_i)}. \tag{II.13.6}$$

This will be a well-defined weighted average as long as the weight function f is positive and the sum appearing as denominator strictly positive. For application in code_saturne the following form of f has been chosen:

$$f\,(x, y, z) = \exp\left(-\frac{x^2 + y^2}{4R_H^2} - \frac{z^2}{4R_V^2}\right) \tag{II.13.7}$$

The radius R_H (R_V) is the so called horizontal (vertical) range of the method. Their influence on the estimates is easy to describe. When the horizontal (vertical) distance of a data point to the estimation point is much larger than the corresponding range this data will see its contribution to the estimate diminish. Only points relatively close to the estimation point will yield appreciable contributions to this estimate. Please notice also that even at the data point the formula will give an estimate different from the data value. The estimation is not an exact interpolator, but a smoothing estimator.

13.3 Radiation parameterizations for atmosphere

For these parameterizations we distinguish two spectral domains: visible ($0.2\,\mu$m $- -5\,\mu$m) for solar radiation and infrared ($5\,\mu$m $- -100\,\mu$m) for thermal radiation.

13.3.1 Thermal infrared radiation

The radiation transfer equation applied to a plan parallel atmosphere (1-D) using the two stream approximation for a non-scattering medium with the broadband emissivity approximation for spectral integration (see [Liou, 2002] and [Makke et al., 2016]) leads to the expressions of the upward and downward fluxes:

$$F^{\uparrow} = \sigma T_g^4 \left(1 - \epsilon\left(z, 0\right)\right) - \int_0^z \sigma T^4\left(z'\right) \frac{\mathrm{d}\epsilon}{\mathrm{d}z'}\left(z, z'\right) \mathrm{d}z' \tag{II.13.8}$$

$$F^{\downarrow} = \int_z^{\infty} \sigma T^4\left(z'\right) \frac{\mathrm{d}\epsilon}{\mathrm{d}z'}\left(z, z'\right) \mathrm{d}z' \tag{II.13.9}$$

Where T is the fluid temperature in K, T_g the ground surface temperature in K,

σ the Stefan Boltzman constant and ϵ the atmosphere emissivity defines by:

$$\epsilon\left(z, z'\right) = \frac{1}{\sigma T^4} \int_0^{\infty} \left(1 - I_{\lambda}\left(z, z'\right)\right) \pi I_{\lambda}^0\left(z, z'\right) \mathrm{d}\lambda \tag{II.13.10}$$

Where $I_{\lambda}^0 = 2hc^2\lambda^{-5}\left(e^{\frac{hc}{k_B \lambda T}} - 1\right)$ is the Planck function,
I_{λ} the transmittance and k_B is the Boltzman constant.

If the ground surface is not considered as a black body, with a ground emissivity ϵ_g the upward flux can be written (see [Ponnulakshmi et al., 2012]):

$$F^{\uparrow} = \epsilon_g \sigma T_g^4 \left(1 - \epsilon\left(z, 0\right)\right) + \left(1 - \epsilon_g\right) \int_0^{\infty} \sigma T^4\left(z'\right) \epsilon\left(-z, z'\right) \mathrm{d}z' - \int_0^z \sigma T^4\left(z'\right) \frac{\mathrm{d}\epsilon}{\mathrm{d}z'}\left(z', z\right) \mathrm{d}z' \tag{II.13.11}$$

In this expression, the second term can be considered as a downward radiation along $-z$ and infinite. Integration by parts and supposing isothermal atmospheric layers above $z_t = 11\,000\,\mathrm{m}$ leads to the following expressions for upward and downward fluxes:

$$F^{\downarrow} = \epsilon\left(\infty, 0\right) \sigma T^4\left(z_t\right) - \int_z^{z_t} \epsilon(z', z) \frac{\mathrm{d}\sigma T^4(z')}{\mathrm{d}z'} \mathrm{d}z' \tag{II.13.12}$$

$$F^{\uparrow} = \epsilon_g T_g^4 + \int_0^z \frac{\mathrm{d}\left(\sigma T^4\left(z'\right)\right)}{\mathrm{d}z'} \epsilon\left(z', z\right) \mathrm{d}z' + \left(1 - \epsilon_g\right) \left[\sigma T^4\left(z_t\right) \epsilon\left(\infty, -z\right) - \int_0^{z_t} \frac{\mathrm{d}\sigma T^4\left(z'\right)}{\mathrm{d}z'} \epsilon(-z, z') \mathrm{d}z'\right] \tag{II.13.13}$$

The heating/cooling for atmospheric layers is deduced from the divergence of the net Flux:

$$F_n = F^{\uparrow} - F^{\downarrow}. \tag{II.13.14}$$

$$\begin{aligned} \frac{\mathrm{d}F_n}{\mathrm{d}z} = & \int_0^z \frac{\mathrm{d}\left(\sigma T^4\left(z'\right)\right)}{\mathrm{d}z'} \frac{\mathrm{d}\epsilon\left(z', z\right)}{\mathrm{d}z'} \mathrm{d}z' + \left(1 - \epsilon_g\right) \left[\sigma T^4\left(z_t\right) \frac{\mathrm{d}\epsilon\left(\infty, -z\right)}{\mathrm{d}z'}\right. \\ & \left. - \int_0^{z_t} \frac{\mathrm{d}\sigma T^4\left(z'\right)}{\mathrm{d}z'} \frac{\mathrm{d}\epsilon\left(-z, z'\right)}{\mathrm{d}z'} \mathrm{d}z'\right] - \frac{\mathrm{d}\epsilon\left(\infty, 0\right)}{\mathrm{d}z} \sigma T^4\left(z_t\right) + \int_z^{z_t} \frac{\mathrm{d}\epsilon(z', z)}{\mathrm{d}z'} \frac{\mathrm{d}\sigma T^4(z')}{\mathrm{d}z'} \mathrm{d}z' \end{aligned} \tag{II.13.15}$$

Clear sky

To determine emissivity it is common to introduce the corrected path length $u_g(z, z')$ for major absorbing gases: water vapor, carbon dioxide and ozone.

$$u_g\left(z,z'\right)=\int_{z'}^{z}\rho_g\left(z"\right)\Psi\left(T_0,P_0,T,P\right)\,\mathrm{d}z" \tag{II.13.16}$$

where Ψ is a scaling function eliminating the dependence of the absorption coefficient on pressure and temperature. These corrections, known as the one-parameter scaling approximation, are attributed to [Chou and Arking, 1980]. The emissivity $\epsilon(z,z')$ for each gas is now a function of the scaled path $\epsilon(u_g(z,z'))$. The total emissivity for the three dominant atmospheric gases in the infrared domain is:

$$\begin{aligned}\epsilon\left(z,z'\right)=\ &\epsilon_{H_2O}(u_{H_2O}\left(z,z'\right)+\epsilon_{O_3}(u_{O_3}\left(z,z'\right)\\&+\epsilon_{CO_2}(u_{CO_2}\left(z,z'\right)T_{15\mu}\left(u_{H_2O}\left(z,z'\right)\right)+\epsilon_{dim}(u_{dim}\left(z,z'\right)T_w\left(u_{H_2O}\left(z,z'\right)\right)\end{aligned} \tag{II.13.17}$$

The optical thickness are:

$$u_{H_2O}\left(z,z'\right)=\int_{z'}^{z}\rho_{H_2O}\left(z"\right)\left(\frac{P(z")}{P_0}\sqrt{\frac{T_0}{T(z")}}\right)^n\,\mathrm{d}z" \tag{II.13.18}$$

$$u_{CO_2}\left(z,z'\right)=\int_{z'}^{z}\rho_{CO_2}\left(z"\right)\left(\frac{P(z")}{P_0}\right)^{3/4}\,\mathrm{d}z" \tag{II.13.19}$$

$$u_{O_3}\left(z,z'\right)=\left|f_{Green}\left(z\right)-f_{Green}(z')\right| \tag{II.13.20}$$

$$u_{dim}\left(z,z'\right)=\int_{z'}^{z}\rho_{H_2O}\left(z"\right)\,e(z")\mathrm{d}z"\,f(T(z"),T_0)\mathrm{d}z" \tag{II.13.21}$$

Where $f\left(T\left(z\right),T_0\right)=\exp\left[1800\left(\frac{1}{T(z)}-\frac{1}{296.0}\right)\right]$ and $f_{Green}\left(z\right)=a\frac{1+e^{-b/c}}{1+e^{(z-b)/c}}$ with $a=0.1\,\mathrm{cm/STP}$ (Standard Temperature and Pressure), $b=20\,\mathrm{km}$, c chosen as $b/c=5$ (see [Green, 1964])

The associated emissivities are:

[Sasamori, 1968] formula for water vapor, carbon dioxide and ozone:

$$\epsilon_{H_2O}\left(u_{H_2O}\right)=\begin{cases}0.846*\left(3.59\times10^{-5}+\frac{u_{H_2O}}{10.}\right)^{0.243} & \text{for } u_{H_2O}<0.1\,\mathrm{kg\,m^{-2}}\\ 0.24\log_{10}\left(0.01+\frac{u_{H_2O}}{10.}\right)+0.622 & \text{for } u_{H_2O}\geq0.1\,\mathrm{kg\,m^{-2}}\end{cases} \tag{II.13.22}$$

$$\epsilon_{O_3}\left(u_{O_3}\right)=\begin{cases}0.209*\left(u_{O_3}+7.0\times10^{-5}\right)^{0.436}-3.21\times10^{-3} & \text{for } u_{O_3}<0.01\,\mathrm{cm/}STP\\ 7.49\times10^{-2}+2.12\times10^{-2}\log_{10}\left(u_{O_3}\right) & \text{for } u_{O_3}\geq0.01\,\mathrm{cm/}STP\end{cases} \tag{II.13.23}$$

$$\epsilon_{CO_2}\left(u_{CO_2}\right)=\begin{cases}0.0676*\left(0.01022+u_{C02}\right)^{0.421}-0.00982 & \text{for } u_{CO_2}<1\,\mathrm{cm/}STP\\ 0.24\,log_{10}\left(0.0676+u_{CO_2}\right)+0.622 & \text{for } u_{CO_2}\geq1\,\mathrm{cm/}STP\end{cases} \tag{II.13.24}$$

$$T_{15\mu}\left(u_{H_2O}\right)=\begin{cases}1.33-0.832\left(0.0286+\frac{u_{H_2O}}{10.}\right)^{0.26} & \text{for } u_{H_2O}<20\,\mathrm{kg\,m^{-2}}\\ 0.33-0.2754\left(log_{10}\left(\frac{u_{H_2O}}{10.}\right)-0.3011\right) & \text{for } u_{H_2O}\geq20\,\mathrm{kg\,m^{-2}}\end{cases} \tag{II.13.25}$$

(see [Veyre et al., 1980]) for water vapor dimer:

$$\epsilon_{dim}\left(u_{dim}\right) = \begin{cases} 0.4614\left[1-\left(\sum_{i=0}^{2} a_i u_{dim}^i\right)/\left(\sum_{j=0}^{3} b_j u_{dim}^j\right)\right] & \text{for } u_{dim} \leq 0.5\,\mathrm{g\,cm^{-2}} \\ 0.4614 \text{ for } u_{dim} > 0.5\,\mathrm{g\,cm^{-2}} \end{cases}$$

with

$$a_0 = 0.015075,\ a_1 = -0.036185,\ a_2 = 0.0019245,\ b_0 = a_0, b_1 = 0.19547,\ b_2 = 0.75271,\ b_3 = 1 \tag{II.13.26}$$

$$T_w\left(u_{H_2O}\right) = \left(\sum_{i=0}^{4} a_i^{'} u_{dim}^i\right)/\left(\sum_{j=0}^{5} b_j^{'} u_{dim}^j\right)$$

with

$$a_0^{'} = 7.76192\times10^{-7},\ a_1^{'} = 1.33836\times10^{-3},\ a_2^{'} = 0.166649,\ a_3^{'} = 2.17686,\ a_4^{'} = 2.6902 \tag{II.13.27}$$

$$b_0^{'} = 7.79097\times10^{-7},\ b_1^{'} = 1.36832\times10^{-3},\ b_2^{'} = 0.179601,\ b_3^{'} = 2.70573,\ b_4^{'} = 5.15119,\ b_5^{'} = 1 \tag{II.13.28}$$

Cloudy sky

The transmittance by cloud droplets can be considered as grey body transmittance and overlapping by gas absorption is taken into account as in [Sasamori, 1972].

$$\epsilon\left(z',z\right) = 1-\left(1-\epsilon_{gas}\left(z',z\right)\right)\tau_l\left(z',z\right) \tag{II.13.29}$$

with $\tau_l(z'z) = 1-\epsilon_l(z',z) = \exp(-K_l\int_{z'}^{z}\rho(z'')q_l\left(z''\right)\mathrm{d}z''\ )$

and $\mathrm{K_l}$ =1.66 $3/4\mathrm{r}_e$ where r_e is the effective radius of the cloud droplet.

In case of partial cloudiness N_{max}, the transmittance for liquid water can be modified following [Bougeault, 1985].

$$\tau_l\left(z',z\right) = 1-N_{max}\left(z',z\right)+N_{max}(z',z)\exp(-K_l\int_{z'}^{z}\rho(z'')q_l\left(z''\right)\mathrm{d}z'') \tag{II.13.30}$$

13.3.2 Solar radiation

The first step consists to determine astronomic factors linked to earth-sun position

Determination of zenithal angle and correction factors for solar constant (see [Paltridge and C.M.R., 1974])

$$\mu_0 = \cos\theta = \sin\delta\sin\phi+\cos\delta\cos\phi\cos th \tag{II.13.31}$$

Where φ is the latitude, δ the inclination which depends on the day of year:

$$\begin{aligned}\delta = 0.006918 &- 0.399912\cos\theta_0+0.070257\sin\theta_0-0.006758\cos2\theta_0 \\ &+0.000907\sin2\theta_0-0.002697\cos3\theta_0+0.001480\sin3\theta_0\end{aligned} \tag{II.13.32}$$

with $\theta_0 = 2\pi J/365$

The solar local time th= local time + longitude correction + Eq (Time equation).

$$Eq = 0.000075 + 0.001868\cos\theta_0 - 0.032077\sin\theta_0 - 0.014615\cos 2\theta_0 - 0.040849\sin 2\theta_0 \quad \text{(II.13.33)}$$

A correction for sun-earth distance is taken into account

$$F = F_0(1.00011 + 0.034221\cos\theta_0 + 0.001280\sin\theta_0 + 0.000719\cos 2\theta_0 + 0.000077\sin 2\theta_0) \quad \text{(II.13.34)}$$

with F_0 =1367 W m^{-2}.

In order to take into account earth curvature a correcting term is added:

$$\mu_0 = \frac{r_1}{\sqrt{\mu_0^2 + r_1(r_1+2)} - \mu_0} \; with\, r_{1=} \frac{H_{atmo}}{R_{earth}} = \frac{8\,km}{6371\,km} \quad \text{(II.13.35)}$$

Solar radiation in clear sky without aerosols

The radiation transfer equation applied to a plan parallel atmosphere (1D) using the two stream approximation in a non-scattering medium with an external radiation source term (the sun) for the global radiation (direct + diffuse) leads to the expression of downward and upward fluxes for a spectral band $\Delta\nu$ (see [Lacis and Hansen, 1974]):

$$S^{\downarrow} = \mu_0 F_0 \tau_{\Delta\nu}\, u_{gas}\,(z, \infty) \quad \text{(II.13.36)}$$

$$S^{\uparrow} = \mu_0 F_0 R_g \tau_{\Delta\nu}\,(u_{gas}(0,\infty), u_{gas}(0,z)) \quad \text{(II.13.37)}$$

Where F_0 is the solar constant, μ_0 is the corrected cosines of the zenithal angle, $\tau_{\Delta\nu}$ the transmittance and u_{gas} the optical thickness for the considered gas.

The solar spectrum is divided in two bands: One for ozone (0.2-0.7 μm) where absorption occurs in the upper layers of the atmosphere, a second one for water vapor (0.7-5 μm) part where absorption occurs in the lower part of the atmosphere.

Ozone Band

For ozone band, the Rayleigh diffusion is parameterized as an albedo for the lower part of the atmosphere combined with earth surface albedo. With these approximations the heating rate due to ozone in the layer (l, l+1) is (LH74):

$$F_{abs}^{O_3} = \mu_0 F\,[\,A_{O_3}\,(x_{l+1}) - A_{O_3}\,(x_l) - \bar{R}(\mu_0)\left(A_{O_3}\left(x_{l+1}^{\star}\right) - A_{O_3}\left(x_l^{\star}\right)\right)] \quad \text{(II.13.38)}$$

With $x = Mu_{O_3}\,(\infty, z)\,, \quad x^{\star} = Mu_{O_3}\,(\infty, 0) + \bar{M}(u_{O_3}\,(\infty, 0) - u_{O_3}\,(\infty, z))$
where $\bar{M} = 1.9$ and $\quad M = 35/(1224\mu_0^2 + 1)^{1/2}$.

$\bar{R}(\mu_0)$ is the composite albedo including Rayleigh diffusion effect and ground reflection

$\bar{R}\,(\mu_0) = \overline{R_a}\,(\mu_0) + (1 - \overline{R_a}\,(\mu_0))(1 - \overline{R_a^{\star}})R_g/(1 - \overline{R_a^{\star}}R_g)$

with $\overline{R_a}\,(\mu_0) = 0.213/(1 + 0.816\mu_0)$ and $\overline{R_a^{\star}} = 0.144$.

The ozone absorption is parameterized as LH74 for both Chappuis and ultraviolet bands:

$A_{O_3}\,(x) = \frac{0.02118x}{1+0.042x+0.000323x^2} + \frac{1.082x}{(1+138.6x)^{0.805}} + \frac{0.0658x}{1+(103.6x)^3}$ where x is in cm/STP.

The global downward flux in the O3 band is (LH74):

$F^{\downarrow}{}_{g}^{O_3} = \mu_0 F(0.647 - A_{O_3}(x) - \overline{R_r}(\mu_0))(1 - \overline{\overline{R_r^\star}} R_g)$
with $\overline{R_r}\mu_0 = 0.28/(1 + 6.43\mu_0)$
and $\overline{\overline{R_r^\star}} = 0.0685$

The direct downward flux is (see [Atwater and Ball, 1978]):

$$F^{\downarrow}{}_{d}^{O_3} = \mu_0 F\left(0.647 - A_d^{O_3}(z)\right) with A_d^{O_3}(z) = 1 - (1.041 - 0.16 * \sqrt{M\left(0.949\,10^{-3} P(z) + 0.051\right)}) \quad \text{(II.13.39)}$$

Vapor water band

For water vapor band (LH74):

$$F_{abs}^{H_2O} = \mu_0 F\left[A_{H_2O}(y_{l+1}) - A_{H_2O}(y_l) - R_g\left(A_{H_2O}\left(y_{l+1}^\star\right) - A_{H_2O}\left(y_l^\star\right)\right)\right] \quad \text{(II.13.40)}$$

With $y = M u_{H_2O}(\infty, z)$, $y^\star = M u_{H_2O}(\infty, 0) + \frac{5}{3}(u_{H_2O}(\infty, 0) - u_{H_2O}(\infty, z))$

The direct downward flux is equal to the global downward flux:

$$F^{\downarrow}{}_{d}^{H_2O} = F^{\downarrow}{}_{g}^{H_2O} = \mu_0 F(0.353 - A_{H_2O}(y_l)) \quad \text{(II.13.41)}$$

The water vapor absorption is parameterized as in [Yamamoto, 1962]

$A_{H_2O}(y) = \frac{0.29}{(1.+14.15y)^{0.635}+0.5925y}$ for y in kg m^{-2}

Solar radiation for cloudy atmosphere or in presence of aerosols

Ozone Band

For ozone band, a similar expression is used to clear sky formulation by replacing Rayleigh diffusion by cloud droplets diffusion in the composite albedo including ground reflection. That leads to the following expression for solar heating:

$$F_{abs}^{c,a,O3} = \mu_0 F\left[A_{O_3}(x_{l+1}) - A_{O_3}(x_l) - \bar{R}(\mu_0)\left(A_{O_3}\left(x_{l+1}^\star\right) - A_{O_3}\left(x_l^\star\right)\right)\right] \quad \text{(II.13.42)}$$

$\bar{R}(\mu_0)$ is the composite albedo including cloud droplet (c) or aerosol (a) diffusion effect and ground reflection

$$\bar{R}(\mu_0) = \overline{R_{c,a}}(\mu_0) + (1 - \overline{R_{c,a}}(\mu_0))(1 - \overline{R_{c,a}^\star}) R_g/(1 - \overline{R_{c,a}^\star} R_g) \quad \text{(II.13.43)}$$

With $\overline{R_{c,a}}(\mu_0) = \overline{R_{c,a}^\star} = \sqrt{3}(1 - g_{c,a})\tau_{c,a}/(2 + \sqrt{3}(1 - g_{c,a})\tau_{c,a})$ Where $g_{c,a}$ is the asymmetry factor of the cloud or aerosols particles phase function,
g_c =0.85, g_a = 0.66, $\tau_{a,c}$ is the total visual optical thickness for cloud liquid water and aerosol concentration:

$\tau_c = 1.5 \int_0^{z_t} \frac{\rho q_l}{r_e} dz$ and $\tau_a = 1.5 \int_0^{z_t} \frac{\rho C_{aero}}{r_{aero}} dz$ (see [Stephens, 1984]).

In case of partial cloudiness N_{max}, the heating rate and the downward flux can be weighted between clear and cloudy sky as follows:

$$F_{abs}^{t,O3} = N_{max} F_{abs}^{c,O3} + (1 - N_{max}) F_{abs}^{O_3} \quad \text{(II.13.44)}$$

The effect of aerosol particles is only taken into account in the clear sky layers with aerosol cloud fraction equal to unity and the absorption in that case is $F_{abs}^{a,O3}$.

The global downward and direct fluxes are:

$$F_g^{\downarrow c,a,O3} = \mu_0 F(0.647 - A_{O_3}(x) - \overline{R_r}(\mu_0))(1 - \overline{R_{c,a}}(\mu_0)/(1 - \overline{R_{c,a}^{\star}} R_g), \quad F_d^{\downarrow c,a,O3} = 0. \quad \text{(II.13.45)}$$

That gives if partial cloudiness is taken into account:

$$F_g^{\downarrow t,O3} = N_{max} F_g^{\downarrow c,a,O3} + (1 - N_{max}) F_g^{\downarrow O_3} \quad \text{(II.13.46)}$$

$$F_d^{\downarrow t,O3} = (1 - N_{max}) F_d^{\downarrow O_3} \quad \text{(II.13.47)}$$

Vapor water band

For water vapor, the diffusion processes are directly modelled using the adding method with k distribution method for overlapping between liquid and vapor water (LH74).

For each cloud layer l in the frequency interval n, the optical thickness $\tau_{l,n}$, the single scattering albedo $\omega_{l,n}$ and the asymmetry factor g are:

$$\tau'_{l,n} = \tau_c + \tau_a + k_n u_{H_2O} \quad \text{(II.13.48)}$$

$$\omega'_{l,n} = (\omega_c \tau_c + \omega_a \tau_a)/\tau'_{l,n} \quad \text{(II.13.49)}$$

$$g' = (\omega_c \tau_c g_c + \omega_a \tau_a g_a)/\omega'_{l,n} \tau'_{l,n} \quad \text{(II.13.50)}$$

[Joseph et al., 1976] corrections are used in order to describe highly forward scattering for cloud droplets. That gives with $f = g'^2$ (see [Stephens, 1984]):

$$\tau_{l,n} = \tau'_{l,n}(1 - f\omega'_{l,n}) \quad \text{(II.13.51)}$$

$$\omega_{l,n} = \omega'_{l,n}(1 - f)/(1 - f\tau'_{l,n}) \quad \text{(II.13.52)}$$

$$g = (g' - f)/(1 - f) \quad \text{(II.13.53)}$$

The transmission and reflection functions for global radiation are:

For clear layer:

$R_l = R_l^{\star} = 0\,, \quad T_l = T_l^{\star} = \exp(-\frac{5}{3}\tau_{l,n})$ except for a clear layer above the highest cloud layer for which $T_l = \exp(-M\tau_{l,n})$

For cloudy layer:

$$R_l = \frac{(u+1)(u-1)\left(e^t - e^{-t}\right)}{(u+1)^2 e^t - (u-1)^2 e^{-t}}\,, \; T_l = \frac{4u}{(u+1)^2 e^t - (u-1)^2 e^{-t}} \quad \text{(II.13.54)}$$

with

$u = \sqrt{\frac{1-g\omega_{l,n}}{1-\omega_{l,n}}}$ and $t = \tau_{l,n}\sqrt{3(1-\omega_{l,n})(1-g\omega_{l,n})}$
The transmission function for direct solar radiation is

$T_l = \exp(-\frac{5}{3}\tau_{l,n})$ for clear layers and

$T_l = \exp(-M\tau_{l,n})$ for cloudy layers

In the case where cloud fraction is taken into account the reflection and transmission functions can be weighted by the cloud fraction

$N_l : T_l = N_l T_{l,cloud} + (1\text{-}N_l) T_{l,clear}$ *and* $R_l = N_l R_{l,cloud}$

The following five steps are carried out for each value of k_n which can yield significant absorption, *e.g.*, n=1-8 for the discrete distribution given in the following table:

Table 13.2: Absorption

$k_n; n \in [1,8]$	4.10^{-6}	2.10^{-4}	0.0035	0.0377	0.195	0.94	4.46	19
$p\,(k_n)$	0.6470	0.0698	0.1443	0.0584	0.0335	0.0225	0.0158	0.0087

1. R_l and T_l for $l = 1$, L are computed for each layer
2. The layers are added, going down, to obtain $R_{1,l}$ and $T_{1,l}$ as follow:

$$R_{1,l} = R_{1,l-1} + \frac{T_{1,l-1} R_l T^\star_{1,l-1}}{1 - R^\star_{1,l-1} R_l} \;,\; T_{1,l} = \frac{T_{1,l-1} T_l}{1 - R^\star_{1,l-1} R_l} \tag{II.13.55}$$

   With similar expressions for $R*$ and $T*$
3. Layers are added one at a time, going up, to obtain $R_{L+1,l}$ and $T_{L+1,l}$
4. As two composite layers, say 1,l and l+1, L+1 are added, the upward and downward fluxes boundary between the two layers are determined:

$$U_l = \frac{T_{1,l}\, R_{L+1,l}}{1 - R^\star_{1,l} R_{L+1,l+1}} \;,\; D_l = \frac{T_{1,l}}{1 - R^\star_{1,l} R_{L+1,l+1}} \tag{II.13.56}$$

   The fraction of the total incident flux absorbed in the upper composite layer is:

$$A_{1,l(n)} = p\,(k_n)\left(1 - R_{1,L+1(n)}\right) + U_{l(n)} - D_{l(n)} \tag{II.13.57}$$

   The total absorption in each layer l is found by differencing, *e.g.*,

$$A_{l(n)} = A_{1,l(n)} - A_{1,l-1(n)} \tag{II.13.58}$$

   The total absorption in each layer l is found by summing over the values of n for which k_n is significant

$$F^{H_2O}_{abs} = \mu_0 F \sum_{n=1}^{n=8} A_{l(n)} \tag{II.13.59}$$

   The downward flux for global radiation is:

$$F^{\downarrow H_2O}_{g} = \mu_0 F \sum_{n=1}^{n=8} D_l\,(n)\, p\,(k_n) \tag{II.13.60}$$

For the downward direct radiation the layers are added as in the step 2) but only for the transmission function and with reflection equal to zero: $T_{1,l} = T_{1,l-1} T_l$ Since the downward direct radiation is:

$$F^{\downarrow H_2O}_{d} = \mu_0 F \sum_{n=1}^{n=8} T_{1,l}\,(n)\, p(k_n) \tag{II.13.61}$$

13.4 Warm cloud parameterization

In order to represent cloud formation new thermodynamic variables are used to be conservative through condensation/evaporation processes. These variables are (see [Betts, 1973]):

$$\theta_l = \theta - \frac{L_v \theta}{C_p T} q_l, \quad q_w = q_v + q_l \tag{II.13.62}$$

where θ_l (K) is the liquid-water potential temperature,
θ(K) the potential temperature,T(K) the temperature,
q_w (kg kg^{-1}) the total water specific humidity (sum of the specific humidity for water vapor q_v (kg kg^{-1}) and liquid water q_l (kg kg^{-1})),
L_v =2.5 (MJ kg^{-1}) and C_p = 1005 (J kg^{-1} K^{-1}).
The ensemble mean of each quantity is denoted by an overbar, whereas primes denotes fluctuating quantities (*e.g.* $q = \bar{q} + q'$).

The energy and water conservation equations become with these variables:

$$\left(\frac{\partial}{\partial t} + \overline{u}_i \frac{\partial}{x_i}\right) \overline{\theta}_l = \frac{\partial}{x_i}\left[\left(\frac{\lambda_c}{C_p} + \frac{\mu_t}{\sigma_t}\right)\frac{\partial \overline{\theta}_l}{\partial x_i}\right] + \frac{\overline{\theta}}{\overline{T}}\frac{1}{C_p}\frac{\partial F_R}{\partial z} - \rho \frac{L}{C_p}\frac{\overline{\theta}}{\overline{T}}\left(\frac{\partial \overline{q}_l}{\partial t}\right)_{sed} \tag{II.13.63}$$

$$\left(\frac{\partial}{\partial t} + \overline{u}_i \frac{\partial}{x_i}\right) \overline{q_w} = \frac{\partial}{x_i}\left[\left(\frac{\lambda_c}{C_p} + \frac{\mu_t}{\sigma_t}\right)\frac{\partial \overline{q}_w}{\partial x_i}\right] + \rho\left(\frac{\partial \overline{q_l}}{\partial t}\right)_{sed} \tag{II.13.64}$$

Where ρ is the air density, u_i the wind components, λ_c the thermal diffusivity, μ_t the turbulent viscosity, σ_t the turbulent Prandtl number and Fr the net radiative flux.

To be comprehensive, we have included the settling term $\left(\frac{\partial \overline{q_l}}{\partial t}\right)_{sed}$ that can play an important role in some cases, as shown for example for fog in [Musson-Genon, 1987].

With the set of variables used, the liquid water content q_l must be diagnosed. This is done by using a subgrid condensation scheme described in details in [Bouzereau et al., 2007].

13.4.1 Condensation processes in θ_l and q_w variables

q_l is diagnosed from the predicted value of q_w by using a subgrid condensation scheme (see [Bouzereau et al., 2007]) which can take into account temperature and specific humidity turbulent fluctuations inside the resolved mesh:

$$q_l = q_w - q_s \tag{II.13.65}$$

$$q_w - q_s = A_l\left(\overline{q_w} - \overline{q_{sl}}\right) + A_l\left(q_w^{'} - \alpha_l \theta_l^{'}\right) \tag{II.13.66}$$

with $A_l = \left(1 + \frac{{L_v}^2 \overline{q_{sl}}}{C_p R_v \overline{T_l}^2}\right)^{-1}$ and $\alpha_l = \frac{L_v \overline{q_{sl}}}{R_v \overline{T_l}^2}\frac{\overline{T}}{\overline{\theta}}$

where q_s (kg kg^{-1}) is water vapor specific humidity at saturation level and R_v is the specific gas constant for water vapor.

$with\, q_s = \frac{0.622\, e_{sat}}{P - 0.378 e_{sat}}$, $e_{sat} = 610.78 \exp(17.269 * \frac{\overline{T} - 273.15}{\overline{T} - 35.86})$ *and* $\overline{q_{sl}} = q_s\left(\overline{T_l}\right)$

where e_{sat} is saturation vapor pressure and P atmospheric pressure.

We assume a bivariate normal distribution function $\tilde{G}(\theta_l q_w)$ in order to represent the subgrid-scale fluctuations of the variables θ_l and q_w:

$$R = \int_{-\infty}^{+\infty}\int_{q_s}^{+\infty} \tilde{G}\left(\theta_l, q_w\right) dq_w \, \mathrm{d}\theta_l \tag{II.13.67}$$

$$\overline{q_w} = \int_{-\infty}^{+\infty} \int_{q_s}^{+\infty} (q_w - q_s) \tilde{G}\left(\theta_l, q_w\right)\, dq_w\, \mathrm{d}\theta_l \tag{II.13.68}$$

We reduce the integration to a single variable $s = \bar{s} + s^{'}\ with\, s = \frac{A_l(\overline{q_w} - \overline{q_{sl}})}{2}, s' = A_l(q_w^{'} - \alpha_l \theta_l^{'})/2$

and introduce $Q_1 = \bar{s}/s'$ a dimensionless measure of the deviation of the mean state from saturation and $t = s^{'}/\sigma_{s'}$ with $\sigma_{s'} = (\frac{A_l}{2})(\overline{q_w'^2} + \alpha_l^2 \overline{\theta_l'^2} - 2\alpha_l \overline{\theta_l' q_w'})$.

In that condition, partial cloudiness R, cloud water specific humidity, and second order momentum can be estimated for different subgrid distribution (all or nothing and gaussian) and are given in the following table:

Table 13.3:

Subgrid distribution	"All or nothing" distribution	Gaussian distribution
G(t)	$\delta(t)$ Dirac distribution	$\frac{1}{\sqrt{2\pi}} e^{-t^2/2}$
R	$Q_1 \geq 0, \quad 1$ $Q_1 < 0, \quad 0$	$\frac{1}{2}\left(1 + erf\left(Q_1/\sqrt{2}\right)\right)$
$\frac{\overline{q_l}}{2\sigma_{s'}}$	$Q_1 \geq 0, \quad Q_1$ $Q_1 < 0, \quad 0$	$RQ_1 + \frac{e^{-Q_1^2/2}}{\sqrt{2\pi}}$
$\frac{\overline{s' q_l'}}{2\sigma_{s'}^2}$	0	$R - \frac{\bar{q}_l}{2\sigma_{s'}} \frac{e^{-Q_1^2/2}}{\sqrt{2\pi}}$

We have now to determine $\overline{q_w'^2}, \overline{\theta_l'^2}$ and $\overline{\theta_l' q_w'}$. This can be done by stationarising (balance between production and dissipation) the equation of evolution of the second order moments. That gives (see [Musson-Genon, 1995]):

$$\overline{q_w'^2} = \frac{2}{c_2}\frac{k}{\varepsilon} K_\theta \left(\frac{\partial \overline{q_w}}{\partial z}\right)^2, \quad \overline{\theta_l'^2} = \frac{2}{c_2}\frac{k}{\varepsilon} K_\theta \left(\frac{\partial \overline{\theta_l}}{\partial z}\right)^2, \quad \overline{\theta_l' q_w'} = -\frac{2}{c_2}\frac{k}{\varepsilon} K_\theta \left(\frac{\partial \overline{q_w}}{\partial z}\right)\left(\frac{\partial \overline{\theta_l}}{\partial z}\right) \tag{II.13.69}$$

With c_2 =2.3 and $K_\theta = c_\mu \frac{k^2}{\varepsilon}$ for the $k-\varepsilon$ turbulent closure

In addition, the buoyancy term in the $k-\varepsilon$ turbulent closure has to be expressed in terms of θ_l and q_w variables. This leads to the following expression (see [Bouzereau et al., 2007]).

$$\overline{w'\theta_v'} = E_\theta \overline{w'\theta_l'} + E_q \overline{w' q_w'} \tag{II.13.70}$$

where

(II.13.71)

$E_\theta = \tau - N A_l \alpha_l D_q\, and\, E_q = 0.608\,\theta + N A_l D_q$ and $\tau = (1 + 0.608\,\overline{q_w} - 1.608\,\bar{q}_l)$, $D_q = \frac{g}{T_0} - 1.608\,\bar{\theta}$ and $N = \overline{s' q_l'}/2\sigma_{s'}^2$ T_0 being the mean temperature of the atmospheric boundary-layer.

13.4.2 Evolution of the droplet spectrum

For computing the evolution of the droplet spectrum, we have chosen a semi-spectral method for its simplicity and its low computational cost.

The prognostic equation for the number of droplet N_d is (here N_d is the ensemble mean of droplet number):

$$\left(\frac{\partial}{\partial t}+\bar{u_i}\frac{\partial}{\partial x_i}\right)N_d=\frac{1}{\rho}\frac{\partial}{x_i}\left[\left(\frac{\lambda_c}{C_p}+\frac{\mu_t}{\sigma_t}\right)\frac{\partial\overline{q_w}}{\partial x_i}\right]+\left(\frac{\partial\bar{q_l}}{\partial t}\right)_{E/C}+\left(\frac{\partial\bar{q_l}}{\partial t}\right)_{nuc}+\left(\frac{\partial\bar{q_l}}{\partial t}\right)_{sed} \quad \text{(II.13.72)}$$

where the last three terms of the right-hand side are respectively the evaporation and condensation term, the nucleation term and the droplet settling term. Besides, this equation is associated with a log-normal distribution law defined as follows:

$N_d\left(r\right)=\frac{N_d}{r\sigma_c\sqrt{2\pi}}exp\left[\frac{-1}{2\sigma_c^2}\left(ln\frac{r}{r_0}\right)^2\right]$

where r is the droplet radius, r_0 is the distribution median value and σ_c the logarithmic standard deviation.

With this law, the relationship between liquid water content q_l and N_d is:

$$\rho\bar{q_l}=\frac{4}{3}\pi\bar{r_3}^3\rho_w N_d \quad \text{(II.13.73)}$$

Where $\bar{r_3}=r_0exp\left(\frac{3}{2}\sigma_c^2\right)$ is the mean volume (or mass) radius and ρ_w =1000kg.m^{-3} the density of liquid water.

We will now describe the different processes which are playing a role in the evolution equation for N_d.

Nucleation

The nucleation is described by the equations of the heterogeneous nucleation occurring in the atmosphere and summarized in [Pruppacher and Klett, 2000]. It depends on the characteristics of the cloud condensation nuclei (CNN). We have:

$$\left(\frac{\partial N_d}{\partial t}\right)_{Nuc}=\frac{1.}{\Delta t}max\left(\left(N_{CCN}\right)_{max}-N_d,\,0.\right) \quad \text{(II.13.74)}$$

where N_{CCN} is the Cloud Condensation Nuclei density and Δt is the numerical time step.

The number of droplet formed through nucleation is strongly correlated to the super-saturation $s=e/e_{sat}-1$

Supersaturation is characterized by s>0 (e being the partial pressure of water vapour in moist air and e_{sat} the saturation vapour pressure).

A simple expression taking into account all cooling processes for s is (see [Zhang et al., 2014]):

$$\frac{ds}{dt}=\left(\frac{\xi L_v g}{R_a T^2 C_p}-\frac{g}{R_a T}\right)\bar{w}-\left(\frac{R_a T}{\xi e_s}+\frac{\xi L_v^2}{pTC_p}\right)\frac{\mathrm{d}\bar{q_l}}{dt}+\frac{\xi L_v g}{\rho R_a T^2 C_p}\frac{\partial F_{rad}}{\partial z} \quad \text{(II.13.75)}$$

lifting latent heat release radiation

$$=A_1 w-A_2\frac{\mathrm{d}\bar{q_l}}{dt}+A_4\frac{\partial F_{rad}}{\partial z} \quad \text{(II.13.76)}$$

where $\xi=0.622$ (molecular weight of water/molecular weight of air), $g(\mathrm{ms}^{-2})$ is the gravity, R_a is the gas constant for dry air, $\bar{w}(\mathrm{ms}^{-1})$ is the vertical air velocity, e_s(Pa) is the saturation vapor pressure water and F_{rad} the net radiation flux (Wm^{-2}).

Abdul-Razzack nucleation scheme

As the primary source of cloud droplets, the nucleation process depends on many factors including the characteristics (size and chemical composition) of the aerosols. In order to take into account size

distribution and chemical composition if they are known, the Abdul-Razzak (AR) scheme for aerosol activation is used [Abdul-Razzak and Ghan, 2000]. With the superposition of three log-normal aerosol distributions, as proposed by AR, the droplet number concentration at the maximum super-saturation s_{max}, is given by:

$$N_d\left(s_{max}\right) = \frac{1}{2}\sum_{i=1}^{3} N_{ai}\left(1 - erf\left[\frac{2\,ln\left(s_i/s_{max}\right)}{3\sqrt{2}\,ln\left(\sigma_{ai}\right)}\right]\right), \tag{II.13.77}$$

where N_{ai} is the total aerosol number concentration of mode i, s_i the critical super-saturation of a particle with the diameter r_{ai} and the geometric mean diameter of the aerosol mode i.

It can be calculated by using Köhler's theory.

The maximum supersaturation s_{max} is given by:

$$s_{max} = \sum_{i=1}^{3} \frac{1}{s_i^2}\left[f_i\left(\frac{\varsigma}{\eta_i}\right)^{3/2} + g_i\left(\frac{s_i^2}{\eta_i + 3\varsigma}\right)^{3/4}\right], \tag{II.13.78}$$

With $f_i = 0.5\exp\left(2.5\,ln^2\sigma_i\right)$, $g_i = 1 + 0.25\,ln\sigma_i$, $s_i = \frac{2}{\sqrt{B_i}}\left(\frac{A}{3r_a}\right)^{3/2}$
and $\varsigma = \frac{2A}{3}\left(\frac{A_1 w + A_4 \partial F_{rad}/\partial z}{A_3}\right)^{1/2}$, $\eta_i = \frac{\left[\left(A_1 w + A_4 \partial F_{rad}/\partial z\right)/A_3\right]^{3/2}}{2\pi\rho_w A_2 N_{ai}}$
where ρ_w (kg m^{-3}) is the water density, A_1, A_2, A_4 are the constants defined above, and A_3, A, B_i can be found in [Abdul-Razzak and Ghan, 2000].

$$A_3 = \left(\frac{\rho_w R_v T}{e_s D_v^\star} - \frac{L_v \rho_w}{k_a^\star T}\left(1 - \frac{L_v}{T R_v}\right)\right)^{-1},\ A = \frac{2\sigma_{vw}}{R_v T \rho_w}, B_i = \frac{\nu\phi_s\epsilon_m M_w \rho_{ai}}{M_{ai}\rho_w} \tag{II.13.79}$$

where $k_a^\star = 4.187\,10^{-3}\left(5.69 + 0.017\,T\left(in\,^\circ C\right)\right)$ Jm^{-1}s^{-1}K^{-1} is the dry air conductivity and

$D_v^\star = 0.211\,10^{-4}\left(\frac{T}{273.15}\right)^{1.94}\left(\frac{1.01325\,10^5}{P}\right)$ m^{2}s^{-1} the water vapour diffusivity.

M_w is the molecular weight of water, ρ_w water density, σ_{vw} is the partial pressure at air/water interface,
ν the number of ions the salt dissociates into within water,
φ the osmotic coefficient, ρ_{ai} the density of aerosols material,
M_{ai} the molecular weight of the aerosol material.
When not any information is available on the characteristic of the aerosols, a simpler model can be used deduced from typical air mass aerosol concentrations.

Cohard and Pinty nucleation scheme

The Cloud Condensation Nuclei (CCN) density is given by:
$N_{ccn}\left(cm^{-3}\right) = \tilde{C}\tilde{s}^k = Cs^k$ when s>0
where the deviation from saturation s is defined by $s = e/e_{sat} - 1$ and
when s>0 is called the supersaturation (e being the partial pressure of water vapour in moist air and e_{sat} the saturation vapor pressure). Note that the nucleation is not possible for negative value of s.
The parameters C and $\tilde{C}$ are constants of the nucleation process.

Due to different usage in the literature, it is important to be aware of the difference between $\tilde{C}$ and $C = 10^{2k}\tilde{C}$ (C is in cm^{-3}) and between $\tilde{s}$, the super-saturation in percentage, and s ($\tilde{s} = 100$s).

In our simulation, we choose the model and methodology of [Cohard et al., 1998] and [Cohard and Pinty, 2000], that is: $N_{ccn}\left(cm^{-3}\right) = \tilde{C}\tilde{s}^k F\left(\mu, \frac{k}{2}; \frac{k}{2} + 1; -\beta\tilde{s}^2\right)$
F is the hyper-geometric function adjusted to give rather good results from weak (s<0.02%) to significant (s>1%) super-saturation, as it could occur in the atmosphere. μ, β and k are some parameters characterizing the aerosols distribution of the air mass.

Sedimentation and deposition

Cloud water sedimentation is included by assuming a lognormal size distribution of droplets falling in a Stokes regime, in which the changes to N_d and q_l are given by:

$$\left(\frac{\partial N_d}{\partial t}\right)_{sed} = \frac{\partial}{\partial z}\int_0^\infty V_g\left(r\right) n_d\left(r\right) dr = \frac{\partial}{\partial z}\left(N_d V_g\left(r_{mv}\right)\exp\left(-\sigma_d^2\right)\right) \tag{II.13.80}$$

$$\left(\frac{\partial \bar{q}_l}{\partial t}\right)_{sed} = \frac{1}{\rho}\frac{\partial}{\partial z}\int_0^\infty V_g\left(r\right)\frac{4\pi}{3}r^3 n_d\left(r\right) dr = \frac{1}{\rho}\frac{\partial}{\partial z}\left(\rho\bar{q}_l V_g\left(r_{mv}\right)\exp\left(5\sigma_d^2\right)\right) \tag{II.13.81}$$

whereV_g is droplet fall velocity.
In this parametrization V_g(r) is calculated as a function of droplet size radius:

$$V_g\left(r\right) = \rho g C_c r^2\left(4.5\mu_{air}\right)^{-1} \tag{II.13.82}$$

where $C_c = 1. + l_{air}\ (1.257 + 0.4\ exp(1.1\ r/l_{air}))/r$ is the slip correction factor to account for non-continuum effects for small droplets and which depends on the mean free path of air molecules l_{air}, μ_{air} is the viscosity of the air.

Fog deposition process over vegetation has long been recognized as an important factor in the water balance of ecosystems. This process results from the turbulence exchange of fog water and collection from the surface. The deposition flux of fog water, F_{dep}, is predicted from the simple inferential model equation of the type:

$$F_{dep} = \bar{q}_l\frac{1}{R_t} = \bar{q}_l V_{dep} \tag{II.13.83}$$

where V_{dep} is the deposition velocity and R_t is the total resistance against deposition and computed as a combination (parallel and serial arrangements) of aerodynamic (R_{aero}) and surface (R_{surf}) resistances within the first layer (between the ground surface and the first grid level):
$R_{aero} = \frac{1}{2}\frac{\ln\left(\frac{z_1}{z_0}\right)-\Psi_h}{\kappa u_\star}$ and $R_{surf} = \frac{1}{\varepsilon_0 u_\star\left(E_{imp}+E_{int}\right)}$
where z_1 is the height at which the deposition velocity is evaluated, z_0 is the roughness height, ψ_h is the stability function, κ is the von Karman constant, $u_\star$ is the friction velocity, $\varepsilon_0 = 3$ is an empirical constant for all types of land. E_{imp}, E_{int} are collection efficiency from impaction and interception respectively (see [Zhang et al., 2001]):

$E_{imp} = \left(\frac{St}{St+1.5}\right)^2$ with $St = \frac{V_g u_\star}{0.01g}$, $E_{int} = 2\frac{r_{mv}}{0.01}$

The collection by Brownian motion of fog droplet is neglected because its effect is only significant for very small particle diameter (<0.1 μm).

If a deposition process is taken into account in the model, the new bulk velocity that is estimated for liquid water will be calculated as the sum of V_g and V_{dep} at the lowest level and this will be used to estimate F_{dep}.

Condensation-evaporation

The condensation processes do not have any effect on N_d, the nucleation being treated independently, thus we can write:

$$\left(\frac{\partial N_d}{\partial t}\right)_C = 0. \tag{II.13.84}$$

For the evaporation, we distinguish two regimes:

- the total evaporation of all droplets (s<0 and $\bar{q}_l$ =0) is simply described by N_d =0 (see [Cohard and Pinty, 2000]).
- the partial evaporation (s<0 and $\bar{q}_l$>0, i.e. a total evaporation only for the smallest droplets), we choose to follow [Chaumerliac et al., 1987] and to force the droplets to disappear for radius less than a critical value r_{crit}:

$$r_{crit} = \sqrt{-2A_3 s\Delta t} \tag{II.13.85}$$

Finally, the expression for partial evaporation is:

$$\left(\frac{\partial N_d}{\partial t}\right)_{E/C} = \frac{1}{\Delta \mathrm{t}} \int_0^{r_{crit}} \frac{N_d}{r\sigma_c\sqrt{2\pi}} exp\left(-\frac{1}{2\sigma_c^2}\left(ln\frac{r}{r_0}\right)^2\right) dr \tag{II.13.86}$$

For $\overline{q_l}$, condensation and evaporation are not treated explicitly but implicitly through the subgrid scheme (see [Bouzereau et al., 2007]).

13.5 Earth-atmosphere interactions

In code_saturne different options can be used in order to determine surface boundary conditions.

13.5.1 Direct estimation of turbulent fluxes

The first one is to directly impose the fluxes for momentum (surface shear stress), temperature and humidity (eventually any scalar). That consists in giving data for $u^{\star 2} = \sqrt{\overline{u'w'}^2 + \overline{v'w'}^2}$ where u', v', w' are the turbulent fluctuations of the three components of the wind speed,

$Q_0 = \overline{w'\theta'}$ the kinematic flux of sensible heat where θ' is the turbulent fluctuation of the potential temperature,

$E_0 = \overline{w'q'}$ being the kinematic evaporative flux and q' the turbulent fluctuation of the specific humidity.

These data could be obtained, for example, by experimental measurements with sonic anemometer.

13.5.2 Impose temperature and humidity from experimental estimation

In that option fluxes are computed from difference of temperature and humidity between their values at ground level ($z_1 = z_0$ roughness height for the wind, $z_1 = z_{ot}$ thermic roughness height for temperature and humidity) and at the first level z_2 in the air. It needs to determine turbulent fluxes from gradients (see [Musson-Genon et al., 2007]).

Monin-Obukhov's similarity theory gives relations between the fluxes of sensible heat, latent heat and momentum by means of universal functions:

$$\frac{\partial |\underline{u}|}{\partial z} = \left(\frac{u_\star}{\kappa z}\right) \phi_m\left(\frac{z}{L}\right), \tag{II.13.87}$$

$$\frac{\partial \theta}{\partial z} = \left(\frac{\theta_\star}{\kappa z}\right) \phi_h\left(\frac{z}{L}\right) \tag{II.13.88}$$

$$\frac{\partial q}{\partial z} = \left(\frac{q_\star}{\kappa z}\right) \phi_q\left(\frac{z}{L}\right), \tag{II.13.89}$$

where $|\underline{u}|$ is the modulus of the horizontal mean wind speed,
$u_\star$ is the friction velocity with $u_\star^2 = \sqrt{(\overline{u'w'}^2 + \overline{v'w'}^2)}$
u', v', w' are the turbulent fluctuations of the three components of the wind speed,
where θ is the mean potential temperature and q is the mean specific humidity:

$$\theta_\star = \frac{-Q_0}{u_\star} \tag{II.13.90}$$

$$q_\star = -E_0/\mathrm{u}_\star \tag{II.13.91}$$

Here, κ denotes Von Karman's constant, ϕ_m ϕ_h and ϕ_q are the universal functions for wind, temperature and humidity, z is the altitude and L is the Obukhov length:

$$L = \frac{-u_\star^3}{\kappa \frac{g}{T_0} Q_0 + 0.61 T_0 E_0}, \tag{II.13.92}$$

where T_0 is the mean temperature in the surface layer (z_2-z_1 layer).

Usually, the function ϕ_q is chosen equal to ϕ_h. Classically, see for example [Garratt, 1992] or [Cheng and Brutsaert, 2005], integration of the relations (II.13.87), (II.13.88) and (II.13.89) between two different levels $0 < z_1 < z_2$ gives:

$$\delta u = u(z_2) - u(z_1) = \left(\frac{u_\star}{\kappa}\right) \Psi_m \left(L, z_2, z_1\right) \tag{II.13.93}$$

$$\delta \theta = \theta(z_2) - \theta(z_1) = \left(\frac{\theta_\star}{\kappa}\right) \Psi_h \left(L, z_2, z_1\right) \tag{II.13.94}$$

$$\delta q = q(z_2) - q(z_1) \left(\frac{q_\star}{\kappa}\right) \Psi_h \left(L, z_2, z_1\right) \tag{II.13.95}$$

The formulation of the universal functions are given in Table 13.4 (two options available) and the integrated form (expressions of the Ψ functions: the dynamical profiles Ψ_m and the thermal profiles Ψ_h) are given in the following paragraph:

The dynamical profiles Ψ_m

Unstable case: $\zeta_2 < \zeta_1 < 0$: $\Psi_m = \log \frac{z_2}{z_1} - 2\log\left(\frac{1+\Psi_2}{1+\Psi_1}\right) - \log\left(\frac{1+\Psi_2^2}{1+\Psi_1^2}\right) + 2\left(\arctan \Psi_2 - \arctan \Psi_1\right)$,
where $\Psi_2 = (1 - b_m\zeta_2)^{1/4}$ and $\Psi_1 = (1 - b_m\zeta_1)^{1/4}$,
with $\zeta_1 = \mathrm{z}_1/\mathrm{L}$, $\zeta_2 = \mathrm{z}_2/\mathrm{L}$, $\mathrm{b}_m = 15$.

Neutral or stable cases: $0 < \zeta_1 < \zeta_2 < 0.5$: $\Psi_m = log\frac{z_2}{z_1} + b_m\left(\zeta_2 - \zeta_1\right)$,

$0.5 < \zeta_2 < 10$: $\Psi_m = c_{1m} log(2\zeta_2) + \frac{4.25}{\zeta_2} - \frac{0.5}{\zeta_2^2} - log(2\zeta_1) - b_m\zeta_1 - c_{2m}$

$\zeta_2 > 10$: $\Psi_m = d_{1m}\zeta_2 + c_{1m} log\left(2\zeta_2\right) - 11.165 - log(2\zeta_1) - b_m\zeta_1$
with $b_m = 4.7, c_{1m} = 7.85, c_{2m} = 4.15, d_{1m} = 0.7435$

In the case of [Cheng and Brutsaert, 2005], Ψ_m may be expressed as:

$0 < \zeta_1 < \zeta_2$: $\Psi_m = a log\frac{z_2}{z_1} - \mathrm{a}\ log\left(\zeta + \left(1 + \zeta^b\right)^{\frac{1}{b}}\right)$ with a = 6.1 and b = 2.5

The thermal profiles Ψ_h

Unstable case: $\zeta_2 < \zeta_1 < 0$: $\Psi_h = \mathrm{a}_h \left(log\left(\frac{z_2}{z_1}\right) - 2\log\left(\frac{1+\Psi_2}{1+\Psi_1}\right)\right)$,

where $\Psi_2 = 1 - b_h\zeta_2)^{1/2}$ and $\Psi_1 = 1 - b_h\zeta_1)^{1/2}$, with $\mathrm{a}_h = 0.74$, $\mathrm{b}_h = 9$

Stable cases: $\zeta_2 > \zeta_1 > 0$: $\Psi_h = alog\frac{z_2}{z_1} + b(\zeta_2 - \zeta_1)$,

with $\mathrm{a}_h = 0.74$, $b_h = 4.7$ for [Businger et al., 1971]

In the case of [Cheng and Brutsaert, 2005], Ψ_h may be expressed as:

$0 < \zeta_1 < \zeta_2$: $\Psi_h = alog\frac{z_2}{z_1} - c\ log\left(\zeta + \left(1+\zeta^d\right)^{\frac{1}{d}}\right)$ with c = 5.3 and d = 1.1.

Table 13.4: Detailed expressions for the universal functions from [Businger et al., 1971]; [Hicks, 1976], and [Cheng and Brutsaert, 2005]

Function	**unstable** $\zeta < 0$	**stable** $0 < \zeta < 0.5, \quad 1 < \zeta < 10, \quad \zeta > 10$
unstable: [Businger et al., 1971] stable: [Hicks, 1976]	$\phi_m = (1 - 15\zeta)^{-1/4}$ $\phi_h = 0.74\,(1 - 9\zeta)^{-1/2}$	$\phi_m = 1 + 4.7\zeta, \quad \phi_m = 7.85 - \frac{4.25}{\zeta} + \frac{1}{\zeta^2}, \quad \phi_m = 0.7435\,\zeta$ $\phi_h = 0.74 + 4.7\zeta$
[Businger et al., 1971] stable: [Cheng and Brutsaert, 2005]	$\phi_m = (1 - 15\zeta)^{-1/4}$ $\phi_h = 0.74\,(1 - 9\zeta)^{-1/2}$	$\phi_m = 1 + a\frac{\zeta+\zeta^b 1+\zeta^b{}^{\frac{1-b}{b}}}{\zeta+(1+\zeta^b)^{\frac{1}{b}}}$ a=6.1, b=2.5 $\phi_h = 1 + c\left(\frac{\zeta+\zeta^d\left(1\zeta^d\right)^{\frac{1-d}{d}}}{\zeta+(1+\zeta^d))^{\frac{1}{d}}}\right)$ c=5.3, d=1.1

But these expressions do not determine explicitly the atmospheric fluxes as functions of the differences of wind, temperature and humidity between two levels. Two methods for solving this implicit problem are described in the next section.

The exact iterative method

From the definition of L, and with the help of Equations (II.13.93), (II.13.94) and (II.13.95) giving $u_\star$, Q_0 and E_0, the following expression is obtained:

$$L = \frac{\delta u^2 \Psi_h}{\frac{g}{T_0}\Psi_m^2(\delta\theta + 0.61 T_0 \delta q)} \tag{II.13.96}$$

Since Ψ_m, Ψ_h depend on L, this relation has to be solved by an iterative method. We follow [Beljaars and Holtslag, 1991] using an initial value of L that depends on the sign of $\delta\theta + 0.61 T_0 \delta q$.

This iterative procedure converges rapidly in all unstable situations. In stable situations for small values of $u_\star$ and L, the solution for L tends to zero. For this reason, a minimum value is imposed for L. On the other hand, we have verified that the solution obtained is not dependent on the initial value of L.

Having determined L and $u_\star$, the kinematic fluxes of sensible heat Q_0 and latent heat E_0 are then simply given by (II.13.94) and (II.13.95), as well as $q_\star$and $\theta_\star$.

The iterative method is well adapted to local determination of the fluxes because this method derives directly from the experimental determination of the MO universal functions, using assumptions on which the MO theory is based: quasi-stationarity and horizontally homogeneity of the flow, with the Coriolis effect negligible.

The approximate analytical method

To determine the turbulent fluxes in a meteorological model it is preferable to use an analytical approximation in order to avoid an expensive iterative resolution of an equation.

This method is based on a formulation of the universal functions differing from those presented in Table 13.4 and leads to a complete analytical method (see [Louis et al., 1982]).

In this formulation the fluxes are estimated with the bulk Richardson number, defined by:

$$Ri = \frac{g(z_2 - z_1)(\delta\theta + 0.61 T_0 \delta q)}{T_0 \delta u^2} \tag{II.13.97}$$

Using Equation (II.13.96) a relation between Ri, the measurement levels z_1 and z_2, and L is established:

$$\frac{L}{z_2 - z_1} = \frac{\Psi_h}{Ri\,\Psi_m^2}, \tag{II.13.98}$$

Then $u_\star$, Q_0, E_0 according to [Louis et al., 1982], are given by:

$$u_\star = \delta u \left(C_n F_d \left(Ri, \frac{z_2}{z_1} \right) \right)^{1/2}, \tag{II.13.99}$$

$$Q_0 = -\delta\theta \delta u C_n F_h \left(Ri, \frac{z_2}{z_1} \right), \tag{II.13.100}$$

$$E_0 = -\delta q \delta u C_n F_h \left(Ri, \frac{z_2}{z_1} \right), \tag{II.13.101}$$

where C_n is defined as: $C_n = \left(\dfrac{\kappa}{\ln(z_2/z_1)} \right)^2$.

The universal functions F_d and F_h depend on the atmospheric stability through the bulk Richardson number, and are tuned to give results close to those of the exact iterative method, at least in the unstable case.

In the unstable case, the functions F_d and F_h must be asymptotically equivalent to $Ri^{1/2}$ in order to be consistent with the free convection regime. Moreover, continuity must be assured when crossing the neutral point and the slope of the wind profile in the neighbourhood of zero must be slightly larger than the slope of the humidity and temperature. Taking all these considerations into account, one obtains,

$$F_d = 1 - \frac{2bRi}{1 + 3bc_\star \sqrt{|Ri|}}, \tag{II.13.102}$$

$$F_h = 1 - \frac{3bRi}{1 + 2bc_\star \sqrt{|Ri|}}, \tag{II.13.103}$$

with $c_\star = C_n \left(1 - z_1/z_2\right)^{1/2} \left((z_1/z_2)^{1/3} - 1 \right)^{3/2}$ and $b = c = 5$.

For the stable case, the total extinction of the turbulence is prevented by a modification of the asymptotic behaviour of the universal functions in order to avoid thermal disconnection in very stable model layers close to the ground, but also to take into account subgrid or intermittent turbulence effects for Richardson number greater than the theoretical critical value of 0.21. The expressions become:

$$F_d = \frac{1}{1 + \frac{2bRi}{\sqrt{1 + d\,Ri}}}, \tag{II.13.104}$$

$$F_h = \frac{1}{1 + 3b\,Ri\sqrt{1 + d\,Ri}}, \tag{II.13.105}$$

with $b = d = 5$.

The functions do not depend on the ratio z_1/z_2.

13.5.3 Estimation of soil temperature and humidity from an energetic budget

Soil temperature evolution

The soil temperature T_s (assimilated to the temperature at z_{ot}, thermal roughness height) is computed by using an energy balance equation for the earth's surface after [Deardorff, 1978].

$$\frac{dT_s}{dt} = \frac{2\pi^{1/2}}{\rho_s c_s \left(\nu_s \tau_1\right)^{1/2}} H_{at} - \frac{2\pi}{\tau_1}\left(T_s - T_p\right), \tag{II.13.106}$$

with $H_{at} = (1-\alpha_g)\,S^{\downarrow} + \epsilon_g(F^{\downarrow} - \sigma T_s^4) - \rho_0 C_p Q_0 - \rho_0 L E_0$,
where ρ_0 is the air density at z_0, T_p the deep soil temperature,
$S^{\downarrow}$ the downward solar flux, $F^{\downarrow}$ the downward infrared flux,
$\rho_s c_s$ the ground calorific capacity, ν_s the soil thermal conductibility,
$\tau_1 = 86400\,s$, the force restore time constant for T_p,
α_g is the soil albedo, ϵ_g the soil emissivity,
C_p =1005. J $K^{-1}\,kg^{-1}$ is the specific heat at constant pressure of air,
L= 2.5 10^6 J kg^{-1} is the latent heat of vaporization and
σ =5.67 10^{-8} W m^{-2} K^{-4} the Stefan constant.

Soil-humidity evolution

The evolution of the soil liquid water content is deduced from the same principles as the surface temperature:

$$\frac{dw_1}{dt} = \frac{P_r - \rho E_0}{C_1} - \frac{w_1 - w_2}{\tau_1}, \tag{II.13.107}$$

$$\frac{dw_2}{dt} = C_2 \frac{w_1 - w_2}{\tau_1}, \tag{II.13.108}$$

Where P_r is the precipitation flux,
$w_1 = w_s/w_{smax}$ is a non-dimensional liquid water content with w_s the liquid water content for the surface reservoir and w_{smax} its maximum value and
$w_2 = w_p/w_{pmax}$ is a non-dimensional liquid water content with w_p the liquid water content for the deep reservoir (which does not contain the surface reservoir) and w_{pmax} its maximum value.

The constants C_1 an C_2 are given by:
$C_1 = w_{smax}/C_{ws}$ and $C_2 = w_{smax}/w_{pmax}$,
where C_{ws} is a non-dimensional constant tuning the depth of the reservoirs and C_2 the ratio of these depth.

The surface specific humidity q_s is computed as a function of w_1, which varies according to the nature of the surface (bare soil or soil covered by vegetation characterized by veg, the percentage of vegetation in the mesh):

$$q_s = H_u q_{sat}\left(P_s, T_s\right) + veg\,\left(1 - H_u\right) q_a \tag{II.13.109}$$

Where $H_u = 0.5\,[1 - cos\,(\pi w_1)]$,

q_{sat} is the saturated value of q_s, P_s is the surface pressure
and q_a the specific humidity in the first level in the air in code_saturne.
Here H_u will be equal to 1 when a dew flux is present.

Chapter 14

Arbitrary Lagrangian Eulerian

The description of the algorithms available in code_saturne can be found in [Ferrand and Harris, 2021].

Chapter 15

Cavitation modelling

15.1 System equations

The cavitation model is based on an homogeneous mixture model. In this model, the physical properties, the density ρ and the dynamic viscosity μ, of the mixture depends on a resolved void fraction α and constant reference properties ρ_l, μ_l for the liquid phase and ρ_v, μ_v for the gas phase, following the relations:

$$\rho = \alpha\rho_v + (1-\alpha)\rho_l, \tag{II.15.1}$$

$$\mu = \alpha\mu_v + (1-\alpha)\mu_l. \tag{II.15.2}$$

In this model, it is assumed that the mixture dynamic is ruled by the classical incompressible Navier–Stokes equations:

$$\frac{\partial \rho}{\partial t} + \text{div}\ (\rho \underline{u}) = 0, \tag{II.15.3}$$

$$\frac{\partial}{\partial t}(\rho \underline{u}) + \underline{\text{div}}\ (\underline{u} \otimes \rho \underline{u}) = -\underline{\nabla} P + \underline{\text{div}}\, \underline{\underline{\tau}}. \tag{II.15.4}$$

Eq. (II.15.4) corresponds to the momentum equation in Eq. (I.2.11) where the volume source terms have been removed for brevity and $\Gamma \equiv 0$ for the mass source term.

Using (II.15.1), the mass equation can be splitted into:

$$\frac{\partial(\alpha\rho_v)}{\partial t} + \text{div}\,(\alpha\rho_v \underline{u}) = \Gamma_V, \tag{II.15.5}$$

$$\frac{\partial\left((1-\alpha)\rho_l\right)}{\partial t} + \text{div}\ ((1-\alpha)\rho_l \underline{u}) = -\Gamma_V, \tag{II.15.6}$$

with Γ_V representing the vaporization (resp. condensation) source (resp. sink) term, appearing with an opposite sign in the liquid part of the density budget.

Using the fact that the reference densities ρ_v and ρ_l are constant, Eqs. (II.15.5) and (II.15.6) can be easily written in the form:

$$\frac{\partial \alpha}{\partial t} + \text{div}\,(\alpha \underline{u}) = \frac{\Gamma_V}{\rho_v}, \tag{II.15.7}$$

$$\text{div}\,(\underline{u}) = \Gamma_V \left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right). \tag{II.15.8}$$

It is seen that the mass equation of the mixture Eq. (II.15.3) has been splitted into two equations: one simple convection equation which can be used to solve the void fraction and one equation structurally similar to the one solved at the correction step of the predictior-corrector solver of code_saturne (see Appendix N). The global resolution scheme of the cavitation module is thus the following:

1. Prediction of the velocity using Eq. (II.15.4) (see Appendix M).
2. Correction of the velocity using Eq. (II.15.8) (see Appendix N).
3. Resolution of the void fraction using Eq. (II.15.7).
4. Update physical properties of the mixture using Eqs. (II.15.1) and (II.15.2).

15.2 Vaporization source term

In the cavitation module of code_saturne, the Γ_V source term is modeled using the Merkle model:

$$\Gamma_v\,(\alpha, P) = m^+ + m^-,$$

with:

$$m^+ = -\frac{C_{prod}\rho_l \min\,(P - P_V, 0)\,\alpha(1-\alpha)}{0.5\rho_l u_\infty^2 t_\infty}, \qquad m^- = -\frac{C_{dest}\rho_v \max\,(P - P_V, 0)\,\alpha(1-\alpha)}{0.5\rho_l u_\infty^2 t_\infty}, \tag{II.15.9}$$

and $C_{prod} = 10000$, $C_{dest} = 50$ empirical constants, $t_\infty = l_\infty/u_\infty$ a reference time scale and P_V the reference saturation pressure. l_∞, u_∞ and P_V may be provided by the user.

15.3 Time stepping

At each time step n, the global resolution scheme follows successively the steps 1 to 4 described above (§15.1). In this resolution scheme, the physical properties are updated at the end of the time step n.

Prediction step The procedure is almost identical to the classical one described at Appendix M. Only the discretization of the time derivative at the l.h.s of Eq. (II.15.4) is modified in order to take into account that the mixture density is updated at the end of the time step. In the cavitation algorithm, the time derivative at time step $n+1$ is discretized by:

$$\frac{\partial}{\partial t}(\rho \underline{u}) \simeq \frac{\rho^n \widetilde{\underline{u}}^{n+1} - \rho^{n-1}\underline{u}^n}{\Delta t^n},$$

with $\widetilde{\underline{u}}^{n+1}$ the predicted velocity at time step $n+1$.

Correction step With the Merkle model described above (§15.2), the correction step equation of the cavitation model writes:

$$\text{div}\left(\frac{\Delta t^n}{\rho}\underline{\nabla}\delta P\right) = \text{div}\left(\widetilde{u}\right) - \Gamma_V(\alpha, P).$$

In this equation, the pressure in the cavitation source term is taken implicit while the void fraction is explicit:

$$\text{div}\left(\frac{\Delta t^n}{\rho}\underline{\nabla}(\delta P)^{n+1}\right) = \text{div}\left(\widetilde{u}^{n+1}\right) - \Gamma_V(\alpha^n, P^{n+1})\left(\frac{1}{\rho_v} - \frac{1}{\rho_l}\right).$$

Void fraction resolution The time discretization of Eq. (II.15.8) is:

$$\frac{\alpha^{n+1} - \alpha^n}{\Delta t^n} + \text{div}\left(\alpha^{n+\theta}\underline{u}^{n+\theta}\right) = \frac{1}{\rho_v}\Gamma_V(\alpha^n, P^{n+1}). \tag{II.15.10}$$

In this equation, the cavitation source term is discretized with the time scheme as the one used at the correction step in order to ensure the exact mass conservation (it is recalled that the void faction transport equation and correction step are two parts of the mixture mass conservation equation, Eq. (II.15.3).

See the programmers reference of the dedicated subroutine for further details.

Part III

Appendices

Calling tree

Each sub-section of this document is associated with an important subroutine. The full list of the subroutines described here is the following: `bilsc2` `clptur` `clsyvt` `codits` `condli` `cs_solve_equation_scalar` `gradrc` `inimas` `itrmas` `matrix` `cs_solve_navier_stokes` `cs_velocity_prediction` `resopv` `cs_turbulence_ke` `turrij` `viscfa` `visort` `visecv`.

The table 1 presents their sequence within a time step. This calling tree is only partial. In particular, it does not account for the number of calls to each subroutine. Also, for the sake of clarity, no reference has been made to the subroutines dedicated to the gradient calculation (`gradmc`, `gradrc`), which are called very often. For the same reason, the calls to `bilsc2` (advection fluxes) and `matrix` (matrix calculation) which are made from within `codits` (treatment of an advection equation with source terms) have not been reported.

The sub-sections where important information can be found are indicated below:

Calculation of gradients
`gradrc`
Least squares method
`gradmc`
Convective schemes
`bilsc2`
Wall-laws (for velocity and temperature)
`clptur`
`condli`
System solve (incremental method)
`codits`
Calculation of the values at the faces (not exhaustive)
`viscfa`
`visort`

Finally, for the reader wishing to become more familiar with the methods implemented in code_saturne, it is recommended to begin with the study of the advection equation for a scalar (`cs_solve_equation_scalar`) which is solved iteratively using an incremental method (`codits`). It will then be useful to look at `cs_solve_navier_stokes` which briefly presents the solution of the system made up of the mass and momentum equations.

Calculation of the physical properties
Boundary Conditions
`condli`
`clptur` "turbulent" conditions at the wall
`clsyvt` symmetry conditions for the vectors and the tensors
Navier-Stokes solution
`cs_solve_navier_stokes`
Velocity prediction
`cs_velocity_prediction`
`visecv` momentum source terms related to the transposed gradient of the velocity
`viscfa` calculation of the viscosity at the faces
`codits` iterative solution of the system using an incremental method
Pressure correction
`resopv`
`viscfa` calculation of the time step at the faces...
`visort` ...according to the selected options
`matrix` calculation of the Poisson equation matrix
`inimas` initialisation of the mass flow rate
`itrmas` update of the mass flow rate
Velocity correction
standard method
$k-\varepsilon$ **model**
`cs_turbulence_ke`
`viscfa` preliminary steps before...
`bilsc2` ...source terms coupling
`viscfa` calculation of the viscosity at the faces
`codits` iterative solution of the systems using an incremental method
Reynolds stress model
`turrij`
`visort` calculation of the viscosity at the faces
`codits` iterative solution of the systems using an incremental method
Equations for the scalars
`cs_solve_equation_scalar`
`viscfa` calculation of the viscosity at the faces
`codits` iterative solution of the systems using an incremental method

Table 1: Partial and simplified calling tree associated with the successive stages within a time step.

Part IV

Base module

A- bilsc2 routine

Function

In this subroutine, called by `codits` and `cs_turbulence_ke`, the contributions to the explicit budget of the reconstructed (on non-orthogonal meshes and if the user chooses to) convective and diffusive terms of the right-hand side of a convection/diffusion equation for a scalar a are computed. These terms write [1]:

$$\mathcal{B}_\beta((\rho\,\underline{u})^n, a) \quad = \underbrace{-\mathrm{div}\,(\,(\rho\underline{u})^n a)}_{\text{convective part}} + \underbrace{\mathrm{div}\,(\,\beta\,\underline{\nabla} a)}_{\text{diffusive part}} \tag{IV.A.1}$$

with ρ, $\underline{u}$, β and a the variables at time t^n.

See the programmers reference of the dedicated reference for further details.

Discretization

Convective Part

Using the notations adopted in the subroutine `cs_solve_navier_stokes`, the explicit budget corresponding to the integration over a cell Ω_i of the convective part $-\mathrm{div}\,(\,(\rho\,\underline{u})^n a)$ of $\mathcal{B}_\beta$ can be written as a sum of the numerical fluxes F_{ij} calculated at the faces of the internal cells, and the numerical fluxes $F_{b_{ik}}$ calculated at the boundary faces of the computational domain Ω. Let's take $Neigh\,(i)$ the set of the centres of the neighbouring cells of Ω_i and $\gamma_b(i)$ the set of the centres of the boundary faces of Ω_i (if they exist). Thus we can write

$$\int_{\Omega_i} \mathrm{div}\,((\rho\underline{u})^n a)\,d\Omega = \sum_{j\in Neigh(i)} F_{ij}((\rho\underline{u})^n, a) + \sum_{k\in\gamma_b(i)} F_{b_{ik}}((\rho\underline{u})^n, a)$$

with :

$$F_{ij}((\rho\underline{u})^n, a) = \left[(\rho\underline{u})^n_{ij}.\,\underline{S}_{ij}\right]\ a_{f,ij} \tag{IV.A.2}$$

$$F_{b_{ik}}((\rho\underline{u})^n, a) = \left[(\rho\underline{u})^n_{b_{ik}}.\,\underline{S}_{b_{ik}}\right]\ a_{f\,b_{ik}} \tag{IV.A.3}$$

where $a_{f,ij}$ and $a_{f\,b_{ik}}$ represent the values of a at the internal and boundary faces of Ω_i, respectively.

Before presenting the different convection schemes available in code_saturne, we define:

$$\alpha_{ij} = \frac{\overline{FJ'}}{\overline{I'J'}} \text{ defined at the internal faces only and}$$

$$\underline{u}_{K'} = \underline{u}_K + (\underline{\underline{\mathrm{grad}}}\,\underline{u})_K.\,\underline{KK'} \text{ at the first order in space, for } K = I \text{ or } J$$

The value of the convective flux F_{ij} depends on the numerical scheme. Three different types of convection schemes are available in this subroutine:

[1] They appear on the right-hand side of the incremental system for cell I of the momentum prediction step: $\mathcal{EM}(\delta\underline{u}^{k+1}, I) = \mathcal{E}(\underline{u}^{n+1/2,k}, I)$ (see `cs_solve_navier_stokes` for more details)

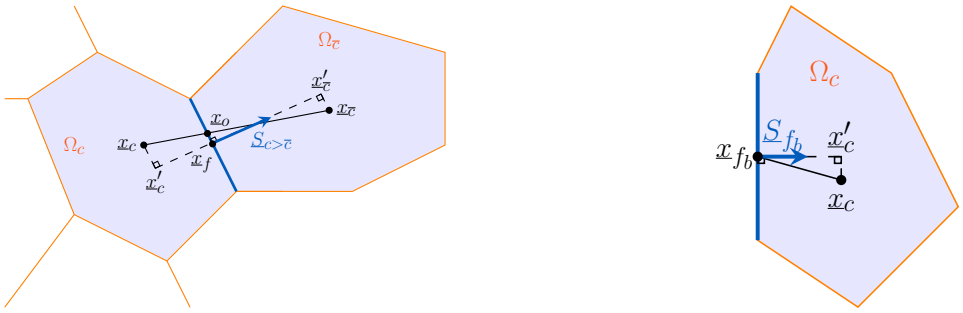


Figure IV.A.1: Definition of the geometric entities for internal (left) and a boundary faces (right).

- a 1^{st} order upwind scheme:

$$F_{ij}((\rho\underline{u})^n, a) = F_{ij}^{upstream}((\rho\underline{u})^n, a)$$

où : $$a_{f,ij} = \begin{cases} a_I \text{ si } (\rho\underline{u})^n_{ij} \cdot \underline{S}_{ij} \geqslant 0 \\ a_J \text{ si } (\rho\underline{u})^n_{ij} \cdot \underline{S}_{ij} < 0 \end{cases}$$

- a centered scheme:

$$F_{ij}((\rho\underline{u})^n, a) = F_{ij}^{centered}((\rho\underline{u})^n, a)$$

with : $a_{f,ij} = \alpha_{ij} a_{I'} + (1-\alpha_{ij}) a_{J'}$

- a Second Order Linear Upwind scheme (SOLU):

$$F_{ij}((\rho\underline{u})^n, a) = F_{ij}^{SOLU}((\rho\underline{u})^n, a)$$

with : $$a_{f,ij} = \begin{cases} a_I + \underline{IF} \cdot (\underline{\nabla} a)_I \text{ si } (\rho\underline{u})^n_{ij} \cdot \underline{S}_{ij} \geqslant 0 \\ a_J + \underline{JF} \cdot (\underline{\nabla} a)_J \text{ si } (\rho\underline{u})^n_{ij} \cdot \underline{S}_{ij} < 0 \end{cases}$$

The value of $F_{b_{ik}}$ is calculated as :

$$a_{f\,b_{ik}} = \begin{cases} a_I & \text{if } (\rho\underline{u})^n_{b_{ik}} \cdot \underline{S}_{b_{ik}} \geqslant 0 \\ a_{b_{ik}} & \text{if } (\rho\underline{u})^n_{b_{ik}} \cdot \underline{S}_{b_{ik}} < 0 \end{cases}$$

$a_{b_{ik}}$ is the boundary value directly computed from the prescribed boundary conditions.

Remark 1

When a centered scheme is used, we actually write (to ensure first order discretization in space for a)

$$a_{f,ij} = \alpha_{ij} a_I + (1-\alpha_{ij}) a_J + \frac{1}{2}\left[(\underline{\nabla} a)_I + (\underline{\nabla} a)_J\right] \cdot \underline{OF}$$

A factor $\frac{1}{2}$ is used for numerical stability reasons.

Remark 2

A slope test (which may introduce non-linearities in the convection operator) allows to switch from

[1] Extrapolation of the upwind value at the faces centre.

the centered or SOLU scheme to the first order upwind scheme (without blending). Additionally, in standard mode $a_{f,ij}$ is computed as a weighted average between the upstream value and the centered value (blending), according to users' choice (variable `BLENCV` in the subroutine `usini1`).

Diffusive Part

Similarly, the diffusive part writes :

$$\int_{\Omega_i} \mathrm{div}\,(\,\beta\,\underline{\nabla} a)\;d\Omega = \sum_{j \in Neigh(i)} D_{ij}(\,\beta, a) + \sum_{k \in \gamma_b(i)} D_{b_{ik}}(\beta, a)$$

with:

$$D_{ij}(\,\beta, a) = \beta_{ij} \frac{a_{J'} - a_{I'}}{\overline{I'J'}} S_{ij} \tag{IV.A.4}$$

and :

$$D_{b_{ik}}(\,\beta, a) = \beta_{b_{ik}} \frac{a_{b_{ik}} - a_{I'}}{\overline{I'F}} S_{b_{ik}} \tag{IV.A.5}$$

using the same notations as before, and with S_{ij} and $S_{b_{ik}}$ being the norms of vectors $\underline{S}_{ij}$, and $\underline{S}_{b_{ik}}$ respectively.

Implementation

In the following, the reader is reminded of the role of the variables used in the different tests: • `IRCFLP`, from array `IRCFLU` ; indicates for the considered variables wether or not the convective and diffusive fluxes are reconstructed

= 0 : no reconstruction

= 1 : reconstruction

• `ICONVP`, from array `ICONV` ; indicates if the considered variables is convected or not.

= 0 : no convection

= 1 : convection

• `IDIFFP`, from array `IDIFF` ; indicates if the diffusion of the considered variables is taken into account or not.

= 0 : no diffusion

= 1 : diffusion

• `IUPWIN` indicates locally, in `bilsc2` (to avoid unnecessary calculations) whether a pure upwind scheme is chosen or not for the considered variables to be convected.

= 0 : no pure upwind

= 1 : pure upwind is used

• `ISCHCP`, from array `ISCHCV` ; indicates which type of second order convection scheme is used on orthogonal meshes for the considered variable to convect (only useful if `BLENCP` > 0).

= 0 : we use the SOLU scheme (Second Order Linear Upwind)

= 1 : we use a centered scheme

In both cases the blending coefficient `BLENCP` needs to be given in `usini1`.

• `BLENCP`, from array `BLENCV` ; indicates the percentage of centered or SOLU convection scheme that one wants to use. This weighting coefficient is between 0 and 1.

• `ISSTPP`, from array `ISSTPC` ; indicates if one wants to remove the slope test that switches the convection scheme from second order to upwind if the test is positive.

= 0 : a slope test is systematically used

= 1 : no slope test

Computation of the gradient $\underline{G}_{c,i}$ of variable a

The computation of the gradient of variable a is necessary for the computation of the explicit budget. `grdcel` is called everytime this gradient is needed, and it is stored in the array (`DPDX`, `DPDY`, `DPDZ`). The computation of the gradient is necessary in the following situations:

- if the convection is activated with a non pure upwind scheme (`ICONVP` $\neq 0$ and `IUPWIN` $= 0$) **and**,
  if we want to reconstruct the fluxes (`IRCFLP` $= 1$),
  or if we want to use the SOLU scheme (`ISCHCP` $= 0$),
  or if we use the slope test (`ISSTPP` $= 0$),

or :

- if there is diffusion and we want to reconstruct the fluxes (`IDIFFP` $\neq 0$ and `IRCFLP` $= 1$).

In all other cases, the array (`DPDX`, `DPDY`, `DPDZ`) is set to zero.

Computation of the upwind gradient $\underline{G}_{c,i}^{amont}$ of variable a

$\underline{G}_{c,i}^{amont}$ refers to the upwind gradient of variable a, for cell Ω_i. It is stored in the array (`DPDXA`, `DPDYA`, `DPDZA`). We also define the scalars a_{ij}^{amont} and $a_{b_{ik}}^{amont}$ as:

$$|\Omega_i|\,\underline{G}_{c,i}^{upwind} \overset{def}{=} \sum_{j\in Neigh(i)} a_{ij}^{upwind}\,\underline{S}_{ij} + \sum_{k\in\gamma_b(i)} a_{b_{ik}}^{upwind}\,\underline{S}_{b_{ik}} \qquad \text{(IV.A.6)}$$

After initializing it to zero, $\underline{G}_{c,i}^{amont}$ **is only computed** when the user wishes to compute **a convection term with a centered or SOLU method, and a slope test**.

- For each cell Ω_i, the face values a_{IF} (variable `PIF`) and a_{JF} (variable `PJF`), are computed as:

$$\begin{aligned} a_{IF} &= a_I + \underline{IF}\,.\,(\underline{\nabla} a)_I \\ a_{JF} &= a_J + \underline{JF}\,.\,(\underline{\nabla} a)_J \end{aligned}$$

Depending on the sign s_{ij}^n of the mass flux $(\rho\underline{u})_{ij}^n$. $\underline{S}_{ij}$, we give a_{IF} or a_{JF} the value a_{ij}^{upwind} of the expression $\sum_{j\in Neigh(i)} a_{ij}^{upwind}\,\underline{S}_{ij}$.

$$a_{ij}^{upwind} = \begin{cases} a_I + \underline{IF}\,.\,(\underline{\nabla} a)_I & \text{si } s_{ij}^n = 1 \\ a_J + \underline{JF}\,.\,(\underline{\nabla} a)_J & \text{si } s_{ij}^n = -1 \end{cases}$$

• The boundary terms are computed in a classic manner as follows (keeping the same notations as in the other subroutines):

$$\begin{aligned} \sum_{k\in\gamma_b(i)} a_{b_{ik}}^{upwind}\,\underline{S}_{b_{ik}} &= \sum_{k\in\gamma_b(i)} (\mathtt{INC}\,A_{b,ik} + B_{b,ik}\,a_{I'})\,\underline{S}_{b_{ik}} \\ &= \sum_{k\in\gamma_b(i)} \left[\mathtt{INC}\,A_{b,ik} + B_{b,ik}\,a_I + B_{b,ik}\,\underline{II'}\,.\,\underline{G}_{c,i}\right]\,\underline{S}_{b_{ik}} \end{aligned}$$

$(A_{b,ik}, B_{b,ik})_{k\in\gamma_b(i)}$ are stored in the arrays (`COEFAP`, `COEFBP`). The vector $\underline{II'}$ is stored in the array (`DIIPBX`, `DIIPBY`, `DIIPBZ`). The surfaces $(\underline{S}_{b_{ik}})_{k\in\gamma_b(i)}$ are stored in the array `SURFBO` .

Summation of the numerical convective and diffusive fluxes

The contributions to the explicit budget $[-\mathrm{div}\,((\rho\underline{u})^n a) + \mathrm{div}\,(\beta\,\underline{\nabla} a)]$ are computed and added to the right-hand side array `SMBR`, which has already been initialized before the call to `BILSC2` (with the explicit source terms for instance, etc.).
The variable `FLUX` gathers the convective and diffusive parts of the numerical fluxes. It is computed in a classic manner, first on the internal faces, and then on the boundary faces. The indices i and j

are represented by II and JJ, respectively.
In order to take into account (when necessary) the sign s_{ij}^n of the mass flux $(\rho \underline{u})_{ij}^n . \underline{S}_{ij}$, the following equations are used :

For any real b, we have :

$$\left\{ \begin{array}{ll} b & = b^+ + b^- \text{ with } b^+ = max\ (b,0), \quad b^- = min\ (b,0) \\ |\,b| & = b^+ - b^- \\ b^+ & = \dfrac{1}{2}\,[\,b + |\,b|\,] \\ b^- & = \dfrac{1}{2}\,[\,b - |\,b\,|\,] \end{array} \right.$$

In this subroutine, b represents the mass flux FLUMAS(IFAC) on an internal face IFAC (FLUMAB(IFAC) for a boundary face IFAC) ; b^+ is stored in FLUI and b^- in FLUJ.

- for an internal face ij (IFAC)

We calculate :

$$\sum_{j \in Neigh(i)} F_{ij}((\rho \underline{u})^n, a) - \sum_{j \in Neigh(i)} D_{ij}(\beta, a) = \sum_{j \in Neigh(i)} \left(\left[(\rho \underline{u})_{ij}^n . \underline{S}_{ij} \right] \ a_{f,ij} - \beta_{ij} \frac{a_{J'} - a_{I'}}{\overline{I'J'}} S_{ij} \right)$$

The above sum corresponds to the numerical operation:

$$\begin{array}{rl} \texttt{FLUX} & = \texttt{ICONVP} \,.\, [\ \texttt{FLUI} \,.\, \texttt{PIF} + \texttt{FLUJ} \,.\, \texttt{PJF}\ ] \\ & + \texttt{IDIFFP} \,.\, \texttt{VISCF(IFAC)} \,.\, [\ \texttt{PIP} - \texttt{PJP}\ ] \end{array} \tag{IV.A.7}$$

The above equation does not depend on the chosen convective scheme, since the latter only affects the quantities PIF (face value of a used when b is positive) and PJF (face value of a used when b is négative). PIP represents $a_{I'}$, PJP $a_{J'}$ and VISCF(IFAC) $\beta_{ij} \dfrac{S_{ij}}{\overline{I'J'}}$.
The treatment of diffusive part is identical (either with or without reconstruction). Consequently, only the numerical scheme relative to the convection differs.

- for a boundary face ik (IFAC)

We compute the terms :

$$\sum_{k \in \gamma_b(i)} F_{b_{ik}}((\rho \underline{u})^n, a) - \sum_{k \in \gamma_b(i)} D_{b_{ik}}(\beta, a) = \sum_{k \in \gamma_b(i)} \left(\left[(\rho \underline{u})_{b_{ik}}^n . \underline{S}_{b_{ik}} \right] \ a_{f\,b_{ik}} - \beta_{b_{ik}} \frac{a_{b_{ik}} - a_{I'}}{\overline{I'F}} S_{b_{ik}} \right)$$

with:

$$\begin{array}{ll} a_{I'} & = a_I + \underline{II'} . \underline{G}_{c,i} \\ a_{b1_{ik}} & = \texttt{INC}\, A_{b,ik} + B_{b,ik}\, a_{I'} \\ a_{b_{ik}} & = \texttt{INC}\, A_{b,ik}^{diff} + B_{b,ik}^{diff}\, a_{I'} \end{array}$$

The coefficients $(A_{b,ik}, B_{b,ik})_{k \in \gamma_b(i)}$ $\left(\text{ resp.}(A_{b,ik}^{diff}, B_{b,ik}^{diff})_{k \in \gamma_b(i)} \right)$ represent the boundary conditions associated with a (resp. the diffusive fluxes [2] of a).
The above sum corresponds to the numerical operation:

$$\begin{array}{rl} \texttt{FLUX} & = \texttt{ICONVP} \,.\, [\ \texttt{FLUI} \,.\, \texttt{PVAR(II)} + \texttt{FLUJ} \,.\, \texttt{PFAC}\ ] \\ & + \texttt{IDIFFP} \,.\, \texttt{VISCB(IFAC)} \,.\, [\ \texttt{PIP} - \texttt{PFACD}\ ] \end{array} \tag{IV.A.8}$$

where PFAC represents $a_{b1_{ik}}$, PIP $a_{I'}$, PFACD $a_{b_{ik}}$ and VISCB(IFAC) $\beta_{b_{ik}} \dfrac{S_{b_{ik}}}{\overline{I'F}}$.
This treatment is common to all schemes, because boundary values only depend on boundary conditions, and because a very simplified expression of $F_{b_{ik}}$ is used (upwind) [3].

We still have to compute, when the convection option is activated (ICONVP = 1), the values of variables PIF and PJF, for any internal face IFAC between cell Ω_i and Ω_j.

[2] see clptur for more details. The difference is actually only effective when the $k-\epsilon$ model is used, and for the velocity only.

[3] Actually, $a_{f\,b_{ik}}$ is a_I if $(\rho \underline{u})_{b_{ik}}^n . \underline{S}_{b_{ik}} \geqslant 0$, $a_{b1_{ik}}$ otherwise.

Calculation of the flux in pure upwind IUPWIN = 1

In this case, there is no reconstruction since only the values PVAR(II) and PVAR(JJ) at the cell centres are needed.

$$\begin{aligned} \texttt{PIF} &= \texttt{PVAR(II)} \\ \texttt{PJF} &= \texttt{PVAR(JJ)} \end{aligned} \tag{IV.A.9}$$

The variable INFAC counts the number of calculations in pure upwind, in order to be printed in the listing file. In order to obtain the global numerical flux FLUX (convective + diffusive) associated, the following operations are performed :
• calculation of vectors II' and JJ',
• calculation of the face gradient (DPXF, DPYF, DPZF) with the half-sum of the cell gradients $\underline{G}_{c,i}$ et $\underline{G}_{c,j}$,
• calculation of the reconstructed (if necessary) values $a_{I'}$ and $a_{J'}$ (variables PIP and PJP, respectively) given by :

$$a_{K'} = a_K + \texttt{IRCFLP} \,.\, \underline{KK'} \,.\, \frac{1}{2} \left(\underline{G}_{c,i} + \underline{G}_{c,j} \right) \quad \text{K = I et J} \tag{IV.A.10}$$

• calculation of the quantities FLUI and FLUJ,
• calculation of the flux FLUX using (IV.A.7).
The computation of the sum in SMBR is straight-forward, following (IV.A.1) [4] .

Calculation of the flux with a centered or SOLU scheme (IUPWIN = 0)

The two available second order schemes on orthogonal meshes are the centered scheme and the SOLU scheme.
In both cases, the following operations are performed:
• calculation of the vector II', the array (DIIPFX, DIIPFY, DIIPFZ) and the vector JJ', the array (DJJPFX, DJJPFY, DJJPFZ)
• calculation of the face gradient (DPXF, DPYF, DPZF) haff-sum of the cell gradients $\underline{G}_{c,i}$ and $\underline{G}_{c,j}$,
• calculation of the possibly reconstructed (if IRCFLP = 1) values $a_{I'}$ and $a_{J'}$ (variables PIP and PJP, respectively) given by :

$$a_{K'} = a_K + \texttt{IRCFLP} \,.\, \underline{KK'} \,.\, \frac{1}{2} \left(\underline{G}_{c,i} + \underline{G}_{c,j} \right) \quad \text{K = I and J} \tag{IV.A.11}$$

• calculation of FLUI and FLUJ.

■ without slope test (ISSTPP = 1)

⋆ with a centered scheme (ISCHCP = 1)
The values of the variables PIF and PJF are equal, and calculated using the weighting coefficient α_{ij} as follows:

$$\begin{aligned} P_{IF} &= \alpha_{ij} \,.\, P_{I'} + (1 - \alpha_{ij}) \,.\, P_{J'} \\ P_{JF} &= P_{IF} \end{aligned} \tag{IV.A.12}$$

⋆ with a SOLU scheme (ISCHCP = 0)

After calculating the vectors $\underline{IF}$ and $\underline{JF}$, the values of the variables PIF and PJF are computed as follows:

$$\begin{aligned} P_{IF} &= P_I + \underline{IF} \,.\, \underline{G}_{c,i} \\ P_{JF} &= P_J + \underline{JF} \,.\, \underline{G}_{c,j} \end{aligned} \tag{IV.A.13}$$

PIF and PJF are systematically reconstructed in order to avoid using pure upwind, *i.e.* this formulae is applied even when the user chooses not to reconstruct (IRCFLP = 0).

■ with slope test (ISSTPP = 0)

[4] taking into account the negative sign of $\mathcal{B}_\beta$.

The procedure is quite similar to the one described in the previous paragraph. There is, in addition to the previous procedure, a slope test that makes under certain conditions the scheme switch locally (but systematically) from the chosen centered or SOLU scheme to a pure upwind scheme.

$\rightsquigarrow$ calculation of the slope test

Equation (IV.A.6) writes on an internal cell Ω_i, with $s_{ij}^n = sgn\left[(\rho\underline{u})_{ij}^n \,.\, \underline{S}_{ij}\right]$:

$$\begin{aligned}
|\Omega_i|\,\underline{G}_{c,i}^{upwind} &= \sum_{j\in Neigh(i)} a_{ij}^{upwind}\,\underline{S}_{ij} \\
&= \sum_{j\in Neigh(i)} \left[\frac{1}{2}(\ s_{ij}^n + 1\ ) \quad a_{IF} + \frac{1}{2}(\ s_{ij}^n - 1\ )\, a_{JF}\right]\ \underline{S}_{ij} \\
&= \sum_{j\in Neigh(i)} \left[\frac{1}{2}(\ s_{ij}^n + 1\ ) \quad (\ a_I + \underline{IF}\,.\,(\underline{\nabla} a)_{\,I}\ )\right. \\
&\qquad\qquad \left. +\frac{1}{2}\ \ (\ s_{ij}^n - 1\ )(\ a_I + \underline{JF}\,.\,(\underline{\nabla} a)_{\,J}\ )\right]\ \underline{S}_{ij}
\end{aligned}$$

On a cell Ω_i with neighbours $(\Omega_j)_{j\in Neigh(i)}$, the classic slope test consists in locating where a variable a is non-monotonic by studying the sign of the scalar product of the cell gradients of $\underline{G}_{c,i}$ and $\underline{G}_{c,j}$. If this product is negative, we switch to an upwind scheme, if it is positive, we use a centered or SOLU scheme.

Another technique which also ensures the monotonicity of the solution is to apply this criterion to the upwind gradients $\underline{G}_{c,k}^{amont}$ or to their normal projection on face $(\,\underline{G}_{c,k}^{amont}\,.\,\underline{S}_{kl}\,)$.

We then study the sign of the product $\underline{G}_{c,i}^{amont}\,.\,\underline{G}_{c,j}^{amont}$ or of the product $(\,\underline{G}_{c,i}^{amont}\,.\,\underline{S}_{ij}\,)\,.\,(\,\underline{G}_{c,j}^{amont}\,.\,\underline{S}_{ij}\,)$.

The slope test implemented is based on the first quantity, $\underline{G}_{c,i}^{amont}\,.\,\underline{G}_{c,j}^{amont}$ (the second one was abandonned because it was found to be less general). The choice of a slope test based on $\underline{G}_{c,i}^{amont}\,.\,\underline{G}_{c,j}^{amont}$ comes from the following line of argument in one-dimension [5]:

Let's take p a second order in x polynomial function. Its value at points $I-1$, I, $I+1$ of coordinates x_{I-1}, x_I and x_{I+1} are p_{I-1}, p_I, and p_{I+1}, respectively. To simplify, we suppose that I is the origin O ($x_I = 0$), and that the grid spacing h is constant, which results in $x_{I+1} = -x_{I-1} = h$. Additionally, we suppose that the velocity is orientated from point I towards point $I+1$, *i.e.* $s_{ij}^n = 1$. Therefore we consider the points $I-1$, I and $I+1$ for the face ij which is located between I and $I+1$.

The sign of the product $p'(x_{I-1})\,.\,p'(x_{I+1})$ indicates the monotonicity of function p. If this product is positive, the function is monotonic and we use a centered or a SOLU scheme, otherwise, we switch to an upwind scheme. By identifying the polynomial coefficients using the equations $p\,(x_{I-1}) = p_{I-1}$, $p\,(x_I) = p_I$, $p\,(x_{I+1}) = p_{I+1}$, we obtain :

$$\begin{aligned}
p'(x_{I-1}) &= +\frac{p_{I+1} - p_{I-1}}{2h} \quad + \left[\frac{p_I - p_{I-1}}{h} - \frac{p_{I+1} - p_I}{h}\right] \\
p'(x_{I+1}) &= +\frac{p_{I+1} - p_{I-1}}{2h} \quad - \left[\frac{p_I - p_{I-1}}{h} - \frac{p_{I+1} - p_I}{h}\right]
\end{aligned} \tag{IV.A.14}$$

or after simplification :

$$\begin{aligned}
p'(x_{I-1}) &= G_{\,c,i} + \left(\ G_{c,i}^{amont} - \frac{p_{I+1} - p_I}{h}\right) \\
p'(x_{I+1}) &= G_{\,c,i} - \left(\ G_{c,i}^{amont} - \frac{p_{I+1} - p_I}{h}\right)
\end{aligned} \tag{IV.A.15}$$

We know that :

♣ $\dfrac{p_{I+1} - p_I}{h}$ representes the upwind derivative at point $I+1$, directly accessible by the values of p in the neighbouring cells of face ij,

♣ $\dfrac{p_{I+1} - p_{I-1}}{2h}$ represents the centered derivative (in finite volume) at point I, namely $G_{\,c,i}$,

[5] Information on the second derivative would permit to study more finely the behaviour and the strong variations of a.

♣ $\dfrac{p_I - p_{I-1}}{h}$ represents the value of the upwind derivative (in finite volume) at point I, namely $G_{c,i}^{amont}$.
The slope test relative to $p'(x_{I-1}) \,.\, p'(x_{I+1})$ reduces to studying the sign of $\mathcal{TP}_{1d}$:

$$\begin{aligned} \mathcal{TP}_{1d} &= \left(G_{c,i} \; + \; [\, G_{c,i}^{amont} - \frac{p_{I+1} - p_I}{h}] \right) . \left(G_{c,i} \; - \; [\, G_{c,i}^{amont} - \frac{p_{I+1} - p_I}{h}] \right) \\ &= |G_{c,i}|^2 - (\, G_{c,i}^{amont} - \frac{p_{I+1} - p_I}{h})^2 \end{aligned} \tag{IV.A.16}$$

Using a similar line of argument, a possible extension to higher dimensions consists in replacing the values $G_{c,k}$ and $G_{c,k}^{amont}$ by $(\underline{G}_{c,k} \,.\, \underline{S}_{kl}\,)$ eand $(\underline{G}_{c,k}^{amont} \,.\, \underline{S}_{kl}\,)$ respectively. After simplifications, this leads us to the formulae $\mathcal{TP}_{3d}^{+}$:

$$\mathcal{TP}_{3d}^{+} = (\underline{G}_{c,i} \,.\, \underline{S}_{ij})^2 - (\underline{G}_{c,i}^{amont} \,.\, \underline{S}_{ij} - \frac{a_{\,J} - a_{\,I}}{\overline{I'J'}} S_{ij})^2 \tag{IV.A.17}$$

for $(\rho \underline{u})_{ij}^n .\, \underline{S}_{ij} > 0$.
Similarly, we can deduce a $\mathcal{TP}_{3d}^{-}$ associated with $(\rho \underline{u})_{ij}^n .\, \underline{S}_{ij} < 0$, defined by :

$$\mathcal{TP}_{3d}^{-} = (\underline{G}_{c,j} \,.\, \underline{S}_{ij})^2 - (\underline{G}_{c,j}^{amont} \,.\, \underline{S}_{ij} - \frac{a_{\,J} - a_{\,I}}{\overline{I'J'}} S_{ij})^2 \tag{IV.A.18}$$

We introduce the variables `TESTI`, `TESTJ` and `TESTIJ` computed as:

$$\begin{aligned} \texttt{TESTI} &= \underline{G}_{c,i}^{amont} \,.\, \underline{S}_{ij} \\ \texttt{TESTJ} &= \underline{G}_{c,j}^{amont} \,.\, \underline{S}_{ij} \\ \texttt{TESTIJ} &= \underline{G}_{c,i}^{amont} \,.\, \underline{G}_{c,j}^{amont} \end{aligned} \tag{IV.A.19}$$

The quantity `TESQCK` corresponding to $\mathcal{TP}_{3d}$, is computed dynamically, depending on the sign of the mass flux s_{ij}^n.

$\rightsquigarrow$ consequently :

$\diamond$ if $(\rho \underline{u})_{ij}^n .\, \underline{S}_{ij} > 0$ and
if $\underbrace{(\underline{G}_{c,i} \,.\, \underline{S}_{ij})^2 - (\underline{G}_{c,i}^{amont} \,.\, \underline{S}_{ij} - \frac{a_{\,J} - a_{\,I}}{\overline{I'J'}} S_{ij})^2}_{\texttt{TESQCK}} < 0$ or $(\underline{G}_{c,i}^{amont} \,.\, \underline{G}_{c,j}^{amont}) < 0$,
or :
if $(\rho \underline{u})_{ij}^n .\, \underline{S}_{ij} < 0$ and
if $\underbrace{(\underline{G}_{c,j} \,.\, \underline{S}_{ij})^2 - (\underline{G}_{c,j}^{amont} \,.\, \underline{S}_{ij} - \frac{a_{\,J} - a_{\,I}}{\overline{I'J'}} S_{ij})^2}_{\texttt{TESQCK}} < 0$ or $(\underline{G}_{c,i}^{amont} \,.\, \underline{G}_{c,j}^{amont}) < 0$,
then we switch to a pure upwind scheme:

$$\begin{aligned} \texttt{PIF} &= \texttt{PVAR(II)} \\ \texttt{PJF} &= \texttt{PVAR(JJ)} \end{aligned} \tag{IV.A.20}$$

and `INFAC` is incremented.

$\diamond$ otherwise :
the centered or the SOLU scheme values values are used as before :

$\star$ with a centered scheme (`ISCHCP` $= 1$)
The values of the variables `PIF` and `PJF` are equal and calculated using the weighting coefficient α_{ij} :

$$\begin{aligned} P_{IF} &= \alpha_{ij} \,.\, P_{I'} + (1 - \alpha_{ij}) \,.\, P_{J'} \\ P_{JF} &= P_{IF} \end{aligned} \tag{IV.A.21}$$

⋆ with a SOLU scheme (ISCHCP = 0)

After calculating the vectors $\underline{IF}$ and $\underline{JF}$, the values of the variables PIF and PJF are computed as follows:

$$\begin{aligned} P_{IF} &= P_I + \underline{IF}\,.\,\underline{G}_{c,i} \\ P_{JF} &= P_J + \underline{JF}\,.\,\underline{G}_{c,j} \end{aligned} \tag{IV.A.22}$$

PIF and PJF are systematically reconstructed in order to avoid using pure upwind, *i.e.* this formulae is applied even when the user chooses not to reconstruct (IRCFLP = 0).

Wether the slope test is activated or not, when the centered or the SOLU schemes are activated, a blending coefficient (BLENCP) between 0 and 1, provided by the user, enables to blend, if desired, the chosen scheme and the pure upwind scheme following the formulae:

$$\begin{aligned} P_{IF} &= \texttt{BLENCP} P_{IF}^{(centre\ ou\ SOLU)} + (1 - \texttt{BLENCP})\ P_{II} \\ P_{JF} &= \texttt{BLENCP} P_{JF}^{(centre\ ou\ SOLU)} + (1 - \texttt{BLENCP})\ P_{JJ} \end{aligned} \tag{IV.A.23}$$

- calculation of FLUI and FLUJ,
- calculation of the flux FLUX using equation (IV.A.7).

The computation of the sum in SMBR is straight-forward, following (IV.A.1)[6]

Remark

For more information on the convection schemes and the slope test in code_saturne (version 1.1), the reader is referred to EDF internal report EDF HI-83/04/020 (F. Archambeau, 2004).

[6] taking into account the negative sign of $\mathcal{B}_\beta$.

Points to treat

• **Convection scheme**

⇝ Upwind scheme

As all first-order schemes, it is robust, but introduces severe numerical diffusion.

⇝ Centered or SOLU scheme

This type of schemes can generate numerical oscillations, that can cause the calculation to blow up. It can also lead to physical scalars taking unphysical values.

Considering these limitations, other schemes are currently being tested and implemented in order to improve the quality of the schemes available to the users.

• **Diffusion scheme**

The formulae :

$$D_{ij}(\beta, a) = \beta_{ij} \frac{a_{J'} - a_{I'}}{\overline{I'J'}} S_{ij} \tag{IV.A.24}$$

is second-order accurate only for $\alpha_{ij} = \frac{1}{2}$. A possible correction may be to write :

$$\underline{G}_{f,ij} \cdot \underline{S}_{ij} = (\nabla a)_{ij} = \frac{a_{J'} - a_{I'}}{\overline{I'J'}} \cdot \underline{S}_{ij} + (\frac{1}{2} - \alpha_{ij}) [(\underline{\nabla} a)_{I'} - (\underline{\nabla} a)_{J'}] \cdot \underline{S}_{ij} \tag{IV.A.25}$$

with a gradient limiter and a computation of β_{ij} which does not alter the order of accuracy.

• **Implementation**

In order to improve the CPU time, an effort on loops can be done. More particularly, there is a test `IF` inside of a loop on variable `IFAC` that needs to be checked.

• **Calculation of the gradient used during the reconstruction of the diffusive fluxes**

Why do we use $\frac{1}{2}(\underline{G}_{c,i} + \underline{G}_{c,j})$ instead of $\underline{G}_{c,k}$, for $k = i$ or for $k = j$ in the reconstructed values $a_{I'}$ or $a_{J'}$ of (IV.A.10) and (IV.A.11)?

B- cs_turbulence_ke routine

Function

The purpose of this function is to solve the system of équations of k and ε in a semi-coupléd manner. The system of equations solved is the following:

$$\left\{\begin{array}{l} \rho\dfrac{\partial k}{\partial t} + \text{div}\left[\rho\underline{u}\,k - (\mu + \dfrac{\mu_t}{\sigma_k})\underline{\nabla} k\right] = \mathcal{P} + \mathcal{G} - \rho\varepsilon + k\text{div}\,(\rho\underline{u}) + \Gamma(k_i - k) \\ \qquad\qquad +\alpha_k k + \beta_k \\ \rho\dfrac{\partial \varepsilon}{\partial t} + \text{div}\left[\rho\underline{u}\,\varepsilon - (\mu + \dfrac{\mu_t}{\sigma_\varepsilon})\underline{\nabla}\varepsilon\right] = C_{\varepsilon_1}\dfrac{\varepsilon}{k}\left[\mathcal{P} + (1 - C_{\varepsilon_3})\mathcal{G}\right] - \rho C_{\varepsilon_2}\dfrac{\varepsilon^2}{k} + \varepsilon\text{div}\,(\rho\underline{u}) \\ \qquad\qquad +\Gamma(\varepsilon_i - \varepsilon) + \alpha_\varepsilon\varepsilon + \beta_\varepsilon \end{array}\right. \tag{IV.B.1}$$

$\mathcal{P}$ is the term of production by mean shear stress:

$$\begin{aligned} \mathcal{P} &= -\rho R_{ij}\frac{\partial u_i}{\partial x_j} = -\left[-\mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \frac{2}{3}\mu_t\frac{\partial u_k}{\partial x_k}\delta_{ij} + \frac{2}{3}\rho k\delta_{ij}\right]\frac{\partial u_i}{\partial x_j} \\ &= \mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} - \frac{2}{3}\mu_t(\text{div}\,\underline{u})^2 - \frac{2}{3}\rho k\text{div}\,(\underline{u}) \\ &= \mu_t\left[2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + 2\left(\frac{\partial w}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)^2\right] \\ &\qquad -\frac{2}{3}\mu_t(\text{div}\,\underline{u})^2 - \frac{2}{3}\rho k\text{div}\,(\underline{u}) \end{aligned}$$

$\mathcal{G}$ is the gravity production term: $\mathcal{G} = -\dfrac{1}{\rho}\dfrac{\mu_t}{\sigma_t}\dfrac{\partial \rho}{\partial x_i}g_i$

The turbulent viscosity is $\mu_t = \rho C_\mu\dfrac{k^2}{\varepsilon}$.

The constants are:
$C_\mu = 0,09$; $C_{\varepsilon_2} = 1,92$; $\sigma_k = 1$; $\sigma_\varepsilon = 1,3$
$C_{\varepsilon_3} = 0$ si $\mathcal{G} \geqslant 0$ (stratification unstable) et $C_{\varepsilon_3} = 1$ si $\mathcal{G} \leqslant 0$ (stratification stable).

Γ is a possible mass source term (such that the mass conservation equation becomes $\dfrac{\partial \rho}{\partial t} + \text{div}\,(\rho\underline{u}) = \Gamma$). φ_i ($\varphi = k$ or ε) is the value of φ associated with the injected or extracted mass. In the case where we remove mass ($\Gamma < 0$), we necessarily have $\varphi_i = \varphi$. Similarly, when we inject mass, we also often specifies $\varphi_i = \varphi$.y In these two cases, the term disappears from the equation. In the following sections, we will qualify as *forced injection* the cases where we have $\Gamma > 0$ and $\varphi_i \neq \varphi$.

α_k, β_k, α_ε, β_ε are possible user source terms, leading to partial implication, imposed if necessary by the function `cs_user_source_terms`.

Discretization

The resolution is done in three steps, in order to partially couple the two variables k and ε. For simplicty, let us rewrite the system as follows:

$$\left\{\begin{array}{l} \rho\dfrac{\partial k}{\partial t} = D(k) + S_k(k,\varepsilon) + k\mathrm{div}\,(\rho\underline{u}) + \Gamma(k_i - k) + \alpha_k k + \beta_k \\ \rho\dfrac{\partial \varepsilon}{\partial t} = D(\varepsilon) + S_\varepsilon(k,\varepsilon) + \varepsilon\mathrm{div}\,(\rho\underline{u}) + \Gamma(\varepsilon_i - \varepsilon) + \alpha_\varepsilon \varepsilon + \beta_\varepsilon \end{array}\right. \tag{IV.B.2}$$

D is the convection/diffusion operator. S_k (resp. S_ε) is the source term of k (resp. ε).

First step: explicit balance

We solve the explicit balance:

$$\left\{\begin{array}{l} \rho^{(n)}\dfrac{k_e - k^{(n)}}{\Delta t} = D(k^{(n)}) + S_k(k^{(n)},\varepsilon^{(n)}) + k^{(n)}\mathrm{div}\,(\rho\underline{u}) + \Gamma(k_i - k^{(n)}) + \alpha_k k^{(n)} + \beta_k \\ \rho^{(n)}\dfrac{\varepsilon_e - \varepsilon^{(n)}}{\Delta t} = D(\varepsilon^{(n)}) + S_\varepsilon(k^{(n)},\varepsilon^{(n)}) + \varepsilon^{(n)}\mathrm{div}\,(\rho\underline{u}) + \Gamma(\varepsilon_i - \varepsilon^{(n)}) + \alpha_\varepsilon \varepsilon^{(n)} + \beta_\varepsilon \end{array}\right. \tag{IV.B.3}$$

(the term in Γ is only taken into account in the case of forced injection)

Second step: coupling of source terms

The source terms are implicit in a coupled way:

$$\left\{\begin{array}{l} \rho^{(n)}\dfrac{k_{ts} - k^{(n)}}{\Delta t} = D(k^{(n)}) + S_k(k_{ts},\varepsilon_{ts}) + k^{(n)}\mathrm{div}\,(\rho\underline{u}) + \Gamma(k_i - k^{(n)}) + \alpha_k k^{(n)} + \beta_k \\ \rho^{(n)}\dfrac{\varepsilon_{ts} - \varepsilon^{(n)}}{\Delta t} = D(\varepsilon^{(n)}) + S_\varepsilon(k_{ts},\varepsilon_{ts}) + \varepsilon^{(n)}\mathrm{div}\,(\rho\underline{u}) + \Gamma(\varepsilon_i - \varepsilon^{(n)}) + \alpha_\varepsilon \varepsilon^{(n)} + \beta_\varepsilon \end{array}\right. \tag{IV.B.4}$$

so:

$$\left\{\begin{array}{l} \rho^{(n)}\dfrac{k_{ts} - k^{(n)}}{\Delta t} = \rho^{(n)}\dfrac{k_e - k^{(n)}}{\Delta t} + S_k(k_{ts},\varepsilon_{ts}) - S_k(k^{(n)},\varepsilon^{(n)}) \\ \rho^{(n)}\dfrac{\varepsilon_{ts} - \varepsilon^{(n)}}{\Delta t} = \rho^{(n)}\dfrac{\varepsilon_e - \varepsilon^{(n)}}{\Delta t} + S_\varepsilon(k_{ts},\varepsilon_{ts}) - S_\varepsilon(k^{(n)},\varepsilon^{(n)}) \end{array}\right. \tag{IV.B.5}$$

The term in $\mathrm{div}\,(\rho\underline{u})$ is not implicit because it is linked to the D terms to ensure that the implication matrix will be diagonally dominant. The term in Γ and the user source terms are not implicit either more, but they will be in the third step.

And we write (for $\varphi = k$ or ε)

$$S_\varphi(k_{ts},\varepsilon_{ts}) - S_\varphi(k^{(n)},\varepsilon^{(n)}) = (k_{ts} - k^{(n)})\left.\frac{\partial S_\varphi}{\partial k}\right|_{k^{(n)},\varepsilon^{(n)}} + (\varepsilon_{ts} - \varepsilon^{(n)})\left.\frac{\partial S_\varphi}{\partial \varepsilon}\right|_{k^{(n)},\varepsilon^{(n)}} \tag{IV.B.6}$$

So we finally solve the system 2×2:

$$\begin{pmatrix} \dfrac{\rho^{(n)}}{\Delta t} - \left.\dfrac{\partial S_k}{\partial k}\right|_{k^{(n)},\varepsilon^{(n)}} & -\left.\dfrac{\partial S_k}{\partial \varepsilon}\right|_{k^{(n)},\varepsilon^{(n)}} \\ -\left.\dfrac{\partial S_\varepsilon}{\partial k}\right|_{k^{(n)},\varepsilon^{(n)}} & \dfrac{\rho^{(n)}}{\Delta t} - \left.\dfrac{\partial S_\varepsilon}{\partial \varepsilon}\right|_{k^{(n)},\varepsilon^{(n)}} \end{pmatrix} \begin{pmatrix} (k_{ts} - k^{(n)}) \\ (\varepsilon_{ts} - \varepsilon^{(n)}) \end{pmatrix} = \begin{pmatrix} \rho^{(n)}\dfrac{k_e - k^{(n)}}{\Delta t} \\ \rho^{(n)}\dfrac{\varepsilon_e - \varepsilon^{(n)}}{\Delta t} \end{pmatrix} \tag{IV.B.7}$$

Third step: implicit convection/diffusion

We solve the system:

$$\left\{\begin{array}{l} \rho^{(n)}\dfrac{k^{(n+1)}-k^{(n)}}{\Delta t} = D(k^{(n+1)}) + S_k(k_{ts},\varepsilon_{ts}) + k^{(n+1)}\mathrm{div}\,(\rho\underline{u}) + \Gamma(k_i - k^{(n+1)}) \\ \qquad\qquad +\alpha_k k^{(n+1)} + \beta_k \\ \rho^{(n)}\dfrac{\varepsilon^{(n+1)}-\varepsilon^{(n)}}{\Delta t} = D(\varepsilon^{(n+1)}) + S_\varepsilon(k_{ts},\varepsilon_{ts}) + \varepsilon^{(n+1)}\mathrm{div}\,(\rho\underline{u}) + \Gamma(\varepsilon_i - \varepsilon^{(n+1)}) \\ \qquad\qquad +\alpha_\varepsilon \varepsilon^{(n+1)} + \beta_\varepsilon \end{array}\right. \tag{IV.B.8}$$

soit

$$\left\{\begin{array}{l} \rho^{(n)}\dfrac{k^{(n+1)}-k^{(n)}}{\Delta t} = D(k^{(n+1)}) - D(k^{(n)}) + \rho^{(n)}\dfrac{k_{ts}-k^{(n)}}{\Delta t} + (k^{(n+1)}-k^{(n)})\mathrm{div}\,(\rho\underline{u}) \\ \qquad\qquad -\Gamma(k^{(n+1)}-k^{(n)}) + \alpha_k(k^{(n+1)}-k^{(n)}) \\ \rho^{(n)}\dfrac{\varepsilon^{(n+1)}-\varepsilon^{(n)}}{\Delta t} = D(\varepsilon^{(n+1)}) - D(\varepsilon^{(n)}) + \rho^{(n)}\dfrac{\varepsilon_{ts}-\varepsilon^{(n)}}{\Delta t} + (\varepsilon^{(n+1)}-\varepsilon^{(n)})\mathrm{div}\,(\rho\underline{u}) \\ \qquad\qquad -\Gamma(\varepsilon^{(n+1)}-\varepsilon^{(n)}) + \alpha_\varepsilon(\varepsilon^{(n+1)}-\varepsilon^{(n)}) \end{array}\right. \tag{IV.B.9}$$

The term in Γ is not yet taken into account except in the case of forced injection. The term in α is only taken into account if α is negative, to avoid weakening the diagonal of the matrix that we are going to invert.

Coupling details

During the coupling step, in order to favor stability and realizability of the result, not all terms are taken into account account. More precisely, we can write:

$$\left\{\begin{array}{l} S_k = \rho C_\mu \dfrac{k^2}{\varepsilon}\left(\tilde{\mathcal{P}}+\tilde{\mathcal{G}}\right) - \dfrac{2}{3}\rho k\mathrm{div}\,(\underline{u}) - \rho\varepsilon \\ S_\varepsilon = \rho C_{\varepsilon_1} C_\mu k\left(\tilde{\mathcal{P}} + (1-C_{\varepsilon_3})\tilde{\mathcal{G}}\right) - \dfrac{2}{3}C_{\varepsilon_1}\rho\varepsilon\mathrm{div}\,(\underline{u}) - \rho C_{\varepsilon_2}\dfrac{\varepsilon^2}{k} \end{array}\right. \tag{IV.B.10}$$

Noting $\tilde{\mathcal{P}} = \left(\dfrac{\partial u_i}{\partial x_j} + \dfrac{\partial u_j}{\partial x_i}\right)\dfrac{\partial u_i}{\partial x_j} - \dfrac{2}{3}(\mathrm{div}\,\underline{u})^2$

and $\tilde{\mathcal{G}} = -\dfrac{1}{\rho\sigma_t}\dfrac{\partial \rho}{\partial x_i}g_i$

We therefore have in theory

$$\left\{\begin{array}{l} \dfrac{\partial S_k}{\partial k} = 2\rho C_\mu \dfrac{k}{\varepsilon}\left(\tilde{\mathcal{P}}+\tilde{\mathcal{G}}\right) - \dfrac{2}{3}\rho\mathrm{div}\,(\underline{u}) \\ \dfrac{\partial S_k}{\partial \varepsilon} = -\rho \\ \dfrac{\partial S_\varepsilon}{\partial k} = \rho C_{\varepsilon_1} C_\mu\left(\tilde{\mathcal{P}} + (1-C_{\varepsilon_3})\tilde{\mathcal{G}}\right) + \rho C_{\varepsilon_2}\dfrac{\varepsilon^2}{k^2} \\ \dfrac{\partial S_\varepsilon}{\partial \varepsilon} = -\dfrac{2}{3}C_{\varepsilon_1}\rho\mathrm{div}\,(\underline{u}) - 2\rho C_{\varepsilon_2}\dfrac{\varepsilon}{k} \end{array}\right. \tag{IV.B.11}$$

In practice, we will try to ensure $k_{ts} > 0$ and $\varepsilon_{ts} > 0$. In itself based on a simplified calculation, as well as on feedback of ESTET, we show that it is preferable not to take into account certain terms. In the end, we realize the following coupling:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} (k_{ts}-k^{(n)}) \\ (\varepsilon_{ts}-\varepsilon^{(n)}) \end{pmatrix} = \begin{pmatrix} \dfrac{k_e-k^{(n)}}{\Delta t} \\ \dfrac{\varepsilon_e-\varepsilon^{(n)}}{\Delta t} \end{pmatrix} \tag{IV.B.12}$$

with

$$\begin{cases} A_{11} = \dfrac{1}{\Delta t} - 2C_\mu \dfrac{k^{(n)}}{\varepsilon^{(n)}} \mathrm{Min}\left[\left(\tilde{\mathcal{P}} + \tilde{\mathcal{G}}\right), 0\right] + \dfrac{2}{3}\mathrm{Max}\left[\mathrm{div}\left(\underline{u}\right), 0\right] \\ A_{12} = 1 \\ A_{21} = -C_{\varepsilon_1} C_\mu \left(\tilde{\mathcal{P}} + (1 - C_{\varepsilon_3})\tilde{\mathcal{G}}\right) - C_{\varepsilon_2}\left(\dfrac{\varepsilon^{(n)}}{k^{(n)}}\right)^2 \\ A_{22} = \dfrac{1}{\Delta t} + \dfrac{2}{3} C_{\varepsilon_1} \mathrm{Max}\left[\mathrm{div}\left(\underline{u}\right), 0\right] + 2C_{\varepsilon_2} \dfrac{\varepsilon^{(n)}}{k^{(n)}} \end{cases} \tag{IV.B.13}$$

(by definition of C_{ε_3}, $\tilde{\mathcal{P}} + (1 - C_{\varepsilon_3})\tilde{\mathcal{G}}$ is always positive)

Implementation

• Calculation of the production term
We call `cs_field_gradient_vector` to calculate the gradients of velocity. In the end, we have
$\texttt{tinstk} = 2\left(\dfrac{\partial u}{\partial x}\right)^2 + 2\left(\dfrac{\partial v}{\partial y}\right)^2 + 2\left(\dfrac{\partial w}{\partial z}\right)^2 + \left(\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right)^2 + \left(\dfrac{\partial u}{\partial z} + \dfrac{\partial w}{\partial x}\right)^2 + \left(\dfrac{\partial v}{\partial z} + \dfrac{\partial w}{\partial y}\right)^2$
et
$\texttt{divu} = \dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} + \dfrac{\partial w}{\partial z}$

(the term $div(\underline{u})$ is not calculated by `cs_divergence`, for correspond exactly to the trace of the strain tensor which is calculated for production)

• Reading user source terms
Call `cs_user_source_terms` to load user source terms. They are stored in the following tables:
$\texttt{W7} = \Omega\beta_k$
$\texttt{W8} = \Omega\beta_\varepsilon$
$\texttt{usimpk} = \Omega\alpha_k$
$\texttt{usimpe} = \Omega\alpha_\varepsilon$

Then we add the term in $(div\underline{u})^2$ à to `TINSTK`. So we have
$\texttt{TINSTK} = \tilde{\mathcal{P}}$

• Calculation of the gravity term
Calculation only if $\texttt{igrake} = 1$.
We call `cs_gradient_scalar` for `rom`, with boundary conditions $\texttt{coefa} = \texttt{romb}$ and $\texttt{coefb} = \texttt{viscb} = 0$.
$\texttt{prdtur} = \sigma_t$ is set to 1 if there is no temperature scalar.

$\tilde{\mathcal{G}}$ is calculated and the source terms are updated:
$\texttt{tinstk} = \tilde{\mathcal{P}} + \tilde{\mathcal{G}}$
$\texttt{tinste} = \tilde{\mathcal{P}} + \mathrm{Max}\left[\tilde{\mathcal{G}}, 0\right] = \tilde{\mathcal{P}} + (1 - C_{\varepsilon_3})\tilde{\mathcal{G}}$

If $\texttt{igrake} = 0$, we simply have
$\texttt{tinstk} = \texttt{tinste} = \tilde{\mathcal{P}}$

• Calculation of the mass accumulation term
We store $\texttt{W1} = \Omega\mathrm{div}\left(\rho\underline{u}\right)$ calculated by `divmas` (to correspond to the convection terms of the matrix).

• Calculation of explicit source terms
We assign the explicit source terms of k and ε for the first step.
$\texttt{smbrk} = \Omega\left(\mu_t(\tilde{\mathcal{P}} + \tilde{\mathcal{G}}) - \dfrac{2}{3}\rho^{(n)}k^{(n)}\mathrm{div}\,\underline{u} - \rho^{(n)}\varepsilon^{(n)}\right) + \Omega k^{(n)}\mathrm{div}\left(\rho\underline{u}\right)$
$\texttt{smbre} = \Omega\dfrac{\varepsilon^{(n)}}{k^{(n)}}\left(C_{\varepsilon_1}\left(\mu_t(\tilde{\mathcal{P}} + (1 - C_{\varepsilon_3})\tilde{\mathcal{G}}) - \dfrac{2}{3}\rho^{(n)}k^{(n)}\mathrm{div}\,\underline{u}\right) - C_{\varepsilon_2}\rho^{(n)}\varepsilon^{(n)}\right) + \Omega\varepsilon^{(n)}\mathrm{div}\left(\rho\underline{u}\right)$

then $\texttt{smbrk} = \Omega S_k^{(n)} + \Omega k^{(n)} \mathrm{div}\,(\rho \underline{u})$ and $\texttt{smbre} = \Omega S_\varepsilon^{(n)} + \Omega \varepsilon^{(n)} \mathrm{div}\,(\rho \underline{u})$.

• Calculation of user source terms
We add the explicit user source terms to `smbrk` and `smber`:
$\texttt{smbrk} = \Omega S_k^{(n)} + \Omega k^{(n)} \mathrm{div}\,(\rho \underline{u}) + \Omega \alpha_k k^{(n)} + \Omega \beta_k$
$\texttt{smbre} = \Omega S_\varepsilon^{(n)} + \Omega \varepsilon^{(n)} \mathrm{div}\,(\rho \underline{u}) + \Omega \alpha_\varepsilon \varepsilon^{(n)} + \Omega \beta_\varepsilon$

Arrays `w7` and `w8` are freed, `usimpk` and `usimpe` are retained for use in the third step of resolution.

• Computation of explicit convection/diffusion terms
`cs_balance_scalar` est called twice, for k et for ε, so as to add to `smbrk` and `smbre` the explicit convection/diffusion terms explicites $D(k^{(n)})$ et $D(\varepsilon^{(n)})$. These termes are first stored in `w7` and `w8`, to be reused in the third resolution stage.

• Mass source terms
In the case of a forced injection of matter, we call `cs_mass_source_terms` twice to add the terms in $\Omega\Gamma(k_i - k^{(n)})$ and $\Omega\Gamma(\varepsilon_i - \varepsilon^{(n)})$ to `smbrk` and `smber`. The implicit part ($\Omega\Gamma$) is stored in the variables `w2` and `w3`, which will be used during the third step (the two variables are indeed necessary, in case we have a forced injection on k and not on ε, for example).

• End of the first step
This completes the first step. We have
$$\texttt{smbrk} = \Omega \rho^{(n)} \frac{k_e - k^{(n)}}{\Delta t}$$
$$\texttt{smbre} = \Omega \rho^{(n)} \frac{\varepsilon_e - \varepsilon^{(n)}}{\Delta t}$$

• Coupling step
(only if $\texttt{ikecou} = 1$)

We renormalize `smbrk` and `smbre` which become the second members of the coupling system.
$$\texttt{smbrk} = \frac{1}{\Omega \rho^{(n)}} \texttt{smbrk} = \frac{k_e - k^{(n)}}{\Delta t}$$
$$\texttt{smbre} = \frac{1}{\Omega \rho^{(n)}} \texttt{smbre} = \frac{\varepsilon_e - \varepsilon^{(n)}}{\Delta t}$$
and $\texttt{divp23} = \frac{2}{3} \mathrm{Max}\,[\mathrm{div}\,(\underline{u}), 0]$.

We set the coupling matrix
$$\texttt{A11} = \frac{1}{\Delta t} - 2 C_\mu \frac{k^{(n)}}{\varepsilon^{(n)}} \mathrm{Min}\left[\left(\tilde{\mathcal{P}} + \tilde{\mathcal{G}}\right), 0\right] + \frac{2}{3} \mathrm{Max}\,[\mathrm{div}\,(\underline{u}), 0]$$
$$\texttt{A12} = 1$$
$$\texttt{A21} = -C_{\varepsilon_1} C_\mu \left(\tilde{\mathcal{P}} + (1 - C_{\varepsilon_3}) \tilde{\mathcal{G}}\right) - C_{\varepsilon_2} \left(\frac{\varepsilon^{(n)}}{k^{(n)}}\right)^2$$
$$\texttt{A22} = \frac{1}{\Delta t} + \frac{2}{3} C_{\varepsilon_1} \mathrm{Max}\,[\mathrm{div}\,(\underline{u}), 0] + 2 C_{\varepsilon_2} \frac{\varepsilon^{(n)}}{k^{(n)}}$$
We invert the system 2×2, and we get:
$\texttt{deltk} = k_{ts} - k^{(n)}$
$\texttt{delte} = \varepsilon_{ts} - \varepsilon^{(n)}$

• End of the second step
We update the variables `smbrk` and `smbre`.
$$\texttt{smbrk} = \Omega \rho^{(n)} \frac{k_{ts} - k^{(n)}}{\Delta t}$$

$\texttt{smbre} = \Omega\rho^{(n)}\dfrac{\varepsilon_{ts} - \varepsilon^{(n)}}{\Delta t}$ If we do not couple ($\texttt{ikecou} = 0$), these two variables keep the same value as at the end of the first step.

• Calculation of implicit terms

We remove from `SMBRK` and `SMBRE` the part in convection diffusion at time n, which was stored in `W7` and `W8`.

$$\texttt{SMBRK} = \Omega\rho^{(n)}\frac{k_{ts} - k^{(n)}}{\Delta t} - \Omega D(k^{(n)})$$

$$\texttt{SMBRE} = \Omega\rho^{(n)}\frac{\varepsilon_{ts} - \varepsilon^{(n)}}{\Delta t} - \Omega D(\varepsilon^{(n)})$$

We calculate the implicit terms, excluding convection/diffusion, which correspond diagonal matrix.

$$\texttt{tinstk} = \frac{\Omega\rho^{(n)}}{\Delta t} - \Omega\text{div}\,(\rho\underline{u}) + \Omega\Gamma + \Omega\text{Max}[-\alpha_k, 0]$$

$$\texttt{tinste} = \frac{\Omega\rho^{(n)}}{\Delta t} - \Omega\text{div}\,(\rho\underline{u}) + \Omega\Gamma + \Omega\text{Max}[-\alpha_\varepsilon, 0]$$

(Γ is only taken into account in forced injection, i.e. it is necessarily positive and does not risk weakening the diagonal of the matrix).

• Final Resolution

We then pass twice in the function `cs_equation_iterative_solve_scalar`, for k and ε, to solve equations of the type:

$\texttt{tinst} \times (\varphi^{(n+1)} - \varphi^{(n)}) = D(\varphi^{(n+1)}) + \texttt{smbr}$.

• Final clipping

We finally pass in the function `cs_turbulence_ke_clip` to clip clip $k^{(n+1)}$ et $\varepsilon^{(n+1)}$ if necessary.

C- clptur routine

Function

This subroutine is dedicated to the calculation of the wall boundary conditions. The notations introduced in CONDLI for the general boundary conditions will be used.

The wall boundary conditions refer to all the boundary conditions for the velocity, the turbulent variables (k, ε, R_{ij}), the temperature when it has a prescribed value at the wall (or the enthalpy and more generally the *VarScalaires*[1] to treat at the wall by using a similarity law for the associated boundary layer). For the *VarScalaire* in particular, when the boundary conditions at the wall are of Neumann type (homogeneous or not), they are treated in condli and don't present them here. In particular, the boundary conditions of the *VarScalaires* are not treated here because their treatment at the wall is of homogeneous Neumann type.

We present the calculation of the pair of coefficients A_b and B_b which are used during the computation of certain discretized terms of the equations to solve, and which allow in particular to determine a value associated with the boundary faces $f_{b,int}$ (at a point located at the "centre" of the boundary face, the barycentre of its vertices) using the formulae $f_{b,int} = A_b + B_b\, f_{I'}$ ($f_{I'}$ is the value of the variable at point I', the projection of the centre of the boundary cell onto the line normal to the boundary face and passing through its centre : see figure IV.C.1).

See the programmers reference of the dedicated subroutine for further details.

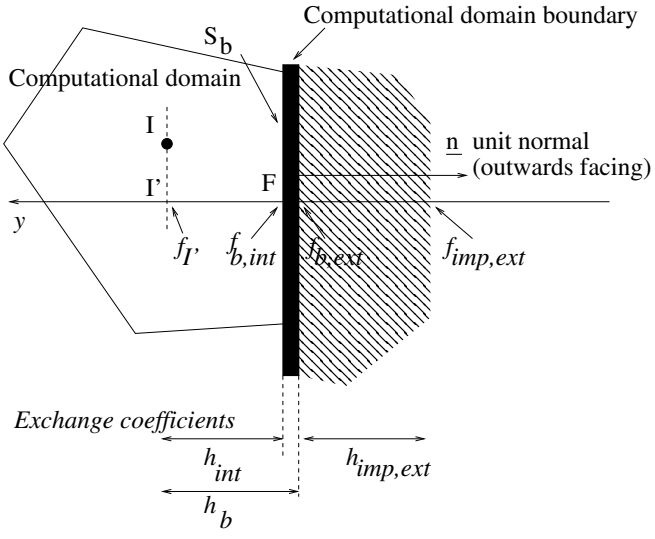


Figure IV.C.1: Boundary cell.

Discretisation

• **Notations**

[1] As in condli the *VarScalaire* are any solution of a convection-diffusion equation apart from the velocity, the pressure and the turbulent variables k, ε and R_{ij}. More specifically, the name *VarScalaire* can refer to the temperature, the enthalpy or a passive scalar.

The velocity of the wall is noted $\underline{v}_p$. We assume it is projected onto the plane tangent to the wall (if it is not, then the code projects it).

The velocity of the fluid is noted $\underline{u}$. Index I, I' or F denotes the point at which the velocity is estimated. The component tangent to the wall writes u_τ. The fluid velocity in the coordinate system attached to the wall ("relative" velocity) writes $\underline{u}^r = \underline{u} - \underline{v}_p$.

The orthonormal coordinate system attached to the wall writes $\hat{\mathcal{R}} = (\underline{\tau}, \underline{\tilde{n}}, \underline{b})$.

- $\underline{\tilde{n}} = -\underline{n}$ is the unit vector orthogonal to the wall and directed towards the interior of the computational domain.
- $\underline{\tau} = \dfrac{1}{\|\underline{u}^r_{I'} - (\underline{u}^r_{I'} \,.\, \underline{\tilde{n}})\|} \left[\underline{u}^r_{I'} - (\underline{u}^r_{I'} \,.\, \underline{\tilde{n}})\right]$ is the unit vector parallel to the projection of the relative velocity at I', $\underline{u}^r_{I'}$, in the plane tangent to the wall (*i.e.* orthogonal to $\underline{\tilde{n}}$) : see figure IV.C.1.
- $\underline{b}$ is the unit vector which completes the positively oriented coordinate system.

The dimensionless limit distance which separates the viscous sublayer from the logarithmic region writes y^+_{lim}. Its value is $1/\kappa$ (with $\kappa = 0,42$) in general (to ensure the continuity of the velocity gradient) and 10.88 in LES (to ensure the continuity of the velocity).

In the case of the **two velocity scale model**,

- u_k is the friction velocity at the wall obtained from the turbulent kinetic energy. We write u^* the friction velocity at the wall calculated from the equation $\dfrac{u^r_{\tau,I'}}{u^*} = f(y^+_k)$.
- y^+_k represents a dimensionless wall distance, $y^+_k = \dfrac{u_k\, I'F}{\nu}$ (ν is the molecular kinematic viscosity taken at the centre I of the boundary cell). The function f gives the ideal shape of the velocity profile. It is piecewisely approximated by the logarithmic law $f(z) = f_1(z) = \dfrac{1}{\kappa} ln(z) + 5,2$ for $z > y^+_{lim}$ and by the linear law $f(z) = f_2(z) = z$ otherwise.
- The two velocity scale u_k and u^* are simple to compute but their computation requires the knowledge of the turbulent kinetic energy k_I at the centre of cell adjoint to the boundary face (with the $R_{ij} - \varepsilon$ model, we use half the trace of the Reynolds stress tensor).
- The two velocity scale model is the default model in code_saturne. It often permits, and in particular in cases with heat transfer, to reduce the effects of certain flaws associated to the $k-\varepsilon$ model.

Later on, we will use u^* and u_k for the boundary conditions of the velocity and scalars (in particular the temperature).

Two velocity scale model

$$\left\{ \begin{array}{l} u_k = C_\mu^{\frac{1}{4}} k_I^{\frac{1}{2}} \\ u^* \text{is solution of } \left\{ \begin{array}{ll} \dfrac{u^r_{\tau,I'}}{u^*} = \dfrac{1}{\kappa} ln(y^+_k) + 5,2 & \text{for } y^+_k > y^+_{lim} \\ \dfrac{u^r_{\tau,I'}}{u^*} = y^+_k & \text{for } y^+_k \leqslant y^+_{lim} \end{array} \right. \\ \qquad \text{with } C_\mu = 0,09 \qquad y^+_k = \dfrac{u_k\, I'F}{\nu} \text{ and } \kappa = 0,42 \end{array} \right. \tag{IV.C.1}$$

In the case of the **one velocity scale model**,

we write u^* the only friction velocity at the wall solution of the equation $\dfrac{u^r_{\tau,I'}}{u^*} = f(y^+)$. y^+ represents a dimensionless wall distance $y^+ = \dfrac{u^*\, I'F}{\nu}$ (ν is the molecular kinematic viscosity taken at the centre I

of the boundary cell). The function f gives the ideal shape of the velocity profile, as in the case of the two velocity scale model. One can note that this friction velocity, calculated using a more complicated approach (Newton method), can however be obtained without making any reference to the turbulent variables (k, ε, R_{ij}). For convenience in the case of the one velocity scale model, we write $u_k = u^*$.

Later on, we will use u^* and u_k for the boundary conditions of the velocity and scalars (in particular the temperature).

Modèle à une échelle de vitesse

$$\left\{ \begin{array}{l} u_k = u^* \\ u^* \text{ solution de } \left\{ \begin{array}{ll} \dfrac{u^r_{\tau,I'}}{u^*} = \dfrac{1}{\kappa} ln(y^+) + 5,2 & \text{pour } y^+ > y^+_{lim} \\ \dfrac{u^r_{\tau,I'}}{u^*} = y^+ & \text{pour } y^+ \leqslant y^+_{lim} \end{array} \right. \\ \qquad \text{avec } y^+ = \dfrac{u^* \, I'F}{\nu} \text{ et } \kappa = 0,42 \end{array} \right. \tag{IV.C.2}$$

Remark : Hereafter, we provide three exemples based on the two velocity scale model.

- In this way, one can implement a specific wall function :

$$\frac{u_{\tau,I'}}{u^*} = g(y^+)$$

  by simply imposing $u^* = u_{\tau,I'}/g(y^+)$.

- It is also possible to use a rough-wall wall function such as :

$$\frac{u_{\tau,I'}}{u^*} = \frac{1}{\kappa} \, ln(\frac{y}{\xi}) + 8,5$$

  where ξ is the height of the roughness elements at the wall : one just has to impose $u^* = u_{\tau,I'} / \left[\frac{1}{\kappa} ln(\frac{y}{\xi}) + 8,5 \right]$, y being deduced from y^+, available as an argument, by the equation $y = y^+ \dfrac{\nu}{u_k}$.

- Even a more general correlation could be used of Colebrook type :

$$u^* = u_{deb} / \left[-4\sqrt{2} log_{10} \left(\frac{2,51}{2\sqrt{2} D^+_H} + \frac{\xi}{3,7 \, D_H} \right) \right]$$

  where D^+_H is the hydraulic diameter made dimensionless using u_k, ν, u_{deb} the mean streamwise velocity and $\dfrac{\xi}{D_H}$ the relative roughness.

• Boundary conditions for the velocity in $k - \varepsilon$

We first consider the boundary conditions used in the case of calculation using the $k-\varepsilon$ model. Indeed these cases are the most complex and general.

The boundary conditions are necessary to prescribe at the boundary the correct tangential stress $\sigma_\tau = \rho_I u^* u_k$ in the momentum equation[2] (ρ_I is the density at the centre of cell I). The term which requires boundary conditions is the one containing the velocity derivative in the normal direction to the wall[3] : $(\mu_I + \mu_{t,I}) \underline{\underline{\text{grad}}} \, \underline{u} \, \underline{n}$. It appears on the right-hand side of the usual momentum equation (see `bilsc2` and `cs_velocity_prediction`).

[2] Proposition de modification des conditions aux limites de paroi turbulente pour le Solveur Commun dans le cadre du modèle $k - \varepsilon$ standard, rapport EDF HI-81/00/019/A, 2000, M. Boucker, J.-D. Mattei.

[3] The transpose gradient term is treated in `visecv` and thus will not be considered here.

In the case where the $k-\varepsilon$ model tends to surestimate the production of turbulent kinetic energy, the length scale of the model, $L_{k-\varepsilon}$, can become significantly larger than the maximum theoretical length scale of the turbulent boundary layer eddies L_{theo}. We write :

$$\left\{ \begin{array}{l} L_{k-\varepsilon} = C_\mu \dfrac{k^{\frac{3}{2}}}{\varepsilon} \\ L_{\text{theo}} = \kappa\, I'F \end{array} \right. \tag{IV.C.3}$$

In the case where $L_{k-\varepsilon} > L_{\text{théo}}$, we thus have $\mu_{t,I} > \mu_t^{lm}$ with $\mu_{t,I}$ the turbulent viscosity of the $k-\varepsilon$ model at point I and $\mu_t^{lm} = \rho_I L_{\text{theo}} u_k$ the turbulent viscosity of the mixing length model. Additionally, the tangential stress can write by making the turbulent viscosity appear :

$$\sigma_\tau = \rho_I u^* u_k = \frac{u^*}{\kappa\, I'F} \underbrace{\rho_I \kappa\, I'F\, u_k}_{\mu_t^{lm}} \tag{IV.C.4}$$

The viscosity scale introduced in the stress thus contradicts the one deduced from the neighbouring turbulence calculated by the model. Consequently we prefer to write the stress, by using the velocity scale of the $k-\varepsilon$ model when it is lower than the limit $L_{\text{théo}}$:

$$\sigma_\tau = \frac{u^*}{\kappa\, I'F} max(\mu_t^{lm}, \mu_{t,I}) \tag{IV.C.5}$$

One can then use this value to calculate the diffusive flux which depends upon it in the Navier-Stokes equations :

$$(\mu_I + \mu_{t,I}) \underline{\underline{\text{grad}}}\, \underline{u}\, \underline{n} = -\sigma_\tau \underline{\tau}. \tag{IV.C.6}$$

But the velocity gradient (face gradient) is computed in the code as :

$$(\mu_I + \mu_{t,I}) \underline{\underline{\text{grad}}}\, \underline{u}\, \underline{n} = \frac{(\mu_I + \mu_{t,I})}{\overline{I'F}} (\underline{u}_F - \underline{u}_{I'}) \tag{IV.C.7}$$

Using (IV.C.6) and (IV.C.7) we obtain the value of $\underline{u}_F$ to be prescribed, referred to as $\underline{u}_{F,flux}$ (conservation of the momentum flux) :

$$\begin{array}{rl} \underline{u}_{F,flux} & = \underline{u}_{I'} - \dfrac{\overline{I'F}}{\mu_I + \mu_{t,I}} \sigma_\tau \underline{\tau} \\ & = \underline{u}_{I'} - \dfrac{u^*}{\kappa\,(\mu_I + \mu_{t,I})} max(\mu_t^{lm}, \mu_{t,I}) \underline{\tau} \end{array} \tag{IV.C.8}$$

In reality, an extra approximation is made. It consists in imposing a zero normal velocity at the wall and in using equation (IV.C.8) projected on the plane parallel to the wall :

$$\underline{u}_{F,flux} = \left[u_{\tau,I'} - \frac{u^*}{\kappa\,(\mu_I + \mu_{t,I})} max(\mu_t^{lm}, \mu_{t,I}) \right] \underline{\tau} \tag{IV.C.9}$$

Moreover, if the value obtained for y^+ is lower than y^+_{lim} a no-slip condition is applied. Finally, one can also make the wall velocity appear in the final expression :

"Flux" boundary conditions of the velocity $(k-\varepsilon)$

$$\left\{ \begin{array}{lll} \underline{u}_{F,flux} & = \underline{v}_p & \text{if } y^+ \leqslant y^+_{lim} \\ \underline{u}_{F,flux} & = \underline{v}_p + \left[u^r_{\tau,I'} - \dfrac{u^*}{\kappa\,(\mu_I + \mu_{t,I})} max(\mu_t^{lm}, \mu_{t,I}) \right] \underline{\tau} & \text{otherwise} \end{array} \right. \tag{IV.C.10}$$

A first pair of coefficients A_{flux} and B_{flux} can then be deduced (for each component of the velocity separately) and it is used only to compute the tangential stress dependent term (see `bilsc2`) :

Coefficients associated with the "flux" boundary conditions of the velocity $(k-\varepsilon)$

$$\left\{ \begin{array}{l} \left\{ \begin{array}{ll} \underline{A}_{flux} & = \underline{v}_p \qquad \text{if } y^+ \leqslant y^+_{lim} \\ \underline{A}_{flux} & = \underline{v}_p + \left[u^r_{\tau,I'} - \dfrac{u^*}{\kappa\,(\mu_I+\mu_{t,I})} max(\mu_t^{lm},\mu_{t,I}) \right] \underline{\tau} \text{ otherwise} \end{array} \right. \\ \underline{B}_{flux} = \underline{0} \end{array} \right. \tag{IV.C.11}$$

We saw above how to impose a boundary condition to compute directly the stress term. Further analysis is necessary to calculate correctly the velocity gradients. We want to find a boundary face value which permits to obtain, with the chosen expression for the gradient, the value the turbulent production as close as possible to its theoretical value (determined by using the logarithmic law), in order to evaluate the normal derivative the tangential velocity. Thus, we define (at point I) :

$$P_{\text{théo}} = \rho_I u^* u_k \| \frac{\partial u_\tau}{\partial \underline{n}} \|_I = \rho_I \frac{u_k (u^*)^2}{\kappa\, I'F} \tag{IV.C.12}$$

Morevoer, the dominant term of the production computed in cell I is, in classical situations (y is the coordinate on the axis whose direction vector is $\underline{\tilde{n}}$),

$$P_{\text{calc}} = \mu_{t,I} \left(\frac{\partial u_\tau}{\partial y} \right)^2_I \tag{IV.C.13}$$

The normal gradient of the tangential velocity (cell gradient) is calculated in the code using finite volume, and its expression on regular orthogonal meshes is (see the notations on figure IV.C.2) :

$$P_{\text{calc}} = \mu_{t,I} \left(\frac{u_{\tau,G} - u_{\tau,F}}{2d} \right)^2 = \mu_{t,I} \left(\frac{u_{\tau,I} + u_{\tau,J} - 2u_{\tau,F}}{4d} \right)^2 \tag{IV.C.14}$$

We then assume that $u_{\tau,J}$ can be obtained from $u_{\tau,I}$ and from the normal gradient of u_τ calculated in G from the logarithmic law :

$$u_{\tau,J} = u_{\tau,I} + IJ\,.\,(\partial_y u_\tau)_G + \mathcal{O}(IJ^2) \approx u_{\tau,I} + IJ\,.\,\left[\partial_y \left(\frac{u^*}{\kappa}\, ln(y^+) + 5,2 \right) \right]_G = u_{\tau,I} + 2d \frac{u^*}{\kappa\, 2d} \tag{IV.C.15}$$

and thus we obtain :

$$\begin{array}{rcl} P_{\text{calc}} & = & \mu_{t,I} \left(\dfrac{u_{\tau,I} + u_{\tau,I} + 2d\frac{u^*}{\kappa\,2d} - 2u_{\tau,F}}{4d} \right)^2 \\ & = & \mu_{t,I} \left(\dfrac{2u_{\tau,I} + 2\frac{u^*}{2\kappa} - 2u_{\tau,F}}{4d} \right)^2 = \mu_{t,I} \left(\dfrac{u_{\tau,I} + \frac{u^*}{2\kappa} - u_{\tau,F}}{2d} \right)^2 \end{array} \tag{IV.C.16}$$

We then use (IV.C.12) and (IV.C.16) to impose that the calculated production is equal to the theoretical production. The preceeding formulae are extended with no precaution to non-orthogonal meshes (the velocity at I is then simply computed at I'). The following expression for $u_{\tau,F}$ is then obtained :

$$u_{\tau,F,grad} = u_{\tau,I'} - \frac{u^*}{\kappa} \left(2\sqrt{\frac{\rho_I \kappa\, u_k I'F}{\mu_{t,I}}} - \frac{1}{2} \right) \tag{IV.C.17}$$

Additionally, we force the gradient to remain as stiff as the one given by the normal derivative of the theoretical velocity profile (logarithmic) at I' :
$\partial_y u_\tau = \partial_y (\frac{u^*}{\kappa}\, ln(y^+) + 5,2) = \dfrac{u^*}{\kappa\, \overline{I'F}}$, thus :

$$u_{\tau,F,grad} = u_{\tau,I'} - \frac{u^*}{\kappa} max \left(1, 2\sqrt{\frac{\rho_I \kappa\, u_k I'F}{\mu_{t,I}}} - \frac{1}{2} \right) \tag{IV.C.18}$$

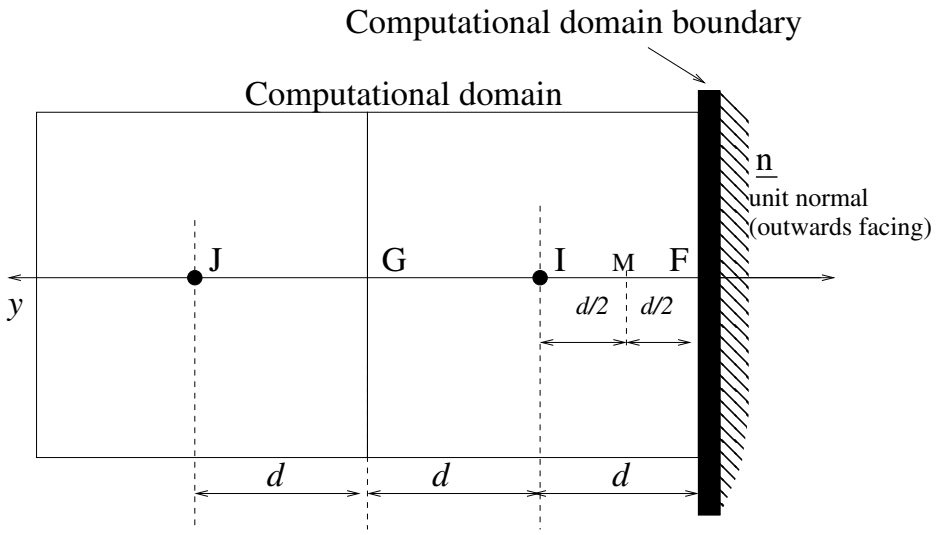


Figure IV.C.2: Cellule de bord - Maillage orthogonal.

Finally, we clip the velocity at the wall with a minimum value calculated assuming that we are in the logarithmic layer :

$$u_{\tau,F,grad} = max\left(u^*\left(\frac{1}{\kappa}ln(y^+_{lim})+5,2\right), u_{\tau,I'} - \frac{u^*}{\kappa}\left[max\left(1, 2\sqrt{\frac{\rho_I \kappa\, u_k I'F}{\mu_{t,I}}} - \frac{1}{2}\right)\right]\right) \quad \text{(IV.C.19)}$$

The normal derivative at the wall is prescribed to zero. If the y^+ value at the wall is lower than y^+_{lim}, a no-slip condition is prescribed. Finally, one can also make explicit the velocity of the wall in the final expression :

"Gradient" boundary conditions of the velocity$(k-\varepsilon)$

$$\begin{cases} \underline{u}_{F,grad} = \underline{v}_p & \text{if } y^+ \leqslant y^+_{lim} \\ \underline{u}_{F,grad} = \underline{v}_p + & \\ \left\{ max\left(u^*\left(\frac{1}{\kappa}ln(y^+_{lim})+5,2\right), u^r_{\tau,I'} - \frac{u^*}{\kappa}\left[max\left(1, 2\sqrt{\frac{\rho_I \kappa\, u_k I'F}{\mu_{t,I}}} - \frac{1}{2}\right)\right]\right) \right\} \underline{\tau} & \text{otherwise} \end{cases} \quad \text{(IV.C.20)}$$

A second pair of coefficients A_{grad} and B_{grad} can then be deduced (for each velocity component separately). It is used when the velocity gradient is necessary (except for the terms depending on the tangential shear, those being treated in `bilsc2` using A_{flux} and B_{flux}) :

Coefficients associated to the "gradient" boundary conditions of the velocity$(k-\varepsilon)$

$$\begin{cases} \begin{cases} \underline{A}_{grad} = \underline{v}_p & \text{if } y^+ \leqslant y^+_{lim} \\ \underline{A}_{grad} = \underline{v}_p + & \\ \left\{ max\left(u^*\left(\frac{1}{\kappa}ln(y^+_{lim})+5,2\right), u^r_{\tau,I'} - \frac{u^*}{\kappa}\left[max\left(1, 2\sqrt{\frac{\rho_I \kappa\, u_k I'F}{\mu_{t,I}}} - \frac{1}{2}\right)\right]\right) \right\} \underline{\tau} & \text{otherwise} \end{cases} \\ \underline{B}_{grad} = \underline{0} \end{cases} \quad \text{(IV.C.21)}$$

• **Boundary conditions of the velocity in** $R_{ij}-\varepsilon$

The boundary conditions of the velocity with the $R_{ij}-\varepsilon$ model are more simple, since there are only of one type. Keeping the same notations as above, we want the tangential velocity gradient (calculated

at I, and to be used to evaluate the turbulent production) to be consistent with the logarithmic law giving the ideal tangential velocity profile. The theoretical gradient is :

$$G_{\text{theo}} = \left(\frac{\partial u_\tau}{\partial y}\right)_{I'} = \frac{u^*}{\kappa\, I'F} \tag{IV.C.22}$$

The normal gradient of the tangential velocity (cell gradient) is calculated in the code using finite volumes, and its expression in the case of regular orthogonal meshes is (see notations in figure IV.C.2) :

$$G_{\text{calc}} = \frac{u_{\tau,G} - u_{\tau,F}}{2d} = \frac{u_{\tau,I} + u_{\tau,J} - 2u_{\tau,F}}{4d} \tag{IV.C.23}$$

We then assume that $u_{\tau,J}$ can be obtained from $u_{\tau,I}$ and from the normal gradient of u_τ calculated in G from the logarithmic law (see equation (IV.C.15)) $u_{\tau,J} = u_{\tau,I} + 2d\dfrac{u^*}{\kappa\, 2d}$ and we thus obtain :

$$G_{\text{calc}} = \frac{u_{\tau,I} + u_{\tau,I} + 2d\dfrac{u^*}{\kappa\, 2d} - 2u_{\tau,F}}{4d} = \frac{2u_{\tau,I} + 2\dfrac{u^*}{2\kappa} - 2u_{\tau,F}}{4d} = \frac{u_{\tau,I} + \dfrac{u^*}{2\kappa} - u_{\tau,F}}{2d} \tag{IV.C.24}$$

We then use the equations (IV.C.22) and (IV.C.24) to derive an expression for $u_{\tau,F}$ (the preceeding formulae are extended with no precaution to the case non-orthogonal meshes, the velocity at I being simply computed at I') :

$$u_{\tau,F} = u_{\tau,I'} - \frac{3u^*}{2\kappa} \tag{IV.C.25}$$

The normal derivative at the wall is prescribed to zero. If the value obtained for y^+ at the wall is lower than y^+_{lim}, a no-slip condition is prescribed. Finally, one can also make explicit the velocity of the wall in the final expression :

Boundary conditions of the velocity $(R_{ij} - \varepsilon)$

$$\begin{cases} \underline{u}_F = \underline{v}_p & \text{if } y^+ \leqslant y^+_{lim} \\ \underline{u}_F = \left[u^r_{\tau,I'} - \dfrac{3u^*}{2\kappa}\right]\underline{\tau} + \underline{v}_p & \text{otherwise} \end{cases} \tag{IV.C.26}$$

Un couple de coefficients A et B s'en déduit (pour chaque composante de vitesse séparément) :

Coefficients associés aux conditions aux limites sur la vitesse $(R_{ij} - \varepsilon)$

$$\begin{cases} \begin{cases} \underline{A} = \underline{v}_p & \text{si } y^+ \leqslant y^+_{lim} \\ \underline{A} = \left[u^r_{\tau,I'} - \dfrac{3u^*}{2\kappa}\right]\underline{\tau} + \underline{v}_p & \text{sinon} \end{cases} \\ \underline{B} = \underline{0} \end{cases} \tag{IV.C.27}$$

A pair of coefficients A_{grad} and B_{grad} can be deduced from the above equation (for each velocity component separately).

• Boundary conditions of the velocity in laminar

When no turbulence model is activated, we implicitly use a one velocity scale model (there is no turbulent variables to compute u_k), and the same conditions [4] as in $R_{ij} - \varepsilon$ are used : the model degenerates automatically.

[4] In other words; the boundary conditions are given by (IV.C.26) and (IV.C.27).

• **Boundary conditions for k and ε (standard $k-\varepsilon$ model)**

We impose k with a Dirichlet condition using the friction velocity u_k (see equation (IV.C.1)) :

$$k = \frac{u_k^2}{C_\mu^{\frac{1}{2}}} \tag{IV.C.28}$$

We want to impose the normal derivative of ε from of the following theoretical law (see the notations in figure IV.C.2) :

$$G_{\text{theo},\varepsilon} = \frac{\partial\left(u_k^3/(\kappa\, y)\right)}{\partial y} \tag{IV.C.29}$$

We use point M to impose a boundary condition with a higher order of accuracy in space. Indeed, using the simple expression $\varepsilon_F = \varepsilon_I + d\partial_y\varepsilon_I + O(d^2)$ leads to first order accuracy. A second order accuracy can be reached using the following Taylor series expansion:

$$\left\{\begin{array}{ll} \varepsilon_M & = \varepsilon_I - \dfrac{d}{2}\partial_y\varepsilon_I + \dfrac{d^2}{8}\partial_y^2\varepsilon_I + O(d^3) \\ \varepsilon_M & = \varepsilon_F + \dfrac{d}{2}\partial_y\varepsilon_F + \dfrac{d^2}{8}\partial_y^2\varepsilon_F + O(d^3) \end{array}\right. \tag{IV.C.30}$$

By substracting these twxo expression, we obtain

$$\varepsilon_F = \varepsilon_I - \frac{d}{2}(\partial_y\varepsilon_I + \partial_y\varepsilon_F) + O(d^3) \tag{IV.C.31}$$

Additionally, we have

$$\left\{\begin{array}{ll} \partial_y\varepsilon_I & = \partial_y\varepsilon_M + d\partial_y^2\varepsilon_M + O(d^2) \\ \partial_y\varepsilon_F & = \partial_y\varepsilon_M - d\partial_y^2\varepsilon_M + O(d^2) \end{array}\right. \tag{IV.C.32}$$

The sum of these last two expressions gives $\partial_y\varepsilon_I + \partial_y\varepsilon_F = 2\partial_y\varepsilon_M + O(d^2)$ and, using equation (IV.C.31), we finally obtain a second order approximation for ε_F :

$$\varepsilon_F = \varepsilon_I - d\partial_y\varepsilon_M + O(d^3) \tag{IV.C.33}$$

The theoretical value (see equation IV.C.29) is then used in order to evaluate $\partial_y\varepsilon_M$ and thus the value to prescribe at the boundary is obtained ($d = I'F$) :

$$\varepsilon_F = \varepsilon_I + d\frac{u_k^3}{\kappa\,(d/2)^2} \tag{IV.C.34}$$

This expression is extended to non-orthogonal mesh without any precautions (which is bound to deteriorate the spatial accuracy ot the method).

Additionally, the velocity u_k is set to zero for $y^+ \leqslant y^+_{lim}$. Consequently, the value of k and the flux of ε are both zero.

Finally we have :

Boundary conditions for k and ε

$$\left\{\begin{array}{ll} k_F & = \dfrac{u_k^2}{C_\mu^{\frac{1}{2}}} \\ \varepsilon_F & = \varepsilon_{I'} + I'F\dfrac{u_k^3}{\kappa\,(I'F/2)^2} \end{array}\right. \tag{IV.C.35}$$

with $u_k = 0$ if $y^+ \leqslant y^+_{lim}$

and the associated pair of coefficients

$$\textbf{Coefficients associated to the boundary conditions of } k \textbf{ et } \varepsilon$$
$$\begin{cases} A_k = \dfrac{u_k^2}{C_\mu^{\frac{1}{2}}} & \text{and } B_k = 0 \\ A_\varepsilon = I'F\dfrac{u_k^3}{\kappa\,(I'F/2)^2} & \text{and } B_\varepsilon = 1 \end{cases} \quad \text{with } u_k = 0 \text{ if } y^+ \leqslant y_{lim}^+ \tag{IV.C.36}$$

• **Boundary conditions for R_{ij} and ε (standard $R_{ij} - \varepsilon$ model)**

The boundary conditions for the Reynolds stresses in the coordinate system attached to the wall write ($\hat{R}$ refers to the local coordinate system) :

$$\partial_{\tilde{n}}\hat{R}_{\tau\tau} = \partial_{\tilde{n}}\hat{R}_{\tilde{n}\tilde{n}} = \partial_{\tilde{n}}\hat{R}_{bb} = 0 \quad \text{et } \hat{R}_{\tau\tilde{n}} = -u^* u_k \quad \text{et } \hat{R}_{\tau b} = \hat{R}_{\tilde{n}b} = 0 \tag{IV.C.37}$$

Additionally, if the value obtained for y^+ is lower than y_{lim}^+, all Reynolds stresses are set to zero (we assume that the turbulent stresses are negligible compared to the viscous stresses).

Although it is done systematically in the code, expressing the above boundary conditions in the computation coordinate system is relatively complex (rotation of a tensor): the reader is referred to the documentation of `clsyvt` where more details are provided. In what follows, the boundary conditions will only be presented in the local coordinate system.

Thus we want to impose the boundary values :

$$\textbf{Boundary conditions of } R_{ij}$$
$$\begin{cases} \text{if } y^+ \leqslant y_{lim}^+ & \hat{R}_{\alpha\alpha,F} = \hat{R}_{\alpha\beta,F} = 0 \\ \text{otherwise} & \begin{cases} \hat{R}_{\alpha\alpha,F} = \hat{R}_{\alpha\alpha,I'} \text{ with } \alpha \in \{\tau, \tilde{n}, b\} \text{ (without summation)} \\ \hat{R}_{\tau\tilde{n}} = -u^* u_k \text{ and } \hat{R}_{\tau b} = \hat{R}_{\tilde{n}b} = 0 \end{cases} \end{cases} \tag{IV.C.38}$$

For the dissipation, the boundary condition applied is identical to the one applied with the $k - \varepsilon$ model :

$$\textbf{Boundary conditions of } \varepsilon \ (R_{ij} - \varepsilon)$$
$$\begin{cases} \varepsilon_F = \varepsilon_{I'} + I'F\dfrac{u_k^3}{\kappa\,(I'F/2)^2} \\ \text{with } u_k = 0 \text{ if } y^+ \leqslant y_{lim}^+ \end{cases} \tag{IV.C.39}$$

These boundary conditions can be imposed explicitly (by default, ICLPTR=0) or (semi-)implicitly (ICLPTR=1). The standard option (explicit) leads to the following values[5] of the coefficients A and B :

[5]It can be noticed that the value of ε is not reconstructed at I'. We thus wish to improve the "stability" since ε has a very steep gradient at the wall ($\varepsilon \approx \dfrac{1}{y}$), and thus only weak recontruction errors at I' could lead to important deterioration of the results. However, it would be necessary to check if stability is altered with the gradient reconstruction of `gradrc`.

Coefficients associated to the explicit boundary conditions of R_{ij}et ε

$$\left\{\begin{array}{l}
\text{If } y^+ \leqslant y^+_{lim} : \\
\qquad A_{\hat{R}_{\alpha\alpha}} = A_{\hat{R}_{\alpha\beta}} = 0 \quad \text{and } B_{\hat{R}_{\alpha\alpha}} = B_{\hat{R}_{\alpha\beta}} = 0 \\
\text{Otherwise :} \\
\qquad \left\{\begin{array}{lll}
A_{\hat{R}_{\alpha\alpha}} = (R_{\alpha\alpha})_I & \text{and } B_{\hat{R}_{\alpha\alpha}} = 0 & \text{with } \alpha \in \{\tau, \tilde{n}, b\} \text{ (without summation)} \\
A_{\hat{R}_{\tau\tilde{n}}} = -u^* u_k & \text{and } B_{\hat{R}_{\tau\tilde{n}}} = 0 & \\
A_{\hat{R}_{\tau b}} = A_{\hat{R}_{\tilde{n}b}} = 0 & \text{and } B_{\hat{R}_{\tau b}} = B_{\hat{R}_{\tilde{n}b}} = 0 &
\end{array}\right. \\
\text{And for all cases :} \\
\qquad A_\varepsilon = \varepsilon_I + I'F \dfrac{u_k^3}{\kappa\,(I'F/2)^2} \text{ and } B_\varepsilon = 0
\end{array}\right.$$

with $u_k = 0$ if $y^+ \leqslant y^+_{lim}$

(IV.C.40)

The semi-implicit option leads to the following values for the coefficients A and B. They differ from the preceeding ones, only as regards as the diagonal Reynolds stresses and dissipation. In the general case, impliciting of some components of the tensor in the local coordinate system leads to partially impliciting all the components in the global computation coordinate system :

Coefficients associated to the semi-implicit boundary conditions of sur les variables R_{ij}et ε

$$\left\{\begin{array}{l}
\text{If } y^+ \leqslant y^+_{lim} : \\
\qquad A_{\hat{R}_{\alpha\alpha}} = A_{\hat{R}_{\alpha\beta}} = 0 \quad \text{and } B_{\hat{R}_{\alpha\alpha}} = B_{\hat{R}_{\alpha\beta}} = 0 \\
\text{Sinon :} \\
\qquad \left\{\begin{array}{lll}
A_{\hat{R}_{\alpha\alpha}} = 0 & \text{and } B_{\hat{R}_{\alpha\alpha}} = 1 & \text{with } \alpha \in \{\tau, \tilde{n}, b\} \text{ (without summation)} \\
A_{\hat{R}_{\tau\tilde{n}}} = -u^* u_k & \text{and } B_{\hat{R}_{\tau\tilde{n}}} = 0 & \\
A_{\hat{R}_{\tau b}} = A_{\hat{R}_{\tilde{n}b}} = 0 & \text{and } B_{\hat{R}_{\tau b}} = B_{\hat{R}_{\tilde{n}b}} = 0 &
\end{array}\right. \\
\text{And for all cases :} \\
\qquad A_\varepsilon = I'F \dfrac{u_k^3}{\kappa\,(I'F/2)^2} \text{ and } B_\varepsilon = 1
\end{array}\right.$$

with $u_k = 0$ if $y^+ \leqslant y^+_{lim}$

(IV.C.41)

• **Boundary conditions of the** *VarScalaires*

Only the boundary conditions when a boundary value is imposed (at the wall or away from the wall with a possible external exchange coefficient) are treated here. The reader is referred to the notations in figure IV.C.1 and to the general presentation provided in `condli` (in what follows only the most essential part of the presentation is repeated).

The conservation of the normal flux at the boundary for variable f writes :

$$\underbrace{h_{int}(f_{b,int}-f_{I'})}_{\phi_{int}} = \underbrace{h_b(f_{b,ext}-f_{I'})}_{\phi_b} = \begin{cases} \underbrace{h_{imp,ext}(f_{imp,ext}-f_{b,ext})}_{\phi_{real\ imposed}} & \text{(Dirichlet condition)} \\ \underbrace{\phi_{imp,ext}}_{\phi_{real\ imposed}} & \text{(Neumann condition)} \end{cases} \tag{IV.C.42}$$

The above two equation are rearranged in order to obtain the value of the numerical flux $f_{b,int} = f_F$ to impose at the wall boundary face, according to the values of $f_{imp,ext}$ and $h_{imp,ext}$ set by the user, and to the value of h_b set by the similarity laws detailed hereafter. The coefficients A and B can then be readily derived, and are presented here.

Boundary conditions of the *VarScalaires*

$$f_{b,int} = \underbrace{\frac{h_{imp,ext}}{h_{int}+h_r h_{imp,ext}} f_{imp,ext}}_{A} + \underbrace{\frac{h_{int}+h_{imp,ext}(h_r-1)}{h_{int}+h_r h_{imp,ext}} f_{I'}}_{B} \text{ with } h_r = \frac{h_{int}}{h_b} \tag{IV.C.43}$$

Similarity principle : calculation of h_b.

The only remaining unknown in expression (IV.C.43) is the value of h_b, since h_{int} has a numerical value which is coherent with the face gradient computation options detailed in `condli` ($h_{int} = \frac{\alpha}{\overline{I'F}}$). The value of h_b must relate the flux to the values $f_{I'}$ and $f_{b,ext}$ by taking into account the boundary layer (the profile of f is not always linear) :

$$\phi_b = h_b \left(f_{b,ext} - f_{I'} \right) \tag{IV.C.44}$$

The following considerations are presented using the general notations. In particular, the Prandtl-Schmidt number writes $\sigma = \dfrac{\nu \rho C}{\alpha}$. When the considered scalar f is the temperature, we have (see `condli`)

- $C = C_p$ (specific heat capacity),
- $\alpha = \lambda$ (molecular conductivity),
- $\sigma = \dfrac{\nu \rho C_p}{\lambda} = Pr$ (Prandtl number),
- $\sigma_t = Pr_t$ (turbulent Prandtl number),
- $\phi = \left(\lambda + \dfrac{C_p \mu_t}{\sigma_t} \right) \dfrac{\partial T}{\partial y}$ (flux in $W m^{-2}$).

The reference "Convection Heat Transfer", Vedat S. Arpaci and Poul S. Larsen, Prentice-Hall, Inc was used.

The flux at the wall writes for the scalar f (the flux is positive if it enters the fluid domain, as shown by the orientation of the y axis) :

$$\phi = - \left(\alpha + C \frac{\mu_t}{\sigma_t} \right) \frac{\partial f}{\partial y} = - \rho C \left(\frac{\alpha}{\rho C} + \frac{\mu_t}{\rho \sigma_t} \right) \frac{\partial f}{\partial y} \tag{IV.C.45}$$

Similarly for the temperature, with $a = \dfrac{\lambda}{\rho C_p}$ and $a_t = \dfrac{\mu_t}{\rho \sigma_t}$, we have :

$$\phi = - \rho C_p (a + a_t) \frac{\partial T}{\partial y} \tag{IV.C.46}$$

In order to make f dimensionless, we introduce f^* defined using the flux at the boundary ϕ_b :

$$f^* = - \frac{\phi_b}{\rho C u_k} \tag{IV.C.47}$$

For the temperature, we thus have :

$$T^* = - \frac{\phi_b}{\rho C_p u_k} \tag{IV.C.48}$$

We then divide both sides of equation (IV.C.45) by ϕ_b. For the left-hand side, we simplify using the conservation of the flux (and thus the fact that $\phi = \phi_b$). For the right-hand side, we replace ϕ_b by its value $-\rho C u_k f^*$. With the notations :

$$\nu = \frac{\mu}{\rho} \qquad \nu_t = \frac{\mu_t}{\rho} \qquad y^+ = \frac{y u_k}{\nu} \qquad f^+ = \frac{f - f_{b,ext}}{f^*} \tag{IV.C.49}$$

we have :

$$1 = \left(\frac{1}{\sigma} + \frac{1}{\sigma_t} \frac{\nu_t}{\nu} \right) \frac{\partial f^+}{\partial y^+} \tag{IV.C.50}$$

One can remark at this stage that with the notations used in the preceeding, h_b can be expressed as a function of $f_{I'}^+$:

$$h_b = \frac{\phi_b}{f_{b,ext} - f_{I'}} = \frac{\rho\, C\, u_k}{f_{I'}^+} \tag{IV.C.51}$$

In order to compute h_b, we integrate equation (IV.C.50) to obtain $f_{I'}^+$. The only difficulty then consists in prescribing a variation law $\mathcal{K} = \frac{1}{\sigma} + \frac{1}{\sigma_t}\frac{\nu_t}{\nu}$.

In the fully developed turbulent region (far enough from the wall, for $y^+ \geqslant y_2^+$), a mixing length hypothesis models the variations of ν_t :

$$\nu_t = l^2 \left|\frac{\partial U}{\partial y}\right| = \kappa\, y\, u^* \tag{IV.C.52}$$

Additionally, the molecular diffusion of f (or the conduction when f represents the temperature) is negligible compared to its turbulent diffusion : therefore we neglect $\frac{1}{\sigma}$ compared to $\frac{1}{\sigma_t}\frac{\nu_t}{\nu}$. Finally we have [6] :

$$\mathcal{K} = \frac{\kappa\, y^+}{\sigma_t} \tag{IV.C.53}$$

On the contrary, in the near-wall region (for $y^+ < y_1^+$) the turbulent contribution becomes negligible compared to the molecular contribution and we thus neglect $\frac{1}{\sigma_t}\frac{\nu_t}{\nu}$ compared to $\frac{1}{\sigma}$.

It would be possible to restrict ourselves to these two regions, but Arpaci and Larsen suggest the model can be improved by introducing an intermediate region ($y_1^+ \leqslant y^+ < y_2^+$) in which the following hypothesis is made :

$$\frac{\nu_t}{\nu} = a_1 (y^+)^3 \tag{IV.C.54}$$

where a_1 is a constant whose value is obtained from experimental correlations :

$$a_1 = \frac{\sigma_t}{1000} \tag{IV.C.55}$$

Thus the following model is used for $\mathcal{K}$ (see a sketch in figure C) :

$$\mathcal{K} = \begin{cases} \dfrac{1}{\sigma} & \text{if } y^+ < y_1^+ \\ \dfrac{1}{\sigma} + \dfrac{a_1 (y^+)^3}{\sigma_t} & \text{if } y_1^+ \leqslant y^+ < y_2^+ \\ \dfrac{\kappa\, y^+}{\sigma_t} & \text{if } y_2^+ \leqslant y^+ \end{cases} \tag{IV.C.56}$$

The values of y_1^+ and y_2^+ are obtained by calculating the intersection points of the variations laws used for $\mathcal{K}$.

The existence of an intermediate region depends upon the values of σ. Let's first consider the case where σ cannot be neglected compared to 1. In practise we consider $\sigma > 0,1$ (this is the common case when scalar f represents the air or the water temperature in normal temperature and pressure conditions). It is assumed that $\frac{1}{\sigma}$ can be neglected compared to $\frac{a_1 (y^+)^3}{\sigma_t}$ in the intermediate region. We thus obtain :

$$y_1^+ = \left(\frac{1000}{\sigma}\right)^{\frac{1}{3}} \qquad y_2^+ = \sqrt{\frac{1000\kappa}{\sigma_t}} \tag{IV.C.57}$$

[6] We make the approximation that the definitions of y^+ from u^* and u_k are equivalent.

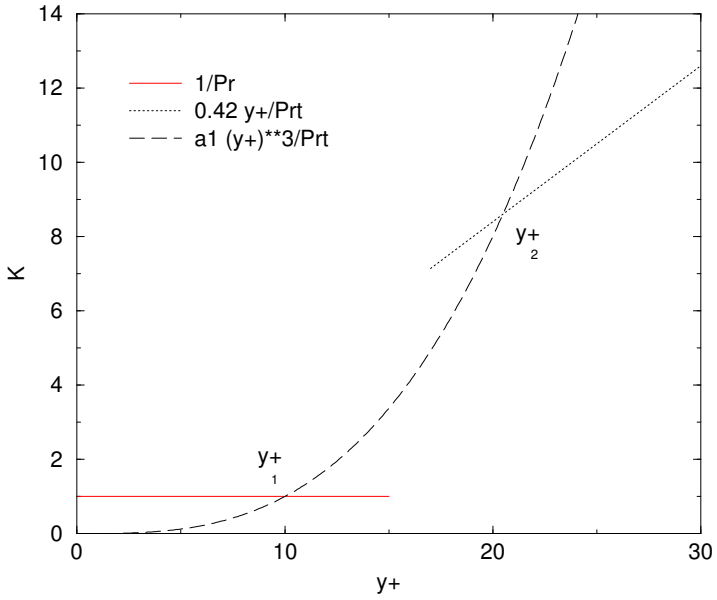


Figure IV.C.3: $(a + a_t)/\nu$ as a function of y^+ obtained for $\sigma = 1$ and $\sigma_t = 1$.

The dimensionless equation (IV.C.50) is integrated under the same hypothesis and we obtain the law of f^+ :

$$\begin{cases} f^+ = \sigma\, y^+ & \text{if } y^+ < y_1^+ \\ f^+ = a_2 - \dfrac{\sigma_t}{2\, a_1\, (y^+)^2} & \text{if } y_1^+ \leqslant y^+ < y_2^+ \\ f^+ = \dfrac{\sigma_t}{\kappa}\, ln(y^+) + a_3 & \text{if } y_2^+ \leqslant y^+ \end{cases} \tag{IV.C.58}$$

where a_2 and a_3 are integration constants, which have been chosen to obtain a continuous profile of f^+ :

$$a_2 = 15\sigma^{\frac{2}{3}} \qquad a_3 = 15\sigma^{\frac{2}{3}} - \frac{\sigma_t}{2\kappa}\left(1 + ln\left(\frac{1000\kappa}{\sigma_t}\right)\right) \tag{IV.C.59}$$

Let's now study the case where σ is much smaller than 1. In practise it is assumed that $\sigma \leqslant 0,1$ (this is for instance the case for liquid metals whose thermal conductivity is very large, and who have Prandtl number of values of the order of 0.01). The intermediate region then disappears and the coordinate of the interface between the law used in the near-wall region and the one used away from the wall is given by :

$$y_0^+ = \frac{\sigma_t}{\kappa\sigma} \tag{IV.C.60}$$

The dimensionless equation (IV.C.50) is then integrated under the same hypothesis, and the law of f^+ is obtained :

$$\begin{cases} f^+ = \sigma\, y^+ & \text{if } y^+ \leqslant y_0^+ \\ f^+ = \dfrac{\sigma_t}{\kappa}\, ln\left(\dfrac{y^+}{y_0^+}\right) + \sigma\, y_0^+ & \text{if } y_0^+ < y^+ \end{cases} \tag{IV.C.61}$$

To summarize, the computation of h_b

$$h_b = \frac{\phi_b}{f_{b,ext} - f_{I'}} = \frac{\rho\, C\, u_k}{f_{I'}^+} \tag{IV.C.62}$$

is performed by calculating $f_{I'}^+$ from $y^+ = y_{I'}^+$ using the following laws.

If $\sigma \leqslant 0,1$, a two-layer model is used :

$$\begin{cases} f^+ = \sigma\, y^+ & \text{if } y^+ \leqslant y_0^+ \\ f^+ = \dfrac{\sigma_t}{\kappa}\, ln\left(\dfrac{y^+}{y_0^+}\right) + \sigma\, y_0^+ & \text{if } y_0^+ < y^+ \end{cases} \tag{IV.C.63}$$

with

$$y_0^+ = \frac{\sigma_t}{\kappa\sigma} \tag{IV.C.64}$$

If $\sigma > 0,1$, a three-layer model is used :

$$\begin{cases} f^+ = \sigma\, y^+ & \text{if } y^+ < y_1^+ \\ f^+ = a_2 - \dfrac{\sigma_t}{2\, a_1\, (y^+)^2} & \text{if } y_1^+ \leqslant y^+ < y_2^+ \\ f^+ = \dfrac{\sigma_t}{\kappa}\, ln(y^+) + a_3 & \text{if } y_2^+ \leqslant y^+ \end{cases} \tag{IV.C.65}$$

with

$$y_1^+ = \left(\frac{1000}{\sigma}\right)^{\frac{1}{3}} \qquad y_2^+ = \sqrt{\frac{1000\kappa}{\sigma_t}} \tag{IV.C.66}$$

and

$$a_2 = 15\sigma^{\frac{2}{3}} \qquad a_3 = 15\sigma^{\frac{2}{3}} - \frac{\sigma_t}{2\kappa}\left(1 + ln\left(\frac{1000\kappa}{\sigma_t}\right)\right) \tag{IV.C.67}$$

Points to treat

The use of `HFLUI/CPP` when `ISCSTH` is 2 (case with radiation) needs to be checked (`CPP` is actually 1 in this case).

The boundary conditions of the velocity are based on derivations focusing on only one term of the tangential stress $(\mu_I + \mu_{t,I})(\underline{\underline{\text{grad}}}\,\underline{u})\,\underline{n}$ without taking into account the tranpose gradient.

In order to establish the boundary conditions of the velocity in $k-\varepsilon$ based on the constraint , a projection onto the plane tangent to the wall and an arbitrary zero normal velocity are introduced.

The hypothesis made in order to establish formulae for the different types of boundary conditions (dissipation, velocities) are based on the assumption that the mesh is orthogonal at the wall. This assumption is extended without any caution to the case of non-orthogonal meshes.

The one velocity scale (`cs_wall_functions.c`) wall function requires solving an equation using a Newton algorithm. The computational cost of the latter is low. One can also used a 1/7 power law (Werner et Wengle) which yields results which are as accurate as the logarithmic law in the logarithmic region, and which permits analytical resolutions (chosen option in LES mode). Be careful however, since with this law, the intersection with the linear law is slightly different, which thus requires some adaptations (intersection around 11.81 instead of 10.88 for the law adopted here $U^+ = 8,3\,(y^+)^{\frac{1}{7}}$).

The values of all the physical properties are taken at the cell centres, without any reconstruction. Without modifying this approach, it would be useful to keep this in mind.

For the thermal law with very small Prandtl numbers compared to 1, Arpaci and Larsen suggest $y_0^+ \simeq 5/Pr$ (with proof from experimental data) rather than $Pr_t/(Pr\,\kappa)$ (current value, computed as the analytical intersection of the linear and logarithmic laws considered). One should address this question.

D- clptrg routine

Function

This subroutine is designed to realize the calculation of wall boundary conditions for rough walls. The notations used are those introduced in `CONDLI` for general boundary conditions.

The wall boundary conditions discussed herein are taken to include all the boundary conditions for the velocity, the turbulent quantities (k, ε), the temperature when it has a prescribed value at the wall (or the enthalpy and more generally the *VarScalaires*[1], to be treated at the wall by applying a similarity law for the associated boundary layer. For the *VarScalaires* especially, when the boundary conditions at the wall are of the Neumann type (homogeneous or not), they are treated in `condli` and hence will not be discussed in this section. In particular, the boundary conditions of the *VarScalaires* representing the flux variance of other *VarScalaires* are not addressed here because their treatment at the wall is of the homogeneous Neumann type.

We explain the method used to calculate the pair of coefficients A_b and B_b, which are used for the computation of certain discrete terms in the equations to solve and notably allow us to determine a value associated to the boundary faces $f_{b,int}$ (at a point located at the

centre

of the boundary face, barycentre of its vertices) using the relation $f_{b,int} = A_b + B_b\, f_{I'}$ ($f_{I'}$ is the value of the variable at point I', projection of the centre of the boundary cell on the normal to the boundary face and passing through its centre: see figure IV.D.1).

See the programmers reference of the dedicated subroutine for further details.

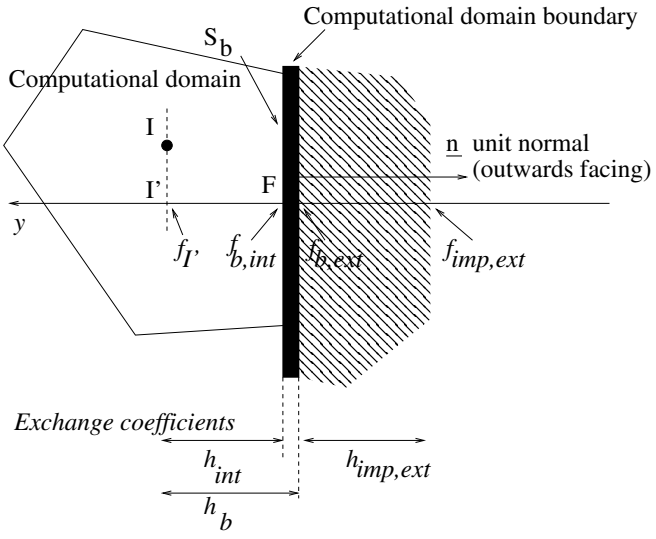


Figure IV.D.1: Boundary cell.

[1] As in `condli`, *VarScalaire* will be used to designate any variable, solution of a convection-diffusion equation, apart from the velocity, pressure and the turbulent quantities k, ε. The designation *VarScalaire* may be employed particularly in relation to the temperature, the enthalpy or to a passive scalar.

Discretisation

• **Notations**

The velocity of the wall is noted $\underline{v}_p$ and is assumed to be projected onto the plane tangent to the wall (if it is not, then it will be projected by the code).

The velocity of the fluid is noted by $\underline{u}$. An index, I, I' or F, serves to designate the point at which it is evaluated. The tangential velocity component in relation to the wall is denoted u_τ. The fluid velocity in the reference frame attached to the wall (

relative

velocity) is written $\underline{u}^r = \underline{u} - \underline{v}_p$.

The orthonormal coordinate system attached to the wall is written $\hat{\mathcal{R}} = (\underline{\tau}, \underline{\tilde{n}}, \underline{b})$.

- $\underline{\tilde{n}} = -\underline{n}$ is the unit vector orthogonal to the wall and directed towards the interior of the computational domain.
- $\underline{\tau} = \dfrac{1}{\|\underline{u}^r_{I'} - (\underline{u}^r_{I'} \,.\, \underline{\tilde{n}})\|} \left[\underline{u}^r_{I'} - (\underline{u}^r_{I'} \,.\, \underline{\tilde{n}})\right]$ is the unit vector parallel to the projection of the relative velocity at I', $\underline{u}^r_{I'}$, in the plane tangent to the wall (*i.e.* orthogonal to $\underline{\tilde{n}}$): see figure IV.D.1.
- $\underline{b}$ is the unit vector completing the direct coordinate system.

In the **two velocity scale model**, we employ the following notations:

- u_k the friction velocity at the wall obtained from the turbulent kinetic energy.
- u^* the wall friction velocity calculated using the relationship given by $\dfrac{u^r_{\tau,I'}}{u^*} = f(z_p)$.
- z_p represents a dimensionless wall distance (this being the distance from the edge of the computational domain), namely $z_p = I'F$ (see figure IV.D.1). The function f expresses the ideal form of the velocity profile. In air, this function is given by a logarithmic law that takes account of the dynamic surface roughness of the wall z_0:
  $f(z_p) = \dfrac{1}{\kappa} ln\left(\dfrac{z_p + z_0}{z_0}\right)$
- The velocity scales, u_k and u^*, are easy to compute, although their computation does require knowledge of the turbulent energy k_I at the centre of the adjoining cell to the boundary face.
- The two-scale model is set as the default model in code_saturne. It frequently enables minimizing, particularly in cases with heat transfer, the effects of certain deficiencies in the $k-\varepsilon$ model (the impacting jet being a classic example).

We will use u^* and u_k later, for the boundary conditions pertaining to the velocity and the scalars (notably, the temperature).

Two-velocity-scale model

$$\begin{cases} u_k = C_\mu^{\frac{1}{4}} k_I^{\frac{1}{2}} \\ u^* \text{solution of } \dfrac{u^r_{\tau,I'}}{u^*} = \dfrac{1}{\kappa} ln(z_k) \\ z_k = \dfrac{I'F + z_0}{z_0} = \dfrac{z_p + z_0}{z_0} \\ \text{with } C_\mu = 0.09 \text{ and } \kappa = 0.42 \end{cases} \tag{IV.D.1}$$

In the case of the one velocity scale model, u^* denotes the only friction velocity at the wall, obtained by solving the equation $\frac{u^r_{\tau,I'}}{u^*} = f(z_p)$. The quantity z_p represents a dimensionless wall distance, given by $z_p = I'F$. The function f reflects the ideal form of the velocity profile as in the case of the two velocity scale model. Note that this friction velocity, the calculation of which is more complicated (fixed point), can nevertheless be obtained without reference to the turbulent variables (k, ε). For convenience, in the one-scale model we write $u_k = u^*$.

We will use u^* and u_k later, for the boundary conditions pertaining to the velocity and the scalars (notably, the temperature).

One velocity scale model

$$\left\{ \begin{array}{l} u_k = u^* \\ u^* \text{solution of } \dfrac{u^r_{\tau,I'}}{u^*} = \dfrac{1}{\kappa} ln(z_k) \\ z_k = \dfrac{I'F + z_0}{z_0} = \dfrac{z_p + z_0}{z_0} \\ \text{with } C_\mu = 0.09 \text{ and } \kappa = 0.42 \end{array} \right. \tag{IV.D.2}$$

• **Boundary conditions for the velocity in $k-\varepsilon$**

We first consider the conditions used in calculations performed with the $k-\varepsilon$ model, as these are the most complex and the more general cases.

The boundary conditions are necessary in order to impose sufficient tangential stress $\sigma_\tau = \rho_I u^* u_k$ at the boundary in the momentum equation[2] (ρ_I is the density at the centre of cell I). The term requiring the boundary conditions is the one that contains the normal derivative of the velocity at the wall, this being[3]: $(\mu_I + \mu_{t,I})\underline{\underline{\text{grad}}}\,\underline{u}\,\underline{n}$. It appears in the second member of the usual momentum equation (see `bilsc2` and `cs_velocity_prediction`).

When the $k-\varepsilon$ model shows a tendency to overestimate the turbulent kinetic energy production, the length scale of the model, $L_{k-\varepsilon}$, can become significantly larger than the maximum theoretical scale of turbulent boundary layer eddies L_{theo}. We write:

$$\left\{ \begin{array}{l} L_{k-\varepsilon} = C_\mu \dfrac{k^{\frac{3}{2}}}{\varepsilon} \\ L_{\text{theo}} = \kappa\ (I'F + z_0) = \kappa\ (z_p + z_0) \end{array} \right. \tag{IV.D.3}$$

If $L_{k-\varepsilon} > L_{\text{theo}}$, we then have $\mu_{t,I} > \mu_t^{lm}$ with $\mu_{t,I}$ the turbulent viscosity of the $k-\varepsilon$ model at point I and $\mu_t^{lm} = \rho_I L_{\text{theo}} u_k$ the turbulent viscosity of the mixing length model. Moreover, the tangential stress can be written with the turbulent viscosity being made to appear:

$$\sigma_\tau = \rho_I u^* u_k = \frac{u^*}{\kappa\ (I'F + z_0)} \underbrace{\rho_I \kappa\ (I'F + z_0)\, u_k}_{\mu_t^{lm}} \tag{IV.D.4}$$

The viscosity scale introduced in this expression of the stress is thus at odds with the one derived from the turbulence computed locally by the model. We therefore prefer to write the stress so that the $k-\varepsilon$ length scale is used whenever it is lower than the limit L_{theo}:

$$\sigma_\tau = \frac{u^*}{\kappa\ (I'F + z_0)} max(\mu_t^{lm}, \mu_{t,I}) \tag{IV.D.5}$$

[2] Proposition de modification des conditions aux limites de paroi turbulente pour le Solveur Commun dans le cadre du modèle $k-\varepsilon$ standard (

Proposed modification of the turbulent wall boundary conditions for the Common Solver in the frame of the standard $k-\varepsilon$ model

), EDF report HI-81/00/019/A, 2000, M. Boucker, J.-D. Mattei.

[3] The transpose gradient term is dealt with in `visecv` and will not be considered here.

This value can then be used for the diffusive flux calculation which is dependent on the stress in the Navier-Stokes equation:

$$(\mu_I + \mu_{t,I})\underline{\underline{\mathrm{grad}}}\,\underline{u}\,\underline{n} = -\sigma_\tau \underline{\tau} \tag{IV.D.6}$$

However, the velocity gradient (gradient at the boundary face) is computed by the code in the following form:

$$(\mu_I + \mu_{t,I})\underline{\underline{\mathrm{grad}}}\,\underline{u}\,\underline{n} = \frac{(\mu_I + \mu_{t,I})}{\overline{I'F}}(\underline{u}_F - \underline{u}_{I'}) \tag{IV.D.7}$$

Drawing on the approximate relationship between (IV.D.6) and (IV.D.7), we derive the value of $\underline{u}_F$ to impose, denoted $\underline{u}_{F,flux}$ (conservation of the momentum flux):

$$\begin{aligned} \underline{u}_{F,flux} &= \underline{u}_{I'} - \frac{\overline{I'F}}{\mu_I + \mu_{t,I}}\sigma_\tau \underline{\tau} \\ &= \underline{u}_{I'} - \frac{u^*}{\kappa\,(\mu_I + \mu_{t,I})} max(\mu_t^{lm}, \mu_{t,I})\,\frac{I'F}{(I'F + z_0)}\underline{\tau} \end{aligned} \tag{IV.D.8}$$

In actual fact, a further approximation is made. This consists in imposing zero normal velocity at the wall and using the equation (IV.D.8) projected onto the plane tangent to the wall:

$$\underline{u}_{F,flux} = \left[u_{\tau,I'} - \frac{u^*}{\kappa\,(\mu_I + \mu_{t,I})} max(\mu_t^{lm}, \mu_{t,I})\,\frac{I'F}{(I'F + z_0)}\right]\underline{\tau} \tag{IV.D.9}$$

Lastly, it is also possible to make the wall velocity appear in the final expression:

Flux-type boundary conditions on the velocity $(k-\varepsilon)$

$$\left\{ \underline{u}_{F,flux} = \underline{v}_p + \left[u^r_{\tau,I'} - \frac{u^*}{\kappa\,(\mu_I + \mu_{t,I})} max(\mu_t^{lm}, \mu_{t,I})\,\frac{I'F}{(I'F + z_0)}\right]\underline{\tau} \right. \tag{IV.D.10}$$

A first pair of coefficients A_{flux} and B_{flux} is thence inferred (separately for each velocity component) and is only used in the calculation of the tangential stress dependent term (see `bilsc2`):

Coefficients associated with the flux-type boundary conditions on the velocity $(k-\varepsilon)$

$$\left\{ \begin{array}{l} \underline{A}_{flux} = \underline{v}_p + \left[u^r_{\tau,I'} - \dfrac{u^*}{\kappa\,(\mu_I + \mu_{t,I})} max(\mu_t^{lm}, \mu_{t,I})\,\dfrac{I'F}{(I'F + z_0)}\right]\underline{\tau} \\ \underline{B}_{flux} = \underline{0} \end{array} \right. \tag{IV.D.11}$$

Having seen above how to impose a boundary condition so as to correctly calculate the stress term, we now need to take a closer look at the calculation of velocity gradients. We are seeking a boundary face value which will allow us to obtain, with the formulation adopted for the gradient, a value of turbulent production as close as possible to the theoretical value (determined using the logarithmic law), in order to evaluate the normal derivative of the tangential velocity. Hence, we define (at point I):

$$P_{\text{theo}} = \rho_I u^* u_k \| \frac{\partial u_\tau}{\partial \underline{n}} \|_I = \rho_I \frac{u_k (u^*)^2}{\kappa\,(I'F + z_0)} \tag{IV.D.12}$$

Moreover, the dominant term of the production computed in cell I for typical situations (where z is the coordinate on the $\underline{\tilde{n}}$ direction vector axis) is:

$$P_{\text{calc}} = \mu_{t,I} \left(\frac{\partial u_\tau}{\partial z}\right)^2_I \tag{IV.D.13}$$

The normal gradient of the tangential velocity (cell gradient) is calculated in the code using finite volumes and takes the following expression on regular orthogonal mesh (see the notations on figure IV.D.2):

$$P_{\text{calc}} = \mu_{t,I} \left(\frac{u_{\tau,G} - u_{\tau,F}}{2d}\right)^2 = \mu_{t,I} \left(\frac{u_{\tau,I} + u_{\tau,J} - 2u_{\tau,F}}{4d}\right)^2 \tag{IV.D.14}$$

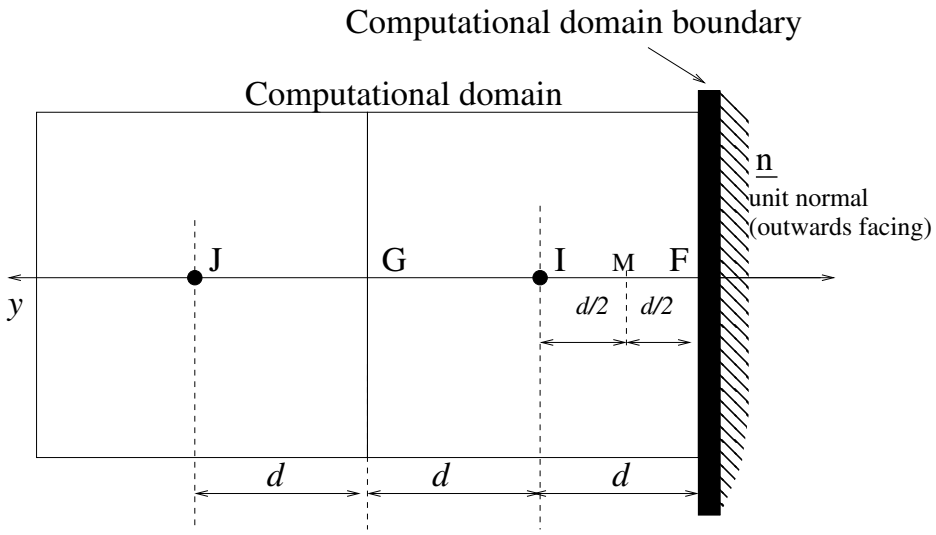


Figure IV.D.2: Boundary cell - Orthogonal mesh.

We then assume that $u_{\tau,J}$ can be obtained from $u_{\tau,I}$ and the normal gradient of u_τ evaluated at G based on the logarithmic law, such that:

$$u_{\tau,J} = u_{\tau,I} + IJ\,.\,(\partial_z u_\tau)_G + \mathcal{O}(IJ^2) \approx u_{\tau,I} + IJ\,.\,\left[\partial_z\left(\frac{u^*}{\kappa}\,ln(z)\right)\right]_G = u_{\tau,I} + 2d\,\frac{u^*}{\kappa\,(2d+z_0)} \quad \text{(IV.D.15)}$$

from which we obtain:

$$\begin{aligned} P_{\text{calc}} &= \mu_{t,I}\left(\frac{2u_{\tau,I} + 2d\,\dfrac{u^*}{\kappa\,(2d+z_0)} - 2u_{\tau,F}}{4d}\right)^2 \\ &= \mu_{t,I}\left(\frac{u_{\tau,I} + d\,\dfrac{u^*}{\kappa\,(2d+z_0)} - u_{\tau,F}}{2d}\right)^2 \end{aligned} \quad \text{(IV.D.16)}$$

We then link equations (IV.D.12) and (IV.D.16) in order to impose the equality of the production calculated and the theoretical turbulent production. Application of the above formulae is extended without caution to non-orthogonal meshes (the velocity at I is simply evaluated at I'). The following expression is then obtained for $u_{\tau,F}$:

$$u_{\tau,F,grad} = u_{\tau,I'} - \frac{u^*}{\kappa}\left(2d\sqrt{\frac{\rho_I \kappa\, u_k}{\mu_{t,I}\,(I'F + z_0)}} - \frac{1}{2 + z_0/I'F}\right) \quad \text{(IV.D.17)}$$

A limit is then set on the gradient so that it remains at least as steep as the gradient obtained from the normal derivative of the theoretical (logarithmic) velocity profile at I':
$\partial_z u_\tau = \partial_z(\frac{u^*}{\kappa}\,ln(z)) = \dfrac{u^*}{\kappa\,(I'F + z_0)}$, thus yielding:

$$u_{\tau,F,grad} = u_{\tau,I'} - \frac{u^*}{\kappa} max\left(1, 2d\sqrt{\frac{\rho_I \kappa\, u_k}{\mu_{t,I}\,(I'F + z_0)}} - \frac{1}{2 + z_0/I'F}\right) \quad \text{(IV.D.18)}$$

A condition of zero normal velocity at the wall is imposed. Moreover, if the tangential velocity at I' is zero (absolute value below an arbitrary numerical limit of 10^{-12}), a no-slip condition is applied. We

can then also make the velocity of the wall appear in the final expression:

Gradient boundary conditions on the velocity$(k-\varepsilon)$

$$\begin{cases} \underline{u}_{F,grad} = \underline{v}_p & \text{if } u^r_{\tau,I'} < 10^{-12} \\ \underline{u}_{F,grad} = \underline{v}_p + u^r_{\tau,I'} - \dfrac{u^*}{\kappa}\left[max\left(1, 2d\sqrt{\dfrac{\rho_I \kappa\, u_k}{\mu_{t,I}\,(I'F + z_0)}} - \dfrac{1}{2 + z_0/I'F}\right)\right]\underline{\tau} \end{cases} \tag{IV.D.19}$$

A second pair of coefficients, A_{grad} and B_{grad}, is thereby inferred (separately for each velocity component) and is used (with the exception of the terms depending on the tangential stress, these being treated in `bilsc2` using the coefficients A_{flux} and B_{flux})whenever the velocity gradient is required:

Coefficients associated with the gradient-type boundary conditions on the velocity $(k-\varepsilon)$

$$\begin{cases} \underline{A}_{grad} = \underline{v}_p & \text{si } u^r_{\tau,I'} < 10^{-12} \\ \underline{A}_{grad} = \underline{v}_p + u^r_{\tau,I'} - \dfrac{u^*}{\kappa}\left[max\left(1, 2d\sqrt{\dfrac{\rho_I \kappa\, u_k}{\mu_{t,I}\,(I'F + z_0)}} - \dfrac{1}{2 + z_0/I'F}\right)\right]\underline{\tau} \\ \underline{B}_{grad} = \underline{0} \end{cases} \tag{IV.D.20}$$

• **Boundary conditions for the k and ε variables (standard $k-\varepsilon$ model)**

We impose a Dirichlet condition derived from the friction velocity u_k (refer to equation (IV.D.1)) on k:

$$k = \frac{u_k^2}{C_\mu^{\frac{1}{2}}} \tag{IV.D.21}$$

We want to set the normal derivative of ε based on the following theoretic law (refer to the notations in figure IV.D.2):

$$G_{\text{theo},\varepsilon} = \frac{\partial}{\partial z}\left[\frac{u_k^3}{\kappa\,(z+z_0)}\right] \tag{IV.D.22}$$

The point M is used to impose a boundary condition with a higher order accuracy in space. In effect, using the simple relation $\varepsilon_F = \varepsilon_I + d\partial_z\varepsilon_I + O(d^2)$ leads to an accuracy of order 1. Second-order accuracy can be obtained using the following truncated series expansions:

$$\left\{\begin{array}{ll} \varepsilon_M & = \varepsilon_I - \dfrac{d}{2}\partial_z\varepsilon_I + \dfrac{d^2}{8}\partial_z^2\varepsilon_I + O(d^3) \\ \varepsilon_M & = \varepsilon_F + \dfrac{d}{2}\partial_z\varepsilon_F + \dfrac{d^2}{8}\partial_z^2\varepsilon_F + O(d^3) \end{array}\right. \tag{IV.D.23}$$

The difference between the above two expansions, can be expressed as

$$\varepsilon_F = \varepsilon_I - \frac{d}{2}(\partial_z\varepsilon_I + \partial_z\varepsilon_F) + O(d^3) \tag{IV.D.24}$$

Moreover, we have

$$\left\{\begin{array}{ll} \partial_z\varepsilon_I & = \partial_z\varepsilon_M + d\partial_z^2\varepsilon_M + O(d^2) \\ \partial_z\varepsilon_F & = \partial_z\varepsilon_M - d\partial_z^2\varepsilon_M + O(d^2) \end{array}\right. \tag{IV.D.25}$$

The sum of the latter two expansions enables us to establish that $\partial_z\varepsilon_I + \partial_z\varepsilon_F = 2\partial_z\varepsilon_M + O(d^2)$ which, when substituted back into (IV.D.24), provides the desired second-order accurate approximation of ε_F:

$$\varepsilon_F = \varepsilon_I - d\partial_z\varepsilon_M + O(d^3) \tag{IV.D.26}$$

The term $\partial_z\varepsilon_M$ is evaluated against the theoretical value (IV.D.22) and the value to impose at the boundary ($d = I'F$) thence obtained:

$$\varepsilon_F = \varepsilon_I + d\frac{u_k^3}{\kappa\,(d/2+z_0)^2} \tag{IV.D.27}$$

This relation is extended without caution for application on non-orthogonal meshes (which is bound to degrade the order of accuracy in space).

We finally obtain the:

Boundary conditions on k and ε

$$\left\{\begin{array}{ll} k_F & = \dfrac{u_k^2}{C_\mu^{\frac{1}{2}}} \\ \varepsilon_F & = \varepsilon_{I'} + I'F\dfrac{u_k^3}{\kappa\,(I'F/2+z_0)^2} \end{array}\right. \tag{IV.D.28}$$

and the associated coefficients

Coefficients associated with the boundary conditions on the variables k and ε

$$\left\{\begin{array}{lll} A_k & = \dfrac{u_k^2}{C_\mu^{\frac{1}{2}}} & \text{et } B_k = 0 \\ A_\varepsilon & = I'F\dfrac{u_k^3}{\kappa\,(I'F/2+z_0)^2} & \text{et } B_\varepsilon = 1 \end{array}\right. \tag{IV.D.29}$$

• **Boundary conditions for the** *VarScalaires*

Only the boundary conditions with a prescribed value (at the wall or in the near-wall region with, eventually, an external transfer coefficient) are addressed herein. We draw on the notations taken from figure IV.D.1 and the general presentation provided in `condli`, the essential elements of which are resumed below.

The conservative form of the normal flux at the boundary for the variable f is written:

$$\underbrace{h_{int}(f_{b,int}-f_{I'})}_{\phi_{int}} = \underbrace{h_b(f_{b,ext}-f_{I'})}_{\phi_b} = \begin{cases} \underbrace{h_{imp,ext}(f_{imp,ext}-f_{b,ext})}_{\phi_{imposed\ real}} & \text{(Dirichlet condition)} \\ \underbrace{\phi_{imp,ext}}_{\phi_{imposed\ real}} & \text{(Neumann condition)} \end{cases} \tag{IV.D.30}$$

These two equations are rearranged in order to obtain the numerical flux value $f_{b,int} = f_F$ to impose at the wall surface, given the values of $f_{imp,ext}$ and $h_{imp,ext}$ set by the user and the value of h_b determined according to the laws of similarity detailed hereafter. Coefficients A and B, are derived directly from the boundary condition, as specified below.

Boundary conditions on the *VarScalaires*

$$f_{b,int} = \underbrace{\frac{h_{imp,ext}}{h_{int}+h_r h_{imp,ext}} f_{imp,ext}}_{A} + \underbrace{\frac{h_{int}+h_{imp,ext}(h_r-1)}{h_{int}+h_r h_{imp,ext}}}_{B} f_{I'} \text{ with } h_r = \frac{h_{int}}{h_b} \tag{IV.D.31}$$

Principle of similarity: calculation of h_b.

The only remaining unknown in the expression (IV.D.31) is h_b, with h_{int} being a numerical value consistent with the calculation method used to compute the face gradients and specified in `condli` ($h_{int} = \frac{\alpha}{\overline{I'F}}$). The value of h_b must relate the flux to the difference between the values of $f_{I'}$ and $f_{b,ext}$ taking the boundary layer into account (the profile of f is not always linear):

$$\phi_b = h_b \left(f_{b,ext} - f_{I'} \right) \tag{IV.D.32}$$

The theoretical considerations and their application are presented below, using the general notations. In particular, the Prandtl-Schmidt number is noted $\sigma = \frac{\nu \rho C}{\alpha}$. When the scalar f under consideration is the temperature, we have (see `condli`):

- $C = C_p$ (specific heat capacity),
- $\alpha = \lambda$ (molecular conductivity),
- $\sigma = \frac{\nu \rho C_p}{\lambda} = Pr$ (Prandtl number),
- $\sigma_t = Pr_t$ (turbulent Prandtl number),
- $\phi = \left(\lambda + \frac{C_p \mu_t}{\sigma_t} \right) \frac{\partial T}{\partial z}$ (flux in $W m^{-2}$).

J. R. Garratt's "The atmospheric boundary layer", Cambridge University Press was used as a supporting reference.

The flux at the wall for the scalar f is written:

$$\phi = - \left(\alpha + C \frac{\mu_t}{\sigma_t} \right) \frac{\partial f}{\partial z} = - \rho C \left(\frac{\alpha}{\rho C} + \frac{\mu_t}{\rho \sigma_t} \right) \frac{\partial f}{\partial z} \tag{IV.D.33}$$

(the flux is positive if it is directed into the fluid domain, as indicated by the orientation of the z axis).

Similarly for the temperature, with $a = \frac{\lambda}{\rho C_p}$ and $a_t = \frac{\mu_t}{\rho \sigma_t}$, we have:

$$\phi = - \rho C_p (a + a_t) \frac{\partial T}{\partial z} \tag{IV.D.34}$$

We introduce f^* in order to make f dimensionless, using the boundary flux value ϕ_b:

$$f^* = - \frac{\phi_b}{\rho C u_k} \tag{IV.D.35}$$

For the temperature, this yields:

$$T^* = - \frac{\phi_b}{\rho C_p u_k} \tag{IV.D.36}$$

As was noted beforehand, in the two-velocity scale model, u_k is the wall friction velocity obtained from the mean kinetic energy of the turbulent motion[4]. For the one velocity scale model, we set $u_k = u^*$ with u^* the wall friction velocity determined from the logarithmic law.

We then divide both sides of equation (IV.D.33) by ϕ_b. Given that the flux is conserved, $\phi = \phi_b$ so the left side of the equation is simplified to 1. On the right-hand side, we replace ϕ_b by its value $-\rho C u_k f^*$. Using the notations:

$$\nu = \frac{\mu}{\rho} \qquad \nu_t = \frac{\mu_t}{\rho} \qquad f^+ = \frac{f - f_{b,ext}}{f^*} \tag{IV.D.37}$$

[4] $u_k = C_\mu^{\frac{1}{4}} k_I^{\frac{1}{2}}$

this provides:

$$1 = \left(\frac{\nu}{\sigma} + \frac{\nu_t}{\sigma_t}\right) \frac{\partial f^+}{\partial z} \frac{1}{u_k} \tag{IV.D.38}$$

With the preceding notations, it can immediately be seen that h_b is expressed as a function of $f^+_{I'}$:

$$h_b = \frac{\phi_b}{f_{b,ext} - f_{I'}} = \frac{\rho\, C\, u_k}{f^+_{I'}} \tag{IV.D.39}$$

To determine h_b, we then integrate equation (IV.D.38) in order to obtain $f^+_{I'}$. The only remaining difficulty consists in establishing the law of variation of $\mathcal{K} = \frac{\nu}{\sigma} + \frac{\nu_t}{\sigma_t}$

In the fully developed turbulent region, a mixing length hypothesis is applied in order to model the variations in ν_t:

$$\nu_t = l^2 |\frac{\partial U}{\partial z}| = \kappa\, u^* \,(z + z_0) \tag{IV.D.40}$$

Moreover, the effects of the molecular diffusion of the scalar f (or of its conductivity when f represents the temperature) are negligible in comparison to the turbulent effects: hence we neglect the contribution of $\frac{\nu}{\sigma}$ with respect to the dominant term $\frac{\nu_t}{\sigma_t}$. This finally leads to:

$$\mathcal{K} = \frac{\kappa\, u_k}{\sigma_t} \,(z + z_0) \tag{IV.D.41}$$

The dimensionless equation (IV.D.38) is integrated under the same hypothesis to obtain the law of variation for f^+, which is given by:

$$f^+ = \frac{\sigma_t}{\kappa}\, ln\left(\frac{z + z_0}{z_{o_T}}\right) \tag{IV.D.42}$$

where z_{o_T} is the thermal roughness length. Its order of magnitude compared to the dynamic roughness length is given by the relation $ln\left(\frac{z_0}{z_{o_T}}\right) \approx 2$ (reference J. R. Garratt).

To summarize, h_b is calculated by determining:

$$f^+_{I'} = \frac{\sigma_t}{\kappa}\, ln\left(\frac{I'F + z_0}{z_{o_T}}\right) \tag{IV.D.43}$$

$$h_b = \frac{\phi_b}{f_{b,ext} - f_{I'}} = \frac{\rho\, C\, u_k}{f^+_{I'}} \tag{IV.D.44}$$

E- clsyvt routine

Function

The aim of this subroutine is to fill the arrays of boundary conditions (COEFA and COEFB) of the velocity and of the Reynolds stress tensor, for the symmetry boundary faces. These conditions are express relatively naturally in the local coordinate system of the boundary face. The function of clsyvt is then to transform these natural boundary conditions (expressed in the local coordinate sytem) in the general coordinate sytem, and then to possibly partly implicit them.

It should be noted that the part of the subroutine clptur (for the wall boundary conditions) relative to the writing in the local coordinate system and to the rotation is totally identical.

See the programmers reference of the dedicated subroutine for further details.

Discretisation

Figure IV.E.1 presents the notations used at the face. The local coordinate system is defined from the normal at the face and the velocity at I':
- $\underline{t} = \dfrac{1}{|\underline{u}_{I',\tau}|}\underline{u}_{I',\tau}$ is the first vector of the local coordinate system.
- $\underline{\tilde{n}} = -\underline{n}$ is the second vector of the local coordinate system.
- $\underline{b} = \underline{t} \wedge \underline{\tilde{n}} = \underline{n} \wedge \underline{t}$ is the third vector of the local coordinate system.

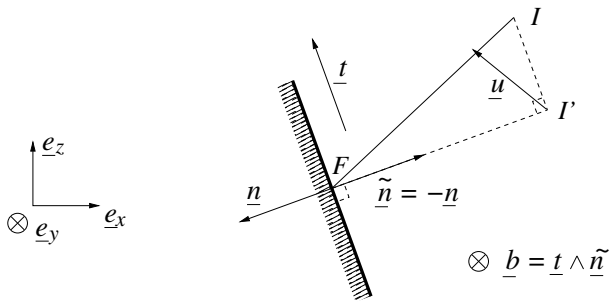


Figure IV.E.1: Definition of the vectors forming the local coordinate system.

Here, $\underline{n}$ is the normalized normal vector to the boundary face in the sense of code_saturne (*i.e.* directed towards the outside of the computational domain) and $\underline{u}_{I',\tau}$ is the projection of the velocity at I' in the plane of the face: $\underline{u}_{I',\tau} = \underline{u}_{I'} - (\underline{u}_{I'}.\underline{n})\underline{n}$.
If $\underline{u}_{I',\tau} = \underline{0}$, the direction of $\underline{t}$ in the plane normal to $\underline{n}$ is irrelevant. thus it is defined as: $\underline{t} = \dfrac{1}{\sqrt{n_y^2 + n_z^2}}(n_z\underline{e}_y - n_y\underline{e}_z)$ where $\underline{t} = \dfrac{1}{\sqrt{n_x^2 + n_z^2}}(n_z\underline{e}_x - n_x\underline{e}_z)$ along the non-zero components of $\underline{n}$ (components in the global coordinate system $(\underline{e}_x, \underline{e}_y, \underline{e}_z)$).

For sake of clarity, the following notations will be used:
- The general coordinate system will write $\mathcal{R} = (\underline{e}_x, \underline{e}_y, \underline{e}_z)$.
- The local coordinate system will write $\hat{\mathcal{R}} = (\underline{t}, -\underline{n}, \underline{b}) = (\underline{t}, \underline{\tilde{n}}, \underline{b})$.
- The matrices of the components of a vector $\underline{u}$ in the coordinate systems $\mathcal{R}$ and $\hat{\mathcal{R}}$ will write $\underline{\mathrm{U}}$ and $\underline{\hat{\mathrm{U}}}$, respectively.

• The matrices of the components of a tensor $\underline{\underline{R}}$ (2^{nd} order) in the coordinate systems $\mathcal{R}$ and $\hat{\mathcal{R}}$ will write $\underline{\underline{\mathrm{R}}}$ and $\underline{\underline{\hat{\mathrm{R}}}}$, respectively.

• $\underline{\underline{\mathrm{P}}}$ refers to the (orthogonal) matrix transforming $\mathcal{R}$ into $\hat{\mathcal{R}}$.

$$\underline{\underline{\mathrm{P}}} = \begin{bmatrix} t_x & -n_x & b_x \\ t_y & -n_y & b_y \\ t_z & -n_z & b_z \end{bmatrix} \tag{IV.E.1}$$

($\underline{\underline{\mathrm{P}}}$ being orthogonal, $\underline{\underline{\mathrm{P}}}^{-1} = {}^t\underline{\underline{\mathrm{P}}}$).

In particular, we have for any vector $\underline{u}$ and for any second order tensor $\underline{\underline{R}}$:

$$\begin{cases} \underline{\mathrm{U}} = \underline{\underline{\mathrm{P}}} \,.\, \underline{\hat{\mathrm{U}}} \\ \underline{\underline{\mathrm{R}}} = \underline{\underline{\mathrm{P}}} \,.\, \underline{\underline{\hat{\mathrm{R}}}} \,.\, {}^t\underline{\underline{\mathrm{P}}} \end{cases} \tag{IV.E.2}$$

Treatment of the velocity

In the local coordinate system, the boundary conditions for $\underline{u}$ naturally write:

$$\begin{cases} u_{F,t} &= u_{I',t} \\ u_{F,\tilde{n}} &= 0 \\ u_{F,b} &= u_{I',b} \end{cases} \tag{IV.E.3}$$

or

$$\underline{\mathrm{U}}_F = \underline{\underline{\mathrm{P}}} \,.\, \underline{\hat{\mathrm{U}}}_F = \underline{\underline{\mathrm{P}}} \,.\, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \,.\, \underline{\hat{\mathrm{U}}}_{I'} = \underline{\underline{\mathrm{P}}} \,.\, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \,.\, {}^t\underline{\underline{\mathrm{P}}} \,.\, \underline{\mathrm{U}}_{I'} \tag{IV.E.4}$$

Let's take $\underline{\underline{\mathrm{A}}} = \underline{\underline{\mathrm{P}}} \,.\, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \,.\, {}^t\underline{\underline{\mathrm{P}}}$ (matrix in the coordinate system $\mathcal{R}$ of the projector orthogonal to the face).

The boundary conditions for $\underline{u}$ thus write:

$$\underline{\mathrm{U}}_F = \underline{\underline{\mathrm{A}}} \,.\, \underline{\mathrm{U}}_{I'} \tag{IV.E.5}$$

Since the matrix $\underline{\underline{\mathrm{P}}}$ is orthogonal, it can be shown that

$$\underline{\underline{\mathrm{A}}} = \begin{bmatrix} 1-\tilde{n}_x^2 & -\tilde{n}_x\tilde{n}_y & -\tilde{n}_x\tilde{n}_z \\ -\tilde{n}_x\tilde{n}_y & 1-\tilde{n}_y^2 & -\tilde{n}_y\tilde{n}_z \\ -\tilde{n}_x\tilde{n}_z & -\tilde{n}_y\tilde{n}_z & 1-\tilde{n}_z^2 \end{bmatrix} \tag{IV.E.6}$$

The boundary conditions can then be partially implicited:

$$u_{F,x}^{(n+1)} = \underbrace{1-\tilde{n}_x^2}_{\texttt{COEFB}} u_{I',x}^{(n+1)} \underbrace{-\tilde{n}_x\tilde{n}_y u_{I',y}^{(n)} - \tilde{n}_x\tilde{n}_z u_{I',z}^{(n)}}_{\texttt{COEFA}} \tag{IV.E.7}$$

The other components have a similar treatment. Since only the coordinates of $\underline{n}$ are useful, we do not need (for $\underline{u}$) to define explicitly the vectors $\underline{t}$ and $\underline{b}$.

Treatment of the Reynolds stress tensor

We saw that we have the following relation:

$$\underline{\underline{\mathrm{R}}} = \underline{\underline{\mathrm{P}}} \,.\, \underline{\underline{\hat{\mathrm{R}}}} \,.\, {}^t\underline{\underline{\mathrm{P}}} \tag{IV.E.8}$$

The boundary conditions we want to write are relations of the type:

$$\mathrm{R}_{F,ij} = \sum_{k,l} \alpha_{ijkl} \mathrm{R}_{I',kl} \tag{IV.E.9}$$

We are then naturally brought to introduce the column matrices of the components of $\underline{\underline{R}}$ in the different coordinate systems.

We write

$$\underline{\mathrm{S}} = {}^t[\mathrm{R}_{11}, \mathrm{R}_{12}, \mathrm{R}_{13}, \mathrm{R}_{21}, \mathrm{R}_{22}, \mathrm{R}_{23}, \mathrm{R}_{31}, \mathrm{R}_{32}, \mathrm{R}_{33}] \tag{IV.E.10}$$

and

$$\underline{\hat{\mathrm{S}}} = {}^t[\hat{\mathrm{R}}_{11}, \hat{\mathrm{R}}_{12}, \hat{\mathrm{R}}_{13}, \hat{\mathrm{R}}_{21}, \hat{\mathrm{R}}_{22}, \hat{\mathrm{R}}_{23}, \hat{\mathrm{R}}_{31}, \hat{\mathrm{R}}_{32}, \hat{\mathrm{R}}_{33}] \tag{IV.E.11}$$

Two functions q and r from $\{1,2,3,4,5,6,7,8,9\}$ to $\{1,2,3\}$ are defined. Their values are given in the following table:

i	1	2	3	4	5	6	7	8	9
$q(i)$	1	1	1	2	2	2	3	3	3
$r(i)$	1	2	3	1	2	3	1	2	3

$i \longmapsto (q(i), r(i))$ is then a bijection from $\{1,2,3,4,5,6,7,8,9\}$ to $\{1,2,3\}^2$, and we have:

$$\left\{ \begin{array}{l} \mathrm{R}_{ij} = \mathrm{S}_{3(i-1)+j} \\ \mathrm{S}_i = \mathrm{R}_{q(i)r(i)} \end{array} \right. \tag{IV.E.12}$$

Using equation IV.E.8, we thus have:

$$\begin{aligned} \mathrm{S}_{F,i} &= \mathrm{R}_{F,q(i)r(i)} = \sum_{(m,n)\in\{1,2,3\}^2} \mathrm{P}_{q(i)m} \hat{\mathrm{R}}_{F,mn} \mathrm{P}_{r(i)n} \\ &= \sum_{j=1}^{9} \mathrm{P}_{q(i)q(j)} \hat{\mathrm{R}}_{F,q(j)r(j)} \mathrm{P}_{r(i)r(j)} \quad \text{(because } i->(q(i),r(i)) \text{ is bijective)} \\ &= \sum_{j=1}^{9} \mathrm{P}_{q(i)q(j)} \mathrm{P}_{r(i)r(j)} \hat{\mathrm{S}}_{F,j} \end{aligned} \tag{IV.E.13}$$

Or

$$\underline{\mathrm{S}}_F = \underline{\underline{\mathrm{A}}} . \underline{\hat{\mathrm{S}}}_F \quad \text{with } \mathrm{A}_{ij} = \mathrm{P}_{q(i)q(j)} \mathrm{P}_{r(i)r(j)} \tag{IV.E.14}$$

It can be shown that $\underline{\underline{\mathrm{A}}}$ is an orthogonal matrix (see Annexe A).

In the local coordinate system, the boundary conditions of $\underline{\underline{R}}$ write naturally[1].

$$\left\{ \begin{array}{lll} \hat{\mathrm{R}}_{F,11} = \hat{\mathrm{R}}_{I',11} & \hat{\mathrm{R}}_{F,21} = 0 & \hat{\mathrm{R}}_{F,31} = B\hat{\mathrm{R}}_{I',31} \\ \hat{\mathrm{R}}_{F,12} = 0 & \hat{\mathrm{R}}_{F,22} = \hat{\mathrm{R}}_{I',22} & \hat{\mathrm{R}}_{F,32} = 0 \\ \hat{\mathrm{R}}_{F,13} = B\hat{\mathrm{R}}_{I',13} & \hat{\mathrm{R}}_{F,23} = 0 & \hat{\mathrm{R}}_{F,33} = \hat{\mathrm{R}}_{I',33} \end{array} \right. \tag{IV.E.15}$$

[1] cf. Davroux A., Archambeau F., *Le $R_{ij} - \varepsilon$ dans code_saturne (version β)*, HI-83/00/030/A

or

$$\hat{\underline{S}}_F = \underline{\underline{B}} \,.\, \hat{\underline{S}}_{I'} \qquad \text{avec } \underline{\underline{B}} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & \ddots & & & & & & \vdots \\ \vdots & \ddots & B & \ddots & & & & & \vdots \\ \vdots & & \ddots & 0 & \ddots & & & & \vdots \\ \vdots & & & \ddots & 1 & \ddots & & & \vdots \\ \vdots & & & & \ddots & 0 & \ddots & & \vdots \\ \vdots & & & & & \ddots & B & \ddots & \vdots \\ \vdots & & & & & & \ddots & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} \tag{IV.E.16}$$

For the symmetry faces which are treated by `clsyvt`, the coefficient B is 1. However a similar treatment is applied in `clptur` for the wall faces, and in this B is zero. This parameter has to be specified when `cs_turbulence_bc_rij_transform` is called (see §E).

Back in the global coordinate system, the following formulae is finally obtained:

$$\underline{S}_F = \underline{\underline{C}} \,.\, \underline{S}_{I'} \qquad \text{avec } \underline{\underline{C}} = \underline{\underline{A}} \,.\, \underline{\underline{B}} \,.\, {}^t\underline{\underline{A}} \tag{IV.E.17}$$

It can be shown that the components of the matrix $\underline{\underline{C}}$ are:

$$C_{ij} = \sum_{k=1}^{9} P_{q(i)q(k)} P_{r(i)r(k)} P_{q(j)q(k)} P_{r(j)r(k)} (\delta_{k1} + B\delta_{k3} + \delta_{k5} + B\delta_{k7} + \delta_{k9}) \tag{IV.E.18}$$

To conclude, it can be noted thatthe tensor $\underline{\underline{R}}$ is symmetric. Thus only the simplified matrices (stored by increasing diagonal) $\underline{S}'$ and $\hat{\underline{S}}'$ will be used:

$$\underline{S}' = {}^t[R_{11}, R_{22}, R_{33}, R_{12}, R_{23}, R_{13}] \tag{IV.E.19}$$

$$\hat{\underline{S}}' = {}^t[\hat{R}_{11}, \hat{R}_{22}, \hat{R}_{33}, \hat{R}_{12}, \hat{R}_{23}, \hat{R}_{13}] \tag{IV.E.20}$$

By gathering different lines of matrix $\underline{\underline{C}}$, equation (IV.E.17) is transformed into the final equation:

$$\underline{S}'_F = \underline{\underline{D}} \,.\, \underline{S}'_{I'} \tag{IV.E.21}$$

The computation of the matrix $\underline{\underline{D}}$ is performed in the subroutine `cs_turbulence_bc_rij_transform`. The methodology is described in annexe B.

From $\underline{\underline{D}}$, the coefficients of the boundary conditions can be partially implicited (`ICLSYR` = 1) or totally explicited (`ICLSYR` = 0).

- Partial implicitation

$$S'_{F,i}{}^{(n+1)} = \underbrace{D_{ii}}_{\texttt{COEFB}} S'_{I',i}{}^{(n+1)} + \underbrace{\sum_{j \neq i} D_{ij} S'_{I',j}{}^{(n)}}_{\texttt{COEFA}} \tag{IV.E.22}$$

- Total explicitation

$$S'_{F,i}{}^{(n+1)} = \underbrace{\sum_{j} D_{ij} S'_{I',j}{}^{(n)}}_{\texttt{COEFA}} \qquad (\texttt{COEFB} = 0) \tag{IV.E.23}$$

Implementation

• Beginning of the loop

Beginnning of the loop on all the boundary faces `IFAC` with symmetry conditions. A face is considered as a symmetry face if `ICODCL(IFAC,IU)` is 4. The tests in `vericl` are designed for `ICODCL` to be equal to 4 for `IU` if and only if it is equal to 4 for the other components of the velocity and for the components of $\underline{\underline{R}}$ (if necessary)
.

The value 0 is given to `ISYMPA`, which identifies the face as a wall or symmetry face (a face where the mass flux will be set to zero as explained in `inimas`).

• Calculation of the basis vectors
The normal vector $\underline{n}$ is stored in `(RNX,RNY,RNZ)`.
$\underline{u}_{I'}$ is calculated in `CONDLI`, passed *via* `COEFU`, and stored in `(UPX,UPY,UPZ)`.

• Case with $R_{ij} - \varepsilon$
With the $R_{ij} - \varepsilon$ model (`ITURB`=30 or 31), the vectors $\underline{t}$ and $\underline{b}$ must be calculated explicitly (we use $\underline{\underline{\mathrm{P}}}$, not simply $\underline{\underline{\mathrm{A}}}$). They are stored in `(TX,TY,TZ)` and `(T2X,T2Y,T2Z)`, respectively.
The transform matrix $\underline{\underline{\mathrm{P}}}$ is then calculated and stored in the array `ELOGLO`.
The subroutine `cs_turbulence_bc_rij_transform` is then called to calculate the reduced matrix $\underline{\underline{\mathrm{D}}}$. It is stored in `ALPHA`. `cs_turbulence_bc_rij_transform` is called with a parameter `CLSYME` which is 1, and which corresponds to the parameter B of equation IV.E.15.

• Filling the arrays `COEFA` and `COEFB`
The arrays `COEFA` and `COEFB` are filled following directly equations IV.E.7, IV.E.22 and IV.E.23.
`RIJIPB(IFAC,.)` corresponds to the vector $\underline{\mathrm{S}}'_{I'}$, computed in `condli`, and passed as an argument to `clsyvt`.

• Filling the arrays `COEFAF` and `COEFBF`
If they are defined, the arrays `COEFAF` and `COEFBF` are filled. They contain the same values as `COEFA` and `COEFB`, respectively.

Annexe A

PROOF OF THE ORTHOGONALITY OF MATRIX $\underline{\underline{\mathrm{A}}}$

All the notations used in paragraphe 2 are kept. We have:

$$\begin{aligned}
({}^t\underline{\underline{\mathrm{A}}}\,.\,\underline{\underline{\mathrm{A}}})_{ij} &= \sum_{k=1}^{9} \mathrm{A}_{ki}\mathrm{A}_{kj} \\
&= \sum_{k=1}^{9} \mathrm{P}_{q(k)q(i)}\mathrm{P}_{r(k)r(i)}\mathrm{P}_{q(k)q(j)}\mathrm{P}_{r(k)r(j)}
\end{aligned} \qquad \text{(IV.E.24)}$$

When k varies from 1 to 3, $q(k)$ remains equal to 1 and $r(k)$ varies from 1 to 3. We thus have:

$$\begin{aligned}\sum_{k=1}^{3} \mathrm{P}_{q(k)q(i)}\mathrm{P}_{r(k)r(i)}\mathrm{P}_{q(k)q(j)}\mathrm{P}_{r(k)r(j)} &= \mathrm{P}_{1q(i)}\mathrm{P}_{1q(j)}\sum_{k=1}^{3}\mathrm{P}_{r(k)r(i)}\mathrm{P}_{r(k)r(j)} \\ &= \mathrm{P}_{1q(i)}\mathrm{P}_{1q(j)}\sum_{k=1}^{3}\mathrm{P}_{kr(i)}\mathrm{P}_{kr(j)} \qquad \text{(IV.E.25)} \\ &= \mathrm{P}_{1q(i)}\mathrm{P}_{1q(j)}\delta_{r(i)r(j)} \qquad \text{(by orthogonality of } \underline{\underline{\mathrm{P}}}\text{)}\end{aligned}$$

Likewise for k varying from 4 to 6 or from 7 to 9, $q(k)$ being 2 or 3, respectively, we obtain:

$$\begin{aligned}({}^{t}\underline{\underline{\mathrm{A}}}\,.\,\underline{\underline{\mathrm{A}}})_{ij} &= \sum_{k=1}^{9}\mathrm{P}_{q(k)q(i)}\mathrm{P}_{r(k)r(i)}\mathrm{P}_{q(k)q(j)}\mathrm{P}_{r(k)r(j)} \\ &= \sum_{k=1}^{3}\mathrm{P}_{kq(i)}\mathrm{P}_{kq(j)}\delta_{r(i)r(j)} \qquad \text{(IV.E.26)} \\ &= \delta_{q(i)q(j)}\delta_{r(i)r(j)} \\ &= \delta_{ij} \qquad \text{(by the bijectivity of } (q,r)\text{)}\end{aligned}$$

Thus ${}^{t}\underline{\underline{\mathrm{A}}}\,.\,\underline{\underline{\mathrm{A}}} = \underline{\underline{\mathrm{Id}}}$. Similarly, it can be shown that $\underline{\underline{\mathrm{A}}}\,.\,{}^{t}\underline{\underline{\mathrm{A}}} = \underline{\underline{\mathrm{Id}}}$. Thus $\underline{\underline{\mathrm{A}}}$ is an orthogonal matrix.

Annexe B

Calculation of the matrix $\underline{\underline{\mathrm{D}}}$

The relation between the matrices of dimension 9×1 of the components of $\underline{R}$ in the coordinate system $\mathcal{R}$ at F and at I' (matrices $\underline{\mathrm{S}}_F$ and $\underline{\mathrm{S}}_{I'}$):

$$\underline{\mathrm{S}}_F = \underline{\underline{\mathrm{C}}}\,.\,\underline{\mathrm{S}}_{I'} \qquad \text{(IV.E.27)}$$

with

$$\mathrm{C}_{ij} = \sum_{k=1}^{9}\mathrm{P}_{q(i)q(k)}\mathrm{P}_{r(i)r(k)}\mathrm{P}_{q(j)q(k)}\mathrm{P}_{r(j)r(k)}(\delta_{k1} + B\delta_{k3} + \delta_{k5} + B\delta_{k7} + \delta_{k9}) \qquad \text{(IV.E.28)}$$

To transform $\underline{\mathrm{S}}$ into the matrix 6×1 $\underline{\mathrm{S}}'$, the function s from $\{1,2,3,4,5,6,7,8,9\}$ to $\{1,2,3,4,5,6\}$ is defined. It takes the following values:

i	1	2	3	4	5	6	7	8	9
$s(i)$	1	4	6	4	2	5	6	5	3

By construction, we have $\mathrm{S}_i = \mathrm{S}'_{s(i)}$ for all i between 1 and 9.

To compute D_{ij}, we can choose a value of m to satisfy $s(m) = i$ and sum all the C_{mn} to have $s(n) = j$. The choice of m is irrelevent. We can also compute the sum over all m so that $s(m) = i$ and then divide by the number of such values of m. We will use this method.

We define $N(i)$ the number of integers m between 1 and 9 so that $s(m) = i$ ($N(i)$ is equal to 1 or 2). According to what preceeds we have

$$\begin{aligned} \mathrm{D}_{ij} &= \frac{1}{N(i)} \sum_{\substack{s(m)=i \\ s(n)=j}} \mathrm{C}_{mn} \\ &= \frac{1}{N(i)} \sum_{\substack{s(m)=i \\ s(n)=j \\ 1 \leqslant k \leqslant 9}} \mathrm{P}_{q(m)q(k)}\mathrm{P}_{r(m)r(k)}\mathrm{P}_{q(n)q(k)}\mathrm{P}_{r(n)r(k)}(\delta_{k1} + B\delta_{k3} + \delta_{k5} + B\delta_{k7} + \delta_{k9}) \end{aligned} \tag{IV.E.29}$$

• First case: $i \leqslant 3$ and $j \leqslant 3$
In this case, we have $N(i) = N(j) = 1$. Additionally, if $s(m) = i$ and $s(n) = j$, then $q(m) = r(m) = i$ and $q(n) = r(n) = j$. Thus

$$\mathrm{D}_{ij} = \sum_{k=1}^{9} \mathrm{P}_{iq(k)}\mathrm{P}_{ir(k)}\mathrm{P}_{jq(k)}\mathrm{P}_{jr(k)}(\delta_{k1} + B\delta_{k3} + \delta_{k5} + B\delta_{k7} + \delta_{k9}) \tag{IV.E.30}$$

When k belongs to $\{1, 5, 9\}$, $q(k) = r(k)$ belongs to $\{1, 2, 3\}$. And for $k = 3$ or $k = 7$, $q(k) = 1$ and $r(k) = 3$, or the inverse (and for k even the the sum of Kronecker symbol is zero). Finally we have:

$$\mathrm{D}_{ij} = \sum_{k=1}^{3} \mathrm{P}_{ik}^2\mathrm{P}_{jk}^2 + 2B\mathrm{P}_{j1}\mathrm{P}_{i3}\mathrm{P}_{i1}\mathrm{P}_{j3} \tag{IV.E.31}$$

• Second case: $i \leqslant 3$ and $j \geqslant 4$
Again we have $N(i) = 1$, and if $s(m) = i$ then $q(m) = r(m) = i$.
On the contrary, we have $N(j) = 2$, the two possibilities being m_1 and m_2.

- if $j = 4$, then $m_1 = 2$ and $m_2 = 4$, $q(m_1) = r(m_2) = 1$ and $r(m_1) = q(m_2) = 2$. We then have $m = 1$ and $n = 2$.
- if $j = 5$, then $m_1 = 6$ and $m_2 = 8$, $q(m_1) = r(m_2) = 1$ and $r(m_1) = q(m_2) = 3$. We then have $m = 1$ and $n = 3$.
- if $j = 6$, then $m_1 = 3$ and $m_2 = 7$, $q(m_1) = r(m_2) = 2$ and $r(m_1) = q(m_2) = 3$. We then have $m = 2$ and $n = 3$.

And we have:

$$\mathrm{D}_{ij} = \sum_{k=1}^{9} \mathrm{P}_{iq(k)}\mathrm{P}_{ir(k)} \left[\mathrm{P}_{mq(k)}\mathrm{P}_{nr(k)} + \mathrm{P}_{nq(k)}\mathrm{P}_{mr(k)}\right] (\delta_{k1} + B\delta_{k3} + \delta_{k5} + B\delta_{k7} + \delta_{k9}) \tag{IV.E.32}$$

But when k is in $\{1, 5, 9\}$, $q(k) = r(k)$ is in $\{1, 2, 3\}$. Thus:

$$\mathrm{D}_{ij} = 2\sum_{k=1}^{3} \mathrm{P}_{ik}^2\mathrm{P}_{mk}\mathrm{P}_{nk} + B\sum_{k=1}^{9} \mathrm{P}_{iq(k)}\mathrm{P}_{ir(k)} \left[\mathrm{P}_{mq(k)}\mathrm{P}_{nr(k)} + \mathrm{P}_{nq(k)}\mathrm{P}_{mr(k)}\right] (\delta_{k3} + \delta_{k7}) \tag{IV.E.33}$$

And for $k = 3$ or $k = 7$, $q(k) = 1$ and $r(k) = 3$, or the opposite. We finally have:

$$\mathrm{D}_{ij} = 2\left[\sum_{k=1}^{3} \mathrm{P}_{ik}^2\mathrm{P}_{mk}\mathrm{P}_{nk} + B\mathrm{P}_{i1}\mathrm{P}_{i3}\left(\mathrm{P}_{m1}\mathrm{P}_{n3} + \mathrm{P}_{n1}\mathrm{P}_{m3}\right)\right] \tag{IV.E.34}$$

with $(m,n) = (1,2)$ if $j = 4$, $(m,n) = (1,3)$ if $j = 5$ and $(m,n) = (2,3)$ if $j = 6$.

• THIRD CASE: $i \geqslant 4$ and $j \leqslant 3$
By symmetry of $\underline{\underline{\mathrm{C}}}$, we obtain a result which is the symmetric of the second case, except that $N(i)$ is now 2. Thus:

$$\mathrm{D}_{ij} = \sum_{k=1}^{3} \mathrm{P}_{jk}^2 \mathrm{P}_{mk}\mathrm{P}_{nk} + B\mathrm{P}_{j1}\mathrm{P}_{j3}\left(\mathrm{P}_{m1}\mathrm{P}_{n3} + \mathrm{P}_{n1}\mathrm{P}_{m3}\right) \tag{IV.E.35}$$

with $(m,n) = (1,2)$ if $i = 4$, $(m,n) = (2,3)$ if $i = 5$ and $(m,n) = (1,3)$ if $i = 6$.

• FOURTH CASE: $i \geqslant 4$ and $j \geqslant 4$
Then $N(i) = N(j) = 2$.
We have $(m,n) = (1,2)$ if $i = 4$, $(m,n) = (2,3)$ if $i = 5$ and $(m,n) = (1,3)$ if $i = 6$. Likewise we define m' and n' as a function of j. We then obtain:

$$\begin{aligned} \mathrm{D}_{ij} &= \frac{1}{2}\sum_{k=1}^{9}\left(\mathrm{P}_{mq(k)}\mathrm{P}_{nr(k)} + \mathrm{P}_{nq(k)}\mathrm{P}_{mr(k)}\right)\left(\mathrm{P}_{m'q(k)}\mathrm{P}_{n'r(k)} + \mathrm{P}_{n'q(k)}\mathrm{P}_{m'r(k)}\right) \\ &\qquad \times(\delta_{k1} + B\delta_{k3} + \delta_{k5} + B\delta_{k7} + \delta_{k9}) \\ &= \frac{1}{2}\left[\sum_{k=1}^{3} 4\mathrm{P}_{mk}\mathrm{P}_{nk}\mathrm{P}_{m'k}\mathrm{P}_{n'k} + 2B\left(\mathrm{P}_{m1}\mathrm{P}_{n3} + \mathrm{P}_{n1}\mathrm{P}_{m3}\right)\left(\mathrm{P}_{m'1}\mathrm{P}_{n'3} + \mathrm{P}_{n'1}\mathrm{P}_{m'3}\right)\right] \end{aligned} \tag{IV.E.36}$$

and finally:

$$\mathrm{D}_{ij} = 2\sum_{k=1}^{3} \mathrm{P}_{mk}\mathrm{P}_{nk}\mathrm{P}_{m'k}\mathrm{P}_{n'k} + B\left(\mathrm{P}_{m1}\mathrm{P}_{n3} + \mathrm{P}_{n1}\mathrm{P}_{m3}\right)\left(\mathrm{P}_{m'1}\mathrm{P}_{n'3} + \mathrm{P}_{n'1}\mathrm{P}_{m'3}\right) \tag{IV.E.37}$$

with $(m,n) = (1,2)$ if $i = 4$, $(m,n) = (2,3)$ if $i = 5$ and $(m,n) = (1,3)$ if $i = 6$
and $(m',n') = (1,2)$ if $j = 4$, $(m',n') = (2,3)$ if $j = 5$ and $(m',n') = (1,3)$ if $j = 6$.

F- codits routine

Function

This subroutine, called by `cs_velocity_prediction`, `cs_turbulence_ke`, `cs_solve_equation_scalar`, `resrij`, `reseps`, amongst others, solves the convection-diffusion equations of a given scalar a with source terms of the type:

$$\begin{aligned} & f_s^{imp}(a^{n+1}-a^n)+\theta \underbrace{\mathrm{div}\left((\rho\underline{u})\,a^{n+1}\right)}_{\text{implicit convection}} -\theta \underbrace{\mathrm{div}\left(\mu_{tot}\,\underline{\nabla} a^{n+1}\right)}_{\text{implicit diffusion}} \\ & = f_s^{exp} -(1-\theta) \underbrace{\mathrm{div}\left((\rho\underline{u})\,a^{n}\right)}_{\text{explicit convection}} +(1-\theta) \underbrace{\mathrm{div}\left(\mu_{tot}\,\underline{\nabla} a^{n}\right)}_{\text{explicit diffusion}} \end{aligned} \tag{IV.F.1}$$

where $\rho\underline{u}$, f_s^{exp} and f_s^{imp} denote respectively the mass flux, the explicit source terms and the terms linearized in a^{n+1}. a is a scalar defined on all cells[1]. For clarification, unless stated otherwise it is assumed that the physical properties Φ (total viscosity μ_{tot},...) are evaluated at time $n+\theta_\Phi$ and the mass flux $(\rho\underline{u})$ at time $n+\theta_F$, with θ_Φ and θ_F dependent on the specific time-integration schemes selected for the respective quantities[2].
The discretisation of the convection and diffusion terms in non-orthogonal grids creates numerical difficulties, notably with respect to the reconstruction terms and the slope test. These are circumvented by using an iterative process for which the limit, if there is one, is the solution of the previous equation.

See the programmers reference of the dedicated subroutine for further details.

Discretisation

In order to explain the procedure employed to address the computational problems, owing to the use of reconstruction terms and the slope (or gradient) test, in treating the convection-diffusion terms, we denote by $\mathcal{E}_n$ (similarly to the definition given in `cs_solve_navier_stokes` though without the associated spatial discretisation) the operator:

$$\begin{aligned} \mathcal{E}_n(a) = & f_s^{imp}\,a+\theta\ \mathrm{div}\left((\rho\underline{u})\,a\right)-\theta\ \mathrm{div}\left(\mu_{tot}\,\underline{\nabla} a\right) \\ & -f_s^{exp}-f_s^{imp}\,a^n+(1-\theta)\ \mathrm{div}\left((\rho\underline{u})\,a^n\right)-(1-\theta)\ \mathrm{div}\left(\mu_{tot}\,\underline{\nabla} a^n\right) \end{aligned} \tag{IV.F.2}$$

Hence, equation (IV.F.1) is written:

$$\mathcal{E}_n(a^{n+1})=0 \tag{IV.F.3}$$

The quantity $\mathcal{E}_n(a^{n+1})$ thus comprises:

$\rightsquigarrow f_s^{imp}\,a^{n+1}$, contribution of the linear, zeroth-order differential terms in a^{n+1},

$\rightsquigarrow \theta\ \mathrm{div}\left((\rho\underline{u})\,a^{n+1}\right)-\theta\ \mathrm{div}\left(\mu_{tot}\,\underline{\nabla} a^{n+1}\right)$, full implicit convection-diffusion terms (terms without flux reconstruction + reconstructed terms), linear[3] in a^{n+1},

$\rightsquigarrow f_s^{exp}-f_s^{imp}\,a^n$ et $(1-\theta)\ \mathrm{div}\left((\rho\underline{u})\,a^n\right)-(1-\theta)\ \mathrm{div}\left(\mu_{tot}\,\underline{\nabla} a^n\right)$ all of the explicit terms (including the explicit parts obtained from the time scheme applied to the convection diffusion equation).

[1] a, in spatially discrete form, corresponds to a vector sized to `NCELET` of component a_I, I describing all of the cells.
[2] cf. `introd`
[3] During the spatial discretisation, the linearity of these terms could however be lost, notably through the use of the slope test.

Likewise, we introduce an operator $\mathcal{EM}_n$ that is close to $\mathcal{E}_n$, linear and simple to invert, such that its expression contains:

- $\rightsquigarrow$ the consideration of the linear terms in a,
- $\rightsquigarrow$ the convection integrated with a first-order upwind scheme in space,
- $\rightsquigarrow$ the diffusive flux without reconstruction.

$$\mathcal{EM}_n(a) = f_s^{imp}\, a + \theta\, [\mathrm{div}\,((\rho \underline{u})\, a)]^{upwind} - \theta\, [\mathrm{div}\,(\mu_{\,tot}\, \underline{\nabla} a)]^{N\ Rec} \tag{IV.F.4}$$

This operator helps circumvent the difficulty caused by the presence of nonlinearities (which may be introduced if the slope test is activated in conjunction with the convection scheme) and the high fill-in that occurs in the matrix structure owing to the presence of reconstruction gradients.
We have, for each cell Ω_I of centre I, the following relation[4]:

$$\mathcal{EM}_{disc}(a, I) = \int_{\Omega_i} \mathcal{EM}_n(a)\, d\Omega$$

and want to solve the following problem:

$$0 = \mathcal{E}_n(a^{n+1}) = \mathcal{EM}_n(a^{n+1}) + \mathcal{E}_n(a^{n+1}) - \mathcal{EM}_n(a^{n+1}) \tag{IV.F.5}$$

Hence:

$$\mathcal{EM}_n(a^{n+1}) = \mathcal{EM}_n(a^{n+1}) - \mathcal{E}_n(a^{n+1}) \tag{IV.F.6}$$

In order to do so, we will use a fixed-point algorithm, defining the sequence $(a^{n+1,\,k})_{k\in\mathbb{N}}$[5]:

$$\left\{ \begin{array}{l} a^{n+1,\,0} = a^n \\ a^{n+1,\,k+1} = a^{n+1,\,k} + \delta a^{n+1,\,k+1} \end{array} \right.$$

where $\delta a^{n+1,\,k+1}$ is the solution of:

$$\mathcal{EM}_n(a^{n+1,\,k} + \delta a^{n+1,\,k+1}) = \mathcal{EM}_n(a^{n+1,\,k}) - \mathcal{E}_n(a^{n+1,\,k}) \tag{IV.F.7}$$

Which, by linearity of $\mathcal{EM}_n$, simplifies out to:

$$\mathcal{EM}_n(\delta a^{n+1,\,k+1}) = -\mathcal{E}_n(a^{n+1,\,k}) \tag{IV.F.8}$$

By applying this sequence with the selected operator $\mathcal{E}_n$, we are able to overcome the computational difficulty that arises when dealing with the convection (discretised using numerical schemes that can lead to the introduction of nonlinearities) and/or reconstruction terms. The user-specified scheme for the convection (which may be nonlinear should the slope test be activated) as well as the reconstruction terms will be evaluated at the iteration k and treated on the right-hand side *via* the subroutine `bilsc2`, while the non-reconstructed terms are taken at iteration $k+1$ and hence represent the unknowns of the linear system solved by `codits`[6].

We assume moreover that the sequence $(a^{n+1,\,k})_k$ converges to the solution a^{n+1} of equation (IV.F.2), *i.e.* $\lim\limits_{k\to\infty} \delta a^{n+1,\,k} = 0$, for any given value of n.
The equation solved by `codits` is the abovementioned (IV.F.8). The matrix $\underline{\underline{EM}}_n$, which is associated to the operator $\mathcal{EM}_n$, has to be inverted; the nonlinear terms are placed on the right-hand side, but in explicit form (index k of $a^{n+1,\,k}$) and thus cease to pose a problem.

Remark 1
The total viscosity $\mu_{\,tot}$ considered in $\mathcal{EM}_n$ and in $\mathcal{E}_n$ is dependent on the turbulence model used.

[4] For further details regarding $\mathcal{EM}_{disc}$, the discrete operator acting on a scalar a, the reader can refer to the subroutine `matrix`.

[5] If the fixed-point iterative algorithm is used to solve the velocity-pressure system (`NTERUP`>1), $a^{n+1,0}$ is initialised by the last value that was obtained for a^{n+1}.

[6] cf. the subroutine `cs_solve_navier_stokes`.

More specifically, the viscous term in $\mathcal{EM}_n$ and in $\mathcal{E}_n$ is, as a general rule, taken as being $\mu_{tot} = \mu_{laminar} + \mu_{turbulent}$, the exception being when an $R_{ij} - \varepsilon$ model is used, in which case $\mu_{tot} = \mu_{laminar}$. The choice of $\mathcal{EM}_n$ being *a priori* arbitrary, ($\mathcal{EM}_n$ has to be linear and the sequence $(a^{n+1,\,k})_{k\in\mathbb{N}}$ must converge for any given n), one of the options in the $R_{ij} - \varepsilon$ models (`IRIJNU` $= 1$) consists in forcing the viscosity μ^n_{tot}, that appears in the expression of $\mathcal{EM}_n$ to take the value $\mu^n_{laminar} + \mu^n_{turbulent}$ when `codits` is called by the subroutine `cs_solve_navier_stokes` for the velocity prediction step. There is no physical reason for doing so (only the laminar viscosity $\mu^n_{laminar}$ is supposed to appear), however this can have a stabilising effect in certain cases without there being any concomitant effect on the limit-values of the sequence $(a^{n+1,\,k})_k$.

Remark 2
When `codits` is used for strengthened transient velocity-pressure coupling (`IPUCOU`=1), a single iteration k is made by initialising the sequence $(a^{n+1,\,k})_{k\in\mathbb{N}}$ to zero. The Dirichlet type boundary conditions are cancelled (we have `INC` $= 0$) and the right-hand side is equal to $\rho|\Omega_i|$, which allows us to obtain a diagonal-type approximation of $\underline{\underline{EM}}_n$, needed for the velocity correction step [7].

Implementation

The algorithm for this subroutine is as follows:
- determination of the properties of the $\underline{\underline{EM}}_n$ matrix (symmetric if there is no convection, asymmetric otherwise)
- automatic selection of the solution method for its inversion if the user has not already specified one for the variable being treated. The Jacobi method is used by default for every convected scalar variable a. The methods available are the conjugate gradient method, Jacobi's and the bi-conjugate gradient stabilised method ($BICGStab$) for asymmetric matrices. Diagonal pre-conditioning can be implemented and is used by default for all of these solvers except for Jacobi.
- consideration of the periodicity (translation or rotation of a scalar, vector or tensor),
- construction of the $\underline{\underline{EM}}_n$ matrix corresponding to the linear operator $\mathcal{EM}_n$ through a call to the subroutine `matrix`[8]. The implicit terms corresponding to the diagonal part of the matrix and hence to the zeroth-order linear differential contributions in a^{n+1},(*i.e.* f_s^{imp}), are stored in the array `ROVSDT` (realized before the subroutine calls `codits`).
- creation of the grid hierarchy if the multigrid algorithm is used (`IMGRP` > 0).
- call to `bilsc2` to take the explicit convection-diffusion into account should $\theta \neq 0$.
- loop over the number of iterations from 1 to `NSWRSM` (which is called `NSWRSP` in `codits`). The iterations are represented by k, which is designated `ISWEEP` in the code and defines the indices of the sequence $(a^{n+1,\,k})_k$ and of $(\delta a^{n+1,\,k})_k$.

The right-hand side is split into two parts:

■ a term that is affine in $a^{n+1,\,k-1}$, easily updated during the course of the incremental-iterative resolution procedure, which is written as:

$$-f_s^{imp}\left(a^{n+1,\,k-1} - a^{n+1,0}\right) + f_s^{exp} - (1-\theta)\left[\operatorname{div}\left((\rho\underline{u})\,a^{n+1,0}\right) - \operatorname{div}\left(\mu_{tot}\,\underline{\nabla} a^{n+1,0}\right)\right]$$

■ the terms arising from the convection/diffusion computation (with reconstruction) performed by `bilsc2`.

$$-\theta\left[\operatorname{div}\left((\rho\underline{u})\,a^{n+1,\,k-1}\right) - \operatorname{div}\left(\mu_{tot}\,\underline{\nabla} a^{n+1,\,k-1}\right)\right]$$

The loop in k is then the following:

[7] cf. subroutine `resopv`.
[8] Remember that in `matrix`, regardless of the user's choice, a first-order in space upwind scheme is always used to treat the convection and there is no reconstruction for the diffusive flux. The user's choice of numerical scheme for the convection is only applied for the integration of the convection terms of $\mathcal{E}_n$, on the right-hand side of (IV.F.8), which computed in the subroutine `bilsc2`.

- Computation of the right-hand side (second member) of the equation, without the contribution of the explicit convection-diffusion terms SMBINI; as for the whole right-hand side corresponding to $\mathcal{E}_n(a^{n+1,\,k-1})$, it is stored in the array SMBRP, initialised by SMBINI and completed with the reconstructed convection-diffusion terms by a call to the subroutine bilsc2.
At iteration k, SMBINI noted $\texttt{SMBINI}^{\,k}$ is equal to:

$$\texttt{SMBINI}^{\,k} = f_s^{exp} - (1-\theta)\,\left[\,\mathrm{div}\,((\rho\underline{u})\,a^n) - \mathrm{div}\,(\mu_{\,tot}\,\underline{\nabla} a^n)\,\right] - \,f_s^{imp}(\,a^{n+1,\,k-1} - a^n\,)$$

• Before starting the loop over k, a first call to the subroutine bilsc2 with THETAP $= 1-\theta$ serves to take the explicit part (from the time advancement scheme) of the convection-diffusion terms into account.

$$\texttt{SMBRP}^{\,0} = f_s^{exp} - (1-\theta)\,\left[\,\mathrm{div}\,((\rho\underline{u})\,a^n) - \mathrm{div}\,(\mu_{\,tot}\,\underline{\nabla} a^n)\,\right]$$

Similarly, before looping on k, the right-hand side $\texttt{SMBRP}^{\,0}$ is stored in the array $\texttt{SMBINI}^{\,0}$ and serves to initialize the computation.

$$\texttt{SMBINI}^{\,0} = \texttt{SMBRP}^{\,0}$$

• for $k = 1$,

$$\begin{aligned}\texttt{SMBINI}^{\,1} &= f_s^{exp} - (1-\theta)\,\left[\,\mathrm{div}\,((\rho\underline{u})\,a^n) - \mathrm{div}\,(\mu_{\,tot}\,\underline{\nabla} a^n)\,\right] - \,f_s^{imp}(\,a^{n+1,\,0} - a^n\,)\\ &= f_s^{exp} - (1-\theta)\,\left[\,\mathrm{div}\,((\rho\underline{u})\,a^{n+1,\,0}) - \mathrm{div}\,(\mu_{\,tot}\,\underline{\nabla} a^{n+1,\,0})\,\right] - f_s^{imp}\,\delta a^{n+1,\,0}\end{aligned}$$

This calculation is thus represented by a corresponding sequence of operations on the arrays:

$$\texttt{SMBINI}^{\,1} \;=\; \texttt{SMBINI}^{\,0} - \texttt{ROVSDT} * (\,\texttt{PVAR} - \texttt{PVARA})$$

and $\texttt{SMBRP}^{\,1}$ is completed by a second call to the subroutine bilsc2 with THETAP $= \theta$, so that the implicit part of the convection-diffusion computation is added to the right-hand side.

$$\begin{aligned}\texttt{SMBRP}^{\,1} &= \texttt{SMBINI}^{\,1} - \theta\,\left[\,\mathrm{div}\,((\rho\underline{u})\,a^{n+1,\,0}) - \mathrm{div}\,(\mu_{\,tot}\,\underline{\nabla} a^{n+1,\,0})\,\right]\\ &= f_s^{exp} \;- (1-\theta)\,\left[\,\mathrm{div}\,((\rho\underline{u})\,a^n) - \mathrm{div}\,(\mu_{\,tot}\,\underline{\nabla} a^n)\,\right] - f_s^{imp}\,(a^{n+1,\,0} - a^n)\\ &\quad -\theta\,\left[\,\mathrm{div}\,((\rho\underline{u})\,a^{n+1,\,0}) - \mathrm{div}\,(\mu_{\,tot}\,\underline{\nabla} a^{n+1,\,0})\,\right]\end{aligned}$$

• for $k = 2$,
In a similar fashion, we obtain:

$$\texttt{SMBINI}^{\,2} = f_s^{exp} - (1-\theta)\,\left[\,\mathrm{div}\,((\rho\underline{u})\,a^n) - \mathrm{div}\,(\mu_{\,tot}\,\underline{\nabla} a^n)\,\right] - \,f_s^{imp}\,(\,a^{n+1,\,1} - a^n\,)$$

Hence, the equivalent array formula:

$$\texttt{SMBINI}^{\,2} \;=\; \texttt{SMBINI}^{\,1} - \texttt{ROVSDT} * \texttt{DPVAR}^{\,1}$$

the call to the subroutine bilsc2 being systematically made thereafter with THETAP $= \theta$, we likewise obtain:

$$\texttt{SMBRP}^{\,2} \;=\; \texttt{SMBINI}^{\,2} - \theta\left[\mathrm{div}\,\left((\rho\underline{u})\,a^{n+1,\,1}\right) - \mathrm{div}\,\left(\mu_{\,tot}\,\underline{\nabla}\,a^{n+1,\,1}\right)\right]$$

where

$$a^{n+1,\,1} = \texttt{PVAR}^{\,1} = \texttt{PVAR}^{\,0} + \texttt{DPVAR}^{\,1} = a^{n+1,\,0} + \delta a^{n+1,\,1}$$

• for iteration $k+1$,
The array $\texttt{SMBINI}^{\,k+1}$ initialises the entire right side $\texttt{SMBRP}^{\,k+1}$ to which will be added the convective and diffusive contributions *via* the subroutine bilsc2.
The array formula is given by:

$$\texttt{SMBINI}^{\,k+1} \;=\; \texttt{SMBINI}^{\,k} - \texttt{ROVSDT} * \texttt{DPVAR}^{\,k}$$

Then follows the computation and the addition of the reconstructed convection-diffusion terms of $-\mathcal{E}_n(a^{n+1,\,k})$, by a call to the `bilsc2` subroutine. Remember that the convection is taken into account at this step by the user-specified numerical scheme (first-order accurate in space upwind scheme, centred scheme with second-order spatial discretisation, second-order linear upwind "SOLU" scheme or a weighted average (blending) of one of the second-order schemes (either centred or SOLU) and the first-order upwind scheme, with potential use of a slope test).
This contribution (convection-diffusion) is then added in to the right side of the equation $\texttt{SMBRP}^{\,k+1}$ (initialised by $\texttt{SMBINI}^{\,k+1}$).

$$\begin{array}{ll}\texttt{SMBRP}^{\,k+1} & = \texttt{SMBINI}^{\,k+1} - \theta\,\left[\,\mathrm{div}\,\left((\rho\underline{u})\,a^{n+1,\,k}\right) - \mathrm{div}\,\left(\mu_{\,tot}\,\underline{\nabla}a^{n+1,\,k}\right)\right] \\ & = f_s^{\,exp} - (1-\theta)\,\left[\,\mathrm{div}\,\left((\rho\underline{u})\,a^{n}\right) - \mathrm{div}\,\left(\mu_{\,tot}\,\underline{\nabla}a^{n}\right)\right] - f_s^{\,imp}\,(a^{n+1,\,k} - a^n) \\ & -\theta\,\left[\,\mathrm{div}\,\left((\rho\underline{u})\,a^{n+1,k}\right) - \mathrm{div}\,\left(\mu_{\,tot}\,\underline{\nabla}a^{n+1,k}\right)\right]\end{array}$$

- Resolution of the linear system in $\delta a^{n+1,\,k+1}$ corresponding to equation (IV.F.8) by inversion of the $\underline{\underline{EM}}_n$ matrix, through a call to the subroutine `invers`. We calculate $a^{n+1,\,k+1}$ based on the formula:

  $$a^{n+1,\,k+1} = a^{n+1,\,k} + \delta a^{n+1,\,k+1}$$

  Represented in array form by:

  $$\texttt{PVAR}^{\,k+1} = \texttt{PVAR}^{\,k} + \texttt{DPVAR}^{\,k+1}$$

- Treatment of parallelism and of the periodicity.

- Test of convergence:
  The test involves the quantity $||\texttt{SMBRP}^{\,k+1}|| < \varepsilon||\underline{\underline{EM}}_n(a^n) + \texttt{SMBRP}^{\,1}||$, where $||\,.\,||$ denotes the Euclidean norm. The solution sought is $a^{\,n+1} = a^{n+1,\,k+1}$. If the test is satisfied, then convergence has been reached and we exit the iteration loop.
  If not, we continue to iterate until the upper limit of iterations imposed by `NSWRSM` in `usini1` is reached.
  The condition for convergence is also written, in continuous form, as:

  $$\begin{array}{ll}||\texttt{SMBRP}^{\,k+1}|| & < \varepsilon||f_s^{\,exp} - \mathrm{div}\,\left((\rho\underline{u})\,a^n\right) + \mathrm{div}\,\left(\mu_{\,tot}\,\underline{\nabla}a^n\right) \\ & +[\mathrm{div}\,\left((\rho\underline{u})\,a^n\right)]^{amont} + [\mathrm{div}\,\left(\mu_{\,tot}\,\underline{\nabla}a^n\right)]^{N\ Rec}||\end{array}$$

  As a consequence, on orthogonal mesh with an upwind convection scheme and in the absence of source terms, the sequence converges in theory in a single iteration because, by construction:

  $$||\texttt{SMBRP}^{\,2}|| = 0 \quad < \varepsilon||f_s^{\,exp}||$$

End of the loop.

Points to treat

• Approximation $\mathcal{EM}_n$ of the operator $\mathcal{E}_n$
Alternative approaches should be investigated with the aim of either modifying the current definition of the approximate operator so that the centred scheme without reconstruction, for example, is taken into account, or abandoning it altogether.

• Test of convergence
The quantity defined as criterion for the convergence test is also needs to be reviewed and potentially simplified.

• **Consideration of T_s^{imp}**
During the resolution of the equation by `codits`, the `ROVSDT` array has two functions: it serves to compute the diagonal elements of the matrix (by calling `matrix`) and it serves to update the right-hand side at each sub-iteration of the incremental resolution. However, if T_s^{imp} is positive, we don't include it in `ROVSDT` so as not to reduce the diagonal dominance of the matrix. In consequence, it is not used in the update of the right-hand side, although it would certainly be feasible to include positive values in the updates. The outcome of this is a source term that has been computed in a fully explicit manner ($T_s^{exp} + T_s^{imp} a^n$), whereas the advantage of the incremental approach is precisely that it allows for almost total implicitization of the solution ($T_s^{exp} + T_s^{imp} a^{n+1,k_{fin}-1}$, where k_{fin} is the final sub-iteration executed).
To achieve this, would require defining two `ROVSDT` arrays in `codits`.

G- condli routine

Function

Boundary conditions are required in at least three principal cases:

- calculation of the convection terms (first order derivative in space) at the boundary: the calculation uses a flux at the boundary and requires the value of the convected variable at the boundary when the latter is entering the domain in the sens of the characteristic curves of the system (in the sens of the velocity, in the case of the single equation of a simple scalar: sufficient interpretation in the current framework of code_saturne[1]) ;
- calculation of the diffusion terms (second order derivative in space): a method to determine the value of the first order spatial derivatives at the boundary is then required (more exactly, the terms that depend upon it are required, such as the stresses of the thermal fluxes at the wall):
- calculation of the cell gradients: the variable at the boundary faces are required (more generally, the discrete terms of the equations which depend upon the gradient inside boundary cells are required, such as the transpose gradient terms in the Navier-Stokes equations).

These considerations only concern the computational variables (velocity, pressure, Reynolds tensor, scalars solution of a convection-diffussion equation). For these variables [2], the user has to define the boundary conditions at every boundary face (`usclim`).

The `condli` subroutine transforms the data provided by the user (in `usclim`) into an internal format of representation of the boundary conditions. Verifications of the completeness and coherence are also performed (in `vericl`). More particularly, the smooth-wall boundary conditions (`clptur`), the rough-wall boundary conditions (`clptrg`) and the symmetry boundary conditions for the velocities and the Reynolds stress tensor (`clsyvt`) are treated in dedicated subroutines.

The `condli` subroutine provides as an output pairs of coefficients A_b and B_b for each variable f and for each boudary face. These are used for the calculation of the discrete terms in the equations to be solved. More specifically, these coefficients are used to calculate a value at the boundary faces $f_{b,int}$ (localised at the centre of the boundary face, barycentre of its vertices) by the relation $f_{b,int} = A_b + B_b\, f_{I'}$, where $f_{I'}$ is the value of the variable at point I'. I' is the projection onto the centre of the cell adjoin to the boundary on the line normal to the boundary face and passing by its centre (see figure IV.G.1).

See the programmers reference of the dedicated subroutine for further details.

Discretization

• Notation
In the following, we will denote by *VarScalaire* any variable

- other than the velocity, pressure, turbulent quantities k, ε, R_{ij}, φ, $\bar{f}$ and ω,
- solution of a convection-diffusion equation.

[1] except with the compressible module, see. `cfxtcl`
[2] The other variables (the physical properties for instance) have a different treatment which will not be detailed here (for instance, for the density, the user defines directly the values at the boundary. This information is then stored ; one is referred to `usphyv` or `phyvar` for more information).

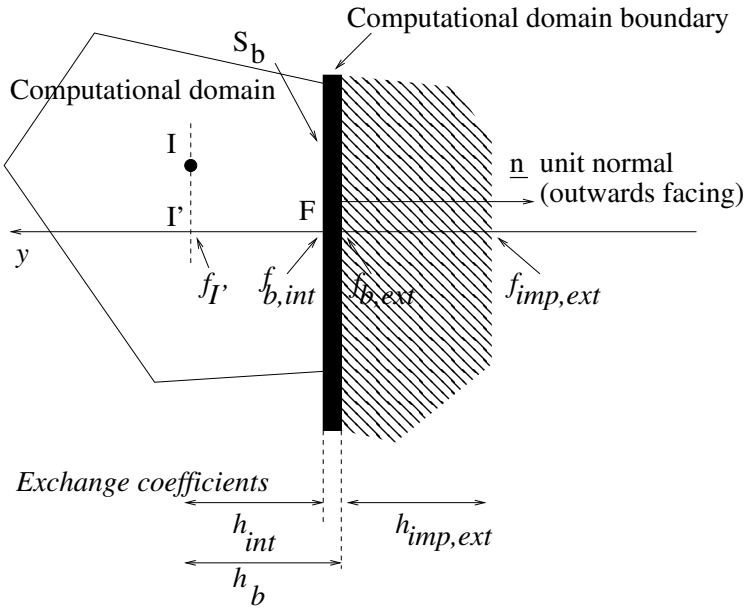


Figure IV.G.1: Boundary cell.

The denomination *VarScalaire* may in particular designate the temperature, a passive scalar, a mass fraction or (unless explicitly stated otherwise) the variance of fluctuations of another *VarScalar*. The inferred state variables (mass volumetric, viscosity...) will not be denoted by *VarScalar*.

• **Representation of standard boundary conditions in `cs_user_boundary_conditions`**

Standard boundary conditions can be provided by the user in `cs_user_boundary_conditions`. For this, it is necessary to assign a type to boundary faces [3]. The default conditions are as follows:

- **CS_INLET**: corresponds to a Dirichlet condition on all the variables transported (velocity, turbulent variables, *VarScalaires*...), and a homogeneous Neumann condition (zero flow) on the pressure.
- **CS_OUTLET**:
  - when the mass flow is effectively directed outwards of the domain, this choice corresponds to a homogeneous Neumann condition on all the transported variables and $\frac{\partial^2 P}{\partial \underline{n} \partial \underline{\tau}_i} = 0$, taken into account as of Dirichlet for the pressure ($\underline{n}$ and $(\underline{\tau}_i)_{i\in\{1,2\}}$ denote respectively the normal vector of the exit face considered and two normed vectors, orthogonal to each other and in the plane of the exit face). This condition is explicit using the pressure field and its gradient at the previous time step. Moreover, the pressure being defined to a constant, it is readjusted to a point of output to store the value `P0` (this avoids any drift towards very large values relatively to the maximum pressure difference on the domain)[4].
  - when the mass flow is directed inwards, an undesirable situation *a priori*[5],
    A Homogeneous Dirichlet condition is imposed on the on velocity (not on the mass flux), its value downstream of the domain being unknown. The pressure is treated as in the previous case where the mass flux is directed outwards of the domain. For variables other than velocity and pressure, two cases can occur:

[3] The assignment of a type is done by filling in the table `BC_TYPE`.

[4] When there is not output, the spectrum of eigen values of the matrix is shifted by a constant value in order to make the system invertible: see `cs_matrix_wrapper_scalar`.

[5] A message indicates to the user how many exit faces see a mass flow entering into the domain (i.e. a backflow).

- one can impose a condition of Dirichlet to represent the value of the scalar introduced into the field by the faces of boundary concerned.
- one can impose, as when the mass flux is outgoing, a homogeneous Neumann condition (this is not a desirable situation, since the information carried on the boundary faces then comes from *the downstream* of the local flow). This is the case by default if no value is given for the Dirichlet.

- **CS_SMOOTHWALL**: refer to `clptur` (or to `clptrg` for rough walls) for a description of the treatment of the boundary conditions of the wall (assumed to be impermeable to fluid). Briefly, we can say here that a wall law approach is used to impose the constraint tangential on speed. The wall may slide[6]. The *VarScalars* are assigned a homogeneous Neumann condition (zero flow) by default. To impose a wall value for these variables (for example, in the case of a wall at imposed temperature) a law of similarity is used to determine the boundary flux taking into account the boundary layer. In the case of couplings with SYRTHES, code_saturne receives a wall temperature and sends a heat flux (decomposed as a pseudo interior temperature and exchange coefficient to allow relaxation). The standard pressure condition is a homogeneous Neumann condition.

- **CS_SYMMETRY**: corresponds to homogeneous Neumann conditions for scalar quantities and to classical symmetry conditions for vectors (velocity) and tensors (Reynolds voltages): see `clsyvt`.

• Representation of specific boundary conditions in `cs_user_boundary_conditions`

We have seen that the assignment to a boundary face of a standard type (entrance, exit, wall, symmetry) makes it possible to apply coherent boundary conditions to all variables in a simple manner for the usual types of physical boundaries.

It is also possible to define specific boundary conditions in `cs_user_boundary_conditions`, for each boundary face and each variable [7] (these, like the standard conditions, ultimately amount to mixed-type conditions).

The two approaches are compatible and often combined. Indeed, the standard (default) boundary conditions can be overridden by the user where necessary, and the defaults used elsewhere. In any case, it should be ensured that a boundary condition has been defined for each boundary face and variable.

Compatibility conditions also exist between the different variables (see `vericl`):

- between, wall, symmetry or free exit, it is important that all the velocity components have the same type of condition ;
- when the velocity receives an exit condition, it is important that the pressure is assigned a Dirichlet-like condition. For more details, refer to the paragraph relating to the exit condition for pressure, page 221;
- when one of the velocity or turbulence variables is assigned a wall condition, so must all such variables;
- when one of the $velocityor R_{ij}$ components is assigned a symmetry condition, it should be the case for all components;
- when a *VarScalar* receives a wall condition, the velocity must have the same type of condition.

[6] We must then provide the components of the wall velocity.

[7] Such conditions are direclty defined through the `icodcl` and `rcodcl` arrays for each face and variable: examples are provided in `cs_user_boundary_conditions`.

• **Internal representation of boundary conditions**

Objective

The conditions defined by the user are converted in the form of pairs of coefficients A_b and B_b for each variable f and boundary face. These coefficients are used for the calculation of discrete terms intervening in the solved equation and make it possible in particular to determine a boundary face value $f_{b,int}$. It is important to insist on the fact that this value is, in general, a simple numerical value which does not necessarily reflect a physical reality (in particular to the walls, for the quantities affected by the turbulent boundary layer). We detail the calculation of A_b, B_b and $f_{b,int}$ below.

Notations

- We consider the equation (IV.G.1) on the scalar f, in which ρ represents the density, $\underline{u}$ the velocity, α the conductivity and S additional source terms. C is defined below.

$$\rho\frac{\partial f}{\partial t} + div(\rho\underline{u}f) = div\left(\frac{\alpha}{C}\underline{\nabla} f\right) + S \tag{IV.G.1}$$

- The coefficient α represents the sum of the molecular and turbulent conductivities (depending on the models used), or $\alpha = \alpha_m + \alpha_t$, with, for a turbulent viscosity based modeling, $\alpha_t = C\,\dfrac{\mu_t}{\sigma_t}$, where σ_t is the turbulent Prandtl number[8].
- The coefficient C_p represents the specific heat, with unit $m^2\,s^{-2}\,K^{-1} = J\,kg^{-1}\,K^{-1}$.
- We note λ the thermal conductivity, with unit $kg\,m\,s^{-3}\,K^{-1} = W\,m^{-1}\,K^{-1}$.
- It should be specified that $C = 1$ for all variables except for the temperature, for which $C = C_p$[9]. In the code, the value which the user must define is $\dfrac{\alpha_m}{C}$ (if the property is constant, values are initialized to `viscl0` for the velocity and `visls0` for *scalar variables*; If the property is variable, equivalent field values must be defined in the GUI or in `cs_user_physical_properties.c`).
- For the variance of the fluctuations of a *scalar variable*, the conductivity α and the coefficient C are inherited from the associated variable.

Simple Dirichlet condition: for a simple Dirichlet condition, we naturally obtain (special case of(IV.G.6)):

$$\underbrace{f_{b,int}}_{\text{boundary value used by the calculation}} = \underbrace{f_{real}}_{\text{real value imposed on the boundary}} \tag{IV.G.2}$$

Other cases: when the boundary condition is based on a flux, this is of a diffusive flow [10]. We then have:

$$\underbrace{\phi_{int}}_{\text{diffusive flux towards the internal domain}} = \underbrace{\phi_{real}}_{\text{real diffusive flux imposed on the boundary}} \tag{IV.G.3}$$

The imposed real diffusive flux can be given

- directly (Neumann condition), or $\phi_{réel} = \phi_{imp,ext}$ or

[8] The turbulent Prandtl number is dimensionless and, in some usual cases, taken as equal to 0.7.

[9] More precisely, we have $C = C_p$ for all *scalar variables* f that we want to consider as the temperature for the boundary conditions. These *scalar variables* can be checked using the `is_temperature` field keyword. By default this value is set to 0 for all variables except for the temperature.

[10] Indeed, the total outgoing flow of the domain is given by the sum of the convective flux (if the variable is actually convected) and diffusive flux. However, for solid walls and symmetries, the mass flow is zero and the condition reduced to a constraint on the diffusive flux. Moreover, for the outlets (exiting mass flow), the boundary condition relates only to the diffusive flux (often a homogeneous Neumann condition), convective flux dependent on the upstream conditions (therefore it does not need boundary conditions). Finally, at the inlets, a simple Dirichlet condition is most often used, and the diffusive flux is deduced from it.

- implicitly deduced from two imposed values: an exterior value $f_{imp,ext}$ and an exchange coefficient $h_{imp,ext}$ (generalized Dirichlet condition).

Depending on the type of condition (Dirichlet or Neumann) and assuming the conservation of the flux normal to the boundary, we can write (see figure IV.G.1):

$$\underbrace{h_{int}(f_{b,int} - f_{I'})}_{\phi_{int}} = \underbrace{h_b(f_{b,ext} - f_{I'})}_{\phi_b} = \begin{cases} \underbrace{h_{imp,ext}(f_{imp,ext} - f_{b,ext})}_{\phi_{imposed\ real}} & \text{(condition of Dirichlet)} \\ \underbrace{\phi_{imp,ext}}_{\phi_{imposed\ real}} & \text{(Neumann condition)} \end{cases} \tag{IV.G.4}$$

The ratio between the coefficient h_b and the coefficient h_{int} accounts for the importance of crossing the near-boundary zone and is of particular importance in the case of walls along which a boundary layer develops (whose properties are then taken into account by h_b: see `clptur`). In the simpler framework considered here, we will limit ourselves to case $h_b = h_{int}$ and $f_{b,ext} = f_{b,int} = f_b$. The relation (IV.G.4) is then written:

$$\underbrace{h_{int}(f_b - f_{I'})}_{\phi_{int}} = \begin{cases} \underbrace{h_{imp,ext}(f_{imp,ext} - f_b)}_{\phi_{imposed\ real}} & \text{(condition of Dirichlet)} \\ \underbrace{\phi_{imp,ext}}_{\phi_{imposed\ real}} & \text{(condition Neumann)} \end{cases} \tag{IV.G.5}$$

By rearranging, we get the boundary value:

$$f_b = \begin{cases} \dfrac{h_{imp,ext}}{h_{int} + h_{imp,ext}} f_{imp,ext} + \dfrac{h_{int}}{h_{int} + h_{imp,ext}} f_{I'} & \text{(Dirichlet condition)} \\ \dfrac{1}{h_{int}} \phi_{imp,ext} + f_{I'} & \text{(Neumann condition)} \end{cases} \tag{IV.G.6}$$

Conclusion: we will therefore note the boundary conditions in the general form:

$$f_b = A_b + B_b\, f_{I'} \tag{IV.G.7}$$

with A_b et B_b defined according to the type of conditions:

$$\text{Dirichlet} \begin{cases} A_b = \dfrac{h_{imp,ext}}{h_{int} + h_{imp,ext}} f_{imp,ext} \\ B_b = \dfrac{h_{int}}{h_{int} + h_{imp,ext}} \end{cases} \quad \text{Neumann} \begin{cases} A_b = \dfrac{1}{h_{int}} \phi_{imp,ext} \\ B_b = 1 \end{cases} \tag{IV.G.8}$$

Remarks

- The value $f_{I'}$ is calculated using the cell gradient of f, that is: $f_{I'} = f_I + \underline{II'}\nabla f_I$.
- The value of h_{int} must yet be specified. This is a *numerical*, value, having *a priori* no relation to a physical exchange coefficient, which depends on the mode of calculation of the diffusive flux in the boundary cell. Thus $h_{int} = \frac{\alpha}{\overline{I'F}}$ (the unit is naturally deduced).
- We also repeat, because it can be a cause of error, that in the code, we have:
  - for the temperature $\alpha_m = \lambda$ et $C = C_p$
  - for the enthalpy $\alpha_m = \frac{\lambda}{C_p}$ et $C = 1$

Examples of special cases

- In the case of a Dirichlet condition, the user is therefore led to provide two values: $f_{imp,ext}$ and $h_{imp,ext}$. To obtain a simple Dirichlet condition (without exchange coefficient) just impose $h_{imp,ext} = +\infty$. This is the most commonl case (in practice, $h_{imp,ext} = cs_math_infinite_r$).
- In the case of a Neumann condition, the user provides a single value $\phi_{imp,ext}$ (zero for homogeneous Neumann conditions).

• Output condition for pressure

We specify here condition applied to the pressure for standard outlets. It is necessary to impose a Dirichlet type condition (accompanied by a homogeneous Neumann condition on the components of the velocity). We compute it based on the values of the variable at the previous time step.

- Reasoning on a simple configuration (a channel, with a flat output, perpendicular to the flow), it can be assumed that the shape of the pressure profiles taken on the surfaces parallel to the outlet are unchanged around this outlet (hypothesis of an established flow, far from any perturbation). In this situation, we can write $\frac{\partial^2 P}{\partial \underline{n} \partial \underline{\tau}_i} = 0$ ($\underline{n}$ is the vector normal to the outlet, $\underline{\tau}_i$ represents a basis of the exit plane).
- If, moreover, it can be assumed that the pressure gradient taken in the direction perpendicular to the outlet faces is uniform in its neighborhood, the profile to impose at the output (values p_b) can be deduced from the profile taken on an upstream plane ($p_{upstream}$ values) by simply adding the constant $R = d\,\underline{\nabla}(p).\underline{n}$ (where d is the distance between the upstream plane and the outlet), or $p_b = p_{upstream} + R$ (the fact that R is identical for all outlet faces is important so we can eliminate it in the equation (IV.G.9) below).
- With the additional assumption that the points I' relative to the outlet faces are on a plane parallel to the output, we can use the values at these points ($p_{I'}$) for upstream values, so $p_{upstream} = p_{I'} = p_I + \underline{II'}.\underline{\nabla}p$.
- Furthermore, the pressure being defined relative to a constant (incompressible flow) one can fix its value arbitrarily at a point A [11]) to p_0 (value fixed by user, equal to `P0` and null by default), and therefore shift the imposed profile at the output by adding:
  $R_0 = p_0 - (p_{upstream,A} + R) = p_0 - (p_{I',A} + R)$.

[11] by default, the center of mass of the outlet faces

- So we finally obtain:

$$\begin{aligned} p_b &= p_{I'} + R + R_0 \\ &= p_{I'} + R + p_0 - (p_{I',A} + R) \\ &= p_{I'} + \underbrace{p_0 - p_{I',A}}_{\text{constant value } R_1} \\ &= p_{I'} + R_1 \end{aligned} \tag{IV.G.9}$$

It is therefore noted that the pressure condition at the outlet is Dirichelet condition whose values are equal to the pressure values (taken at previous time step) on the plane upstream of the points I' and readjusted to obtain `P0` in an arbitrary exit point.

Remaining issues

• Representation of conditions by face value
Although the method used allows simplicity and homogeneity of treatment of all boundary conditions, it is quite restrictive in the sense that a single value is not always sufficient to represent the conditions to be applied when calculating different terms.

Thus, in $k-\varepsilon$ it was necessary, when calculating the boundary conditions of the wall, to use two (A_b, B_b) pairs in order to take into account the conditions to apply* for the calculation of the shear stress and those to be used when calculating the production term (and a third set of coefficients would be necessary for allow the treatment of the gradients intervening in the terms of gradient transposed, in `visecv`).

Perhaps it could be useful to set up a method allowing to direclty use (at least in some strategic points of the code) forces, stresses or fluxes, without requiring a general a face value computation.

• Pressure outlet condition
The outlet pressure condition translates to $p_f = p_{I'} + R1$ and the profile obtained corresponds to the upstream profile taken at points I' and readjusted to obtain p_0 at an arbitrary point A. This type of condition is applied without precautions, but is not always justified (a Dirichlet condition based on the calculated value directly at the boundary faces might be more suitable). The hypotheses are particularly faulty in the following cases:

- the outlet is close to an area where the flow is not established in space (or varies in time);
- the outlet is not perpendicular to the flow ;
- the pressure gradient in the direction normal to the outlet is not the same for all output faces (in the case of multiple outputs, for example, the gradient is probably not the same across all outputs);
- points I' are not on a surface parallel to the output (case of irregular meshes for example).

Moreover, in the absence of an outlet condition, it could perhaps prove useful to set a reference value on a given cell or to bring the average pressure back to a reference value (with the shift of the spectrum, one ensures the invertibility of the matrix at each step time, but we should also check whether the pressure is not likely to drift during of the calculation).

• Terms unaccounted for
The current boundary conditions seem to cause issues when treating the transposed velocity gradient term in the Navier-Stokes equations (handled as explicit term in the time scheme). This term can be deactivated setting the `ivisse` keyword to 0 in the velocity-pressure model.

H- gradrc routine

Fonction

Le but de ce sous-programme est de calculer, au centre des cellules, le gradient d'une fonction scalaire, également connue au centre des cellules. Pour obtenir la valeur du gradient, une méthode itérative de reconstruction pour les maillages non orthogonaux est mise en œuvre : elle fait appel à un développement limité d'ordre 1 en espace sur la variable, obtenu à partir de la valeur de la fonction et de son gradient au centre de la cellule. Cette méthode, choisie comme option par défaut, correspond à `imrgra` = 0 et est utilisée pour le calcul des gradients de toutes les grandeurs.

Discrétisation

La méthode est décrite á la section 4.4.1.

Remarque
Pour les conditions aux limites en pression, un traitement particulier est mis en œuvre, surtout utile dans les cas où un gradient de pression (hydrostatique ou autre) nécessite une attention spécifique aux bords, où une condition à la limite de type Neumann homogène est généralement inadaptée. Soit $P_{F_{b\,ik}}$ la valeur de la pression à la face associée, que l'on veut calculer.

On note que :

$$\underline{I'F}_{b\,ik} \cdot (\underline{\nabla} P)_I = \underline{I'F}_{b\,ik} \cdot \underline{G}_{c,i} = \overline{I'F}_{b\,ik} \cdot \left.\frac{\delta P}{\delta n}\right|_{F_{b\,ik}}$$

avec les conventions précédentes.

Sur maillage orthogonal On se place dans le cas d'un maillage orthogonal , *i.e.* pour toute cellule Ω_I, I et son projeté I' sont identiques. Soit $M_{b\,ik}$ le milieu du segment $IF_{b\,ik}$.
On peut écrire les égalités suivantes :

$$\begin{aligned} P_{F_{b\,ik}} &= P_{M_{b\,ik}} + \overline{M_{b\,ik}F_{b\,ik}} \cdot \left.\frac{\delta P}{\delta n}\right|_{M_{b\,ik}} + \overline{M_{b\,ik}F_{b\,ik}}^2 \cdot \frac{1}{2}\left.\frac{\delta^2 P}{\delta n^2}\right|_{M_{b\,ik}} + \mathcal{O}(h^3) \\ P_I &= P_{M_{b\,ik}} + \overline{M_{b\,ik}I} \cdot \left.\frac{\delta P}{\delta n}\right|_{M_{b\,ik}} + \overline{M_{b\,ik}I}^2 \cdot \frac{1}{2}\left.\frac{\delta^2 P}{\delta n^2}\right|_{M_{b\,ik}} + \mathcal{O}(h^3) \end{aligned}$$

avec $\overline{M_{b\,ik}I} = -\overline{M_{b\,ik}F_{b\,ik}}$.
On obtient donc :

$$P_{F_{b\,ik}} - P_I = \overline{IF}_{b\,ik} \cdot \left.\frac{\delta P}{\delta n}\right|_{M_{b\,ik}} + \mathcal{O}(h^3) \qquad \text{(IV.H.1)}$$

Grâce à la formule des accroissements finis :

$$\left.\frac{\delta P}{\delta n}\right|_{M_{b\,ik}} = \frac{1}{2}\left[\left.\frac{\delta P}{\delta n}\right|_I + \left.\frac{\delta P}{\delta n}\right|_{F_{b\,ik}}\right] + \mathcal{O}(h^2) \qquad \text{(IV.H.2)}$$

On s'intéresse aux cas suivants :

• condition à la limite de type Dirichlet
$P_{F_{b\,ik}} = P_{IMPOSE}$, aucun traitement particulier

• condition à la limite de type Neumann homogène
On veut imposer :

$$\left.\frac{\delta P}{\delta n}\right|_{F_{b\,ik}} = 0 + \mathcal{O}(h) \tag{IV.H.3}$$

On a :

$$\left.\frac{\delta P}{\delta n}\right|_{I} = \left.\frac{\delta P}{\delta n}\right|_{F_{b\,ik}} + \mathcal{O}(h)$$

et comme :

$$P_{F_{b\,ik}} = P_I + \overline{IF}_{b\,ik} \cdot \left.\frac{\delta P}{\delta n}\right|_{I} + \mathcal{O}(h^2) \tag{IV.H.4}$$

on en déduit :

$$P_{F_{b\,ik}} = P_I + \overline{IF}_{b\,ik} \cdot \left.\frac{\delta P}{\delta n}\right|_{F_{b\,ik}} + \mathcal{O}(h^2) \tag{IV.H.5}$$

soit :

$$P_{F_{b\,ik}} = P_I + \mathcal{O}(h^2) \tag{IV.H.6}$$

On obtient donc une approximation d'ordre 1.

Sur maillage non orthogonal Dans ce cas, on peut seulement écrire :

$$P_{F_{b\,ik}} = P_{I'} + \frac{1}{2}\underline{I'F}_{b\,ik} \cdot [\,(\underline{\nabla} P)_{I'} + (\underline{\nabla} P)_{F_{b\,ik}}\,] + \mathcal{O}(h^3) \tag{IV.H.7}$$

• condition à la limite de type Dirichlet
$P_{F_{b\,ik}} = P_{IMPOSE}$, toujours aucun traitement particulier

• condition à la limite de type Neumann homogène
On veut :

$$\left.\frac{\delta P}{\delta n}\right|_{F_{b\,ik}} = 0 + \mathcal{O}(h) \tag{IV.H.8}$$

ce qui entraîne :

$$\underline{I'F}_{b\,ik} \cdot (\underline{\nabla} P)_{F_{b\,ik}} = \mathcal{O}(h^2) \tag{IV.H.9}$$

On peut écrire :

$$(\underline{\nabla} P)_{I'} = (\underline{\nabla} P)_{F_{b\,ik}} + \mathcal{O}(h)$$

d'où :

$$P_{F_{b\,ik}} = P_{I'} + \mathcal{O}(h^2) \tag{IV.H.10}$$

On obtient donc une approximation d'ordre 1.

• **Conclusion**
On peut récapituler toutes ces situations *via* la formule :

$$P_{F_{b\,ik}} = P_{I'}$$

I- inimas routine

Fonction

Le but de ce sous-programme est principalement de calculer le flux de masse aux faces. Il prend une variable vectorielle associée au centre des cellules (généralement la vitesse), la projette aux faces en la multipliant par la masse volumique, et la multiplie scalairement par le vecteur surface. Plus généralement, `inimas` est aussi appelé comme première étape du calcul d'une divergence (terme en $\mathrm{div}\,(\rho\underline{\underline{R}})$ en $R_{ij}-\varepsilon$, filtre Rhie & Chow, ...).

Discrétisation

La figure IV.I.1 rappelle les diverses définitions géométriques pour les faces internes et les faces de bord. On notera $\alpha = \dfrac{\overline{FJ'}}{\overline{I'J'}}$ (défini aux faces internes uniquement).

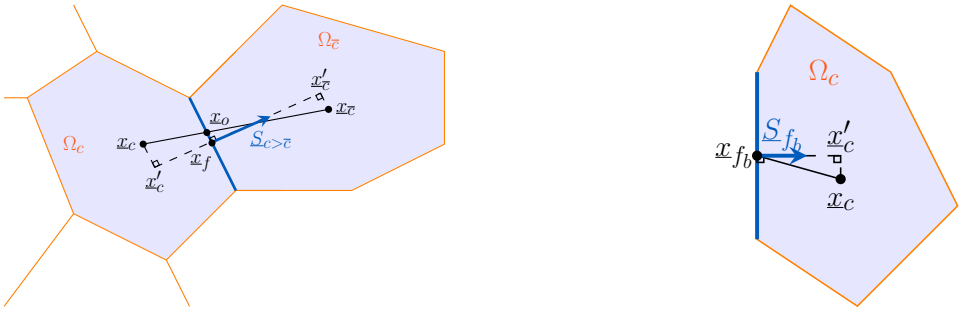


Figure IV.I.1: Définition des différentes entités géométriques pour les faces internes (gauche) et de bord (droite).

Faces internes

On ne connaît pas la masse volumique à la face, cette dernière doit donc aussi être interpolée. On utilise la discrétisation suivante :

$$(\rho\underline{u})_F = \alpha(\rho_I\underline{u}_I) + (1-\alpha)(\rho_J\underline{u}_J) + \underline{\underline{\mathrm{grad}}}(\rho\underline{u})_O.\underline{OF} \tag{IV.I.1}$$

La partie en $\alpha(\rho_I\underline{u}_I) + (1-\alpha)(\rho_J\underline{u}_J)$ correspondant en fait à $(\rho\underline{u})_O$. Le gradient en O est calculé par interpolation : $\underline{\underline{\mathrm{grad}}}(\rho\underline{u})_O = \dfrac{1}{2}\left[\underline{\underline{\mathrm{grad}}}(\rho\underline{u})_I + \underline{\underline{\mathrm{grad}}}(\rho\underline{u})_J\right]$. La valeur $\dfrac{1}{2}$ s'est imposée de manière heuristique au fil des tests comme apportant plus de stabilité à l'algorithme global qu'une interpolation faisant intervenir α. L'erreur commise sur $\rho\underline{u}$ est en $O(h^2)$.

Faces de bord

Le traitement des faces de bord est nécessaire pour y calculer le flux de masse, bien sûr, mais aussi pour obtenir des conditions aux limites pour le calcul du $\underline{\underline{\mathrm{grad}}}(\rho\underline{u})$ utilisé pour les faces internes.

Pour les faces de bord, on connaît la valeur de ρ_F, qui est stockée dans la variable `ROMB`. De plus, les conditions aux limites pour $\underline{u}$ sont données par des coefficients A et B tels que :

$$u_{k,F} = A_k + B_k u_{k,I'} = A_k + B_k \left(u_{k,I} + \underline{\nabla}(u_k)_I . \underline{II'}\right) \tag{IV.I.2}$$

($k \in \{1,2,3\}$ est la composante de la vitesse, l'erreur est en $O(B_k h)$)

On a donc à l'ordre 1 :

$$(\rho u_k)_F = \rho_F \left[A_k + B_k \left(u_{k,I} + \underline{\nabla}(u_k)_I . \underline{II'}\right)\right] \tag{IV.I.3}$$

Mais pour utiliser cette formule, il faudrait calculer $\underline{\underline{\text{grad}}}(\underline{u})$ (trois appels à `GRDCEL`), alors qu'on a déjà calculé $\underline{\underline{\text{grad}}}(\rho\underline{u})$ pour les faces internes. Le surcoût en temps serait alors important. On réécrit donc :

$$\begin{aligned}
(\rho u_k)_F &= \rho_F A_k + \rho_F B_k u_{k,I'} && \text{(IV.I.4)}\\
&= \rho_F A_k + B_k \frac{\rho_F}{\rho_{I'}} (\rho u_k)_{I'} && \text{(IV.I.5)}\\
&= \rho_F A_k + B_k \frac{\rho_F}{\rho_{I'}} (\rho u_k)_I + B_k \frac{\rho_F}{\rho_{I'}} \underline{\nabla}(\rho u_k)_I . \underline{II'} && \text{(IV.I.6)}
\end{aligned}$$

Pour calculer les gradients de $\rho\underline{u}$, il faudrait donc en théorie utiliser les coefficients de conditions aux limites équivalents :
$\tilde{A}_k = \rho_F A_k$
$\tilde{B}_k = B_k \dfrac{\rho_F}{\rho_{I'}}$

Ceci paraît délicat, à cause du terme en $\dfrac{\rho_F}{\rho_{I'}}$, et en particulier à l'erreur que l'on peut commettre sur $\rho_{I'}$ si la reconstruction des gradients est imparfaite (sur des maillages fortement non orthogonaux par exemple). On réécrit donc l'équation (IV.I.6) sous la forme suivante :

$$(\rho u_k)_F = \rho_F A_k + B_k \frac{\rho_I \rho_F}{\rho_{I'}} u_{k,I} + B_k \frac{\rho_F}{\rho_{I'}} \underline{\nabla}(\rho u_k)_I . \underline{II'} \tag{IV.I.7}$$

Pour le calcul du flux de masse au bord, on va faire deux approximations. Pour le deuxième terme, on va supposer $\rho_{I'} \approx \rho_I$ (ce qui conduit à une erreur en $O(B_k h)$ sur $\rho\underline{u}$ si $\rho_{I'} \neq \rho_I$). Pour le troisième terme, on va supposer $\rho_{I'} \approx \rho_F$. Cette dernière approximation est plus forte, mais elle n'intervient que dans la reconstruction des non-orthogonalités ; l'erreur finale reste donc faible (erreur en $O(B_k h^2)$ sur $\rho\underline{u}$ si $\rho_{I'} \neq \rho_F$). Et au final, le flux de masse au bord est calculé par :

$$\dot{m}_F = \sum_{k=1}^{3} \left[\rho_F A_k + B_k \rho_F u_{k,I} + B_k \underline{\nabla}(\rho u_k)_I . \underline{II'}\right] S_k \tag{IV.I.8}$$

Pour le calcul des gradients, on repart de l'équation (IV.I.5), sur laquelle on fait l'hypothèse que $\rho_{I'} \approx \rho_F$. Encore une fois, cette hypothèse peut être assez forte, mais les gradients obtenus ne sont utilisés que pour des reconstructions de non-orthogonalités ; l'erreur finale reste donc là encore assez faible. Au final, les gradients sont calculés à partir de la formule suivante :

$$(\rho u_k)_F = \rho_F A_k + B_k (\rho u_k)_{I'} \tag{IV.I.9}$$

ce qui revient à utiliser les conditions aux limites suivantes pour $\rho\underline{u}$:
$\tilde{A}_k = \rho_F A_k$
$\tilde{B}_k = B_k$

Remarque

Dans la plupart des cas, les approximations effectuées n'engendrent aucune erreur. En effet :
- dans le cas d'une entrée on a généralement $B_k = 0$, avec un flux de masse imposé par la condition à

la limite.
- dans le cas d'une sortie, on a généralement flux nul sur les scalaires donc sur ρ, soit $\rho_F = \rho_{I'} = \rho_I$.
- dans le cas d'une paroi, on a généralement $B_k = 0$ et le flux de masse est imposé nul.
- dans le cas d'une symétrie, on a généralement $\rho_F = \rho_{I'} = \rho_I$ et le flux de masse est imposé nul.
Pour sentir un effet de ces approximations, il faudrait par exemple une paroi glissante ($B_k \neq 0$) avec un gradient de température ($\rho_F \neq \rho_I$).

Mise en œuvre

La vitesse est passée par les arguments `UX`, `UY` et `UZ`. Les conditions aux limites de la vitesse sont `COEFAX`, `COEFBX`, ... Le flux de masse résultat est stocké dans les variables `FLUMAS` (faces internes) et `FLUMAB` (faces de bord). `QDMX`, `QDMY` et `QDMZ` sont des variables de travail qui serviront à stocker $\rho\underline{u}$, et `COEFQA` servira à stocker les $\tilde{A}$.

• Initialisation éventuelle du flux de masse
Si `INIT` vaut 1, le flux de masse est remis à zéro. Sinon, le sous-programme rajoute aux variables `FLUMAS` et `FLUMAB` existantes le flux de masse calculé.

• Remplissage des tableaux de travail
$\rho\underline{u}$ est stocké dans `QDM`, et $\tilde{A}$ dans `COEFQA`.

• Cas sans reconstruction
On calcule alors directement

$$\texttt{FLUMAS} = \sum_{k=1}^{3} \left[\alpha(\rho_I u_{k,I}) + (1-\alpha)(\rho_J u_{k,J})\right] S_k$$

et

$$\texttt{FLUMAB} = \sum_{k=1}^{3} \left[\rho_F A_k + B_k \rho_F u_{k,I}\right] S_k$$

• Cas avec reconstruction
On répète trois fois de suite les opérations suivantes, pour $k = 1$, 2 et 3 :
- Appel de `GRDCEL` pour le calcul de $\underline{\nabla}(\rho u_k)$.
- Mise à jour du flux de masse

$$\texttt{FLUMAS} = \texttt{FLUMAS} + \left[\alpha(\rho_I u_{k,I}) + (1-\alpha)(\rho_J u_{k,J}) + \frac{1}{2}\left[\underline{\nabla}(\rho u_k)_I + \underline{\nabla}(\rho u_k)_J\right].\underline{OF}\right] S_k$$

et

$$\texttt{FLUMAB} = \texttt{FLUMAB} + \left[\rho_F A_k + B_k \rho_F u_{k,I} + B_k \underline{\nabla}(\rho u_k)_I.\underline{II'}\right] S_k$$

• Annulation du flux de masse au bord
Quand le sous-programme a été appelé avec la valeur `IFLMB0=1` (c'est-à-dire quand il est réellement appelé pour calculer un flux de masse, et pas pour calculer le terme en div $(\rho\underline{\underline{R}})$ par exemple), le flux de masse au bord `FLUMAB` est forcé à 0, pour les faces de paroi et de symétrie (identifiées par `ISYMPA=0`).

J- itrmas/itrgrp routine

Fonction

Le but du sous-programme `itrmas` est de calculer un gradient de pression "facette". Il est utilisé dans la phase de correction de pression (deuxième phase de `cs_solve_navier_stokes`) où le flux de masse est mis à jour à l'aide de termes en $-\Delta t_{ij}(\underline{\nabla}_f P)_{ij}.\underline{S}_{ij}$ et en $-\Delta t_{b_{ik}}(\underline{\nabla}_f P)_{b_{ik}}\cdot\underline{S}_{b_{ik}}$.

Le sous-programme `itrgrp` calcule un gradient "facette" de pression et en prend la divergence, c'est-à-dire calcule le terme :

$$-\sum_{j\in Vois(i)}\Delta t_{ij}(\underline{\nabla}_f P)_{ij}.\underline{S}_{ij}-\sum_{k\in\gamma_b(i)}\Delta t_{b_{ik}}(\underline{\nabla}_f P)_{b_{ik}}\cdot\underline{S}_{b_{ik}}$$

En pratique `itrgrp` correspond à la combinaison de `itrmas` et de `divmas`, mais permet par son traitement en un seul bloc d'éviter la définition de tableaux de travail de taille `NFAC` et `NFABOR`.

See the programmers reference of the dedicated subroutine for further details.

Discrétisation

La figure IV.J.1 rappelle les diverses définitions géométriques pour les faces internes et les faces de bord.

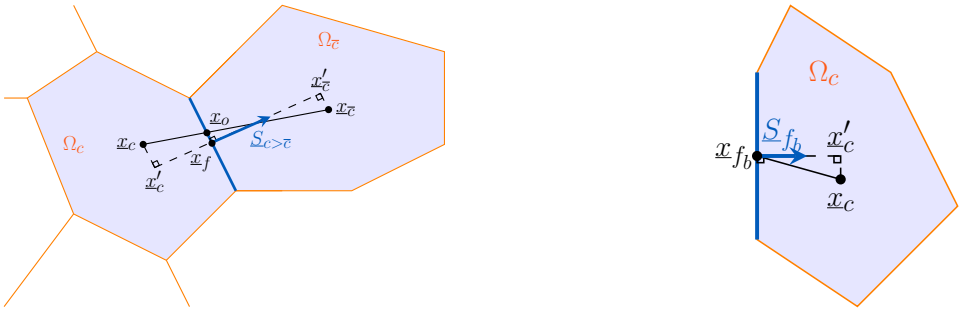


Figure IV.J.1: Définition des différentes entités géométriques pour les faces internes (gauche) et de bord (droite).

Calcul sans reconstruction des non orthogonalités

Pour les faces internes, on écrit simplement :

$$-\Delta t_{ij}(\underline{\nabla}_f P)_{ij}\cdot\underline{S}_{ij}=\frac{\Delta t_{ij}S_{ij}}{\overline{I'J'}}(P_I-P_J) \tag{IV.J.1}$$

Pour les faces de bord, on écrit :

$$-\Delta t_{b_{ik}}(\underline{\nabla}_f P)_{b_{ik}}\cdot\underline{S}_{b_{ik}}=\frac{\Delta t_{b_{ik}}S_{b_{ik}}}{\overline{I'F}}\left((1-B_{b,ik})P_I-\texttt{INC}\times A_{b,ik}\right) \tag{IV.J.2}$$

Les pas de temps aux faces Δt_{ij} et $\Delta t_{b_{ik}}$ sont calculés par interpolation par les sous-programmes `viscfa` (cas isotrope, `IPUCOU=0`) ou `visort` (cas anisotrope, `IPUCOU=1`).

Calcul avec reconstruction des non orthogonalités

Plusieurs discrétisations peuvent être proposées pour le traitement des non orthogonalités. Celle retenue dans le code est issue des premiers tests réalisés sur le prototype, et fait intervenir non seulement le pas de temps interpolé à la face, mais aussi les pas de temps dans chaque cellule. Il serait sans doute bon de revenir sur cette écriture et évaluer d'autres solutions. La forme utilisée pour les faces internes est :

$$-\Delta t_{ij}(\underline{\nabla}_f P)_{ij}\cdot\underline{S}_{ij} = \frac{\Delta t_{ij} S_{ij}}{\overline{I'J'}}(P_I - P_J) + (\underline{II'} - \underline{JJ'}).\frac{1}{2}\left[\Delta t_I(\underline{\nabla}P)_I + \Delta t_J(\underline{\nabla}P)_J\right]\frac{S_{ij}}{\overline{I'J'}} \quad \text{(IV.J.3)}$$

Pour les faces de bord, on écrit :

$$-\Delta t_{b_{ik}}(\underline{\nabla}_f P)_{b_{ik}}\cdot\underline{S}_{b_{ik}} = \frac{\Delta t_{b_{ik}} S_{b_{ik}}}{\overline{I'F}}\left[(1-B_{b,ik})(P_I + \underline{II'}.(\underline{\nabla}P)_I) - \texttt{INC}\times A_{b,ik}\right] \quad \text{(IV.J.4)}$$

Mise en œuvre

Les principaux arguments passés à `itrmas` et `itrgrp` sont la variable traitée `PVAR` (la pression), ses conditions aux limites, le pas de temps projeté aux faces[1] (`VISCF` et `VISCB`), le pas de temps au centre des cellules, éventuellement anisotrope (`VISELX`, `VISELY`, `VISELZ`). `itrmas` retourne les tableaux `FLUMAS` et `FLUMAB` (flux de masse aux faces) mis à jour. `itrgrp` retourne directement la divergence du flux de masse mis à jour, dans le tableau `DIVERG`.

• Initialisation
Si `INIT` vaut 1, les variables `FLUMAS` et `FLUMAB` ou `DIVERG` sont mises à zéro.

• Cas sans reconstruction
La prise en compte ou non des non orthogonalités est déterminée par l'indicateur `NSWRGR` de la variable traitée (nombre de sweeps de reconstruction des non orthogonalités dans le calcul des gradients), passé par la variable `NSWRGP`. Une valeur inférieure ou égale à 1 enclenche le traitement sans reconstruction. Des boucles sur les faces internes et les faces de bord utilisent directement les formules (IV.J.1) et (IV.J.2) pour remplir les tableaux `FLUMAS` et `FLUMAB` (`itrmas`) ou des variables de travail `FLUMAS` et `FLUMAB` qui servent à mettre à jour le tableau `DIVERG` (`itrgrp`).

À noter que les tableaux `VISCF` et `VISCB` contiennent respectivement $\frac{\Delta t_{ij} S_{ij}}{\overline{I'J'}}$ et $\frac{\Delta t_{b_{ik}} S_{b_{ik}}}{\overline{I'F}}$.

• Cas avec reconstruction
Après un appel à `GRDCEL` pour calculer le gradient cellule de pression, on remplit les tableaux `FLUMAS` et `FLUMAB` ou `DIVERG` là encore par une application directe des formules (IV.J.3) et (IV.J.4).

Points à traiter

Il est un peu redondant de passer en argument à la fois le pas de temps projeté aux faces et le pas de temps au centre des cellules. Il faudrait s'assurer de la réelle nécessité de cela, ou alors étudier des formulations plus simples de la partie reconstruction.

[1] Plus précisément, le pas de temps projeté aux faces, multiplié par la surface et divisé par $\overline{I'J'}$ ou $\overline{I'F}$, cf. `viscfa`

K- matrix routine

Fonction

Le but de ce sous-programme, appelé par `codits` et `cs_solve_equation_scalar`, est de construire la matrice de convection/diffusion, incluant les contributions adéquates des termes sources, intervenant dans le membre de gauche d'équations discrétisées telles que celle de la quantité de mouvement, d'une équation de convection diffusion d'un scalaire ou de modèle de turbulence.
Le type d'équation considérée est, pour la variable scalaire a :

$$\frac{\partial a}{\partial t} + \mathrm{div}\,(\,(\rho \underline{u})\,a) - \frac{\partial}{\partial x}\left(\beta\,\frac{\partial a}{\partial x}\right) = 0$$

La matrice ne s'applique qu'aux termes non reconstruits, les autres étant pris en compte au second membre et traités dans le sous-programme `bilsc2`. La partie convective, lorsqu'elle existe, est issue du schéma upwind pur, quelque soit le type de schéma convectif choisi par l'utilisateur. En effet, c'est, à l'heure actuelle, la seule façon d'obtenir un opérateur linéaire à diagonale dominante.

La matrice est donc associée à $\mathcal{EM}_{scal}$, opérateur agissant sur un scalaire a (inspiré de celui vectoriel $\mathcal{EM}$ défini dans `cs_solve_navier_stokes`) d'expression actuelle, pour tout cellule Ω_i de centre I :

$$\begin{aligned}\mathcal{EM}_{scal}(a,I) &= f_s^{imp}\ a_I \\ &+ \sum_{j\in Vois(i)} F_{ij}^{amont}((\rho\underline{u})^n, a) + \sum_{k\in\gamma_b(i)} F_{b_{ik}}^{amont}((\rho\underline{u})^n, a) \\ &- \sum_{j\in Vois(i)} D_{ij}^{NRec}(\beta, a) - \sum_{k\in\gamma_b(i)} D_{b_{ik}}^{NRec}(\beta, a)\end{aligned}$$

avec :
• f_s^{imp} le coefficient issu du terme instationnaire $\dfrac{\rho\,|\Omega_i|}{\Delta t}$, s'il y a lieu, et de l'implicitation de certains termes sources (contribution découlant de la prise en compte de la variation $\dfrac{\partial \rho}{\partial t}$ de la masse volumique ρ au cours du temps, diagonale du tenseur de pertes de charges par exemple...): cette initialisation est en fait effectuée en amont de ce sous-programme,
• F_{ij}^{amont} le flux numérique convectif scalaire décentré amont calculé à la face interne ij de la cellule Ω_i,
• $F_{b_{ik}}^{amont}$ le flux numérique convectif scalaire décentré amont associé calculé à la face de bord ik de la cellule Ω_i jouxtant le bord du domaine Ω,
• D_{ij}^{NRec} le flux numérique diffusif scalaire non reconstruit associé calculé à la face interne ij de la cellule Ω_i,
• $D_{b_{ik}}^{NRec}$ le flux numérique diffusif scalaire non reconstruit associé calculé à la face de bord ik de la cellule Ω_i jouxtant le bord du domaine Ω,
• $Vois(i)$ représente toujours l'ensemble des cellules voisines de Ω_i et $\gamma_b(i)$ l'ensemble des faces de bord de Ω_i.
Du fait de la résolution en incréments, a est un incrément et ses conditions aux limites associées sont donc de type Dirichlet homogène ou de type Neumann homogène.

See the programmers reference of the dedicated subroutine for further details.

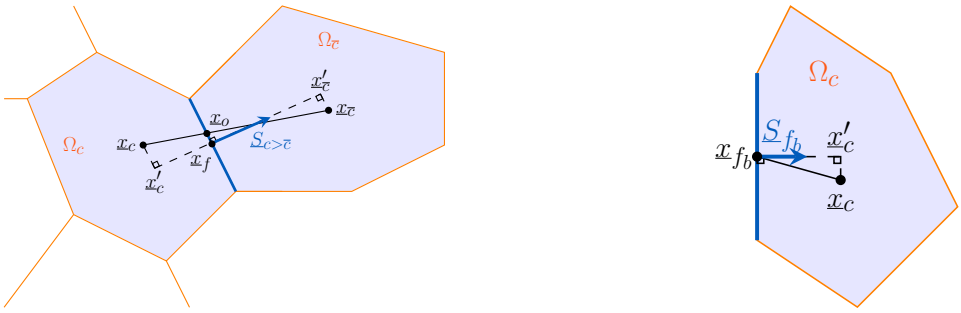


Figure IV.K.1: Définition des différentes entités géométriques pour les faces internes (gauche) et de bord (droite).

Discrétisation

L'opérateur $\mathcal{EM}_{scal}$ s'écrit, pour tout I centre de cellule :

$$\begin{aligned}\mathcal{EM}_{scal}(a,I) &= f_s^{imp}\, a_I \\ &+ \sum_{j\in Vois(i)} \left[(\rho\underline{u})^n_{ij}.\underline{S}_{ij}\right]\, a_{f,ij} + \sum_{k\in\gamma_b(i)} \left[(\rho\underline{u})^n_{b_{ik}}.\underline{S}_{b_{ik}}\right]\, a_{f\,b_{ik}} \\ &- \sum_{j\in Vois(i)} \beta_{ij}\frac{a_J - a_I}{\overline{I'J'}}S_{ij} - \sum_{k\in\gamma_b(i)} \beta_{b_{ik}}\frac{a_{b_{ik}} - a_I}{\overline{I'F}}S_{b_{ik}}\end{aligned} \qquad \text{(IV.K.1)}$$

où $a_{f,ij} = a_I$ ou a_J selon le signe de $(\rho\underline{u})^n_{ij}.\underline{S}_{ij}$ (schéma convectif upwind systématique), et avec $\overline{I'J'}$, mesure algébrique, orientée comme la normale sortante à la face, *i.e.* allant de I vers J pour la cellule Ω_i de centre I. On la notera $\overline{I'J'}^{\,I}$ lorsqu'on aura besoin d'expliciter clairement l'orientation.
$a_{f\,b_{ik}} = a_I$ ou $a_{\,b_{ik}}$ selon le signe de $(\rho\underline{u})^n_{b_{ik}}.\underline{S}_{b_{ik}}$ (schéma upwind systématique) et $a_{\,b_{ik}}$ valeur au bord est donnée directement par les conditions aux limites (valeur non reconstruite). $\overline{I'F}$, mesure algébrique, orientée relativement à la normale sortante à la face, *i.e.* allant de I vers l'extérieur du domaine.
En général, sauf cas pathologiques, les mesures algébriques $\overline{I'J'}$ et $\overline{I'F}$ sont positives et correspondent aux distances $I'J'$ et $I'F$. On se reportera au paragraphe Points à traiter pour plus de détails.
Soit $\underline{\underline{EM}}_{scal}$ la matrice associée ; sa taille est *a priori* de `NCEL*NCEL`, mais compte-tenu de la nature de la structure de données, seuls deux tableaux `DA(NCEL)` contenant les valeurs diagonales et `XA(NFAC,*)` les contributions des termes extra-diagonaux sont nécessaires, avec `NCEL` nombre de cellules du maillage considéré et `NFAC` nombre de faces internes associé.
Du fait des simplifications effectuées sur la matrice (non reconstruction des termes), les composantes extradiagonales de la ligne I ne sont différentes de zéro que pour les indices J des cellules voisines de I. On peut donc stocker toutes les contributions non nulles de la matrice dans deux tableaux `DA(NCEL)` et `XA(NFAC,2)` :
• `DA(I)` est le coefficient de la colonne I dans la ligne I,
• si `IFAC` est une face qui sépare les cellules Ω_i et Ω_j, orientée de I vers J, alors :
`XA(IFAC,1)` est le coefficient de la colonne J dans la ligne I et `XA(IFAC,2)` est le coefficient de la colonne I dans la ligne J. Lorsque la matrice est symétrique, *i.e.* lorsqu'il n'y a pas de convection (`ICONVP` $= 0$) et que seule la diffusion est à prendre en compte, alors `XA(IFAC,1)` = `XA(IFAC,2)` et on réduit `XA` à `XA(NFAC,1)`.

Soit m^n_{ij} ($m^n_{b_{ik}}$) la valeur de $(\rho\underline{u})^n_{ij}.\underline{S}_{ij}$ (respectivement $(\rho\underline{u})^n_{b_{ik}}.\underline{S}_{b_{ik}}$).
Alors :

- contribution volumique : $f_s^{imp}\, a_I$

- contribution d'une face purement interne ij

L'expression

$$+ \sum_{j \in Vois(i)} F_{ij}^{amont}((\rho \underline{u})^n, a) - \sum_{j \in Vois(i)} D_{ij}^{NRec}(\beta, a)$$

s'écrit :

$$\begin{aligned} &\sum_{j \in Vois(i)} \left(\left[(\rho \underline{u})^n_{ij} . \underline{S}_{ij} \right] \quad a_{f,ij} - \beta_{ij} \frac{a_J - a_I}{\overline{I'J'}} S_{ij} \right) \\ &= \sum_{j \in Vois(i)} \left[\frac{1}{2} (\, m^n_{ij} + | \, m^n_{ij} | \,) \, a_I + \frac{1}{2} (\, m^n_{ij} - | \, m^n_{ij} |) \, a_J \right] - \sum_{j \in Vois(i)} \beta_{ij} \frac{a_J - a_I}{\overline{I'J'}} S_{ij} \end{aligned} \tag{IV.K.2}$$

Ici, $\overline{I'J'} = \overline{I'J'}^{\,I}$.

■ contribution d'une face de bord ik

De même :

$$\begin{aligned} &\sum_{k \in \gamma_b(i)} F_{b_{ik}}^{amont}((\rho \underline{u})^n, a) - \sum_{k \in \gamma_b(i)} D_{b_{ik}}^{NRec}(\beta, a) \\ &= \sum_{k \in \gamma_b(i)} \left(\left[(\rho \underline{u})^n_{b_{ik}} . \underline{S}_{b_{ik}} \right] \; a_{f\,b_{ik}} - \beta_{b_{ik}} \frac{a_{b_{ik}} - a_I}{\overline{I'F}} S_{b_{ik}} \right) \\ &= \sum_{k \in \gamma_b(i)} \left[\frac{1}{2} (\, m^n_{b_{ik}} + | \, m^n_{b_{ik}} | \,) \, a_I + \frac{1}{2} (\, m^n_{b_{ik}} - | m^n_{b_{ik}} |) \, a_{b_{ik}} \right] - \sum_{k \in \gamma_b(i)} \beta_{b_{ik}} \frac{a_{b_{ik}} - a_I}{\overline{I'F}} S_{b_{ik}} \end{aligned} \tag{IV.K.3}$$

avec :

$$a_{b_{ik}} = \texttt{INC}\, A_{b,ik} + B_{b,ik}\, a_I = B_{b,ik}\, a_I$$

a n'étant associée qu'à des conditions aux limites de type Dirichlet homogène ou de type Neumann homogène.

Mise en œuvre

Initialisations

L'indicateur de symétrie `ISYM` de la matrice considérée est affecté comme suit :

• `ISYM` $= 1$, si la matrice est symétrique ; on travaille en diffusion pure , `ICONVP` $= 0$ et `IDIFFP` $= 1$,

• `ISYM` $= 2$, si la matrice est non symétrique ; on travaille soit en convection pure (`ICONVP` $= 1$, `IDIFFP` $= 0$), soit en convection/diffusion (`ICONVP` $= 1$, `IDIFFP` $= 1$).

Les termes diagonaux de la matrice sont stockés dans le tableau `DA(NCEL)`. Ceux extra-diagonaux le sont dans `XA(NFAC,1)` si la matrice est symétrique, dans `XA(NFAC,2)` sinon.

Le tableau `DA` est initialisé à zéro pour un calcul avec `ISTATP` $= 0$ (en fait, ceci ne concerne que les calculs relatifs à la pression). Sinon, on lui affecte la valeur `ROVSDT` comprenant la partie instationnaire, la contribution du terme continu en $-\ a\ \mathrm{div}\,(\rho \underline{u})^n$ et la partie diagonale des termes sources implicités. Le tableau `XA(NFAC,*)` est initialisé à zéro.

Calcul des termes extradiagonaux stockés dans `XA`

Ils ne se calculent que pour des faces purement internes `IFAC`, les faces de bord ne contribuant qu'à la diagonale.

matrice non symétrique (présence de convection)

Pour chaque face interne `IFAC`, les contributions extradiagonales relatives au terme a_I et à son voisin associé a_J sont calculées dans `XA(IFAC,1)` et `XA(IFAC,2)` respectivement (pour une face orientée de I

vers J).
On a les relations suivantes :

$$\begin{array}{ll} \texttt{XA(IFAC,1)} & = \texttt{ICONVP} * \texttt{FLUI} - \texttt{IDIFFP} * \texttt{VISCF(IFAC)} \\ \texttt{XA(IFAC,2)} & = \texttt{ICONVP} * \texttt{FLUJ} - \texttt{IDIFFP} * \texttt{VISCF(IFAC)} \end{array} \tag{IV.K.4}$$

avec `FLUMAS(IFAC)` correspondant à m_{ij}^n, `FLUI` à $\frac{1}{2}(\ m_{ij}^n - |\ m_{ij}^n|\ )$, `VISCF(IFAC)` à $\beta_{ij}\frac{S_{ij}}{\overline{I'J'}}$.

`XA(IFAC,1)` représente le facteur de a_J dans la dernière expression de (IV.K.2).

`FLUJ` correspond à $-\frac{1}{2}(\ m_{ij}^n + |\ m_{ij}^n|\ )$. En effet, `XA(IFAC,2)` est le facteur de a_I dans l'expression équivalente de la dernière ligne de (IV.K.2), mais écrite en J.
Ce qui donne :

$$\sum_{i\in Vois(j)} \left[\frac{1}{2}(\ m_{ji}^n + |\ m_{ji}^n|\ )\, a_J + \frac{1}{2}(\ m_{ji}^n - |\ m_{ji}^n|)\, a_I\right] - \sum_{i\in Vois(j)} \beta_{ji}\frac{a_I - a_J}{\overline{J'I'}} S_{ji} \tag{IV.K.5}$$

Le terme recherché est donc : $\frac{1}{2}(\ m_{ji}^n - |\ m_{ji}^n|\ ) - \beta_{ji}\frac{S_{ji}}{\overline{J'I'}}$.
Or :
$m_{ji}^n = -m_{ij}^n$ ($\underline{S}_{ji} = -\underline{S}_{ij}$ et $(\rho\underline{u})_{ji}^n = (\rho\underline{u})_{ij}^n$), avec $\overline{J'I'}$ mesure algébrique, orientée relativement à la normale sortante à la face, *i.e.* allant de J vers I. On la note $\overline{J'I'}^{\,J}$.
On a la relation :

$$\overline{J'I'}^{\,J} = \overline{I'J'}^{\,I} = (\ \overline{I'J'}) \tag{IV.K.6}$$

d'où la deuxième égalité dans (IV.K.4).

matrice symétrique (diffusion pure)

Pour chaque face interne `IFAC`, la contribution extradiagonale relative au terme a_I est calculée dans `XA(IFAC,1)` par la relation suivante :

$$\texttt{XA(IFAC,1)} = -\texttt{IDIFFP} * \texttt{VISCF(IFAC)} \tag{IV.K.7}$$

avec `VISCF(IFAC)` à $\beta_{ij}\frac{S_{ij}}{\overline{I'J'}}$.

Calcul des termes diagonaux stockés dans DA

matrice non symétrique (présence de convection)

Pour chaque face interne ij (`IFAC`) séparant les cellules Ω_i de centre I et Ω_j de centre J, `DA(II)` est la quantité en facteur de a_I dans la dernière expression de (IV.K.2), soit :

$$\frac{1}{2}(\ m_{ij}^n + |\ m_{ij}^n|\ ) + \beta_{ij}\frac{S_{ij}}{\overline{I'J'}} \tag{IV.K.8}$$

De même, pour `DA(JJ)`, on a :

$$\frac{1}{2}(\ -m_{ij}^n + |\ m_{ij}^n|\ ) + \beta_{ji}\frac{S_{ij}}{\overline{I'J'}} \tag{IV.K.9}$$

d'après l'expression de (IV.K.5) et l'égalité (IV.K.6).
L'implantation dans code_saturne associée est la suivante :
pour toute face `IFAC` d'éléments voisins `II` = `IFACEL(1,IFAC)` et `JJ` = `IFACEL(2,IFAC)`,
on ajoute à `DA(II)` la contribution croisée $-$`XA(IFAC,2)` (*cf.* (IV.K.8)) et à `DA(JJ)` la contribution $-$`XA(IFAC,1)` (*cf.* (IV.K.9)).

Prise en compte des conditions aux limites

Elles interviennent juste dans le tableau `DA`, compte-tenu de leur écriture et définition. Elles se calculent *via* la dernière expression de (IV.K.3). Pour chaque face `IFAC`, de l'élément de centre I, jouxtant le bord, on s'intéresse à :

$$\sum_{k\in\gamma_b(i)}\left[\frac{1}{2}(\ m^n_{b_{ik}}+|\ m^n_{b_{ik}}|\ )\,a_I+\frac{1}{2}(\ m^n_{b_{ik}}-|m^n_{b_{ik}}|)\,a_{\,b_{ik}}\right]-\sum_{k\in\gamma_b(i)}\beta_{b_{ik}}\frac{a_{\,b_{ik}}-a_I}{\overline{I'F}}S_{b_{ik}} \qquad \text{(IV.K.10)}$$

avec :

$$a_{\,b_{ik}}=B_{b,ik}\,a_I$$

soit :

$$\left(\sum_{k\in\gamma_b(i)}\left[\frac{1}{2}(\ m^n_{b_{ik}}+|\ m^n_{b_{ik}}|\ )+\frac{1}{2}(\ m^n_{b_{ik}}-|m^n_{b_{ik}}|)B_{b,ik}\right]+\sum_{k\in\gamma_b(i)}\beta_{b_{ik}}\frac{1-B_{b,ik}}{\overline{I'F}}S_{b_{ik}}\right)a_I \qquad \text{(IV.K.11)}$$

ce qui, pour le terme sur lequel porte la somme, se traduit par :
`ICONVP` $*$ ($-$`FLUJ` $+$ `FLUI` $*$ `COEFBP(IFAC)` $+$ `IDIFFP` $*$ `VISCB(IFAC)` $*$ ($1-$ `COEFBP(IFAC)`)
avec, $m^n_{b_{ik}}$ représenté par `FLUMAB(IFAC)` , $\frac{1}{2}\ (\ m^n_{b_{ik}}+|\ m^n_{b_{ik}}|\ )$ par `- FLUJ` ,
$\frac{1}{2}\ (\ m^n_{b_{ik}}-|m^n_{b_{ik}}|)B_{b,ik}$ par `FLUI` , $B_{b,ik}$ par `COEFBP(IFAC)`, $\beta_{b_{ik}}\frac{S_{b_{ik}}}{\overline{I'F}}$ par `VISCB(IFAC)`.

Décalage du spectre

Lorsqu'il n'existe aucune condition à la limite de type Dirichlet et que `ISTATP` $=0$ (c'est-à-dire pour la pression uniquement), on déplace le spectre de la matrice $\underline{\underline{EM}}_{scal}$ de `EPSI` (*i.e.* on multiplie chaque terme diagonal par (1 + `EPSI`)) afin de la rendre inversible. `EPSI` est fixé en dur dans `matrix` à 10^{-7}.

Points à traiter

• Initialisation

Le tableau `XA` est initialisé à zéro lorsqu'on veut annuler la contribution du terme en $\frac{\rho\,|\Omega_i|}{\Delta t}$, *i.e.* `ISTATP` $=0$. Ce qui ne permet donc pas la prise en compte effective des parties diagonales des termes sources à impliciter, décidée par l'utilisateur. Actuellement, ceci ne sert que pour la variable pression et reste donc *a priori* correct, mais cette démarche est à corriger dans l'absolu.

• Nettoyage

La contribution `ICONVP FLUI`, dans le calcul du terme `XA(IFAC,1)` lorsque la matrice est symétrique est inutile, car `ICONVP` $=0$.

• Prise en compte du type de schéma de convection dans $\underline{\underline{EM}}_{scal}$

Actuellement, les contributions des flux convectifs non reconstruits sont traitées par schéma décentré amont, quelque soit le schéma choisi par l'utilisateur. Ceci peut être handicapant. Par exemple, même sur maillage orthogonal, on est obligé de faire plusieurs sweeps pour obtenir une vitesse prédite correcte. Un schéma centré sans test de pente peut être implanté facilement, mais cette écriture pourrait, dans l'état actuel des connaissances, entraîner des instabilités numériques. Il serait souhaitable d'avoir d'autres schémas tout aussi robustes, mais plus adaptés à certaines configurations.

• Maillage localement pathologique

Il peut arriver, notamment au bord, que les mesures algébriques, $\overline{I'J'}$ ou $\overline{I'F}$ soient négatives (en cas de maillages non convexes par exemple). Ceci peut engendrer des problèmes plus ou moins graves : perte de l'existence et l'unicité de la solution (l'opérateur associé n'ayant plus les bonnes propriétés de régularité ou de coercivité), dégradation de la matrice (perte de la positivité) et donc résolution par solveur linéaire associé non approprié (gradient conjugué par exemple).
Une impression permettant de signaler et de localiser le problème serait souhaitable.

L- cs_solve_navier_stokes routine

Consideration is given to solving the unsteady, pressure-based, single-phase, three-dimensional Navier-Stokes system of equations for incompressible or weakly compressible flows based on first-order implicit Euler or second-order Crank-Nicolson time discretization with collocated finite volume spatial discretization.

See the programmers reference of the dedicated subroutine for further details.

Function

This subroutine is used to calculate, at a given time step, the velocity and pressure variables of the problem following an approach that proceeds in two steps based on a decomposition of operators (fractional step method).

The variables are assumed to be known at the instant t^n and a determination of their new values is sought at the instant[1] t^{n+1}. Therefore, the associated time step is $\Delta t^n = t^{n+1} - t^n$. To begin with, the velocity prediction step is performed by solving the momentum balance equation with an explicit pressure. This is followed by the pressure correction (or velocity projection) step allowing one to obtain a divergence-free velocity field.

The continuous equations are thus:

$$\left\{ \begin{array}{l} \dfrac{\partial}{\partial t}\left(\rho \underline{u}\right) + \underline{\mathrm{div}}\ \left(\rho \underline{u} \otimes \underline{u}\right) = \underline{\mathrm{div}}\ \left(\underline{\underline{\sigma}}\right) + \underline{TS} - \underline{\underline{K}} \cdot \underline{u} + \Gamma\left(\underline{u} - \underline{u}^{in}\right), \\ \dfrac{\partial \rho}{\partial t} + \mathrm{div}\ \left(\rho \underline{u}\right) = \Gamma, \end{array} \right. \tag{IV.L.1}$$

with ρ the density field, $\underline{u}$ the velocity field, $[\underline{TS} - \underline{\underline{K}}\,\underline{u}]$ the other source terms ($\underline{\underline{K}}$ is a symmetric positive definite tensor by definition), $\underline{\underline{\sigma}}$ the stress tensor, $\underline{\underline{\tau}}$ the tensor of viscous stresses, μ the dynamic viscosity (molecular and possibly turbulent), κ the volume viscosity (usually null and therefore neglected in the code and hereafter in this document, apart from the compressible module), $\underline{\underline{D}}$ the deformation rate tensor[2], Γ the mass source term and $\underline{u}^{in}$ is the velocity of the injection.

$$\left\{ \begin{array}{rcl} \underline{\underline{\sigma}} & = & \underline{\underline{\tau}} - P\underline{\underline{Id}}, \\ \underline{\underline{\tau}} & = & 2\mu\underline{\underline{D}} + (\kappa - \frac{2}{3}\mu) tr(\underline{\underline{D}})\underline{\underline{Id}}, \\ \underline{\underline{D}} & = & \frac{1}{2}\left(\underline{\underline{\nabla u}} + \underline{\underline{\nabla u}}^T\right). \end{array} \right. \tag{IV.L.2}$$

Recalling the definition of the notations employed[3]:

$$\left\{ \begin{array}{rcl} \left[\underline{\underline{\nabla a}}\right]_{ij} & = & \dfrac{\partial a_i}{\partial x_j}, \\ \left[\mathrm{div}\left(\underline{\underline{\sigma}}\right)\right]_i & = & \dfrac{\partial \sigma_{ij}}{\partial x_j}, \\ \left[\underline{a} \otimes \underline{b}\right]_{ij} & = & a_i\, b_j, \end{array} \right.$$

and thus:

$$\left[\underline{\mathrm{div}}\left(\underline{a} \otimes \underline{b}\right)\right]_i = \frac{\partial (a_i\, b_j)}{\partial x_j}$$

Remark L.1 *When accounting for variable density, the continuity equation is written:*

$$\frac{\partial \rho}{\partial t} + div\left(\rho\, \underline{u}\right) = \Gamma$$

[1] The pressure is assumed known at instant $t^{n-1+\theta}$ and its new value sought at $t^{n+\theta}$, with $\theta = 1$ or $1/2$ depending on the considered time-stepping scheme.

[2] Not to be confused, despite an identical notation D, with the diffusive fluxes $\underline{D}_{ij}$ and $\underline{D}_{b_{ik}}$ described later in this subroutine.

[3] Using the Einstein summation convention.

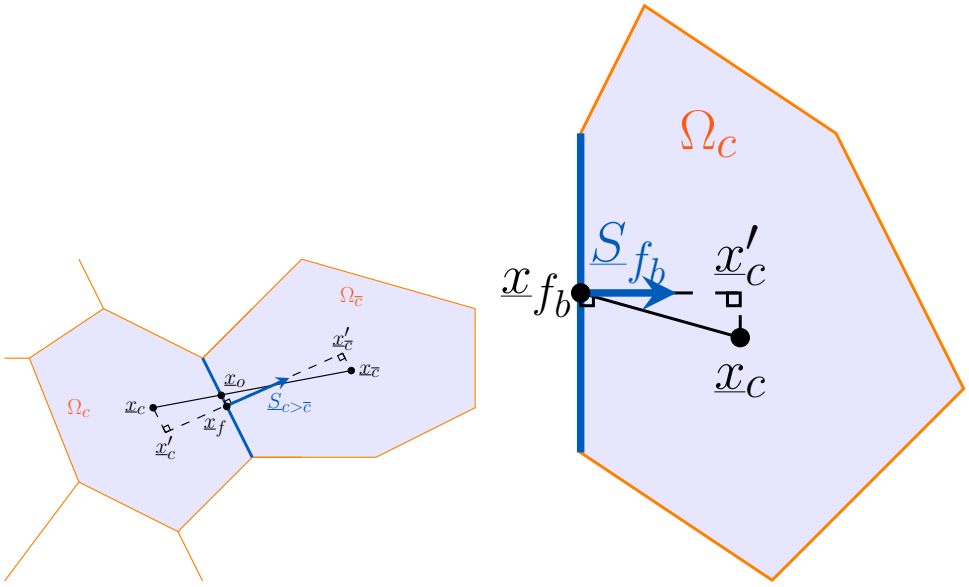
This equation is not taken into account in the projection step (we continue to resolve only $div(\rho\,\underline{u}) = \Gamma$*) whereas the term* $\dfrac{\partial\rho}{\partial t}$ *does appear during the velocity prediction step in the subroutine* `cs_velocity_prediction`*. If this term plays a significant role, the code_saturne compressible algorithm (which solves the full equation) is probably better adapted.*

Discretization

We define beforehand:

$$\alpha_{ij} = \frac{\overline{FJ'}}{\overline{I'J'}} \text{ defined at the internal faces only and}$$

$$\underline{u}_{K'} = \underline{u}_K + (\underline{\underline{\text{grad}}}\,\underline{u})_K . \underline{KK'} \text{ at first order in space, for } K = I \text{ or } J$$



(a) Internal face. (b) Boundary face.

Figure IV.L.1: Schema of the geometric entities defined for internal and boundary faces.

Fractional step method

Introduction

One of the methods used to numerically solve the Navier-Stokes equations consists of decomposing the corresponding operators into a set of simpler operators (which can then be treated by efficient algorithms) *via* the division of a single time step into a number of intermediate sub-steps. In this case, two sub-steps are used: the first takes up the convective, diffusive and source terms of the momentum equation and constitutes the velocity prediction step, the second treats the continuity equation and is designated as the pressure correction or velocity projection step.

Velocity prediction step

The discretization in time is achieved by applying a θ-scheme at time $n+\theta$ to the resolved variable, following the approach used for the transport equation of a scalar[4].

[4] cf. `cs_solve_equation_scalar`

The velocity at time $n+1$ not being available until after the projection step, a predicted velocity at time $n+1$ is used herein to interpolate:

$$\underline{\widetilde{u}}^{n+\theta} = \theta\,\underline{\widetilde{u}}^{n+1} + (1-\theta)\,\underline{u}^{n} \tag{IV.L.3}$$

With[5] :

$$\begin{cases} \theta = 1 & \text{For a first-order implicit Euler scheme,} \\ \theta = 1/2 & \text{For a second-order Crank-Nicolson scheme.} \end{cases} \tag{IV.L.4}$$

The predicted velocity field $\underline{\widetilde{u}}^{n+1}$ is then obtained by:

- Partial linearization of the convection operator giving rise to a decoupling of the velocity components.
- Explicitation of the pressure.
- Explicitation or extrapolation of the physical quantities (*i.e* ρ, μ, Cp ...) and the mass flux.
- Explicitation or extrapolation of the explicit source terms (for example injected mass $\Gamma\,\underline{u}^{in}$, ...) at time $n+\theta_S$.
- The implicit source terms that are linear with respect to the velocity (implicit user-specified source terms, the head losses $\underline{\underline{K}}\,\underline{u}$, sources of mass $\Gamma\,\underline{u}$, etc...) are assumed to be evaluated at time $n+\theta$.

For clarification, unless otherwise specified, the physical properties $\Phi = \rho$, μ ... and the mass flux $(\rho\underline{u})$ are assumed to be evaluated respectively at the instants $n+\theta_\Phi$ and $n+\theta_F$, with θ_Φ and θ_F being dependent on the specific time-stepping schemes employed for these quantities[6].

After rewriting the unsteady terms based on the mass conservation relation, we solve the following system:

$$\rho\left(\frac{\underline{\widetilde{u}}^{n+1} - \underline{u}^{n}}{\Delta t}\right) + \underline{\mathrm{div}}\left(\underline{\widetilde{u}}^{n+\theta} \otimes (\rho\underline{u})\right) = \underline{\mathrm{div}}\left(\underline{\underline{\sigma}}^{n+\theta}\right) + \underline{TS}^{n+\theta_S} - \underline{\underline{K}}^{n}\,\underline{u}^{n+\theta} + \underline{\widetilde{u}}^{n+\theta}\,\mathrm{div}\,(\rho\underline{u}), \tag{IV.L.5}$$

with:

$$\underline{\underline{\sigma}}^{n+\theta} = \mu\,\underline{\underline{\nabla}}\underline{\widetilde{u}}^{n+\theta} \underbrace{-P^{n-1+\theta}\underline{\underline{Id}} + \left(\mu\left(\underline{\underline{\nabla}}\underline{u}\right)^{n+\theta_S}\right)^{T} - \frac{2}{3}(\mu\,\mathrm{div}\,\underline{u})^{n+\theta_S}\underline{\underline{Id}}}_{\text{explicit source terms}}. \tag{IV.L.6}$$

Pressure correction step (or projection of velocities)

The predicted velocity has *a priori* non-zero divergence. The second step corrects the pressure by imposing the nullity of the steady constraint for the velocity computed at time instant t^{n+1}. We then solve:

$$\begin{cases} \dfrac{(\rho\underline{u})^{n+1} - (\rho\underline{\widetilde{u}})^{n+1}}{\Delta t} &= -\underline{\nabla}\delta P^{n+\theta}, \\ \mathrm{div}\,(\rho\underline{u})^{n+1} = \Gamma, \end{cases} \tag{IV.L.7}$$

where the pressure increment $\delta P^{n+\theta}$ is defined as:

$$\delta P^{n+\theta} = P^{n+\theta} - P^{n-1+\theta}. \tag{IV.L.8}$$

Remark L.2 *The ρ and μ quantities remain constant over the course of both steps. If there has been any variation in the interval, their values will be modified at the start of the next time step, after the scalars (temperature, mass fraction,...) have been updated.*

[5] In the $\theta = 1/2$ case, the time step Δt is assumed to be constant in time and uniform in space.

[6] cf. `introd`

Spatial discretization

Following the classical approach, the time-discretized equations are then integrated over the control volumes Ω_i (or cells).

Velocity prediction step

Second member

If we do not take account of the θ-scheme convection and diffusion terms, the explicit volume source terms of the equation (IV.L.5) for the system concerning the quantity $(\widetilde{\underline{u}}^{n+1} - \underline{u}^n)$ are expressed as:

$$-\underline{\nabla} P^{n-1+\theta} + \mathrm{div}\left[(\mu\left(\underline{\underline{\nabla}}\underline{u}\right)^T)^{n+\theta_S} - \frac{2}{3}(\mu \mathrm{div}\,\underline{u})^{n+\theta_S}\underline{\underline{Id}}\right] + \underline{TS}^{n+\theta_S} - \underline{\underline{K}}^n\underline{u}^n + \underline{u}^n \mathrm{div}\,(\rho\underline{u}).$$

To integrate these terms over a cell V_c, we multiply their local value at the cell centre by the volume of the cell.

Convection

After decomposition of $\widetilde{\underline{u}}$ based on the relation (IV.L.3), spatial integration of the convective parts of the velocity field $\theta\,\mathrm{div}\left(\widetilde{\underline{u}}^{n+1} \otimes (\rho\underline{u})\right)$ and $(1-\theta)\,\mathrm{div}\,(\underline{u}^n \otimes (\rho\underline{u}))$ yields a sum of the numerical fluxes $\underline{F}_{\,ij}$ calculated at the faces of the internal cells and $\underline{F}_{\,b_{ik}}$ calculated at the boundary faces of the domain Ω. Taking $Neigh(i)$ as the set of the centres of the neighbouring cells of Ω_i and $\gamma_b(i)$ the set of the centres of the boundary faces of Ω_i, we obtain:

$$\int_{\Omega_i} \mathrm{div}\,(\underline{u} \otimes (\rho\underline{u}))\,d\Omega = \sum_{j \in Neigh(i)} \underline{F}_{\,ij}((\rho\underline{u}),\underline{u}) + \sum_{k \in \gamma_b(i)} \underline{F}_{\,b_{ik}}((\rho\underline{u}),\underline{u}),$$

with:

$$\underline{F}_{\,ij}((\rho\underline{u}),\underline{u}) = \left[(\rho\underline{u})_{\,ij}.\,\underline{S}_{\,ij}\right]\underline{u}_{\,f,ij}. \tag{IV.L.9}$$

$$\underline{F}_{\,b_{ik}}((\rho\underline{u}),\underline{u}) = \left[(\rho\underline{u})_{\,b_{ik}}.\,\underline{S}_{\,b_{ik}}\right]\underline{u}_{\,f\,b_{ik}}. \tag{IV.L.10}$$

The value of the flux $\underline{F}_{\,ij}$ depends on the numerical scheme selected. Three different schemes are available in code_saturne:

i/ a first-order upwind scheme:
$\underline{F}_{\,ij}((\rho\underline{u}),\underline{u}) = \underline{F}_{ij}^{upwind}((\rho\underline{u}),\underline{u})$
where:

$$\begin{array}{lll} \underline{u}_{f_{c|\overline{c}}} = \underline{u}_c & \text{if} & (\rho\underline{u})_{f_{c|\overline{c}}} \cdot \underline{S}_{ij} \geqslant 0 \\ \underline{u}_{f_{c|\overline{c}}} = \underline{u}_{\overline{c}} & \text{if} & (\rho\underline{u})_{f_{c|\overline{c}}} \cdot \underline{S}_{ij} < 0, \end{array} \tag{IV.L.11}$$

ii/ a second-order centred scheme:
$\underline{F}_{\,ij}((\rho\underline{u}),\underline{u}) = \underline{F}_{ij}^{centred}((\rho\underline{u}),\underline{u})$
with:

$$\underline{u}_{f_{c|\overline{c}}} = \alpha_{ij}\underline{u}_{\underline{x}'_c} + (1-\alpha_{ij})\underline{u}_{\underline{x}'_{\overline{c}}} \text{ and } \underline{u}_{K'} = \underline{u}_K + (\underline{\underline{\nabla}}\underline{u})_K \cdot \underline{KK'} \text{ for } K = \underline{x}_c \text{ or } \underline{x}_{\overline{c}} \tag{IV.L.12}$$

iii/ a Second-Order Linear Upwind scheme (SOLU):
$\underline{F}_{\,ij}((\rho\underline{u}),\underline{u}) = \underline{F}_{\,ij}^{SOLU}((\rho\underline{u}),\underline{u})$
with:

$$\begin{array}{rl} \underline{u}_{\,f,ij} = & \underline{u}_c + (\underline{\underline{\nabla}}\underline{u})_{\underline{x}_c} \cdot \underline{x_c x_f} \text{ if } (\rho\underline{u})_{f_{c|\overline{c}}} \cdot \underline{S}_{ij} \geqslant 0, \\ & \underline{u}_{\overline{c}} + (\underline{\underline{\nabla}}\underline{u})_{\overline{c}} \cdot \underline{x_{\overline{c}} x_f} \text{ if } (\rho\underline{u})_{\,ij}.\underline{S}_{\,ij} < 0. \end{array} \tag{IV.L.13}$$

The value of $\underline{F}_{b_{ik}}$ is calculated as:

$$\begin{aligned} \underline{u}_{f\,b_{ik}} &= \underline{u}_I \text{ if } (\rho\underline{u})_{b_{ik}} \cdot \underline{S}_{b_{ik}} \geqslant 0 \\ &= \underline{u}_{b_{ik}} \text{ if } (\rho\underline{u})_{b_{ik}} \cdot \underline{S}_{b_{ik}} < 0 \end{aligned} \tag{IV.L.14}$$

with the boundary value $\underline{u}_{\,b_{ik}}$ obtained from the prescribed boundary conditions.

Remark L.3 *When a centred scheme is used, for the simple reason of numerical stability we actually write:*

$$\underline{u}_{f_{c|\overline{c}}} = \alpha_{ij}\underline{u}_c + (1-\alpha_{ij})\underline{u}_{\overline{c}} + \frac{1}{2}\left[(\underline{\underline{\nabla}}u)_c + (\underline{\underline{\nabla}}u)_{\overline{c}}\right] \cdot \underline{x_o x_f}$$

to preserve the order in space.

Remark L.4 *A slope test (which may introduce nonlinearities in the convection operator) allows switching between a second-order scheme and the first-order upwind scheme. Moreover, with standard models, we use at all points a value of the velocity field $\widetilde{\underline{u}}_{f_{c|\overline{c}}}$ derived from a barycentric mean between the upwind value and the second-order value (blending, specified by the user).*

Diffusion

Likewise, the diffusive components $\theta\ \mathrm{div}\,(\mu\,\underline{\underline{\mathrm{grad}}}\,\widetilde{\underline{u}}^{n+1})$ and $(1-\theta)\ \mathrm{div}\,(\mu\underline{\underline{\mathrm{grad}}}\,\underline{u}^n)$ are written:

$$\int_{V_c} \underline{\mathrm{div}}\,(\mu\underline{\underline{\nabla}}u)d\Omega = \sum_{j\in Neigh(i)} \underline{D}_{ij}(\mu,\underline{u}) + \sum_{k\in\gamma_b(i)} \underline{D}_{c>f_b}(\mu,\underline{u})$$

with:

$$\underline{D}_{ij}(\mu,\underline{u}) = \mu_{f_{c|\overline{c}}} \frac{\underline{u}_{\underline{x}'_{\overline{c}}} - \underline{u}_{\underline{x}'_c}}{\overline{\underline{x}'_c\underline{x}'_{\overline{c}}}} S_{ij} \tag{IV.L.15}$$

and:

$$\underline{D}_{c>f_b}(\mu,\underline{u}) = \mu_{f_b} \frac{\underline{u}_{f_b} - \underline{u}_{\underline{x}'_c}}{\overline{\underline{x}'_c\underline{x}_f}} S_{c>f_b} \tag{IV.L.16}$$

keeping the same notations employed beforehand and with the value $\underline{u}_{f_b}$ at the boundary provided directly by the specified boundary conditions.

The face value of the viscosity $\mu_{f_{c|\overline{c}}}$ is calculated from the cell centre values using a specified function f:

$$\mu_{f_{c|\overline{c}}} = f(\mu_c, \mu_{\overline{c}})$$

this function being either an arithmetic mean:

$$f(\mu_c,\mu_{\overline{c}}) = \frac{1}{2}(\mu_c + \mu_{\overline{c}}) \tag{IV.L.17}$$

or a geometric mean:

$$f(\mu_c,\mu_{\overline{c}}) = \frac{\mu_c\mu_{\overline{c}}}{\alpha_{ij}\mu_c + (1-\alpha_{ij})\mu_{\overline{c}}} \tag{IV.L.18}$$

and the viscosity value μ_{f_b} at the boundary face is given by the equality:

$$\mu_{f_b} = \mu_c \tag{IV.L.19}$$

We also introduce here, for later use, the following additional notations:

$$\underline{D}^{NRec}_{ij}(\mu,\underline{u}) = \mu_{ij} \frac{\underline{u}_{\underline{x}_{\overline{c}}} - \underline{u}_{\underline{x}_c}}{\overline{\underline{x}'_c\underline{x}'_{\overline{c}}}} S_{ij} \tag{IV.L.20}$$

$$\underline{D}^{NRec}_{c>f_b}(\mu,\underline{u}) = \mu_{f_b} \frac{\underline{u}_{f_b} - \underline{u}_{\underline{x}_c}}{\overline{\underline{x}'_c\underline{x}_f}} S_{c>f_b} \tag{IV.L.21}$$

which correspond to the non-reconstructed values at the internal and boundary faces, respectively.

Solution

As the system (IV.L.5) may contain nonlinearities owing to use of the slope test or may give rise *via* the (cell) gradient reconstruction to a nearly-full matrix when non orthogonalities are present, we solve the system using the iterative sequence $(\widetilde{\underline{u}}^{n+1,k})_{k\in\mathbb{N}}$ defined by:

$$\left\{\begin{array}{rcl} \widetilde{\underline{u}}^{n+1,0} & = & \underline{u}^n \\ \widetilde{\underline{u}}^{n+1,k+1} & = & \widetilde{\underline{u}}^{n+1,k} + \delta\widetilde{\underline{u}}^{k+1} \\ \mathcal{EM}(\delta\widetilde{\underline{u}}^{k+1},c) & = & -\mathcal{E}(\widetilde{\underline{u}}^{n+1,k},c) \end{array}\right. \tag{IV.L.22}$$

which also defines the sequence $(\delta\widetilde{\underline{u}}^{k+1})_{k\in\mathbb{N}}$.
The two operators $\mathcal{E}$ and $\mathcal{EM}$ have the respective expressions:

$$\begin{aligned} \mathcal{E}(\widetilde{\underline{u}},c) &= \theta\,\mathcal{J}(\widetilde{\underline{u}},c) + (1-\theta)\,\mathcal{J}(\underline{u}^n,c) + |\Omega_I|\frac{\rho_c}{\Delta t}(\widetilde{\underline{u}}_c - \underline{u}_c^n) \\ &+ |V_c|\left[(\underline{\underline{TS}})_c^{n+\theta_S} - (\underline{\nabla}P)_c^{n-1+\theta}\right] \\ &+ \sum_{\overline{c}\in Neigh(c)} \underbrace{\left((\mu\,(\underline{\underline{\nabla u}})^T)^{n+\theta_S} - \frac{2}{3}(\mu \mathrm{div}\ \underline{u})^{n+\theta_S}\underline{\underline{Id}}\right)_{f_{c|\overline{c}}}}_{\text{average or linear interpolation between } \underline{x}'_c \text{ and } \underline{x}'_{\overline{c}}} \cdot\underline{S}_{ij} \\ &+ \sum_{k\in\gamma_b(c)} \underbrace{\left((\mu\,(\underline{\underline{\nabla u}})^T)^{n+\theta_S} - \frac{2}{3}(\mu \mathrm{div}\,\underline{u})^{n+\theta_S}\underline{\underline{Id}}\right)_{f_b}}_{\text{obtained from the boundary conditions}} \cdot\underline{S}_{c>f_b} \end{aligned} \tag{IV.L.23}$$

with:

$$\begin{aligned} \mathcal{J}(\underline{v},c) &= |V_c|[\underline{\underline{K}}^n - \mathrm{div}\,(\rho\underline{u})\,]_c\ \underline{v}_c \\ &+ \left(\sum_{\overline{c}\in Neigh(c)} \underline{F}_{ij}((\rho\underline{u}),\underline{v}) + \sum_{k\in\gamma_b(c)} \underline{F}_{c>f_b}((\rho\underline{u}),\underline{v})\right) \\ &- \left(\sum_{\overline{c}\in Neigh(c)} \underline{D}_{ij}(\mu,\underline{v}) + \sum_{k\in\gamma_b(c)} \underline{D}_{c>f_b}(\mu,\underline{v})\right) \end{aligned}$$

$$\begin{aligned} \mathcal{EM}(\delta\underline{u},c) &= |V_c|\left(\frac{\rho_c}{\Delta t} + \theta\,[\underline{\underline{K}}^n - \mathrm{div}\,(\rho\underline{u})\,]_c\right)\ \delta\underline{u}_c \\ &+\theta\left(\sum_{\overline{c}\in Neigh(c)} \underline{F}_{ij}^{upwind}((\rho\underline{u}),\delta\underline{u}) + \sum_{k\in\gamma_b(c)} \underline{F}_{c>f_b}^{upwind}((\rho\underline{u}),\delta\underline{u})\right) \\ &-\theta\left(\sum_{\overline{c}\in Neigh(c)} \underline{D}_{ij}^{NRec}(\mu,\delta\underline{u}) + \sum_{k\in\gamma_b(c)} \underline{D}_{c>f_b}^{NRec}(\mu,\delta\underline{u})\right) \end{aligned}$$

Furthermore, we assume that the sequence $(\widetilde{\underline{u}}^{n+1,k})_{k\in\mathbb{N}}$ converges towards $\widetilde{\underline{u}}^{n+1}$.

Remark L.5 *The boundary conditions associated with the operators $\mathcal{E}$ and $\mathcal{EM}$ of the system (IV.L.23) are those relating to the velocity field $\underline{u}$. They are either homogeneous Dirichlet- or homogeneous Neumann-type conditions on $\delta\underline{u}$, this being dependent on whether the same type of condition, in this case a Dirichlet and a Neumann-type condition has been used for the velocity $\underline{u}$. They are mixed if a symmetry condition applies to a face that is skewed with respect to the reference frame axes.*

Remark L.6 *The first two summations of the type $(\sum_{k\in\gamma_b(c)})$, i.e. comprising the $\underline{F}_{c>f_b}((\rho\underline{u}),\underline{u})$ and $\underline{D}_{c>f_b}(\mu,\underline{u})$ terms, use the velocity boundary conditions.*
The volume term:

$$\sum_{\overline{c}\in Neigh(c)} \underbrace{\left((\mu\,(\underline{\underline{\nabla u}})^T)^{n+\theta_S} - \frac{2}{3}(\mu div\,\underline{u})^{n+\theta_S}\underline{\underline{Id}}\right)_{ij}}_{\text{average or linear interpolation between } \underline{x}'_c \text{ and } \underline{x}'_{\overline{c}}} \cdot\underline{S}_{ij}$$

The associated boundary term:

$$\sum_{k\in\gamma_b(c)} \underbrace{\left((\mu \left(\underline{\underline{grad}}\,\underline{u}\right)^T)^{n+\theta_S} - \frac{2}{3}(\mu\, div\,\underline{u})^{n+\theta_S}\,\underline{\underline{Id}} \right)_{f_b}}_{from\ the\ boundary\ conditions} \cdot \underline{S}_{c>f_b}$$

is treated in a specific way. In effect, the contribution of the first term (transposed gradient) is cancelled out for a cell V_c adjoining the boundary, no proper boundary condition being attributed to it at this time.

Remark L.7 *The operator $\mathcal{EM}$ approaches $\mathcal{E}$ (there is no reconstruction of terms and the convective term is systematically treated using an upwind scheme). This can introduce non-negligible numerical diffusion if the sequence $(\widetilde{\underline{u}}^{n+1,k})_{k\in\mathbb{N}}$ has not converged.*

Pressure correction step

Applying the divergence of the first equation of the system (IV.L.7), we obtain:

$$\text{div}\left[(\rho\underline{u})^{n+1} - (\rho\widetilde{\underline{u}})^{n+1}\right] = \text{div}\left(-\Delta t\,\underline{\nabla}\delta P^{n+\theta}\right) \tag{IV.L.24}$$

By applying the steady constraint $\text{div}\,(\rho\underline{u})^{n+1} = \Gamma$, this becomes:

$$\text{div}\left(\Delta t\,\underline{\nabla}\,\delta P^{n+\theta}\right) = \text{div}\left((\rho\widetilde{\underline{u}})^{n+1}\right) - \Gamma \tag{IV.L.25}$$

where:

$$\begin{cases} \text{div}\left(\Delta t\,\underline{\nabla}\,\delta P^{n+\theta}\right) = \text{div}\left((\rho\widetilde{\underline{u}})^{n+1}\right) - \Gamma \\ (\rho\underline{u})^{n+1} = (\rho\widetilde{\underline{u}})^{n+1} - \Delta t\,\underline{\nabla}\delta P^{n+\theta} \end{cases} \tag{IV.L.26}$$

and:

$$\underline{u}^{n+1} = \widetilde{\underline{u}}^{n+1} - \frac{\Delta t}{\rho}\,\underline{\nabla}\delta P^{n+\theta} \tag{IV.L.27}$$

Integrating over a cell gives the discrete equation:

$$\int_{V_c} \text{div}\left(\Delta t\underline{\nabla}\,\delta P^{n+\theta}\right) d\Omega = \sum_{\overline{c}\in Neigh(c)} \underline{D}_{ij}(\Delta t, \delta P^{n+\theta}) + \sum_{k\in\gamma_b(c)} \underline{D}_{c>f_b}(\Delta t, \delta P^{n+\theta}) \tag{IV.L.28}$$

and:

$$\int_{\Omega_i} \text{div}\,(\rho\widetilde{\underline{u}})^{n+1} d\Omega = \sum_{j\in Neigh(i)} \left[(\rho\widetilde{\underline{u}})^{n+1}_{f_{c|\overline{c}}} \cdot \underline{S}_{ij}\right] + \sum_{k\in\gamma_b(c)} \left[(\rho\widetilde{\underline{u}})^{n+1}_{f_b} \cdot \underline{S}_{c>f_b}\right] \tag{IV.L.29}$$

The formalism introduced previously for the integration of the diffusion term in the prediction step is also adopted here: the boundary conditions on the pressure gradient δP are either of the homogeneous Dirichlet or homogeneous Neumann type and correspond to the type of condition imposed on the pressure P, *i.e.* a Dirichlet or Neumann-type condition respectively.

Computation of the second member of the equation relating to the pressure increment.

Discretization of the sum $\sum_{\overline{c}\in Neigh(c)} \left[(\rho\widetilde{\underline{u}})^{n+1}_{f_{c|\overline{c}}} \cdot \underline{S}_{ij}\right] + \sum_{k\in\gamma_b(c)} \left[(\rho\widetilde{\underline{u}})^{n+1}_{f_b} \cdot \underline{S}_{c>f_b}\right]$ presents a particularity. The following choice denoted $[\ ]^{Init}$, for example, yields unsatisfactory results for a cell not touching the boundary with the discretization and numerical scheme here used, particularly for the following equation (IV.L.27):

$$(\rho\widetilde{\underline{u}})^{n+1}_{f_{c|\overline{c}}} \cdot \underline{S}_{ij} = \left[(\rho\widetilde{\underline{u}})^{n+1}_{f_{c|\overline{c}}} \cdot \underline{S}_{ij}\right]^{Init} = \left[\alpha_{ij}(\rho\widetilde{\underline{u}})^{n+1}_{\underline{x}'_c} + (1-\alpha_{ij})(\rho\widetilde{\underline{u}})^{n+1}_{\underline{x}'_{\overline{c}}}\right] \cdot \underline{S}_{ij} \tag{IV.L.30}$$

As for the numerical flux calculation in the centred scheme, we legitimately use the following approximation:

$$\begin{aligned}(\rho\widetilde{\underline{u}})_{f_{c|\overline{c}}}^{n+1} &= \alpha_{ij}\rho_c\widetilde{\underline{u}}_c^{n+1} + (1-\alpha_{ij})\rho_{\overline{c}}\widetilde{\underline{u}}_{\overline{c}}^{n+1} \\ &+\frac{1}{2}\left[(\underline{\underline{\nabla}}(\rho\widetilde{\underline{u}})^{n+1})_c + (\underline{\underline{\mathrm{grad}}}\,(\rho\widetilde{\underline{u}})^{n+1})_{\overline{c}}\right]\cdot\underline{x_o x_f}\end{aligned} \tag{IV.L.31}$$

which does not, in itself, pose a problem.

On the other hand, $\left[(\rho\widetilde{\underline{u}})_{f_{c|\overline{c}}}^{n+1}\right]^{Init}$ contains the term $\underline{G}_{cell,f_{c|\overline{c}}}^{n}$, effectively inherited from the prediction step, whose value is:

$$\underline{G}_{cel,f_{c|\overline{c}}}^{n-1+\theta} = \alpha_{ij}\,\underline{\nabla}P_{\underline{x}'_c}^{n-1+\theta} + (1-\alpha_{ij})\,\underline{\nabla}P_{\underline{x}'_{\overline{c}}}^{n-1+\theta}$$

However, on a regular orthogonal Cartesian mesh, with a velocity field $\widetilde{\underline{u}}^{n+1} = \underline{0}$ and a pressure field $P_I^{n-1+\theta} = (-1)^I$, we obtain $\underline{G}_{cel,f_{c|\overline{c}}}^{n-1+\theta} = 0$ and hence the pressure gradient $\delta P^{n+\theta} = 0$. In this case, the initial pressure anomaly can evidently never be corrected.

To fix this, we modify the $[\ ]^{Init}$ expression of $(\rho\widetilde{\underline{u}})_{ij}^{n+1}$ and $(\rho\widetilde{\underline{u}})_{f_b}^{n+1}$ by substituting the corrected value $[\ ]^{Corr}$:

$$\begin{aligned}(\rho\widetilde{\underline{u}})_{f_{c|\overline{c}}}^{n+1}\cdot\underline{S}_{ij} &= \left[(\rho\widetilde{\underline{u}})_{ij}^{n+1}\right]^{Corr}\cdot\underline{S}_{ij} \\ &= \left(\left[(\rho\widetilde{\underline{u}})^{n+1} - \beta(-\Delta t\underline{\nabla}P^{n-1+\theta})\right]_{f_{c|\overline{c}}}^{Init}\cdot\underline{S}_{ij} + \beta\,\left(-D_{ij}(\Delta t, P^{n-1+\theta})\right)\right)\end{aligned} \tag{IV.L.32}$$

with, for the inlet, symmetry (if any) or wall boundary conditions:

$$(\rho\widetilde{\underline{u}})_{f_b}^{n+1}\cdot\underline{S}_{c>f_b} = \left[(\rho\widetilde{\underline{u}})_{f_b}^{n+1}\right]^{Corr}\cdot\underline{S}_{c>f_b} = \rho_{f_b}\underline{u}_{f_b}^{n+1}\cdot\underline{S}_{c>f_b}$$

and for the outflow boundary conditions:

$$\begin{aligned}(\rho\widetilde{\underline{u}})_{f_b}^{n+1}\cdot\underline{S}_{c>f_b} &= \left[(\rho\underline{u})_{f_b}^{n+1}\right]^{Corr}\cdot\underline{S}_{c>f_b} \\ &= \left(\left[\rho_{f_b}\underline{u}_{f_b}^{n+1} - \beta(-\Delta t\underline{\nabla}P^{n-1+\theta})_{\underline{x}'_c}\right]\cdot\underline{S}_{c>f_b} + \beta\,\left(-D_{c>f_b}(\Delta t, P^{n-1+\theta})\right)\right)\end{aligned}$$

β is the Arakawa coefficient. When it has a value of 1 (default value), it is the weighting factor in the Rhie & Chow filter.

Remark L.8 *This approach can be generalized to other source terms of the same type, for example those in the $R_{ij}-\varepsilon$ model.*

Solution

To solve the equation (IV.L.28), which can lead, *via* reconstruction of the cell gradient, to a nearly-full matrix when non-orthogonalities are present, we construct a sequence $(\delta P^{n+1,k})_{k\in\mathbb{N}}$ defined by:

$$\begin{cases}\delta P^{n+\theta,0} = 0 \\ \delta P^{n+\theta,k+1} = \delta P^{n+\theta,k} + C_{relax}\delta(\delta P)^{n+\theta,k+1} \\ \mathcal{FM}(\delta(\delta P)^{n+\theta,k+1}, c) = \mathcal{F}(\delta P^{n+\theta,k}, c)\end{cases} \tag{IV.L.33}$$

which also defines the sequence $(\delta(\delta P)^{n+\theta,k+1})_{k\in\mathbb{N}}$.

The operators $\mathcal{F}$ and $\mathcal{FM}$ are expressed by:

$$\begin{aligned}\mathcal{F}(\delta P, c) &= \sum_{\overline{c}\in Neigh(c)}\left[D_{ij}(\Delta t, \delta P) - \left[(\rho\widetilde{\underline{u}})_{ij}^{n+1}\right]^{Corr}\right] \\ &+ \sum_{k\in\gamma_b(c)}\left[D_{c>f_b}(\Delta t^n, \delta P) - \left[(\rho\widetilde{\underline{u}})_{f_b}^{n+1}\right]^{Corr}\right] + \Gamma\end{aligned} \tag{IV.L.34}$$

and:

$$\mathcal{FM}(\delta(\delta P),c) = \sum_{\overline{c}\in Neigh(c)} \left[-D_{ij}^{NRec}(\Delta t,\delta(\delta P))\right] + \sum_{k\in\gamma_b(c)} \left[-D_{c>f_b}^{NRec}(\Delta t,\delta(\delta P))\right] \tag{IV.L.35}$$

respectively.

C_{relax} is a given coefficient of relaxation set to 1 by default. We assume that the sequence $(\delta P^{n+\theta,k})_{k\in\mathbb{N}}$ converges to $\delta P^{n+\theta}$.

The mass flux is updated at each iteration step, using $\delta(\delta P)$. At convergence, the updated mass flux equation is:

$$(\rho\underline{u})_{ij}^{n+1}\cdot\underline{S}_{ij} = \left[(\rho\widetilde{\underline{u}})_{f_{c|\overline{c}}}^{n+1}\right]^{Corr}\cdot\underline{S}_{ij} - \underline{D}_{ij}(\Delta t^n,\delta P^{n+\theta}) \tag{IV.L.36}$$

and we compute the new velocity field at the centre of the cells thanks to the equality:

$$\underline{u}^{n+1} = \widetilde{\underline{u}}^{n+1} - \frac{\Delta t}{\rho}\underline{\nabla}\,\delta P^{n+\theta} \tag{IV.L.37}$$

Remark L.9 *A specific treatment is implemented in order to ensure the conservation of mass expressing the mass budget for the fluxes computed at the cell faces is always verified to strictly and perfectly uphold at the end of the correction step, whether or not the sequence $(\delta P^{n+\theta,k+1})_{k\in\mathbb{N}}$ has reached convergence. The term $\delta P^{n+\theta,k_{end}+1}$, which is in effect the last term evaluated in the sequence, is given by:*

$$\mathcal{FM}(\delta(\delta P)^{n+\theta,k_{end}+1},c) = \mathcal{F}(\delta P^{n+\theta,k_{end}},c) \tag{IV.L.38}$$

Instead of updating the mass flux in the classical way, we compute it as follows:

$$(\rho\underline{u})_{ij}^{n+1}\cdot\underline{S}_{ij} = \left[(\rho\widetilde{\underline{u}})_{ij}^{n+1}\right]^{Corr}\cdot\underline{S}_{ij} - D_{ij}(\Delta t,\delta P^{n+\theta,k_{end}}) - D_{ij}^{NRec}(\Delta t,\delta(\delta P)^{n+\theta,k_{end}+1})$$

and:

$$(\rho\underline{u})_{c>f_b}^{n+1}\cdot\underline{S}_{c>f_b} = \left[(\rho\widetilde{\underline{u}})_{f_b}^{n+1}\right]^{Corr}\cdot\underline{S}_{c>f_b} - D_{c>f_b}(\Delta t,\delta P^{n+\theta,k_{end}}) - D_{c>f_b}^{NRec}(\Delta t,\delta(\delta P)^{n+\theta,k_{end}+1})$$

enabling to arrive at the formulation:

$$\begin{aligned}
&\sum_{j\in Neigh(c)}(\rho\underline{u})_{f_{c|\overline{c}}}^{n+1}\cdot\underline{S}_{ij} + \sum_{k\in\gamma_b(c)}(\rho\underline{u})_{f_b}^{n+1}\cdot\underline{S}_{c>f_b}\\
&\quad= \sum_{\overline{c}\in Neigh(c)}\left[(\rho\widetilde{\underline{u}})_{f_{c|\overline{c}}}^{n+1}\right]^{Corr}\cdot\underline{S}_{ij} + \sum_{k\in\gamma_b(c)}\left[(\rho\widetilde{\underline{u}})_{f_b}^{n+1}\right]^{Corr}\cdot\underline{S}_{c>f_b}\\
&\quad- \sum_{\overline{c}\in Neigh(c)} D_{ij}(\Delta t,\delta P^{n+\theta,k_{end}}) - \sum_{k\in\gamma_b(c)} D_{c>f_b}(\Delta t,\delta P^{n+\theta,k_{end}})\\
&\quad- \sum_{\overline{c}\in Neigh(c)} D_{ij}^{NRec}(\Delta t,\delta(\delta P)^{n+\theta,k_{end}+1}) - \sum_{k\in\gamma_b(c)} D_{c>f_b}^{NRec}(\Delta t,\delta(\delta P)^{n+\theta,k_{end}+1})
\end{aligned}$$

corresponding to:

$$\sum_{\overline{c}\in Neigh(c)}(\rho\underline{u})_{f_{c|\overline{c}}}^{n+1}\cdot\underline{S}_{ij} + \sum_{k\in\gamma_b(c)}(\rho\underline{u})_{f_b}^{n+1}\cdot\underline{S}_{c>f_b} = -\mathcal{F}(\delta P^{n+\theta,k_{end}},c) + \mathcal{FM}(\delta(\delta P)^{n+\theta,k_{end}+1},c) + \Gamma$$

and thus:

$$\int_{V_c} div\,(\rho\underline{u})^{n+1}d\Omega = \Gamma$$

Implementation

Refer to the following appendices relating to the subroutines `cs_velocity_prediction` (prediction of velocities) and `Resopv` (pressure correction).

M- cs_velocity_prediction routine

Function

This subroutine implements the prediction step for the velocity field $\underline{u}$. This consists of solving the momentum equation by treating the pressure explicitly. Once the pressure correction step has been performed in the subroutine `resopv`, the velocity-pressure solution is obtained using the mass conservation law:

$$\frac{\partial \rho}{\partial t} + \mathrm{div}\,(\rho \underline{u}) = \Gamma, \tag{IV.M.1}$$

where Γ is the mass source term[1].
The Reynolds-averaged momentum conservation equation obtained by applying the fundamental theorem of dynamics is:

$$\frac{\partial (\rho \underline{u})}{\partial t} + \mathrm{div}\,(\rho \underline{u} \otimes \underline{u}) = \mathrm{div}\,(\underline{\underline{\sigma}}) + \underline{S} - \mathrm{div}\,(\rho \underline{\underline{R}}) \tag{IV.M.2}$$

where:

$$\underline{\underline{\sigma}} = -p\underline{\underline{Id}} + \underline{\underline{\tau}} \tag{IV.M.3}$$

with the following linear relationship for the Newtonian flows:

$$\begin{aligned} \underline{\underline{\tau}} &= 2\ \mu\ \underline{\underline{D}} + \ \lambda\ tr(\underline{\underline{D}})\ \underline{\underline{Id}} \\ \underline{\underline{D}} &= \frac{1}{2}\ (\underline{\underline{\mathrm{grad}}}\,\underline{u} + \ {}^{t}\underline{\underline{\mathrm{grad}}}\,\underline{u}) \end{aligned} \tag{IV.M.4}$$

$\underline{\underline{\sigma}}$ represents the stress tensor, $\underline{\underline{\tau}}$ the viscous stress tensor, μ the dynamic viscosity (molecular and potentially turbulent),$\underline{\underline{D}}$ the rate of deformation tensor[2], $\underline{\underline{R}}$ the Reynolds which appears when applying the average operator to the simultaneous equation, $\underline{S}$ the source terms.
λ is the second coefficient of viscosity. It is related to the volume viscosity κ via the expression

$$\lambda = \kappa - \frac{2}{3}\mu \tag{IV.M.5}$$

When the Stokes hypothesis holds, the volume viscosity κ is zero, in other words $3\lambda + 2\mu = 0$. This hypothesis is implicit in the code and in the rest of this, except in the compressible module.

The equation for the conservation of momentum is finally written as:

$$\begin{aligned} \rho \frac{\partial \underline{u}}{\partial t} = & - \underbrace{\mathrm{div}\,(\rho \underline{u} \otimes \underline{u})}_{\text{convection}} + \underbrace{\mathrm{div}\,(\mu\ \underline{\underline{\mathrm{grad}}}\,\underline{u})}_{\text{diffusion}} \\ & \underbrace{+\ \mathrm{div}\,(\mu\ {}^{t}\underline{\underline{\mathrm{grad}}}\,\underline{u})}_{\text{transpose of the velocity gradient term}} \quad \underbrace{-\ \frac{2}{3}\ \underline{\nabla}(\mu\ \mathrm{div}\,\underline{u})}_{\text{secondary viscosity}} \quad -\mathrm{div}\,(\rho \underline{\underline{R}}) - \ \underline{\nabla}(p) + (\rho - \rho_0)\,\underline{g} + \underline{u}\,\mathrm{div}\,(\rho\,\underline{u}) \\ & + \quad \underbrace{\Gamma(\underline{u}_i - \underline{u})}_{\text{mass source term}} \quad - \quad \underbrace{\rho\ \underline{\underline{K}}\underline{u}}_{\text{head loss}} \quad + \underbrace{\underline{T}_s^{exp} + T_s^{imp}\ \underline{u}}_{\text{other source terms}} \end{aligned} \tag{IV.M.6}$$

with p denoting the difference in pressure relative to the reference hydrostatic pressure (the real hydrostatic pressure being calculated with the density ρ and not with ρ_0):

$$p = p^* - \rho_0\ \underline{g} \cdot \underline{X} \tag{IV.M.7}$$

[1] In $kg.m^{-3}.s^{-1}$

[2] Not to be confused, despite the same notation D, with the diffusive fluxes described in the subroutine `cs_solve_navier_stokes`

($\underline{X}$ being the vector of components x, y and z).
μ_t, $\underline{\underline{K}}$, $\underline{u}_i$ represent respectively the turbulent dynamic viscosity, the tensor related to head losses and the value of the variable associated with the mass source.
The divergence of the Reynolds stress tensor is expressed by:

$$-\mathrm{div}\,(\rho\,\underline{\underline{R}}) = \begin{cases} \underline{0} & \text{in laminar,} \\ -\frac{2}{3}\,\underline{\nabla}(\mu_t\ \mathrm{div}\,\underline{u}) + \mathrm{div}\,(\mu_t\ (\underline{\underline{\mathrm{grad}}}\,\underline{u} + {}^t\underline{\underline{\mathrm{grad}}}\,\underline{u})) - \frac{2}{3}\,\underline{\nabla}(\rho\,k) & \text{for turbulent} \\ -\mathrm{div}\,(\rho\,\underline{\underline{R}}) & \text{viscosity,} \\ & \text{for second-order} \\ & \text{models,} \\ -\frac{2}{3}\,\underline{\nabla}(\mu_t\ \mathrm{div}\,\underline{u}) + \mathrm{div}\,(\mu_t\ (\underline{\underline{\mathrm{grad}}}\,\underline{u} + {}^t\underline{\underline{\mathrm{grad}}}\,\underline{u})) & \text{in LES} \end{cases} \tag{IV.M.8}$$

The mass source term involves terms corresponding to the local velocity field $\underline{u}$ as well as a velocity field $\underline{u}_i$ associated with the mass injected (or extracted). When $\Gamma < 0$, we remove mass from the system thereby obtaining $\underline{u}_i = \underline{u}$. The mass term is zero (*i.e.* $\Gamma(\underline{u}_i - \underline{u}) = \underline{0}$). When $\Gamma > 0$, a non-zero mass term if $\underline{u}_i \neq \underline{u}$. All terms appearing in the conservation of momentum equation, apart from the convection and diffusion terms, are computed in this subroutine and transmitted to the subroutine `codits` which solves the full equation (convection-diffusion with source terms).

See the programmers reference of the dedicated subroutine for further details.

Discretization

The convective term in $\mathrm{div}\,(\underline{u}\otimes\rho\,\underline{u})$ introduces a nonlinearity and coupling of the components of the velocity $\underline{u}$ in the equation (IV.M.6). The three velocity components are linearized and decoupled during discretization of the momentum equation in this prediction step.
For instance, taking:

$$\underline{\widetilde{u}} = \underline{u}^n + \delta\underline{u} \tag{IV.M.9}$$

The exact contribution of the convective term to take into account in the prediction step would be:

$$\mathrm{div}\,(\underline{\widetilde{u}}\otimes\rho\,\underline{\widetilde{u}}) = \mathrm{div}\,(\underline{u}^n\otimes\rho\,\underline{u}^n) + \mathrm{div}\,(\delta\underline{u}\otimes\rho\,\underline{u}^n) + \underbrace{\mathrm{div}\,(\underline{u}^n\otimes\rho\,\delta\underline{u})}_{\text{linear coupling term}} + \underbrace{\mathrm{div}\,(\delta\underline{u}\otimes\rho\,\delta\underline{u})}_{\text{nonlinear coupling term}} \tag{IV.M.10}$$

The last two terms of the expression (IV.M.10) are *a priori* neglected so as to obtain a velocity system that is decoupled and thereby avoid the inversion of a potentially very large matrix. These two terms can nevertheless be taken into account in an approximate manner by explicit extrapolation of the mass flux in $n+\theta_F$ (for the linear coupling term arising from the convection of $\underline{u}^n$ by $\delta\underline{u}$) and by applying a fixed-point subiteration over the subroutine `cs_solve_navier_stokes` (for the nonlinear term) implemented by specifying `NTERUP` > 1.

The equation (IV.M.6) is discretized at the time level $n+\theta$ using a θ-scheme and the tensor of velocity head losses decomposed into the sum of its diagonal $\underline{\underline{K}}_d$ and extra diagonal $\underline{\underline{K}}_e$ terms (such that $\underline{\underline{K}} = \underline{\underline{K}}_d + \underline{\underline{K}}_e$).
• The pressure is assumed to be known at the point in time $n-1+\theta$ (pressure/velocity time lag) and the gradient evidently calculated at this instant.
• The secondary viscosity and the gradient transpose source terms, those coming from the turbulence model[3], $\rho\,\underline{\underline{K}}_e\,\underline{u}$, $(\rho-\rho_0)\underline{g}$ as well as $\underline{T}_s^{exp}$ and $\Gamma\,\underline{u}_i$ are evaluated explicitly at time n or else extrapolated according to the time scheme selected for the physical properties and the source terms.
• The source terms $\underline{u}\ \mathrm{div}\,(\rho\,\underline{u})$, $\Gamma\ \underline{u}$, $T_s^{imp}\ \underline{u}$ and $-\rho\,\underline{\underline{K}}_d\,\underline{u}$ are implicit and calculated at instant $n+\theta$.
• The diffusion term $\mathrm{div}\,(\mu_{tot}\,\underline{\underline{\mathrm{grad}}}\,\underline{u})$ is implicitized: the velocity is taken at the instant $n+\theta$ and the viscosity either explicitized or extrapolated.

[3] with the exception of $\mathrm{div}\,(\mu_t\ (\underline{\underline{\mathrm{grad}}}\,\underline{u}))$

• Lastly, the convection term in $\mathrm{div}\,(\,\underline{u}\otimes(\rho\underline{u})\,)$ is treated implicitly: the resolved velocity component is taken at $n+\theta$ and the mass flux determined explicitly or extrapolated at $n+\theta_F$.

For clarification, unless explicitly stated otherwise the physical properties Φ (ρ, μ_{tot}, ...) and the mass flux ($\rho\underline{u}$) are to be evaluated respectively at time steps $n+\theta_\Phi$ and $n+\theta_F$, with θ_Φ and θ_F dependent upon the specific time schemes selected for these quantities[4].

The time-discrete form of the momentum equation (IV.M.6) then reads:

$$\begin{aligned}
\rho\,\frac{\widetilde{\underline{u}}^{n+1}-\underline{u}^n}{\Delta t}+\mathrm{div}\,(\,\widetilde{\underline{u}}^{n+\theta}\otimes(\rho\underline{u})\,)-\mathrm{div}\,(\mu_{\,tot}\,\underline{\underline{\mathrm{grad}}}\,\widetilde{\underline{u}}^{n+\theta})=\\
-\underline{\nabla}p^{n-1+\theta}+\mathrm{div}\,(\rho\,\underline{u})\,\widetilde{\underline{u}}^{n+\theta}+(\Gamma\,\underline{u}_{\,i})^{n+\theta_S}-\Gamma^n\,\widetilde{\underline{u}}^{n+\theta}\\
-\rho\,\underline{\underline{K}}_d^{\,n}\,\widetilde{\underline{u}}^{n+\theta}-(\rho\,\underline{\underline{K}}_{\,e}\,\underline{u})^{n+\theta_S}+(\underline{T}_s^{\,exp})^{\,n+\theta_S}+T_s^{\,imp}\,\widetilde{\underline{u}}^{n+\theta}\\
+[\mathrm{div}\,(\mu_{\,tot}\,{}^t\underline{\underline{\mathrm{grad}}}\,\underline{u})]^{n+\theta_S}-\frac{2}{3}[\,\underline{\nabla}(\mu_{\,tot}\,\mathrm{div}\,\underline{u})]^{n+\theta_S}+(\rho-\rho_0)\underline{g}-(\underline{turb})^{n+\theta_S}
\end{aligned}\tag{IV.M.11}$$

where, for simplification, the following definitions have been set:

$$\mu_{\,tot}=\begin{cases}\mu+\mu_t & \text{for turbulent viscosity or LES models,}\\ \mu & \text{for second order models or in laminar regimes}\end{cases}\tag{IV.M.12}$$

and:

$$\underline{turb}^n=\begin{cases}\dfrac{2}{3}\underline{\nabla}(\rho^n\,k^n) & \text{for turbulent viscosity models,}\\ \mathrm{div}\,(\rho^n\,\underline{\underline{R}}^n) & \text{for second order models,}\\ 0 & \text{in laminar or LES}\end{cases}\tag{IV.M.13}$$

By analogy with the θ-scheme equation for a scalar variable, $\widetilde{\underline{u}}^{n+\theta}$ is interpolated from the predicted velocity $\widetilde{\underline{u}}^{n+1}$ in the following manner[5] :

$$\widetilde{\underline{u}}^{n+\theta}=\theta\,\widetilde{\underline{u}}^{n+1}+(1-\theta)\,\underline{u}^n\tag{IV.M.14}$$

With:

$$\begin{cases}\theta=1 & \text{For a first-order implicit Euler scheme.}\\ \theta=1/2 & \text{For a second-order Crank-Nicolson scheme.}\end{cases}\tag{IV.M.15}$$

Using the above interpolation function, (IV.M.11) is then rewritten in the following form:

$$\begin{aligned}
&\underbrace{\left(\frac{\rho}{\Delta t}-\theta\,\mathrm{div}\,(\rho\,\underline{u})+\theta\;\Gamma^n+\theta\;\rho\,\underline{\underline{K}}_d^{\,n}-\theta\,T_s^{\,imp}\right)}_{f_s^{imp}}(\widetilde{\underline{u}}^{n+1}-\underline{u}^n)\\
&+\theta\,\mathrm{div}\,(\widetilde{\underline{u}}^{n+1}\otimes(\rho\underline{u}))-\theta\,\mathrm{div}\,(\mu_{\,tot}\,\underline{\underline{\mathrm{grad}}}\,\widetilde{\underline{u}}^{n+1})=\\
&-(1-\theta)\,\mathrm{div}\,(\underline{u}^n\otimes(\rho\underline{u}))+(1-\theta)\,\mathrm{div}\,(\mu_{\,tot}\,\underline{\underline{\mathrm{grad}}}\,\underline{u}^n)\\
f_s^{exp}\Bigg\{\;&-\underline{\nabla}p^{n-1+\theta}+\mathrm{div}\,(\rho\,\underline{u})\,\underline{u}^n+(\,\Gamma^n\,\underline{u}_{\,i}\,)^{n+\theta_S}-\Gamma^n\,\underline{u}^n\\
&-(\,\rho\,\underline{\underline{K}}_{\,e}\,\underline{u}\,)^{n+\theta_S}-\rho\,\underline{\underline{K}}_d^{\,n}\,\underline{u}^n+(\underline{T}_s^{\,exp})^{\,n+\theta_S}+T_s^{\,imp}\,\underline{u}^n\\
&+[\mathrm{div}\,(\mu_{\,tot}\,{}^t\underline{\underline{\mathrm{grad}}}\,\underline{u}\,)]^{n+\theta_S}-\frac{2}{3}[\,\underline{\nabla}(\mu_{\,tot}\,\mathrm{div}\,\underline{u}\,)]^{n+\theta_S}+(\rho-\rho_0)\underline{g}-(\underline{turb})^{n+\theta_S}
\end{aligned}\tag{IV.M.16}$$

from which the equation solved by the subroutine `codits` is obtained:

$$\begin{aligned}
f_s^{\,imp}(\widetilde{\underline{u}}^{n+1}-\underline{u}^n)+\theta\,\mathrm{div}\,(\widetilde{\underline{u}}^{n+1}\otimes(\rho\underline{u}))-\theta\,\mathrm{div}\,(\,\mu_{\,tot}\,\underline{\underline{\mathrm{grad}}}\,\widetilde{\underline{u}}^{n+1})=\\
-(1-\theta)\,\mathrm{div}\,(\underline{u}^n\otimes(\rho\underline{u}))+(1-\theta)\,\mathrm{div}\,(\,\mu_{\,tot}\,\underline{\underline{\mathrm{grad}}}\,\underline{u}^n)+\underline{f}_{\,s}^{\,exp}
\end{aligned}\tag{IV.M.17}$$

[4] cf. `introd`

[5] if $\theta=1/2$, or a time extrapolation is used, second-order is only obtained provided the time step Δt is constant and spatially uniform.

The spatial discretization scheme is elaborated further in the section relating to the subroutine `codits`.

REMARKS:
■ In the standard case without extrapolation, the term $-T_s^{imp}$ is only added to the coefficient f_s^{imp} if it is positive so as not to weaken the diagonal dominance of the matrix to invert.
■ If, on the other hand, an extrapolation is used, T_s^{imp} is added to f_s^{imp} whatever its sign. In effect, the intuitive idea which consists of taking:

$$\begin{cases} (\underline{T}_s^{exp} + T_s^{imp}\,\underline{u})^{\,n+\theta_S} & \text{if } T_s^{imp} > 0 \\ (\underline{T}_s^{exp})^{\,n+\theta_S} + T_s^{imp}\,\underline{u}^{n+\theta} & \text{if not} \end{cases} \tag{IV.M.18}$$

leads to inconsistency (loss of coherence) in the numerical treatment of the problem should T_s^{imp} changes sign between two time steps.
■ The diagonal part $\underline{\underline{K}}_d$ of the head loss tensor term is used in f_s^{imp}. Strictly speaking, only the positive contributions should in fact be retained (as pointed out in the associated user subroutine `uskpdc`). The approach currently in use requires improvement.
■ The term $\theta\,\Gamma^n - \theta\,\mathrm{div}\,(\rho\,\underline{u})$ is not problematic for the diagonal dominance of the matrix as it is exactly counterbalanced by the convection term (cf. `cs_solve_equation_scalar`).

Implementation

The conservation of momentum equation is solved in a decoupled manner. The integration of the different terms has therefore been carried out in order to treat separately the equations obtained for each component of the velocity.
For each of these components, the second member f_s^{exp} of the system (IV.M.16), the implicit terms of the system (with the exception of the convection-diffusion terms) and the term representing the total viscosity at the internal[6] and boundary faces is calculated in the subroutine `cs_velocity_prediction`. These terms are then transmitted to the subroutine `codits` which constructs and then solves the resulting complete system of equations, with the convection-diffusion terms, for each component of the velocity.

The normalized residual for convergence of the solution of the system for the pressure correction (`resopv`) is calculated in `cs_velocity_prediction`. It is defined as the norm of magnitude

$$\mathrm{div}\,(\rho\,\widetilde{\underline{u}}^{n+1} + \Delta t \underline{\nabla} P^{n-1+\theta}) - \Gamma$$

integrated over each cell `IEL` of the mesh (Ω_{iel}) or, symbolically, as the square root of the sum over the mesh cells of the quantity

$$\texttt{XNORMP(IEL)} = \int_{\Omega_{iel}} [\mathrm{div}\,(\rho\,\widetilde{\underline{u}}^{n+1} + \Delta\,t\,\underline{\nabla} P^{n-1+\theta}) - \Gamma\,]\,d\Omega.$$

It represents the second member of the equation system that would be solved for the pressure if the pressure gradient were not taken into account in the velocity prediction step. Note that, if the second member of the pressure increment equation were to be used directly, a normalized residual tending to zero would be obtained for a stationary calculation that had been run to convergence. This result would be penalizing and of little use for the computations.

At the start of `cs_velocity_prediction`, $\widetilde{\underline{u}}^{n+1}$ is not yet available and it is not possible therefore to calculate the total normalized residual. On the other hand, calculating the total residual at the end of `cs_velocity_prediction` is undesirable as this would require monopolising a working array to store

[6] value required for the integration of the diffusion term in `codits`, $(\mu_{\,tot})_{ij}\,\frac{\texttt{SURFN}}{\texttt{DIST}}$

the pressure gradient for the duration of the subroutine `cs_velocity_prediction`. The calculation of the normalized residual is therefore carried out in two stages.

The quantity $\int\limits_{\Omega_{iel}} \mathrm{div}\,(\Delta t \underline{\nabla} P^{n-1+\theta}) - \Gamma d\Omega$ is calculated at the beginning of `cs_velocity_prediction` and the complement $\int\limits_{\Omega_{iel}} \mathrm{div}\,(\rho \widetilde{\underline{u}}^{n+1}) d\Omega$ added at the end of `cs_velocity_prediction`.

The pressure gradient at the cells is thus first computed at instant $n-1+\theta$ by a call to `grdcel`. The subroutine `inimas` is then used to evaluate $\Delta t\, S \underline{\nabla} P^{n-1+\theta} \cdot \underline{n}$ at the faces (of surface S and normal $\underline{n}$). On entry into `inimas`, the working array `TRAV` contains $\frac{\Delta t}{\rho} \underline{\nabla} P^{n-1+\theta}$; when exiting this subroutine, the `VISCF` and `VISCB` arrays contain the value of $\Delta t\, S \underline{\nabla} P^{n-1+\theta} \cdot \underline{n}$ at the internal and boundary faces respectively.

We then use `divmas` takes the value of $\int\limits_{\Omega_{iel}} \mathrm{div}\,(\Delta t \underline{\nabla} P^{n-1+\theta}) d\Omega$ at the cells from the contents of `VISCF` and `VISCB` and places it into the array `XNORMP`.

Lastly, the contribution $\int\limits_{\Omega_{iel}} -\Gamma^n d\Omega$ of the mass source term is added to `XNORMP`.

For $\rho \widetilde{\underline{u}} + \Delta t \underline{\nabla} P$, we apply the velocity boundary conditions. For the computation of $\int\limits_{\Omega_{iel}} \mathrm{div}\,(\Delta t \underline{\nabla} P^{n-1+\theta}) d\Omega$, the boundary conditions for the pressure gradient (or more precisely for $\frac{\Delta t}{\rho} \underline{\nabla} P^{n-1+\theta}$) are therefore the homogenized boundary conditions of the velocity. The pressure gradient (or rather $\frac{\Delta t}{\rho} \underline{\nabla} P^{n-1+\theta}$) is consequently assumed to be zero normal to the walls and the inlets and to remain constant in the direction normal to symmetries and to the outlets.

In addition, to save computation time when passing through `inimas`, the face values on non-orthogonal meshes are evaluated merely to first-order accuracy in space (no reconstruction: `NSWRP=1`). Local accuracy is moreover of no particular interest as the aim here is simply to determine a normalized global residual.

The computation of the residual will be completed at the end of `cs_velocity_prediction`.

• Partial calculation of the normalized residual for the pressure step

During this first step, we use the array `XNORMP(NCELET)` to compute the quantity

$$\mathrm{div}\,(\Delta t \underline{\nabla} P^{n-1+\theta}) - \Gamma$$

integrated over each cell `IEL` of the mesh (Ω_{iel}), which is expressed as:

$$\texttt{XNORMP(IEL)} = \int\limits_{\Omega_{iel}} \mathrm{div}\,(\Delta t \underline{\nabla} P^{n-1+\theta}) - \Gamma d\Omega$$

This operation is carried out by successively implementing `inimas` (calculation at the faces in `VISCF` and `VISCB` of $\Delta t \underline{\nabla} P^{n-1+\theta}$ using the contents of the working array `TRAV`$=\frac{\Delta t}{\rho} \underline{\nabla} P^{n-1+\theta}$ together with homogeneous velocity boundary conditions and without reconstruction) followed by `divmas` (computation in `XNORMP` of the integral over the cells). With a simple loop, we then add the mass source term contribution Γ. This calculation is completed at the end of `cs_velocity_prediction`.

• Partial calculation of the term $\underline{f}_s^{exp}$

To represent the second member corresponding to each component of the velocity, we use the arrays `TRAV(IEL,DIR)`, `TRAVA(IEL,DIR)` and `PROPCE`, with `IEL` being the number of the cell and `DIR` the direction (x, y, z). Four cases need to be considered depending on whether or not the source terms

are extrapolated at $n+\theta_S$ and whether or not a fixed-point iteration is implemented on the velocity-pressure system (NTERUP>1).
• If the source terms are extrapolated and we iterate over cs_solve_navier_stokes

- TRAV receives the source terms which are recomputed during each iteration over cs_solve_navier_stokes but are not extrapolated ($\underline{\nabla} P^{n-1+\theta}$ and $(\rho-\rho_0)\underline{g}$[7]).

- TRAVA receives the source terms whose values do not change during the course of the iterations over cs_solve_navier_stokes nor are they extrapolated ($T_s^{imp}\,u^n$, $-\rho\,\underline{\underline{K}}_d\,u^n$, $-\Gamma^n\,u^n$, ...).

- PROPCE receives the source terms that have to be extrapolated.

• When there is no iteration over cs_solve_navier_stokes, TRAVA has no use and its contents are directly stored in TRAV.

• If there is no source term extrapolation, PROPCE serves no purpose and its contents are directly stored in TRAVA (or in TRAV if TRAVA is also of no use).
Therefore, when there is neither source term extrapolation nor iteration over cs_solve_navier_stokes, all the source terms go directly into TRAV.

- We already have a pressure gradient on the cells at the instant in time $n-1+\theta$. The gravity term is then added to the vector TRAV which already contains pressure gradient. This means we have, for example, for the direction x:

$$\texttt{TRAV}(\texttt{IEL},1) = |\Omega_{IEL}|(-(\frac{\partial p}{\partial x})_{\texttt{IEL}} + (\rho(\texttt{IEL})-\rho_0)g_x) \qquad \text{(IV.M.19)}$$

- If source term extrapolation is used, at the first iteration over cs_solve_navier_stokes, the vector TRAV (or TRAVA) receives $-\theta_S$ times the contribution at time $n-1$ of the source terms to be extrapolated[8] (stored in PROPCE). PROPCE is then reinitialised to zero so as to be able to receive afterwards the contribution at the current time step of the extrapolated source terms.
- The value of the term corresponding to the turbulence model is only calculated during the first iteration over cs_solve_navier_stokes then added to TRAVA, TRAV or PROPCE depending on whether the source terms are extrapolated, and/or whether we iterate over cs_solve_navier_stokes.

  ■ Turbulent viscosity models:
  If IGRHOK $=1$, then we calculate $-\frac{2}{3}\,\rho\,\underline{\nabla} k$ (and not, as should be the case, $-\frac{2}{3}\underline{\nabla}(\rho k)$) by way of simplification (cf. paragraph M). The gradient of the turbulent kinetic energy k is computed on the cell by the subroutine grdcel.
  If IGRHOK $=0$, this term is expected to be implicitly taken into account in the pressure.
  ■ Second order models:
  Computation of the term $-\mathrm{div}\,(\rho\underline{\underline{R}})$ is implemented in two steps. A call is first made to the subroutine divrij, which projects the vector $\underline{\underline{R}}.\underline{e}_{\texttt{DIR}}$ onto the cell faces along the direction DIR, following which we then call the subroutine divmas to compute the divergence.

[7] In reality, the value of $(\rho-\rho_0)\underline{g}$ doesn't change, however this is a computation and it avoids the need for additional treatment of this term.
[8] Because $(\underline{T}_s^{exp})^{n+\theta_S} = (1+\theta_S)\,(\underline{T}_s^{exp})^n - \theta_S\,(\underline{T}_s^{exp})^{n-1}$

- The secondary viscosity $-\frac{2}{3}\underline{\nabla}(\mu_{tot}\mathrm{div}\,\underline{u})$ and transposed gradient $\mathrm{div}\,(\mu_{tot}\,{}^t\underline{\underline{\mathrm{grad}}}\,\underline{u})$ terms, when these taken into account (*i.e.* `IVISSE(IPHAS)` $= 1$, where `IPHAS` is the number of the phase treated) are computed by the subroutine `visecv`. They are only calculated at the first iteration over `cs_solve_navier_stokes`. During this step, the array `TRAV` serves as a working array when the subroutine `visecv` is being called. It reverts to its normal value at the end of the call during which its contents are stored temporarily in the vectors `W7` to `W9`.

- The terms corresponding to the velocity head losses ($\rho\underline{\underline{K}}u$), if there are any (`NCEPDP` > 0), are calculated by the subroutine `st_exp_head_loss` at the first iteration over `cs_solve_navier_stokes`. They are decomposed into two parts:
  ■ The first part, corresponding to the contribution of the diagonal terms ($-\rho\,\underline{\underline{K}}_d\underline{u}$), is not extrapolated.
  ■ The second, which corresponds to the extra diagonal terms ($-\rho\,\underline{\underline{K}}_e\underline{u}$), may or may not be extrapolated.
  During this step, the array `TRAV` is used as a working array while the call is being made to the subroutine `st_exp_head_loss`. It reverts to its normal value at the end of this call, its contents having been meanwhile stored in the vectors `W7` to `W9`.

• Calculation of the viscosity term at the faces $(\mu_{tot})_{ij}\dfrac{\mathtt{SURFN}}{\mathtt{DIST}}$

The calculation of the total viscosity term at the faces is performed by the subroutine `viscfa` and the solutions stored in the arrays `VISCF` and `VISCB` for the internal and boundary faces respectively.
During integration of the convection-diffusion terms in the subroutine `codits`, the non-reconstructed terms, integrated in the matrix $\underline{\underline{EM}}$, are singled out from the full set of terms (unreconstructed + reconstruction gradients) associated to the operator $\mathcal{E}$ (nonlinear)[9]. Likewise, we differentiate between the total viscosity at the faces used in $\mathcal{E}$, arrays `VISCF` and `VISCB`, and the total viscosity at the faces employed in $\underline{\underline{EM}}$, arrays `VISCFI` and `VISCBI`.
For turbulent viscosity models and in LES, these two arrays are identical and contain $\mu_t + \mu$. For second order models, they normally contain μ however, for the simple reason of numerical stability, we can choose to put $\mu_t + \mu$ in the matrix (*i.e.* in $\underline{\underline{EM}}$) while keeping μ in the second member (*i.e.* in $\mathcal{E}$). Because the solution is obtained by increments, this manipulation doesn't change the result. This option is activated by the indicator `IRIJNU` $= 1$
When there is no diffusion of the velocity (`IDIFF(IUIPH)` < 1), the terms `VISCF` and `VISCB` are set to zero.

• Calculation of the whole second member, of f_s^{imp} and solving of the equation

The evolution equations for the components of the momentum are solved in a coupled manner. Consequently, we use a single array, `ROVSDT`, to represent the diagonal of the matrix obtained for each of the velocity components.
For each of these components:

- During the first iteration over the subroutine `cs_solve_navier_stokes`, the implicit and explicit parts of the user-prescribed source terms are computed by a call to the subroutine `ustsns`.
  ■ The implicit part (T_s^{imp}) is saved in the vector `XIMPA` for subsequent iterations if the fixed-point method is used on the velocity-pressure system, with the contribution resulting from the same implicit terms ($T_s^{imp}\,\underline{u}^n$) being added either to `TRAVA` or to `TRAV`.
  ■ The explicit part (T_s^{exp}) is added to `TRAVA`, `TRAV` or `PROPCE` depending on whether the source terms are extrapolated or not, and whether we iterate over `cs_solve_navier_stokes`.

[9] Ensuring coherence with the definition of the operators $\mathcal{EM}$ and $\mathcal{E}$ employed in `cs_solve_navier_stokes`

- The mass accumulation term ($\mathrm{div}\,(\rho\underline{u})$) is calculated by calling the subroutine `divmas` with the mass flux as argument. During the first iteration of the subroutine `cs_solve_navier_stokes`, the term corresponding to the explicit contribution of the mass accumulation ($\underline{u}^n\ \mathrm{div}\,(\rho\underline{u})$) is added to `TRAVA` or to `TRAV`. The vector `ROVSDT` is initialized by $\theta\,\mathrm{div}\,(\rho\underline{u})$ (to remain coherent with the approach implemented in the subroutine `bilsc2`) after which the contribution of the non-stationary term ($\frac{\rho}{\Delta t}$) is added to the latter.

- The vector `ROVSDT` is next completed with the contribution of the user-defined implicit source terms (stored in `XIMPA`) and with that of the head loss term ($\rho\,\underline{\underline{K}}_d$) should `NCEPDP` > 0.
  ■ When there is no source term extrapolation, the implicit part of the user source terms is only added to `ROVSDT` if it is negative so as not to weaken the diagonal of the system.
  ■ When the source terms are extrapolated however, it is taken into account whatever its sign may be.

- The implicit and explicit mass source terms, if any (`NCESMP` > 0), are computed at the first iteration by the subroutine `catsma` over `cs_solve_navier_stokes`. $\Gamma\,\underline{u}_i$ is added to `TRAV`, `TRAVA` or `PROPCE` for potential extrapolation. $\Gamma\,\underline{u}^n$ is added to `TRAV` or `TRAVA` and $-\Gamma$ to `ROVSDT`.

- The second member is then assembled taking account of all of the contributions stored in the arrays `PROPCE`, `TRAVA` and `TRAV`.
  ■ If the source terms are extrapolated, then the second member reads:

$$\mathtt{SMBR} = (1-\theta_S)\,\mathtt{PROPCE} + \mathtt{TRAVA} + \mathtt{TRAV}$$

  ■ If not, we directly obtain:

$$\mathtt{SMBR} = \mathtt{TRAVA} + \mathtt{TRAV}$$

- If a specific physics (lagrangian, electric arc, ...) is taken into account, its contribution is directly added to `SMBR`.

- The solution of the linear system is computed by the subroutine `codits`, taking as argument `ROVSDT` and `SMBR`.

- When strengthened transient velocity-pressure coupling (`IPUCOU` $= 1$) (only available with a first-order scheme, where there is neither source term extrapolation nor iteration over `cs_solve_navier_stokes`) is implemented, `codits` solves the following equation:

$$\underline{\underline{EM}}_{\mathtt{DIR}}\,.\,(\underline{\underline{RHO}}^{\,n})^{-1}\,.\,\underline{T}_{\mathtt{DIR}} = \underline{\underline{\Omega}}\,.\,\underline{1} \qquad \text{(IV.M.20)}$$

  with $\underline{\underline{RHO}}^{\,n}$ the diagonal tensor of element ρ^n_{IEL}, $\underline{\underline{\Omega}}$ the diagonal tensor of element $|\Omega_{IEL}|$ and $\underline{1}$ denoting the vector whose components are all equal to one.
  Inversion of the system by `codits` yields $(\underline{\underline{RHO}}^{\,n})^{-1}\,.\,\underline{T}_{\mathtt{DIR}}$, which is subsequently multiplied by $\underline{\underline{RHO}}^{\,n}$ to obtain $\underline{T}_{\mathtt{DIR}}$. This process is repeated for each component `DIR` of the velocity. $\underline{T}_{\mathtt{DIR}}$ is therefore a diagonal matrix approximation of $\underline{\underline{RHO}}^{\,n}\,.\,\underline{\underline{EM}}^{-1}_{\mathtt{DIR}}$, with $\underline{\underline{EM}}_{\mathtt{DIR}}$ representing the implicit part of the solution to the discrete momentum equation (*i.e.* `ROVSDT` + contribution of the convection-diffusion terms accounted for in the subroutine `matrix`). $\underline{T}_{\mathtt{DIR}}$ is also involved in the correction step (cf. subroutine `resopv`).

This terminates the loop over the velocity components.

- **Final calculation of the normalized residual for the pressure step**

As mentioned beforehand, the computation of the normalized residual for the pressure step of `resopv` can now be completed.

The `XNORMP` array already contains $\int\limits_{\Omega_{iel}} \mathrm{div}\,(\Delta t \underline{\nabla} P^{n-1+\theta}) - \Gamma d\Omega$ to which we now add $\int\limits_{\Omega_{iel}} \mathrm{div}\,(\rho \widetilde{\underline{u^{n+1}}}) d\Omega$.

To do so, we follow the same procedure used previously to compute the integral $\int\limits_{\Omega_{iel}} \mathrm{div}\,(\Delta t \underline{\nabla} P^{n-1+\theta}) d\Omega$.

A call to `inimas` enables to obtain $\rho S \widetilde{\underline{u}}^{n+1} \cdot \widetilde{\underline{n}}$ at the faces calculated from the of $\widetilde{\underline{u}}^{n+1}$ known at the cells (array `RTP`). The boundary conditions employed in `inimas` are naturally those of the velocity. As beforehand, evaluation of the face values is only up to first-order in space on non-orthogonal meshes (no reconstruction : `NSWRP=1`) in order to save computation time when passing through `inimas`. The subroutine `divmas` is then used to compute the divergence $\int\limits_{\Omega_{iel}} \mathrm{div}\,(\rho \widetilde{\underline{u}}^{n+1}) d\Omega$ at the cells, which is then added directly to `XNORMP`.

Lastly, the normalized residual is determined and then stored in `RNORMP(IIPHAS)` by a call to `prodsc` (which computes the sum of the squares of the values at the faces contained in `XNORMP` and then takes the square root of the result).

The different contributions (excluding convection-diffusion) assigned to each of the vectors TRAV, TRAVA, PROPCE and ROVSDT at iteration n are summarized in tables (IV.M.21), (IV.M.22), (IV.M.23) and (IV.M.24). We differentiate two cases for each of the time-stepping schemes applied to the source terms, depending on whether or not a fixed-point algorithm is used to solve the velocity-pressure system (iteration over cs_solve_navier_stokes for NTERUP>1). Unless otherwise specified, the physical properties Φ (ρ,μ,etc...) are assumed to be evaluated at the time level $n+\theta_\Phi$, and the mass flux ($\rho\underline{u}$) at $n+\theta_F$, where θ_Φ and θ_F are dependent on the specific time-stepping schemes selected for these quantities (cf. introd).

The terms appearing in these tables are written as they have been implemented in the code, this being the source of some differences in relation to the form adopted in equation (IV.M.16).

In order to simplify the problem, we pose:

$$\mu_{tot} = \begin{cases} \mu+\mu_t & \text{for turbulent viscosity models and in LES,} \\ \mu & \text{for second-order models and in laminar regime} \end{cases}$$

With extrapolation of source terms

$$\underline{turb}^n = \begin{cases} \dfrac{2}{3}\rho^n \underline{\nabla}(k^n) & \text{for turbulent viscosity models,} \\ \mathrm{div}\,(\rho^n \underline{\underline{R}}^n) & \text{for second-order models,} \\ 0 & \text{in laminar or LES computations.} \end{cases}$$

- NTERUP $=1$: $\mathtt{SMBR}^n = (1-\theta_S)\,\mathtt{PROPCE}^n + \mathtt{TRAV}^n$

$\mathtt{ROVSDT}^n$	$\dfrac{\rho}{\Delta t} - \theta\,\mathrm{div}\,(\rho\,\underline{u}) + \theta\ \Gamma^n + \theta\ \rho\,\underline{\underline{K}}_d^n - \theta\,T_s^{imp}$
$\mathtt{PROPCE}^n$	$\underline{T}_s^{exp,\,n} - \rho^n\,\underline{\underline{K}}_e^n\ \underline{u}^n + \Gamma^n\,\underline{u}_i^n$ $-\underline{turb}^n + \mathrm{div}\,(\mu_{tot}^n\,{}^t\underline{\underline{\mathrm{grad}}}\,\underline{u}^n) + \dfrac{2}{3}\underline{\nabla}(\mu_{tot}^n\,\mathrm{div}\,\dfrac{(\rho\underline{u})}{\rho^n})$
$\mathtt{TRAV}^n$	$-\underline{\nabla}p^{n-1+\theta} + (\rho-\rho_0)\underline{g}$ $-\theta_S\,\mathtt{PROPCE}^{n-1} - \rho\,\underline{\underline{K}}_d^n\ \underline{u}^n$ $+T_s^{imp}\ \underline{u}^n + \mathrm{div}\,(\rho\,\underline{u})\,\underline{u}^n - \Gamma^n\ \underline{u}^n$

(IV.M.21)

- NTERUP >1 (sub-iteration k) : $\mathtt{SMBR}^n = (1-\theta_S)\,\mathtt{PROPCE}^n + \mathtt{TRAVA}^n + \mathtt{TRAV}^n$

$\mathtt{ROVSDT}^n$	$\dfrac{\rho}{\Delta t} - \theta\,\mathrm{div}\,(\rho\,\underline{u}) + \theta\ \Gamma^n + \theta\ \rho\,\underline{\underline{K}}_d^n - \theta\,T_s^{imp}$
$\mathtt{PROPCE}^n$	$\underline{T}_s^{exp,\,n} - \rho^n\,\underline{\underline{K}}_e^n\ \underline{u}^n + \Gamma^n\,\underline{u}_i^n$ $-\underline{turb}^n + \mathrm{div}\,(\mu_{tot}^n\,{}^t\underline{\underline{\mathrm{grad}}}\,\underline{u}^n) + \dfrac{2}{3}\underline{\nabla}(\mu_{tot}^n\,\mathrm{div}\,\dfrac{(\rho\underline{u})}{\rho^n})$
$\mathtt{TRAVA}^n$	$-\theta_S\,\mathtt{PROPCE}^{n-1} - \rho\,\underline{\underline{K}}_d^n\ \underline{u}^n + T_s^{imp}\ \underline{u}^n + \mathrm{div}\,(\rho\,\underline{u})\,\underline{u}^n - \Gamma^n\ \underline{u}^n$
$\mathtt{TRAV}^n$	$-\underline{\nabla}(p^{n+\theta})^{(k-1)} + (\rho-\rho_0)\underline{g}$

(IV.M.22)

Without source term extrapolation

$$\underline{turb}^n = \begin{cases} \dfrac{2}{3}\rho\underline{\nabla}(k^n) & \text{for turbulent viscosity models,} \\ \mathrm{div}\,(\rho\underline{\underline{R}}^n) & \text{for second-order models,} \\ 0 & \text{in laminaire or LES computations.} \end{cases}$$

• NTERUP $= 1$: $\texttt{SMBR}^n = \texttt{TRAV}^n$

$\texttt{ROVSDT}^n$	$\dfrac{\rho}{\Delta t} - \theta \,\mathrm{div}\,(\rho\,\underline{u}) + \Gamma^n + \rho\,\underline{\underline{K}}^n_d + Max(-T_s^{imp}, 0)$
$\texttt{TRAV}^n$	$-\underline{\nabla} p^{n-1+\theta} + (\rho - \rho_0)\underline{g}$ $+\underline{T}_s^{exp} - \rho\,\underline{\underline{K}}^n_e\,\underline{u}^n + \Gamma^n\,\underline{u}^n_i$ $-\underline{turb}^n + \mathrm{div}\,(\mu_{tot}\,{}^t\underline{\underline{\mathrm{grad}}}\,\underline{u}^n) + \dfrac{2}{3}\,\underline{\nabla}(\mu_{tot}\,\mathrm{div}\,\dfrac{(\rho\underline{u})}{\rho})$ $-\rho\,\underline{\underline{K}}^n_d\,\underline{u}^n + T_s^{imp}\,\underline{u}^n + \mathrm{div}\,(\rho\,\underline{u})\,\underline{u}^n - \Gamma^n\,\underline{u}^n$

(IV.M.23)

• NTERUP > 1 (sub-iteration k) : $\texttt{SMBR}^n = \texttt{TRAVA}^n + \texttt{TRAV}^n$

$\texttt{ROVSDT}^n$	$\dfrac{\rho}{\Delta t} - \theta \,\mathrm{div}\,(\rho\,\underline{u}) + \Gamma^n + \rho\,\underline{\underline{K}}^n_d + Max(-T_s^{imp}, 0)$
$\texttt{TRAVA}^n$	$\underline{T}_s^{exp} - \rho\,\underline{\underline{K}}^n_e\,\underline{u}^n + \Gamma^n\,\underline{u}^n_i$ $-\underline{turb}^n + \mathrm{div}\,(\mu_{tot}\,{}^t\underline{\underline{\mathrm{grad}}}\,\underline{u}^n) + \dfrac{2}{3}\,\underline{\nabla}(\mu_{tot}\,\mathrm{div}\,\dfrac{(\rho\underline{u})}{\rho})$ $-\rho\,\underline{\underline{K}}^n_d\,\underline{u}^n + T_s^{imp}\,\underline{u}^n + \mathrm{div}\,(\rho\,\underline{u})\,\underline{u}^n - \Gamma^n\,\underline{u}^n$
$\texttt{TRAV}^n$	$-\underline{\nabla}(p^{n+\theta})^{(k-1)} + (\rho - \rho_0)\underline{g}$

(IV.M.24)

Points to treat

• Evaluation of the term $\underline{\nabla}(\rho k)$ for turbulent viscosity models
When modelling turbulent viscosity, we compute $\rho\,\underline{\nabla} k$ instead of $\underline{\nabla}(\rho k)$. This approximation was implemented from the outset because the ρk boundary conditions are not directly accessible, in contrast to those of k.

• Expression of $\underline{\underline{EM}}$
With incremental solving, it is not absolutely necessary for convergence that the value of the viscosity appearing in the expression of the operator $\mathcal{E}$ be the same as that taken into account in the incremental system matrix, $\underline{\underline{EM}}$. In $R_{ij} - \varepsilon$, for example, the total viscosity used in $\underline{\underline{EM}}$ contains the molecular viscosity but can also contain the turbulent viscosity if the option `IRIJNU = 1` is selected, whereas only the former is included in the operator $\mathcal{E}$. This addition of the turbulent viscosity, which is not at all relevant to the $R_{ij} - \varepsilon$ model, has been inherited from practices implemented in ESTET and N3S-EF to improve numerical stability (incremental smoothing). However, it may have other, potentially less desirable effects. Moreover, it has not been demonstrated to date that this practice is of absolute necessity in the use of code_saturne. An in-depth study would therefore be of some interest.

• Normalized residual relating to the pressure step
We could check the normalized residual computation and in particular the use of the velocity boundary conditions.

• Calculation of the head losses
When the source terms are extrapolated in time, we now obtain:

$$(\underline{\underline{K}}_e\underline{u})^{n+\theta_S} + \underline{\underline{K}}^n_d\,\underline{u}^{n+\theta}$$

It would also be possible to envisage using:

$$(\underline{\underline{K}}_e\underline{u})^{n+\theta_S} + \underline{\underline{K}}^{n+\theta_S}_d\,\underline{u}^{n+\theta}$$

N- resopv routine

Function

The velocity projection (or pressure correction) step is effected in this subroutine, called from `cs_solve_navier_stokes`. The equation of motion (prediction) is solved in `cs_velocity_prediction` with a fully explicit treatment of the pressure term. The resulting velocity field does not satisfy the continuity equation. Two correction algorithms are proposed:

1. The algorithm that we will call "weak velocity-pressure coupling". This algorithm is implemented extensively in industrial source codes. It only couples the velocity and the pressure through the mass term (it is the algorithm proposed by default). It consists of a *SIMPLEC*-type algorithm, similar to *SIMPLE* although the latter accounts for the simplified diagonals of the convection, diffusion and implicit source terms in addition to the mass term.
2. The strengthened velocity-pressure coupling algorithm (option `IPUCOU = 1`). This algorithm couples the velocity and pressure through all of the terms (convection, diffusion and implicit source terms) of the equation of motion, though it is still approximate. In practice, the advantage of this algorithm is that it allows large time steps without entirely decoupling the velocity and the pressure.

Taking δp as the pressure increment (*i.e.* $p^{n+1} = p^n + \delta p$) and $\widetilde{\underline{u}}$ the velocity field resulting from the prediction step, from a continuous point of view the projection step essentially comes down to solving a Poisson equation for the pressure:

$$\mathrm{div}\,(\,\underline{\underline{T}}^n\ \underline{\nabla}\delta p) = \mathrm{div}\,(\,\rho\,\widetilde{\underline{u}}) \tag{IV.N.1}$$

and then correcting the velocity:

$$\underline{u}^{n+1} = \underline{u}^n - \frac{1}{\rho}\,\underline{\underline{T}}^n\ \underline{\nabla}\delta p \tag{IV.N.2}$$

$\underline{\underline{T}}^n$ is a second-order tensor whose components are homogeneous over a time step.

See the programmers reference of the dedicated subroutine for further details.

Discretization

The velocity prediction step is described in `cs_velocity_prediction`. An operator notation is adopted here to provide a simplified explanation of the code-based algorithms. The discrete equation of motion at an intermediate point in time of solution is written, in each direction of space α ($\alpha \in \{1,2,3\}$)in the form:

$$\underline{\underline{M}}^n_\alpha\ \underline{\underline{R}}^n\ (\widetilde{\underline{V_\alpha}} - \underline{V_\alpha}^n) + \underline{\underline{A_\alpha}}^n\ \widetilde{\underline{V_\alpha}} = -\ \underline{\underline{G_\alpha}}\ \underline{P}^n + \underline{S_\alpha}^n + \underline{\underline{I}}_{s,\alpha}\ \widetilde{\underline{V_\alpha}} \tag{IV.N.3}$$

- $\underline{\underline{M}}^n_\alpha$, is a diagonal matrix, of dimension `NCEL` $\times$ `NCEL`, containing the ratio of the volume of a cell to the local time step ($\underline{\underline{M}}^n_\alpha(i,i) = \dfrac{|\Omega_i|}{\Delta t^n_{\alpha,I}}$), $\Delta t^n_{\alpha,I}$ represents the time step at time level n in the (spatial) direction α at the cell Ω_i,

- $\underline{\underline{R}}^n$, is the diagonal matrix of dimension NCEL×NCEL, that contains the density (which is separated out from the mass matrix during this step so that a projection can be made of the value of $\rho\underline{u}$). By definition, $\underline{\underline{R}}^n(i,i) = \rho_I^n$, which means that the appearance of a vacuum (or null space) is naturally excluded and the matrix will therefore always be invertible,
- $\underline{\widetilde{V_\alpha}}$, of dimension NCEL, is an array in which the α^{th} component of the predicted velocity field $\underline{\widetilde{u}}$ is stored,
- $\underline{V_\alpha}^n$, of dimension NCEL, is an array storing the α^{th} component of the velocity $\underline{u}^n$ at the previous instant of time n,
- $\underline{\underline{A_\alpha}}^n$ denotes the convection/diffusion operator (it is not necessarily linear owing to the possible use of slope tests in the convection scheme and may be dependent on $\underline{V_\alpha}$),
- $\underline{\underline{G_\alpha}}$ is the linear, "cell" gradient operator[1] in the direction α (it is therefore applied on the vectors of dimension NCEL),
- $\underline{P}^n$, of dimension NCEL, is an array used to store the pressure p^n computed at each cell during the previous time step,
- $\underline{S_\alpha}^n$ is the array of dimension NCEL that contains all of the explicit source terms (see `cs_velocity_prediction` for more detail),
- $\underline{\underline{I}}_{s,\alpha}$ is the diagonal tensor related to the implicit source terms of the velocity components.

The correction step consists of imposing the continuity constraint for the mass conservation:

$$\text{div}\,(\,\rho\,\underline{u}\,) = \Gamma \tag{IV.N.4}$$

where Γ is an eventual mass source term.

Let $\underline{W}$ be the array of dimension $3 \times$ NCEL that contains all of the momentum components ($\underline{V}$ denotes $\underline{V}^n$, $\underline{V}^{n+1}$ or $\underline{\widetilde{V}}$).

$$\underline{W} = \underline{\underline{R}}^n\ \underline{V} = \begin{pmatrix} \rho^n\ \underline{V_1} \\ \rho^n\ \underline{V_2} \\ \rho^n\ \underline{V_3} \end{pmatrix}$$

Let $\underline{\underline{D}}$ be the divergence operator. The continuity equation (IV.N.4) is rewritten in compact form as:

$$\underline{\underline{D}}\ \underline{W} = \underline{\Gamma}$$

$\underline{\Gamma}$ contains the values of Γ at the cell centroids.

Rearranging the discrete equation (IV.N.3), we write for all $\alpha \in \{1,2,3\}$:

$$(\underline{\underline{M}}_\alpha^n + \underline{\underline{A_\alpha}}^n\ (\underline{\underline{R}}^n)^{-1} - \underline{\underline{I}}_{s,\alpha}\ (\underline{\underline{R}}^n)^{-1})\ \underline{\underline{R}}^n\ \underline{\widetilde{V_\alpha}} = -\ \underline{\underline{G_\alpha}}\ \underline{P}^n + \underline{S_\alpha}^n + \underline{\underline{M_\alpha}}\ \underline{\underline{R}}^n\ \underline{V_\alpha}^n \tag{IV.N.5}$$

By grouping and posing the following definitions:

$$\underline{\underline{B_\alpha}} = \underline{\underline{M}}_\alpha^n + \underline{\underline{A_\alpha}}^n\ (\underline{\underline{R}}^n)^{-1} - \underline{\underline{I}}_{s,\alpha}\ (\underline{\underline{R}}^n)^{-1}$$

$$\underline{\underline{B}} = \begin{pmatrix} \underline{\underline{B_1}} & 0 & 0 \\ 0 & \underline{\underline{B_2}} & 0 \\ 0 & 0 & \underline{\underline{B_3}} \end{pmatrix}$$

[1] strictly speaking, this operator would no longer be truly linear if a gradient constraint option were to have been activated by the user

$$\underline{\underline{G}} = \begin{pmatrix} \underline{\underline{G_1}} \\ \underline{\underline{G_2}} \\ \underline{\underline{G_3}} \end{pmatrix}$$

$$\underline{S}^n = \begin{pmatrix} \underline{S_1}^n + \underline{\underline{M}}_1 \, \underline{\underline{R}}^n \, \underline{V_1}^n \\ \underline{S_2}^n + \underline{\underline{M}}_2 \, \underline{\underline{R}}^n \, \underline{V_2}^n \\ \underline{S_3}^n + \underline{\underline{M}}_3 \, \underline{\underline{R}}^n \, \underline{V_3}^n \end{pmatrix}$$

we can thus write the simplified equation as:

$$\underline{\underline{B}} \, \underline{\widetilde{W}} = - \, \underline{\underline{G}} \, \underline{P}^n + \underline{S}^n \tag{IV.N.6}$$

With the fractional step method, (IV.L.1) is decomposed into a sequence of two steps:

1. solution of the equation (IV.N.3) (this equation yields equation(IV.N.6)), namely:

$$\underline{\underline{M}}^n_\alpha \, \underline{\underline{R}}^n \, (\underline{\widetilde{V_\alpha}} - \underline{V_\alpha}^n) + \underline{\underline{A_\alpha}}^n \, \underline{\widetilde{V_\alpha}} - \underline{\underline{I}}_{s,\alpha} \, \underline{\widetilde{V_\alpha}} = - \, \underline{\underline{G_\alpha}} \, \underline{P}^n + \underline{S_\alpha}^n \tag{IV.N.7}$$

2. subtraction[2] of the solved velocity prediction equation (IV.N.7) from the equation of motion evaluated at the subsequent time level $(n+1)$:

$$\underline{\underline{M}}^n_\alpha \, \underline{\underline{R}}^n \, (\underline{V_\alpha}^{n+1} - \underline{\widetilde{V_\alpha}}) + \underline{\underline{A_\alpha}}^n \, (\underline{V_\alpha}^{n+1} - \underline{\widetilde{V_\alpha}}) - \underline{\underline{I}}_{s,\alpha} \, (\underline{V_\alpha}^{n+1} - \underline{\widetilde{V_\alpha}}) = - \, \underline{\underline{G_\alpha}} \, (\underline{P}^{n+1} - \underline{P}^n) \tag{IV.N.8}$$

Reverting to compact notation, Equation (IV.N.8) gives:

$$\underline{\underline{B}} \, (\underline{W}^{n+1} - \underline{\widetilde{W}}) = - \, \underline{\underline{G}} \, (\underline{P}^{n+1} - \underline{P}^n) \tag{IV.N.9}$$

with:

$$\underline{W}^{n+1} = \underline{\underline{R}}^n \, \underline{V}^{n+1}$$

The continuity constraint has yet to be enforced:

$$\underline{\underline{D}} \, \underline{W}^{n+1} = \underline{\Gamma}$$

By combining the equations of continuity and motion and postulating that $\delta \underline{P} = \underline{P}^{n+1} - \underline{P}^n$, the following Poisson-type equation is derived:

$$\underline{\underline{D}} \, \underline{\underline{B}}^{-1} \, \underline{\underline{G}} \, \delta \underline{P} = \underline{\underline{D}} \, \underline{\widetilde{W}} - \underline{\Gamma} \tag{IV.N.10}$$

We still need to invert the system (IV.N.10) in order to determine δp (and thus p^{n+1}) and correct the projected velocity field so as to obtain $\underline{u}^{n+1}$. The velocity correction is handled in `cs_solve_navier_stokes`, by incrementing the velocity by the calculated magnitude of the gradient of the pressure increment δp.

The problem initially arises with the calculation of $\underline{\underline{B}}^{-1}$. It has already been judged too expensive computationally to calculate the inverse of $\underline{\underline{B}}$. The solutions computed by the "weak velocity-pressure coupling" and the "strengthened velocity-pressure coupling" algorithms correspond to an approximation of this operator.

In the case of the "weak velocity-pressure coupling" algorithm, we assume $\underline{\underline{B}}^{-1}_\alpha = \underline{\underline{M}}^{-1}_\alpha$ (we could also include the diagonal terms of the convection, diffusion and implicit source terms).

With "strengthened velocity-pressure coupling", we invert the system[3] $\underline{\underline{B}} \, \underline{T} = \underline{\Omega}$ and we define the equality $\underline{\underline{B}}^{-1}_\alpha = \mathrm{diag}(T_\alpha)$. This step takes place in the subroutine `cs_velocity_prediction`.

[2] We neglect any variation in the explicit source terms $\underline{S_\alpha}^n$ which are still those estimated at n.

[3] $\underline{\Omega} = (|\Omega_1|, ..., |\Omega_{\texttt{NCEL}}|, |\Omega_1|, ..., |\Omega_{\texttt{NCEL}}|, |\Omega_1|, ..., |\Omega_{\texttt{NCEL}}|)$.

The use of operators when writing out the equations presents a major inconvenience when used in conjunction with collocated discretization. More specifically, the operator[4] $\underline{\underline{D}}\,\underline{\underline{B}}^{-1}\,\underline{\underline{G}}$ can lead to odd-even decoupling of the nodes on a regular Cartesian mesh[5]. To avoid this problem, we use the operator $\underline{\underline{L}}$ (already present in the collocated finite volume formulation of the operator div ($\underline{\nabla}$)) defined in each cell[6] Ω_i of centre I by[7]:

$$(\underline{\underline{L}}\ \delta\underline{P})_I = \sum_{j\in Neigh(i)} [\ \underline{\underline{T}}^n_{ij}\ (\underline{\nabla}\delta p)_{f_{ij}}]\cdot\underline{S}_{ij} + \sum_{k\in\gamma_b(i)} [\ \underline{\underline{T}}^n_{b_{ik}}\ (\underline{\nabla}\delta p)_{f_{b_{ik}}}]\cdot\underline{S}_{b_{ik}} \qquad \text{(IV.N.11)}$$

$\underline{\underline{T}}^n$ is a diagonal tensor of order 2 containing the time steps in the three spatial directions with $\underline{S}_{ij}$ and $\underline{S}_{b_{ik}}$ the surface vector respectively of the purely internal face and of the boundary face ik. The gradient $(\underline{\nabla}\delta p)_f$ present in equation (IV.N.11) is a facet[8] gradient.

From a continuous perspective, we can write[9]:

$$(\underline{\underline{L}}\ \underline{P}^{n+1})_I = \int_{\Omega_i} \text{div}\,(\ \underline{\underline{T}}^n\ \underline{\nabla}p^{n+1})\ d\Omega$$

The operator $\underline{\underline{L}}$ does not pose a problem of odd/even decoupling on regular Cartesian meshes.

With the "weak velocity-pressure coupling" algorithm: $\underline{\underline{T}}^n_I = \begin{pmatrix} \Delta t^n_I & 0 & 0 \\ 0 & \Delta t^n_I & 0 \\ 0 & 0 & \Delta t^n_I \end{pmatrix}$

and with the "strengthened velocity-pressure coupling" algorithm: $\underline{\underline{T}}^n_I = \begin{pmatrix} T^n_{11,I} & 0 & 0 \\ 0 & T^n_{22,I} & 0 \\ 0 & 0 & T^n_{33,I} \end{pmatrix}$

The pressure system matrix is not easy to compute when "strengthened velocity-pressure coupling" is being used, this being due to the term:

$$[\ \underline{\underline{T}}^n_{ij}\ (\underline{\nabla}\delta p)_{f_{ij}}]\cdot\underline{S}_{ij}$$

for an internal face ij and the term:

$$[\ \underline{\underline{T}}^n_{b_{ik}}\ (\underline{\nabla}\delta p)_{f_{b_{ik}}}]\cdot\underline{S}_{b_{ik}}$$

for a boundary face.
The difficulty lies with the viscosity, which changes according to the spatial direction. As the tensor $\underline{\underline{T}}^n$ is effectively anisotropic, the pressure gradients for each direction must consequently be computed as a function of the normal gradient.

Without reference, for the moment, to the nature of the gradient $\underline{\nabla}$, we define for every scalar a:

$$\widetilde{\underline{G}}\ a = \underline{\underline{T}}^n\ \underline{\nabla}a = \begin{pmatrix} T^n_{11}\ \dfrac{\partial a}{\partial x} \\ T^n_{22}\ \dfrac{\partial a}{\partial y} \\ T^n_{33}\ \dfrac{\partial a}{\partial z} \end{pmatrix}$$

[4] We emphasize again that the operator $\underline{\underline{G}}$ is the "cell gradients" operator applied to the explicit pressure during velocity prediction.

[5] If u_i is the value of a variable at the cell centres on a 1D Cartesian mesh, the Laplacian of u calculated by this operator in i is written: $\dfrac{u_{i-2}+2u_i-u_{i+2}}{4h^2}$, where h is the space step. This is where the decoupling of the cells originates.

[6] Recalling that $Neigh(i)$ is the set of cell centres of the neighbouring cells of Ω_i and $\gamma_b(i)$ the set of centres of the boundary faces, if any, of Ω_i.

[7] If u_i is the value of a variable at the cell centres on a 1D Cartesian mesh, the Laplacian of u calculated by the latter operator in i reads: $\dfrac{u_{i-1}+2u_i-u_{i+1}}{4h^2}$, with h the spatial step.

[8] On orthogonal mesh, $(\underline{\nabla}p)_{f_{ij}}\ .\ \underline{S}_{ij} = \dfrac{p_J-p_I}{\overline{IJ}}S_{ij}$. I (resp. J) et I' (resp. J') are in effect superimposed.

[9] In the "weak velocity-pressure coupling" algorithm case, $\widetilde{T}^n_I = \Delta t^n_I$.

We can also define:

$$\left[(\widetilde{\underline{G}}\ a)_{\,cell}\right]_{ij} \cdot \underline{S}_{ij} \overset{def}{=} \left[\underline{\underline{T}}^{n}\ (\underline{\nabla} a)\right]_{ij} \cdot \underline{S}_{ij}$$

where $\underline{\nabla} a$ is the standard cell gradient of a.

and likewise for the facet gradient $(\underline{\nabla} a)_{\,f}$ of a:

$$\left[(\widetilde{\underline{G}}\ a)_{\,f}\right]_{ij} \cdot \underline{S}_{ij} \overset{def}{=} \left[\underline{\underline{T}}^{n}\ ((\underline{\nabla} a))_{\,f}\right]_{ij} \cdot \underline{S}_{ij}$$

We need to calculate $\widetilde{\underline{G}}\ (\delta p) \cdot \underline{S}$ at the face.
This is relatively simple with a cell gradient as all the components of the latter are perfectly calculable. The difficulty lies, on the contrary, with a facet gradient because the only exploitable options is the decomposition of the face normal gradient[10]. We need the quantities $\frac{\partial(\delta p)}{\partial x}$, $\frac{\partial(\delta p)}{\partial y}$ and $\frac{\partial(\delta p)}{\partial z}$ both explicitly and separately, which makes it difficult to use the normal pressure increment gradient.
We therefore make an approximation of the gradient of the pressure increment by assuming it to be equal to its normal component[11], which is:

$$(\underline{\nabla}(\delta p))_{\,f} \approx ((\underline{\nabla}(\delta p))_{\,f} \cdot \underline{n})\ \underline{n} \tag{IV.N.12}$$

where $\underline{n}$ denotes the outer unit normal vector. We then obtain:

$$\widetilde{\underline{G}}\ (\delta p) \approx ((\underline{\nabla}(\delta p))_{\,f} \cdot \underline{n})\ (\underline{\underline{T}}^{n}\ \underline{n})$$

This enables reducing the computations to those of a scalar time step $\widetilde{T}^{n}_{ij}$, given by:

$$\widetilde{T}^{n}_{ij} = (\underline{\underline{T}}^{n}_{ij}\ \underline{n}) \cdot \underline{n} \tag{IV.N.13}$$

In which case:

$$\begin{aligned} [\ \underline{\underline{T}}^{n}_{ij}\ (\underline{\nabla}\delta p)_{\,f_{ij}}] \cdot \underline{S}_{ij} &\approx (\underline{\underline{T}}^{n}_{ij}\ \underline{n}) \cdot \underline{S}_{ij}\ (\underline{\nabla}(\delta p)_{f_{ij}} \cdot \underline{n}) \\ &= (\underline{\underline{T}}^{n}_{ij}\ \underline{n}) \cdot \underline{n}\ (\underline{\nabla}(\delta p)_{f_{ij}} \cdot \underline{n})\ S_{ij} \\ &= \widetilde{T}^{n}_{ij}\ (\underline{\nabla}(\delta p)_{f_{ij}} \cdot \underline{n})\ S_{ij} \end{aligned} \tag{IV.N.14}$$

or on the other hand:

$$\begin{aligned} \widetilde{T}^{n}_{ij}\ (\underline{\nabla}(\delta p)_{f_{ij}} \cdot \underline{n})\ S_{ij} &= \widetilde{T}^{n}_{ij}\ \underline{\nabla}(\delta p)_{f_{ij}} \cdot \underline{S}_{ij} \\ &= \widetilde{T}^{n}_{ij}\ \left[P_J - P_I + (\ \underline{JJ'} - \underline{II'}\ )\ \frac{1}{2}\ (\underline{\nabla} P_I + \underline{\nabla} P_J\ )\right] \frac{S_{ij}}{\overline{I'J'}} \end{aligned} \tag{IV.N.15}$$

This is used during reconstruction of the gradients in the second member of the final system of equations to solve, by calling the subroutines `itrmas` and `itrgrp`.
The approximation $(\widetilde{T}^{n}_{ij} = (\underline{\underline{T}}^{n}_{ij}\ \underline{n}) \cdot \underline{n})$ is not in fact used. This is because the directional gradient of the pressure increment is computed using the subroutine `grdcel`, allowing us to take the liberty of using the tensor $\underline{\underline{T}}$ to correct the term $[\ \underline{\underline{T}}^{n}_{ij}\ (\underline{\nabla}\delta p)_{\,f_{ij}}] \cdot \underline{S}_{ij}$.
In practice, the latter is discretized, for an internal face ij, by[12] :

$$[\ \underline{\underline{T}}^{n}_{ij}\ (\underline{\nabla}\delta p)_{\,f_{ij}}] \cdot \underline{S}_{ij} = \left[\widetilde{T}^{n}_{ij}\ \frac{P_J - P_I}{\overline{I'J'}} + \frac{\underline{JJ'} - \underline{II'}}{\overline{I'J'}}\ \frac{1}{2}\ (\underline{\underline{T}}^{n}_{I}\ \underline{\nabla} P_I + \underline{\underline{T}}^{n}_{J}\ \underline{\nabla} P_J\ )\right] S_{ij} \tag{IV.N.16}$$

[10] According to the formula $\frac{p_{\,J'} - p_{\,I'}}{\overline{I'J'}} S_{ij}$.

[11] *i.e.* no account is taken of the tangential component $((\underline{\nabla}(\delta p))_{\,f} \cdot \underline{\tau}).\underline{\tau}$, $\underline{\tau}$ denoting the tangential unit vector.

[12] The factor $\frac{1}{2}$ appearing in the discretization (IV.N.16) is introduced for reasons of numerical stability. Although the use of weighting coefficients on the faces would in effect yield a more accurate numerical solution, it would probably also be less stable.

rather than by (IV.N.15). We use the same approach for the boundary terms.
The expression $\widetilde{T}_{ij}^n \ (\underline{\nabla}(\delta p)_{f_{ij}} \cdot \underline{S}_{ij})$ will nevertheless be seen subsequently.

The last issue relating to the inversion of the system (IV.N.10) concerns the tensor term $\underline{\underline{D}} \ \widetilde{\underline{W}}$. This term tends, *via* the cell pressure gradients present in the equation of motion, to decouple the odd and even-numbered cells on a Cartesian grid. To avoid this problem, we apply a variant of the Rhie & Chow filter which enables to dissipate (or smooth) the pressure field contribution in the equation of motion. Expressed in discretized form, this yields:

$$\begin{aligned}(\underline{\underline{D}} \ \widetilde{\underline{W}})_I = & \sum_{j \in Neigh(i)} [\rho^n \widetilde{\underline{u}} + \alpha_{Arak} \ (\widetilde{\underline{G}} \ (p^n))_{cell}]_{f_{ij}} \cdot \underline{S}_{ij} - \alpha_{Arak} \sum_{j \in Neigh(i)} \widetilde{T}_{ij}^n \ (\underline{\nabla} p^n)_{f_{ij}} \cdot \underline{S}_{ij} \\ & + \sum_{k \in \gamma_b(i)} [\rho^n \widetilde{\underline{u}} + \alpha_{Arak} \ (\widetilde{\underline{G}} \ (p^n))_{cell}]_{f_{b_{ik}}} \cdot \underline{S}_{b_{ik}} - \alpha_{Arak} \sum_{k \in \gamma_b(i)} \widetilde{T}_{b_{ik}}^n \ (\underline{\nabla} p^n)_{f_{b_{ik}}} \cdot \underline{S}_{b_{ik}} \\ & = \sum_{j \in Neigh(i)} \widetilde{m}_{ij} + \sum_{k \in \gamma_b(i)} \widetilde{m}_{b_{ik}}\end{aligned} \tag{IV.N.17}$$

where $\widetilde{m}_{ij}$ and $\widetilde{m}_{b_{ik}}$ denote respectively the mass flux at the purely internal face and the boundary face ik) modified by the Rhie & Chow filter.
$(\underline{\nabla} p^n)_{f_{ij}}$ represents a facet gradient while $[\rho^n \widetilde{\underline{u}} + \alpha_{Arak} \ (\widetilde{\underline{G}} \ (p^n))_{cell}]_{f_{ij}}$ denotes a face value interpolated from the estimated cell values (the pressure gradient in this term is a cell-based gradient). This clarification also applies to the boundary terms.
For historical reasons, α_{Arak} is known as "the Arakawa coefficient d'Arakawa" in code_saturne. It is denoted `ARAK`.

It should be reiterated that, for the Rhie & Chow filter, the tensor $\underline{\underline{T}}^n$ is used in the volumetric term $\widetilde{\underline{G}}$ whereas the approximation (IV.N.13) is applied when computing the facet gradient.

In accordance with equations (IV.N.10) and (IV.N.11), we finally solve the Poisson equation under the form:

$$\sum_{j \in Neigh(i)} \widetilde{T}_{ij}^n \ (\underline{\nabla}(\delta p))_{f_{ij}} \cdot \underline{S}_{ij} + \sum_{k \in \gamma_b(i)} \widetilde{T}_{b_{ik}}^n \ (\underline{\nabla}(\delta p))_{f_{b_{ik}}} \cdot \underline{S}_{b_{ik}} = \sum_{j \in Neigh(i)} \widetilde{m}_{ij} + \sum_{k \in \gamma_b(i)} \widetilde{m}_{b_{ik}} - \Gamma_I \tag{IV.N.18}$$

To take account of the non-orthogonal elements, we use an incremental method to solve the linear equation system, including the reconstructed terms in the second member. If $\delta(\delta p)$ denotes the increment of the increment (the increment of the variable δp to compute) and k the fixed-point iteration index, we solve exactly :

$$\begin{aligned}\sum_{j \in Neigh(i)} \widetilde{T}_{ij}^n \ \frac{(\delta(\delta p))_I^{k+1} - (\delta(\delta p))_J^{k+1}}{\overline{I'J'}} \ S_{ij} + \sum_{k \in \gamma_b(i)} \widetilde{T}_{b_{ik}}^n \ \frac{(\delta(\delta p))_I^{k+1} - (\delta(\delta p))_{b_{ik}}^{k+1}}{\overline{I'F}} \ S_{b_{ik}} & \\ = \sum_{j \in Neigh(i)} \widetilde{m}_{ij} + \sum_{k \in \gamma_b(i)} \widetilde{m}_{b_{ik}} & \\ - \sum_{j \in Neigh(i)} \widetilde{T}_{ij}^n \ (\underline{\nabla}(\delta p)^k)_{f_{ij}} \cdot \underline{S}_{ij} - \sum_{k \in \gamma_b(i)} \widetilde{T}_{b_{ik}}^n \ (\underline{\nabla}(\delta p)^k)_{f_{b_{ik}}} \cdot \underline{S}_{b_{ik}} - \Gamma_I & \\ = \sum_{j \in Neigh(i)} m_{ij}^k + \sum_{k \in \gamma_b(i)} m_{b_{ik}}^k - \Gamma_I & \end{aligned} \tag{IV.N.19}$$

with:

$$\left\{ \begin{array}{ll} (\delta(\delta p))^0 & = 0 \\ (\delta(\delta p))^{k+1} & = (\delta p)^{k+1} - (\delta p)^k \quad \forall k \in \mathbb{N} \end{array} \right. \tag{IV.N.20}$$

The facet gradients, denoted $(\underline{\nabla}(\delta p)^k)_{f_{ij}}$ and $(\underline{\nabla}(\delta p)^k)_{f_{b_{ik}}}$, will be reconstructed.

Implementation

We present hereafter the algorithm as it is written in `resopv`.

$\underline{T}^n$ designates an array of dimension 3 containing the local time steps in each direction (for use with

"strengthened velocity-pressure coupling"). We keep the same notation for the "weak velocity-pressure coupling" algorithm although in this case the time steps are equal in the three spatial directions.

• **Computation of the matrix of the equation system to be solved**

- calculation of the diffusion coefficient at the cell faces for use in the Laplacian of pressure (the diffusion coefficient uses the calculation time step or that of the "strengthened velocity-pressure coupling" algorithm). One of two cases will apply, depending on the algorithm selected by the user for the velocity-pressure coupling:

  1. Call to `viscfa` with a total viscosity equal to the time step Δt_I^n for the "weak velocity-pressure coupling" algorithm (`IPUCOU` $= 0$),
  2. Call to `visort` with a diagonal total viscosity for the "strengthened velocity-pressure coupling" algorithm (`IPUCOU` $= 1$). It is at this level that $\widetilde{T}_{ij}^n$ is calculated. The equivalent time steps calculated in the subroutine `cs_velocity_prediction` beforehand are stored in the array `TPUCOU`.

- Call to `matrix` for construction of the pressure diffusion matrix (without the reconstruction terms which cannot be taken into account if we wish to preserve a sparse matrix structure) using the previously calculated viscosity and the array `COEFB` of pressure boundary conditions p^n (we impose a homogeneous Neumann condition on δp for a Neumann condition on p and *vice versa*.

• **Calculation of the normalized residual** `RNORMP`

Since version 1.1.0.s, this step is accomplished within the subroutine `cs_velocity_prediction` and the normalized residual transmitted *via* the variable RNORMP(IPHAS).

At this level of code_saturne the array `TRAV` contains the second member obtained from `cs_velocity_prediction` without the user source terms. The computational procedure for the calculation of `RNORMP` is enumerated below:

1. $\texttt{TRAV}(I) = \widetilde{\underline{u}}_I - \dfrac{\Delta t_I^n}{\rho_I\ |\Omega_i|}\texttt{TRAV}(I) + \dfrac{(\rho_I - \rho_0)\Delta t_I^n}{\rho_I}\underline{g}$,
2. Call to `inimas` to compute the mass flux of the vector `TRAV` (we calculated at each face $\rho_{\,ij}\ \texttt{TRAV}_{\,ij} \,.\, \underline{S}_{\,ij}$, where $\underline{S}$ is the surface vector). We set the total number of sweeps (or iterations) to 1 (`NSWRP` $=$ 1), which means that there is no reconstruction of the gradients during this run through `inimas` (to save computation time). The boundary condition arrays passed into `inimas` contain those of the velocity $\underline{u}^n$.
3. Call to `divmas` to compute the divergence at each cell of the above mass flux, which is stored in the working are `W1`.
4. The mass source terms stored in the array `SMACEL` are added to `W1`.

$$\texttt{W1}(I) = \texttt{W1}(I) - \frac{|\Omega_i|}{\rho_I}\texttt{SMACEL}(I) \qquad \text{(IV.N.21)}$$

5. Call to `prodsc` (`RNORMP` $= \sqrt{\texttt{W1.W1}}$). `RNORMP` will be used in the stop test of the iterative pressure solver to normalize the residual (see routine `gradco` for the conjugate gradient inversion method).

• **Preparation for solving the system**

- Call to `grdcel` for computation of the pressure gradient p^n. The result is stored in `TRAV`. At this level, `TRAV` contains $\frac{\partial p^n}{\partial x}$, $\frac{\partial p^n}{\partial y}$, $\frac{\partial p^n}{\partial z}$.

- Introduction of the explicit pressure cell-based gradient p^n for use with the Rhie & Chow filter.

$$\texttt{TRAV}(I) = \widetilde{\underline{u}}_I + \frac{\texttt{ARAK}}{\rho_I} \underline{\underline{T}}^n_I \, \underline{\nabla} p^n{}_I$$

  `ARAK` represents the "Arakawa" coefficient, so-named within the code, whose default setting is 1 although this value can be modified by the user in `usini1`. To simplify the notations, we define $\texttt{ARAK} = \alpha_{Arak}$.

- Call to `inimas` which calculates the mass flux in `TRAV`. The boundary conditions applied in this case are those of the velocity (*cf.* subroutine `cs_solve_navier_stokes`). This is still just an approximation of the boundary conditions contained in `TRAV`. The mass flux (*cf.* subroutine `inimas` for further details concerning the calculation at the boundary faces) is thus equal to:

$$m_{ij} = \left[\rho \widetilde{\underline{u}} + \alpha_{Arak} \underline{\underline{T}}^n \, \underline{\nabla}(p^n)\right]_{f_{ij}} \cdot \underline{S}_{ij}$$

- Call to `itrmas` to increment the mass flux at the faces[13] by

$$-\alpha_{Arak} \widetilde{T}^n_{ij} \, (\underline{\nabla} p^n)_{f_{ij}} \cdot \underline{S}_{ij}.$$

- Call to `clmlga` to compute the inversion of the pressure matrix with the algebraic multigrid algorithm.

- initialisation of δp, $\delta(\delta p)$ and `SMBR` to 0. `SMBR` will serve to store the second member. In the code, δp and $\delta(\delta p)$ are contained respectively in `RTP(*,IPRIPH)` and `DRTP`.

- Call to `divmas` for calculation of the divergence of the mass flux resulting from the last call to `itrmas`. This divergence is stored in the working array `W7`.

- Addition of the contributions of the mass source terms[14] to `W7`.

$$\texttt{W7}(I) = \texttt{W7}(I) - |\Omega_i| \, \texttt{SMACEL}(I) \tag{IV.N.22}$$

• Loop over the non orthogonalities

Assuming that the mesh is orthogonal, a single inversion would enable resolving the problem. The loop over the non-orthogonal elements is described below:

- Start of the loop at k (thereafter, we are at $k+1$)

  ⋆ Updating of `SMBR` at the start of the loop[15].

$$\texttt{SMBR}(I) = -\texttt{W7}(I) - \texttt{SMBR}(I)$$

  ⋆ Calculation of the norm of `SMBR` in `prodsc`. It is called `RESIDU` in the code. As we solve the system incrementally, the second member must cancel out convergence.

  ⋆ If $\texttt{RESIDU} < 10 \times \varepsilon \times \texttt{RNORMP}$, convergence is attained[16].

[13] $(\underline{\nabla} p^n)_{f_{ij}} \cdot \underline{S}_{ij}$ is the gradient normal to the face that is equal to $\frac{p^n_J - p^n_I}{\overline{IJ}} S_{ij}$ on an orthogonal mesh.

[14] The array `W7` contains the second member without the gradient of δp. It therefore remains invariant at each sweep. On the other hand, `SMBR` contains the entire second member and consequently varies at each sweep.

[15] The sign "-" results from the construction of the matrix.

[16] ε is the pressure convergence tolerance which can be modified by the user in `usini1`, *via* the array `EPSILO`.

⇒ Call to `itrmas` to reupdate the mass flux with the facet gradient $(\underline{\nabla}(\delta p)^k)_f$. We calculate at each face $\widetilde{T}_{ij}^n\ (\underline{\nabla}(\delta p)^k)_{f_{ij}} \cdot \underline{S}_{ij}$ et $\widetilde{T}_{b_{ik}}^n\ (\underline{\nabla}(\delta p)^k)_{f_{b_{ik}}} \cdot \underline{S}_{b_{ik}}$.

⇒ Reupdate[17] of the pressure $p^{n+1} = p^n + \sum_{l=1}^{l=k} (\delta(\delta p))^l$.

⋆ Alternatively,

⇒ $(\delta(\delta p))^{k+1} = 0$,

⇒ Call to `invers` for inversion of the system (IV.N.19). The inversion algorithm stop test data `RESIDU` is normalized by `RNORMP` (see `gradco` for the inversion of the pressure operator).

⋆ If the maximum number of sweeps is attained,

⇒ Call to `itrmas` to increment the mass flux by the pressure gradient $(\delta p)^k$.

⇒ Second call to `itrmas` for incrementation of the mass flux with the non-reconstructed gradient of $(\delta(\delta p))^{k+1}$ to ensure a final divergence-free field is obtained, thereby assuring consistency with the pressure matrix which does not take non orthogonalities into account[18].

⇒ Update of the pressure increment $(\delta p)^{k+1} = (\delta p)^k + (\delta(\delta p))^{k+1}$.

⋆ Otherwise,

⇒ Incrementation of the mass flux taking into account a coefficient of relaxation. $(\delta p)^{k+1} = (\delta p)^k + \texttt{RELAX} \times (\delta(\delta)p)^{k+1}$. The relaxation factor has a default setting of 1; however this can be modified in `usini1`.

⇒ Call to `itrgrp` for calculation of the $\underline{\underline{T}}^n\ \underline{\nabla}(\delta p)$ part of the second member (to which the array `W7` will be added at the start of a (new) loop).

$$\texttt{SMBR}(I) = \sum_{j \in Neigh(i)} \widetilde{T}_{ij}^n\ (\underline{\nabla}(\delta p)^k)_{f_{ij}} \cdot \underline{S}_{ij} + \sum_{k \in \gamma_b(i)} \widetilde{T}_{b_{ik}}^n\ (\underline{\nabla}(\delta p)^k)_{f_{b_{ik}}} \cdot \underline{S}_{b_{ik}}$$

- End of the loop

• Updating of the pressure

We update the pressure with the sum of the increments of δp.

$$p^{n+1} = p^n + (\delta p)_{k_{conv}}$$

where,

$$(\delta p)_{k_{conv}} = \sum_{k=1}^{k=k_{conv}} (\delta(\delta p))^k$$

In practice, `RTP` contains $(\delta p)_{conv}^k$. We therefore increment p^n by `RTP`:

$$\texttt{RTP} = \texttt{RTPA} + \texttt{RTP}$$

[17] $(\delta p)^k = \sum_{l=1}^{l=k} (\delta(\delta p))^l$ is stored in `RTP(*,IPRIPH)`.

[18] Reference can be made to the subroutine `cs_solve_navier_stokes` for further detail.

Points to treat

There are a number of outstanding issues that still need to be resolved:

1. The use of the normal gradient as an approximation of the pressure increment gradient can pose problems in terms of consistency, as indicated notably in the remark below.
2. The approximation $\widetilde{T}^n \approx (\underline{\underline{T}}^n \ \underline{n}) \, . \, \underline{n}$ is not made for the reconstruction of the gradients in the second member of the pressure equation. Nor is it made when calculating the cell-based gradient during application of the Rhie & Chow filter.
3. Use of the weighting factor $\dfrac{1}{2}$ to improve numerical stability when computing calculations based on the values at the faces (e.g. in `itrmas` or `itrgrp` during reconstruction of the pressure increment gradient at the face).
4. We could verify the normalized residual calculation (see `cs_velocity_prediction`).
5. When computing the mass flux of $\widetilde{u} + \dfrac{\alpha}{\rho} \, \underline{\underline{T}} \, \underline{\nabla} p^n$ in `inimas`, we use the boundary conditions of the velocity at the time level n. The validity of this approach remains to be clarified, particularly for the boundary conditions at the outlet. More generally, further analysis of the boundary conditions of the variables in `cs_solve_navier_stokes` is required. This issue is linked to the problem highlighted at the end of `visecv`.
6. During the convergence test for the loop over the non-orthogonal elements, we multiply the tolerance by a factor of 10. Is this really necessary?
7. The problem with the relaxation factor used during updates of the pressure-correction term in the loop over the non orthogonalities (it might be worthwhile replacing this with a dynamic relaxation factor).
8. Use of the Rhie & Chow filter can prove quite problematic in some configurations. However, before undertaking any work to modify this, it would be worthwhile to first verify its utility by assessing whether or not it plays a clear and positive role in any of the configurations.

O- turrij routine

Fonction

Le but de ce sous-programme est de résoudre le système des équations des tensions de Reynolds et de la dissipation ε de manière totalement découplée dans le cadre de l'utilisation du modèle $R_{ij}-\varepsilon$ LRR[1] (option `ITURB` = 30 dans `usini1`).
Le tenseur symétrique des tensions de Reynolds est noté $\underline{\underline{R}}$. Les composantes de ce tenseur représentent le moment d'ordre deux de la vitesse : $R_{ij}=\overline{u_i u_j}$.

Pour chaque composante R_{ij}, on résout :

$$\rho\frac{\partial R_{ij}}{\partial t}+\mathrm{div}\left(\rho\underline{u}\,R_{ij}-\mu\underline{\nabla}R_{ij}\right)= \begin{array}{l}\mathcal{P}_{ij}+\mathcal{G}_{ij}+\Phi_{ij}+d_{ij}-\varepsilon_{ij}+R_{ij}\,\mathrm{div}\left(\rho\underline{u}\right)\\ +\Gamma(R_{ij}^{in}-R_{ij})+\alpha_{R_{ij}}R_{ij}+\beta_{R_{ij}}\end{array} \tag{IV.O.1}$$

$\underline{\underline{\mathcal{P}}}$ est le tenseur de production par cisaillement moyen :

$$\mathcal{P}_{ij}=-\rho\left[R_{ik}\frac{\partial u_j}{\partial x_k}+R_{jk}\frac{\partial u_i}{\partial x_k}\right] \tag{IV.O.2}$$

$\underline{\underline{\mathcal{G}}}$ est le tenseur de production par gravité :

$$\mathcal{G}_{ij}=\left[G_{ij}-C_3(G_{ij}-\frac{1}{3}\delta_{ij}G_{kk})\right] \tag{IV.O.3}$$

avec

$$\left\{\begin{array}{c}G_{ij}=-\frac{3}{2}\frac{C_\mu}{\sigma_t}\frac{k}{\varepsilon}(r_i g_j+r_j g_i)\\ k=\frac{1}{2}R_{ll}\\ r_i=R_{ik}\frac{\partial\rho}{\partial x_k}\end{array}\right. \tag{IV.O.4}$$

Dans ce qui précède, k représente l'énergie turbulente[2], g_i la composante de la gravité dans la direction i, σ_t le nombre de Prandlt turbulent et C_μ, C_3 des constantes définies dans Tab. O.1.

$\underline{\underline{\Phi}}$ est le terme de corrélations pression-déformation. Il est modélisé avec le terme de dissipation $\underline{\underline{\varepsilon}}$ de la manière suivante :

$$\Phi_{ij}-[\varepsilon_{ij}-\frac{2}{3}\rho\,\delta_{ij}\varepsilon]=\phi_{ij,1}+\phi_{ij,2}+\phi_{ij,w} \tag{IV.O.5}$$

Il en résulte :

$$\Phi_{ij}-\varepsilon_{ij}=\phi_{ij,1}+\phi_{ij,2}+\phi_{ij,w}-\frac{2}{3}\rho\,\delta_{ij}\varepsilon \tag{IV.O.6}$$

[1] la description du SSG est prévue pour une version ultérieure de la documentation
[2] Les sommations se font sur l'indice l et on applique plus généralement la convention de sommation d'Einstein.

Le terme $\phi_{ij,1}$ est un terme "lent" de retour à l'isotropie. Il est donné par :

$$\phi_{ij,1} = -\rho\, C_1 \frac{\varepsilon}{k}(R_{ij} - \frac{2}{3}k\delta_{ij}) \tag{IV.O.7}$$

Le terme $\phi_{ij,2}$ est un terme "rapide" d'isotropisation de la production. Il est donné par :

$$\phi_{ij,2} = -C_2(\mathcal{P}_{ij} - \frac{2}{3}\mathcal{P}\delta_{ij}) \tag{IV.O.8}$$

avec,

$$\mathcal{P} = \frac{1}{2}\mathcal{P}_{kk}$$

Le terme $\phi_{ij,w}$ est appelé "terme d'echo de paroi". Il n'est pas utilisé par défaut dans code_saturne, mais peut être activé avec `IRIJEC` $= 1$. Si y représente la distance à la paroi :

$$\begin{aligned}\phi_{ij,w} = \quad & \rho\, C_1' \frac{k}{\varepsilon}\left[R_{km}n_k n_m \delta_{ij} - \frac{3}{2}R_{ki}n_k n_j - \frac{3}{2}R_{kj}n_k n_i\right] f(\frac{l}{y}) \\ & +\rho\, C_2' \left[\phi_{km,2}n_k n_m \delta_{ij} - \frac{3}{2}\phi_{ki,2}n_k n_j - \frac{3}{2}\phi_{kj,2}n_k n_i\right] f(\frac{l}{y})\end{aligned} \tag{IV.O.9}$$

f est une fonction d'amortissement construite pour valoir 1 en paroi et tendre vers 0 en s'éloignant des parois.
La longueur l représente $\dfrac{k^{\frac{3}{2}}}{\varepsilon}$, caractéristique de la turbulence. On prend :

$$f(\frac{l}{y}) = min(1,\ C_\mu^{0,75}\ \frac{k^{\frac{3}{2}}}{\varepsilon\ \kappa y}) \tag{IV.O.10}$$

d_{ij} est un terme de diffusion turbulente[3] qui vaut :

$$d_{ij} = C_S \frac{\partial}{\partial x_k}(\rho \frac{k}{\varepsilon} R_{km} \frac{\partial R_{ij}}{\partial x_m}) \tag{IV.O.11}$$

On notera par la suite $\underline{\underline{A}} = C_S\, \rho\, \dfrac{k}{\varepsilon}\, \underline{\underline{R}}$. Ainsi, $d_{ij} = \mathrm{div}\,(\,\underline{\underline{A}}\,\underline{\nabla}(R_{ij}))$ est une diffusion avec un coefficient tensoriel.

Le terme de dissipation turbulente $\underline{\underline{\varepsilon}}$ est traité dans ce qui précède avec le terme $\underline{\underline{\Phi}}$.

Γ est le terme source de masse[4], R_{ij}^{in} est la valeur de R_{ij} associée à la masse injectée ou retirée.

$(\alpha_{R_{ij}}\, R_{ij} + \beta_{R_{ij}})$ représente le terme source utilisateur (sous-programme `cs_user_turbulence_source_terms`) éventuel avec une décomposition permettant d'impliciter la partie $\alpha_{R_{ij}}\, R_{ij}$ si $\alpha_{R_{ij}} \geqslant 0$.

De même, on résout une équation de convection/diffusion/termes sources pour la dissipation ε. Cette équation est très semblable à celle du modèle $k-\varepsilon$ (voir `cs_turbulence_ke`), seuls les termes de viscosité turbulente et de gravité changent. On résout :

$$\begin{aligned}\rho\frac{\partial \varepsilon}{\partial t} + \mathrm{div}\,[\rho \underline{u}\,\varepsilon - (\mu \underline{\nabla}\varepsilon)] = \quad & d_\varepsilon + C_{\varepsilon_1}\frac{\varepsilon}{k}\,[\mathcal{P} + \mathcal{G}_\varepsilon] - \rho C_{\varepsilon_2}\frac{\varepsilon^2}{k} + \varepsilon \mathrm{div}\,(\rho \underline{u}) \\ & +\Gamma(\varepsilon^{\,in} - \varepsilon) + \alpha_\varepsilon \varepsilon + \beta_\varepsilon\end{aligned} \tag{IV.O.12}$$

[3] Dans la littérature, ce terme contient en général la dissipation par viscosité moléculaire.

[4] Dans ce cas, l'équation de continuité s'écrit : $\dfrac{\partial \rho}{\partial t} + \mathrm{div}\,(\rho \underline{u}) = \Gamma$.

C_μ	C_ε	C_{ε_1}	C_{ε_2}	C_1	C_2	C_3	C_S	C_1'	C_2'
0,09	0,18	1,44	1,92	1,8	0,6	0,55	0,22	0,5	0,3

Table O.1: Définition des constantes utilisées.

d_ε est le terme de diffusion turbulente :

$$d_\varepsilon = C_\varepsilon \frac{\partial}{\partial x_k}\left(\rho \frac{k}{\varepsilon} R_{km} \frac{\partial \varepsilon}{\partial x_m}\right) \tag{IV.O.13}$$

On notera par la suite $\underline{\underline{A'}} = \rho\, C_\varepsilon \frac{k}{\varepsilon} \underline{\underline{R}}$. Le terme de diffusion turbulente est donc modélisé par :

$$d_\varepsilon = \mathrm{div}\left(\underline{\underline{A'}}\, \underline{\nabla}(\varepsilon)\right)$$

La viscosité turbulente usuelle (ν_t) en modèle $k-\varepsilon$ est remplacée par le tenseur visqueux $\underline{\underline{A'}}$.

$\mathcal{P}$ est le terme de production par cisaillement moyen : $\mathcal{P} = \frac{1}{2}\mathcal{P}_{kk}$. Ce terme est modélisé avec la notion de viscosité turbulente dans le cadre du modèle $k-\varepsilon$. Dans le cas présent, il est calculé à l'aide des tensions de Reynolds (à partir de $\mathcal{P}_{ij}$).

$\mathcal{G}_\varepsilon$ est le terme de production des effets de gravité pour la variable ε.

$$\mathcal{G}_\varepsilon = max(0, \frac{1}{2} G_{kk}) \tag{IV.O.14}$$

See the programmers reference of the dedicated subroutine for further details.

Discrétisation

La résolution se fait en découplant totalement les tensions de Reynolds entre elles et la dissipation ε. On résout ainsi une équation de convection/diffusion/termes sources pour chaque variable. Les variables sont résolues dans l'ordre suivant : R_{11}, R_{22}, R_{33}, R_{12}, R_{13}, R_{23} et ε. L'ordre de la résolution n'est pas important puisque l'on a opté pour une résolution totalement découplée en n'implicitant que les termes pouvant être linéarisés par rapport à la variable courante[5].

Les équations sont résolues à l'instant $n+1$.

Variables tensions de Reynolds

Pour chaque composante R_{ij}, on écrit :

$$\begin{aligned} \rho^n \frac{R_{ij}^{n+1} - R_{ij}^n}{\Delta t^n} + \mathrm{div}\left[(\rho \underline{u})^n R_{ij}^{n+1} - \mu^n \underline{\nabla} R_{ij}^{n+1}\right] = \quad & \mathcal{P}_{ij}^n + \mathcal{G}_{ij}^n \\ & + \phi_{ij,1}^{n,n+1} + \phi_{ij,2}^n + \phi_{ij,w}^n \\ & + d_{ij}^{n,n+1} - \frac{2}{3}\rho^n \varepsilon^n \delta_{ij} + R_{ij}^{n+1} \mathrm{div}\,(\rho \underline{u})^n \\ & + \Gamma(R_{ij}^{in} - R_{ij}^{n+1}) \\ & + \alpha_{R_{ij}}^n R_{ij}^{n+1} + \beta_{R_{ij}}^n \end{aligned} \tag{IV.O.15}$$

μ^n est la viscosité moléculaire[6].
L'indice $(n, n+1)$ est relatif à une semi implicitation des termes (voir ci-dessous). Quand seul l'indice

[5] En effet, aucune variable n'est actualisée pour la résolution de la suivante.
[6] La viscosité peut dépendre éventuellement de la température ou d'autres variables.

(n) est utilisé, il suffit de reprendre l'expression des termes et de considérer que toutes les variables sont explicites.

Dans le terme $\phi_{ij,1}^{n,n+1}$ donné ci-dessous, la tension de Reynolds R_{ij} est implicite (les tensions diagonales apparaissent aussi dans l'énergie turbulente k). Ainsi :

$$\phi_{ij,1}^{n,n+1} = -\rho^n\, C_1\, \frac{\varepsilon^n}{k^n} \left[(1 - \frac{\delta_{ij}}{3}) R_{ij}^{n+1} - \delta_{ij} \frac{2}{3} (k^n - \frac{1}{2} R_{ii}^n) \right] \tag{IV.O.16}$$

Le terme de diffusion turbulente $\underline{\underline{d}}$ s'écrit : $d_{ij} = \text{div}\left[\underline{\underline{A}}\, \underline{\nabla} R_{ij}\right]$. Le tenseur $\underline{\underline{A}}$ est toujours explicite. En intégrant sur un volume de contrôle (cellule) Ω_l, le terme $\underline{\underline{d}}$ de diffusion turbulente de R_{ij} s'écrit :

$$\int_{\Omega_l} d_{ij}^{n,n+1}\ d\Omega = \sum_{m \in Vois(l)} \left[\underline{\underline{A}}^n\, \underline{\nabla} R_{ij}^{n+1}\right]_{lm} \cdot \underline{n}_{lm} S_{lm} \tag{IV.O.17}$$

$\underline{n}_{lm}$ est la normale unitaire à la face[7] $\partial\Omega_{lm} = \Gamma_{lm}$ de la frontière $\partial\Omega_l = \underset{m}{\cup} \partial\Omega_{lm}$ de Ω_l, face désignée par abus par lm et S_{lm} sa surface associée.

On décompose $\underline{\underline{A}}^n$ en partie diagonale $\underline{\underline{D}}^n$ et extra-diagonale $\underline{\underline{E}}^n$:

$$\underline{\underline{A}}^n = \underline{\underline{D}}^n + \underline{\underline{E}}^n$$

Ainsi,

$$\begin{aligned} \int_{\Omega_l} d_{ij}\ d\Omega = & \sum_{m \in Vois(l)} \underbrace{\left[\underline{\underline{D}}^n\, \underline{\nabla} R_{ij}\right]_{lm} \cdot \underline{n}_{lm} S_{lm}}_{\text{partie diagonale}} \\ + & \sum_{m \in Vois(l)} \underbrace{\left[\underline{\underline{E}}^n\, \underline{\nabla} R_{ij}\right]_{lm} \cdot \underline{n}_{lm} S_{lm}}_{\text{partie extra-diagonale}} \end{aligned} \tag{IV.O.18}$$

La partie extra-diagonale sera prise totalement explicite et interviendra donc dans l'expression regroupant les termes purement explicites f_s^{exp} du second membre de `codits`.
Pour la partie diagonale, on introduit la composante normale du gradient de la variable principale R_{ij}. Cette contribution normale sera traitée en implicite pour la variable et interviendra à la fois dans l'expression de la matrice simplifiée du système résolu par `codits` et dans le second membre traité par `bilsc2`. La contribution tangentielle sera, elle, purement explicite et donc prise en compte dans f_s^{exp} intervenant dans le second membre de `codits`.
On a :

$$\underline{\nabla} R_{ij} = \underline{\nabla} R_{ij} - (\underline{\nabla} R_{ij} \cdot \underline{n}_{lm})\, \underline{n}_{lm} + (\underline{\nabla} R_{ij} \cdot \underline{n}_{lm})\, \underline{n}_{lm} \tag{IV.O.19}$$

Comme

$$\left[\underline{\underline{D}}^n\ [(\underline{\nabla} R_{ij} \cdot \underline{n}_{lm})\, \underline{n}_{lm}]\right] \cdot \underline{n}_{lm} = \gamma_{lm}^n (\underline{\nabla} R_{ij} \cdot \underline{n}_{lm})$$

avec :

$$\gamma_{lm}^n = (D_{11}^n)\, n_{1,lm}^2 + (D_{22}^n)\, n_{2,lm}^2 + (D_{33}^n)\, n_{3,lm}^2$$

on peut traiter ce terme γ_{lm}^n comme une diffusion avec un coefficient de diffusion indépendant de la direction.

[7] La notion de face purement interne ou de bord n'est pas explicitée ici, pour alléger l'exposé. Pour être rigoureux et homogène avec les notations adoptées, il faudrait distinguer $m \in Vois(l)$ et $m \in \gamma_b(l)$.

Finalement, on écrit :

$$\begin{aligned}
&\int_{\Omega_l} d_{ij}^{n,n+1}\ d\Omega = \\
&+\sum_{m\in Vois(l)} \left[\ \underline{\underline{E}}^n\ \underline{\nabla} R_{ij}^n\right]_{lm} \cdot \underline{n}_{lm} S_{lm} \\
&+\sum_{m\in Vois(l)} \left[\ \underline{\underline{D}}^n\ \underline{\nabla} R_{ij}^n\right]_{lm} \cdot \underline{n}_{lm} S_{lm} \\
&-\sum_{m\in Vois(l)} \gamma_{lm}^n \left(\underline{\nabla} R_{ij}^n \cdot \underline{n}_{lm}\right) S_{lm} + \sum_{m\in Vois(l)} \gamma_{lm}^n \left(\underline{\nabla} R_{ij}^{n+1} \cdot \underline{n}_{lm}\right) S_{lm}
\end{aligned} \tag{IV.O.20}$$

Les trois premiers termes sont totalement explicites et correspondent à la discrétisation de l'opérateur continu :

$$\mathrm{div}\left(\ \underline{\underline{E}}^n\ \underline{\nabla} R_{ij}^n\right) + \mathrm{div}\left(\ \underline{\underline{D}}^n\ [\ \underline{\nabla} R_{ij}^n - (\underline{\nabla} R_{ij}^n \cdot \underline{n})\ \underline{n}\ ]\ \right)$$

en omettant la notion de face.
Le dernier terme est implicite relativement à la variable R_{ij} et correspond à l'opérateur continu :

$$\mathrm{div}\left(\ \underline{\underline{D}}^n\ (\underline{\nabla} R_{ij}^{n+1} \cdot \underline{n})\ \underline{n}\right)$$

Variable ε

On résout l'équation de ε de façon analogue à celle de R_{ij}.

$$\begin{aligned}
\rho^n\ \frac{\varepsilon^{n+1}-\varepsilon^n}{\Delta t^n} + \mathrm{div}\left((\rho\,\underline{u})^n \varepsilon^{n+1}\right) - \mathrm{div}\left(\mu^n\ \underline{\nabla}\varepsilon^{n+1}\right) = \quad & d_\varepsilon^{n,n+1} \\
& + C_{\varepsilon_1} \frac{k^n}{\varepsilon^n}\left[\mathcal{P}^n + \mathcal{G}_\varepsilon^n\right] - \rho^n C_{\varepsilon_2} \frac{(\varepsilon^n)^2}{k^n} \\
& + \varepsilon^{n+1} \mathrm{div}\left(\rho \underline{u}\right)^n \\
& + \Gamma(\varepsilon^{in} - \varepsilon^{n+1}) + \alpha_\varepsilon^n \varepsilon^{n+1} + \beta_\varepsilon^n
\end{aligned} \tag{IV.O.21}$$

Le terme de diffusion turbulente $d_\varepsilon^{n,n+1}$ est traité comme celui des variables R_{ij} et s'écrit :

$$d_\varepsilon^{n,n+1} = \mathrm{div}\left[\ \underline{\underline{A'}}^n\ \underline{\nabla}\varepsilon^{n+1}\right]$$

Le tenseur $\underline{\underline{A'}}$ est toujours explicite. On le décompose en une partie diagonale $\underline{\underline{D'}}^n$ et une partie extra-diagonale $\underline{\underline{E'}}^n$:

$$\underline{\underline{A'}}^n = \underline{\underline{D'}}^n + \underline{\underline{E'}}^n$$

Ainsi :

$$\begin{aligned}
\int_{\Omega_l} d_\varepsilon^{n,n+1}\ d\Omega = &\sum_{m\in Vois(l)} \left[\underline{\underline{E'}}^n\ \underline{\nabla}\varepsilon^n\right]_{lm} \cdot \underline{n}_{lm} S_{lm} \\
+\sum_{m\in Vois(l)} \left[\underline{\underline{D'}}^n\ \underline{\nabla}\varepsilon^n\right]_{lm} \cdot \underline{n}_{lm} S_{lm} &- \sum_{m\in Vois(l)} \eta_{lm}^n \left(\underline{\nabla}\varepsilon^n \cdot \underline{n}_{lm}\right) S_{lm} \\
&+ \sum_{m\in Vois(l)} \eta_{lm}^n \left(\underline{\nabla}\varepsilon^{n+1} \cdot \underline{n}_{lm}\right) S_{lm}
\end{aligned} \tag{IV.O.22}$$

avec :

$$\eta_{lm}^n = (D'^n_{11})\, n^2_{1,\,lm} + (D'^n_{22})\, n^2_{2,\,lm} + (D'^n_{33})\, n^2_{3,\,lm}.$$

On peut traiter ce terme η_{lm}^n comme une diffusion avec un coefficient de diffusion indépendant de la direction.
Les trois premiers termes sont totalement explicites et correspondent à l'opérateur :

$$\mathrm{div}\left(\ \underline{\underline{E'}}^n\ \varepsilon^n\right) + \mathrm{div}\left(\ \underline{\underline{D'}}^n\ [\underline{\nabla}\varepsilon^n - (\underline{\nabla}\varepsilon^n . \underline{n})\ \underline{n}]\ \right)$$

en omettant la notion de face.
Le dernier terme est implicite relativement à la variable ε et correspond à l'opérateur :

$$\mathrm{div}\left(\ \underline{\underline{D'}}^n\ (\underline{\nabla}\varepsilon^{n+1} . \underline{n})\ \underline{n}\right)$$

Mise en œuvre

La numéro de la phase traitée fait partie des arguments de `turrij`. On omettra volontairement de le préciser dans ce qui suit, on indiquera par ( ) la notion de tableau s'y rattachant.

• **Calcul des termes de production** $\underline{\underline{\mathcal{P}}}$

⋆ Initialisation à zéro du tableau `PRODUC` dimensionné à `NCEL` $\times 6$.

⋆ On appelle trois fois `grdcel` pour calculer les gradients des composantes de la vitesse u, v et w prises au temps n.

Au final, on a :

$$\begin{aligned}
\texttt{PRODUC(1,IEL)} &= -2\left[R_{11}^n\frac{\partial u^n}{\partial x}+R_{12}^n\frac{\partial u^n}{\partial y}+R_{13}^n\frac{\partial u^n}{\partial z}\right] \text{ (production de } R_{11}^n)\\
\texttt{PRODUC(2,IEL)} &= -2\left[R_{12}^n\frac{\partial v^n}{\partial x}+R_{22}^n\frac{\partial v^n}{\partial y}+R_{23}^n\frac{\partial v^n}{\partial z}\right] \text{ (production de } R_{22}^n)\\
\texttt{PRODUC(3,IEL)} &= -2\left[R_{13}^n\frac{\partial w^n}{\partial x}+R_{23}^n\frac{\partial w^n}{\partial y}+R_{33}^n\frac{\partial w^n}{\partial z}\right] \text{ (production de } R_{33}^n)\\
\texttt{PRODUC(4,IEL)} &= -\left[R_{12}^n\frac{\partial u^n}{\partial x}+R_{22}^n\frac{\partial u^n}{\partial y}+R_{23}^n\frac{\partial u^n}{\partial z}\right]\\
&\quad -\left[R_{11}^n\frac{\partial v^n}{\partial x}+R_{12}^n\frac{\partial v^n}{\partial y}+R_{13}^n\frac{\partial v^n}{\partial z}\right] \text{ (production de } R_{12}^n)\\
\texttt{PRODUC(5,IEL)} &= -\left[R_{13}^n\frac{\partial u^n}{\partial x}+R_{23}^n\frac{\partial u^n}{\partial y}+R_{33}^n\frac{\partial u^n}{\partial z}\right]\\
&\quad -\left[R_{11}^n\frac{\partial w^n}{\partial x}+R_{12}^n\frac{\partial w^n}{\partial y}+R_{13}^n\frac{\partial w^n}{\partial z}\right] \text{ (production de } R_{13}^n)\\
\texttt{PRODUC(6,IEL)} &= -\left[R_{13}^n\frac{\partial v^n}{\partial x}+R_{23}^n\frac{\partial v^n}{\partial y}+R_{33}^n\frac{\partial v^n}{\partial z}\right]\\
&\quad -\left[R_{12}^n\frac{\partial w^n}{\partial x}+R_{22}^n\frac{\partial w^n}{\partial y}+R_{23}^n\frac{\partial w^n}{\partial z}\right] \text{ (production de } R_{23}^n)
\end{aligned}$$

• **Calcul du gradient de la masse volumique ρ^n prise au début du pas de temps courant**[8] $(n+1)$

Ce calcul n'a lieu que si les termes de gravité doivent être pris en compte (`IGRARI()` $= 1$).

⋆ Appel de `grdcel` pour calculer le gradient de ρ^n dans les trois directions de l'espace. Les conditions aux limites sur ρ^n sont des conditions de Dirichlet puisque la valeur de ρ^n aux faces de bord ik (variable `IFAC`) est connue et vaut $\rho_{b\,ik}$. Pour écrire les conditions aux limites sous la forme habituelle,

$$\rho_{b\,ik} = \texttt{COEFA} + \texttt{COEFB}\,\rho_{I'}^n$$

on pose alors `COEFA` $=$ `PROPCE(IFAC,IPPROB(IROM))` et `COEFB` $=$ `VISCB` $= 0$.
`PROPCE(1,IPPROB(IROM))` (resp.`VISCB`) est utilisé en lieu et place de l'habituel `COEFA` (`COEFB`), lors de l'appel à `grdcel`.
On a donc :
`GRAROX` $= \dfrac{\partial\rho^n}{\partial x}$, `GRAROY` $= \dfrac{\partial\rho^n}{\partial y}$ et `GRAROZ` $= \dfrac{\partial\rho^n}{\partial z}$.

Le gradient de ρ^n servira à calculer les termes de production par effets de gravité si ces derniers sont pris en compte.

• **Boucle `ISOU` de 1 à 6 sur les tensions de Reynolds**

Pour `ISOU` $= 1, 2, 3, 4, 5, 6$, on résout respectivement et dans l'ordre les équations de R_{11}, R_{22}, R_{33},

[8] *i.e.* calculée à partir des variables du pas de temps précédent n si nécessaire.

R_{12}, R_{13} et R_{23} par l'appel au sous-programme `resrij`.
La résolution se fait par incrément $\delta R_{ij}^{n+1,k+1}$, en utilisant la même méthode que celle décrite dans le sous-programme `codits`. On adopte ici les mêmes notations. `SMBR` est le second membre du système à inverser, système portant sur les incréments de la variable. `ROVSDT` représente la diagonale de la matrice, hors convection/diffusion.
On va résoudre l'équation (IV.O.15) sous forme incrémentale en utilisant `codits`, soit :

$$
\begin{aligned}
&\underbrace{\left(\frac{\rho_L^n}{\Delta t^n}+\rho_L^n\,C_1\,\frac{\varepsilon_L^n}{k_L^n}(1-\frac{\delta_{ij}}{3})-m_{lm}^n+\Gamma_L\,+max(-\alpha_{R_{ij}}^n,0)\right)|\Omega_l|\,(\delta R_{ij}^{n+1,p+1})_L}_{\texttt{ROVSDT}\text{ contribuant à la diagonale de la matrice simplifiée de }\texttt{matrix}}\\
&\underbrace{+\sum_{m\in Vois(l)}\left[m_{lm}^n\delta R_{ij,\,f_{lm}}^{n+1,p+1}-(\mu_{lm}^n+\gamma_{lm}^n)\,\frac{(\delta R_{ij}^{n+1,p+1})_M-(\delta R_{ij}^{n+1,p+1})_L)}{\overline{L'M'}}\,S_{lm}\right]}_{\text{convection upwind pur et diffusion non reconstruite relatives à la matrice simplifiée de }\texttt{matrix}\text{[9]}}\\
&=-\frac{\rho_L^n}{\Delta t^n}\,\left((R_{ij}^{n+1,p})_L-(R_{ij}^n)_L\right)\\
&\underbrace{-\int_{\Omega_l}\left(\text{div}\,[\,(\rho\,\underline{u})^n\,R_{ij}^{n+1,p}-(\mu^n\,+\gamma^n\,)\,\underline{\nabla}R_{ij}^{n+1,p}\,]\right)\,d\Omega}_{\text{convection et diffusion traitées par }\texttt{bilsc2}}\\
&+\int_{\Omega_l}\left[\mathcal{P}_{ij}^{n+1,p}+\mathcal{G}_{ij}^{n+1,p}-\rho^n\,C_1\,\frac{\varepsilon^n}{k^n}\left[R_{ij}^{n+1,p}-\frac{2}{3}\,k^n\,\delta_{ij}\right]+\phi_{ij,2}^{n+1,p}+\phi_{ij,w}^{n+1,p}\right]\,d\Omega\\
&+\int_{\Omega_l}\left[-\frac{2}{3}\rho^n\varepsilon^n\delta_{ij}+\Gamma\,(\,R_{ij}^{in}-R_{ij}^{n+1,p}\,)+\alpha_{R_{ij}}^n\,R_{ij}^{n+1,p}+\beta_{R_{ij}}^n\right]\,d\Omega\\
&+\sum_{m\in Vois(l)}\left[\,\underline{\underline{E}}^n\,\underline{\nabla}R_{ij}^{n+1,p}\right]_{lm}\cdot\underline{n}_{lm}S_{lm}\\
&+\sum_{m\in Vois(l)}\left[\,\underline{\underline{D}}^n\,\underline{\nabla}R_{ij}^{n+1,p}\right]_{lm}\cdot\underline{n}_{lm}S_{lm}\\
&-\sum_{m\in Vois(l)}\gamma_{lm}^n\left(\underline{\nabla}R_{ij}^{n+1,p}\cdot\underline{n}_{lm}\right)S_{lm}\\
&+\sum_{m\in Vois(l)}m_{lm}^n\,(R_{ij}^{n+1,p})_L
\end{aligned}
\qquad \text{(IV.O.23)}
$$

où on rappelle :
pour n donné entier positif, on définit la suite $(R_{ij}^{n+1,p})_{p\in\mathbb{N}}$ par :

$$
\left\{\begin{array}{l} R_{ij}^{n+1,0}=R_{ij}^n \\ R_{ij}^{n+1,p+1}=R_{ij}^{n+1,p}+\delta R_{ij}^{n+1,p+1}\end{array}\right.
$$

$(\delta R_{ij}^{n+1,p+1})_L$ désigne la valeur sur l'élément Ω_l du $(\,p+1\,)$-ième incrément de R_{ij}^{n+1}, m_{lm}^n le flux de masse à l'instant n à travers la face lm, $\delta R_{ij,\,f_{lm}}^{n+1,p+1}$ vaut $(\delta R_{ij}^{n+1,p+1})_L$ si $m_{lm}^n \geqslant 0$, $(\delta R_{ij}^{n+1,p+1})_M$ sinon, $\mathcal{P}_{ij}^{n+1,p}$, $\phi_{ij,2}^{n+1,p}$, $\phi_{ij,w}^{n+1,p}$ les valeurs des quantités associées correspondant à l'incrément $(\delta R_{ij}^{n+1,p})$.

Tous ces termes sont calculés comme suit :

- Terme de gauche de l'équation (IV.O.23)
  Dans `resrij` est calculée la variable `ROVSDT`. Les autres termes sont complétés par `codits`, lors de la construction de la matrice simplifiée , *via* un appel au sous-programme `matrix`. La quantité $(\mu_{lm}^n+\gamma_{lm}^n)$ à la face lm est calculée lors de l'appel à `visort`.

[9] Si `IRIJNU` = 1, on remplace μ_{lm}^n par $(\mu+\mu_t)_{lm}^n$ dans l'expression de la diffusion non reconstruite associée à la matrice simplifiée de `matrix` (μ_t désigne la viscosité turbulente calculée comme en $k-\varepsilon$).

- Second membre de l'équation (IV.O.23)
  Le premier terme non détaillé est calculé par le sous-programme `bilsc2`, qui applique le schéma convectif choisi par l'utilisateur, qui reconstruit ou non selon le souhait de l'utilisateur les gradients intervenants dans la convection-diffusion.
  Les termes sans accolade sont, eux, complètement explicites et ajoutés au fur et à mesure dans `SMBR` pour former l'expression f_s^{exp} de `codits`.

On décrit ci-dessous les étapes de `resrij` :

- DELTIJ = 1, pour `ISOU` $\leqslant 3$ et DELTIJ = 0 Si `ISOU` > 3. Cette valeur représente le symbole de Kroeneker δ_{ij}.
- Initialisation à zéro de `SMBR` (tableau contenant le second membre) et `ROVSDT` (tableau contenant la diagonale de la matrice sauf celle relative à la contribution de la diagonale des opérateurs de convection et de diffusion linéarisés [10]), tous deux de dimension `NCEL`.
- Lecture et prise en compte des termes sources utilisateur pour la variable R_{ij}

  Appel à `cs_user_turbulence_source_terms` pour charger les termes sources utilisateurs. Ils sont stockés comme suit. Pour la cellule Ω_l de centre L, représentée par `IEL`, on a :

$$\left\{\begin{array}{ll} \texttt{ROVSDT(IEL)} & = |\Omega_l|\ \alpha_{R_{ij}} \\ \texttt{SMBR(IEL)} & = |\Omega_l|\ \beta_{R_{ij}} \end{array}\right.$$

  On affecte alors les valeurs adéquates au second membre `SMBR` et à la diagonale `ROVSDT` comme suit :

$$\left\{\begin{array}{ll} \texttt{SMBR(IEL)} & = \texttt{SMBR(IEL)} + |\Omega_l|\ \alpha_{R_{ij}}\ (R_{ij}^n)_L \\ \texttt{ROVSDT(IEL)} & = \max\left(-|\Omega_l|\ \alpha_{R_{ij}}, 0\right) \end{array}\right.$$

  La valeur de `ROVSDT` est ainsi calculée pour des raisons de stabilité numérique. En effet, on ne rajoute sur la diagonale que les valeurs positives, ce qui correspond physiquement à impliciter les termes de rappel uniquement.
- Calcul du terme source de masse si $\Gamma_L > 0$

  Appel de `catsma` et incrémentation si nécessaire de `SMBR` et `ROVSDT` *via* :

$$\left\{\begin{array}{l} \texttt{SMBR(IEL)} = \texttt{SMBR(IEL)} + |\Omega_l|\ \Gamma_L\ \left[(R_{ij}^{in})_L - (R_{ij}^n)_L\right] \\ \texttt{ROVSDT(IEL)} = \texttt{ROVSDT(IEL)} + |\Omega_l|\ \Gamma_L \end{array}\right.$$

- Calcul du terme d'accumulation de masse et du terme instationnaire

  On stocke `W1` $= \displaystyle\int_{\Omega_l} \text{div}\ (\rho\,\underline{u})\, d\Omega$ calculé par `divmas` à l'aide des flux de masse aux faces internes $m_{lm}^n = \displaystyle\sum_{m\in Vois(l)} (\rho\underline{u})_{lm}^n . \underline{S}_{lm}$ (tableau `FLUMAS`) et des flux de masse aux bords $m_{b_{lk}}^n = \displaystyle\sum_{k\in\gamma_b(l)} (\rho\underline{u})_{b_{lk}}^n . \underline{S}_{b_{lk}}$ (tableau `FLUMAB`). On incrémente ensuite `SMBR` et `ROVSDT`.

$$\left\{\begin{array}{ll} \texttt{SMBR(IEL)} & = \texttt{SMBR(IEL)} + \texttt{ICONV}\ (R_{ij}^n)_L\ (\displaystyle\int_{\Omega_l} \text{div}\ (\rho\,\underline{u})\ d\Omega) \\ \texttt{ROVSDT(IEL)} & = \texttt{ROVSDT(IEL)} + \texttt{ISTAT}\ \dfrac{\rho_L^n\ |\Omega_l|}{\Delta t^n} - \texttt{ICONV}\ (\displaystyle\int_{\Omega_l} \text{div}\ (\rho\,\underline{u})\ d\Omega) \end{array}\right.$$

- Calcul des termes sources de production, des termes $\phi_{ij,1} + \phi_{ij,2}$ et de la dissipation $-\dfrac{2}{3}\varepsilon\,\delta_{ij}$:

  On effectue une boucle d'indice `IEL` sur les cellules Ω_l de centre L :

$$\Rightarrow\ \texttt{TRPROD} = \frac{1}{2}(\mathcal{P}_{ii}^n)_L = \frac{1}{2}\left[\texttt{PRODUC(1,IEL)} + \texttt{PRODUC(2,IEL)} + \texttt{PRODUC(3,IEL)}\right]$$

[10] qui correspondent aux schémas convectif upwind pur et diffusif sans reconstruction.

$$\Rightarrow \texttt{TRRIJ} = \frac{1}{2}(R_{ii}^n)_L$$

$$\Rightarrow \texttt{SMBR(IEL)} = \texttt{SMBR(IEL)} + \rho_L^n |\Omega_l| \left[\frac{2}{3}\delta_{ij} \left(\frac{C_2}{2}(\mathcal{P}_{ii}^n)_L + (C_1 - 1)\,\varepsilon_L^n \right) + (1 - C_2)\,\texttt{PRODUC(ISOU,IEL)} - C_1 \frac{2\,\varepsilon_L^n}{(R_{ii}^n)_L}(R_{ij}^n)_L \right]$$

$$\Rightarrow \texttt{ROVSDT(IEL)} = \texttt{ROVSDT(IEL)} + \rho_L^n\,|\Omega_l|\,(-\frac{1}{3}\,\delta_{ij} + 1)\,C_1\,\frac{2\,\varepsilon_L^n}{(R_{ii}^n)_L}$$

- Appel de `rijech` pour le calcul des termes d'écho de paroi $\phi_{ij,w}^n$ si `IRIJEC()` $= 1$ et ajout dans `SMBR`.
  `SMBR` $=$ `SMBR` $+ \phi_{ij,w}^n$
  Suivant son mode de calcul (`ICDPAR`), la distance à la paroi est directement accessible par `RA(IDIPAR+IEL-1)` (`|ICDPAR|` $= 1$) ou bien est calculée à partir de `IA(IIFAPA+IEL - 1)`, qui donne pour l'élément `IEL` le numéro de la face de bord paroi la plus proche (`|ICDPAR|` $= 2$). Ces tableaux ont été renseigné une fois pour toutes au début de calcul.

- Appel de `rijthe` pour le calcul des termes de gravité $\mathcal{G}_{ij}^n$ et ajout dans `SMBR`.
  Ce calcul n'a lieu que si `IGRARI()` $= 1$. `SMBR` $=$ `SMBR` $+ \mathcal{G}_{ij}^n$

- Calcul de la partie explicite du terme de diffusion div $\left[\underline{\underline{A}}\,\underline{\nabla} R_{ij}^n\right]$, plus précisément des contributions du terme extradiagonal pris aux faces purement internes (remplissage du tableau `VISCF`), puis aux faces de bord (remplissage du tableau `VISCB`).

  ⋆ Appel de `grdcel` pour le calcul du gradient de R_{ij}^n dans chaque direction. Ces gradients sont respectivement stockés dans les tableaux de travail `W1`, `W2` et `W3`.

  ⋆ boucle d'indice `IEL` sur les cellules Ω_l de centre L pour le calcul de $\underline{\underline{E}}^n\,\underline{\nabla} R_{ij}^n$ aux cellules dans un premier temps :

$$\Rightarrow \texttt{TRRIJ} = \frac{1}{2}(R_{ii}^n)_L$$

$$\Rightarrow \texttt{CSTRIJ} = \rho_L^n\,C_S\,\frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}$$

$$\Rightarrow \texttt{W4(IEL)} = \rho_L^n\,C_S\,\frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}\,[\,(R_{12}^n)_L\,\texttt{W2(IEL)} + (R_{13}^n)_L\,\texttt{W3(IEL)}\,]$$

$$\Rightarrow \texttt{W5(IEL)} = \rho_L^n\,C_S\,\frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}\,[\,(R_{12}^n)_L\,\texttt{W1(IEL)} + (R_{23}^n)_L\,\texttt{W3(IEL)}\,]$$

$$\Rightarrow \texttt{W6(IEL)} = \rho_L^n\,C_S\,\frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}\,[\,(R_{13}^n)_L\,\texttt{W1(IEL)} + (R_{23}^n)_L\,\texttt{W2(IEL)}\,]$$

  ⋆ Appel de `vectds`[11] pour assembler $\left[\underline{\underline{E}}^n\,\underline{\nabla} R_{ij}^n\right] \cdot \underline{n}_{lm} S_{lm}$ aux faces lm.

  ⋆ Appel de `divmas` pour calculer la divergence du flux défini par `VISCF` et `VISCB`. Le résultat est stocké dans `W4`.
  Ajout au second membre `SMBR`.
  `SMBR` $=$ `SMBR` $+$ `W4`

  A l'issue de cette étape, seule la partie extradiagonale de la diffusion prise entièrement explicite :

$$\sum_{m \in Vois(l)} \left[\,\underline{\underline{E}}^n\,\underline{\nabla} R_{ij}^n\right]_{lm} \cdot \underline{n}_{lm} S_{lm}$$

  a été calculée.

[11] Le résultat est stocké dans `VISCF` et `VISCB`. Dans `vectds`, les valeurs aux faces internes sont interpolées linéairement sans reconstruction et `VISCB` est mis à zéro.

- Calcul de la partie diagonale du terme de diffusion[12] :
  Comme on l'a déja vu, une partie de cette contribution sera traitée en implicite (celle relative à la composante normale du gradient) et donc ajoutée au second membre par `bilsc2` ; l'autre partie sera explicite et prise en compte dans f_s^{exp}.

  ⋆ On effectue une boucle d'indice `IEL` sur les cellules Ω_l de centre L :

  ⇒ $\texttt{TRRIJ} = \frac{1}{2}(R_{ii}^n)_L$

  ⇒ $\texttt{CSTRIJ} = \rho_L^n \; C_S \; \frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}$

  ⇒ $\texttt{W4(IEL)} = \rho_L^n \; C_S \; \frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n} \; (R_{11}^n)_L$

  ⇒ $\texttt{W5(IEL)} = \rho_L^n \; C_S \; \frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n} \; (R_{22}^n)_L$

  ⇒ $\texttt{W6(IEL)} = \rho_L^n \; C_S \; \frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n} \; (R_{33}^n)_L$

  ⋆ On effectue une boucle d'indice `IFAC` sur les faces purement internes lm pour remplir le tableau `VISCF` :

  ⇒ Identification des cellules Ω_l et Ω_m de centre respectif L (variable `II`) et M (variable `JJ`), se trouvant de chaque côté de la face lm[13].

  ⇒ Calcul du carré de la surface de la face. La valeur est stockée dans le tableau `SURFN2`.

  ⇒ Interpolation du gradient de R_{ij}^n à la face lm (gradient facette $\left[\underline{\nabla} R_{ij}^n\right]_{lm}$) :

  $$\begin{cases} \texttt{GRDPX} = \frac{1}{2}\left(\texttt{W1(II)} + \texttt{W1(JJ)}\right) \\ \texttt{GRDPY} = \frac{1}{2}\left(\texttt{W2(II)} + \texttt{W2(JJ)}\right) \\ \texttt{GRDPZ} = \frac{1}{2}\left(\texttt{W3(II)} + \texttt{W3(JJ)}\right) \end{cases}$$

  ⇒ Calcul du gradient de R_{ij}^n normal à la face lm, $\left[\underline{\nabla} R_{ij}^n\right]_{lm} . \underline{n}_{lm} \, S_{lm}$.

  `GRDSN` = `GRDPX SURFAC(1,IFAC)` + `GRDPY SURFAC(2,IFAC)` + `GRDPZ SURFAC(3,IFAC)`
  `SURFAC` est le vecteur surface de la face `IFAC`.

  ⇒ calcul de $\left[\underline{\nabla} R_{ij}^n - (\underline{\nabla} R_{ij}^n \,.\, \underline{n}_{lm})\underline{n}_{lm}\right]$, les vecteurs étant calculés à la face lm :

  $$\begin{cases} \texttt{GRDPX} = \texttt{GRDPX} - \frac{\texttt{GRDSN}}{\texttt{SURFN2}} \texttt{SURFAC(1,IFAC)} \\ \texttt{GRDPY} = \texttt{GRDPY} - \frac{\texttt{GRDSN}}{\texttt{SURFN2}} \texttt{SURFAC(2,IFAC)} \\ \texttt{GRDPZ} = \texttt{GRDPZ} - \frac{\texttt{GRDSN}}{\texttt{SURFN2}} \texttt{SURFAC(3,IFAC)} \end{cases}$$

  ⇒ finalisation du calcul de l'expression totalement explicite

  $$\left[\underline{\underline{D}}^n \left(\underline{\nabla} R_{ij}^n - (\underline{\nabla} R_{ij}^n \,.\, \underline{n}_{lm})\,\underline{n}_{lm}\right)\right] . \underline{n}_{lm}$$

[12] Seule la composante normale du gradient de R_{ij} aux faces sera implicite.
[13] La normale à la face est orientée de L vers M.

$$\texttt{VISCF} = \frac{1}{2}(\ \texttt{W4(II)} + \ \texttt{W4(JJ)})\ \texttt{GRDPX SURFAC(1,IFAC)}) +$$
$$\frac{1}{2}(\ \texttt{W5(II)} + \ \texttt{W5(JJ)})\ \texttt{GRDPY SURFAC(2,IFAC)}) +$$
$$\frac{1}{2}(\ \texttt{W6(II)} + \ \texttt{W6(JJ)})\ \texttt{GRDPZ SURFAC(3,IFAC)})$$

⋆ Mise à zéro du tableau `VISCB`.

⋆ Appel de `divmas` pour calculer la divergence de :

$$\underline{\underline{D}}^{\,n}\ \left(\underline{\nabla} R_{ij}^n - (\underline{\nabla} R_{ij}^n \,.\, \underline{n}_{lm})\underline{n}_{lm}\right)$$

défini aux faces dans `VISCF` et `VISCB`.
Le résultat est stocké dans le tableau `W1`.
Ajout au second membre `SMBR`.
`SMBR` = `SMBR` + `W1`

- Calcul de la viscosité orthotrope γ_{lm}^n à la face lm de la variable principale R_{ij}^n
  Ce calcul permet au sous-programme `codits` de compléter le second membre `SMBR` par :

$$\begin{aligned}&\sum_{m\in Vois(l)} \mu_{lm}^n\ \left(\underline{\nabla} R_{ij}^n \,.\, \underline{n}_{lm}\right) S_{lm} + \sum_{m\in Vois(l)} \left(\underline{\nabla} R_{ij}^n \,.\, \underline{n}_{lm}\right)\ \left[\underline{\underline{D}}^{\,n}\ \underline{n}_{lm}\right]_{lm} \,.\, \underline{n}_{lm}\ S_{lm} \\ &= \sum_{m\in Vois(l)} (\ \mu_{lm}^n\ +\ \gamma_{lm}^n\ )\ \left(\underline{\nabla} R_{ij}^n \,.\, \underline{n}_{lm}\right) S_{lm}\end{aligned} \qquad \text{(IV.O.24)}$$

  sans préciser la nature de la face lm, *via* l'appel à `bilsc2` et de disposer de la quantité $(\mu_{lm}^n + \gamma_{lm}^n)$ pour construire sa matrice simplifiée.

  ⋆ On effectue une boucle d'indice `IEL` sur les cellules Ω_l :

  $\Rightarrow$ `TRRIJ` $= \frac{1}{2}(R_{ii}^n)_L$

  $\Rightarrow$ `RCSTE` $= \rho_L^n\ C_S\ \frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}$

  $\Rightarrow$ `W1(IEL)` $= \mu^n + \rho_L^n\ C_S\ \frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}\ (R_{11}^n)_L$

  $\Rightarrow$ `W2(IEL)` $= \mu^n + \rho_L^n\ C_S\ \frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}\ (R_{22}^n)_L$

  $\Rightarrow$ `W3(IEL)` $= \mu^n + \rho_L^n\ C_S\ \frac{(R_{ii}^n)_L}{2\,\varepsilon_L^n}\ (R_{33}^n)_L$

  ⋆ Appel de `visort` pour calculer la viscosité orthotrope [14] $\gamma_{lm}^n = (\underline{\underline{D}}^{\,n}\ \underline{n}_{lm}).\underline{n}_{lm}$ aux faces lm
  Le résultat est stocké dans les tableaux `VISCF` et `VISCB`.

- appel de `codits` pour la résolution de l'équation de convection/diffusion/termes sources de la variable R_{ij}. Le terme source est `SMBR`, la viscosité `VISCF` aux faces purement internes (resp. `VISCB` aux faces de bord) et `FLUMAS` le flux de masse interne (resp. `FLUMAB` flux de masse au bord). Le résultat est la variable R_{ij} au temps $n+1$, donc R_{ij}^{n+1}. Elle est stockée dans le tableau `RTP` des variables mises à jour.

- **Appel de `reseps` pour la résolution de la variable ε**

[14] Comme dans le sous-programme `viscfa`, on multiplie la viscosité par $\frac{S_{lm}}{\overline{L'M'}}$, où S_{lm} et $\overline{L'M'}$ représentent respectivement la surface de la face lm et la mesure algébrique du segment reliant les projections des centres des cellules voisines sur la normale à la face. On garde dans ce sous-programme la possibilité d'interpoler la viscosité aux cellules linéairement ou d'utiliser une moyenne harmonique. La viscosité au bord est celle de la cellule de bord correspondante.

On décrit ci-dessous le sous-programme `reseps`, les commentaires du sous-programme `resrij` ne seront pas répétés puisque les deux sous-programmes ne diffèrent que par leurs termes sources.

- Initialisation à zéro de `SMBR` et `ROVSDT`.
- Lecture et prise en compte des termes sources utilisateur pour la variable ε :

  Appel de `cs_user_turbulence_source_terms` pour charger les termes sources utilisateurs. Ils sont stockés dans les tableaux suivants :
  pour la cellule Ω_l représentée par `IEL` de centre L, on a :

$$\left\{\begin{array}{ll} \texttt{ROVSDT(IEL)} & = |\Omega_l|\ \alpha_\varepsilon \\ \texttt{SMBR(IEL)} & = |\Omega_l|\ \beta_\varepsilon \end{array}\right.$$

  On affecte alors les valeurs adéquates au second membre `SMBR` et à la diagonale `ROVSDT` comme suit :

$$\left\{\begin{array}{ll} \texttt{SMBR(IEL)} & = \texttt{SMBR(IEL)} + \ |\Omega_l|\ \alpha_\varepsilon\ \varepsilon_L^n \\ \texttt{ROVSDT(IEL)} & = \max\ (-\ |\Omega_l|\ \alpha_\varepsilon, 0) \end{array}\right.$$

- Calcul du terme source de masse si $\Gamma_L > 0$:

$$\left\{\begin{array}{l} \texttt{SMBR(IEL)} = \texttt{SMBR(IEL)} + |\Omega_l|\ \Gamma_L\ (\varepsilon_L^{in} - \varepsilon_L^n) \\ \texttt{ROVSDT(IEL)} = \texttt{ROVSDT(IEL)} + |\Omega_l|\ \Gamma_L \end{array}\right.$$

- Calcul du terme d'accumulation de masse et du terme instationnaire
  On stocke `W1` $= \displaystyle\int_{\Omega_l} \text{div}\ (\rho\,\underline{u})\ d\Omega$ calculé par `divmas` à l'aide des flux de masse internes et aux bords.
  On incrémente ensuite `SMBR` et `ROVSDT`.

$$\left\{\begin{array}{ll} \texttt{SMBR(IEL)} & = \texttt{SMBR(IEL)} + \texttt{ICONV}\ \varepsilon_L^n\ (\displaystyle\int_{\Omega_l} \text{div}\ (\rho\,\underline{u})\ d\Omega) \\ \texttt{ROVSDT(IEL)} & = \texttt{ROVSDT(IEL)} + \texttt{ISTAT}\ \dfrac{\rho_L^n\ |\Omega_l|}{\Delta t^n} - \texttt{ICONV}\ (\displaystyle\int_{\Omega_l} \text{div}\ (\rho\,\underline{u})\ d\Omega) \end{array}\right.$$

- Traitement du terme de production $\rho\, C_{\varepsilon_1}\, \dfrac{\varepsilon}{k}\, \mathcal{P}$ et du terme de dissipation $-\,\rho\, C_{\varepsilon_2}\, \dfrac{\varepsilon}{k}\, \varepsilon$
  pour cela, on effectue une boucle d'indice `IEL` sur les cellules Ω_l de centre L :

$$\Rightarrow\ \texttt{TRPROD} = \frac{1}{2}(\mathcal{P}_{ii}^n)_L = \frac{1}{2}\left[\texttt{PRODUC(1,IEL)} + \texttt{PRODUC(2,IEL)} + \texttt{PRODUC(3,IEL)}\right]$$

$$\Rightarrow\ \texttt{TRRIJ}\ = \frac{1}{2}(R_{ii}^n)_L$$

$$\Rightarrow\ \texttt{SMBR(IEL)} = \texttt{SMBR(IEL)} + \rho_L^n|\Omega_l|\left[-C_{\varepsilon_2}\ \frac{2\,(\varepsilon_L^n)^2}{(R_{ii}^n)_L} + C_{\varepsilon_1}\ \frac{\varepsilon_L^n}{(R_{ii}^n)_L}\ (\mathcal{P}_{ii}^n)_L\right]$$

$$\Rightarrow\ \texttt{ROVSDT(IEL)} = \texttt{ROVSDT(IEL)} + C_{\varepsilon_2}\ \rho_L^n\ |\Omega_l|\ \frac{2\,\varepsilon_L^n}{(R_{ii}^n)_L}$$

- Appel de `rijthe` pour le calcul des termes de gravité $\mathcal{G}_\varepsilon^n$ et ajout dans `SMBR`.
  `SMBR` $=$ `SMBR` $+\ \mathcal{G}_\varepsilon^n$
  Ce calcul n'a lieu que si `IGRARI()` $= 1$.
- Calcul de la diffusion de ε
  Le terme div $\left[\mu\,\underline{\nabla}(\varepsilon) + \underline{\underline{A'}}\,\underline{\nabla}(\varepsilon)\right]$ est calculé exactement de la même manière que pour les tensions de Reynolds R_{ij} en remplaçant $\underline{\underline{A}}$ par $\underline{\underline{A'}}$.
- Appel de `codits` pour la résolution de l'équation de convection/diffusion/termes sources de la variable principale ε. Le résultat ε^{n+1} est stocké dans le tableau `RTP` des variables mises à jour.

• **clippings finaux**
On passe enfin dans le sous-programme `clprij` pour faire un clipping éventuel des variables R_{ij}^{n+1} et ε^{n+1}. Le sous-programme `clprij` est appelé[15] avec `ICLIP` = 2 . Cette option [16] contient l'option `ICLIP` = 1 et permet de vérifier l'inégalité de Cauchy-Schwarz sur les grandeurs extra-diagonales du tenseur $\underline{\underline{R}}$ (pour $i \neq j$, $|R_{ij}|^2 \leq R_{ii}R_{jj}$).

Points à traiter

Sauf mention explicite, ϕ représentera une tension de Reynolds ou la dissipation turbulente ($\phi = R_{ij}$ ou ε).

- La vitesse utilisée pour le calcul de la production est explicite. Est-ce qu'une implicitation peut améliorer la précision temporelle de ϕ [17] ?
- Dans quelle mesure le terme d'écho de paroi est-il valide ? En effet, ce terme est remis en question par certains auteurs.
- On peut envisager la résolution d'un système hyperbolique pour les tensions de Reynolds afin d'introduire un couplage avec le champ de vitesse.
- Le flux au bord `VISCB` est annulé dans le sous-programme `vectds`. Peut-on envisager de mettre au bord la valeur de la variable concernée à la cellule de bord correspondant? De même, il faudrait se pencher sur les hypothèses sous-jacentes à l'annulation des contributions aux bords de `VISCB` lors du calcul de :

$$\left[\underline{\underline{D}}^n \left(\underline{\nabla} R_{ij}^n - (\underline{\nabla} R_{ij}^n \cdot \underline{n}_{lm})\, \underline{n}_{lm}\right)\right] \cdot \underline{n}_{lm}.$$

- Un problème de pondération apparaît plus généralement. Si on prend la partie explicite de $\underline{\underline{D}}\,\underline{\nabla}(\phi)$, la pondération aux faces internes utilise le coefficient $\frac{1}{2}$ avec pondération séparée de $\underline{\underline{D}}$ et $\underline{\nabla}(\phi)$, alors que pour $\underline{\underline{E}}\,\underline{\nabla}(\phi)$, on calcule d'abord ce terme aux cellules pour ensuite l'interpoler linéairement aux faces [18]. Ceci donne donc deux types d'interpolations pour des termes de même nature.
- On laisse la possibilité dans `visort` d'utiliser une moyenne harmonique aux faces. Est-ce que ceci est valable puisque les interpolations utilisées lors du calcul de la partie explicite de $\underline{\underline{A}}\,\underline{\nabla}\phi$ sont des moyennes arithmétiques ?
- Les techniques adoptées lors du clipping sont à revoir.
- On utilise dans le cadre du modèle $R_{ij} - \varepsilon$ une semi-implicitation de termes comme $\phi_{ij,1}$ ou $-\rho\, C_{\varepsilon_2} \dfrac{\varepsilon}{k}\, \varepsilon$. On peut envisager le même type d'implicitation dans `cs_turbulence_ke` même en présence du couplage $k - \varepsilon$.
- L'adoption d'une résolution découplée fait perdre l'invariance par rotation.
- La formulation et l'implantation des conditions aux limites de paroi devront être vérifiées. En effet, il semblerait que, dans certains cas, des phénomènes "oscillatoires" apparaissent, sans qu'il soit aisé d'en déterminer la cause.

[15] L'option `ICLIP` = 1 consiste en un clipping minimal des variables R_{ii} et ε en prenant la valeur absolue de ces variables puisqu'elles ne peuvent être que positives.

[16] Quand la valeur des grandeurs R_{ii} ou ε est négative, on la remplace par le minimum entre sa valeur absolue et (1,1) fois la valeur obtenue au pas de temps précédent.

[17] Cette remarque peut être généralisée. En effet, peut-on envisager d'actualiser les variables déjà résolues (sans réactualiser les variables turbulentes après leur résolution)? Ceci obligerait à modifier les sous-programmes tels que `condli` qui sont appelés au début de la boucle en temps.

[18] Cette interpolation se fait dans `vectds` avec des coefficients de pondération aux faces.

- L'implicitation partielle (du fait de la résolution découplée) des conditions aux limites conduit souvent à des calculs instables. Il conviendrait d'en connaître la raison. L'implicitation partielle avait été mise en œuvre afin de tenter d'utiliser un pas de temps plus grand dans le cas de jets axisymétriques en particulier.

P- viscfa routine

Fonction

Dans ce sous-programme est calculé le coefficient de diffusion isotrope aux faces. Ce coefficient fait intervenir la valeur de la viscosité aux faces multipliée par le rapport surface de la face sur la distance algébrique $\overline{I'J'}$ ou $\overline{I'F}$(*cf.* figure IV.P.1), rapport résultant de l'intégration du terme de diffusion. Par analogie du terme calculé, ce sous-programme est aussi appelé par le sous-programme `resopv` pour calculer le coefficient "diffusif" de la pression faisant intervenir le pas de temps.
La valeur de la viscosité aux faces est déterminée soit par une moyenne arithmétique, soit par une moyenne harmonique de la viscosité au centre des cellules, suivant le choix de l'utilisateur. Par défaut, cette valeur est calculée par une moyenne arithmétique.

See the programmers reference of the dedicated subroutine for further details.

Discrétisation

On rappelle dans la figure IV.P.1, la définition des différents points géométriques utilisés par la suite.

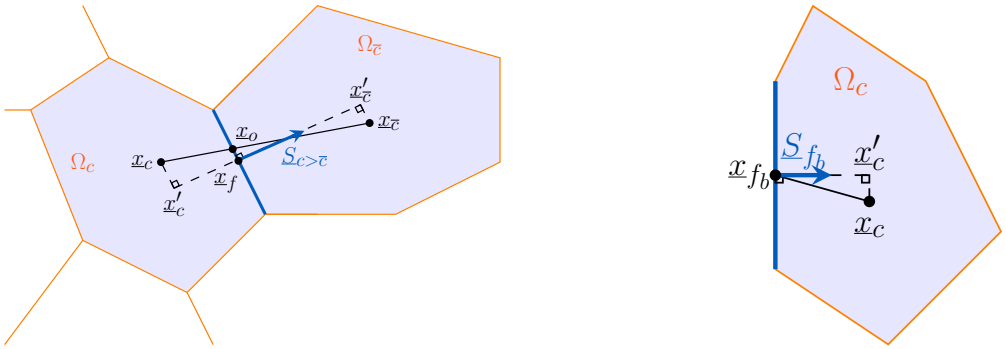


Figure IV.P.1: Définition des différentes entités géométriques pour les faces internes (gauche) et de bord (droite).

L'intégration du terme de diffusion sur une cellule Ω_i est la suivante :

$$\int_{\Omega_i} \mathrm{div}\left(\mu \underline{\nabla}(f)\right) d\Omega = \sum_{j \in Vois(i)} \mu_{ij} \frac{f_{J'} - f_{I'}}{\overline{I'J'}} \cdot S_{ij} + \sum_{k \in \gamma_b(i)} \mu_{b_{ik}} \frac{f_{b_{ik}} - f_{I'}}{\overline{I'F}} \cdot S_{b_{ik}} \qquad \text{(IV.P.1)}$$

Dans ce sous-programme, on calcule les termes de diffusion $\mu_{ij} \dfrac{S_{ij}}{\overline{I'J'}}$ et $\mu_{b_{ik}} \cdot \dfrac{S_{b_{ik}}}{\overline{I'F}}$.

La valeur de la viscosité sur la face interne ij, μ_{ij}, est calculée :

⋆ soit par moyenne arithmétique :

$$\mu_{ij} = \alpha_{ij}\mu_i + (1 - \alpha_{ij})\mu_j \qquad \text{(IV.P.2)}$$

avec $\alpha_{ij} = 0.5$ car ce choix semble stabiliser, bien que cette interpolation soit d'ordre 1 en espace en convergence.

⋆ soit par moyenne harmonique :

$$\mu_{ij} = \frac{\mu_i \, \mu_j}{\alpha_{ij}\mu_i + (1 - \alpha_{ij})\mu_j}$$

avec $\alpha_{ij} = \dfrac{\overline{FJ'}}{\overline{I'J'}}$.

La valeur de la viscosité sur la face de bord ik, $\mu_{b_{ik}}$, est définie ainsi :

$$\mu_{b_{ik}} = \mu_I.$$

Remarque

Lors de l'appel de `viscfa` par le sous-programme `resopv`, le terme à considérer est :

$$\mathrm{div}\,(\,\Delta t^n\ \underline{\nabla}(\delta p)\,)$$

soit :

$$\mu = \mu^n = \Delta t$$

Mise en œuvre

La valeur de la viscosité au centre des cellules est entrée en argument *via* la variable `VISTOT`. On calcule sa valeur moyenne aux faces et on la multiplie par le rapport surface `SURFN` sur la distance algébrique `DIST` pour une face interne (`SURFBN` et `DISTBR` respectivement pour une face de bord). La valeur du terme de diffusion résultant est mise dans le vecteur `VISCF` pour une face interne et `VISCB` pour une face de bord. La variable `IMVISF` détermine quel type de moyenne est utilisé pour calculer la viscosité aux faces.
Si `IMVISF`$= 0$, la moyenne est arithmétique, sinon la moyenne est harmonique.

Points à traiter

L'obtention des interpolations utilisées dans code_saturne et synthétisées dans l'annexe P est détaillée dans le rapport de Davroux et *al.*[1]. Les auteurs de ce rapport ont montré que, pour des solutions régulières et sur un maillage monodimensionnel irrégulier avec une viscosité non constante, la convergence mesurée est d'ordre 2 en espace avec l'interpolation harmonique et d'ordre 1 en espace avec l'interpolation linéaire.

Par conséquent, il serait préférable d'utiliser l'interpolation harmonique pour calculer la valeur de la viscosité aux faces. Des tests de stabilité seront nécessaires au préalable.
De même, on envisage d'extrapoler la viscosité sur les faces de bord plutôt que de prendre la valeur de la viscosité au centre de la cellule jouxtant cette face.
Dans le cas de la moyenne arithmétique, l'utilisation de la valeur 0.5 pour les coefficients α_{ij} serait à revoir.

[1] Davroux A., Archambeau F. et Hérard J.M., Tests numériques sur quelques méthodes de résolution d'une équation de diffusion en volumes finis, HI-83/00/027/A.

Q- `visort` routine

Fonction

Dans ce sous-programme est calculé le coefficient de diffusion "orthotrope" aux faces. Ce type de coefficient se rencontre pour la diffusion de R_{ij} et ε en $R_{ij} - \varepsilon$ (*cf.* `turrij`), ainsi que pour la correction de pression dans le cadre de l'algorithme avec couplage vitesse-pression renforcé (`resopv`). Ce coefficient fait intervenir la valeur de la viscosité aux faces multipliée par le rapport surface de la face sur la distance algébrique $\overline{I'J'}$, rapport résultant de l'intégration du terme de diffusion. La valeur de la viscosité aux faces est basée soit sur une moyenne arithmétique, soit sur une moyenne harmonique de la viscosité au centre des cellules.

See the programmers reference of the dedicated subroutine for further details.

Discrétisation

La figure IV.Q.1 rappelle les diverses définitions géométriques pour les faces internes et les faces de bord.

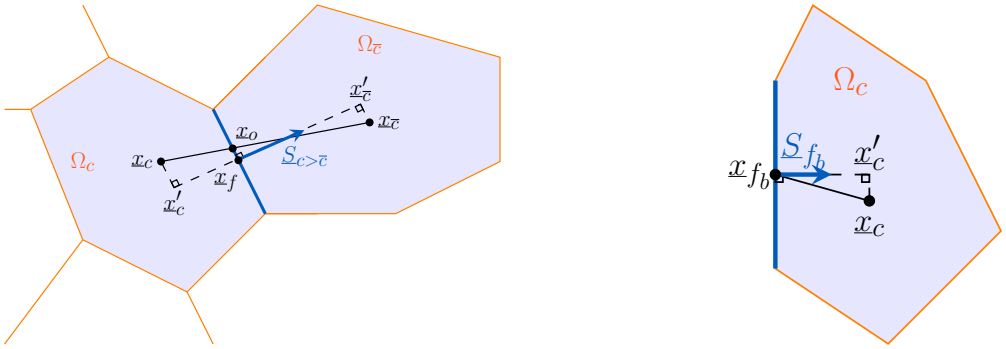


Figure IV.Q.1: Définition des différentes entités géométriques pour les faces internes (gauche) et de bord (droite).

L'intégration du terme de diffusion "orthotrope" sur une cellule est la suivante :

$$\int_{\Omega_i} \text{div}\,(\underline{\underline{\mu}}\,\underline{\nabla} f)\,d\Omega = \sum_{j \in Vois(i)} (\underline{\underline{\mu}}\,\underline{\nabla} f)_{ij} \cdot \underline{S}_{ij} + \sum_{k \in \gamma_b(i)} (\underline{\underline{\mu}}\,\underline{\nabla} f)_{b_{ik}} \cdot \underline{S}_{b_{ik}} \tag{IV.Q.1}$$

avec :

$$\underline{\underline{\mu}} = \begin{bmatrix} \mu_x & 0 & 0 \\ 0 & \mu_y & 0 \\ 0 & 0 & \mu_z \end{bmatrix} \tag{IV.Q.2}$$

et :

$$\begin{aligned} \underline{S}_{ij} &= S_{ij}\underline{n}_{ij} \\ \underline{S}_{b_{ik}} &= S_{b_{ik}}\underline{n}_{b_{ik}} \end{aligned} \tag{IV.Q.3}$$

Le terme $(\underline{\underline{\mu}}\,\underline{\nabla}(f))_{ij}\underline{n}_{ij}$ est calculé à l'aide de la décomposition suivante :

$$(\underline{\underline{\mu}}\,\underline{\nabla} f)_{ij} = (\underline{\nabla} f \cdot \underline{n}_{ij})\,\underline{\underline{\mu}}\,\underline{n}_{ij} + (\underline{\nabla} f . \underline{\tau}_{ij})\,\underline{\underline{\mu}}\,\underline{\tau}_{ij} \tag{IV.Q.4}$$

où $\underline{\tau}_{ij}$ représente un vecteur tangent (unitaire) à la face. Une décomposition similaire est utilisée aux faces de bord.

Dans la matrice, seul le terme $(\underline{\nabla} f \,.\, \underline{n}_{ij})\ \underline{\underline{\mu}}\ \underline{n}_{ij}$ est intégrable facilement en implicite. Par conséquent, la partie projetée sur $\underline{\tau}_{ij}$ est :

- négligée dans le cas du calcul des échelles de temps relatives au couplage vitesse-pression renforcé,
- traitée en explicite dans les termes de diffusion de $R_{ij} - \varepsilon$ (*cf.* `turrij`).

L'intégration implicite du terme de diffusion s'écrit :

$$\int_{\Omega_i} \text{div}\ (\underline{\underline{\mu}}\ \underline{\nabla} f)\, d\Omega = \sum_{j \in Vois(i)} (\underline{\underline{\mu}}\ \underline{n}_{ij}) \,.\, \underline{S}_{ij}\ \frac{f_{J'} - f_{I'}}{\overline{I'J'}} + \sum_{k \in \gamma_b(i)} (\underline{\underline{\mu}}\ \underline{n}_{b_{ik}}) \,.\, \underline{S}_{b_{ik}}\ \frac{f_{b_{ik}} - f_{I'}}{\overline{I'F}} \qquad \text{(IV.Q.5)}$$

Dans ce sous-programme, on calcule le terme $\dfrac{(\underline{\underline{\mu}}\ \underline{n}_{ij}) \,.\, \underline{S}_{ij}}{\overline{I'J'}}$ à l'aide la formule :

$$(\underline{\underline{\mu}}\ \underline{n}_{ij}) \,.\, \underline{n}_{ij} = \mu_{ij}^{moy} = \mu_{ij}^{x} (n_{ij}^{x})^2 + \mu_{ij}^{y} (n_{ij}^{y})^2 + \mu_{ij}^{z} (n_{ij}^{z})^2$$

soit encore :

$$\mu_{ij}^{moy} = \frac{\mu_{ij}^{x} (S_{ij}^{x})^2 + \mu_{ij}^{y} (S_{ij}^{y})^2 + \mu_{ij}^{z} (S_{ij}^{z})^2}{S_{ij}^2}$$

Au bord, on calcule de même :

$$\frac{(\underline{\underline{\mu}}\ \underline{n}_{b_{ik}}) \,.\, \underline{S}_{b_{ik}}}{\overline{I'F}}$$

avec :

$$(\underline{\underline{\mu}}\ \underline{n}_{b_{ik}}) \,.\, \underline{n}_{b_{ik}} = \mu_{b_{ik}}^{moy} = \frac{\mu_{I}^{x} (S_{b_{ik}}^{x})^2 + \mu_{I}^{y} (S_{b_{ik}}^{y})^2 + \mu_{I}^{z} (S_{b_{ik}}^{z})^2}{S_{b_{ik}}^2}$$

La valeur de la viscosité dans une direction l sur la face, μ_{ij}^{l}, est calculée :

- soit par interpolation linéaire :

$$\mu_{ij}^{l} = \alpha_{ij} \mu_{i}^{l} + (1 - \alpha_{ij}) \mu_{j}^{l} \qquad \text{(IV.Q.6)}$$

  avec $\alpha_{ij} = 0.5$ car ce choix semble stabiliser bien que cette interpolation soit d'ordre 1 en espace en convergence,

- soit par interpolation harmonique :

$$\mu_{ij}^{l} = \frac{\mu_{i}^{l}\ \mu_{j}^{l}}{\alpha_{ij} \mu_{i}^{l} + (1 - \alpha_{ij}) \mu_{j}^{l}}$$

  où :

$$\alpha_{ij} = \frac{\overline{FJ'}}{\overline{I'J'}}$$

Mise en œuvre

La viscosité orthotrope au centre des cellules est entrée en argument *via* les variables $\mathtt{W}_1$, $\mathtt{W}_2$ et $\mathtt{W}_3$. On calcule la valeur moyenne de chaque viscosité aux faces de façon arithmétique ou harmonique. Ensuite, on calcule la viscosité équivalente correspondant à $(\underline{\underline{\mu}}\ \underline{n}_{ij}) \,.\, \dfrac{\underline{S}_{ij}}{\overline{I'J'}}$ pour les faces internes et à $(\underline{\underline{\mu}}\ \underline{n}_{b_{ik}}) \,.\, \dfrac{\underline{S}_{b_{ik}}}{\overline{I'F}}$ pour les faces de bord.

Cette écriture fait intervenir les vecteurs surface stockés dans le tableau `SURFAC`, la norme de la surface `SURFN` et la distance algébrique `DIST` pour une face interne (`SURFBO`, `SURFBN` et `DISTBR` respectivement pour une face de bord). La valeur du terme de diffusion résultant est mise dans le vecteur `VISCF` (`VISCB` aux faces de bord).
La variable `IMVISF` détermine quel type de moyenne est utilisé pour calculer la viscosité dans une direction à la face. Si `IMVISF`= 0, alors la moyenne est arithmétique, sinon la moyenne est harmonique).

Points à traiter

L'obtention des interpolations utilisées dans le code code_saturne du paragraphe Q est résumée dans le rapport de Davroux et al[1]. Les auteurs de ce rapport ont montré que, pour un maillage monodimensionnel irrégulier et avec une viscosité non constante, la convergence mesurée est d'ordre 2 en espace avec l'interpolation harmonique et d'ordre 1 en espace avec l'interpolation linéaire (pour des solutions régulières). Par conséquent, il serait préférable d'utiliser l'interpolation harmonique pour calculer la valeur de la viscosité aux faces. Des tests de stabilité seront nécessaires au préalable.
De même, on envisage d'extrapoler la viscosité sur les faces de bord plutôt que de prendre la valeur de la viscosité de la cellule jouxtant cette face.
Dans le cas de la moyenne arithmétique, l'utilisation de la valeur 0.5 pour les coefficients α_{ij} serait à revoir.

[1] Davroux A., Archambeau F. et Hérard J.M., Tests numériques sur quelques méthodes de résolution d'une équation de diffusion en volumes finis, HI-83/00/027/A.

R- visecv routine

Fonction

Dans ce sous-programme sont calculés les termes de gradient transposé et de viscosité secondaire (fluide newtonien). Ils seront traités en explicite pur. En effet, si on traitait ces termes en implicite, cela reviendrait à coupler les équations des différentes composantes de la vitesse, ce qui n'est pas compatible avec le schéma de résolution actuel (*cf.* sous-programme `cs_solve_navier_stokes`).

See the programmers reference of the dedicated subroutine for further details.

Discrétisation

L'intégration des termes de gradient transposé div $(\mu_{tot}\,{}^t\underline{\underline{\mathrm{grad}}}\,(\underline{v}))$ et de viscosité secondaire $-\frac{2}{3}\underline{\nabla}(\mu_{tot}\,\mathrm{div}\,(\underline{v}))$ est la suivante[1] :

$$\begin{aligned}
&\int_{\Omega_i} \mathrm{div}\,(\mu_{tot}\,{}^t\underline{\underline{\mathrm{grad}}}\,(\underline{v}))\,d\Omega \\
&= \sum_{l=x,y,z}\left[\sum_{j\in Vois(i)} \mu_{tot,ij}((\frac{\partial v_x}{\partial l})_{moy,ij}\,n^x_{ij} + (\frac{\partial v_y}{\partial l})_{moy,ij}\,n^y_{ij} + (\frac{\partial v_z}{\partial l})_{moy,ij}\,n^z_{ij})S_{ij}\right. \\
&\left. + \sum_{k\in\gamma_b(i)} \mu_{tot,b_{ik}}((\frac{\partial v_x}{\partial l})_{moy,b_{ik}}\,n^x_{b_{ik}} + (\frac{\partial v_y}{\partial l})_{moy,b_{ik}}\,n^y_{b_{ik}} + (\frac{\partial v_z}{\partial l})_{moy,b_{ik}}\,n^z_{b_{ik}})S_{b_{ik}}\right]\underline{e}_l \\
&-\frac{2}{3}\int_{\Omega_i}\underline{\nabla}(\mu_{tot}\mathrm{div}\,(\underline{v}))\,d\Omega \\
&= -\frac{2}{3}\sum_{l=x,y,z}\left[\sum_{j\in Vois(i)}(\mu_{tot}\,\mathrm{div}\,(\underline{v}))_{ij}S^l_{ij} + \sum_{k\in\gamma_b(i)}(\mu_{tot}\,\mathrm{div}\,(\underline{v}))_{b_{ik}}S^l_{b_{ik}}\right]\underline{e}_l
\end{aligned}$$

Le terme de viscosité μ_{tot} est calculé en fonction du modèle de turbulence utilisé :

$$\mu_{tot} = \begin{cases} \mu + \mu_t & \text{pour les modèles à viscosité turbulente ou en LES,} \\ \mu & \text{pour les modèles au second ordre ou en laminaire.} \end{cases}$$

où μ et μ_t représentent respectivement la viscosité dynamique moléculaire et la viscosité dynamique turbulente.

Mise en œuvre

Terme de gradient transposé

Avant de calculer l'intégration des deux termes, on calcule dans un premier temps la viscosité totale en fonction du modèle de turbulence considéré.
Les termes calculés dans ce sous-programme, appelé par `cs_velocity_prediction`, interviennent dans le second membre de l'équation de quantité de mouvement, et sont donc directement stockés dans le tableau correspondant `TRAV`.

[1] la viscosité de volume κ est supposée nulle, cf. `cs_solve_navier_stokes`

Le terme div $(\mu_{tot}\,{}^t\underline{\underline{\text{grad}}}\,(\underline{v}))$ est calculé en opérant comme suit.
On effectue une boucle sur les composantes v_α où $\alpha = x, y, z$ de la vitesse (α correspond à `ISOU` dans le code) :

- on calcule le gradient cellule de v_α par un appel au sous-programme `grdcel`.
- on initialise un tableau nommé $\texttt{W}_6$ à la valeur 1 pour les cellules internes, et à la valeur 0 pour les cellules de bord. Ce tableau sert par la suite à ne pas considérer la contribution du terme de gradient transposé sur les cellules de bord. En effet, on ne sait pas écrire de conditions aux limites correctes pour le gradient transposé. On préfère donc tout simplement annuler son effet sur les cellules de bord (*cf.* paragraphe R).
- pour chaque direction l ($l = x, y, z$), l correspondant à `IDIM` dans le code, on calcule pour chaque cellule Ω_i dont le centre correspond à la variable `II` ou `JJ` (pour les centres voisins) dans le code :

$$\begin{aligned}
&\texttt{TRAV}(i,l) = \texttt{TRAV}(i,l) \\
&+\texttt{W}_6(i)\left[\sum_{j\in Vois(i)} \mu_{tot,ij}\,(\frac{\partial v_\alpha}{\partial l})_{moy,ij}\,S^\alpha_{ij} \;+\; \sum_{k\in\gamma_b(i)} \mu_{tot,b_{ik}}\,(\frac{\partial v_\alpha}{\partial l})_{moy,b_{ik}}\,S^\alpha_{b_{ik}}\right] \\
&\text{avec } (\frac{\partial v_\alpha}{\partial l})_{moy,ij} = \frac{1}{2}\left[(\frac{\partial v_\alpha}{\partial l})_i + (\frac{\partial v_\alpha}{\partial l})_j\right]
\end{aligned}$$

Fin de la boucle sur les composantes de la vitesse.

Terme de viscosité secondaire

Le terme de seconde viscosité $-\frac{2}{3}\underline{\nabla}(\mu_{tot}\text{div}\,(\underline{v}))$ est calculé de la façon suivante :

- on calcule la valeur de la vitesse sur la face ij en divisant le flux masse connu à la face par la densité moyenne $\rho_{moy,ij}$ de la face ($\rho_{moy,ij} = \frac{\rho_i + \rho_j}{2}$).
- on calcule ensuite l'intégrale volumique de la divergence de la vitesse sur chaque cellule en appelant le sous-programme `divmas`.
- on calcule alors pour chaque cellule Ω_i le terme $-\frac{2}{3}(\mu_{tot}\,\text{div}\,(\underline{v}))_i$ que l'on met dans le tableau de travail $\texttt{W}_4$. La valeur de ce terme sur la face interne ij est obtenue en prenant la moyenne arithmétique des valeurs des deux cellules avoisinantes (tableau `VISCF`) et celle sur la face de bord est prise égale la valeur de la cellule avoisinante (tableau `VISCB`).
- on calcule alors pour chaque direction l le terme final, *i.e.* :

$$\texttt{TRAV}(i,l) = \texttt{TRAV}(i,l) - \frac{2}{3}\left[\sum_{j\in Vois(i)} (\mu_{tot}\,\text{div}\,(\underline{v}))_{moy,ij}\;S^l_{ij} + \sum_{k\in\gamma_b(i)} (\mu_{tot}\text{div}\,(\underline{v}))_{moy,b_{ik}}\;S^l_{b_{ik}}\right]$$

Le traitement est similaire pour le terme de viscosité de volume dans le module compressible.

Points à traiter

L'intégration du terme de gradient transposé pose un problème de compatibilité. En effet, le gradient transposé provient de l'écriture de la divergence du tenseur des contraintes (*cf.* `cs_velocity_prediction`),

soit :

$$\text{div}\ (\underline{\underline{\sigma}}) = \text{div}\ (-p\underline{\underline{Id}} + \underline{\underline{\tau}})$$

où :

$$\underline{\underline{\tau}} = 2\mu\ [\underbrace{\frac{1}{2}\ (\underline{\underline{\text{grad}}}\ \underline{v}\ +\ {}^t\underline{\underline{\text{grad}}}\ \underline{v})}_{\text{partie 1}} - \underbrace{\frac{2}{3}\ tr(\ \frac{1}{2}\ (\underline{\underline{\text{grad}}}\ \underline{v}\ +\ {}^t\underline{\underline{\text{grad}}}\ \underline{v}))\ \underline{\underline{Id}}}_{\text{partie 2}}\ ]$$

Or, lorsque l'on intègre la première partie de la divergence de $\underline{\underline{\tau}}$, on implicite le terme div $(\mu\ \underline{\underline{\text{grad}}}\ \underline{v})$ et on explicite le gradient transposé div $(\mu\ {}^t\underline{\underline{\text{grad}}}\ \underline{v})$. Ce traitement fait intervenir la vitesse au centre des cellules. Elle ne vérifie pas exactement la condition div $(\rho\underline{v}) = 0$. En effet, au cours de l'étape de correction, on utilise un filtre Rhie et Chow (*cf.* `resopv`) et la vitesse n'est mise à jour qu'à la fin de l'étape. Par contre, lorsque l'on intègre la deuxième partie de la divergence de $\underline{\underline{\tau}}$ de façon explicite, on utilise la vitesse issue du flux masse aux faces qui vérifie la condition div $(\rho\underline{v}) = 0$ (du moins à ρ constant, l'interpolation de ρ à la face étant également un point à considérer). Ainsi, la discrétisation de ces deux parties n'est pas totalement cohérente. Il serait utile de baser la discrétisation de ces termes sur une vitesse vérifiant la contrainte div $(\rho\underline{v}) = 0$.
Pour la même raison, il est difficile de connaître les conditions aux limites du terme en gradient transposé. Sur les cellules de bord, on sait uniquement que la contrainte totale normale doit équilibrer le frottement et toutes les autres forces. Or, le tenseur des contraintes est scindé en une partie explicite et une partie implicite, donc c'est un peu difficile d'utiliser cette condition physique.
Actuellement, la contribution aux cellules de bord du terme de gradient transposé est annulée, ce qui élimine l'influence des conditions aux limites mais n'est naturellement pas satisfaisant. Quelques essais d'intégration des conditions aux limites pour ce terme n'ont pas été concluants jusqu'à présent. Cependant, des essais supplémentaires sont envisageables.

Part V

Compressible module

A- cfbl** routine

Fonction

On s'intéresse à la résolution des équations de Navier-Stokes en compressible, en particulier pour des configurations sans choc. Le schéma global correspond à une extension des algorithmes volumes finis mis en œuvre pour simuler les équations de Navier-Stokes en incompressible.

Dans les grandes lignes, le schéma est constitué d'une étape "acoustique" fournissant la masse volumique (ainsi qu'une prédiction de pression et un débit acoustique), suivie de la résolution de l'équation de la quantité de mouvement ; on résout ensuite l'équation de l'énergie et, pour terminer, la pression est mise à jour. Moyennant une contrainte sur la valeur du pas de temps, le schéma permet d'assurer la positivité de la masse volumique.

La thermodynamique prise en compte à ce jour est celle des gaz parfaits, mais l'organisation du code à été prévue pour permettre à l'utilisateur de fournir ses propres lois.

Pour compléter la présentation, on pourra se reporter à la référence suivante :
[**Mathon**] P. Mathon, F. Archambeau, J.-M. Hérard : "Implantation d'un algorithme compressible dans code_saturne ", HI-83/03/016/A

Le cas de validation "tube à choc" de la version 1.2 de code_saturne permettra également d'apporter quelques compléments (tube à choc de Sod, discontinuité de contact instationnaire, double détente symétrique, double choc symétrique).

Notations

Symbole	Unité	Signification
C_p, C_{p_i}	$J/(kg.K)$	capacité calorifique à pression constante $C_p = \frac{\partial h}{\partial T}\big)_P$
C_v, C_{vi}	$J/(kg.K)$	capacité calorifique à volume constant $C_v = \frac{\partial \varepsilon}{\partial T}\big)_\rho$
$\mathcal{D}_{f/b}$	m^2/s	diffusivité moléculaire du composant f dans le bain
E	J/m^3	énergie totale volumique $E = \rho e$
F		centre de gravité d'une face
H	J/kg	enthalpie totale massique $H = \frac{E+P}{\rho}$
I		point de co-location de la cellule i
I'		pour une face ij partagée entre les cellules i et j, I' est le projeté de I sur la normale à la ij passant par F, centre de ij
K	$kg/(m.s)$	diffusivité thermique
M, M_i	kg/mol	masse molaire (M_i pour le constituant i)
P	Pa	pression
$\underline{Q}$	$kg/(m^2.s)$	vecteur quantité de mouvement $\underline{Q} = \rho \underline{u}$
$\underline{Q}_{ac}$	$kg/(m^2.s)$	vecteur quantité de mouvement issu de l'étape acoustique
Q	$kg/(m^2.s)$	norme de $\underline{Q}$
R	$J/(mol.K)$	constante universelle des gaz parfaits
S	$J/(K.m^3)$	entropie volumique
$\mathcal{S}$	$[f].kg/(m^3.s)$	Terme de production/dissipation volumique pour le scalaire f
T	K	température (> 0)
Y_i		fraction massique du composé i ($0 \leqslant Y_i \leqslant 1$)

Symbole	Unité	Signification
c^2	$(m/s)^2$	carré de la vitesse du son $c^2 = \frac{\partial P}{\partial \rho}\Big)_s$
e	J/kg	énergie totale massique $e = \varepsilon + \frac{1}{2}u^2$
$\underline{f}_v$	N/kg	$\rho \underline{f}_v$ représente le terme source volumique pour la quantité de mouvement : gravité, pertes de charges, tenseurs des contraintes turbulentes, forces de Laplace...
$\underline{g}$	m/s^2	accélération de la pesanteur
h	J/kg	enthalpie massique $h = \varepsilon + \frac{P}{\rho}$
i		indice faisant référence à la cellule i ; f_i est la valeur de la variable f associée au point de co-location I
I'		indice faisant référence à la cellule i ; f'_I est la valeur de la variable f associée au point I'
$\underline{j} \wedge \underline{B}$	N/m^3	forces de Laplace
r, r_i	$J/(kg.K)$	constante massique des gaz parfaits $r = \frac{R}{M}$ (pour le constituant i, on a $r_i = \frac{R}{M_i}$)
s	$J/(K.kg)$	entropie massique
t	s	temps
$\underline{u}$	m/s	vecteur vitesse
u	m/s	norme de $\underline{u}$

Symbole	Unité	Signification
β	$kg/(m^3.K)$	$\beta = \frac{\partial P}{\partial s}\big)_\rho$
γ	$kg/(m^3.K)$	constante caractéristique d'un gaz parfait $\gamma = \frac{C_p}{C_v}$
ε	J/kg	énergie interne massique
κ	$kg/(m.s)$	viscosité dynamique en volume
λ	$W/(m.K)$	conductivité thermique
μ	$kg/(m.s)$	viscosité dynamique ordinaire
ρ	kg/m^3	densité
$\underline{\varphi}_f$	$[f].kg/(m^2.s)$	vecteur flux diffusif du composé f
φ_f	$[f].kg/(m^2.s)$	norme de $\underline{\varphi}_f$
$\underline{\underline{\Sigma}}^v$	$kg/(m^2.s^2)$	tenseur des contraintes visqueuses
$\underline{\Phi}_s$	W/m^2	vecteur flux conductif de chaleur
Φ_s	W/m^2	norme de $\underline{\Phi}_s$
Φ_v	W/kg	$\rho\Phi_v$ représente le terme source volumique d'énergie, comprenant par exemple l'effet Joule $\underline{j} \cdot \underline{E}$, le rayonnement...

Système d'équations laminaires de référence

L'algorithme développé propose de résoudre l'équation de continuité, les équations de Navier-Stokes ainsi que l'équation d'énergie totale de manière conservative, pour des écoulements compressibles.

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \operatorname{div}\left(\underline{Q}\right) = 0 \\ \dfrac{\partial \underline{Q}}{\partial t} + \underline{\operatorname{div}}\left(\underline{u} \otimes \underline{Q}\right) + \underline{\nabla} P = \rho \underline{f}_v + \underline{\operatorname{div}}\left(\underline{\underline{\Sigma}}^v\right) \\ \dfrac{\partial E}{\partial t} + \operatorname{div}\left(\underline{u}(E+P)\right) = \rho \underline{f}_v \cdot \underline{u} + \operatorname{div}\left(\underline{\underline{\Sigma}}^v \underline{u}\right) - \operatorname{div}\, \underline{\Phi}_s + \rho \Phi_v \end{cases} \tag{V.A.1}$$

Nous avons présenté ici le système d'équations laminaires, mais il faut préciser que la turbulence ne pose pas de problème particulier dans la mesure où les équations supplémentaires sont découplées du système (V.A.1).

Expression des termes intervenant dans les équations

- Énergie totale volumique :

$$E = \rho e = \rho \varepsilon + \frac{1}{2} \rho u^2 \tag{V.A.2}$$

  avec l'énergie interne $\varepsilon(P, \rho)$ donnée par l'équation d'état

- Forces volumiques : $\rho \underline{f}_v$ (dans la plupart des cas $\rho \underline{f}_v = \rho \underline{g}$)

- Tenseur des contraintes visqueuses pour un fluide Newtonien :

$$\underline{\underline{\Sigma}}^v = \mu(\underline{\underline{\nabla u}} + {}^t\underline{\underline{\nabla u}}) + (\kappa - \frac{2}{3}\mu)\operatorname{div}\, \underline{u}\, \underline{\underline{Id}} \tag{V.A.3}$$

  avec $\mu(T, \ldots)$ et $\kappa(T, \ldots)$ mais souvent $\kappa = 0$

- Flux de conduction de la chaleur : loi de Fourier

$$\underline{\Phi}_s = -\lambda \underline{\nabla} T \tag{V.A.4}$$

  avec $\lambda(T, \ldots)$

- Source de chaleur volumique : $\rho \Phi_v$

Équations d'état et expressions de l'énergie interne

Gaz parfait

Équation d'état : $P = \rho r T$

Énergie interne massique : $\varepsilon = \dfrac{P}{(\gamma - 1)\rho}$

Soit :

$$P = (\gamma - 1)\rho(e - \frac{1}{2}u^2) \tag{V.A.5}$$

Mélange de gaz parfaits

On considère un mélange de N constituants de fractions massiques $(Y_i)_{i=1...N}$

Équation d'état : $P = \rho\, r_{mélange}\, T$

Énergie interne massique : $\varepsilon = \dfrac{P}{(\gamma_{mélange} - 1)\rho}$

Soit :

$$P = (\gamma_{mélange} - 1)\rho(e - \frac{1}{2}u^2) \tag{V.A.6}$$

avec $\gamma_{mélange} = \dfrac{\sum_{i=1}^{N} Y_i C_{pi}}{\sum_{i=1}^{N} Y_i C_{vi}}$ et $r_{mélange} = \sum_{i=1}^{N} Y_i r_i$

Equation d'état de Van der Waals

Cette équation est une correction de l'équation d'état des gaz parfaits pour tenir compte des forces intermoléculaires et du volume des molécules constitutives du gaz. On introduit deux coefficients correctifs : a $[Pa.\, m^6/kg^2]$ est lié aux forces intermoléculaires et b $[m^3/kg]$ est le covolume (volume occupé par les molécules).

Équation d'état : $(P + a\rho^2)(1 - b\rho) = \rho r T$

Énergie interne massique : $\varepsilon = \dfrac{(P + a\rho^2)(1 - b\rho)}{(\hat{\gamma} - 1)\rho} - a\rho$

Soit :

$$P = (\hat{\gamma} - 1)\frac{\rho}{(1 - b\rho)}(e - \frac{1}{2}u^2 + a\rho) - a\rho^2 \tag{V.A.7}$$

avec $\hat{\gamma} = 1 + \dfrac{r}{C_v} = \dfrac{C_p}{C_v}\left(\dfrac{P - a\rho^2(1 - 2b\rho)}{P + a\rho^2}\right) + \dfrac{2a\rho^2(1 - b\rho)}{P + a\rho^2}$

Calcul des grandeurs thermodymamiques

Pour un gaz parfait à γ constant

Equation d'état : $P = \rho r T$

On suppose connues la chaleur massique à pression constante C_p et la masse molaire M du gaz, ainsi que les variables d'état.

Chaleur massique à volume constant : $C_v = C_p - \dfrac{R}{M} = C_p - r$

Constante caractéristique du gaz : $\gamma = \dfrac{C_p}{C_v} = \dfrac{C_p}{C_p - r}$

Vitesse du son : $c^2 = \gamma \dfrac{P}{\rho}$

Entropie : $s = \dfrac{P}{\rho^\gamma}$ et $\beta = \dfrac{\partial P}{\partial s}\Big)_\rho = \rho^\gamma$

Remarque : L'entropie choisie ici n'est pas l'entropie physique, mais une entropie mathématique qui vérifie $c^2 \dfrac{\partial s}{\partial P}\Big)_\rho + \dfrac{\partial s}{\partial \rho}\Big)_P = 0$

Pression : $P = (\gamma - 1)\rho\varepsilon$

Energie interne : $\varepsilon = C_v T = \dfrac{1}{\gamma - 1}\dfrac{P}{\rho}$ avec $\varepsilon_{sup} = 0$

Enthalpie : $h = C_p T = \dfrac{\gamma}{\gamma - 1}\dfrac{P}{\rho}$

Pour un mélange de gaz parfaits

Une intervention de l'utilisateur dans le sous-programme utilisateur `uscfth` est nécessaire pour pouvoir utiliser ces lois.

Equation d'état : $P = \rho\, r_{mél}\, T$ avec $r_{mél} = \displaystyle\sum_{i=1}^{N} Y_i r_i = \sum_{i=1}^{N} Y_i \frac{R}{M_i}$

On suppose connues la chaleur massique à pression constante des différents constituants C_{p_i}, la masse molaire M_i des constituants du gaz, ainsi que les variables d'état (dont les fractions massiques Y_i).

Masse molaire du mélange : $M_{mél} = \left(\displaystyle\sum_{i=1}^{N} \frac{Y_i}{M_i} \right)^{-1}$

Chaleur massique à pression constante du mélange :

$C_{p_{mél}} = \displaystyle\sum_{i=1}^{N} Y_i C_{p_i}$

Chaleur massique à volume constant du mélange :

$C_{v\,mél} = \displaystyle\sum_{i=1}^{N} Y_i C_{vi} = C_{p_{mél}} - \frac{R}{M_{mél}} = C_{p_{mél}} - r_{mél}$

Constante caractéristique du gaz : $\gamma_{mél} = \dfrac{C_{p_{mél}}}{C_{v\,mél}} = \dfrac{C_{p_{mél}}}{C_{p_{mél}} - r_{mél}}$

Vitesse du son : $c^2 = \gamma_{mél}\dfrac{P}{\rho}$

Entropie : $s = \dfrac{P}{\rho^{\gamma_{mél}}}$ et $\beta = \left(\dfrac{\partial P}{\partial s}\right)_\rho = \rho^{\gamma_{mél}}$

Pression : $P = (\gamma_{mél} - 1)\rho\varepsilon$

Energie interne : $\varepsilon = C_{v\,mél}\, T$ avec $\varepsilon_{sup} = 0$

Enthalpie : $h = C_{p_{mél}}\, T = \dfrac{\gamma_{mél}}{\gamma_{mél} - 1}\dfrac{P}{\rho}$

Pour un gaz de Van der Waals

Ces lois n'ont pas été programmées, mais l'utilisateur peut intervenir dans le sous-programme utilisateur `uscfth` s'il souhaite le faire.

Equation d'état : $(P + a\rho^2)(1 - b\rho) = \rho r T$

avec a $[Pa.\,m^6/kg^2]$ lié aux forces intermoléculaires et b $[m^3/kg]$ le covolume (volume occupé par les molécules).

On suppose connus les coefficients a et b, la chaleur massique à pression constante C_p, la masse molaire M du gaz et les variables d'état.

Chaleur massique à volume constant : $C_v = C_p - r\dfrac{P + a\rho^2}{P - a\rho^2(1 - 2b\rho)}$

Constante "équivalente" du gaz : $\hat{\gamma} = 1 + \dfrac{r}{C_v} = \dfrac{C_p}{C_v}\left(\dfrac{P - a\rho^2(1 - 2b\rho)}{P + a\rho^2}\right) + \dfrac{2a\rho^2(1 - b\rho)}{P + a\rho^2}$

Vitesse du son : $c^2 = \hat{\gamma}\dfrac{P + a\rho^2}{\rho(1 - b\rho)} - 2a\rho$

Entropie : $s = (P + a\rho^2)\left(\dfrac{1 - b\rho}{\rho}\right)^{\hat{\gamma}}$ et $\beta = \left(\dfrac{\partial P}{\partial s}\right)_\rho = \left(\dfrac{\rho}{1 - b\rho}\right)^{\hat{\gamma}}$

Pression : $P = (\hat{\gamma} - 1)\dfrac{\rho}{(1 - b\rho)}(\varepsilon + a\rho) - a\rho^2$

Energie interne : $\varepsilon = C_v T - a\rho$ avec $\varepsilon_{sup} = -a\rho$

Enthalpie : $h = \dfrac{\hat{\gamma} - b\rho}{\hat{\gamma} - 1}\dfrac{P + a\rho^2}{\rho} - 2a\rho$

Algorithme de base

On suppose connues toutes les variables au temps t^n et on cherche à les déterminer à l'instant t^{n+1}. On résout en deux blocs principaux : d'une part le système masse-quantité de mouvement, de l'autre

l'équation portant sur l'énergie et les scalaires transportés. Dans le premier bloc, on distingue le traitement du système (couplé) acoustique et le traitement de l'équation de la quantité de mouvement complète.

Au début du pas de temps, on commence par mettre à jour les propriétés physiques variables (par exemple $\mu(T)$, $\kappa(T)$, $C_p(Y_1, \ldots, Y_N)$ ou $\lambda(T)$), puis on résout les étapes suivantes :

1. **Acoustique : sous-programme `cfmsvl`**
   Résolution d'une équation de convection-diffusion portant sur ρ^{n+1}.
   On obtient à la fin de l'étape ρ^{n+1}, Q_{ac}^{n+1} et éventuellement une prédiction de la pression $P^{pred}(\rho^{n+1}, e^n)$.

2. **Quantité de mouvement : sous-programme `cfqdmv`**
   Résolution d'une équation de convection-diffusion portant sur u^{n+1} qui fait intervenir Q_{ac}^{n+1} et P^{pred}.
   On obtient à la fin de l'étape u^{n+1}.

3. **Énergie totale : sous-programme `cfener`**
   Résolution d'une équation de convection-diffusion portant sur e^{n+1} qui fait intervenir Q_{ac}^{n+1}, P^{pred} et u^{n+1}.
   On obtient à la fin de l'étape e^{n+1} et une valeur actualisée de la pression $P(\rho^{n+1}, e^{n+1})$.

4. **Scalaires passifs**
   Résolution d'une équation de convection-diffusion standard par scalaire, avec Q_{ac}^{n+1} pour flux convectif.

Discrétisation

On se reportera aux sections relatives aux sous-programmes `cfmsvl` (masse volumique), `cfqdmv` (quantité de mouvement) et `cfener` (énergie). La documentation du sous-programme `cfxtcl` fournit des éléments relatifs aux conditions aux limites.

Mise en œuvre

Le module compressible est une "physique particulière" activée lorsque le mot-clé `IPPMOD(ICOMPF)` est positif ou nul.

Dans ce qui suit, on précise les inconnues et les propriétés principales utilisées dans le module. On fournit également un arbre d'appel simplifié des sous-programmes du module : initialisation avec `initi1` puis (`iniva0` et) `inivar` et enfin, boucle en temps avec `tridim`.

Inconnues et propriétés

Les `NSCAPP` inconnues scalaires associées à la physique particulière sont définies dans `cfvarp` dans l'ordre suivant :

- la masse volumique `RTP(*,ISCA(IRHO))`,
- l'énergie totale `RTP(*,ISCA(IENERG))`,
- la température `RTP(*,ISCA(ITEMPK))`

On souligne que la température est définie en tant que variable "RTP" et non pas en tant que propriété physique "PROPCE". Ce choix a été motivé par la volonté de simplifier la gestion des conditions aux limites, au prix cependant d'un encombrement mémoire légèrement supérieur (une grandeur RTP consomme plus qu'une grandeur PROPCE).

La pression et la vitesse sont classiquement associées aux tableaux suivants :

- pression : RTP(*,IPR)
- vitesse : RTP(*,IU), RTP(*,IV), RTP(*,IW).

Outre les propriétés associées en standard aux variables identifiées ci-dessus, le tableau PROPCE contient également :

- la chaleur massique à volume constant C_v, stockée dans PROPCE(*,IPPROC(ICV)), si l'utilisateur a indiqué dans uscfth qu'elle était variable.
- la viscosité en volume PROPCE(*,IPPROC(IVISCV)) si l'utilisateur a indiqué dans uscfx2 qu'elle était variable.

Pour la gestion des conditions aux limites et en particulier pour le calcul du flux convectif par le schéma de Rusanov aux entrées et sorties (hormis en sortie supersonique), on dispose des tableaux suivants dans PROPFB :

- flux convectif de quantité de mouvement au bord pour les trois composantes dans les tableaux PROPFB(*,IPPROB(IFBRHU)) (composante x), PROPFB(*,IPPROB(IFBRHV)) (composante y) et PROPFB(*,IPPROB(IFBRHW)) (composante z)
- flux convectif d'énergie au bord PROPFB(*,IPPROB(IFBENE))

et on dispose également dans IA :

- d'un tableau d'entiers dont la première "case" est IA(IIFBRU), dimensionné au nombre de faces de bord et permettant de repérer les faces de bord pour lesquelles on calcule le flux convectif par le schéma de Rusanov,
- d'un tableau d'entiers dont la première "case" est IA(IIFBET), dimensionné au nombre de faces de bord et permettant de repérer les faces de paroi à température ou à flux thermique imposé.

Arbre d'appel simplifié

`usini1`				Initialisation des mots-clés utilisateur généraux et positionnement des variables
	`usppmo`			Définition du module "physique particulière" employé
	`varpos`			Positionnement des variables
		`pplecd`		Branchement des physiques particulières pour la lecture du fichier de données éventuel
		`ppvarp`		Branchement des physiques particulières pour le positionnement des inconnues
			`cfvarp`	Positionnement des inconnues spécifiques au module compressible
				`uscfth` Appelé avec ICCFTH=-1, pour indiquer que C_p et C_v sont constants ou variables
				`uscfx2` Conductivité thermique moléculaire constante ou variable et viscosité en volume constante ou variable (ainsi que leur valeur, si elles sont constantes)
		`ppprop`		Branchement des physiques particulières pour le positionnement des propriétés
			`cfprop`	Positionnement des propriétés spécifiques au module compressible
`ppini1`				Branchement des physiques particulières pour l'initialisation des mots-clés spécifiques
	`cfini1`			Initialisation des mots-clés spécifiques au module compressible
	`uscfi1`			Initialisation des mots-clés utilisateur spécifiques au module compressible

Table A.1: Sous-programme `initi1` : initialisation des mots-clés et positionnement des variables

`ppiniv`		Branchement des physiques particulières pour l'initialisation des variables
	`cfiniv`	Initialisation des variables spécifiques au module compressible
	`memcfv`	Réservation de tableaux de travail locaux
	`uscfth`	Initialisation des variables par défaut (en calcul suite : seulement C_v ; si le calcul n'est pas une suite : C_v, la masse volumique et l'énergie)
	`uscfxi`	Initialisation des variables par l'utilisateur (seulement si le calcul n'est pas une suite)

Table A.2: Sous-programme `inivar` : initialisation des variables

`phyvar`		Calcul des propriétés physiques variables
	`ppphyv`	Branchement des physiques particulières pour le calcul des propriétés physiques variables
	`cfphyv`	Calcul des propriétés physiques variables pour le module compressible
	`usphyv`	Calcul par l'utilisateur des propriétés physiques variables pour le module compressible (C_v est calculé dans `uscfth` qui est appelé par `usphyv`)

Table A.3: Sous-programme `tridim` : partie 1 (propriétés physiques)

`dttvar`			Calcul du pas de temps variable
	`cfdttv`		Calcul de la contrainte liée au CFL en compressible
		`memcft`	Gestion de la mémoire pour le calcul de la contrainte en CFL
		`cfmsfl`	Calcul du flux associé à la contrainte en CFL
`precli`			Initialisation des tableaux avant calcul des conditions aux limites (`IITYPF`, `ICODCL`, `RCODCL`)
	`ppprcl`		Initialisations spécifiques aux différentes physiques particulières avant calcul des conditions aux limites (pour le module compressible : `IZFPPP`, `IA(IIFBRU)`, `IA(IIFBET)`, `RCODCL`, flux convectifs pour la quantité de mouvement et l'énergie)
`ppclim`			Branchement des physiques particulières pour les conditions aux limites (en lieu et place de `usclim`)
	`uscfcl`		Intervention de l'utilisateur pour les conditions aux limites (en lieu et place de `usclim`, même pour les variables qui ne sont pas spécifiques au module compressible)
`condli`			Traitement des conditions aux limites
	`pptycl`		Branchement des physiques particulières pour le traitement des conditions aux limites
		`cfxtcl`	Traitement des conditions aux limites pour le compressible
		`uscfth`	Calculs de thermodynamique pour le calcul des conditions aux limites
		`cfrusb`	Flux de Rusanov (entrées ou sorties sauf sortie supersonique)

Table A.4: Sous-programme `tridim` : partie 2 (pas de temps variable et conditions aux limites)

`memcfm`			Gestion de la mémoire pour la résolution de l'étape "acoustique"
`cfmsvl`			Résolution de l'étape "acoustique"
	`cfmsfl`		Calcul du "flux de masse" aux faces (noté $\rho\,\underline{w}\cdot\underline{n}\,S$ dans la documentation du sous-programme `cfmsvl`)
		`cfdivs`	Calcul du terme en divergence du tenseur des contraintes visqueuses (trois appels), éventuellement
			Après `cfmsfl`, on impose le flux de masse aux faces de bord à partir des conditions aux limites
	`cfmsvs`		Calcul de la "viscosité" aux faces (notée $\Delta t\,c^2\frac{S}{d}$ dans la documentation du sous-programme `cfmsvl`)
			Après `cfmsvs`, on annule la viscosité aux faces de bord pour que le flux de masse soit bien celui souhaité
	`codits`		Résolution du système portant sur la masse volumique
	`clpsca`		Impression des bornes et clipping éventuel (pas de clipping en standard)
	`uscfth`		Gestion éventuelle des bornes par l'utilisateur
	`cfbsc3`		Calcul du flux de masse acoustique aux faces (noté $\underline{\underline{Q}}_{ac}\cdot\underline{n}$ dans la documentation du sous-programme `cfmsvl`)
	`uscfth`		Actualisation de la pression, éventuellement
`cfqdmv`			Résolution de la quantité de mouvement
	`cfcdts`		Résolution du système
		`cfbsc2`	Calcul des termes de convection et de diffusion au second membre

Table A.5: Sous-programme `tridim` : partie 3 (Navier-Stokes)

`scalai`			Résolution des équations sur les scalaires
	`cfener`		Résolution de l'équation sur l'énergie totale
		`memcfe`	Gestion de la mémoire locale
		`cfdivs`	Calcul du terme en divergence du produit "tenseur des contraintes par vitesse"
		`uscfth`	Calcul de l'écart "énergie interne - $C_v\,T$" (ε_{sup})
		`cfcdts`	Résolution du système
			`cfbsc2` Calcul des termes de convection et de diffusion au second membre
		`clpsca`	Impression des bornes et clipping éventuel (pas de clipping en standard)
		`uscfth`	Gestion éventuelle des bornes par l'utilisateur
		`uscfth`	Mise à jour de la pression

Table A.6: Sous-programme `tridim` : partie 4 (scalaires)

Le sous-programme `cfbsc3` est similaire à `bilsc2`, mais il produit des flux aux faces et n'est écrit que pour un schéma upwind, à l'ordre 1 en temps (ce qui est cohérent avec les choix faits dans l'algorithme compressible).

Le sous-programme `cfbsc2` est similaire à `bilsc2`, mais n'est écrit que pour un schéma d'ordre 1 en temps. Le sous-programme `cfbsc2` permet d'effectuer un traitement spécifique aux faces de bord pour lesquelles on a appliqué un schéma de Rusanov pour calculer le flux convectif total. Ce sous-programme est appelé pour la résolution de l'équation de la quantité de mouvement et de l'équation de l'énergie. On pourra se reporter à la documentation du sous-programme `cfxtcl`.

Le sous-programme `cfcdts` est similaire à `codits` mais fait appel à `cfbsc2` et non pas à `bilsc2`. Il diffère de `codits` par quelques autres détails qui ne sont pas gênants dans l'immédiat : initialisation de PVARA et de SMBINI, ordre en temps (ordre 2 non pris en compte).

Points à traiter

Des actions complémentaires sont identifiées ci-après, dans l'ordre d'urgence décroissante (on se reportera également à la section "Points à traiter" de la documentation des autres sous-programmes du module compressible).

- Assurer la cohérence des sous-programmes suivants (ou, éventuellement, les fusionner pour éviter qu'ils ne divergent) :
  - `cfcdts` et `codits`,
  - `cfbsc2` et `bilsc2`,
  - `cfbsc3` et `bilsc2`.
- Permettre les suites de calcul incompressible/compressible et compressible/incompressible.
- Apporter un complément de validation (exemple : IPHYDR).
- Assurer la compatibilité avec certaines physiques particulières, selon les besoins. Par exemple : arc électrique, rayonnement, combustion.
- Identifier les causes des difficultés rencontrées sur certains cas académiques, en particulier :
  - canal subsonique (comment s'affranchir des effets indésirables associés aux conditions d'entrée et de sortie, comment réaliser un calcul périodique, en particulier pour la température dont le gradient dans la direction de l'écoulement n'est pas nul, si les parois sont adiabatiques),
  - cavité fermée sans vitesse ni effets de gravité, avec température ou flux thermique imposé en paroi (il pourrait être utile d'extrapoler le gradient de pression au bord : la pression dépend de la température et une simple condition de Neumann homogène est susceptible de créer un terme source de quantité de mouvement parasite),
  - maillage non conforme (non conformité dans la direction transverse d'un canal),
  - "tube à choc" avec terme source d'énergie.
- Compléter certains points de documentation, en particulier les conditions aux limites thermiques pour le couplage avec SYRTHES.
- Améliorer la rapidité à faible nombre de Mach (est-il possible de lever la limite actuelle sur la valeur du pas de temps ?).
- Enrichir, au besoin :
  - les thermodynamiques prises en compte (multiconstituant, gamma variable, Van der Waals...),
  - la gamme des conditions aux limites d'entrée disponibles (condition à débit massique et débit enthalpique imposés par exemple).
- Tester des variantes de l'algorithme :
  - prise en compte des termes sources de l'équation de la quantité de mouvement autres que la gravité dans l'équation de la masse résolue lors de l'étape "acoustique" (les tests réalisés avec cette variante de l'algorithme devront être repris dans la mesure où, dans `cfmsfl`, IIROM et IIROMB n'étaient pas initialisés),
  - implicitation du terme de convection dans l'équation de la masse (éliminer cette possibilité si elle n'apporte rien),
  - étape de prédiction de la pression,

  - non reconstruction de la masse volumique pour le terme convectif (actuellement, les termes convectifs sont traités avec décentrement amont, d'ordre 1 en espace ; pour l'équation de la quantité de mouvement et l'équation de l'énergie, on utilise les valeurs prises au centre des cellules sans reconstruction : c'est l'approche standard de code_saturne, traduite dans `cfbsc2` ; par contre, dans `cfmsvl`, on reconstruit les valeurs de la masse volumique utilisées pour le terme convectif ; il n'y a pas de raison d'adopter des stratégies différentes, d'autant plus que la reconstruction de la masse volumique ne permet pas de monter en ordre et augmente le risque de dépassement des bornes physiques),
  - montée en ordre en espace (en vérifier l'utilité et la robustesse, en particulier relativement au principe du maximum pour la masse volumique),
  - montée en ordre en temps (en vérifier l'utilité et la robustesse).

- Optimiser l'encombrement mémoire.

B- cfener routine

Fonction

Pour les notations et l'algorithme dans son ensemble, on se reportera à `cfbase`.

Après masse (acoustique) et quantité de mouvement, on considère un dernier pas fractionnaire (de t^{**} à t^{***}) au cours duquel seule varie l'énergie totale $E = \rho e$.

$$\begin{cases} \rho^{***} = \rho^{**} = \rho^{n+1} \\ \underline{Q}^{***} = \underline{Q}^{**} = \underline{Q}^{n+1} \\ \dfrac{\partial \rho e}{\partial t} + \text{div}\left(\underline{Q}_{ac}\left(e + \dfrac{P}{\rho}\right)\right) = \rho \underline{f}_v \cdot \underline{u} + \text{div}\left(\underline{\underline{\Sigma}}^v \underline{u}\right) - \text{div}\ \underline{\Phi}_s + \rho \Phi_v \end{cases} \tag{V.B.1}$$

Pour conserver la positivité de l'énergie, il est indispensable ici, comme pour les scalaires, d'utiliser le flux de masse convectif acoustique $\underline{Q}_{ac}^{n+1}$ compatible avec l'équation de la masse. De plus, pour obtenir des propriétés de positivité sur les scalaires, un schéma upwind pour le terme convectif doit être utilisé (mais les termes sources introduisent des contraintes supplémentaires qui peuvent être prépondérantes et gênantes).

À la fin de cette étape, on actualise éventuellement (mais par défaut non) une deuxième et dernière fois la pression en utilisant la loi d'état pour obtenir la pression finale :

$$P^{n+1} = P(\rho^{n+1}, \varepsilon^{n+1}) \tag{V.B.2}$$

See the programmers reference of the dedicated subroutine for further details.

Discrétisation

Discrétisation en temps

La modélisation des flux de chaleur choisie jusqu'à présent est de la forme $-\text{div}\,(\underline{\Phi}_s) = \text{div}\,(\lambda \underline{\nabla} T)$.

Pour faire apparaître un terme diffusif stabilisant dans la matrice de résolution, on cherche à exprimer le flux diffusif de chaleur ($-\text{div}\,(\underline{\Phi}_s)$) en fonction de la variable résolue (l'énergie totale).

Avec $\varepsilon_{sup}(P, \rho)$ dépendant de la loi d'état, on exprime l'énergie totale de la façon suivante :

$$e = \varepsilon + \frac{1}{2}u^2 = (C_v T + \varepsilon_{sup}) + \frac{1}{2}u^2 \tag{V.B.3}$$

En supposant C_v constant[1], on a alors :

$$-\text{div}\,(\underline{\Phi}_s) = \text{div}\,(K \underline{\nabla}(e - \frac{1}{2}u^2 - \varepsilon_{sup})) \qquad \text{avec } K = \lambda / C_v \tag{V.B.4}$$

[1] Pour C_v non constant, les développements restent à faire : on pourra se reporter à P. Mathon, F. Archambeau, J.-M. Hérard : "Implantation d'un algorithme compressible dans code_saturne ", HI-83/03/016/A

Lorsqu'un modèle de turbulence est activé, on conserve la même forme de modélisation pour les flux thermiques et K intègre alors la diffusivité turbulente. On pourra se reporter à la documentation de `cfxtcl` à ce sujet.

Avec la formulation (V.B.4), on peut donc impliciter le terme en $\underline{\nabla} e$.

De plus, puisque la vitesse a déjà été résolue, on implicite également le terme en $\underline{\nabla}\frac{1}{2}u^2$. L'exposant $n+\frac{1}{2}$ de ε_{sup} indique que l'implicitation de ce terme est partielle (elle dépend de la forme de la loi d'état).

Par ailleurs, on implicite le terme de convection, le terme de puissance des forces volumiques, éventuellement le terme de puissance des forces de pression (suivant la valeur de `IGRDPP`, on utilise la prédiction de pression obtenue après résolution de l'équation portant sur la masse volumique ou bien la pression du pas de temps précédent) et le terme de puissance des forces visqueuses. On implicite le terme de puissance volumique en utilisant ρ^{n+1}.

On obtient alors l'équation discrète portant sur e :

$$\begin{aligned}\frac{(\rho e)^{n+1}-(\rho e)^n}{\Delta t^n} + \mathrm{div}\,(\underline{Q}_{ac}^{n+1}e^{n+1}) - \mathrm{div}\,(K^n\underline{\nabla}e^{n+1}) = \rho^{n+1}\underline{f}_v\cdot\underline{u}^{n+1} - \mathrm{div}\,(\underline{Q}_{ac}^{n+1}\frac{\widetilde{P}}{\rho^{n+1}}) \\ + \mathrm{div}\,((\underline{\underline{\Sigma}}^v)^{n+1}\underline{u}^{n+1}) - \mathrm{div}\,(K^n\underline{\nabla}(\tfrac{1}{2}(u^2)^{n+1}+\varepsilon_{sup}^{n+\frac{1}{2}})) + \rho^{n+1}\Phi_v\end{aligned} \tag{V.B.5}$$

avec $\widetilde{P}=P^{Pred}$ ou P^n suivant la valeur de `IGRDPP` (P^n par défaut).

En pratique, dans code_saturne, on résout cette équation en faisant apparaître à gauche l'écart $e^{n+1}-e^n$. Pour cela, on écrit la dérivée en temps discrète sous la forme suivante :

$$\begin{aligned}\frac{(\rho e)^{n+1}-(\rho e)^n}{\Delta t^n} &= \frac{\rho^{n+1}\,e^{n+1}-\rho^n\,e^n}{\Delta t^n} \\ &= \frac{\rho^n\,e^{n+1}-\rho^n\,e^n}{\Delta t^n} + \frac{\rho^{n+1}\,e^{n+1}-\rho^n\,e^{n+1}}{\Delta t^n} \\ &= \frac{\rho^n}{\Delta t^n}\left(e^{n+1}-e^n\right) + e^{n+1}\frac{\rho^{n+1}-\rho^n}{\Delta t^n}\end{aligned} \tag{V.B.6}$$

et l'on utilise l'équation de la masse discrète pour écrire :

$$\frac{(\rho e)^{n+1}-(\rho e)^n}{\Delta t^n} = \frac{\rho^n}{\Delta t^n}\left(e^{n+1}-e^n\right) - e^{n+1}\mathrm{div}\,\underline{Q}_{ac}^{n+1} \tag{V.B.7}$$

Discrétisation en espace

Introduction

On intègre l'équation (V.B.5) sur la cellule i de volume Ω_i et l'on procède comme pour l'équation de la masse et de la quantité de mouvement.

On obtient alors l'équation discrète suivante :

$$\begin{aligned}&\frac{\Omega_i}{\Delta t^n}(\rho_i^{n+1}e_i^{n+1}-\rho_i^n e_i^n) + \sum_{j\in V(i)}\left(e^{n+1}\underline{Q}_{ac}^{n+1}\right)_{ij}\cdot\underline{S}_{ij} - \sum_{j\in V(i)}\left(K^n\underline{\nabla}(e^{n+1})\right)_{ij}\cdot\underline{S}_{ij} \\ &= \Omega_i\rho_i^{n+1}\underline{f}_{vi}\cdot\underline{u}_i^{n+1} - \sum_{j\in V(i)}\left(\frac{P^{Pred}}{\rho^{n+1}}\,\underline{Q}_{ac}^{n+1}\right)_{ij}\cdot\underline{S}_{ij} + \sum_{j\in V(i)}\left((\underline{\underline{\Sigma}}^v)^{n+1}\underline{u}^{n+1}\right)_{ij}\cdot\underline{S}_{ij} \\ &- \sum_{j\in V(i)}\left(K^n\underline{\nabla}\left(\frac{1}{2}(u^2)^{n+1}+\varepsilon_{sup}^{n+\frac{1}{2}}\right)\right)_{ij}\cdot\underline{S}_{ij} + \Omega_i\rho_i^{n+1}\Phi_{vi}\end{aligned} \tag{V.B.8}$$

Discrétisation de la partie “convective”

La valeur à la face s'écrit :

$$\left(e^{n+1}\underline{Q}_{ac}^{n+1}\right)_{ij}\cdot\underline{S}_{ij}=e_{ij}^{n+1}(\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij} \tag{V.B.9}$$

avec un décentrement sur la valeur de e^{n+1} aux faces :

$$\begin{array}{rcll} e_{ij}^{n+1} & = & e_i^{n+1} & \text{si} \quad (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}\geqslant 0 \\ & = & e_j^{n+1} & \text{si} \quad (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}<0 \end{array} \tag{V.B.10}$$

que l'on peut noter :

$$e_{ij}^{n+1}=\beta_{ij}e_i^{n+1}+(1-\beta_{ij})e_j^{n+1} \tag{V.B.11}$$

avec

$$\left\{\begin{array}{lll} \beta_{ij}=1 & \text{si} & (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}\geqslant 0 \\ \beta_{ij}=0 & \text{si} & (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}<0 \end{array}\right. \tag{V.B.12}$$

Discrétisation de la partie “diffusive”

La valeur à la face s'écrit :

$$\begin{array}{c} \left(K^n\underline{\nabla}(e^{n+1})\right)_{ij}\cdot\underline{S}_{ij}=K_{ij}^n\left(\dfrac{\partial e}{\partial n}\right)_{ij}^{n+1}S_{ij} \\ \text{et} \\ \left(K^n\underline{\nabla}\left(\frac{1}{2}(u^2)^{n+1}+\varepsilon_{sup}^{n+\frac{1}{2}}\right)\right)_{ij}\cdot\underline{S}_{ij}=K_{ij}^n\left(\dfrac{\partial\left(\frac{1}{2}u^2+\varepsilon_{sup}\right)}{\partial n}\right)_{ij}^{n+\frac{1}{2}}S_{ij} \end{array} \tag{V.B.13}$$

avec une interpolation linéaire pour K^n aux faces (et en pratique, $\alpha_{ij}=\frac{1}{2}$) :

$$K_{ij}^n=\alpha_{ij}K_i^n+(1-\alpha_{ij})K_j^n \tag{V.B.14}$$

et un schéma centré avec reconstruction pour le gradient normal aux faces :

$$\left(\frac{\partial e}{\partial n}\right)_{ij}^{n+1}=\frac{e_{J'}^{n+1}-e_{I'}^{n+1}}{\overline{I'J'}} \quad \text{et} \quad \left(\frac{\partial\left(\frac{1}{2}u^2+\varepsilon_{sup}\right)}{\partial n}\right)_{ij}^{n+\frac{1}{2}}=\frac{(\frac{1}{2}u^2+\varepsilon_{sup})_{J'}^{n+\frac{1}{2}}-(\frac{1}{2}u^2+\varepsilon_{sup})_{I'}^{n+\frac{1}{2}}}{\overline{I'J'}} \tag{V.B.15}$$

Discrétisation de la puissance des forces de pression

Ce terme est issu du terme convectif, on le discrétise donc de la même façon.

$$\left(\frac{\widetilde{P}}{\rho^{n+1}}\,\underline{Q}_{ac}^{n+1}\right)_{ij}\cdot\underline{S}_{ij}=\left(\frac{\widetilde{P}}{\rho^{n+1}}\right)_{ij}(\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij} \tag{V.B.16}$$

avec un décentrement sur la valeur de $\dfrac{P}{\rho}$ aux faces :

$$\left(\frac{\widetilde{P}}{\rho^{n+1}}\right)_{ij}=\beta_{ij}\frac{\widetilde{P}_i}{\rho_i^{n+1}}+(1-\beta_{ij})\frac{\widetilde{P}_j}{\rho_j^{n+1}} \quad \text{avec} \quad \left\{\begin{array}{lll} \beta_{ij}=1 & \text{si} & (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}\geqslant 0 \\ \beta_{ij}=0 & \text{si} & (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}<0 \end{array}\right. \tag{V.B.17}$$

Discrétisation de la puissance des forces visqueuses

On calcule les termes dans les cellules puis on utilise une interpolation linéaire (on utilise $\alpha_{ij} = \frac{1}{2}$ dans la relation ci-dessous) :

$$\left(\left(\underline{\underline{\Sigma}}^{v}\right)^{n+1}\underline{u}^{n+1}\right)_{ij} \cdot \underline{S}_{ij} = \left\{\alpha_{ij}\left(\left(\underline{\underline{\Sigma}}^{v}\right)^{n+1}\underline{u}^{n+1}\right)_{i} + (1-\alpha_{ij})\left(\left(\underline{\underline{\Sigma}}^{v}\right)^{n+1}\underline{u}^{n+1}\right)_{j}\right\} \cdot \underline{S}_{ij} \tag{V.B.18}$$

Remarques

Les termes "convectifs" associés à $\text{div}\left(\left(e^{n+1} + \dfrac{\widetilde{P}}{\rho^{n+1}}\right)\underline{Q}_{ac}^{n+1}\right)$ sont calculés avec un décentrement amont (consistant, d'ordre 1 en espace). Les valeurs utilisées sont bien prises au centre de la cellule amont (e_i, P_i, ρ_i) et non pas au projeté I' du centre de la cellule sur la normale à la face passant par son centre de gravité (sur un cas test en triangles, l'utilisation de P'_I et de ρ'_I pour le terme de transport de pression a conduit à un résultat insatisfaisant, mais des corrections ont été apportées aux sources depuis et il serait utile de vérifier que cette conclusion n'est pas remise en question).

Les termes diffusifs associés à $\text{div}\left(K\,\underline{\nabla}\left(e + \dfrac{1}{2}u^2 + \varepsilon_{sup}\right)\right)$ sont calculés en utilisant des valeurs aux faces reconstruites pour s'assurer de la consistance du schéma.

Mise en œuvre

Après une étape de gestion de la mémoire (`memcfe`), on calcule les différents termes sources (au centre des cellules) :

- source volumique de chaleur (`ustssc`),
- source associée aux sources de masse (`catsma`),
- source associée à l'accumulation de masse $\text{div}\,\underline{Q}_{ac}$ (directement dans `cfener`),
- dissipation visqueuse (`cfdivs`),
- transport de pression (directement dans `cfener`),
- puissance de la pesanteur (directement dans `cfener`),
- termes diffusifs en $\text{div}\left(K\,\underline{\nabla}\left(\dfrac{1}{2}u^2 + \varepsilon_{sup}\right)\right)$ (calcul de ε_{sup} par `uscfth`, puis calcul du terme diffusif directement dans `cfener`).

Le système (V.B.8) est résolu par une méthode d'incrément et résidu en utilisant une méthode de Jacobi (`cfcdts`).

L'impression des bornes et la limitation éventuelle de l'énergie sont ensuite effectuées par `clpsca` suivi de `uscfth` (intervention utilisateur optionnelle).

On actualise enfin la pression et on calcule la température (`uscfth`).

Pour terminer, en parallèle ou en périodique, on échange les variables pression, énergie et température.

Points à traiter

• **Choix de $\widetilde{P}$**
En standard, on utilise $\widetilde{P} = P^n$, mais ce n'est pas le seul choix possible. On pourrait étudier le comportement de l'algorithme avec P^{Pred} et P^{n+1} (avec P^{n+1}, en particulier, $\dfrac{\widetilde{P}}{\rho^{n+1}}$ est évalué avec la masse volumique et l'énergie prises au même instant).

• **Terme source dans l'équation de l'énergie**
La présence d'un terme source externe dans l'équation de l'énergie génère des oscillations de vitesse qu'il est important d'analyser et de comprendre.

C- cfmsvl routine

Fonction

Pour les notations et l'algorithme dans son ensemble, on se reportera à `cfbase`.

On considère un premier pas fractionnaire au cours duquel l'énergie totale est fixe. Seules varient la masse volumique et le flux de masse acoustique normal aux faces (défini et calculé aux faces).

On a donc le système suivant, entre t^n et t^* :

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \mathrm{div}\, \underline{Q}_{ac} = 0 \\ \dfrac{\partial \underline{Q}_{ac}}{\partial t} + \underline{\nabla} P = \rho \underline{f} \\ \underline{Q}^* = \underline{Q}^n \\ e^* = e^n \end{cases} \tag{V.C.1}$$

Une partie des termes sources de l'équation de la quantité de mouvement peut être prise en compte dans cette étape (les termes les plus importants, en prêtant attention aux sous-équilibres).

Il faut noter que si $\underline{f}$ est effectivement nul, on aura bien un système "acoustique", mais que si l'on place des termes supplémentaires dans $\underline{f}$, la dénomination est abusive (on la conservera cependant).

On obtient $\rho^* = \rho^{n+1}$ en résolvant (V.C.1), et l'on actualise alors le flux de masse acoustique $\underline{Q}_{ac}^{n+1}$, qui servira pour la convection (en particulier pour la convection de l'enthalpie totale et de tous les scalaires transportés).

Suivant la valeur de `IGRDPP`, on actualise éventuellement la pression, en utilisant la loi d'état :

$$P^{Pred} = P(\rho^{n+1}, \varepsilon^n)$$

Discrétisation

Discrétisation en temps

Le système (V.C.1) discrétisé en temps donne :

$$\begin{cases} \dfrac{\rho^{n+1} - \rho^n}{\Delta t^n} + \mathrm{div}\, \underline{Q}_{ac}^{n+1} = 0 \\ \dfrac{\underline{Q}_{ac}^{n+1} - \underline{Q}^n}{\Delta t^n} + \underline{\nabla} P^* = \rho^n \underline{f}^n \\ Q^* = Q^n \\ e^* = e^n \end{cases} \tag{V.C.2}$$

$$\begin{array}{lrcl} \text{avec} & \underline{f}^n & = & \underline{0} \\ \text{ou} & \underline{f}^n & = & \underline{g} \\ \text{ou même} & \underline{f}^n & = & \underline{f}_v + \dfrac{1}{\rho^n}\left(-\mathrm{div}\,(\underline{u} \otimes \underline{Q}) + \underline{\mathrm{div}}\,(\underline{\underline{\Sigma}}^v) + \underline{j} \wedge \underline{B}\right)^n \end{array} \tag{V.C.3}$$

Dans la pratique nous avons décidé de prendre $\underline{f}^n = \underline{g}$:

- le terme $\underline{j} \wedge \underline{B}$ n'a pas été testé,
- le terme $\underline{\text{div}}\,(\underline{\underline{\Sigma}}^v)$ était négligeable sur les tests réalisés,
- le terme $\text{div}\,(\underline{u} \otimes \underline{Q})$ a paru déstabiliser les calculs (mais au moins une partie des tests a été réalisée avec une erreur de programmation et il faudrait donc les reprendre).

Le terme $\underline{Q}^n$ dans la 2ème équation de (V.C.2) est le vecteur "quantité de mouvement" qui provient de l'étape de résolution de la quantité de mouvement du pas de temps précédent, $\underline{Q}^n = \rho^n \underline{u}^n$. On pourrait théoriquement utiliser un vecteur quantité de mouvement issu de l'étape acoustique du pas de temps précédent, mais il ne constitue qu'un "prédicteur" plus ou moins satisfaisant (il n'a pas "vu" les termes sources qui ne sont pas dans $\underline{f}^n$) et cette solution n'a pas été testée.

On écrit alors la pression sous la forme :

$$\underline{\nabla} P = c^2\, \underline{\nabla}\rho + \beta\, \underline{\nabla} s \tag{V.C.4}$$

avec $c^2 = \left.\dfrac{\partial P}{\partial \rho}\right|_s$ et $\beta = \left.\dfrac{\partial P}{\partial s}\right|_\rho$ tabulés ou analytiques à partir de la loi d'état.

On discrétise l'expression précédente en :

$$\underline{\nabla} P^* = (c^2)^n \underline{\nabla}(\rho^{n+1}) + \beta^n \underline{\nabla}(s^n) \tag{V.C.5}$$

On obtient alors une équation portant sur ρ^{n+1} en substituant l'expression de $\underline{Q}_{ac}^{n+1}$ issue de la 2ème équation de (V.C.2) dans la 1ère équation de (V.C.2) :

$$\frac{\rho^{n+1} - \rho^n}{\Delta t^n} + \text{div}\,(\underline{w}^n \rho^n) - \text{div}\,\left(\Delta t^n (c^2)^n \underline{\nabla}(\rho^{n+1})\right) = 0 \tag{V.C.6}$$

où :

$$\underline{w}^n \quad = \quad \underline{u}^n + \Delta t^n \left(\underline{f}^n - \frac{\beta^n}{\rho^n} \underline{\nabla}(s^n) \right) \tag{V.C.7}$$

Formulation alternative (programmée mais non testée) avec le terme de convection implicite :

$$\frac{\rho^{n+1} - \rho^n}{\Delta t^n} + \text{div}\,(\underline{w}^n \rho^{n+1}) - \text{div}\,\left(\Delta t^n (c^2)^n \underline{\nabla}(\rho^{n+1})\right) = 0 \tag{V.C.8}$$

Discrétisation en espace

Introduction

On intègre l'équation précédente ((V.C.6) ou (V.C.8)) sur la cellule i de volume Ω_i. On transforme les intégrales de volume en intégrales surfaciques et l'on discrétise ces intégrales. Pour simplifier l'exposé, on se place sur une cellule i dont aucune face n'est sur le bord du domaine.

On obtient alors l'équation discrète suivante[1] :

$$\Omega_i \frac{\rho_i^{n+1} - \rho_i^n}{\Delta t^n} + \sum_{j \in Vois(i)} (\rho^{n+\frac{1}{2}} \underline{w}^n)_{ij} \cdot \underline{S}_{ij} - \sum_{j \in Vois(i)} \left(\Delta t^n (c^2)^n \underline{\nabla}(\rho^{n+1})\right)_{ij} \cdot \underline{S}_{ij} = 0 \tag{V.C.9}$$

[1] L'exposant $^{n+\frac{1}{2}}$ signifie que le terme peut être implicite ou explicite. En pratique on a choisi $\rho^{n+\frac{1}{2}} = \rho^n$.

Discrétisation de la partie “convective”

La valeur à la face s'écrit :

$$(\rho^{n+\frac{1}{2}}\underline{w}^n)_{ij}\cdot\underline{S}_{ij}=\rho_{ij}^{n+\frac{1}{2}}\underline{w}_{ij}^n\cdot\underline{S}_{ij} \tag{V.C.10}$$

avec, pour $\underline{w}_{ij}^n$, une simple interpolation linéaire :

$$\underline{w}_{ij}^n=\alpha_{ij}\underline{w}_i^n+(1-\alpha_{ij})\underline{w}_j^n \tag{V.C.11}$$

et un décentrement sur la valeur de $\rho^{n+\frac{1}{2}}$ aux faces :

$$\begin{array}{rcll}\rho_{ij}^{n+\frac{1}{2}} & = & \rho_{I'}^{n+\frac{1}{2}} & \text{si} \quad \underline{w}_{ij}^n\cdot\underline{S}_{ij}\geqslant 0\\ & = & \rho_{J'}^{n+\frac{1}{2}} & \text{si} \quad \underline{w}_{ij}^n\cdot\underline{S}_{ij}<0\end{array} \tag{V.C.12}$$

que l'on peut noter :

$$\rho_{ij}^{n+\frac{1}{2}}=\beta_{ij}\rho_{I'}^{n+\frac{1}{2}}+(1-\beta_{ij})\rho_{J'}^{n+\frac{1}{2}} \tag{V.C.13}$$

avec

$$\left\{\begin{array}{l}\beta_{ij}=1 \quad \text{si} \quad \underline{w}_{ij}^n\cdot\underline{S}_{ij}\geqslant 0\\ \beta_{ij}=0 \quad \text{si} \quad \underline{w}_{ij}^n\cdot\underline{S}_{ij}<0\end{array}\right. \tag{V.C.14}$$

Discrétisation de la partie “diffusive”

La valeur à la face s'écrit :

$$\left(\Delta t^n(c^2)^n\underline{\nabla}(\rho^{n+1})\right)_{ij}\cdot\underline{S}_{ij}=\Delta t^n(c^2)_{ij}^n\left(\frac{\partial\rho}{\partial n}\right)_{ij}^{n+1}S_{ij} \tag{V.C.15}$$

avec, pour assurer la continuité du flux normal à l'interface, une interpolation harmonique de $(c^2)^n$:

$$(c^2)_{ij}^n=\frac{(c^2)_i^n(c^2)_j^n}{\alpha_{ij}(c^2)_i^n+(1-\alpha_{ij})(c^2)_j^n} \tag{V.C.16}$$

et un schéma centré pour le gradient normal aux faces :

$$\left(\frac{\partial\rho}{\partial n}\right)_{ij}^{n+1}=\frac{\rho_{J'}^{n+1}-\rho_{I'}^{n+1}}{\overline{I'J'}} \tag{V.C.17}$$

Système final

On obtient maintenant le système final, portant sur $(\rho_i^{n+1})_{i=1\ldots N}$:

$$\frac{\Omega_i}{\Delta t^n}(\rho_i^{n+1}-\rho_i^n)+\sum_{j\in Vois(i)}\rho_{ij}^{n+\frac{1}{2}}\underline{w}_{ij}^n\cdot\underline{S}_{ij}-\sum_{j\in Vois(i)}\Delta t^n(c^2)_{ij}^n\frac{\rho_{J'}^{n+1}-\rho_{I'}^{n+1}}{\overline{I'J'}}S_{ij}=0 \tag{V.C.18}$$

Remarque : interpolation aux faces pour le terme de diffusion

Le choix de la forme de la moyenne pour le cofacteur du flux normal n'est pas sans conséquence sur la vitesse de convergence, surtout lorsque l'on est en présence de fortes inhomogénéités.

On utilise une interpolation harmonique pour c^2 afin de conserver la continuité du flux diffusif normal $\Delta t(c^2)\dfrac{\partial\rho}{\partial n}$ à l'interface ij. En effet, on suppose que le flux est dérivable à l'interface. Il doit donc y

être continu.
Écrivons la continuité du flux normal à l'interface, avec la discrétisation suivante[2] :

$$\left(\Delta t(c^2)\frac{\partial \rho}{\partial n}\right)_{ij} = \Delta t(c^2)_i \frac{\rho_{ij}-\rho_{I'}}{\overline{I'F}} = \Delta t(c^2)_j \frac{\rho_{J'}-\rho_{ij}}{\overline{FJ'}} \tag{V.C.19}$$

En égalant les flux à gauche et à droite de l'interface, on obtient

$$\rho_{ij} = \frac{\overline{I'F}\,(c^2)_j \rho_{J'} + \overline{FJ'}\,(c^2)_i \rho_{I'}}{\overline{I'F}\,(c^2)_j + \overline{FJ'}\,(c^2)_i} \tag{V.C.20}$$

On introduit cette formulation dans la définition du flux (par exemple, du flux à gauche) :

$$\left(\Delta t(c^2)\frac{\partial \rho}{\partial n}\right)_{ij} = \Delta t(c^2)_i \frac{\rho_{ij}-\rho_{I'}}{\overline{I'F}} \tag{V.C.21}$$

et on utilise la définition de $(c^2)_{ij}$ en fonction de ce même flux

$$\left(\Delta t(c^2)\frac{\partial \rho}{\partial n}\right)_{ij} \overset{\text{déf}}{=} \Delta t(c^2)_{ij} \frac{\rho_{J'}-\rho_{I'}}{\overline{I'J'}} \tag{V.C.22}$$

pour obtenir la valeur de $(c^2)_{ij}$ correspondant à l'équation (V.C.16) :

$$(c^2)_{ij} = \frac{\overline{I'J'}\,(c^2)_i (c^2)_j}{\overline{FJ'}\,(c^2)_i + \overline{I'F}\,(c^2)_j} \tag{V.C.23}$$

Mise en œuvre

Le système (V.C.18) est résolu par une méthode d'incrément et résidu en utilisant une méthode de Jacobi pour inverser le système si le terme convectif est implicite et en utilisant une méthode de gradient conjugué si le terme convectif est explicite (qui est le cas par défaut).

Attention, les valeurs du flux de masse $\rho\,\underline{w}\cdot\underline{S}$ et de la viscosité $\Delta t\, c^2 \frac{S}{d}$ aux faces de bord, qui sont calculées dans `cfmsfl` et `cfmsvs` respectivement, sont modifiées immédiatement après l'appel à ces sous-programmes. En effet, il est indispensable que la contribution de bord de $\left(\rho\,\underline{w} - \Delta t\,(c^2)\,\underline{\nabla}\,\rho\right)\cdot\underline{S}$ représente exactement $\underline{Q}_{ac}\cdot\underline{S}$. Pour cela,

- immédiatement après l'appel à `cfmsfl`, on remplace la contribution de bord de $\rho\,\underline{w}\cdot\underline{S}$ par le flux de masse exact, $\underline{Q}_{ac}\cdot\underline{S}$, déterminé à partir des conditions aux limites,
- puis, immédiatement après l'appel à `cfmsvs`, on annule la viscosité au bord $\Delta t\,(c^2)$ pour éliminer la contribution de $-\Delta t\,(c^2)\,(\underline{\nabla}\,\rho)\cdot\underline{S}$ (l'annulation de la viscosité n'est pas problématique pour la matrice, puisqu'elle porte sur des incréments).

Une fois qu'on a obtenu ρ^{n+1}, on peut actualiser le flux de masse acoustique aux faces $(\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}$, qui servira pour la convection des autres variables :

$$(\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij} = -\left(\Delta t^n (c^2)^n \underline{\nabla}(\rho^{n+1})\right)_{ij}\cdot\underline{S}_{ij} + \left(\rho^{n+\frac{1}{2}}\underline{w}^n\right)_{ij}\cdot\underline{S}_{ij} \tag{V.C.24}$$

Ce calcul de flux est réalisé par `cfbsc3`. Si l'on a choisi l'algorithme standard, équation (V.C.6), on complète le flux dans `cfmsvl` immédiatement après l'appel à `cfbsc3`. En effet, dans ce cas, la convection est explicite ($\rho^{n+\frac{1}{2}} = \rho^n$, obtenu en imposant `ICONV(ISCA(IRHO))=0`) et le sous-programme `cfbsc3`, qui calcule le flux de masse aux faces, ne prend pas en compte la contribution du terme $\rho^{n+\frac{1}{2}}\,\underline{w}^n\cdot\underline{S}$. On ajoute donc cette contribution dans `cfmsvl`, après l'appel à `cfbsc3`. Au bord, en particulier, c'est bien le flux de masse calculé à partir des conditions aux limites que l'on obtient.

On actualise la pression à la fin de l'étape, en utilisant la loi d'état :

$$P_i^{pred} = P(\rho_i^{n+1}, \varepsilon_i^n) \tag{V.C.25}$$

[2] On ne reconstruit pas les valeurs de $\Delta t\, c^2$ aux points I' et J'.

Points à traiter

Le calcul du flux de masse au bord n'est pas entièrement satisfaisant si la convection est traitée de manière implicite (algorithme non standard, non testé, associé à l'équation (V.C.8), correspondant au choix $\rho^{n+\frac{1}{2}} = \rho^{n+1}$ et obtenu en imposant `ICONV(ISCA(IRHO))=1`). En effet, après `cfmsfl`, il faut déterminer la vitesse de convection $\underline{w}^n$ pour qu'apparaisse $\rho^{n+1}\underline{w}^n \cdot \underline{n}$ au cours de la résolution par `codits`. De ce fait, on doit déduire une valeur de $\underline{w}^n$ à partir de la valeur du flux de masse. Au bord, en particulier, il faut donc diviser le flux de masse issu des conditions aux limites par la valeur de bord de ρ^{n+1}. Or, lorsque des conditions de Neumann sont appliquées à la masse volumique, la valeur de ρ^{n+1} au bord n'est pas connue avant la résolution du système. On utilise donc, au lieu de la valeur de bord inconnue de ρ^{n+1} la valeur de bord prise au pas de temps précédent ρ^n. Cette approximation est susceptible d'affecter la valeur du flux de masse au bord.

D- cfqdmv routine

Fonction

Pour les notations et l'algorithme dans son ensemble, on se reportera à `cfbase`.

Dans le premier pas fractionnaire (`cfmsvl`), on a résolu une équation sur la masse volumique, obtenu une prédiction de la pression et un flux convectif "acoustique". On considère ici un second pas fractionnaire au cours duquel seul varie le vecteur flux de masse $\underline{Q} = \rho \underline{u}$ (seule varie la vitesse au centre des cellules). On résout l'équation de Navier-Stokes indépendamment pour chaque direction d'espace, et l'on utilise le flux de masse acoustique calculé précédemment comme flux convecteur (on pourrait aussi utiliser le vecteur quantité de mouvement du pas de temps précédent). De plus, on résout en variable $\underline{u}$ et non $\underline{Q}$.

Le système à résoudre entre t^* et t^{**} est (on exclut la turbulence, dont le traitement n'a rien de particulier dans le module compressible) :

$$\left\{ \begin{array}{l} \rho^{**} = \rho^* = \rho^{n+1} \\ \\ \dfrac{\partial \rho \underline{u}}{\partial t} + \underline{\mathrm{div}}\,(\underline{u} \otimes \underline{Q}_{ac}) + \underline{\nabla} P = \rho \underline{f}_v + \underline{\mathrm{div}}\,(\underline{\underline{\Sigma}}^v) \\ \\ e^{**} = e^* = e^n \end{array} \right. \tag{V.D.1}$$

La résolution de cette étape est similaire à l'étape de prédiction des vitesses du schéma de base de code_saturne.

Discrétisation

Discrétisation en temps

On implicite le terme de convection, éventuellement le gradient de pression (suivant la valeur de `IGRDPP`, en utilisant la pression prédite lors de l'étape acoustique) et le terme en gradient du tenseur des contraintes visqueuses. On explicite les autres termes du tenseur des contraintes visqueuses. On implicite les forces volumiques en utilisant ρ^{n+1}.

On obtient alors l'équation discrète suivante :

$$\begin{array}{l} \dfrac{(\rho \underline{u})^{n+1} - (\rho \underline{u})^n}{\Delta t^n} + \underline{\mathrm{div}}\,(\underline{u}^{n+1} \otimes \underline{Q}_{ac}^{n+1}) - \underline{\mathrm{div}}\,(\mu^n \underline{\underline{\nabla}} \underline{u}^{n+1}) \\ \\ \quad = \rho^{n+1} \underline{f}_v - \underline{\nabla} \widetilde{P} + \underline{\mathrm{div}}\,\left(\mu^n \ {}^t\underline{\underline{\nabla}} \underline{u}^n + (\kappa^n - \tfrac{2}{3}\mu^n) \mathrm{div}\,\underline{u}^n \ \underline{\underline{Id}} \right) \end{array} \tag{V.D.2}$$

avec $\widetilde{P} = P^n$ ou P^{Pred} suivant la valeur de `IGRDPP` (P^n par défaut).

En pratique, dans code_saturne, on résout cette équation en faisant apparaître à gauche l'écart $\underline{u}^{n+1} - \underline{u}^n$. Pour cela, on écrit la dérivée en temps discrète sous la forme suivante :

$$\begin{aligned}\frac{(\rho\underline{u})^{n+1}-(\rho\underline{u})^n}{\Delta t^n} &= \frac{\rho^{n+1}\,\underline{u}^{n+1}-\rho^n\,\underline{u}^n}{\Delta t^n}\\ &= \frac{\rho^n\,\underline{u}^{n+1}-\rho^n\,\underline{u}^n}{\Delta t^n}+\frac{\rho^{n+1}\,\underline{u}^{n+1}-\rho^n\,\underline{u}^{n+1}}{\Delta t^n}\\ &= \frac{\rho^n}{\Delta t^n}\left(\underline{u}^{n+1}-\underline{u}^n\right)+\underline{u}^{n+1}\frac{\rho^{n+1}-\rho^n}{\Delta t^n}\end{aligned} \tag{V.D.3}$$

et l'on utilise alors l'équation de la masse discrète pour écrire :

$$\frac{(\rho\underline{u})^{n+1}-(\rho\underline{u})^n}{\Delta t^n} = \frac{\rho^n}{\Delta t^n}\left(\underline{u}^{n+1}-\underline{u}^n\right)-\underline{u}^{n+1}\mathrm{div}\;\underline{Q}_{ac}^{n+1} \tag{V.D.4}$$

Discrétisation en espace

Introduction

On intègre l'équation (V.D.2) sur la cellule i de volume Ω_i et on obtient l'équation discrétisée en espace :

$$\begin{aligned}&\frac{\Omega_i}{\Delta t^n}(\rho_i^{n+1}\underline{u}_i^{n+1}-\rho_i^n\underline{u}_i^n)+\sum_{j\in V(i)}(\underline{u}^{n+1}\otimes\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}-\sum_{j\in V(i)}\left(\mu^n\underline{\underline{\nabla}}\underline{u}^{n+1}\right)_{ij}\cdot\underline{S}_{ij}\\ &=\Omega_i\rho_i^{n+1}\underline{f}_{vi}-\Omega_i(\underline{\nabla}\widetilde{P})_i+\sum_{j\in V(i)}\left(\mu^n\;{}^t\underline{\underline{\nabla}}\underline{u}^n+(\kappa^n-\frac{2}{3}\mu^n)\mathrm{div}\,\underline{u}^n\;\underline{\underline{Id}}\right)_{ij}\underline{S}_{ij}\end{aligned} \tag{V.D.5}$$

Discrétisation de la partie "convective"

La valeur à la face s'écrit :

$$(\underline{u}^{n+1}\otimes\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}=\underline{u}_{ij}^{n+1}(\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij} \tag{V.D.6}$$

avec un décentrement sur la valeur de $\underline{u}^{n+1}$ aux faces :

$$\begin{aligned}\underline{u}_{ij}^{n+1} &= \underline{u}_i^{n+1} \quad \text{si} \quad (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}\geqslant 0\\ &= \underline{u}_j^{n+1} \quad \text{si} \quad (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}<0\end{aligned} \tag{V.D.7}$$

que l'on peut noter :

$$\underline{u}_{ij}^{n+1}=\beta_{ij}\underline{u}_i^{n+1}+(1-\beta_{ij})\underline{u}_j^{n+1} \tag{V.D.8}$$

avec

$$\begin{cases}\beta_{ij}=1 \quad \text{si} \quad (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}\geqslant 0\\ \beta_{ij}=0 \quad \text{si} \quad (\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij}<0\end{cases} \tag{V.D.9}$$

Discrétisation de la partie "diffusive"

La valeur à la face s'écrit :

$$\left(\mu^n\underline{\underline{\nabla}}\underline{u}^{n+1}\right)_{ij}\underline{S}_{ij}=\mu_{ij}^n\left(\frac{\partial\underline{u}}{\partial n}\right)_{ij}^{n+1}S_{ij} \tag{V.D.10}$$

avec une interpolation linéaire pour μ^n aux faces (en pratique avec $\alpha_{ij}=\frac{1}{2}$) :

$$\mu_{ij}^n=\alpha_{ij}\mu_i^n+(1-\alpha_{ij})\mu_j^n \tag{V.D.11}$$

et un schéma centré pour le gradient normal aux faces :

$$\left(\frac{\partial\underline{u}}{\partial n}\right)_{ij}^{n+1}=\frac{\underline{u}_{J'}^{n+1}-\underline{u}_{I'}^{n+1}}{\overline{I'J'}} \tag{V.D.12}$$

Discrétisation du gradient de pression

On utilise `grdcel` standard. Suivant la valeur de `IMRGRA`, cela correspond à une reconstruction itérative ou par moindres carrés.

Discrétisation du "reste" du tenseur des contraintes visqueuses

On calcule des gradients aux cellules et on utilise une interpolation linéaire aux faces (avec, en pratique, $\alpha_{ij} = \frac{1}{2}$) :

$$\begin{aligned}\left(\mu^n\ {}^t\underline{\underline{\nabla}}\underline{u}^n + (\kappa^n - \tfrac{2}{3}\mu^n)\mathrm{div}\,\underline{u}^n\ \underline{\underline{Id}}\right)_{ij} \cdot \underline{S}_{ij} = \Big\{ &\alpha_{ij}\left(\mu^n\ {}^t\underline{\underline{\nabla}}\underline{u}^n + (\kappa^n - \tfrac{2}{3}\mu^n)\mathrm{div}\,\underline{u}^n\ \underline{\underline{Id}}\right)_i \\ &+(1-\alpha_{ij})\left(\mu^n\ {}^t\underline{\underline{\nabla}}\underline{u}^n + (\kappa^n - \tfrac{2}{3}\mu^n)\mathrm{div}\,\underline{u}^n\ \underline{\underline{Id}}\right)_j \Big\} \cdot \underline{S}_{ij}\end{aligned} \tag{V.D.13}$$

Mise en œuvre

On résout les trois directions d'espace du système (V.D.5) successivement et indépendamment :

$$\left\{\begin{array}{l} \dfrac{\Omega_i}{\Delta t^n}(\rho_i^{n+1}u_{i(\alpha)}^{n+1} - \rho_i^n u_{i(\alpha)}^n) + \displaystyle\sum_{j\in V(i)} u_{ij\,(\alpha)}^{n+1}(\underline{Q}_{ac}^{n+1})_{ij}\cdot\underline{S}_{ij} - \sum_{j\in V(i)}\mu_{ij}^n\frac{u_{j(\alpha)}^{n+1} - u_{i(\alpha)}^{n+1}}{\overline{I'J'}}S_{ij} \\ \qquad = \Omega_i\rho_i^{n+1}f_{vi(\alpha)} - \Omega_i(\underline{\nabla}\widetilde{P})_{i(\alpha)} \\ \qquad + \displaystyle\sum_{j\in V(i)}\left((\mu^n\ {}^t\underline{\underline{\nabla}}\underline{u}^n)_{ij}\cdot\underline{S}_{ij}\right)_{(\alpha)} + \sum_{j\in V(i)}\left((\kappa^n - \frac{2}{3}\mu^n)\mathrm{div}\,\underline{u}^n\right)_{ij}S_{ij\,(\alpha)} \\ i = 1\ldots N \qquad \text{et} \qquad (\alpha) = x,\ y,\ z \end{array}\right. \tag{V.D.14}$$

Chaque système associé à une direction est résolu par une méthode d'incrément et résidu en utilisant une méthode de Jacobi.

E- cfxtcl routine

Fonction

Pour le traitement des conditions aux limites, on considère le système (V.E.1)

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \operatorname{div}(\underline{Q}) = 0 \\ \dfrac{\partial \underline{Q}}{\partial t} + \underline{\operatorname{div}}(\underline{u} \otimes \underline{Q}) + \nabla P = \rho \underline{f}_v + \underline{\operatorname{div}}(\underline{\underline{\Sigma}}^v) \\ \dfrac{\partial E}{\partial t} + \operatorname{div}(\underline{u}(E+P)) = \rho \underline{f}_v \cdot \underline{u} + \operatorname{div}(\underline{\underline{\Sigma}}^v \underline{u}) - \operatorname{div}\, \underline{\Phi}_s + \rho \Phi_v \end{cases} \tag{V.E.1}$$

en tant que système hyperbolique portant sur la variable vectorielle $\underline{W} = {}^t(\rho, \underline{Q}, E)$.

Le système s'écrit alors :

$$\frac{\partial \underline{W}}{\partial t} + \sum_{i=1}^{3} \frac{\partial}{\partial x_i} \underline{F}_i(\underline{W}) = \sum_{i=1}^{3} \frac{\partial}{\partial x_i} \underline{F}_i^D(\underline{W}, \nabla \underline{W}) + \underline{\mathcal{S}} \tag{V.E.2}$$

où les $\underline{F}_i(\underline{W})$ sont les vecteurs flux convectifs et les $\underline{F}_i^D(\underline{W})$ sont les vecteurs flux diffusifs dans les trois directions d'espace, et $\underline{\mathcal{S}}$ est un terme source.

La démarche classique de code_saturne est adoptée : on impose les conditions aux limites en déterminant, pour chaque variable, des valeurs numériques de bord. Ces valeurs sont calculées de telle façon que, lorsqu'on les utilise dans les formules standard donnant les flux discrets, on obtienne les contributions souhaitées au bord.

Pour rendre compte des flux convectifs (aux entrées et aux sorties en particulier), on fait abstraction des flux diffusifs et des termes sources pour résoudre un problème de Riemann qui fournit un vecteur d'état au bord. Celui-ci permet de calculer un flux, soit directement (par les formules discrètes standard), soit en appliquant un schéma de Rusanov (schéma de flux décentré).

En paroi, on résout également, dans certains cas, un problème de Riemann pour déterminer une pression au bord.

See the programmers reference of the dedicated subroutine for further details.

Discrétisation

Introduction

Objectif

On résume ici les différentes conditions aux limites utilisées pour l'algorithme compressible afin de fournir une vue d'ensemble. Pour atteindre cet objectif, il est nécessaire de faire référence à des éléments relatifs à la discrétisation et au mode d'implantation des conditions aux limites.

Lors de l'implantation, on a cherché à préserver la cohérence avec l'approche utilisée dans le cadre standard de l'algorithme incompressible de code_saturne. Il est donc conseillé d'avoir pris connaissance du mode de traitement des conditions aux limites incompressibles avant d'aborder les détails de l'algorithme compressible.

Comme pour l'algorithme incompressible, les conditions aux limites sont imposées par le biais d'une valeur de bord associée à chaque variable. De plus, pour certaines frontières (parois à température imposée ou à flux thermique imposé), on dispose de deux valeurs de bord pour la même variable, l'une d'elles étant dédiée au calcul du flux diffusif. Enfin, sur certains types d'entrée et de sortie, on définit également une valeur du flux convectif au bord.

Comme pour l'algorithme incompressible, l'utilisateur peut définir, pour chaque face de bord, des conditions aux limites pour chaque variable, mais on conseille cependant d'utiliser uniquement les types prédéfinis décrits ci-après (entrée, sortie, paroi, symétrie) qui ont l'avantage d'assurer la cohérence entre les différentes variables et les différentes étapes de calcul.

Parois

Pression : on doit disposer d'une condition pour le calcul du gradient qui intervient dans l'étape de quantité de mouvement. On dispose de deux types de condition, au choix de l'utilisateur :

- par défaut, la pression imposée au bord est proportionnelle à la valeur interne (la pression au bord est obtenue comme solution d'un problème de Riemann sur les équations d'Euler avec un état miroir ; on distingue les cas de choc et de détente et, dans le cas d'une détente trop forte, une condition de Dirichlet homogène est utilisée pour éviter de voir apparaître une pression négative),
- si l'utilisateur le souhaite (`ICFGRP=1`), le gradient de pression est imposé à partir du profil de pression hydrostatique.

Vitesse et turbulence : traitement standard (voir la documentation des sous-programmes `condli` et `clptur`).

Scalaires passifs : traitement standard (flux nul par défaut imposé dans `typecl`).

Masse volumique : traitement standard des scalaires (flux nul par défaut imposé dans `typecl`).

Énergie et température[1] : traitement standard des scalaires (flux nul par défaut imposé dans `typecl`), hormis pour le calcul du flux diffusif dans le cas de parois à température imposée ou à flux thermique imposé.

Flux diffusif pour l'énergie en paroi : l'utilisateur peut choisir (dans `uscfcl`) entre une température de paroi imposée et un flux thermique diffusif (ou "conductif") imposé. S'il ne précise rien, on considère que la paroi est adiabatique (flux thermique diffusif imposé et de valeur nulle). Dans tous les cas, il faut donc disposer d'un moyen d'imposer le flux diffusif souhaité. Pour cela, on détermine une valeur de bord pour l'énergie qui, introduite dans la formule donnant le flux discret, permettra d'obtenir la contribution attendue (voir le paragraphe E). Conformément à l'approche classique de code_saturne, cette valeur est stockée sous la forme d'un couple de coefficients (de type `COEFAF`, `COEFBF`). Il est important de souligner que cette valeur de bord ne doit être utilisée que pour le calcul du flux diffusif : dans les autres situations pour lesquelles une valeur de bord de l'énergie ou de la température est requise (calcul de gradient par exemple), on utilise une condition de flux nul (traitement standard des scalaires). Pour cela, on dispose d'une seconde valeur de bord qui est stockée au moyen d'un couple de coefficients (`COEFA`, `COEFB`) distinct du précédent.

Flux convectifs : le flux de masse dans la direction normale à la paroi est pris nul. De ce fait, les flux convectifs seront nuls quelle que soit les valeurs de bord imposées pour les différentes variables transportées.

Symétrie

Les conditions appliquées sont les conditions classiques de l'algorithme incompressible (vitesse normale nulle, flux nul pour les autres variables).

[1] Le gradient de température est *a priori* inutile, mais peut être requis par l'utilisateur.

Elles sont imposées dans le sous-programme `typecl` essentiellement. Pour la pression, la condition de flux nul est imposée dans `cfxtcl` (au début des développements, on appliquait le même traitement qu'en paroi, mais une condition de flux nul a été préférée afin de s'affranchir des problèmes potentiels dans les configurations 2D).

Entrées et sorties

On obtient, par résolution d'un problème de Riemann au bord, complété par des relations de thermodynamique (`uscfth`), des valeurs de bord pour toutes les variables (on suppose qu'en entrée, toutes les composantes de la vitesse sont fournies ; elles sont supposées nulles par défaut, hormis pour les entrées á $(\rho, \underline{u})$ imposés, `IERUCF`, pour lesquelles il faut fournir la vitesse explicitement).

Ces valeurs de bord sont utilisées de deux façons :

- elles sont utilisées pour calculer les flux convectifs, en faisant appel au schéma de Rusanov (sauf en sortie supersonique) ; ces flux sont directement intégrés au second membre des équations à résoudre.
- elles servent de valeur de Dirichlet dans toutes les autres configurations pour lesquelles une valeur de bord est requise (calcul de flux diffusif, calcul de gradient...)

Deux cas particuliers :

- aux entrées ou sorties pour lesquelles toutes les variables sont imposées (`IESICF`), on utilise une condition de Neumann homogène pour la pression (hormis pour le calcul du gradient intervenant dans l'équation de la quantité de mouvement, qui est pris en compte par le flux convectif déterminé par le schéma de Rusanov). Ce choix est arbitraire (on n'a pas testé le comportement de l'algorithme si l'on conserve une condition de Dirichlet sur la pression), mais a été fait en supposant qu'une condition de Neumann homogène serait *a priori* moins déstabilisante, dans la mesure où, pour ce type de frontière, l'utilisateur peut imposer une valeur de pression très différente de celle régnant à l'intérieur du domaine (la valeur imposée est utilisée pour le flux convectif).
- pour les grandeurs turbulentes et les scalaires utilisateur, si le flux de masse est entrant et que l'on a fourni une valeur de Dirichlet (`RCODCL(*,*,1)` dans `uscfcl`), on l'utilise, pour le calcul du flux convectif et du flux diffusif ; sinon, on utilise une condition de Neumann homogène (le concept de sortie de type 9 ou 10 est couvert par cette approche).

Problème de Riemann au bord

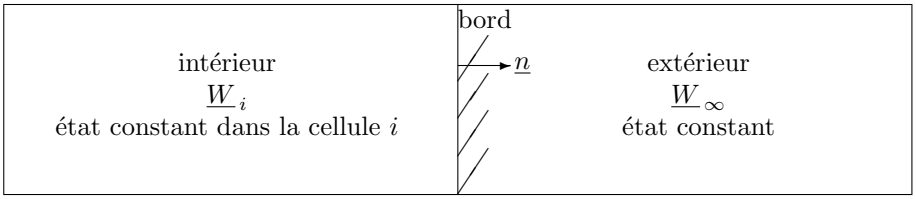
Introduction

On cherche à obtenir un état au bord, pour les entrées, les sorties et les parois.

Pour cela, on fait abstraction des flux diffusifs et des sources. Le système résultant est alors appelé système d'équations d'Euler. On se place de plus dans un repère orienté suivant la normale au bord considéré $(\underline{\tau}_1, \underline{\tau}_2, \underline{n})$ et l'on ne considère que les variations suivant cette normale. Le système devient donc :

$$\frac{\partial \underline{W}}{\partial t} + \frac{\partial}{\partial n}\underline{F}_n(\underline{W}) = 0 \quad \text{avec} \quad \underline{F}_n(\underline{W}) = \sum_{i=1}^{3} n_i \underline{F}_i(\underline{W}) \quad \text{et} \quad \frac{\partial}{\partial n} = \sum_{i=1}^{3} n_i \frac{\partial}{\partial x_i} \tag{V.E.3}$$

Pour déterminer les valeurs des variables au bord, on recherche l'évolution du problème instationnaire suivant, appelé problème de Riemann :

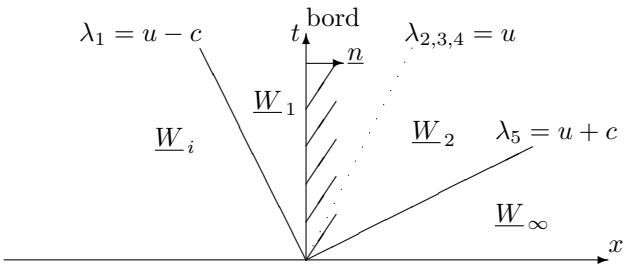


avec $\underline{W}_\infty$ dépendant du type de bord et différent de $\underline{W}_i$ *a priori*.

Pour résoudre ce problème de Riemann, on utilisera les variables non-conservatives $\widetilde{\underline{W}} = {}^t(\rho, \underline{u}, P)$ et l'on retrouvera l'énergie grâce à l'équation d'état.

Pour alléger l'écriture, dans le présent paragraphe E, on notera aussi $\underline{W}$ le vecteur ${}^t(\rho, \underline{u}, P)$ et $\underline{u} = \underline{u}_\tau + u\,\underline{n}$ (en posant $u = \underline{u}\cdot\underline{n}$ et $\underline{u}_\tau = \underline{u} - (\underline{u}\cdot\underline{n})\underline{n}$).

La solution est une suite d'états constants, dont les valeurs dépendent de $\underline{W}_i$ et $\underline{W}_\infty$, séparés par des ondes se déplaçant à des vitesses données par les valeurs propres du système $(\lambda_i)_{i=1\ldots5}$. On représente les caractéristiques du système sur le schéma suivant :



Comme valeurs des variables au bord, on prendra les valeurs correspondant à l'état constant qui contient le bord ($\underline{W}_1$ dans l'exemple précédent).

Il faut remarquer que la solution du problème de Riemann dépend de la thermodynamique et devra donc être calculée et codée par l'utilisateur si la thermodynamique n'a pas été prévue (en version 1.2, la seule thermodynamique prévue est celle des gaz parfaits).

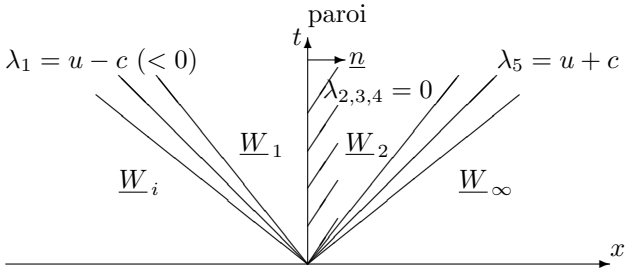
En paroi, pour la condition de pression (sans effet de gravité)

Pour les faces de paroi, on définit à l'extérieur du domaine un état miroir $\underline{W}_\infty$ par :

$$\underline{W}_i = \begin{pmatrix} \rho_i \\ \underline{u}_{\tau i} \\ u_i \\ P_i \end{pmatrix} \qquad \underline{W}_\infty = \begin{pmatrix} \rho_\infty = \rho_i \\ \underline{u}_{\tau\infty} = \underline{u}_{\tau i} \\ u_\infty = -u_i \\ P_\infty = P_i \end{pmatrix} \tag{V.E.4}$$

Les solutions dépendent de l'orientation de la vitesse dans la cellule de bord :

1. Si $u_i \leqslant 0$, la solution est une double détente symétrique.

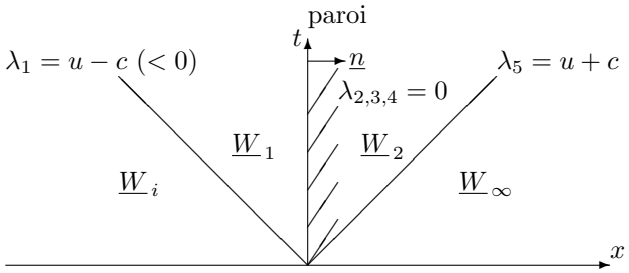


$$\underline{W}_{paroi} = \underline{W}_1 = \underline{W}_2 \quad \begin{cases} \rho_p = \rho_1 = \rho_2 \\ u_p = u_1 = u_2 \\ P_p = P_1 = P_2 \end{cases}$$

La conservation de la vitesse tangentielle à travers la 1-onde donne $\underline{u}_{\tau p} = \underline{u}_{\tau i}$. Par des considérations de symétrie on trouve $u_p = 0$. Puis on obtient ρ_p et P_p en écrivant la conservation des invariants de Riemann à travers la 1-détente :

$$\begin{cases} u_1 + \displaystyle\int_0^{\rho_1} \frac{c}{\rho} d\rho = u_i + \int_0^{\rho_i} \frac{c}{\rho} d\rho \\ \\ s_1 = s_i \end{cases} \Rightarrow \begin{cases} \displaystyle\int_{\rho_i}^{\rho_1} \frac{c}{\rho} d\rho = u_i & \Rightarrow \rho_p = \rho_1 \\ \\ s(P_1, \rho_1) = s(P_i, \rho_i) & \Rightarrow P_p = P_1 \end{cases} \tag{V.E.5}$$

2. Si $u_i > 0$, la solution est un double choc symétrique.



$$\underline{W}_{paroi} = \underline{W}_1 = \underline{W}_2 \quad \begin{cases} \rho_p = \rho_1 = \rho_2 \\ u_p = u_1 = u_2 \\ P_p = P_1 = P_2 \end{cases}$$

De même que précédemment, on trouve $\underline{u}_{\tau p} = \underline{u}_{\tau i}$ et $u_p = 0$, puis ρ_p et P_p en écrivant les relations de saut à travers le 1-choc :

$$\begin{cases} \rho_1 \rho_i (u_1 - u_i)^2 = (P_1 - P_i)(\rho_1 - \rho_i) \\ \\ 2\rho_1 \rho_i (\varepsilon_1 - \varepsilon_i) = (P_1 + P_i)(\rho_1 - \rho_i) \end{cases} \text{ avec } \varepsilon = \varepsilon(P, \rho) \Rightarrow \begin{cases} \rho_p = \rho_1 \\ \\ P_p = P_1 \end{cases} \tag{V.E.6}$$

Pour les gaz parfaits, avec $M_i = \dfrac{\underline{u}_i \cdot \underline{n}}{c_i}$ (Nombre de Mach de paroi), on a :

- Cas détente ($M_i \leqslant 0$) :

$$\begin{cases} P_p = 0 & \text{si} \quad 1 + \dfrac{\gamma - 1}{2} M_i < 0 \\ \\ P_p = P_i \left(1 + \dfrac{\gamma - 1}{2} M_i \right)^{\frac{2\gamma}{\gamma - 1}} & \text{sinon} \end{cases}$$

$$\rho_p = \rho_i \left(\frac{P_p}{P_i} \right)^{\frac{1}{\gamma}}$$

- Cas choc ($M_i > 0$) :

$$P_p = P_i \left(1 + \frac{\gamma(\gamma+1)}{4} M_i^2 + \gamma M_i \sqrt{1 + \frac{(\gamma+1)^2}{16} M_i^2} \right)$$

$$\rho_p = \rho_i \left(\frac{P_p - P_i}{P_p - P_i - \rho_i (\underline{u}_i \cdot \underline{n})^2} \right)$$

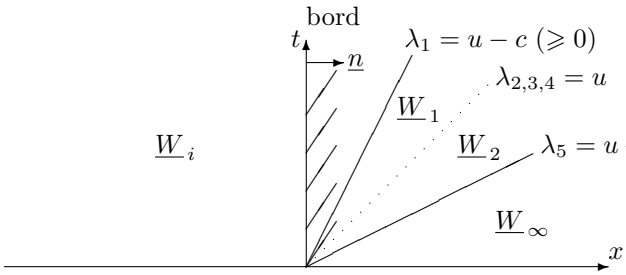
En pratique, le flux convectif normal à la paroi est nul et seule la condition de pression déterminée ci-dessus est effectivement utilisée (pour le calcul du gradient sans effet de gravité).

En sortie

Il existe deux cas de traitement des conditions en sortie, selon le nombre de Mach normal à la face de bord (c_i est la vitesse du son dans la cellule de bord) :

$$M_i = \frac{u_i}{c_i} = \frac{\underline{u}_i \cdot \underline{n}}{c_i}$$

Sortie supersonique (condition ISSPCF de `uscfcl`) : $M_i \geqslant 1 \Rightarrow u_i - c_i \geqslant 0$

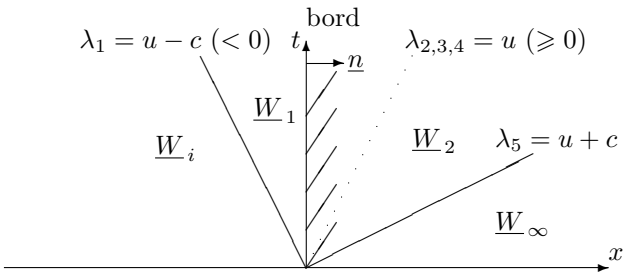


$\underline{W}_{bord} = \underline{W}_i$

Toutes les caractéristiques sont sortantes, on connaît donc toutes les conditions au bord :

$$\begin{cases} \rho_b = \rho_i \\ \underline{u}_{\tau b} = \underline{u}_{\tau i} \\ u_b = u_i \\ P_b = P_i \end{cases} \tag{V.E.7}$$

Sortie subsonique (condition ISOPCF de `uscfcl`) : $0 \leqslant M_i < 1 \Rightarrow (u_i \geqslant 0 \text{ et } u_i - c_i < 0)$



$\underline{W}_{bord} = \underline{W}_1 \quad \begin{cases} \rho_b = \rho_1 \\ u_b = u_1 \\ P_b = P_1 \end{cases}$

On a une caractéristique entrante, on doit donc imposer une seule condition au bord (en général la pression de sortie P_{ext}).

On connaît alors $P_b = P_{ext}$ et $\underline{u}_{\tau b} = \underline{u}_{\tau i}$ (par conservation de la vitesse tangentielle à travers la 1-onde). Pour trouver les inconnues manquantes (ρ_b et u_b) on doit résoudre le passage de la 1-onde :

1. Si $P_{ext} \leqslant P_i$, on a une 1-détente.

   On écrit la conservation des invariants de Riemann à travers la 1-détente :

$$\left\{ \begin{array}{l} s_1 = s_i \\ \\ u_1 + \displaystyle\int_0^{\rho_1} \frac{c}{\rho} d\rho = u_i + \int_0^{\rho_i} \frac{c}{\rho} d\rho \end{array} \right. \Rightarrow \left\{ \begin{array}{ll} s(P_{ext}, \rho_1) = s(P_i, \rho_i) & \Rightarrow \rho_b = \rho_1 \\ \\ u_1 = u_i - \displaystyle\int_{\rho_i}^{\rho_1} \frac{c}{\rho} d\rho & \Rightarrow u_b = u_1 \end{array} \right. \tag{V.E.8}$$

2. Si $P_{ext} > P_i$, on a un 1-choc.

   On écrit les relations de saut à travers le 1-choc :

$$\left\{ \begin{array}{l} \rho_1 \rho_i (u_1 - u_i)^2 = (P_{ext} - P_i)(\rho_1 - \rho_i) \\ \\ 2\rho_1 \rho_i (\varepsilon(P_{ext}, \rho_1) - \varepsilon(P_i, \rho_i)) = (P_{ext} + P_i)(\rho_1 - \rho_i) \end{array} \right. \Rightarrow \left\{ \begin{array}{l} \rho_b = \rho_1 \\ \\ u_b = u_1 \end{array} \right. \tag{V.E.9}$$

   Pour les gaz parfaits, on a :

   - Cas détente ($P_{ext} \leqslant P_i$) :

$$P_b = P_{ext}$$

$$\rho_b = \rho_i \left(\frac{P_{ext}}{P_i} \right)^{\frac{1}{\gamma}}$$

   - Cas choc ($P_{ext} > P_i$) :

$$P_b = P_{ext}$$

$$\rho_b = \rho_i \left(\frac{P_{ext} - P_i}{P_{ext} - P_i - \rho_i (\underline{u}_i \cdot \underline{n} - \underline{u}_b \cdot \underline{n})^2} \right) = \rho_i \left(\frac{(\gamma+1) P_{ext} + (\gamma-1) P_i}{(\gamma-1) P_{ext} + (\gamma+1) P_i} \right)$$

La valeur de la masse volumique au bord intervient en particulier dans le flux de masse.

En entrée

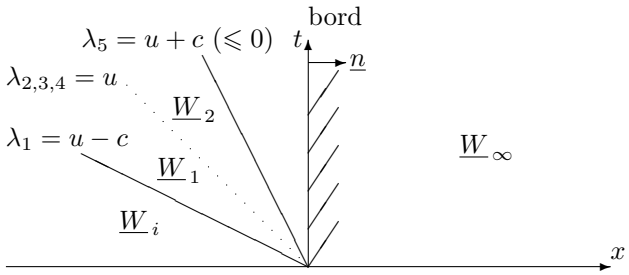
L'utilisateur impose les valeurs qu'il souhaite pour les variables en entrée :

$$\underline{W}_{ext} = \left(\begin{array}{c} \rho_{ext} \\ \underline{u}_{\tau ext} \\ u_{ext} \\ P_{ext} \end{array} \right)$$

De même que précédemment, il existe deux cas de traitement des conditions en entrée, pilotés par le nombre de Mach entrant, normalement à la face de bord (avec c_{ext} la vitesse du son en entrée) :

$$M_{ext} = \frac{u_{ext}}{c_{ext}} = \frac{\underline{u}_{ext} \cdot \underline{n}}{c_{ext}}$$

Entrée supersonique (condition IESICF de `uscfcl`) : $M_{ext} \leqslant -1 \Rightarrow u_{ext} + c_{ext} \leqslant 0$



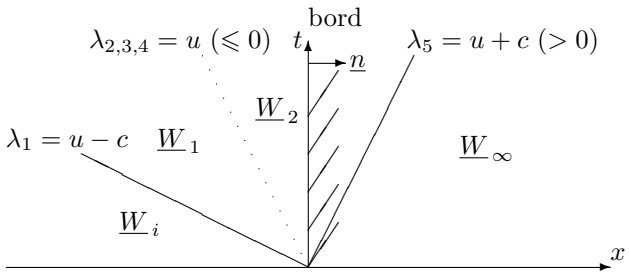
$\underline{W}_\infty$ $\qquad$ $\underline{W}_{bord} = \underline{W}_\infty = \underline{W}_{ext}$

Toutes les caractéristiques sont entrantes, toutes les conditions au bord sont donc imposées par l'utilisateur.

$$\left\{ \begin{array}{l} \rho_b = \rho_{ext} \\ \underline{u}_{\tau b} = \underline{u}_{\tau ext} \\ u_b = u_{ext} \\ P_b = P_{ext} \end{array} \right. \tag{V.E.10}$$

Entrée subsonique (condition IERUCF de `uscfcl`) :

$$-1 < M_{ext} \leqslant 0 \Rightarrow (u_{ext} \leqslant 0 \text{ et } u_{ext} + c_{ext} > 0)$$



$\underline{W}_\infty$ $\qquad$ $\underline{W}_{bord} = \underline{W}_2$ $\quad \left\{ \begin{array}{l} \rho_b = \rho_2 \\ u_b = u_2 \\ P_b = P_2 \end{array} \right.$

On a une caractéristique sortante. L'utilisateur doit donc laisser un degré de liberté.

En général, on impose le flux de masse en entrée, donc ρ_{ext} et u_{ext}, et l'on calcule la pression au bord en résolvant le passage des 1∼4-ondes. On connaît aussi $\underline{u}_{\tau b} = \underline{u}_{\tau ext}$, par conservation de la vitesse tangentielle à travers la 5-onde.

1. Si $u_{ext} \geqslant u_i$, on a une 1-détente.

   On écrit la conservation des invariants de Riemann à travers la 1-détente et la conservation de la vitesse et de la pression à travers le contact :

$$\left\{ \begin{array}{l} u_1 + \displaystyle\int_0^{\rho_1} \frac{c}{\rho} d\rho = u_i + \int_0^{\rho_i} \frac{c}{\rho} d\rho \\ \\ s_1 = s_i \end{array} \right. \qquad \left\{ \begin{array}{l} u_1 = u_2 = u_{ext} \\ \\ P_1 = P_2 \end{array} \right.$$

$$\Rightarrow \left\{ \begin{array}{ll} \displaystyle\int_{\rho_i}^{\rho_1} \frac{c}{\rho} d\rho = u_i - u_{ext} & \Rightarrow \rho_1 \\ \\ s(P_2, \rho_1) = s(P_i, \rho_i) & \Rightarrow P_b = P_2 \end{array} \right. \tag{V.E.11}$$

2. Si $u_{ext} < u_i$, on a un 1-choc.

   On écrit les relations de saut à travers le 1-choc et la conservation de la vitesse et de la pression à travers le contact :

$$\left\{\begin{array}{l} \rho_1\rho_i(u_1-u_i)^2 = (P_1-P_i)(\rho_1-\rho_i) \\ 2\rho_1\rho_i(\varepsilon_1-\varepsilon_i) = (P_1+P_i)(\rho_1-\rho_i) \\ \varepsilon = \varepsilon(P,\rho) \end{array}\right. \Rightarrow \left\{\begin{array}{l} \rho_1 \\ P_b = P_2 \end{array}\right. \qquad \text{(V.E.12)}$$

$$\left\{\begin{array}{l} u_1 = u_2 = u_{ext} \\ P_1 = P_2 \end{array}\right.$$

   Pour les gaz parfaits, on a :

   - Cas détente ($\delta M \leqslant 0$) :

$$\left\{\begin{array}{ll} P_b = 0 & \text{si} \quad 1+\dfrac{\gamma-1}{2}\delta M < 0 \\ P_b = P_i\left(1+\dfrac{\gamma-1}{2}\delta M\right)^{\frac{2\gamma}{\gamma-1}} & \text{sinon} \end{array}\right.$$

$$\rho_b = \rho_{ext}$$

   - Cas choc ($\delta M > 0$) :

$$P_b = P_i\left(1+\frac{\gamma(\gamma+1)}{4}\delta M^2 + \gamma\delta M\sqrt{1+\frac{(\gamma+1)^2}{16}\delta M^2}\right)$$

$$\rho_b = \rho_{ext}$$

Condition de pression en paroi avec effets de gravité

Le problème de Riemann considéré précédemment ne prend pas en compte les effets de la gravité. Or, dans certains cas, si l'on ne prend pas en compte le gradient de pression "hydrostatique", on peut obtenir une solution erronée (en particulier, par exemple, on peut créer une source de quantité de mouvement non physique dans un milieu initialement au repos).

Écrivons l'équilibre local dans la maille de bord :

$$\underline{\nabla} P = \rho \underline{g} \qquad \text{(V.E.13)}$$

Pour simplifier la résolution, on peut utiliser la formulation de (V.E.13) en incompressible (c'est cette approche qui a été adoptée dans code_saturne) :

$$\left(\underline{\nabla} P\right)_i = \rho_i \underline{g} \quad \text{ce qui donne} \quad P_{paroi} = P_i + \rho_i \underline{g}\cdot(\underline{x}_{paroi} - \underline{x}_i) \qquad \text{(V.E.14)}$$

Une autre approche (dépendante de l'équation d'état) consiste à résoudre l'équilibre local avec la formulation compressible (V.E.13), en supposant de plus que la maille est isentropique :

$$\left\{\begin{array}{l} \underline{\nabla} P = \rho \underline{g} \\ P = P(\rho, s_i) \end{array}\right. \qquad \text{(V.E.15)}$$

Ce qui donne, pour un gaz parfait :

$$P_{paroi} = P_i\left(1+\frac{\gamma-1}{\gamma}\frac{\rho_i}{P_i}\underline{g}\cdot(\underline{x}_{paroi}-\underline{x}_i)\right)^{\frac{\gamma}{\gamma-1}} \qquad \text{(V.E.16)}$$

Remarque : la formule issue de l'incompressible (V.E.14) est une linéarisation de la formule (V.E.16). Dans les cas courants elle s'éloigne très peu de la formule exacte. Dans des conditions extrêmes, si l'on considère par exemple de l'air à $1000K$ et $10bar$, avec une accélération de la pesanteur $g = 1000m/s^2$ et une différence de hauteur entre le centre de la cellule et le centre de la face de bord de $10m$, l'expression (V.E.16) donne $P_{paroi} = 1034640,4Pa$ et l'expression (V.E.14) donne $P_{paroi} = 1034644,7Pa$, soit une différence relative de moins de $0,001\%$. On voit aussi que la différence entre la pression calculée au centre de la cellule et celle calculée au bord est de l'ordre de 3%.

Schéma de Rusanov pour le calcul de flux convectifs au bord

Introduction

Le schéma de Rusanov est utilisé pour certains types de conditions aux limites afin de passer du vecteur d'état calculé au bord comme indiqué précédemment (solution du problème de Riemann) à un flux convectif de bord (pour la masse, la quantité de mouvement et l'énergie). L'utilisation de ce schéma (décentré amont) permet de gagner en stabilité.

Le schéma de Rusanov est appliqué aux frontières auxquelles on considère qu'il est le plus probable de rencontrer des conditions en accord imparfait avec l'état régnant dans le domaine, conditions qui sont donc susceptibles de déstabiliser le calcul : il s'agit des entrées et des sorties (frontières de type IESICF, ISOPCF, IERUCF, IEQHCF). En sortie supersonique (ISSPCF) cependant, le schéma de Rusanov est inutile et n'est donc pas appliqué : en effet, pour ce type de frontière, l'état imposé au bord est exactement l'état amont et le décentrement du schéma de Rusanov n'apporterait donc rien.

Principe

Pour le calcul du flux décentré de Rusanov, on considère le système hyperbolique constitué des seuls termes convectifs issus des équations de masse, quantité de mouvement et énergie. Ce système est écrit, par changement de variable, en non conservatif (on utilise la relation $P = \dfrac{\rho\varepsilon}{\gamma - 1}$ et on note u_ξ les composantes de $\underline{u}$) :

$$\left\{ \begin{array}{lcl} \dfrac{\partial \rho}{\partial t} + \rho \underline{\mathrm{div}}\ \underline{u} + \underline{u}\,\underline{\nabla}\,\rho = 0 \\ \dfrac{\partial u_\xi}{\partial t} + \underline{u}\,\underline{\nabla} u_\xi + \dfrac{1}{\rho}\dfrac{\partial P}{\partial \xi} = 0 \\ \dfrac{\partial P}{\partial t} + \gamma\, P\, \mathrm{div}\,\underline{u} + \underline{u}\,\underline{\nabla} P = 0 \end{array} \right. \qquad \text{(V.E.17)}$$

En notant le vecteur d'état $\underline{W} = (\rho, \underline{u}, P)^t$, ce système est noté :

$$\frac{\partial \underline{W}}{\partial t} + \mathrm{div}\ \underline{F}(\underline{W}) = 0 \qquad \text{(V.E.18)}$$

Avec $\delta\,\underline{W}$ l'incrément temporel du vecteur d'état, $\underline{n}$ la normale à une face, ij la face interne partagée par les cellules i et j et ik la face de bord k associée à la cellule i, la discrétisation spatiale conduit à :

$$\frac{|\Omega_i|}{\Delta t}\delta\,\underline{W}_i + \sum_{j\in Vois(i)} \int_{S_{ij}} \underline{F}(\underline{W})\,\underline{n}\,dS + \sum_{k\in\gamma_b(i)} \int_{S_{b_{ik}}} \underline{F}(\underline{W})\,\underline{n}\,dS = 0 \qquad \text{(V.E.19)}$$

Sur une face de bord donnée, on applique le schéma de Rusanov pour calculer le flux comme suit :

$$\frac{1}{|S_{b_{ik}}|}\int_{S_{b_{ik}}} \underline{F}(\underline{W})\,\underline{n}\,dS = \frac{1}{2}\left(\underline{F}(\underline{W}_i) + \underline{F}(\underline{W}_{b_{ik}})\right)\cdot \underline{n}_{b_{ik}} - \frac{1}{2}\rho_{rus\,b_{ik}}\left(\underline{W}_{b_{ik}} - \underline{W}_i\right) = \underline{F}_{rus\,b_{ik}}(\underline{W}) \qquad \text{(V.E.20)}$$

Dans cette relation, $\underline{W}_{b_{ik}}$ est le vecteur d'état $\underline{W}_\infty$, connu au bord (tel qu'il résulte de la résolution du problème de Riemann au bord présentée plus haut pour chaque type de frontière considéré).

Paramètre de décentrement $\rho_{rus\,b_{ik}}$

Pour chaque face de bord, le scalaire $\rho_{rus\,b_{ik}}$ est la plus grande valeur du rayon spectral de la matrice jacobienne $\dfrac{\partial\,\underline{F}_n(\underline{W})}{\partial\underline{W}}$ obtenu pour les vecteurs d'état $\underline{W}_i$ et $\underline{W}_{b_{ik}}$.

$\underline{F}_n$ est la composante du flux $\underline{F}$ dans la direction de la normale à la face de bord, $\underline{n}_{b_{ik}}$. Utiliser $\underline{F}_n$ pour la détermination du paramètre de décentrement $\rho_{rus\,b_{ik}}$ relève d'une approche classique qui consiste à remplacer le système tridimensionnel initial par le système unidimensionnel projeté dans la direction normale à la face, en négligeant les variations du vecteur d'état $\underline{W}$ dans la direction tangeante à la face :

$$\frac{\partial\underline{W}}{\partial t}+\frac{\partial\,\underline{F}_n(\underline{W})}{\partial\underline{W}}\,\frac{\partial\underline{W}}{\partial n}=0 \tag{V.E.21}$$

De manière plus explicite, si l'on se place dans un repère de calcul ayant $\underline{n}_{b_{ik}}$ comme vecteur de base, et si l'on note u la composante de vitesse associée, le système est le suivant (les équations portant sur les composantes transverses de la vitesse sont découplées, associées à la valeur propre u, comme le serait un scalaire simplement convecté et ne sont pas écrites ci-après) :

$$\left\{\begin{array}{lcl} \dfrac{\partial\rho}{\partial t}+\rho\dfrac{\partial\,u}{\partial\,n}+u\,\dfrac{\partial\,\rho}{\partial\,n} &=& 0\\ \dfrac{\partial u}{\partial t}+u\,\dfrac{\partial\,u}{\partial\,n}+\dfrac{1}{\rho}\,\dfrac{\partial P}{\partial n} &=& 0\\ \dfrac{\partial P}{\partial t}+\gamma\,P\,\dfrac{\partial\,u}{\partial\,n}+u\,\dfrac{\partial\,P}{\partial\,n} &=& 0 \end{array}\right. \tag{V.E.22}$$

La matrice jacobienne associée est donc :

$$\begin{pmatrix} u & \rho & 0\\ 0 & u & \dfrac{1}{\rho}\\ 0 & \gamma\,P & 0 \end{pmatrix} \tag{V.E.23}$$

Les valeurs propres sont u et $u\pm c$ (avec $c=\sqrt{\frac{\gamma\,P}{\rho}}$). Le rayon spectral est donc $|u|+c$ et le paramètre de décentrement s'en déduit :

$$\rho_{rus\,b_{ik}}=max\left(|u_i|+c_i,|u_{b_{ik}}|+c_{b_{ik}}\right) \tag{V.E.24}$$

Expression des flux convectifs

Les flux convectifs calculés par le schéma de Rusanov pour les variables masse, quantité de mouvement et énergie représentent donc la discrétisation des termes suivants :

$$\left\{\begin{array}{l} \mathrm{div}\,(\underline{Q})\\ \underline{\mathrm{div}}\,(\underline{u}\otimes\underline{Q})+\underline{\nabla}\,P\\ \mathrm{div}\,\left(\underline{Q}\,(e+\dfrac{P}{\rho})\right) \end{array}\right. \tag{V.E.25}$$

Pour une face de bord ik adjacente à la cellule i et avec la valeur précédente de $\rho_{rus\,b_{ik}}$, on a :

$$\left\{\begin{array}{lcl}
\displaystyle\int_{S_{b_{ik}}} \underline{Q}\cdot\underline{n}\,dS & = & \displaystyle\frac{1}{2}\left((\underline{Q}_i+\underline{Q}_{b_{ik}})\cdot\underline{n}_{b_{ik}}\right)\,S_{b_{ik}} \\
 & & \displaystyle -\frac{1}{2}\,\rho_{rus\,b_{ik}}\,(\rho_{b_{ik}}-\rho_i)\,S_{b_{ik}} \\
\displaystyle\int_{S_{b_{ik}}} (\underline{u}\otimes\underline{Q}+\underline{\nabla}\,P)\cdot\underline{n}\,dS & = & \displaystyle\frac{1}{2}\left(\underline{u}_i(\underline{Q}_i\cdot\underline{n}_{b_{ik}})+P_i\,\underline{n}_{b_{ik}}+\underline{u}_{b_{ik}}(\underline{Q}_{b_{ik}}\cdot\underline{n}_{b_{ik}})+P_{b_{ik}}\,\underline{n}_{b_{ik}}\right)\,S_{b_{ik}} \\
 & & \displaystyle -\frac{1}{2}\,\rho_{rus\,b_{ik}}\left(\underline{Q}_{b_{ik}}-\underline{Q}_i\right)\,S_{b_{ik}} \\
\displaystyle\int_{S_{b_{ik}}} (e+\frac{P}{\rho})\,\underline{Q}\cdot\underline{n}\,dS & = & \displaystyle\frac{1}{2}\left((e_i+\frac{P_i}{\rho_i})\,(\underline{Q}_i\cdot\underline{n}_{b_{ik}})+(e_{b_{ik}}+\frac{P_{b_{ik}}}{\rho_{b_{ik}}})(\underline{Q}_{b_{ik}}\cdot\underline{n}_{b_{ik}})\right)\,S_{b_{ik}} \\
 & & \displaystyle -\frac{1}{2}\,\rho_{rus\,b_{ik}}\,(\rho_{b_{ik}}\,e_{b_{ik}}-\rho_i\,e_i)\,S_{b_{ik}}
\end{array}\right. \tag{V.E.26}$$

Conditions aux limites pour le flux diffusif d'énergie

Rappel

Pour le flux de diffusion d'énergie, les conditions aux limites sont imposées de manière similaire à ce qui est décrit dans la documentation de `clptur` et de `condli`. La figure (V.E.1) rappelle quelques notations usuelles et l'équation (V.E.27) traduit la conservation du flux normal au bord pour la variable f.

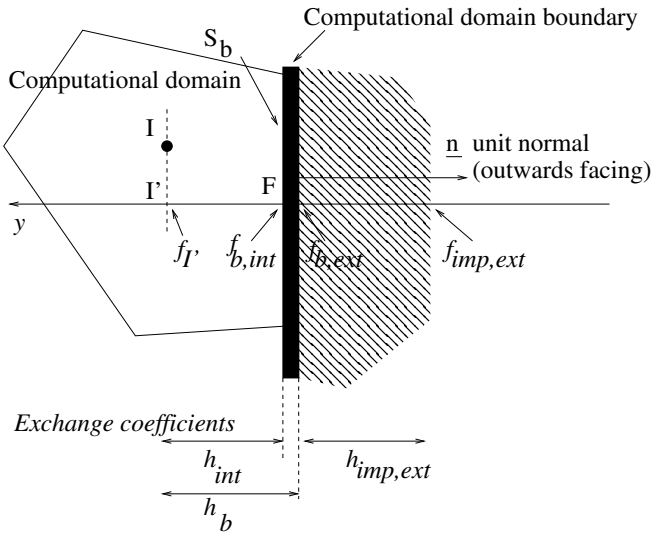


Figure V.E.1: Cellule de bord.

$$\underbrace{h_{int}(f_{b,int}-f_{I'})}_{\phi_{int}}=\underbrace{h_b(f_{b,ext}-f_{I'})}_{\phi_b}=\left\{\begin{array}{ll}
\underbrace{h_{imp,ext}(f_{imp,ext}-f_{b,ext})}_{\phi_{réel\ imposé}} & \text{(condition de Dirichlet)} \\
\underbrace{\phi_{imp,ext}}_{\phi_{réel\ imposé}} & \text{(condition de Neumann)}
\end{array}\right. \tag{V.E.27}$$

L'équation (V.E.28) rappelle la formulation des conditions aux limites pour une variable f.

$$f_{b,int} = \begin{cases} \dfrac{h_{imp,ext}}{h_{int}+h_r h_{imp,ext}} & f_{imp,ext} + \dfrac{h_{int}+h_{imp,ext}(h_r-1)}{h_{int}+h_r h_{imp,ext}} f_{I'} & \text{(condition de Dirichlet)} \\ \dfrac{1}{h_{int}} & \phi_{imp,ext} + f_{I'} & \text{(condition de Neumann)} \end{cases} \tag{V.E.28}$$

Les coefficients d'échange sont définis comme suit[2] :

$$\begin{cases} h_{int} &= \dfrac{\alpha}{\overline{\overline{I'F}}} \\ h_r &= \dfrac{h_{int}}{h_b} \\ h_b &= \dfrac{\phi_b}{f_{b,ext}-f_{I'}} = \dfrac{\rho\, C\, u_k}{f^+_{I'}} \end{cases} \tag{V.E.29}$$

Dans code_saturne, on note les conditions aux limites de manière générale sous la forme suivante :

$$f_{b,int} = A_b + B_b\, f_{I'} \tag{V.E.30}$$

avec A_b et B_b définis selon le type des conditions :

$$\text{Dirichlet} \begin{cases} A_b = \dfrac{h_{imp,ext}}{h_{int}+h_r h_{imp,ext}} f_{imp,ext} \\ B_b = \dfrac{h_{int}+h_{imp,ext}(h_r-1)}{h_{int}+h_r h_{imp,ext}} \end{cases} \quad \text{Neumann} \begin{cases} A_b = \dfrac{1}{h_{int}} \phi_{imp,ext} \\ B_b = 1 \end{cases} \tag{V.E.31}$$

Flux diffusif d'énergie

Dans le module compressible, on résout une équation sur l'énergie, qui s'écrit, si l'on excepte tous les termes hormis le flux de diffusion et le terme instationnaire, pour faciliter la présentation :

$$\begin{aligned} \frac{\partial \rho e}{\partial t} &= -\text{div}\, \underline{\Phi}_s \\ &= \text{div}\,(K\, \underline{\nabla} T) \quad \text{avec} \quad K = \lambda + C_p \frac{\mu_t}{\sigma_t} \\ &= \text{div}\left(K\, \underline{\nabla} \frac{e - \frac{1}{2}u^2 - \varepsilon_{sup}}{C_v} \right) \\ &= \text{div}\left(\frac{K}{C_v} \underline{\nabla}(e - \frac{1}{2}u^2 - \varepsilon_{sup}) \right) \quad \text{si} \quad C_v \text{ est constant} \\ &= \text{div}\left(\frac{K}{C_v} \underline{\nabla} e \right) - \text{div}\left(\frac{K}{C_v} \underline{\nabla}(\frac{1}{2}u^2 + \varepsilon_{sup}) \right) \end{aligned} \tag{V.E.32}$$

La décomposition en e et $\frac{1}{2}u^2 + \varepsilon_{sup}$ est purement mathématique (elle résulte du fait que l'on résout en énergie alors que le flux thermique s'exprime en fonction de la température). Aussi, pour imposer un flux de bord ou une température de bord (ce qui revient au même puisque l'on impose toujours finalement la conservation du flux normal), on *choisit* de reporter la totalité de la condition à la limite sur le terme $\frac{K}{C_v} \underline{\nabla} e$ et donc d'annuler le flux associé au terme $\frac{K}{C_v} \underline{\nabla}(\frac{1}{2}u^2 + \varepsilon_{sup})$ (en pratique, pour l'annuler, on se contente de ne pas l'ajouter au second membre de l'équation). Conformément à l'approche retenue dans code_saturne et rappelée précédemment, on déterminera donc une valeur

[2] On rappelle que, comme dans `condli`, α désigne $\lambda + C_p \frac{\mu_t}{\sigma_t}$ si f est la température, $\frac{\lambda}{C_p} + \frac{\mu_t}{\sigma_t}$ si f représente l'enthalpie. Le coefficient C représente C_p pour la température et vaut 1 pour l'enthalpie. La grandeur adimensionnelle f^+ est obtenue par application d'un principe de similitude en paroi : pour la température, elle dépend du nombre de Prandlt moléculaire, du nombre de Prandtl turbulent et de la distance adimensionnelle à la paroi y^+ dans la cellule de bord.

de bord *fictive* de l'énergie qui permette de reconstruire le flux diffusif total attendu à partir de la discrétisation du seul terme $\dfrac{K}{C_v}\underline{\nabla}\, e$.

Remarque : dans la version 1.2.0, on utilise $\dfrac{K}{C_v} = \left(\dfrac{\lambda}{C_v} + \dfrac{\mu_t}{\sigma_t}\right)$, à partir de 1.2.1, on utilise la valeur $\dfrac{K}{C_v} = \left(\dfrac{\lambda}{C_v} + \dfrac{C_p}{C_v}\dfrac{\mu_t}{\sigma_t}\right)$. On notera que le nombre de Prandtl turbulent σ_t est associé à la variable résolue et peut être fixé par l'utilisateur.

Condition de Neumann

La conservation du flux s'écrit :

$$\underbrace{h_{int}(e_{b,int} - e_{I'})}_{\phi_{int}} = \underbrace{\phi_{imp,ext}}_{\phi_{réel\ imposé}} \tag{V.E.33}$$

On a donc dans ce cas :

$$\left\{\begin{array}{lcl} A_b &=& \dfrac{1}{h_{int}}\phi_{imp,ext} \\ B_b &=& 1 \end{array}\right. \tag{V.E.34}$$

Condition de Dirichlet

On suppose que la condition de Dirichlet porte sur la température $T_{b,ext}$.

La conservation du flux s'écrit :

$$\underbrace{h_{int}(e_{b,int} - e_{I'})}_{\phi_{int}\ \text{(forme numérique du flux)}} = \underbrace{h_b(T_{b,ext} - T_{I'})}_{\phi_b\ \text{qui intègre l'effet de couche limite}} = \underbrace{h'_{imp,ext}(T_{imp,ext} - T_{b,ext})}_{\phi_{réel\ imposé}} \tag{V.E.35}$$

Avec pour les coefficients d'échange :

$$\left\{\begin{array}{lcl} h_{int} &=& \dfrac{K}{C_v\,\overline{I'F}} \\ h_b &=& \dfrac{\phi_b}{T_{b,ext} - T_{I'}} = \dfrac{\rho\, C_p\, u_k}{T^+_{I'}} \end{array}\right. \tag{V.E.36}$$

On tire $T_{b,ext}$ de la seconde partie de l'égalité (V.E.35) traduisant la conservation du flux :

$$T_{b,ext} = \frac{h'_{imp,ext}\, T_{imp,ext} + h_b\, T_{I'}}{h_b + h'_{imp,ext}} \tag{V.E.37}$$

En utilisant cette valeur et la première partie de l'équation de conservation du flux (V.E.35), on obtient :

$$e_{b,int} = \frac{h_b\, h'_{imp,ext}}{h_{int}\,(h_b + h'_{imp,ext})}\,(T_{imp,ext} - T_{I'}) + e_{I'} \tag{V.E.38}$$

On utilise alors $T_{I'} = \dfrac{1}{C_v}\left(e_{I'} - \dfrac{1}{2}u_i^2 - \varepsilon_{sup,i}\right)$ pour écrire (sans reconstruction pour la vitesse et ε_{sup}) :

$$e_{b,int} = \frac{-\frac{h_b\, h'_{imp,ext}}{C_v} + h_{int}\,(h_b + h'_{imp,ext})}{h_{int}\,(h_b + h'_{imp,ext})}\, e_{I'} + \frac{h_b\, h'_{imp,ext}}{h_{int}\,(h_b + h'_{imp,ext})}\left(T_{imp,ext} + \frac{\frac{1}{2}u_i^2 + \varepsilon_{sup,i}}{C_v}\right) \tag{V.E.39}$$

Et on a donc, avec $h'_r = \dfrac{h_{int}}{\frac{h_b}{C_v}}$:

$$e_{b,int} = \underbrace{\frac{h'_{imp,ext}}{C_v\, h_{int} + h'_r\, h'_{imp,ext}} \left(C_v\, T_{imp,ext} + \frac{1}{2} u_i^2 + \varepsilon_{sup,i} \right)}_{A_b} + \underbrace{\frac{C_v\, h_{int} + h'_{imp,ext}(h'_r - 1)}{C_v\, h_{int} + h'_r\, h'_{imp,ext}}}_{B_b} e_{I'} \tag{V.E.40}$$

Avec ces notations, h_b est le coefficient habituellement calculé pour la température.

Le coefficient $h'_{imp,ext}$ est le coefficient d'échange externe qui est imposé pour la température[3]. Pour obtenir l'équivalent dimensionnel de $h'_{imp,ext}$ pour l'énergie, il faut diviser sa valeur par C_v (ce qui ne fait pas de différence dans la majorité des cas, car il est habituellement pris infini).

Mise en œuvre

Introduction

Les conditions aux limites sont imposées par une suite de sous-programmes, dans la mesure où l'on a cherché à rester cohérent avec la structure standard de code_saturne.

Dans `ppprcl` (appelé par `precli`), on initialise les tableaux avant le calcul des conditions aux limites :

- `IZFPPP` (numéro de zone, inutilisé, fixé à zéro),
- `IA(IIFBRU)` (repérage des faces de bord pour lesquelles on applique un schéma de Rusanov : initialisé à zéro, on imposera la valeur 1 dans `cfrusb` pour les faces auxquelles on applique le schéma de Rusanov)
- `IA(IIFBET)` (repérage des faces de paroi à température ou à flux thermique imposé : initialisé à 0, on imposera la valeur 1 dans `cfxtcl` lorsque la température ou le flux est imposé),
- `RCODCL(*,*,1)` (initialisé à `-RINFIN` en prévision du traitement des sorties réentrantes pour lesquelles l'utilisateur aurait fourni une valeur à imposer en Dirichlet),
- flux convectifs de bord pour la quantité de mouvement et l'énergie (initialisés à zéro).

Les types de frontière (`ITYPFB`) et les valeurs nécessaires (`ICODCL`, `RCODCL`) sont imposés par l'utilisateur dans `uscfcl`.

On convertit ensuite ces données dans `condli` pour qu'elles soient directement utilisables lors du calcul des matrices et des seconds membres.

Pour cela, `cfxtcl` permet de réaliser le calcul des valeurs de bord et, pour certaines frontières, des flux convectifs. On fait appel, en particulier, à `uscfth` (utilisation de la thermodynamique) et à `cfrusb` (flux convectifs par le schéma de Rusanov). Lors de ces calculs, on utilise `COEFA` et `COEFB` comme tableaux de travail (transmission de valeurs à `uscfth` en particulier) afin de renseigner `ICODCL` et `RCODCL`. Après `cfxtcl`, le sous-programme `typecl` complète quelques valeurs par défaut de `ICODCL` et de `RCODCL`, en particulier pour les scalaires passifs.

Après `cfxtcl` et `typecl`, les tableaux `ICODCL` et `RCODCL` sont complets. Ils sont utilisés dans la suite de `condli` et en particulier dans `clptur` pour construire les tableaux `COEFA` et `COEFB` (pour l'énergie, on dispose de deux couples (`COEFA`, `COEFB`) afin de traiter les parois).

[3] Le coefficient $h'_{imp,ext}$ est utile pour les cas où l'on souhaite relaxer la condition à la limite : pour la température, cela correspond à imposer une valeur sur la face externe d'une paroi unidimensionnelle idéale, sans inertie, caractérisée par un simple coefficient d'échange.

On présente ci-après les points dont l'implantation diffère de l'approche standard. Il s'agit de l'utilisation d'un schéma de Rusanov pour le calcul des flux convectifs en entrée et sortie (hormis sortie supersonique) et du mode de calcul des flux diffusifs d'énergie en paroi. On insiste en particulier sur l'impact des conditions aux limites sur la construction des seconds membres de l'équation de la quantité de mouvement et de l'équation de l'énergie (`cfqdmv` et `cfener`).

Flux de Rusanov pour le calcul des flux convectifs en entrée et sortie

Le schéma de Rusanov est utilisé pour calculer des flux convectifs de bord (masse, quantité de mouvement et énergie) aux entrées et des sorties de type IESICF, ISOPCF, IERUCF, IEQHCF.

La gestion des conditions aux limites est différente de celle adoptée classiquement dans code_saturne, bien que l'on se soit efforcé de s'y conformer le mieux possible.

En volumes finis, il faut disposer de conditions aux limites pour trois utilisations principales au moins :

- imposer les flux de convection,
- imposer les flux de diffusion,
- calculer les gradients pour les reconstructions.

Dans l'approche standard de code_saturne, les conditions aux limites sont définies par variable et non pas par terme discret[4]. On dispose donc, *pour chaque variable*, d'une valeur de bord dont devront être déduits les flux de convection, les flux de diffusion et les gradients[5]. Ici, avec l'utilisation d'un schéma de Rusanov, dans lequel le flux convectif est traité dans son ensemble, il est impératif de disposer d'un moyen d'imposer directement sa valeur au bord[6].

Le flux convectif calculé par le schéma de Rusanov sera ajouté directement au second membre des équations de masse, de quantité de mouvement et d'énergie. Comme ce flux contient, outre la contribution des termes convectifs "usuels" ($\mathrm{div}\,(\underline{Q})$, $\mathrm{div}\,(\underline{u}\otimes\underline{Q})$ et $\mathrm{div}\,(\underline{Q}\,e)$), celle des termes en $\underline{\nabla}\,P$ (quantité de mouvement) et $\mathrm{div}\,(\underline{Q}\,\frac{P}{\rho})$ (énergie), il faut veiller à ne pas ajouter une seconde fois les termes de bord issus de $\underline{\nabla}\,P$ et de $\mathrm{div}\,(\underline{Q}\,\frac{P}{\rho})$ au second membre des équations de quantité de mouvement et d'énergie.

Pour la masse, le flux convectif calculé par le schéma de Rusanov définit simplement le flux de masse au bord (`PROPFB(IFAC,IPPROB(IFLUMA(ISCA(IENERG))))`).

Pour la quantité de mouvement, le flux convectif calculé par le schéma de Rusanov est stocké dans les tableaux `PROPFB(IFAC,IPPROB(IFBRHU))`, `PROPFB(IFAC,IPPROB(IFBRHV))` et `PROPFB(IFAC,IPPROB(IFBRHW))`. Il est ensuite ajouté au second membre de l'équation directement dans `cfqdmv` (boucle sur les faces de bord). Comme ce flux contient la contribution du terme convectif usuel $\underline{\mathrm{div}}\,(\underline{u}\otimes\underline{Q})$, il ne faut pas l'ajouter dans le sous-programme `cfbsc2`. De plus, le flux convectif calculé par le schéma de Rusanov contient la contribution du gradient de pression. Or, le gradient de pression est calculé dans `cfqdmv` au moyen de `grdcel` et ajouté au second membre sous forme de contribution volumique (par cellule) : il faut donc retirer la contribution des faces de bord auxquelles est appliqué le schéma de Rusanov, pour ne pas la compter deux fois (cette opération est réalisée dans `cfqdmv`).

Pour l'énergie, le flux convectif calculé par le schéma de Rusanov est stocké dans le tableau `PROPFB(IFAC,IPPROB(IFBENE)`. Pour les faces auxquelles n'est pas appliqué le schéma de Rusanov, on ajoute la contribution du terme

[4] Par exemple, pour un scalaire convecté et diffusé, on définit une valeur de bord unique *pour le scalaire* et non pas une valeur de bord pour le *flux convectif* et une valeur de bord pour le *flux diffusif*.

[5] Néanmoins, pour certaines variables comme la vitesse par exemple, code_saturne dispose de deux valeurs de bord (et non pas d'une seule) afin de pouvoir imposer de manière indépendante le gradient normal et le flux de diffusion.

[6] Il serait possible de calculer une valeur de bord fictive des variables d'état qui permette de retrouver le flux convectif calculé par le schéma de Rusanov, mais cette valeur ne permettrait pas d'obtenir un flux de diffusion et un gradient satisfaisants.

de transport de pression div $(\underline{Q}\,\frac{P}{\rho})$ au second membre de l'équation dans `cfener` et on complète le second membre dans `cfbsc2` avec la contribution du terme convectif usuel div $(\underline{Q}\,e)$. Pour les faces auxquelles est appliqué le schéma de Rusanov, on ajoute directement le flux de Rusanov au second membre de l'équation dans `cfener`, en lieu et place de la contribution du terme de transport de pression et l'on prend garde de ne pas comptabiliser une seconde fois le flux convectif usuel $\underline{\text{div}}\,(\underline{Q}\,e)$ dans le sous-programme `cfbsc2`.

C'est l'indicateur `IA(IIFBRU)` (renseigné dans `cfrusb`) qui permet, dans `cfbsc2`, `cfqdmv` et `cfener`, de repérer les faces de bord pour lesquelles on a calculé un flux convectif avec le schéma de Rusanov.

Flux diffusif d'énergie

Introduction

Une condition doit être fournie sur toutes les frontières pour le calcul du flux diffusif d'énergie.

Il n'y a pas lieu de s'étendre particulièrement sur le traitement de certaines frontières. Ainsi, aux entrées et sorties, on dispose d'une valeur de bord (issue de la résolution du problème de Riemann) que l'on utilise dans la formule discrète classique donnant le flux[7]. La situation est simple aux symétries également, où un flux nul est imposé.

Par contre, en paroi, les conditions de température ou de flux thermique imposé doivent être traitées avec plus d'attention, en particulier lorsqu'une couche limite turbulente est présente.

Coexistence de deux conditions de bord

Comme indiqué dans la partie "discrétisation", les conditions de température ou de flux conductif imposé en paroi se traduisent, pour le flux d'énergie, au travers du terme $\text{div}\left(\frac{K}{C_v}\,\underline{\nabla}\,e\right)$, en imposant une condition de flux nul sur le terme $-\text{div}\left(\frac{K}{C_v}\,\underline{\nabla}(\frac{1}{2}\,u^2+\varepsilon_{sup})\right)$. Les faces IFAC concernées sont repérées dans `cfxtcl` par l'indicateur `IA(IIFBET+IFAC-1) = 1` (qui vaut 0 sinon, initialisé dans `ppprcl`).

Sur ces faces, on calcule une valeur de bord de l'énergie, qui, introduite dans la formule générale de flux utilisée au bord dans code_saturne, permettra de retouver le flux souhaité. La valeur de bord est une simple valeur numérique sans signification physique et ne doit être utilisée que pour calculer le flux diffusif.

En plus de cette valeur de bord destinée à retrouver le flux diffusif, il est nécessaire de disposer d'une seconde valeur de bord de l'énergie afin de pouvoir en calculer le gradient.

Ainsi, comme pour la vitesse en $k-\varepsilon$, il est nécessaire de disposer pour l'énergie de deux couples de coefficients (`COEFA`,`COEFB`), correspondant à deux valeurs de bord distinctes, dont l'une est utilisée pour le calcul du flux diffusif spécifiquement.

Calcul des `COEFA` et `COEFB` pour les faces de paroi à température imposée

Les faces de paroi `IFAC` à température imposée sont identifées par l'utilisateur dans `uscfcl` au moyen de l'indicateur `ICODCL(IFAC,ISCA(ITEMPK))=5` (noter que ce tableau est associé à la température).

Dans `cfxtcl`, on impose alors `ICODCL(IFAC,ISCA(IENERG))=5` et on calcule la quantité $C_v\,T_{imp,ext}+\frac{1}{2}u_I^2+\varepsilon_{sup,I}$, que l'on stocke dans `RCODCL(IFAC,ISCA(IENERG),1)` (on ne reconstruit pas les valeurs de u^2 et ε_{sup} au bord, cf. §E).

[7] Les valeurs de u^2 et de ε_{sup} ne sont pas reconstruites pour le calcul du gradient au bord dans $\text{div}\left(\frac{K}{C_v}\,\underline{\nabla}(\frac{1}{2}\,u^2+\varepsilon_{sup})\right)$

À partir de ces valeurs de ICODCL et RCODCL, on renseigne ensuite dans clptur les tableaux de conditions aux limites permettant le calcul du flux : COEFA(*,ICLRTP(ISCA(IENERG),ICOEFF)) et COEFB(*,ICLRTP(ISCA(IENERG),ICOEFF)) (noter l'indicateur ICOEFF qui renvoie aux coefficients dédiés au flux diffusif).

Calcul des COEFA et COEFB pour les faces de paroi à flux thermique imposé

Les faces de paroi IFAC à flux thermique imposé sont identifées par l'utilisateur dans uscfcl au moyen de l'indicateur ICODCL(IFAC,ISCA(ITEMPK))=3 (noter que le tableau est associé à la température).

Dans cfxtcl, on impose alors ICODCL(IFAC,ISCA(IENERG))=3 et on transfère la valeur du flux de RCODCL(IFAC,ISCA(ITEMPK),3) à RCODCL(IFAC,ISCA(IENERG),3).

À partir de ces valeurs de ICODCL et RCODCL, on renseigne ensuite dans condli les tableaux de conditions aux limites permettant le calcul du flux, COEFA(*,ICLRTP(ISCA(IENERG),ICOEFF)) et COEFB(*,ICLRTP(ISCA(IENERG),ICOEFF)) (noter l'indicateur ICOEFF qui renvoie aux coefficients dédiés au flux diffusif).

Gradient de l'énergie en paroi à température ou à flux thermique imposé

Dans les deux cas (paroi à température ou à flux thermique imposé), on utilise les tableaux COEFA(*,ICLRTP(ISCA(II),IC
COEFB(*,ICLRTP(ISCA(II),ICOEF)) (noter le ICOEF) pour disposer d'une condition de flux nul pour l'énergie (avec II=IENERG) et pour la température (avec II=ITEMPK) si un calcul de gradient est requis.

Un gradient est en particulier utile pour les reconstructions de l'énergie sur maillage non orthogonal. Pour la température, il s'agit d'une précaution, au cas où l'utilisateur aurait besoin d'en calculer le gradient.

Autres frontières que les parois à température ou à flux thermique imposé

Pour les frontières qui ne sont pas des parois à température ou à flux thermique imposé, les conditions aux limites de l'énergie et de la température sont complétées classiquement dans condli selon les choix faits dans cfxtcl pour ICODCL et RCODCL.

En particulier, dans le cas de conditions de Dirichlet sur l'énergie (entrées, sorties), les deux jeux de conditions aux limites sont identiques (tableaux COEFA, COEFB avec ICOEFF et ICOEF).

Si un flux est imposé pour l'énergie totale (condition assez rare, l'utilisateur ne raisonnant pas, d'ordinaire, en énergie totale), on le stocke au moyen de COEFA(*,ICLRTP(ISCA(IENERG),ICOEFF)) et COEFB(*,ICLRTP(ISCA(IENERG),ICOEFF)) (tableaux associés au flux diffusif). Pour le gradient, une condition de flux nul est stockée dans COEFA(*,ICLRTP(ISCA(IENERG),ICOEF)) et COEFB(*,ICLRTP(ISCA(IENERG),ICOE
On peut remarquer que les deux jeux de conditions aux limites sont identiques pour les faces de symétrie.

Impact dans cfener

Lors de la construction des seconds membres, dans cfener, on utilise les conditions aux limites stockées dans les tableaux associés au flux diffusif COEFA(*,ICLRTP(ISCA(IENERG),ICOEFF)) et COEFB(*,ICLRTP(ISCA(IENERG),
pour le terme de flux diffusif $\text{div}\left(\frac{K}{C_v}\underline{\nabla} e\right)$ en prenant soin d'annuler la contribution de bord du terme $-\text{div}\left(\frac{K}{C_v}\underline{\nabla}(\frac{1}{2}u^2+\varepsilon_{sup})\right)$ sur les faces pour lesquelles cette condition prend les deux termes en compte, c'est-à-dire sur les faces pour lesquelles IA(IIFBET+IFAC-1) = 1.

Pour tous les autres termes qui requièrent une valeur de bord, on utilise les conditions aux limites que l'on a stockées au moyen des deux tableaux COEFA(*,ICLRTP(ISCA(IENERG),ICOEF)) et

`COEFB(*,ICLRTP(ISCA(IENERG),ICOEF))`. Ces conditions sont donc en particulier utilisées pour le calcul du gradient de l'énergie, lors des reconstructions sur maillage non orthogonal.

Points à traiter

Apporter un complément de test sur une cavité fermée sans vitesse et sans gravité, avec flux de bord ou température de bord imposée. Il semble que le transfert d'énergie *via* les termes de pression génère de fortes vitesses non physiques dans la première maille de paroi et que la conduction thermique ne parvienne pas à établir le profil de température recherché. Il est également possible que la condition de bord sur la pression génère une perturbation (une extrapolation pourrait se révéler indispensable).

Il pourrait être utile de généraliser à l'incompressible l'approche utilisée en compressible pour unifier simplement le traitement des sorties de type 9 et 10.

Il pourrait être utile d'étudier plus en détail l'influence de la non orthogonalité des mailles en sortie supersonique (pas de reconstruction, ce qui n'est pas consistant pour les flux de diffusion).

De même, il serait utile d'étudier l'influence de l'absence de reconstruction pour la vitesse et ε_{sup} dans la relation $T_{I'} = \dfrac{1}{C_v}\left(e_{I'} - \dfrac{1}{2}u_i^2 - \varepsilon_{sup,i}\right)$ utilisée pour les parois à température imposée.

Apporter un complément de documentation pour le couplage avec SYRTHES (conversion énergie température). Ce n'est pas une priorité.

Pour les thermodynamiques à γ variable, il sera nécessaire de modifier non seulement `uscfth` mais également `cfrusb` qui doit disposer de γ en argument.

Pour les thermodynamiques à C_v variable, il sera nécessaire de prendre en compte un terme en $\underline{\nabla} C_v$, issu des flux diffusifs, au second membre de l'équation de l'énergie (on pourra cependant remarquer qu'actuellement, en incompressible, on néglige le terme en $\underline{\nabla} C_p$ dans l'équation de l'enthalpie).

Part VI

Electric Arcs

A- cs_elec_model routine

On s'intéresse à la résolution des équations de la magnétohydrodynamique, constituées de la réunion des équations de l'aérothermodynamique et des équations de Maxwell.

On se place dans deux cadres d'utilisation bien spécifiques et distincts, qui permettront chacun de réaliser des simplifications : les études dites "d'arc électrique" (dans lesquelles sont prises en compte les forces de Laplace et l'effet Joule) et les études dites "Joule" (dans lesquelles seul l'effet Joule est pris en compte).

Les études d'arc électrique sont associées en grande partie, pour EDF, aux problématiques relatives aux transformateurs. Les études Joule sont plus sécifiquement liées aux phénomènes rencontrés dans les fours verriers.

Outre la prise en compte ou non des forces de Laplace, ces deux types d'études se différencient également par le mode de détermination de l'effet Joule (utilisation d'un potentiel complexe pour les études Joule faisant intervenir un courant alternatif non monophasé).

On décrit tout d'abord les équations résolues pour les études d'arc électrique. Les spécificités des études Joule seront abordées ensuite.

Pour l'arc électrique, les références [douce] et [delalondre] pourront compléter la présentation :

[delalondre] Delalondre, Clarisse : "Modélisation aérothermodynamique d'arcs électriques à forte intensité avec prise en compte du déséquilibre thermodynamique local et du transfert thermique à la cathode", Thèse de l'Université de Rouen, 1990

[douce] Douce, Alexandre : "Modélisation 3-D du chauffage d'un bain métallique par plasma d'arc transféré. Application à un réacteur axisymétrique", HE-26/99/027A, HE-44/99/043A, Thèse de l'Ecole Centrale Paris et EDF, 1999

See the programmers reference of the dedicated subroutine for further details.

Fonction

Notations

Variables utilisées

$\underline{A}$	potentiel vecteur réel	$kg\,m\,s^{-2}\,A^{-1}$
$\underline{B}$	champ magnétique	T (ou $kg\,s^{-2}\,A^{-1}$)
$\underline{D}$	déplacement électrique	$A\,s\,m^{-2}$
$\underline{E}$	champ électrique	$V\,m^{-1}$
E	énergie totale massique	$J\,kg^{-1}$ (ou $m^2\,s^{-2}$)
e	énergie interne massique	$J\,kg^{-1}$ (ou $m^2\,s^{-2}$)
e_c	énergie cinétique massique	$J\,kg^{-1}$ (ou $m^2\,s^{-2}$)
$\underline{H}$	excitation magnétique	$A\,m^{-1}$
h	enthalpie massique	$J\,kg^{-1}$ (ou $m^2\,s^{-2}$)
$\underline{j}$	densité de courant	$A\,m^{-2}$
P	pression	$kg\,m^{-1}\,s^{-2}$
P_R, P_I	potentiel scalaire réel, imaginaire	V (ou $kg\,m^2\,s^{-3}\,A^{-1}$)
$\underline{u}$	vitesse	$m\,s^{-1}$
ε	permittivité électrique	$F\,m^{-1}$ (ou $m^{-3}\,kg^{-1}\,s^4\,A^2$)
ε_0	permittivité électrique du vide	$8,854\,10^{-12}\ F\,m^{-1}$ (ou $m^{-3}\,kg^{-1}\,s^4\,A^2$)
μ	perméabilité électrique	$H\,m^{-1}$ (ou $m\,kg\,s^{-2}\,A^{-2}$)
μ_0	perméabilité électrique du vide	$4\,\pi\,10^{-7}\ H\,m^{-1}$ (ou $m\,kg\,s^{-2}\,A^{-2}$)
σ	conductivité électrique	$S\,m^{-1}$ (ou $m^{-3}\,kg^{-1}\,s^3\,A^2$)

Notations d'analyse vectorielle

On rappelle également la définition des notations employées[1] :

$$\left\{\begin{array}{lcl} \left[\underline{\underline{\mathrm{grad}}}\,\underline{a}\right]_{ij} & = & \partial_j a_i \\ \left[\mathrm{div}\,(\underline{\underline{\sigma}})\right]_i & = & \partial_j \sigma_{ij} \\ \left[\underline{a}\otimes\underline{b}\right]_{ij} & = & a_i\,b_j \end{array}\right.$$

et donc :

$$\left[\mathrm{div}\,(\underline{a}\otimes\underline{b})\right]_i = \partial_j(a_i\,b_j)$$

Arcs électriques

Introduction

Pour les études d'arc électrique, on calcule, à un pas de temps donné :

- la vitesse $\underline{u}$, la pression P, la variable énergétique enthalpie h (et les grandeurs turbulentes),
- un potentiel scalaire réel P_R (dont le gradient permet d'obtenir le champ électrique $\underline{E}$ et la densité de courant $\underline{j}$),
- un potentiel vecteur réel $\underline{A}$ (dont le rotationnel permet d'obtenir le champ magnétique $\underline{B}$).

Le champ électrique, la densité de courant et le champ magnétique sont utilisés pour calculer les termes sources d'effet Joule et les forces de Laplace qui interviennent respectivement dans l'équation de l'enthalpie et dans celle de la quantité de mouvement.

[1] en utilisant la convention de sommation d'Einstein.

Équations continues

Système d'équations

Les équations continues qui sont résolues sont les suivantes :

$$\left\{\begin{array}{l} \mathrm{div}\,(\rho\underline{u}) = 0 \\ \dfrac{\partial}{\partial t}(\rho\underline{u}) + \mathrm{div}\,(\rho\,\underline{u}\otimes\underline{u}) = \mathrm{div}\,(\underline{\underline{\sigma}}) + \underline{TS} + \underline{j}\times\underline{B} \\ \dfrac{\partial}{\partial t}(\rho h) + \mathrm{div}\,(\rho\,\underline{u}h) = \Phi_v + \mathrm{div}\left(\left(\dfrac{\lambda}{C_p} + \dfrac{\mu_t}{\sigma_t}\right)\underline{\nabla}h\right) + P_J \\ \mathrm{div}\,(\sigma\,\underline{\nabla}P_R) = 0 \\ \mathrm{div}\,(\underline{\underline{\mathrm{grad}}}\,\underline{A}) = -\mu_0\underline{j} \end{array}\right. \tag{VI.A.1}$$

avec les relations suivantes :

$$\left\{\begin{array}{l} P_J = \underline{j}\cdot\underline{E} \\ \underline{E} = -\underline{\nabla}P_R \\ \underline{j} = \sigma\underline{E} \end{array}\right. \tag{VI.A.2}$$

Équation de la masse

C'est l'équation résolue en standard par code_saturne (contrainte stationnaire). Elle n'a pas de traitement particulier dans le cadre du module présent. Un terme source de masse peut être pris en compte au second membre si l'utilisateur le souhaite. Pour simplifier l'exposé le terme source sera supposé nul ici, dans la mesure où il n'est pas spécifique au module électrique.

Équation de la quantité de mouvement

Elle présente, par rapport à l'équation standard résolue par code_saturne, un seul terme additionnel ($\underline{j}\times\underline{B}$) qui rend compte des forces de Laplace. Pour l'obtenir, on fait l'hypothèse que le milieu est électriquement neutre.

En effet, une charge q_i (Coulomb) animée d'une vitesse $\underline{v}_i$ subit, sous l'effet du champ électrique $\underline{E}$ ($V\,m^{-1}$) et du champ magnétique $\underline{B}$ (Tesla), une force $\underline{f}_i$ ($kg\,m\,s^{-2}$) :

$$\underline{f}_i = q_i\,(\underline{E} + \underline{v}_i\times\underline{B}) \tag{VI.A.3}$$

Avec n_i charges de type q_i par unité de volume et en sommant sur tous les types de charge i (électrons, ions, molécules ionisées...), on obtient la force de Laplace totale $\underline{F}_L$ ($kg\,m^{-2}\,s^{-2}$) subie par unité de volume :

$$\underline{F}_L = \sum_i \left[n_i\,q_i\,(\underline{E} + \underline{v}_i\times\underline{B})\right] \tag{VI.A.4}$$

On introduit alors la densité de courant $\underline{j}$ ($A\,m^{-2}$) :

$$\underline{j} = \sum_i n_i\,q_i\,\underline{v}_i \tag{VI.A.5}$$

Avec l'hypothèse que le milieu est électriquement neutre (à un niveau macroscopique) :

$$\sum_i n_i\,q_i = 0 \tag{VI.A.6}$$

la force totale $\underline{F}_L$ s'écrit alors :

$$\underline{F}_L = \underline{j}\times\underline{B} \tag{VI.A.7}$$

et on peut donc écrire l'équation de la quantité de mouvement :

$$\frac{\partial}{\partial t}(\rho\underline{u}) + \mathrm{div}\,(\rho\,\underline{u}\otimes\underline{u}) = \mathrm{div}\,(\underline{\underline{\sigma}}) + \underline{TS} + \underline{j}\times\underline{B} \tag{VI.A.8}$$

Équation de l'enthalpie

Elle est obtenue à partir de l'équation de l'énergie après plusieurs approximations utilisées en standard dans code_saturne et en prenant en compte le terme d'effet Joule lié à l'énergie électromagnétique.

Énergie électromagnétique

Avec les mêmes notations que précédemment mais sans qu'il soit besoin de supposer que le milieu est électriquement neutre, la puissance reçue par une charge q_i (particule douée de masse) de vitesse $\underline{v}_i$ (vitesse du porteur de charge, contenant éventuellement l'effet de la vitesse du fluide) sous l'effet du champ électrique $\underline{E}$ ($V\,m^{-1}$) et du champ magnétique $\underline{B}$ (T) est (sans sommation sur i) :

$$P_i = \underline{f}_i \cdot \underline{v}_i = q_i(\underline{E} + \underline{v}_i \times \underline{B}) \cdot \underline{v}_i = q_i \underline{v}_i \cdot \underline{E} \tag{VI.A.9}$$

Avec n_i charges par unité de volume et en sommant sur tous les types de charges i, on obtient la puissance totale par unité de volume :

$$P_J = \sum_i n_i\, q_i\, \underline{v}_i \cdot \underline{E} \tag{VI.A.10}$$

On introduit alors la densité de courant $\underline{j} = \sum_i n_i\, q_i\, \underline{v}_i$ (en $A\,m^{-2}$) et on obtient l'expression usuelle de la puissance électromagnétique dissipée par effet Joule (en $W\,m^{-3}$) :

$$P_J = \underline{j} \cdot \underline{E} \tag{VI.A.11}$$

Pour reformuler la puissance dissipée par effet Joule et obtenir une équation d'évolution de l'énergie électromagnétique, on utilise alors les équations de Maxwell. Les équations s'écrivent (lois d'Ampère et de Faraday) :

$$\left\{ \begin{array}{l} \dfrac{\partial \underline{D}}{\partial t} - \underline{\mathrm{curl}}\ \underline{H} = -\underline{j} \\ \dfrac{\partial \underline{B}}{\partial t} + \underline{\mathrm{curl}}\ \underline{E} = 0 \end{array} \right. \tag{VI.A.12}$$

On a donc :

$$P_J = \underline{j} \cdot \underline{E} = \left(-\frac{\partial \underline{D}}{\partial t} + \underline{\mathrm{curl}}\ \underline{H} \right) \cdot \underline{E} \tag{VI.A.13}$$

On utilise alors la relation suivante :

$$\underline{\mathrm{curl}}\ \underline{H} \cdot \underline{E} = \underline{H} \cdot \underline{\mathrm{curl}}\ \underline{E} - \mathrm{div}\,(\underline{E} \times \underline{H}) \tag{VI.A.14}$$

En effet, elle permet de faire apparaître un terme en divergence, caractéristique d'une redistribution spatiale :

$$\underline{j} \cdot \underline{E} \quad = \quad -\frac{\partial \underline{D}}{\partial t} \cdot \underline{E} + \underline{H} \cdot \underline{\mathrm{curl}}\ \underline{E} - \mathrm{div}\,(\underline{E} \times \underline{H}) \tag{VI.A.15}$$

Et en utilisant la loi de Faraday pour faire apparaître la dérivée en temps du champ magnétique :

$$\underline{j} \cdot \underline{E} \quad = \quad -\frac{\partial \underline{D}}{\partial t} \cdot \underline{E} - \underline{H} \cdot \frac{\partial \underline{B}}{\partial t} - \mathrm{div}\,(\underline{E} \times \underline{H}) \tag{VI.A.16}$$

Dans le cadre de code_saturne, on fait les hypothèses suivantes :

- la perméabilité ε et la permittivité μ sont constantes et uniformes (pour les gaz, en pratique, on utilise les propriétés du vide ε_0 et μ_0).
- on utilise $\underline{B} = \mu \underline{H}$ et $\underline{D} = \varepsilon \underline{E}$

On a alors :

$$\underline{j}\cdot\underline{E} \quad = \quad -\frac{\varepsilon_0}{2}\frac{\partial E^2}{\partial t}-\frac{1}{2\,\mu_0}\frac{\partial B^2}{\partial t}-\frac{1}{\mu_0}\mathrm{div}\,(\underline{E}\times\underline{B}) \tag{VI.A.17}$$

Énergie totale

On établit l'équation de l'énergie totale en prenant en compte la puissance des forces de Laplace et le terme d'effet Joule.

Sans prendre en compte l'énergie électromagnétique, le premier principe de la thermodynamique s'écrit d'ordinaire sous la forme suivante (pour un volume matériel suivi sur une unité de temps) :

$$d\int_V \rho E dV = \delta Q + \delta W \tag{VI.A.18}$$

Dans cette relation, E est l'énergie totale par unité de masse[2], soit $E = e + e_c$, e étant l'énergie interne massique et $e_c = \frac{1}{2}\,\underline{u}\cdot\underline{u}$ l'énergie cinétique massique. Le terme δQ représente la chaleur reçue au travers des frontières du domaine considéré tandis que le terme δW représente le travail des forces extérieures reçu par le système (y compris les forces dérivant d'une énergie potentielle).

Pour prendre en compte l'énergie électromagnétique, il suffit d'intégrer à la relation (VI.A.18) la puissance des forces de Laplace $(\underline{j}\times\underline{B})\cdot\underline{u}$ et le terme d'effet Joule $\underline{j}\cdot\underline{E}$ (transformation volumique d'énergie électromagnétique en énergie totale[3]). Dans cette relation, la vitesse $\underline{u}$ est la vitesse du fluide et non pas celle des porteurs de charge : elle n'est donc pas nécessairement coliéaire au vecteur $\underline{j}$ (par exemple, si le courant est dû à des électrons, la vitesse du fluide pourra être considérée comme décorrélée de la vitesse des porteurs de charges ; par contre, si le courant est dû à des ions, la vitesse du fluide pourra être plus directement influencée par le déplacement des porteurs de charge). Ainsi, le premier principe de la thermodynamique s'écrit :

$$d\int_V \rho E dV = \delta Q + \delta W + \underline{j}\cdot\underline{E}\,V\,dt + (\underline{j}\times\underline{B})\cdot\underline{u}\,V\,dt \tag{VI.A.19}$$

et l'équation locale pour l'énergie totale est alors :

$$\frac{\partial}{\partial t}(\rho E)+\mathrm{div}\,(\rho\,\underline{u}E) = \mathrm{div}\,(\underline{\underline{\sigma}}\,\underline{u})+\underline{TS}\cdot\underline{u}+(\underline{j}\times\underline{B})\cdot\underline{u}+\Phi_v-\mathrm{div}\,\underline{\Phi}_s+\underline{j}\cdot\underline{E} \tag{VI.A.20}$$

Le terme Φ_v représente les termes sources volumiques d'énergie autres que l'effet Joule (par exemple, il inclut le terme source de rayonnement, pour un milieu optiquement non transparent). Le terme $\underline{\Phi}_s$ est le flux d'énergie surfacique[4].

Enthalpie

Pour obtenir une équation sur l'enthalpie, qui est la variable énergétique choisie dans code_saturne dans le module électrique, on soustrait tout d'abord à l'équation de l'énergie totale celle de l'énergie cinétique pour obtenir une équation sur l'énergie interne.

L'équation de l'énergie cinétique (obtenue à partir de l'équation de la quantité de mouvement écrite sous forme non conservative) est :

$$\frac{\partial}{\partial t}(\rho e_c)+\mathrm{div}\,(\rho\,\underline{u}e_c) = \mathrm{div}\,(\underline{\underline{\sigma}}\,\underline{u})-\underline{\underline{\sigma}}:\left(\underline{\underline{\mathrm{grad}}}\,(\underline{u})\right)^t+\underline{TS}\cdot\underline{u}+(\underline{j}\times\underline{B})\cdot\underline{u} \tag{VI.A.21}$$

de sorte que, pour l'énergie interne, on a :

$$\frac{\partial}{\partial t}(\rho e)+\mathrm{div}\,(\rho\,\underline{u}e) = \underline{\underline{\sigma}}:\left(\underline{\underline{\mathrm{grad}}}\,(\underline{u})\right)^t+\Phi_v-\mathrm{div}\,\Phi_s+\underline{j}\cdot\underline{E} \tag{VI.A.22}$$

[2] Ne pas confondre le scalaire E, énergie totale, avec le vecteur $\underline{E}$, champ électrique.

[3] Le terme en divergence $-\frac{1}{\mu_0}\mathrm{div}\,(\underline{E}\times\underline{B})$ traduit une redistribution spatiale d'énergie électromagnétique : ce n'est donc pas un terme source pour l'énergie totale.

[4] Dans code_saturne, il est modélisé par une hypothèse de gradient et inclut également la "diffusion" turbulente.

et enfin, pour l'enthalpie $h = e + \frac{P}{\rho}$:

$$\frac{\partial}{\partial t}(\rho h) + \mathrm{div}\,(\rho\,\underline{u}h) = \underline{\underline{\sigma}} : \left(\underline{\underline{\mathrm{grad}}}\,(\underline{u})\right)^t + \Phi_v - \mathrm{div}\,\Phi_s + \underline{j}\cdot\underline{E} + \rho\frac{d}{dt}\left(\frac{P}{\rho}\right) \tag{VI.A.23}$$

En faisant apparaître la pression dans le tenseur des contraintes $\underline{\underline{\sigma}} = -P\underline{\underline{Id}} + \underline{\underline{\tau}}$, on peut écrire :

$$\frac{\partial}{\partial t}(\rho h) + \mathrm{div}\,(\rho\,\underline{u}h) = \underline{\underline{\tau}} : \left(\underline{\underline{\mathrm{grad}}}\,(\underline{u})\right)^t + \Phi_v - \mathrm{div}\,\Phi_s + \underline{j}\cdot\underline{E} + \frac{dP}{dt} \tag{VI.A.24}$$

Les approximations habituelles de code_saturne consistent alors à négliger le terme "d'échauffement" issu du tenseur des contraintes $\underline{\underline{\tau}} : \left(\underline{\underline{\mathrm{grad}}}\,(\underline{u})\right)^t$ et le terme en dérivée totale de la pression $\frac{dP}{dt}$, supposés faibles en comparaison des autres termes dans les applications traitées (exemple : terme d'effet Joule important, effets de compressibilité faibles...). De plus, le terme de flux est modélisé en suivant une hypothèse de gradient appliqué à l'enthalpie (et non pas à la température), soit donc :

$$\frac{\partial}{\partial t}(\rho h) + \mathrm{div}\,(\rho\,\underline{u}h) = \Phi_v - \mathrm{div}\left(\left(\frac{\lambda}{C_p} + \frac{\mu_t}{\sigma_t}\right)\underline{\nabla} h\right) + \underline{j}\cdot\underline{E} \tag{VI.A.25}$$

Équations électromagnétiques

Elles sont obtenues à partir des équations de Maxwell sous les hypothèses détaillées dans [douce], paragraphe 3.3.

Densité de courant

La relation liant la densité de courant et le champ électrique est issue de la loi d'Ohm que l'on suppose pouvoir utiliser sous la forme simplifiée suivante :

$$\underline{j} = \sigma\,\underline{E} \tag{VI.A.26}$$

Champ électrique

Le champ électrique s'obtient à partir d'un potentiel vecteur.

En effet, la loi de Faraday s'écrit :

$$\frac{\partial \underline{B}}{\partial t} + \underline{\mathrm{curl}}\ \underline{E} = 0 \tag{VI.A.27}$$

Avec une hypothèse quasi-stationnaire, il reste :

$$\underline{\mathrm{curl}}\ \underline{E} = 0 \tag{VI.A.28}$$

Il est donc possible de postuler l'existence d'un potentiel scalaire P_R tel que :

$$\underline{E} = -\underline{\nabla} P_R \tag{VI.A.29}$$

Potentiel scalaire

Le potentiel scalaire est solution d'une équation de Poisson.

En effet, la conservation de la charge q s'écrit :

$$\frac{\partial q}{\partial t} + \mathrm{div}\,(\underline{j}) = 0 \tag{VI.A.30}$$

Pour un milieu électriquement neutre (à l'échelle macroscopique), on a $\dfrac{\partial q}{\partial t} = 0$ soit donc :

$$\mathrm{div}\,(\underline{j}) = 0 \tag{VI.A.31}$$

C'est-à-dire, avec la loi d'Ohm (VI.A.26),

$$\mathrm{div}\,(\sigma\,\underline{E}) = 0 \tag{VI.A.32}$$

Avec (VI.A.29), on obtient donc une équation permettant de calculer le potentiel scalaire :

$$\mathrm{div}\,(\sigma\,\underline{\nabla} P_R) = 0 \tag{VI.A.33}$$

Champ magnétique

Le champ magnétique s'obtient à partir d'un potentiel vecteur.

En effet, la loi d'Ampère s'écrit :

$$\frac{\partial \underline{D}}{\partial t} - \underline{\mathrm{curl}}\ \underline{H} = -\underline{j} \tag{VI.A.34}$$

Sous les hypothèses indiquées précédemment, on écrit :

$$\varepsilon_0\,\mu_0\,\frac{\partial \underline{E}}{\partial t} - \underline{\mathrm{curl}}\ \underline{B} = -\mu_0 \underline{j} \tag{VI.A.35}$$

Avec une hypothèse quasi-stationnaire, il reste :

$$\underline{\mathrm{curl}}\ \underline{B} = \mu_0 \underline{j} \tag{VI.A.36}$$

De plus, la conservation du flux magnétique s'écrit[5] :

$$\mathrm{div}\ \underline{B} = 0 \tag{VI.A.37}$$

et on peut donc postuler l'existence d'un potentiel vecteur $\underline{A}$ tel que :

$$\underline{B} = \underline{\mathrm{curl}}\ \underline{A} \tag{VI.A.38}$$

Potentiel vecteur

Le potentiel vecteur est solution d'une équation de Poisson.

En prenant le rotationnel de (VI.A.38) et avec (VI.A.36), on obtient :

$$-\underline{\mathrm{curl}}\ (\underline{\mathrm{curl}}\ \underline{A}) = -\mu_0 \underline{j} \tag{VI.A.39}$$

Avec la relation donnant le Laplacien[6] d'un vecteur $\mathrm{div}\,(\underline{\underline{\mathrm{grad}}}\,\underline{a}) = \underline{\nabla}(\mathrm{div}\,\underline{a}) - \underline{\mathrm{curl}}\ (\underline{\mathrm{curl}}\ \underline{a})$ et sous la contrainte[7] que $\mathrm{div}\,\underline{A} = 0$, on obtient finalement une équation permettant de calculer le potentiel vecteur :

$$\mathrm{div}\ (\underline{\underline{\mathrm{grad}}}\,\underline{A}) = -\mu_0 \underline{j} \tag{VI.A.40}$$

Effet Joule

Introduction

Pour les études Joule, on calcule, à un pas de temps donné :

- la vitesse $\underline{u}$, la pression P, la variable énergétique enthalpie h (et les grandeurs turbulentes éventuelles),
- un potentiel scalaire réel P_R,

[5] Prendre la divergence de la loi de Faraday, avec $\mathrm{div}\,(\underline{\mathrm{curl}}\ \underline{E}) = 0$ (par analyse vectorielle) donne $\mathrm{div}\,\underline{B} = \mathrm{cst}$.

[6] En coordonnées cartésiennes, le Laplacien du vecteur $\underline{a}$ est le vecteur dont les composantes sont égales au Laplacien de chacune des composantes de $\underline{a}$.

[7] La condition $\mathrm{div}\,\underline{A} = 0$, dite "jauge de Coulomb", est nécessaire pour assurer l'unicité du potentiel vecteur.

- et, si le courant n'est ni continu, ni alternatif monophasé, un potentiel scalaire imaginaire P_I.

Le gradient du potentiel permet d'obtenir le champ électrique $\underline{E}$ et la densité de courant $\underline{j}$ (partie réelle et, éventuellement, partie imaginaire). Le champ électrique et la densité de courant sont utilisés pour calculer le terme source d'effet Joule qui intervient dans l'équation de l'enthalpie.

La puissance instantanée dissipée par effet Joule est égale au produit instantané $\underline{j} \cdot \underline{E}$. Dans le cas général, $\underline{j}$ et $\underline{E}$ sont des signaux alternatifs ($\underline{j} = \underline{|j|} cos(\omega\, t + \phi_j)$ et $\underline{E} = \underline{|E|} cos(\omega\, t + \phi_E)$) que l'on peut représenter par des complexes ($\underline{j} = \underline{|j|}\, e^{i\,(\omega\, t + \phi_j)}$ et $\underline{E} = \underline{|E|}\, e^{i\,(\omega\, t + \phi_E)}$). La puissance instantanée s'écrit alors $(\underline{|j|} \cdot \underline{|E|}) cos(\omega\, t + \phi_j) cos(\omega\, t + \phi_E)$.

- **En courant continu** ($\omega = \phi_j = \phi_E = 0$), la puissance se calcule donc simplement comme le produit scalaire $P_J = \underline{|j|} \cdot \underline{|E|}$. Le calcul de la puissance dissipée par effet Joule ne pose donc pas de problème particulier car les variables densité de courant et champ électrique résolues par code_saturne sont précisément $\underline{|j|}$ et $\underline{|E|}$ (les variables sont réelles).
- **En courant alternatif**, la periode du courant est beaucoup plus petite que les échelles de temps des phénomènes thermohydrauliques pris en compte. Il n'est donc pas utile de disposer de la puissance instantanée dissipée par effet Joule : la moyenne sur une période est suffisante et elle s'écrit[8] : $P_J = \frac{1}{2}(\underline{|j|} \cdot \underline{|E|}) cos(\phi_j - \phi_E)$. Cette formule peut également s'écrire de manière équivalente sous forme complexe : $P_J = \frac{1}{2}\underline{j} \cdot \underline{E}^*$, où $\underline{E}^*$ est le complexe conjugué de $\underline{E}$.
  - En courant alternatif monophasé ($\phi_j = \phi_E$), en particulier, la formule donnant la puissance se simplifie sous la forme $P_J = \frac{1}{2}(\underline{|j|} \cdot \underline{|E|})$, ou encore : $P_J = \frac{1}{\sqrt{2}}\underline{|j|} \cdot \frac{1}{\sqrt{2}}\underline{|E|}$. Il s'agit donc du produit des valeurs efficaces. Or, les variables résolues par code_saturne en courant alternatif monophasé sont précisément les valeurs efficaces (valeurs que l'on dénomme abusivement "valeurs réelles" dans le code source).
  - En courant alternatif non monophasé (triphasé, en particulier), la formule donnant la puissance est utilisée directement sous la forme $P_J = \frac{1}{2}\underline{j} \cdot \underline{E}^*$. On utilise pour la calculer les variables résolues qui sont la partie réelle et la partie imaginaire de $\underline{j}$ et $\underline{E}$.
- **En conclusion**,
  - en continu, les variables résolues $\underline{j}_{Res}$ et $\underline{E}_{Res}$ sont les variables réelles continues et la puissance se calcule par la formule suivante : $P_J = \underline{j}_{Res} \cdot \underline{E}_{Res}$
  - en alternatif monophasé, les variables résolues $\underline{j}_{Res}$ et $\underline{E}_{Res}$ sont les valeurs efficaces et la puissance se calcule par la formule suivante : $P_J = \underline{j}_{Res} \cdot \underline{E}_{Res}$
  - en alternatif non monophasé, les variables résolues $\underline{j}_{Res,R}$, $\underline{j}_{Res,I}$ et $\underline{E}_{Res,R}$, $\underline{E}_{Res,I}$ sont la partie réelle et la partie imaginaire de $\underline{j}$ et $\underline{E}$, et la puissance se calcule par la formule suivante : $P_J = \frac{1}{2}(\underline{j}_{Res,R} \cdot \underline{E}_{Res,R} - \underline{j}_{Res,I}\underline{E}_{Res,I})$

Le potentiel imaginaire n'est donc utilisé dans le code que lorsque le courant est alternatif et non monophasé. En particulier, le potentiel imaginaire n'est pas utilisé lorsque le courant est continu ou alternatif monophasé. En effet, la partie imaginaire n'est introduite en complément de la partie réelle que dans le cas où il est nécessaire de disposer de deux grandeurs pour définir le potentiel, c'est-à-dire lorsqu'il importe de connaître son amplitude et sa phase. En courant continu, on n'a naturellement besoin que d'une seule information. En alternatif monophasé, la valeur de la phase importe peu (on ne travaille pas sur des grandeurs électriques instantanées) : il suffit de connaître l'amplitude du potentiel et il est donc inutile d'introduire une variable imaginaire.

La variable dénommée "potentiel réel", P_R, représente une valeur efficace si le courant est monophasé et une partie réelle sinon. De manière plus explicite, pour un potentiel physique

[8] L'intégrale de $cos^2 x$ sur un intervalle de longueur $2\,\pi$ est π.

alternatif sinusoïdal Pp, de valeur maximale notée $Pp_{\max}$, de phase notée ϕ, la variable P_R représente $\frac{1}{\sqrt{2}}\,Pp_{\max}$ en monophasé et $Pp_{\max}\,cos\phi$ sinon. En courant continu, P_R représente naturellement le potentiel (réel, continu). **Il est donc indispensable de prêter une attention particulière aux valeurs de potentiel imposées aux limites** (facteur $\frac{1}{\sqrt{2}}$ ou $cos\phi$).

Équations continues

Système d'équations

Les équations continues qui sont résolues sont les suivantes :

$$\left\{\begin{array}{l} \mathrm{div}\,(\rho\underline{u}) = 0 \\ \dfrac{\partial}{\partial t}(\rho\underline{u}) + \mathrm{div}\,(\rho\,\underline{u}\otimes\underline{u}) = \mathrm{div}\,(\underline{\underline{\sigma}}) + \underline{TS} \\ \dfrac{\partial}{\partial t}(\rho h) + \mathrm{div}\,(\rho\,\underline{u}h) = \Phi_v + \mathrm{div}\left(\left(\dfrac{\lambda}{C_p} + \dfrac{\mu_t}{\sigma_t}\right)\underline{\nabla} h\right) + P_J \\ \mathrm{div}\,(\sigma\,\underline{\nabla} P_R) = 0 \\ \mathrm{div}\,(\sigma\,\underline{\nabla} P_I) = 0 \quad \text{en alternatif non monophasé uniquement} \end{array}\right. \tag{VI.A.41}$$

avec, en continu ou alternatif monophasé :

$$\left\{\begin{array}{l} P_J = \underline{j}\cdot\underline{E} \\ \underline{E} = -\underline{\nabla} P_R \\ \underline{j} = \sigma\underline{E} \end{array}\right. \tag{VI.A.42}$$

et, en alternatif non monophasé (avec $i^2 = -1$) :

$$\left\{\begin{array}{l} P_J = \dfrac{1}{2}\,\underline{j}\cdot\underline{E}^* \\ \underline{E} = -\underline{\nabla}(P_R + i\,P_I) \\ \underline{j} = \sigma\underline{E} \end{array}\right. \tag{VI.A.43}$$

Équation de la masse

C'est l'équation résolue en standard par code_saturne (contrainte stationnaire d'incompressibilité). Elle n'a pas de traitement particulier dans le cadre du module présent. Un terme source de masse peut être pris en compte au second membre si l'utilisateur le souhaite. Pour simplifier l'exposé, le terme source sera supposé nul ici, dans la mesure où il n'est pas spécifique au module électrique.

Équation de la quantité de mouvement

C'est l'équation résolue en standard par code_saturne (les forces de Laplace ($\underline{j}\times\underline{B}$) sont supposées négligeables).

Équation de l'enthalpie

On l'établit comme dans le cas des arcs électriques[9] à partir de l'équation de l'énergie après plusieurs approximations utilisées en standard dans code_saturne et en prenant en compte le terme d'effet Joule lié à l'énergie électromagnétique.

Par rapport à l'équation utilisée pour les études d'arc électrique, seule l'expression de l'effet Joule

[9] À ceci près que la puissance des forces de Laplace n'apparaît pas du tout, au lieu de disparaître lorsque l'on soustrait l'équation de l'énergie cinétique à celle de l'énergie totale.

diffère lorsque le courant est alternatif non monophasé.

Équations électromagnétiques

Elles sont obtenues comme indiqué dans la partie relative aux arcs électriques, mais on ne conserve que les relations associées à la densité de courant, au champ électrique et au potentiel dont il dérive.

Discrétisation

La discrétisation des équations ne pose pas de problème particulier (ajout de termes sources explicites pour l'effet Joule et les forces de Laplace, équations de Poisson pour la détermination des potentiels).

Un point sur les conditions aux limites doit cependant être fait ici, en particulier pour préciser la méthode de recalage automatique des potentiels.

Arcs électriques

Conditions aux limites

Seules les conditions aux limites pour les potentiels sont à préciser.

Les conditions aux limites sur le potentiel scalaire sont des conditions de Neumann homogènes sur toutes les frontières hormis à la cathode et à l'anode. À la cathode, on impose une condition de Dirichlet homogène (potentiel nul par convention). À l'anode, on impose une condition de Dirichlet permettant de fixer la différence de potentiel souhaitée entre l'anode et la cathode. L'utilisateur peut fixer le potentiel de l'anode directement ou demander qu'un recalage automatique du potentiel soit effectué pour atteindre une intensité de courant prédéterminée.

Lorsque le recalage automatique est demandé (`IELCOR`=1), l'utilisateur doit fixer la valeur cible de l'intensité, `COUIMP`, (A) et une valeur élevée de départ de la différence de potentiel entre l'anode et la cathode[10], `DPOT`, (V). Le recalage est effectué en fin de pas temps et permet de disposer, pour le pas de temps suivant, de valeurs recalées des forces de Laplace et de l'effet Joule.

- Pour effectuer le recalage, code_saturne détermine l'intégrale de l'effet Joule estimé sur le domaine (en W) et en compare la valeur au produit de l'intensité `COUIMP` par la différence de potentiel[11] `DPOT`. Un coefficient multiplicatif de recalage `COEPOT` en est déduit (pour éviter des variations trop brusques, on s'assure qu'il reste borné).
- On multiplie alors par `COEPOT` la différence de potentiel entre l'anode et la cathode, `DPOT`, et le vecteur $\underline{j}$. L'effet Joule, produit de $\underline{j}$ par $\underline{E}$, est multiplié par le carré de `COEPOT`. Pour assurer la cohérence du post-traitement des variables, le potentiel vecteur et le potentiel scalaire sont également multipliés par `COEPOT`.
- Le champ électrique n'étant pas explicitement stocké, on ne le recale pas. Le potentiel vecteur et les forces de Laplace seront déduits de la densité de courant et intégreront donc naturellement le recalage.

Les conditions aux limites sur le potentiel vecteur sont des conditions de Neumann homogène sur toutes les frontières hormis sur une zone de bord arbitrairement choisie (paroi par exemple) pour laquelle une condition de Dirichlet est utilisée afin que le système soit inversible (la valeur imposée est la valeur du potentiel vecteur calculée au pas de temps précédent).

[10] Plus précisément, l'utilisateur doit imposer un potentiel nul en cathode et le potentiel `DPOT` à l'anode, en utilisant explicitement, dans le sous-programme utilisateur `cs_user_boundary_conditions`, la variable `DPOT` qui sera automatiquement recalée au cours du calcul.

[11] `DPOT` est la différence de potentiel imposée entre l'anode et la cathode au pas de temps qui s'achève. `DPOT` a conditionné le champ électrique et la densité de courant utilisés pour le calcul de l'effet Joule.

Effet Joule

Conditions aux limites

Seules les conditions aux limites pour les potentiels sont à préciser.

Les conditions aux limites sur le potentiel scalaire sont à préciser au cas par cas selon la configuration des électrodes. Ainsi, on dispose classiquement de conditions de Neumann homogènes ou de Dirichlet (potentiel imposé). On peut également avoir besoin d'imposer des conditions d'antisymétrie (en utilisant des conditions de Dirichlet homogènes par exemple). L'utilisateur peut également souhaiter qu'un recalage automatique du potentiel soit effectué pour atteindre une valeur prédéterminée de la puissance dissipée par effet Joule.

Lorsque le recalage automatique est demandé (`IELCOR`=1), l'utilisateur doit fixer la valeur cible de la puissance dissipée dans le domaine, `PUISIM`, ($V.A$). Il doit en outre, sur les frontières où il souhaite que le potentiel (réel ou complexe) s'adapte automatiquement, fournir en condition à la limite une valeur initiale du potentiel et la multiplier par la variable `COEJOU` qui sera automatiquement recalée au cours du calcul (`COEJOU` vaut 1 au premier pas de temps). Le recalage est effectué en fin de pas temps et permet de disposer, pour le pas de temps suivant, d'une valeur recalée de l'effet Joule.

- Pour effectuer le recalage, code_saturne détermine l'intégrale de l'effet Joule estimé sur le domaine (en W) et en compare la valeur à la puissance cible. Un coefficient multiplicatif de recalage `COEPOT` en est déduit (pour éviter des variations trop brusques, on s'assure qu'il reste borné entre 0,75 et 1,5).
- On multiplie alors par `COEPOT` le facteur multiplicatif `COEJOU` utilisé pour les conditions aux limites. La puissance dissipée par effet Joule est multipliée par le carré de `COEPOT`. Pour assurer la cohérence du post-traitement des variables, le potentiel est également multiplié par `COEPOT`.
- Le champ électrique n'étant pas explicitement stocké, on ne le recale pas.

On notera que la variable `DPOT` est également recalée et qu'elle peut donc être utilisée si besoin pour imposer les conditions aux limites.

Mise en œuvre

Introduction

Le module électrique est une "physique particulière" activée lorsque les mots-clés `IPPMOD(IELARC)` (arc électrique) ou `IPPMOD(IELJOU)` (Joule) sont strictement positifs. Les développements concernant la conduction ionique (mot-clé `IPPMOD(IELION)`) ont été prévus dans le code mais restent à réaliser. Pour l'arc électrique, dans la version actuelle de code_saturne, seule est opérationnelle l'option `IPPMOD(IELARC)`=2 : la version 2D axisymétrique qui permettrait de s'affranchir du potentiel vecteur (option `IPPMOD(IELARC)`=1) n'est pas activable. Pour l'effet Joule, lorsqu'il n'est pas utile d'introduire un potentiel scalaire complexe (en courant continu ou alternatif monophasé), on utilise `IPPMOD(IELJOU)`=1. Lorsqu'un potentiel scalaire complexe est indispensable (courant alternatif triphasé, par exemple), on utilise `IPPMOD(IELJOU)`=2.

Dans ce qui suit, on précise les inconnues et les propriétés principales utilisées dans le module. On fournit également un arbre d'appel simplifié des sous-programmes du module (initialisation avec `initi1` puis `inivar` et boucle en temps avec `tridim`). Les sous-programmes marqués d'un astérisque sont détaillés ensuite.

Inconnues et propriétés

Les développements ont été réalisés pour une unique phase (NPHAS=1).

Les NSCAPP inconnues scalaires associées à la physique particulière sont définies dans cs_elec_add_variable_fields dans l'ordre suivant (en particulier afin de limiter le stockage en mémoire lors de la résolution séquentielle des scalaires par scalai) :

- l'enthalpie RTP(*,ISCA(IHM)),
- un potentiel scalaire réel RTP(*,ISCA(IPOTR)),
- un potentiel scalaire imaginaire RTP(*,ISCA(IPOTI)) *ssi* IPPMOD(IELJOU)=2 (études Joule en courant alternatif non monophasé),
- les trois composantes d'un potentiel vecteur réel RTP(*,ISCA(IPOTVA(i))) (avec i variant de 1 à 3) *ssi* IPPMOD(IELARC)=2 (arc électrique),
- NGAZG-1 fractions massiques RTP(*,ISCA(IYCOEL(j))) (avec j variant de 1 à NGAZG-1) pour un fluide à NGAZG constituants (avec NGAZG strictement supérieur à 1). En arc électrique, la composition est fournie dans le fichier de données dp_ELE. La fraction massique du dernier constituant n'est pas stockée en mémoire. Elle est déterminée chaque fois que nécessaire en calculant le complément à l'unité des autres fractions massiques (et, en particulier, lorsque cs_elec_convert_h_t est utilisé pour le calcul des propriétés physiques).

Outre les propriétés associées en standard aux variables scalaires identifiées ci-dessus, le tableau PROPCE contient également :

- la température, PROPCE(*,IPPROC(ITEMP)). En théorie, on pourrait éviter de stocker cette variable, mais l'utilisateur est presque toujours intéressé par sa valeur en post-traitement et les propriétés physiques sont souvent données par des lois qui en dépendent explicitement. Son unité (Kelvin ou Celsius) dépend des tables enthalpie-température fournies par l'utilisateur.
- la puissance électromagnétique dissipée par effet Joule, PROPCE(*,IPPROC(IEFJOU)) (terme source positif pour l'enthalpie),
- les trois composantes des forces de Laplace, PROPCE(*,IPPROC(ILAPLA(i))) (avec i variant de 1 à 3) en arc électrique (IPPMOD(IELARC)=2).

La conductivité électrique est *a priori* variable et conservée dans le tableau de propriétés aux cellules PROPCE(*,IPPROC(IVISLS(IPOTR))). Elle intervient dans l'équation de Poisson portant sur le potentiel scalaire. Lorsque le potentiel scalaire a une partie imaginaire, la conductivité n'est pas dupliquée : les entiers IPPROC(IVISLS(IPOTI)) et IPPROC(IVISLS(IPOTR)) pointent sur la même case du tableau PROPCE. La conductivité associée au potentiel vecteur est uniforme et de valeur unité (VISLS0(IPOTVA(i))=1.D0 avec i variant de 1 à 3).

Le champ électrique, la densité de courant et le champ magnétique ne sont stockés que de manière temporaire (voir cs_compute_electric_field).

Arbre d'appel simplifié

usini1				Initialisation d
				des variables
	usppmo			Définition du
	varpos			Positionnemer
		pplecd		Branchement
				de données
			cs_electrical_properties_read*	Lecture du fic
		ppvarp		Branchement
				des inconnues
			cs_elec_add_variable_fields*	Positionnemer
				massiques)
		ppprop		Branchement
				des propriétés
			cs_elec_add_property_fields*	Positionnemer
				Laplace)
ppini1				Branchement
				mots-clés spéc
	cs_electrical_model_specific_initialization			Initialisation d
	cs_user_parameters			Initialisation d

Table A.1: Sous-programme `initi1` : initialisation des mots-clés et positionnement des variables

ppiniv				Branchement des phy
				variables
	cs_electrical_model_initialize*			Initialisation des varia
		cs_elec_convert_h_t*		Transformation temp
				par interpolation sur
				électrique uniquement
		cs_user_initialization		Initialisation des varia
			cs_elec_convert_h_t*	Transformation temp
				par interpolation sur
				électrique uniquement

Table A.2: Sous-programme `inivar` : initialisation des variables

phyvar			Calcul des propriétés physiques variables
	ppphyv		Branchement des physiques particulières pr priétés physiques variables
	cs_elec_physical_properties		Calcul des propriétés physiques variabl électrique. En arc électrique, les propriétés terpolation à partir des tables fournies dans dp_ELE
		cs_elec_convert_h_t*	Transformation température-enthalpie et e par interpolation sur la base du fichier de électrique uniquement)
		cs_user_physical_properties	Calcul par l'utilisateur des propriétés phy le module électrique. Pour les études Jou propriétés doivent être fournies ici sous form sont disponibles)
		cs_user_physical_properties	Transformation température-enthalpie et e fournie par l'utilisateur (plus spécifiquement pour lesquelles on ne dispose pas d'un fichie duquel réaliser des interpolations avec cs_e

Table A.3: Sous-programme tridim : partie 1 (propriétés physiques)

Précisions

• cs_electrical_properties_read

Ce sous-programme réalise la lecture du fichier de données spécifique aux arcs électriques. On donne ci-dessous, à titre d'exemple, l'entête explicative et deux lignes de données d'un fichier type. Ces valeurs sont interpolées chaque fois que nécessaire par cs_elec_convert_h_t pour déterminer les propriétés physiques du fluide à une température (une enthalpie) donnée.

```
# Nb d'especes NGAZG et Nb de points NPO (le fichier contient NGAZG blocs de NPO lignes chacun)
# NGAZG NPO
   1   238
#
#  Proprietes
#      T           H          ROEL       CPEL          SIGEL        VISEL        XLABEL        XKABEL
#  Temperature  Enthalpie   Masse vol. Chaleur       Conductivite  Viscosite   Conductivite   Coefficient
#                           volumique  massique       electrique   dynamique     thermique   d'absorption
#       K         J/kg         kg/m3     J/(kg K)       Ohm/m        kg/(m s)     W/(m K)         -
#
   300.00       14000.       1.6225       520.33      0.13214E-03  0.34224E-04  0.26712E-01   0.0000
   400.00       65800.       1.2169       520.33      0.13214E-03  0.34224E-04  0.26712E-01   0.0000
```

• cs_elec_add_variable_fields

Ce sous-programme permet de positionner les inconnues de calcul listées précédemment. On y précise également que la chaleur massique à pression constante est variable, ainsi que la conductivité de tous les scalaires associés au module électrique, hormis la conductivité de l'éventuel potentiel vecteur (celle-ci est uniforme et de valeur unité).

• cs_elec_add_property_fields

C'est dans ce sous-programme que sont positionnées les propriétés stockées dans le tableau PROPCE, et en particulier la température, l'effet Joule et les forces de Laplace.

ppclim		Branchement des physiques particulières pour les conditi limites
	cs_user_boundary_conditions	Intervention de l'utilisateur pour les conditions aux lim lieu et place de usclim, même pour les variables qui ne spécifiques au module électrique). Si un recalage automat potentiels est demandé (IELCOR=1), il doit être pris en co par le biais des variables DPOT ou COEJOU (voir la descrip conditions aux limites).
cs_solve_navier_stokes		Résolution des équations de Navier-Stokes
	cs_velocity_prediction	Prédiction de la vitesse : prise en compte des forces de calculées dans cs_compute_electric_field au pas d précédent
‘‘turb’’		Turbulence : résolution des équations pour les modèles né des équations de convection-diffusion
scalai*		Résolution des équations portant sur les scalaires asso physiques particulières et des scalaires “utilisateur”
	cs_solve_equation_scalar	Résolution successive de l'enthalpie, du potentiel scal et, si IPPMOD(IELJOU)=2, de la partie imaginaire du p scalaire (appels successifs à cs_solve_equation_scalar pelle cs_elec_source_terms pour le calcul du terme d'ef au second membre de l'équation de l'enthalpie)
	cs_compute_electric_field*	Calcul du champ électrique, de la densité de courant et Joule (premier de deux appels au cours du pas de temps
	cs_user_electric_scaling*	Recalage automatique éventuel de la densité de courant, Joule, des potentiels et des coefficients DPOT et COEJOU. calage, s'il a été demandé par l'utilisateur (IELCOR=1), est à partir du deuxième pas de temps.
	cs_solve_equation_scalar	Résolution successive, si IPPMOD(IELARC)=2, des tro posantes du potentiel vecteur. On procède par appels s à cs_solve_equation_scalar qui appelle cs_elec_sourc pour le calcul du second membre de l'équation de Poisson sur chaque composante du potentiel.
	cs_solve_equation_scalar	Résolution successive des NGAZG-1 fractions massiqu actérisant la composition du fluide, s'il est multiconstitu procède par appels successifs à cs_solve_equation_scal
	cs_compute_electric_field*	En arc électrique, calcul du champ magnétique et des tr posantes des forces de Laplace (deuxième et dernier appel du pas de temps courant)
	cs_solve_equation_scalar	Résolution des scalaires “utilisateur”

Table A.4: Sous-programme tridim : partie 2 (conditions aux limites, Navier-Stokes, turbulence et scalaires)

• `cs_elec_fields_initialize`

Ce sous-programme permet de réaliser les initialisations par défaut spécifiques au module.

En particulier, en $k-\varepsilon$, les deux variables turbulentes sont initialisées à 10^{-10} (choix historique arbitraire, mais réputé, lors de tests non référencés, permettre le démarrage de certains calculs qui échouaient avec une initialisation classique).

Les potentiels sont initialisés à zéro, de même que l'effet Joule. En arc électrique, les forces de Laplace sont initialisées à zéro.

Le fluide est supposé monoconstituant (seule est présente la première espèce).

En arc électrique, l'enthalpie est initialisée à la valeur de l'enthalpie du mélange supposé monoconstituant à la température `T0` donnée dans `usini1`. En effet Joule, l'enthalpie est initialisée à zéro (mais l'utilisateur peut fournir une valeur différente dans `cs_user_physical_properties`).

• `cs_elec_convert_h_t`

Ce sous-programme permet de réaliser (en arc électrique) les interpolations nécessaires à la détermination des propriétés physiques du fluide, à partir des tables fournies dans le fichier de données `dp_ELE`.

On notera en particulier que ce sous-programme prend en argument le tableau `YESP(NESP)` qui représente la fraction massique des `NGAZG` constituants du fluide. Dans le code, on ne résout que la fraction massique des `NGAZG`-1 premiers constituants. Avant chaque appel à `cs_elec_convert_h_t`, la fraction massique du dernier constituant doit être calculée comme le complément à l'unité des autres fractions massiques.

• `scalai`, `cs_compute_electric_field`, `cs_user_electric_scaling`

Le sous-programme `scalai` permet de calculer, dans l'ordre souhaité, les `NSCAPP` scalaires "physique particulière" associés au module électrique, puis de calculer les grandeurs intermédiaires nécessaires et enfin de réaliser les opérations qui permettent d'assurer le recalage automatique des potentiels, lorsqu'il est requis par l'utilisateur (*i.e.* si `IELCOR=1`).

Les `NSCAPP` scalaires "physique particulière" sont calculés successivement par un appel à `cs_solve_equation_scalar` placé dans une boucle portant sur les `NSCAPP` scalaires. L'algorithme tire profit de l'ordre spécifique dans lequel ils sont définis et donc résolus (dans l'ordre : enthalpie, potentiel scalaire, potentiel vecteur, fractions massiques).

Pour éviter des variations trop brutales en début de calcul, le terme source d'effet Joule n'est pris en compte dans l'équation de l'enthalpie qu'à partir du troisième pas de temps.

Après la résolution de l'enthalpie et du potentiel scalaire (réel ou complexe), le sous-programme `cs_compute_electric_field` permet de calculer les trois composantes du champ électrique (que l'on stocke dans des tableaux de travail), puis la densité de courant et enfin l'effet Joule, que l'on conserve dans le tableau `PROPCE(*,IPPROC(IEFJOU))` pour le pas temps suivant (après recalage éventuel dans `cs_user_electric_scaling` comme indiqué ci-après). Lorsque `IPPMOD(IELJOU)=2`, l'apport de la partie imaginaire est pris en compte pour le calcul de l'effet Joule. Lorsque `IPPMOD(IELARC)=2` (arc électrique), le vecteur densité de courant est conservé dans `PROPCE`, en lieu et place des forces de Laplace `PROPCE(*,IPPROC(ILAPLA(i)))` : il est utilisé pour le calcul du potentiel vecteur dans le second appel à `cs_compute_electric_field`, après recalage éventuel par `cs_user_electric_scaling` (en effet, il n'est plus nécessaire de conserver les forces de Laplace à ce stade puisque la seule équation dans laquelle elles interviennent est l'équation de la quantité de mouvement et qu'elle a déjà été résolue).

À la suite de `cs_compute_electric_field`, le sous-programme `cs_user_electric_scaling` effectue le

recalage permettant d'adapter automatiquement les conditions aux limites portant sur les potentiels, si l'utilisateur l'a demandé (*i.e.* si `IELCOR=1`). On se reportera au paragraphe relatif aux conditions aux limites. On précise ici que le coefficient de recalage `COEPOT` permet d'adapter l'effet Joule `PROPCE(*,IPPROC(IEFJOU))` et la différence de potentiel `DPOT` (utile pour les conditions aux limites portant sur le potentiel scalaire au pas de temps suivant[12]). Pour les cas d'arc électrique, `COEPOT` permet également de recaler le vecteur densité de courant que l'on vient de stocker temporairement dans `PROPCE(*,IPPROC(ILAPLA(i)))` et qui va servir immédiatement à calculer le potentiel vecteur. Pour les cas Joule, on recale en outre le coefficient `COEJOU` (utile pour les conditions aux limites portant sur le potentiel scalaire au pas de temps suivant).

Pour les cas d'arc électrique (`IPPMOD(IELARC)=2`), après `cs_compute_electric_field` et `cs_user_electric_scaling`, la résolution séquentielle des inconnues scalaires se poursuit dans `scalai` avec le calcul des trois composantes du potentiel vecteur. Le second membre de l'équation de Poisson considérée dépend de la densité de courant qui, dans `cs_compute_electric_field`, a été temporairement stockée dans le tableau `PROPCE(*,IPPROC(ILAPLA(i)))` et qui, dans `cs_user_electric_scaling`, vient d'être recalée si `IELCOR=1`. Les valeurs du potentiel vecteur obtenues intègrent donc naturellement l'éventuel recalage.

Pour les cas d'arc électrique (`IPPMOD(IELARC)=2`), un second appel à `cs_compute_electric_field` permet alors de calculer le champ magnétique que l'on stocke dans des tableaux de travail et les forces de Laplace que l'on stocke dans `PROPCE(*,IPPROC(ILAPLA(i)))` pour le pas de temps suivant (la densité de courant, que l'on avait temporairement conservée dans ce tableau, ne servait qu'à calculer le second membre de l'équation de Poisson portant sur le potentiel vecteur : il n'est donc plus nécessaire de la conserver).

La résolution séquentielle des inconnues scalaires spécifiques au module se poursuit dans `scalai`, avec le calcul des `NGAZG`-1 fractions massiques permettant de définir la composition du fluide.

Pour terminer, `scalai` permet la résolution des scalaires "utilisateurs" (appel à `cs_solve_equation_scalar` dans une boucle portant sur les `NSCAUS` scalaires utilisateur).

On peut remarquer pour finir que les termes sources des équations de la quantité de mouvement (forces de Laplace) et de l'enthalpie (effet Joule) sont disponibles à la fin du pas de temps n pour une utilisation au pas de temps $n+1$ (de ce fait, pour permettre les reprises de calcul, ces termes sources sont stockés dans le fichier suite auxiliaire, ainsi que `DPOT` et `COEJOU`).

[12] *A priori*, `DPOT` n'est pas nécessaire pour les cas Joule.

Points à traiter

• **Mobilité ionique**
Le module est à développer.

• **Conditions aux limite en Joule**
La prise en compte de conditions aux limites couplées entre électrodes reste à faire.

• **Compressible en arc électrique**
Les développements du module compressible de code_saturne doivent être rendus compatibles avec le module arc électrique.

Part VII

Mesh Handling

Appendix B

Mesh Algorithms

In this chapter, we will describe algorithms used for several operations done by code_saturne.

Geometric Quantities

See the programmers reference of the dedicated subroutine for further details.

Normals and Face Centers

To calculate face normals, we take care to use an algorithm that is correct for any planar simple polygon, including non-convex cases. The principle is as follows: take an arbitrary point P_a in the same plane as the polygon, then compute the sum of the vector normals of triangles $\{P_a, P_i, P_{i+1}\}$, where $\{P_1, P_2, ..., P_i, ..., P_n\}$ are the polygon vertices and $P_{n+1} \equiv P_0$. As shown on figure VII.B.1, some normals have a "positive" contribution while others have a "negative" contribution (taking into account the ordering of the polygon's vertices). The length of the final normal obtained is equal to the polygon's surface.

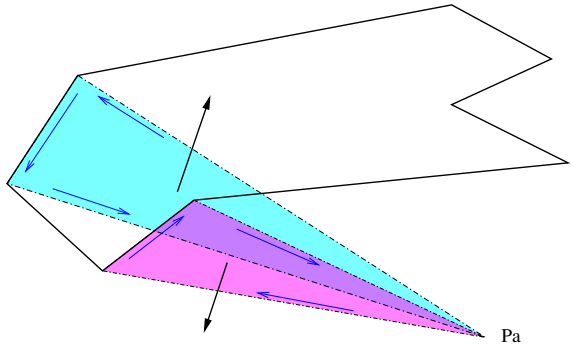


Figure VII.B.1: Face normals calculation principle

In our implementation, we take the "arbitrary" P_a point as the center of the polygon's vertices, so as to limit precision problems due to truncation errors and to ensure that the chosen point is on the polygon's plane.

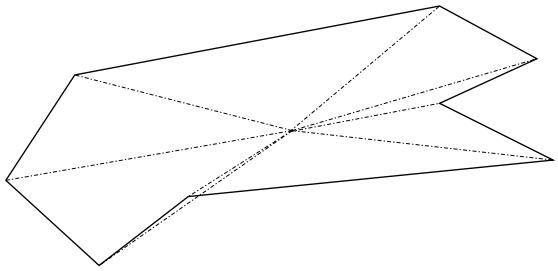


Figure VII.B.2: Triangles for calculation of face quantities

A face's center is defined as the weighted center G of triangles T_i defined as $\{P_a, P_i, P_{i+1}\}$ and whose centers are noted G_i. Let O be the center of the coordinate system and $\overrightarrow{n_f}$ the face normal, then:

$$\overrightarrow{OG} = \frac{\sum_{i=1}^{n} \text{surf}(T_i).\overrightarrow{OG_i}}{\sum_{i=1}^{n} \text{surf}(T_i)} \qquad \text{avec} \quad \text{surf}(T_i) = \frac{\overrightarrow{n_{T_i}}.\overrightarrow{n_f}}{\| \overrightarrow{n_f} \|}$$

It is important to use the signed surface of each triangle so that this formula remains true in the case of non convex faces.

In real cases, some faces are nor perfectly planar. In this case, a slight error is introduced, but it is difficult to choose an exact and practical (implementation and computing cost wise) definition of a polygon's surface when its edges do not all lie in a same plane.

So as to limit errors due to warped faces, we compare the contribution of a given face to the neighboring cell volumes (through Stoke's formula) with the contribution obtained from the separate triangles $\{P_a, P_i, P_{i+1}\}$, and we translate the initial center of gravity along the face normal axis so as to obtain the same contribution.

Cell Centers

If we consider that in theory, the Finite Volume method uses constant per-cell values, we can make without a precise knowledge of a given cell's center, as any point inside the cell could be chosen. In practice, precision (spatial order) depends on a good choice of the cell's center, as this point is used for the computation of values and gradients at mesh faces. We do not compute the center of the circumscribed sphere as a cell center, as this notion is usually linked to tetrahedral meshes, and is not easily defined and calculated for general polyhedra.

Let us consider a cell $\mathcal{C}$ with p faces of centers of gravity G_k and surfaces of norm S_k. If O is the origin of the coordinate system, $\mathcal{C}$'s center G is defined as:

$$\overrightarrow{OG} = \frac{\sum_{k=1}^{p} S_k . \overrightarrow{OG_k}}{\sum_{k=1}^{p} S_k}$$

An older algorithm computed a cell $\mathcal{C}$'s center of gravity as the center of gravity of its vertices q of coordinates X_l:

$$\overrightarrow{OG} = \sum_{l=1}^{q} \frac{\overrightarrow{OX_l}}{q}$$

In most cases, the newer algorithm gives better results, though there are exceptions (notably near the axis of partial meshes with rotational symmetry).

On figure VII.B.3, we show the center of gravity calculated with both algorithms on a 2D cell. On the left, we have a simple cell. On the right, we have added additional vertices as they occur in the case of a joining of non conforming faces. The position of the center of gravity is stable using the newer algorithm, whilst this point is shifted towards the joined sub-faces with the older, vertex-based algorithm (which worsens the mesh quality).

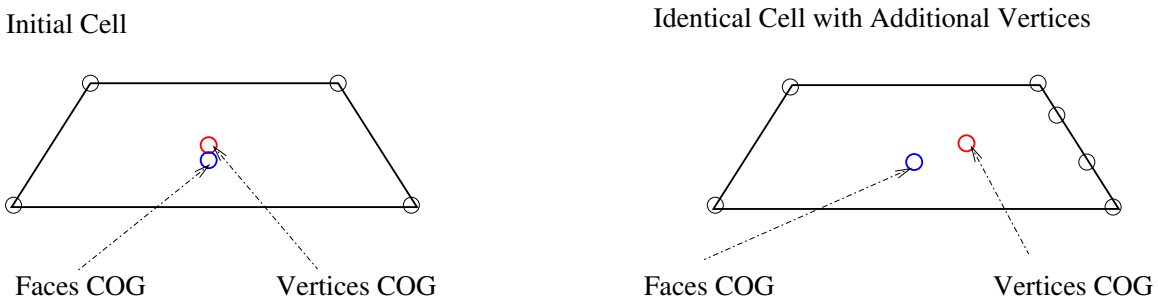


Figure VII.B.3: Choice of cell center

On figure VII.B.4, we show the possible effect of the choice of a cell's COG on the position of line segments joining the COG's of neighboring cells after a joining of non-conforming cells.

We see here that the vertex-based algorithm tends to increase non-orthogonality of faces, compared to the present face-based algorithm.

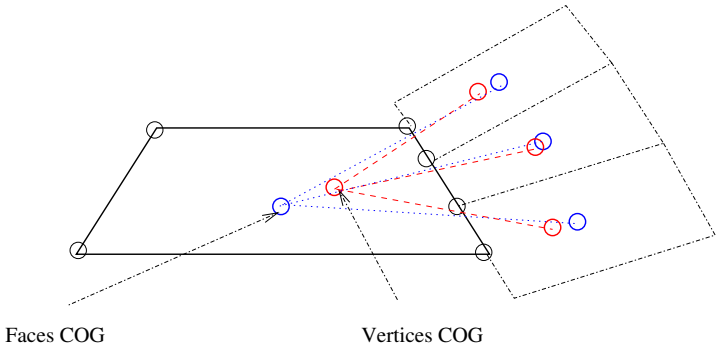


Figure VII.B.4: Choice of cell center and face orthogonality

Conforming Joining

The use of non conforming meshes is one of code_saturne's key features, and the associated algorithms constitute the most complex part of the code's preprocessor. The idea is to build new faces corresponding to the intersections of the initial faces to be joined. Those initial faces are then replaced by their covering built from the new faces, as shown on figure VII.B.5 (from a 2D sideways perspective):

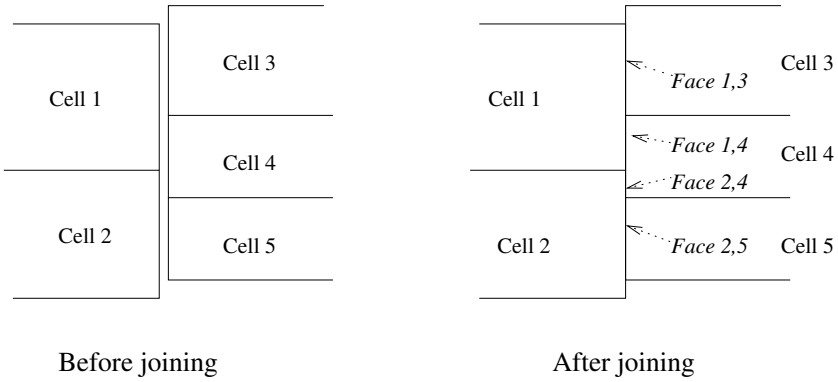


Figure VII.B.5: Principle of face joinings

We speak of *conforming joining*, as the mesh resulting from this joining is conforming, whereas the initial mesh was not.

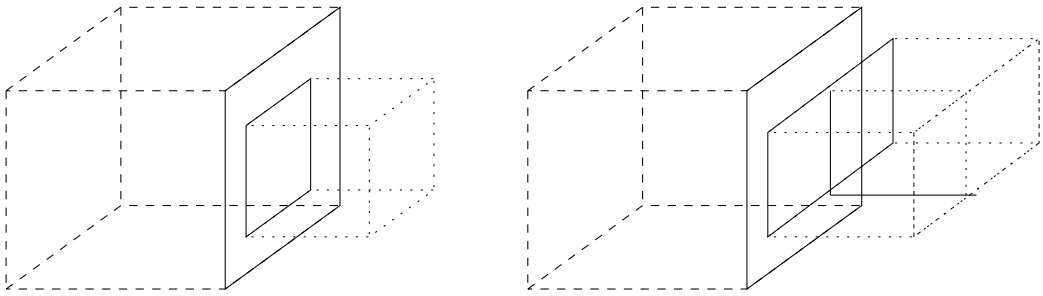
The number of faces of a cell of which one side has been joined will be greater or equal to the initial number of faces, and the new faces resulting from the joining will not always be triangles or quadrangles, even if all of the initial faces were of theses types. For this reason, the data representations of code_saturne and its preprocessor are designed with arbitrary simple polygonal faces and polyhedral cells in mind. We exclude polygons and polyhedra with holes from this representation, as shown on figure VII.B.6. Thus the cells shown in figure VII.B.6 cannot be joined, as this would require opening a "hole" in one of the larger cell's faces. In the case of figure VII.B.6b, we have no such problem, as the addition of another smaller cell splits the larger face into pieces that do not contain holes.

Robustness Factors

We have sought to build a joining algorithm that could function with a minimum of user input, on a wide variety of cases.

Several criteria were deemed important:

1. **determinism**: we want to be able to predict the algorithm's behavior. We especially want the



a) Impossible Definition b) Possible Definition

Figure VII.B.6: Possible case joinings

algorithm to produce the same results whether some mesh A was joined to mesh B or B to A. This might not be perfectly true in the implementation due to truncation errors, but the main point is that the user should not have to worry about the order in which he enters his meshes for the best computational results. [1]

2. **non planar surfaces**: We must be able to join both curved surface meshes and meshes of surfaces assembled from a series of planar sections, but whose normal is not necessarily a continuous function of space, as shown on figure VII.B.7.

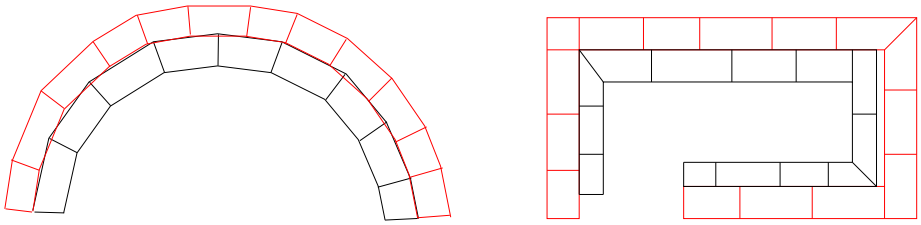


Figure VII.B.7: Initial surfaces

3. **spacing between meshes**: the surfaces to be joined may not match perfectly, due to truncation errors or precision differences, or to the approximation of curved surfaces by a set of planar faces. The algorithm must not leave gaps where none are desired.

Basic Principle

Let us consider two surfaces to join, as in figure VII.B.8: We seek to determine the intersections of the edges of the mesh faces, and to split these edges at those intersections, as shown on figure VII.B.9. We will describe more precisely what we mean by "intersection" of two edges in a later section, as the notion involves spanning of small gaps in our case.

The next step consists of reconstructing sub-faces derived from the initial faces. Starting from an edge of an initial face, we try to find closed loops, choosing at each vertex the leftmost edge (as seen standing on that face, normal pointing upwards, facing in the direction of the current edge), until we have returned to the starting vertex. This way, we find the shortest loop turning in the trigonometric direction. Each face to be joined is replaced by its covering of sub-faces constructed in this manner.

When traversing the loops, we must be careful to stay close to the plane of the original face. We thus consider only the edges belonging to a face whose normal has a similar direction to that of the face

[1] The geometry produced by a joining is in theory independent from the order mesh inputs, but mesh numbering is not. The input order used for a calculation should thus be kept for any calculation restart.

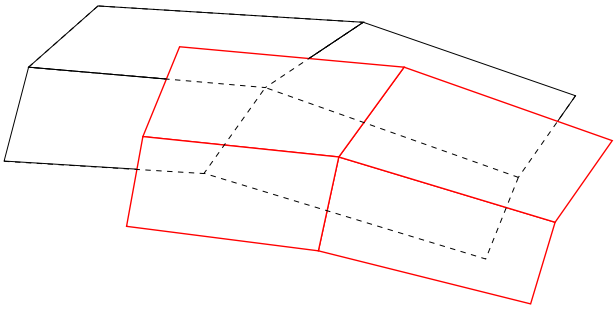


Figure VII.B.8: Surfaces to be joined

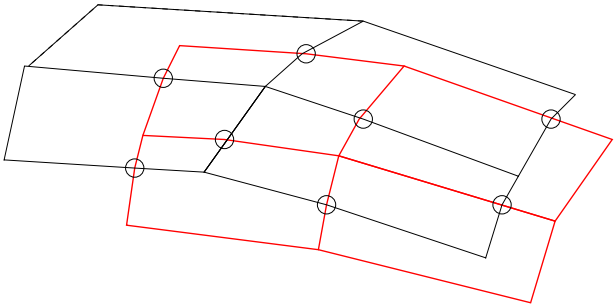


Figure VII.B.9: After edge intersections

being subdivided (i.e. the absolute value of the dot product of the two unitary normals should be close to 1).

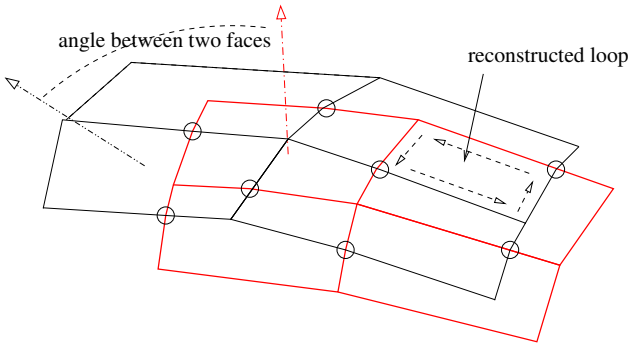


Figure VII.B.10: Sub-face reconstruction.

Once all the sub-faces are built, we should have obtained for two tangent initial faces two topologically identical sub-faces, each descending from one of the initial faces and thus belonging to a different cell. All that is required at this stage is to merge those two sub-faces, conserving the properties of both. The merged sub-face thus belongs to two cells, and becomes an internal face. The joining is thus finalized.

Simplification of Face Joinings

For a finite-volume code such as code_saturne, it is best that faces belonging to one same cell have neighboring sizes. This is hard to ensure when non-conforming boundary faces are split so as to be joined in a conforming way. On figure VII.B.11, we see that this can produce faces of highly varying sizes when splitting a face for conformal joining.

It is possible to simplify the covering of a face so as to limit this effect, by moving vertices slightly on each side of the covering. We seek to move vertices so as to simplify the covering while deforming the mesh as little as possible.

One covering example is presented figure VII.B.11, where we show several simplification opportunities. We note that all these possibilities are associated with a given edge. Similar possibilities associated with other edges are not shown so that the figure remains readable. After simplification, we obtain the situation of figure VII.B.12.

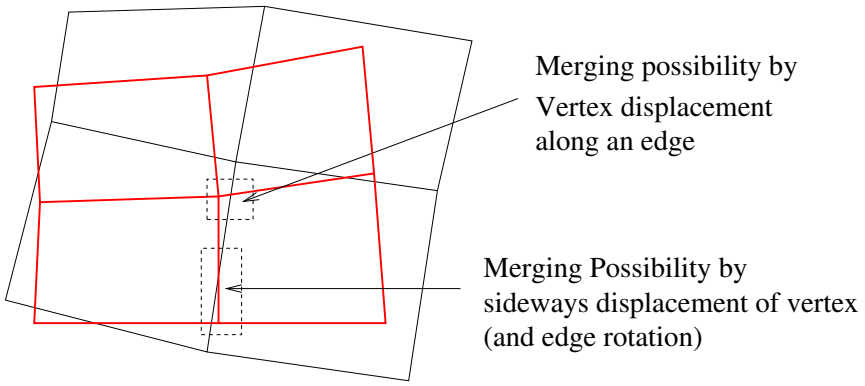


Figure VII.B.11: Simplification possibilities

After simplification, we have the following situation:

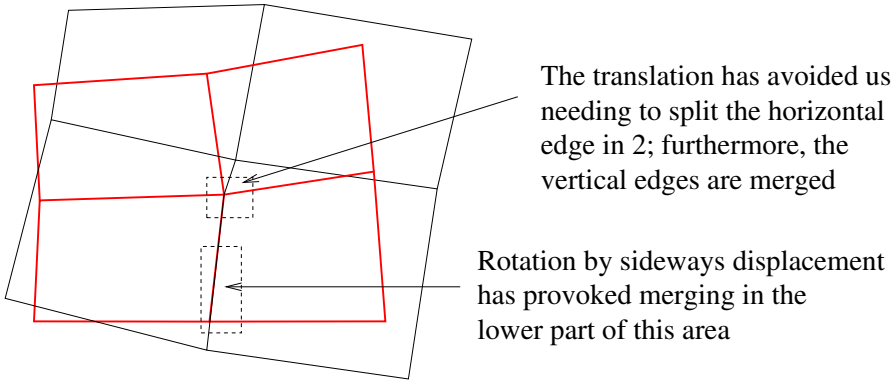


Figure VII.B.12: Faces after simplification

Processing

The algorithm's starting point is the search for intersections of edges belonging to the faces selected for joining. In 3D, we do not restrict ourselves to "true" intersections, but we consider that two edges intersect as soon as the minimum distance between those edges is smaller than a certain tolerance.

To each vertex we associate a maximum distance, some factor of the length of the smallest edge incident to that vertex. This factor is adjustable (we use 0.1 by default), but should always be less than 0.5. By default, this factor is usually multiplied by the smallest sine of the angles between the edges considered, so that the tolerance assigned to a given vertex is a good estimation of the smallest height/width/depth or the adjacent cells.

On figure VII.B.14, we illustrate this tolerance in 2d with a factor of 0.25, with red circles showing the tolerance region without the sine factor correction, and blue circle showing the tolerance with the sine correction.

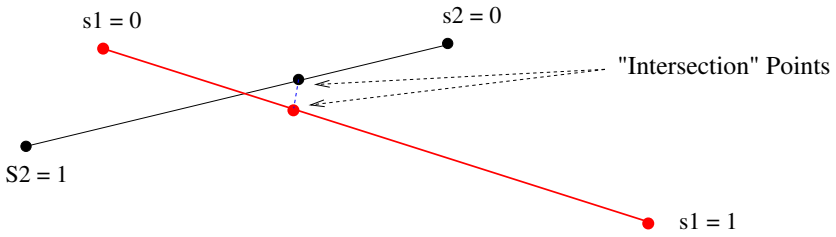


Figure VII.B.13: Intersection of edges in 3d space

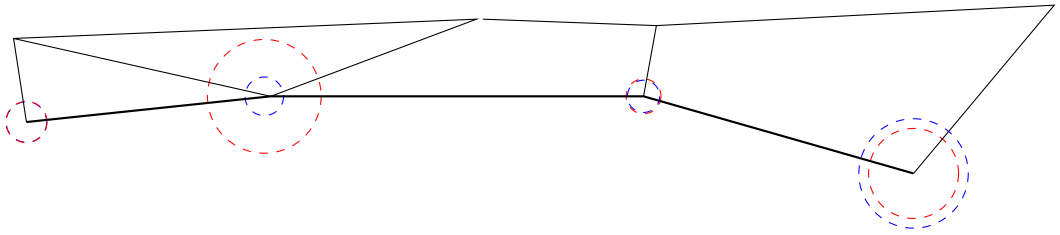


Figure VII.B.14: Join tolerance for vertices of joinable faces

We consider a neighborhood of an edge defined by the spheres associated to the minimal distances around each vertex and the shell joining this two spheres, as shown on figure VII.B.15. More precisely, at a point on the edge of linear abscissa s, the neighborhood is defined to be the sphere of radius $d_{max}(s) = (1-s)d_{max}\mid_{s=0} + s.d_{max}\mid_{s=1}$.

We will thus have intersection between edges $E1$ and $E2$ as soon as the point of $E1$ closest to $E2$ is within the neighborhood of $E2$, and that simultaneously, the point of $E2$ closest to $E1$ is inside the neighborhood of $E1$.

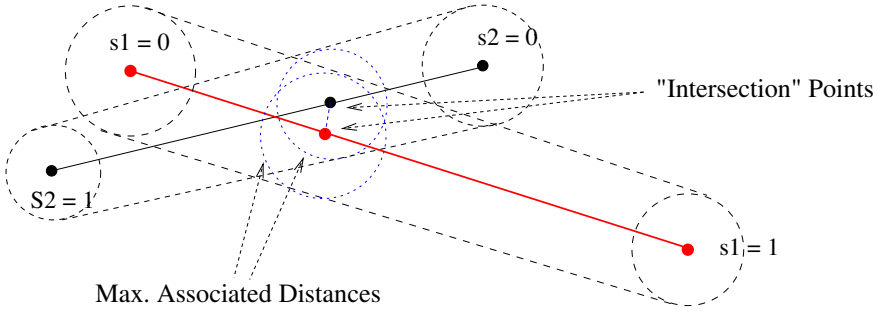


Figure VII.B.15: Tolerances for intersection of edges

If edge $E2$ cuts the neighborhood of a vertex of edge $E1$, and this vertex is also in $E2$'s neighborhood, we choose this vertex to define the intersection rather than the point of $E1$ closest to $E2$. This avoids needlessly cutting edges. We thus search for intersections with the following order of priority: vertex-vertex, vertex-edge, then edge-edge. If the neighborhoods associated with two edges intersect, but the criterion:
$\exists P1 \in A1, \exists P2 \in A2, d(P1,P2) < min(d_{max}(P1), d_{max}(P2))$
is not met, we do not have intersection. These cases are shown on figure VII.B.16.

Problems Arising From the Merging of Two Neighboring Vertices

If we have determined that a vertex V_1 should be merged with a vertex V_2 and independently that this vertex V_2 should be merged with a vertex V_3, then V_1 and V_3 should be merged as a result, even though these vertices share no intersection. We refer to this problem as merging transitivity and show

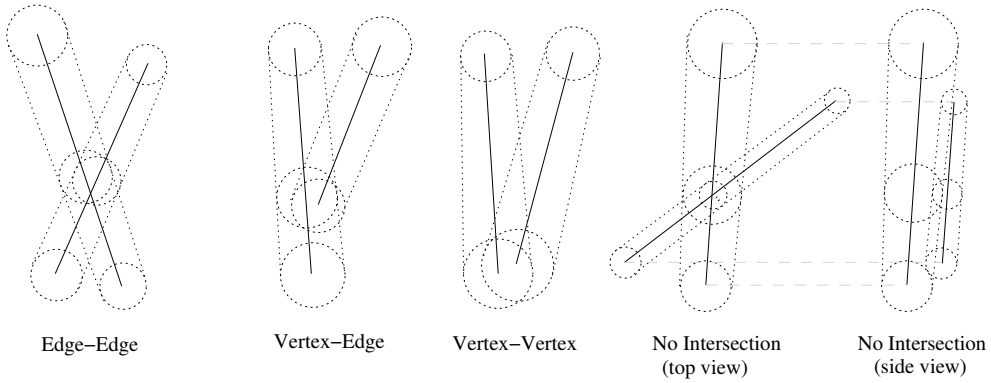


Figure VII.B.16: Tolerances for intersection of edges

theoretical situations leading to it on figure VII.B.17.

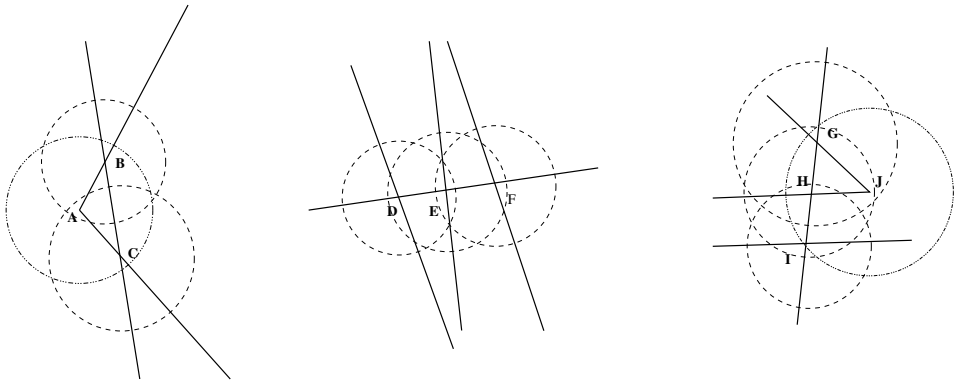


Figure VII.B.17: Merging transitivity.

On figure VII.B.18, we show cases more characteristic of what we can obtain with real meshes, given that the definition of local tolerances reduces the risk and possible cases of this type of situation.

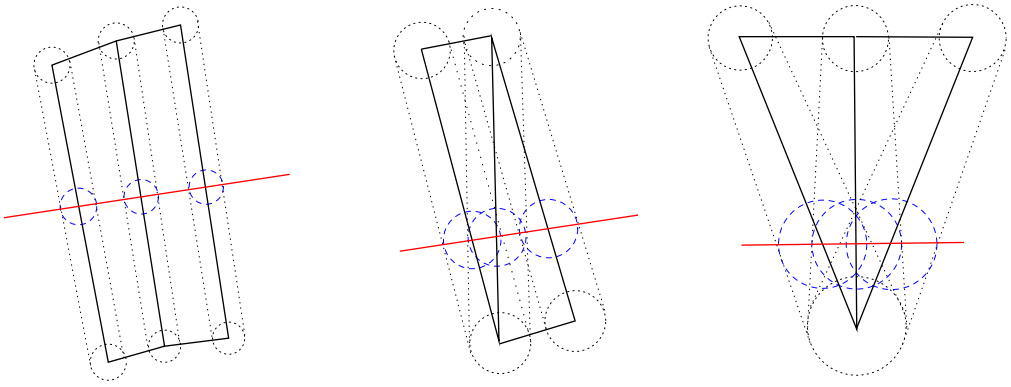


Figure VII.B.18: Merging transitivity (real cases)

Figure VII.B.19 illustrates the effect of a combination of merges on vertices belonging to a same edge. We see that in this case, edges initially going through vertices G and J are strongly deformed (i.e. cut into sub-edges that are not well aligned). Without transitivity, edges going through vertices descended only from the merging of (G, H) on one hand and (L, J) on the other hand would be much closer to the initial edges.

To avoid excessive simplifications of the mesh arising from a combination of merges, we first build "chains" of intersections which should be merged, compute the coordinates of the merged intersection, and then check for all intersection couples of a given chain if excessive merging would occur. If this is the case, we compute a local multiplicative factor (< 1) for all the intersections from this chain, so as to reduce the merge tolerance and "break" this chain into smaller subchains containing only intersections withing each other's merging distances.

Tolerance reduction may be done in multiple steps, as we try to break the weakest equivalences (those closest to the merge tolerance bounds) first.

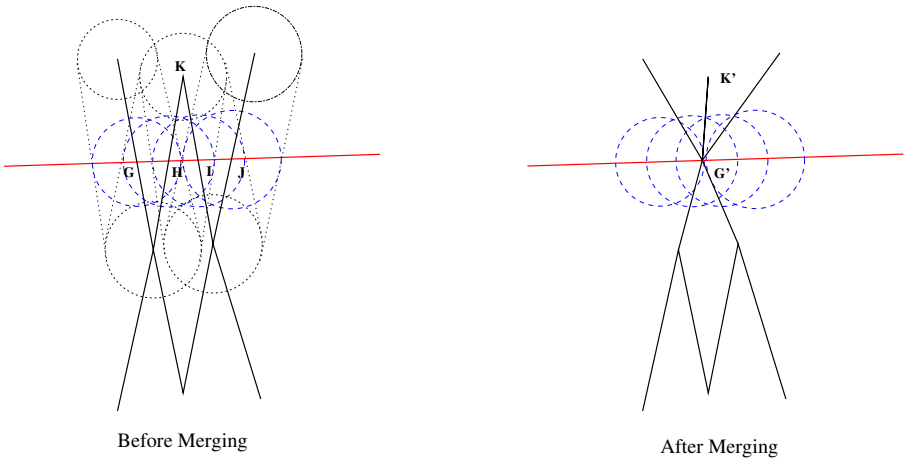


Figure VII.B.19: Merging transitivity for an edge

On figure VII.B.20, we show the possible effect of merge transitivity limitation on vertices belonging to several edges. Here, breaking of excessive intersection mergings should lead to merging of intersections (G, H, J), while I is not merged with another intersection.

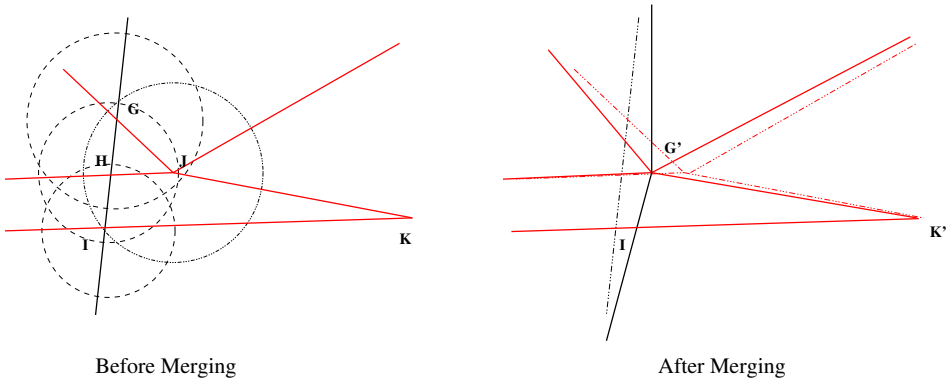


Figure VII.B.20: Limited merging transitivity

Algorithm Optimization

Certain factors influence both memory and CPU requirements. We always try to optimize both, with a slight priority regarding memory requirements.

When searching for edge intersections, we try to keep the number of intersection tests to a minimum. We compute coordinate axis-aligned bounding boxes associated with each joinable face (augmented by the tolerance radii of associated vertices), and run a full edge intersection test only for edges belonging to faces whose bounding boxes intersect.

To determine which face bounding boxes intersect, we build a "bounding box-tree" , similar to an octree, but with boxes belonging to all the octree leaves they overlap. When running in parallel using MPI, a first, coarser version of the tree with the same maximum depth on all ranks is built so as to estimate the optimal distribution of data, to balance both the computational and memory load. Bounding boxes are then migrated to the target ranks, where the final box-tree (which may have different depth on different ranks) is built. Assigning the global ids of the matching faces to the bounding boxes ensures the search results are usable independently of the distribution across MPI ranks.

Influence on mesh quality

It is preferable for a FV solver such as code_saturne that the mesh be as "orthogonal" as possible (a face is perfectly orthogonal when the segment joining its center of mass to the center of the other cell to which it belongs is perfectly aligned with its normal). It is also important to avoid non planar faces [2]. By the joining algorithm's principle, orthogonal faces are split into several non-orthogonal sub-faces. In addition, the higher the tolerance setting, the more merging of neighboring vertices will tend to warp faces on the sides of cells with joined faces.

It is thus important to know when building a mesh that a mesh built by joining two perfectly regular hexahedral meshes may be of poor quality, especially if a high tolerance value was used and the cell-sizes of each mesh are very different. It is thus important to use the mesh quality criteria visualizations available in code_saturne, and to avoid arbitrary joinings in sensible areas of the mesh. When possible, joining faces of which one set is already a subdivision of another (to construct local refinements for example) is recommended.

Periodicity

We use an extension of the non-conforming faces joining algorithm to build periodic structures. The basic principle is described figure VII.B.21:

- Let us select a set of boundary faces. These faces (and their edges and vertices) are duplicated, and the copy is moved according to the periodic step (a combination of translation and rotation). A link between original and duplicate entities is kept.
- We use a conforming joining on the union of selected faces and their duplicates. This joining will slightly deform the mesh so that vertices very close to each other may be merged, and periodic faces may be split into conforming sub-faces (if they are not already conforming).
- If necessary, the splitting of duplicated, moved, and joined faces is also applied to the faces from which they were descended.
- Duplicated entities which were not joined (i.e. excess entities) are deleted.

It is thus not necessary to use periodic boundary conditions that the periodic surfaces be meshed in a periodic manner, though it is always best not to make too much use of the comfort provided by conforming joinings, as it can lower mesh quality (and as a consequence, quality of computational results).

We note that it could seem simpler to separate periodic faces in two sets of "base" and "periodic" faces, so as to duplicate and transform only the first set. This would allow for some algorithm simplifications and optimizations, but here we gave a higher priority to the consistency of user input for specification of face selections, and the definition of two separate sets of faces would have made user input more

[2] Computation of face COG's includes a correction such that the contribution of a warped face to a cell's volume is the same as that of the same face split into triangles joining that face's COG and outer edges, but this correction may not be enough for second order values.

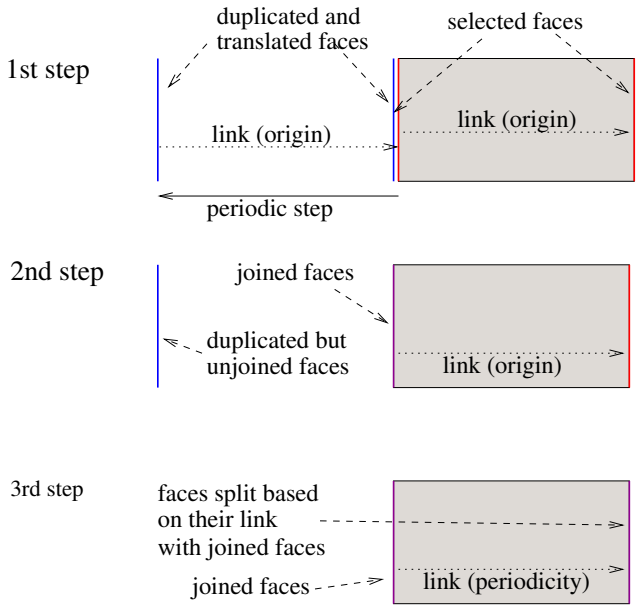


Figure VII.B.21: Periodic joining principle (translation example)

complex. As for the basic conforming joining, it is usually not necessary to specify face selections for relatively simple meshes (in which case all faces on the mesh boundary are selected).

Triangulation of faces

Face triangulation is done in two stages. The first stage uses an *ear cutting* algorithm, which can triangulate any planar polygon, whether convex or not. The triangulation is arbitrary, as it depends on the vertex chosen to start the loop around the polygon.

The second stage consists of flipping edges so that the final triangulation is constrained Delaunay, which leads to a more regular triangulation.

Initial triangulation

The algorithm used is based on the one described in [Theußl, 1998]. Its principle is illustrated figure VII.B.22. We start by checking if the triangle defined by the first vertices of the polygon, (P_0, P_1, P_2) is an "ear", that is if it is interior to the polygon and does not intersect it. As this is the case on this example, we obtain a first triangle, and we must then process the remaining part of the polygon. At the next stage triangle (P_0, P_2, P_3) is also an ear, and may be removed.

At stage 2, we see that the next triangle which is a candidate for removal, (P_0, P_3, P_4) is not an ear, as it is not contained in the remaining polygon. We thus shift the indexes of vertices to consider, and see at stage 4 that triangle (P_3, P_4, P_5) is an ear and may be removed.

The algorithm is built in such a way that a triangle is selected based on the last vertex of the last triangle considered (starting from triangle (P_0, P_1, P_2)). Thus, we consider at stage 5 the triangle ending with P_5, that is (P_0, P_3, P_5). Once this triangle is removed, the remaining polygon is a triangle, and its handling is trivial.

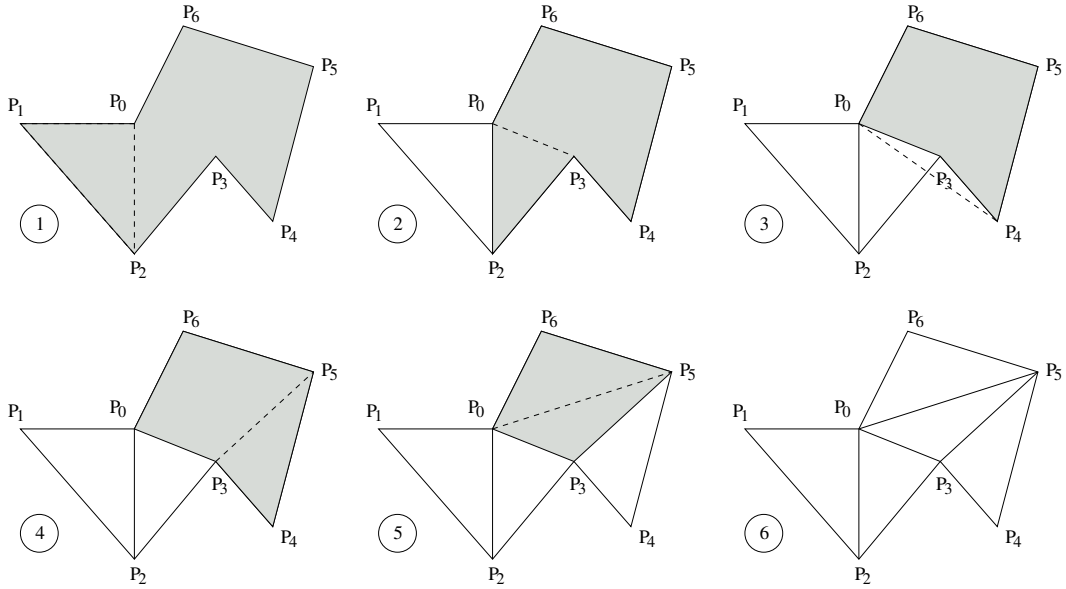


Figure VII.B.22: Principle of face triangulation

Improving the Triangulation

We show on figures VII.B.23 and VII.B.24 two examples of a triangulation on similar polygons whose vertices are numbered in a different manner.

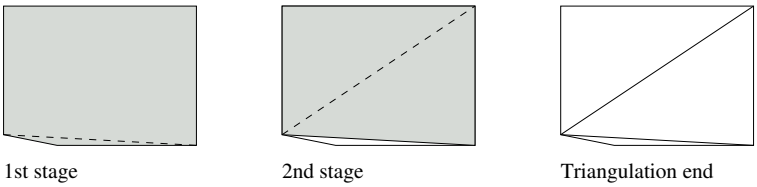


Figure VII.B.23: Triangulation example (1)

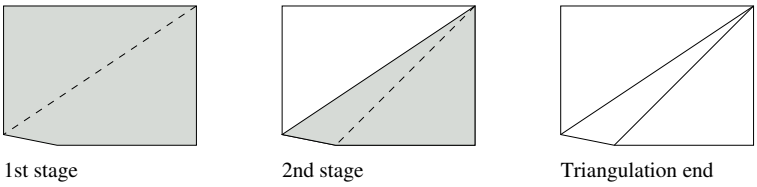


Figure VII.B.24: Triangulation example (2)

Not only is the obtained triangulation different, but it has a tendency to produce very flat triangles. Once a first triangulation is obtained, we apply a corrective algorithm, based on edge flips so as to respect the Delaunay condition [Shewchuk, 1999].

This condition is illustrated figure VII.B.25. In the first case, edge $\overline{P_iP_j}$ does not fulfill the condition, as vertex P_l is contained in the circle going through P_i, P_j, P_k. In the second case, edge $\overline{P_kP_l}$ fulfills this condition, as P_i is not contained in the circle going through P_j, P_k, P_l.

If triangles (P_i, P_k, P_j) and (P_i, P_j, P_l) originated from the initial triangulation of a same polygon, they would thus be replaced by triangles (P_i, P_k, P_l) and (P_l, P_k, P_j), which fulfill the Delaunay condition.

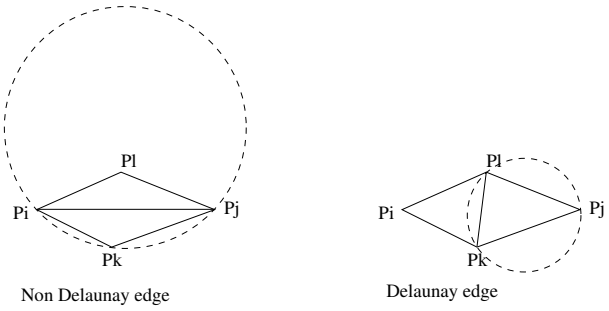


Figure VII.B.25: Delaunay condition (2)

In the case of a quadrangle, we would be done, but with a more complex initial polygon, these new triangles could be replaced by others, depending on their neighbors from the same polygon.

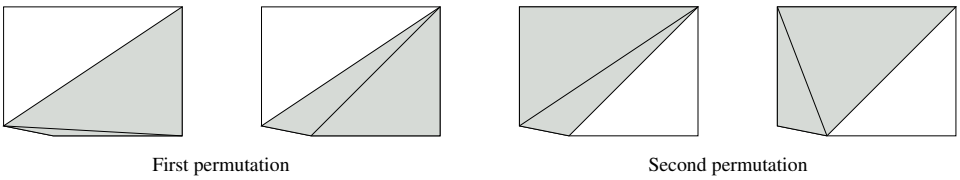


Figure VII.B.26: Edge flip example (1)

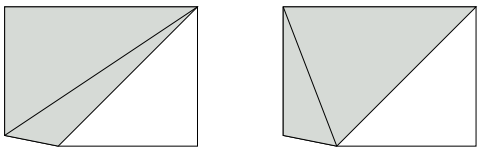


Figure VII.B.27: Edge flip example (2)

On figures VII.B.26 and VII.B.27, we illustrate this algorithm on the same examples as before (figures VII.B.23 and VII.B.24). We see that the final result is the same. In theory, the edge flipping algorithm always converges. To avoid issues due to truncation errors, we allow a tolerance before deciding to flip two edges. Thus, we allow that the final triangulation only "almost" fulfill the Delaunay condition.

In some cases, especially that of perfect rectangles, two different triangulations may both fulfill the Delaunay condition, notwithstanding truncation errors. For periodicity, we must avoid having two periodic faces triangulated in a non-periodic manner (for example, along different diagonals of a quadrangle). In this specific case, we triangulate only one face using the geometric algorithm, and then apply the same triangulation to it's periodic face, rather then triangulate both faces independently.

Unwarping algorithm

The unwarping algorithm is a smoother, its principle is to mitigate the local defects by averaging the mesh quality. It moves vertices using an iterative process which is expected to converge to a mesh with better averaged warping criteria.

See the programmers reference of the dedicated subroutine for further details.

Warping criterion in code_saturne

The warp face quality criterion in code_saturne represents the non coplanarity in space of N points $P_{i=1:N}$ $(N > 3)$.

Let f be the face defined by $P_{i=1:Nbv(f)}$, the center of gravity O_f and $\overrightarrow{n_f}$ the face normal, the warping criterion is calculated for each face by the "maximum" angle between $\overrightarrow{P_iP_{i+1}}$ and $\overrightarrow{n_f}^{\perp}$ $\forall i \in [1, Nbv(f)]$ where $P_{Nbv(f)+1} = P_1$. For consistency purposes, the warping criterion $warp_f$ is defined in degree for each face of the mesh $\mathcal{M}$ as:

$$\forall f \in \mathcal{M}, \qquad warp_f = 90 - \arccos\left(\max_{\forall i \in [1,Nbv(f)]}\left(\cos(\overrightarrow{P_iP_{i+1}}, \overrightarrow{n_f})\right)\right)\frac{180}{\pi}$$

Unwarping method

The principle of unwarping algorithm is to move vertices in the midplane of the faces using an iterating process. At each iteration the algorithm tries to approach vertices from the midplane of the face without increasing the warping of the neighbours faces.

The displacement is calculated by projecting vertices onto the plane defined by $\overrightarrow{n_f}$ and the center of gravity O.

For the face f, $\forall i \in [1, Nbv(f)]$, the vertices P_i are displaced by the vector $\lambda^f_{P_i}$

$$\forall i \in [1, Nbv(f)], \qquad \lambda^f_{P_i} = (\overrightarrow{P_iO_f}.\overrightarrow{n_f})\overrightarrow{n_f}$$

In most cases, a vertex is shared by many faces $f_{j=1:Nbf(P_i)}$, and its final displacement is:

$$\forall f \in \mathcal{M},\ \forall i \in [1, Nbv(f)], \qquad \lambda^f_{P_i} = \sum_{j=1:Nbf(P_i)} \lambda^{f_j}_{P_i}$$

This displacement technique may cause problems because the contributions $\lambda^{f_l}_P$ and $\lambda^{f_k}_P$ can be "contradictory". Moreover, if a small and a large face are neighbours, the large face contribution imposes a too big displacement to the shared vertices and the warping criterion can be deteriorated.

Displacements control

The weighting coefficients shown below allow us to reduce the conflicting contributions and equilibrate the contributions between small and large faces.

Face weighting Every iteration, for each face the warping criterion is computed. It is used to give more weight to the warp faces. After the renormalisation of the warping criterion, a new displacement formula is obtained:

$$\forall f \in \mathcal{M},\ \forall i \in [1, Nbv(f)], \qquad \lambda^f_{P_i} = \sum_{j=1:Nbf(P_i)} \frac{warp_{f_j}}{\max_{f\in\mathcal{M}} warp_f}\lambda^{f_j}_{P_i}$$

Vertex weighting Every iteration, for each vertex $P \in \mathcal{M}$, the vertex tolerance is computed:

$$\forall P \in \mathcal{M}, \qquad vtxtol_P = \frac{\min_{P\in nb(P)} PP\prime}{\max_{Q\in\mathcal{M}} \min_{Q\prime\in nb(Q)} QQ\prime}$$

where $nb(Q)$ are the first neighbors of the point Q.

This criterion is used to reduce the vertex displacement when a vertex lies on an edge much smaller than the average length. Another vertex tolerance may have been chosen. For example a more "local coefficient" can be defined as:

$$\forall P \in \mathcal{M}, \qquad vtxtol_P^{local} = \frac{\min_{P\prime \in nb(P)} PP\prime}{\max_{P\prime \in nb(P)} PP\prime}$$

This "local coefficient" has been noticed to be less efficient than the first one. That is why the first definition is used in code_saturne.

The unwarping displacement is updated as below:

$$\forall f \in \mathcal{M},\ \forall i \in [1, Nbv(f)], \qquad \lambda_{P_i}^f = vtxtol_{P_i} \sum_{j=1:Nbf(P_i)} \frac{warp_{f_j}}{\max_{f\in\mathcal{M}} warp_f} \lambda_{P_i}^{f_j}$$

Movement coefficient To ensure a better convergence, all vertices' movements are multiplied by a scale factor Cm (where $Cm \leqslant 1$). This coefficient helps to converge to the best solution by preventing a too big movement in the wrong direction. In code_saturne the default value is set to 0.10.

The unwarping displacement is then:

$$\forall f \in \mathcal{M},\ \forall i \in [1, Nbv(f)], \qquad \lambda_{P_i}^f = Cm * vtxtol_{P_i} \sum_{j=1:Nbf(P_i)} \frac{warp_{f_j}}{\max_{f\in\mathcal{M}} warp_f} \lambda_{P_i}^{f_j}$$

Maximum displacement To reduce the "cell reversal" risk, the maximum displacement is limited for each vertex $P \in \mathcal{M}$ to $Md = 0.10 \min_{P\in nb(P)} PP\prime$.

Finally, the complete unwarping displacement formula is defined as:

$$\forall f \in \mathcal{M}, \forall i \in [1, Nbv(f)],$$
$$\lambda_{P_i}^f = \min(Cm * vtxtol_{P_i} \sum_{j=1:Nbf(P_i)} \frac{warp_{f_j}}{\max_{f\in\mathcal{M}} warp_f} \lambda_{P_i}^{f_j},\ Md)$$

Stop criterion

The algorithm automatically stops according to the warp face criterion.

1^{st} case: the algorithm converges The algorithm stops when, at the i^{th} iteration, the relation below is verified.

$$1 - \frac{\max_{f\in\mathcal{M}} warp_f^i}{\max_{f\in\mathcal{M}} warp_f^{i-1}} < 1.E-4$$

2^{nd} case: the algorithm diverges The algorithm stops when at the i^{th} iteration the relation below is verified.

$$\frac{\max_{f\in\mathcal{M}} warp_f^i}{\max_{f\in M} warp_f^{i-1}} > 1.05$$

It means that the current iteration degrades the previous one. The obtained mesh is the result of the $(i-1)^{th}$ iteration.

3^{rd} **case: the maximum number of iterations is reached** The algorithm stops after N_{max} iterations (51 by default in code_saturne).

Specific treatment for boundary faces

The unwarping algorithm may modify the mesh geometry. The function `fix_by_feature` allows to fix boundary faces according to a feature angle. The feature angle between a vertex and one of its adjacent faces is defined by the angle between the vertex normal and the face normal.

A vertex normal is defined by the average of the normals of the faces sharing this vertex.

This function fixes a vertex if one of its feature angles is less than $cos(\theta)$ where θ is the maximum feature angle (in degrees) defined by the user. In fact, if $\theta = 0°$ all boundary vertices will be fixed, and if $\theta = 90°$ all boundary vertices will be free.

Fixing all boundary vertices ensures the geometry is preserved, but reduces the smoothing algorithm's effectiveness.

See the programmers reference of the dedicated subroutine for further details.

Appendix C

Mesh Quality

Flagging of bad cells

No current meshing tool that we know of is capable of generating a complex mesh in reasonable time without some cells of bad "quality". To reduce the number of iterations required to obtain an acceptable mesh, we may try to flag the few bad quality cells that are considered almost unavoidable, so as to limit their impact on the solver's robustness.

As of the current version of code_saturne, no specific treatment is done, but "bad cells" are marked, so as as to allow the user to know where to expect issues with the mesh.

By default, the mesh quality is inspected at the very beginning of every calculation. The estimated quality is defined by at least five criteria:

- the cell's non-orthogonality,
- the cell's offset,
- the cell's distorsion (or least-squares gradient criteria),
- the cell's volume ratio,
- "guilt by association",
- and possibly supplementary user criteria.

Supplementary user criteria will not be discussed here but an example of how to define them is provided in the user subroutine `cs_user_mesh.c`.

As a rule of thumb users need to be aware that bad cells could lead to a degradation of the solution quality, or worse, to a failed calculation. Generally, a cell quality is degraded by non-conforming joining operations, especially when joined cells have different sizes or thicknesses, but even painstakingly built block-structured meshes may have cells of bad quality when they involve a combination of warping, anisotropy and refinement variation.

See the programmers reference of the dedicated subroutine for further details.

Cell non-orthogonality

For a finite volume solver, the mesh cells should be as "orthogonal" as possible. Consequently, the compliance with this criterion is of particular interest in order to avoid the degradation of the solution quality.

A cell's non-orthogonality relative to a face is evaluated as:

$$Q_{f_{c|\overline{c}}}^{ortho} = \frac{\underline{\underline{x_c x_{\overline{c}}}} \cdot \underline{S}_{ij}}{\left|\underline{\underline{x_c x_{\overline{c}}}}\right| \ \left|\underline{S}_{ij}\right|} \tag{VII.C.1}$$

where $\underline{\underline{x_c x_{\overline{c}}}}$ is a distance vector between two consecutive cell centers and $\underline{S}_{ij}$ is the surface vector normal to the face.

Orthogonal cells have a value of Q_f^{ortho} which tends towards 1.0 for all the faces. Therefore, a cell is flagged bad if $Q_f^{ortho} < 0.1$ for any of its faces.

Cell offset

A cell's offset (relative to a face) is evaluated in a manner consistent with iterative gradient recontruction:

$$Q_{f_{c|\overline{c}}}^{offset} = 1 - \sqrt[3]{\frac{\left|\underline{\underline{x_o x_f}}\right| \left|\underline{S}_{ij}\right|}{|V_c|}} \tag{VII.C.2}$$

where V_c is the cell's volume, $\underline{S}_{ij}$ is the surface normal to the face, and $\underline{\underline{x_o x_f}}$ is the difference between the face's center and its intersection with the segment joining adjacent cell centers.

Orthogonal cells have a value of Q^{offset} which tends towards 1.0. Therefore, a cell is flagged bad if $Q^{offset} < 0.1$ for any of its faces.

Cell distorsion

This criterion evaluates a distorsion level based on least squares gradient computation. As a first step, the geometric matrix containing information on distance vectors between neighboring cells is built (see the construction of the C matrix in the least squares gradient computation for more details). In a second step, the matrix's eigenvalues are estimated using a Jacobi transformation method. The min/max eigenvalues ratio is used as the cell's distorsion criteria:

$$Q_{LSQ} = \frac{min(|C_{egv}|)}{max(|C_{egv}|)}$$

where C_{egv} are the eigenvalues of the geometric matrix build according to the least squares gradient computation.
Cubic cells have a value of Q_{LSQ} which tends towards 1.0. Therefore, a cell is flagged bad if $Q_{LSQ} < 0.1$.

Cell volume ratio

A cell's volume ratio criteria gives an estimation of cell's caracteristic size continuity. It is evaluated as:

$$Q_{vol} = min\left(\frac{V_1}{V_2}, \frac{V_2}{V_1}\right)$$

where V_1 and V_2 are the respective volumes of two neighboring cells.
Neighboring cells with the same size have a value of Q_{vol} which tends towards 1.0. Therefore, two neighboring cells are flagged bad if $Q_{vol} < 0.1^2$.

Guilt by association

Once we have finished with determining which cells are flagged "bad", the last step is to mark initially "good" cells (according to the above criteria) as "bad" when all their neighbors are flagged "bad".

Appendix D

Extended neigbourhood

D.1 Extended cell neighbourhood

The neighbourhood is defined as the collection of cells J_i around the cell I under consideration for various algorithms.

We consider:

- the *standard* neighborhhod, containing all cells sharing a face with I.
- the *extended* neighborhhod, containing a subset of cells sharing a vertex with I, and excluding the standard neighborhood.

The standard neighbourhood is used in most finite volume operators, whether computing fluxes and balances, or in gradient reconstruction.

The extended neighborhood may be used to supplement the base neighborhood when using least-squares gradient reconstruction, so as to obtain a smoother and more stable result in the case of significant face non-orthogonality of offset (which is always the case with tetrahedral meshes, and may be the case with strongly distorted hexahedra).

It may also be used for some other smoothing operations, such as LES filters or other smoothing operators (though it is recommended to use another approach for operators not akin to gradients, so that tuning for gradients does not interfere with those operators).

D.1.1 Reduced extended neighborhood

Whereas the standard neighborhood usually 6 cells for hexahedra and 4 for tetrahedra, the matching extended neighborhood contains 20 cells (26 - 6) for hexahedra, and often more than 60 cells on average for tetrahedra (some tetrahedra may even have more than 150 neighbors). So accessing and computing values based on this neighborhood can be quite expensive.

So when possible, we will use a reduced (more precisely, restricted) extended neighborhood, including only a subset of a cell's vertex-adjacent neighbors, so as to improve performance.

Several algorithms for such reduction are possible (see the user guide).

The simplest one selects cells with centers best aligned opposite to face-adjacent cell centers, and discards others, leading to a number of additional cells in the extended neighbourhood at most equal to the number of cell faces. Using this restriction significantly improves stability with tetradral meshes compared with using only the standard neighborhood, with little additional cost, but does not provide the same robustness as using the full neighbourhood.

Another optimized neighbourhood reduction (restriction) algorithm based on heuristics provides better robustness, at a reasonable additional cost (much faster than using the complete extended neighborhood) is described in the next section.

D.2 Optimized (heuristics) neighbourhood

D.2.1 Principle

This optimized neighbourhood is designed to:

- have a calculation cost reduced compared to full neighbourhood (all the cells sharing at least one vertex with the cell I under consideration) by selecting only some cells in the full neighbourhood and
- minimise both the offset of the neighbourhood $||\underline{IC}||$ and the average distance of the neighbours.

The optimized neighbourhood is also designed to improve a weakness of the full neighbourhood: a tendency to produce more prominent spurious modes (repeated non physical jumps of a given quantity passing from one cell to another) than the restricted neighbourhood (all the cells sharing at least one face with the cell I under consideration). The spurious modes appear in regular hexahedral or prismatic portions of the mesh where the restricted neighbourhood is stable enough. The optimized neighbourhood has a criterion to detect these zones and if the criterion is met it switches to a restricted neighbourhood.

Moreover, the optimized neighbourhood will keep all the cells in the full neighbourhood if the cell I has a very large aspect ratio.

Finally, at the boundary, the default behaviour is also to keep the restricted neighbourhood, only the cells with more than one boundary face will use more neighbours. These choices come from numerical experiments.

D.2.2 Implementation

First, the cell type is deduced from the number of triangles and quadrangles of the cell I. For example, a cell with 6 quadrangles only is assumed to be a hexahedra, while a cell with 2 triangles and 3 quadrangles is assumed to be a prism, and one with 4 triangles a tetrahedra.

Remark D.1 *In rare cases, a cell with 2 triangles and 3 quadrangles might be a tetrahedron with a split face, as cells in figure D.2.3), but this is rare (expected to occur only when an initial face is subdivided through a mesh joining operation), so we prefer to keep tests simple, for better performance.*

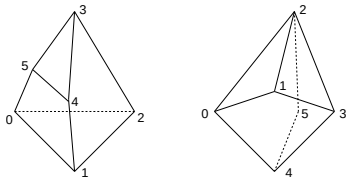


Figure VII.D.1: Tetrahedra with split face: degenerate and false prism

Second, the center of gravity of the restricted neighbourhood C is calculated together with the average distance of the cells in the restricted neighbourhood. Also, if the cell is identified as an hexahedra or a prism, the following regularity criterion is calculated for each face[1]:

$$r_{crit} = \underline{IF}.\underline{FJ} - 0.9\,||\underline{IF}||\,||\underline{FJ}|| \tag{VII.D.1}$$

If for all faces the criterion is > 0 then the mesh is considered to be locally regular and only the restricted neighbourhood will be retained.

Another geometric criterion has an impact on the retained neighbourhood, the aspect ratio defined as follow:

$$ar = \frac{6\,\Omega^{2/3}}{\sum_{n_F} S} \tag{VII.D.2}$$

Obviously for a cube, $ar = 1$. For a rectangular solid with an height a hundred times smaller than the base, $ar = 0.1365$. Any cell with $ar < 0.1365$ will have the full neighbourhood activated.

[1] only triangles face in case of a prism since a prismatic mesh is assumed to possibly be regular only in the direction normal to the triangle faces

The last part of the code is the definition of the optimized neighbourhood for cells that are not "regular" nor "too flat".

Inside a loop on the cells in the extended neighbourhood only, the cell J_i which minimises the sum of the updated offset $\underline{IC}$ and its distance to the cell I is selected. With n the number of cells already in the neighbourhood (restricted included) it writes

$$\text{argmin}\left(\left|\left|n\underline{IC} + (n+2)\underline{IJ_i}\right|\right|\right) \tag{VII.D.3}$$

After that, both the offset $\underline{IC}$ and the average length of the selected cells $\overline{||\underline{IJ_i}||}$ are updated. Then, if the total number of cells retained inside the extended neighbourhood reaches a min value fixed at 10 in the code and if the new selected cell J_i improves the neighbourhood, the whole set of cells already selected is retained.

The improvement of the neighbourhood is written as:

$$\left(||\underline{IC}|| + 0.1\,\Omega^{1/3}\right)\,\overline{||\underline{IJ_i}||} \tag{VII.D.4}$$

Also, as soon as the offset $||\underline{IC}||$ is small enough ($< 0.01\,\Omega^{1/3}$) the search for new "optimal" neighbours is stopped.

D.2.3 Suggested improvements

- There is a possibility that the degenerated or false prism on figure appears as a regular prism. This would be unfortunate because such a cell would be of very poor quality and would need more than the restricted neighbourhood. Further geometric tests could detect this.
- Selecting n_s cells among the n_e ones in the full neighbourhood that collectively minimise the average distance and the neighbourhood offset is probably NP hard (it looks similar to the Travelling Salesman Problem - TSP). *A priori* we are only able to solve it with an heuristic. The approach used here is the most simple one, more clever algorithms are possible and could be studied. A first step could be for the cells I where the neighbourhood offset stays high to switch from finding the one cell J that minimises (VII.D.3) to finding the pair of cells J that minimises (VII.D.3). This might be reminiscent of the $k-opt$ algorithms in the TSP.
- Globally many aspects of this code are "heuristic" perhaps more formal/satisfactory definitions would be possible.

Part VIII

Appendices

D.1 Tensorial and index notations

In a fixed reference frame ($\underline{e}_1$, $\underline{e}_2$, $\underline{e}_3$), taking the Einstein convention on indices into account:

0^{th} **order tensor:** scalar T

1^{st} **order tensor:** vector $\underline{u} = u_i\,\underline{e}_i$

2^{nd} **order tensor:** matrix $\underline{\underline{\sigma}} = \sigma_{ij}\,\underline{e}_i \otimes \underline{e}_j$

3^{rd} **order tensor:** $\underline{\underline{\underline{a}}} = a_{ijk}\,\underline{e}_i \otimes \underline{e}_j \otimes \underline{e}_k$

$\vdots$

n^{th} **order tensor:** $a^{(n)} = a_{i_1 i_2 \cdots i_n}\,\underline{e}_{i_1} \otimes \underline{e}_{i_2} \otimes \cdots \otimes \underline{e}_{i_n}$

Operators	Symbols	Formulae	
tensorial product	$\otimes$	$\underline{u} \otimes \underline{u} = u_i u_j$	$\underline{e}_i \otimes \underline{e}_j$
		$a^{(n)} \otimes b^{(m)} = a_{i_1 \cdots i_n} b_{j_1 \cdots j_m}$	$\underline{e}_{i_1} \otimes \cdots \otimes \underline{e}_{i_n} \otimes \underline{e}_{j_1} \otimes \cdots \otimes \underline{e}_{j_m}$
dot product	.	$\underline{u} \cdot \underline{u} = u_i u_i$	
		$a^{(n)} \cdot b^{(m)} = a_{i_1 \cdots i_{n-1} k} b_{k j_2 \cdots j_m}$	$\underline{e}_{i_1} \otimes \cdots \otimes \underline{e}_{i_{n-1}} \otimes \underline{e}_{j_2} \otimes \cdots \otimes \underline{e}_{j_m}$
double dot	:	$\underline{\underline{\sigma}} : \underline{\underline{\sigma}} = \sigma_{ij}\sigma_{ij}$	
product		$a^{(n)} : b^{(m)} = a_{i_1 \cdots i_{n-2} k l} b_{k l j_3 \cdots j_m}$	$\underline{e}_{i_1} \otimes \cdots \otimes \underline{e}_{i_{n-2}} \otimes \underline{e}_{j_3} \otimes \cdots \otimes \underline{e}_{j_m}$
vectorial product	$\times$ *or* $\wedge$	$\underline{u} \times \underline{v} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}$	

Table D.1: Tensorial operators.

D.2 Differential operators and standard relationships

$$\begin{aligned} \text{div}\,(\underline{\text{curl}}\ \underline{u}) &= 0 & \underline{\text{curl}}\ (\underline{\nabla}\, T) &= 0 \\ \text{div}\,(\alpha \underline{u}) &= \alpha \text{div}\ \underline{u} + \underline{\nabla}\,\alpha . \underline{u} & \underline{\text{curl}}\ (\alpha \underline{u}) &= \alpha \underline{\text{curl}}\ \underline{u} + \underline{\nabla}\,\alpha \times \underline{u} \\ \text{div}\,(\underline{u} \times \underline{v}) &= \underline{v} . \underline{\text{curl}}\ \underline{u} - \underline{u} . \underline{\text{curl}}\ \underline{v} & \underline{\text{curl}}\ (\underline{\text{curl}}\ \underline{u}) &= \underline{\nabla}\,(\text{div}\ \underline{u}) - \underline{\Delta}\,\underline{u} \end{aligned}$$

$$\underline{\nabla}\,(\underline{u}.\underline{v}) = \left(\underline{\underline{\text{grad}}}\,\underline{u}\right).\underline{v} + \left(\underline{\underline{\text{grad}}}\,\underline{v}\right).\underline{u} + \underline{u} \times \underline{\text{curl}}\ \underline{v} + \underline{v} \times \underline{\text{curl}}\ \underline{u}$$

Operators	Symbols	Definitions	Cartesian formulae
gradient	∇	$\nabla^{(n+1)}\left[a^{(n)}\right] . \underline{\mathrm{d}x} = \mathrm{d}a^{(n)}\big\vert_{\underline{\mathrm{d}x}}$	$\underline{\nabla}\, T = \dfrac{\partial T}{\partial x_i}\underline{e}_i$
	or		$\underline{\underline{\nabla}}\,\underline{u} = \dfrac{\partial \underline{u}}{\partial x_j} \otimes \underline{e}_j = \dfrac{\partial u_i}{\partial x_j}\underline{e}_i \otimes \underline{e}_j$
	$\underline{\nabla}$		$\nabla^{(n+1)}\left[a^{(n)}\right] = \dfrac{\partial a^{(n)}}{\partial x_{i_{n+1}}} \otimes \underline{e}_{i_{n+1}}$
divergence	$\nabla.$	$\nabla^{(n-1)}.\left[a^{(n)}\right] = \nabla^{(n+1)}\left[a^{(n)}\right] : \underline{\underline{1}}$	$\underline{\mathrm{div}}\ \underline{u} = \dfrac{\partial u_i}{\partial x_i}$
	or		$\underline{\underline{\mathrm{div}}}\ \underline{\underline{\sigma}} = \dfrac{\partial \sigma_{ij}}{\partial x_j}\underline{e}_i$
	$\underline{\mathrm{div}}$		$\underline{\mathrm{div}}^{\,(n-1)}\left[a^{(n)}\right] = \dfrac{\partial a_{i_1 \cdots i_n}}{\partial x_{i_n}}\underline{e}_{i_1} \otimes \cdots \otimes \underline{e}_{i_{n-1}}$
Laplacian	Δ	$\Delta a^{(n)} = \underline{\mathrm{div}}^{\,(n)}\left\{\nabla^{(n+1)}\left[a^{(n)}\right]\right\}$	$\Delta\, T = \dfrac{\partial^2 T}{\partial x_i \partial x_i}$
	or		$\underline{\Delta}\,\underline{u} = \dfrac{\partial^2 \underline{u}}{\partial x_j \partial x_j} = \dfrac{\partial^2 u_i}{\partial x_j \partial x_j}\underline{e}_i$
	∇^2		$\Delta^{(n)} a^{(n)} = \dfrac{\partial^2 a^{(n)}}{\partial x_{i_{n+1}} \partial x_{i_{n+1}}} = \dfrac{\partial^2 a_{i_1 \cdots i_n}}{\partial x_{i_{n+1}} \partial x_{i_{n+1}}}\underline{e}_{i_1} \otimes \cdots \otimes \underline{e}_{i_n}$
rotational	$\underline{\nabla}\times$		$\underline{\mathrm{curl}}\ \underline{u} = \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} \end{pmatrix} \times \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3} \\ \frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1} \\ \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2} \end{pmatrix}$
	or		
	$\underline{\mathrm{curl}}$		

Table D.2: Differential operators.

Stokes Theorem

$$\int_{\mathcal{S}} \underline{\mathrm{curl}}\ \underline{u} \,.\, \underline{\mathrm{d}S} = \int_{\partial\mathcal{S}} \underline{u} \,.\, \underline{\mathrm{d}l}$$

where $\underline{\mathrm{d}S}$ is the outward surface element.

Divergence theorem [Green-Ostrogradski]

$$\int_{\Omega} \underline{\mathrm{div}}^{\,(n-1)}\left[a^{(n)}\right] \mathrm{d}\Omega = \int_{\partial\Omega} a^{(n)} \,.\, \underline{\mathrm{d}S}$$

Rotational theorem

$$\int_{\Omega} \underline{\mathrm{curl}}\ \underline{u}\, \mathrm{d}\Omega = \int_{\partial\Omega} \underline{u} \times \underline{\mathrm{d}S}$$

Leibnitz theorem

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega} a^{(n)}\, \mathrm{d}\Omega = \int_{\Omega} \frac{\partial a^{(n)}}{\partial t}\, \mathrm{d}\Omega + \int_{\partial\Omega} a^{(n)} \underline{v} \,.\, \underline{\mathrm{d}S}$$

where $\underline{v}$ is control volume Ω velocity.

Part IX

References

[Abdul-Razzak and Ghan, 2000] Abdul-Razzak, H. and Ghan, S. (2000). *A parametrisation of aerosol activation.2.Multiple aerosol types*, pages 105:6837–6844. J. Geophys Res.

[Amino et al., 2022] Amino, H., Flageul, C., Benhamadouche, S., Tiselj, I., Carissimo, B., and Ferrand, M. (2022). A time-staggered second order conservative time scheme for variable density flow. *International Journal for Numerical Methods in Fluids*.

[Atwater and Ball, 1978] Atwater, M. and Ball, J. (1978). A numerical solar radiation model based on standard meteorological observations. *Solar Energy*, 21:163–170.

[Aupoix and Spalart, 2003] Aupoix, B. and Spalart, P. (2003). Extensions of the spalart-allmaras turbulence model to account for wall roughness. *International Journal of Heat and Fluid Flow*, 24(4):454 – 462.

[Balvet et al., 2023] Balvet, G., Minier, J.-P., Henry, C., Roustan, Y., and Ferrand, M. (2023). A time-step-robust algorithm to compute particle trajectories in 3-d unstructured meshes for lagrangian stochastic methods. *Monte Carlo Methods and Applications*, 29(2):95–126.

[Barnes, 1964] Barnes, A. (1964). A technique for maximizing details in numerical weather map analysis. *Journal of Applied Meteorology*, 3:369–409.

[Beljaars and Holtslag, 1991] Beljaars, A. and Holtslag, A. (1991). Flux parameterization over land surfaces for atmospheric models. *J. Appl. Meteorol.*, 30:327–341.

[Betts, 1973] Betts, A. (1973). Non-precipitating cumulus convection and its parameterization. *Q. Jl R. Met. Soc.*, 99:178–196.

[Billard and Laurence, 2012] Billard, F. and Laurence, D. (2012). A robust $k-\varepsilon-\overline{v^2}/k$ elliptic blending turbulence model applied to near-wall, separated and buoyant flows. *International Journal of Heat and Fluid Flow*, 33(1):45 – 58.

[Bonelle, 2012] Bonelle, J. (2012). Une introduction aux méthodes "compatible discrete operators". Cas d'un problème elliptique. Technical Report H-I83-2012-00741-FR, EDF R&D.

[Bonelle, 2014] Bonelle, J. (2014). *Compatible Discrete Operator schemes on polyhedral meshes for elliptic and Stokes equations*. PhD thesis, Université Paris-Est.

[Bonelle et al., 2015] Bonelle, J., Di Pietro, D. A., and Ern, A. (2015). Low-order reconstruction operators on polyhedral meshes: application to compatible discrete operator schemes. *Comput. Aided Geom. Des.*, 35:27–41.

[Bonelle and Ern, 2014] Bonelle, J. and Ern, A. (2014). Analysis of Compatible Discrete Operator Schemes for Elliptic Problems on Polyhedral Meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(2):553–581.

[Bonelle et al., 2020] Bonelle, J., Ern, A., and Milani, R. (2020). Compatible discrete operator schemes for the steady incompressible Stokes and Navier–Stokes equations. In *International Conference on Finite Volumes for Complex Applications*, pages 93–101. Springer.

[Borghi and Dutoya, 1978] Borghi, R. and Dutoya, D. (1978). On the scale of the fluctuation in turbulent combustion. *17th Int.Symp. on Combustion*.

[Boucker and Mattéi, 2000] Boucker, M. and Mattéi, J. (2000). Proposition de modification des conditions aux limites de paroi turbulente pour le solveur commun dans le cadre du modèle $k-\varepsilon$ standard. EDF Report HI-81/00/019/A(in French).

[Bougeault, 1985] Bougeault, P. (1985). The diurnal cycle of the marine stratocumulus layer: a higher-order model study. *J. Atmos. Sci.*, 42:2826–2843.

[Bouzereau et al., 2007] Bouzereau, E., Musson-Genon, L., and Carissimo, B. (2007). On the definition of the cloud water content fluctuations and its effects on the computation of a second-order liquid water correlation. *J. Atmos. Sci*, 64:665–669.

[Businger et al., 1971] Businger, J., Wyngaard, J., Izumi, Y., and Bradley, E. (1971). Flux-profile relationships in the atmospheric surface layer. *J. Atmos. Sci*, 28:181–189.

[Cantin, 2016] Cantin, P. (2016). *Approximation of scalar and vector transport problems on polyhedral meshes*. PhD thesis, Université Paris-Est.

[Chaumerliac et al., 1987] Chaumerliac, N., Richard, E., Pinty, J.-P., and Nickerson, E. (1987). Sulfur scavenging in a mesoscale model with quasi-spectral microphysics. *J. geophys. Res.*, 92:3114–3126.

[Cheng and Brutsaert, 2005] Cheng, Y. and Brutsaert, W. (2005). Flux-profile relationships for wind speed and temperature in the stable atmospheric boundary layer. *Boundary-Layer Meteorol.*, 115:519–538.

[Chorin, 1968] Chorin, A. J. (1968). Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762.

[Chou and Arking, 1980] Chou, M. and Arking, A. (1980). Computation of infrared cooling rates in the water vapor bands. *J. Atmos. Sci*, 37:855–867.

[Cohard et al., 1998] Cohard, J., Pinty, J.-P., and Bedos, C. (1998). Extending twomey's analytical estimate of nucleated cloud droplet concentration from ccn spectra. *J. atmos. Sci.*, 55:3348–3357.

[Cohard and Pinty, 2000] Cohard, J.-M. and Pinty, J.-P. (2000). A comprehensive two-moment warm micro-physical bulk scheme. *Q. Jl. R. Met. Soc.*, 126:1815–1842.

[Colas et al., 2019] Colas, C., Ferrand, M., Hérard, J.-M., Latché, J.-C., and Le Coupanec, E. (2019). An implicit integral formulation to model inviscid fluid flows in obstructed media. *Computers & Fluids*, 188:136–163.

[Cressman, 1959] Cressman (1959). An operational objective analysis system. monthly weather review. *Monthly Weather review*, 87:367–374.

[Curl, 1963] Curl, R. (1963). Dispersed phase mixing: theory and effects in simple reactors. *AIChE J.*, 9:175–181.

[Deardorff, 1978] Deardorff, J. (1978). Efficient prediction of ground surface temperature and moisture, with inclusion of a layer of vegetation. *J Geophys Res*, 83:1889–1903.

[Durbin, 1991] Durbin, P. (1991). Near-wall turbulence closure modeling without "damping functions". *Theoretical and Computational Fluid Dynamics*, 3(1):1–13.

[Escaich, 2011] Escaich, A. (2011). *Une description améliorée de la combustion turbulente dans les flammes de charbon pulvérisé*. PhD thesis, Université de Rouen.

[Ferrand, 2022] Ferrand, M. (2022). *Free-surface flow simulations with a Lagrangian and an Arbitrary Lagrangian–Eulerian methods*. Theses, École des Ponts ParisTech.

[Ferrand et al., 2014] Ferrand, M., Fontaine, J., and Angelini, O. (2014). An anisotropic diffusion finite volume algorithm using a small stencil. In *Finite Volumes for Complex Applications VII-Elliptic, Parabolic and Hyperbolic Problems: FVCA 7, Berlin, June 2014*, pages 577–585. Springer.

[Ferrand and Harris, 2021] Ferrand, M. and Harris, J. C. (2021). Finite volume arbitrary lagrangian-eulerian schemes using dual meshes for ocean wave applications. *Computers & Fluids*, 219:104860.

[Garratt, 1992] Garratt, J. (1992). *The Atmospheric Boundary Layer*. Cambridge University Press.

[Green, 1964] Green, A. (1964). Attenuation by ozone and the earth albedo in the middle ultraviolet. *Appl opt*, 3:203–208.

[Guermond and Minev, 2015] Guermond, J.-L. and Minev, P. D. (2015). High-order time stepping for the incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 37(6):A2656–A2681.

[Hicks, 1976] Hicks, B. (1976). Wind profile relationships from the "wangara" experiment'. *Quart. J. Roy. Meteorol. Soc*, 102:535–551.

[Holton, 1979] Holton, J. (1979). *An introduction to dynamic meteorology.* Academic Press.

[Iaccarino, 2001] Iaccarino, G. (2001). Predictions of a turbulent separated flow using commercial cfd codes. *Journal of Fluids Engineering*, 123:819.

[Jakirlic and Hanjalic, 2002] Jakirlic, S. and Hanjalic, K. (2002). A new approach to modelling near-wall turbulence energy and stress dissipation. *Journal of fluid mechanics*, 459(1):139–166.

[Jones and Launder, 1972] Jones, W. and Launder, B. (1972). The prediction of laminarization with a two-equation model of turbulence. *International Journal of Heat and Mass Transfer*, 15(2):301–314.

[Joseph et al., 1976] Joseph, J., Wiscombe, W., and Weinman, J. (1976). The delta-eddigton approximation for radiative flux transfer. *J.Atmos.Sci*, 33:2452–2459.

[Kobayashi and ???, 1976] Kobayashi, H. and ??? (1976). *16th Int.Symp. on Combustion*, pages 425–441.

[Lacis and Hansen, 1974] Lacis, A. and Hansen, J. (1974). A parameterization for the absorption of solar radiation in the earth's atmosphere. *J. Atmos. Sci.*, 31:118–133.

[Launder et al., 1975] Launder, B., Reece, G. J., and Rodi, W. (1975). Progress in the development of a reynolds-stress turbulence closure. *Journal of fluid mechanics*, 68(03):537–566.

[Launder and Spalding, 1974] Launder, B. and Spalding, D. (1974). The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269 – 289.

[Libby and Williams, 2000] Libby, P. and Williams, F. (2000). A presumed pdf analysis of lean partially premixed turbulent combustion. *Combust. Sci. Technol.*, 161:351–390.

[Liou, 2002] Liou, K.-N. (2002). An introduction to atmospheric radiation. *Academic Press.*, 84.

[Louis et al., 1982] Louis, J., Tiedke, M., and Geleyn, J. (1982). A short history of the pbl parameterization at ecmwf. *Proc. ECMWF Workshop on Planetary Boundary Layer Parameterization*, pages 59–79.

[Makke et al., 2016] Makke, L., Musson-Genon, L., Carissimo, B., Plion, P., Milliez, M., and Douce, A. (2016). A new method for fast computation of three dimensional atmospheric infrared radiative transfer in a non-scattering medium and application to dynamical simulation of radiation fog in a built environment. *J. Atmos. Sci.*, pages DOI:10–1175/JAS–D–15–0012.1.

[Manceau and Hanjalić, 2002] Manceau, R. and Hanjalić, K. (2002). Elliptic blending model: A new near-wall reynolds-stress turbulence closure. *Physics of Fluids*, 14:744.

[Milani, 2020] Milani, R. (2020). *Compatible Discrete Operator schemes for the unsteady incompressible Navier–Stokes equations.* PhD Thesis, Université Paris-Est.

[Milani et al., 2022] Milani, R., Bonelle, J., and Ern, A. (2022). Artificial compressibility methods for the incompressible Navier–Stokes equations using lowest-order face-based schemes on polytopal meshes. *Comput. Methods Appl. Math.*, 22(1):133–154.

[Musson-Genon, 1987] Musson-Genon, L. (1987). Numerical simulation of a fog event with a one-dimensional boundary layer model. *Mon. wea. Rev.*, 115:592–607.

[Musson-Genon, 1995] Musson-Genon, L. (1995). Comparison of different simple turbulence closures with a one-dimensional boundary layer model. *Mon. wea. Rev.*, 123:163–180.

[Musson-Genon et al., 2007] Musson-Genon, L., Dupont, E., and Wendum, D. (2007). Reconstruction of the surface-layer vertical structure from measurements of wind, temperature and humidity at two levels. *DOI 10.1007/s10546-007-9178-5*, 124:235–250.

[Nerisson, 2009] Nerisson, P. (2009). *Modélisation du Transfert des Aérosols dans un Local Ventilé.* PhD thesis, IRSN/INPT.

[Paltridge and C.M.R., 1974] Paltridge, G. and C.M.R., P. (1974). Radiative processes in meteorology and climatology. *ELSEVIER*, page 378.

[Parneix et al., 1996] Parneix, S., Laurence, D., and Durbin, P. (1996). Second moment closure analysis of the backstep flow database. In *Proc. 1996 Summer Program, Center for Turbulence Research, NASA Ames/Stanford Univ*, pages 47–62.

[Pielke, 1984] Pielke, R. (1984). *Mesoscale Meteorological Modeling.* Academic Press Inc.

[Ponnulakshmi et al., 2012] Ponnulakshmi, V., Mukund, V., Singh, D., Sreenivas, K., and Subramanian, G. (2012). Hypercooling in the nocturnal boundary layer: Broadband emissivity schemes. *J. Atmos. Sci.*, 69:2892–2905.

[Pruppacher and Klett, 2000] Pruppacher, H. and Klett, J. (2000). Microphysics of clouds and precipitation. atmospheric sciences library. *Kluwer Academic Publishers, Boston, MA,Second Revised and Enlarged Edition with an Introduction to Cloud Chemistry and Cloud Electricity.*

[Ribert et al., 2004] Ribert, G., Champion, M., and Plion, P. (2004). Modeling turbulent reactive flows with variable equivalence ratio: application to the calculation of a reactive shear layer. *Combust. Sci. Technol.*, 176:907–923.

[Robin and ???, 2005] Robin, V. and ??? (2005). Relevance of approximated pdf shapes for turbulent combustion modeling with variable equivalence ratio. *ICDERS.*

[Sapa, 2011] Sapa, B. (2011). *Contribution à l'extension d'un schéma incompressible pour la flammes à bas nombre de Froude - Pré-requis à la modélisation de l'incendie.* PhD thesis, Université de Poitiers.

[Sasamori, 1968] Sasamori, T. (1968). The radiative cooling calculation for application to general circulation experiments. *J. Appl. Meteor.*, 7:721–729.

[Sasamori, 1972] Sasamori, T. (1972). A linear harmonic analysis of atmospheric motion with radiative dissipation. *J. Meteor. Soc. Jap.*, 50:505–518.

[Shewchuk, 1999] Shewchuk, J. R. (1999). Lecture notes on delaunay mesh generation. http://www.cs.berkeley.edu/~jrs/meshf99/.

[Spalart and Allmaras, 1992] Spalart, P. R. and Allmaras, S. R. (1992). Slicing an ear using prune and search. *AIAA.*

[Spalding and ???, 1971] Spalding, D. and ??? (1971). Mixing and chemical reaction in steady confined turbulent turbulent flames. *13th Int.Symp. on Combustion*, pages 649–657.

[Speziale et al., 1991] Speziale, C. G., Sarkar, S., and Gatski, T. B. (1991). Modelling the pressure–strain correlation of turbulence: an invariant dynamical systems approach. *Journal of Fluid Mechanics*, 227(1):245–272.

[Stephens, 1984] Stephens, G. (1984). The parameterization of radiation for numerical weather prediction and climate models. *Mon. Wea. Rev*, 12:826–866.

[Stull, 1988] Stull, R. (1988). *An introduction to Boundary Layer Meteorology.* Atmospheric Sciences Library, Kluwer Academic Publishers.

[Theußl, 1998] Theußl, T. (1998). A review of two simple polygon triangulation algorithms. http://www.cg.tuwien.ac.at/~theussl/.

[Veyre et al., 1980] Veyre, P., Sommeria, G., and Fouquart, Y. (1980). Modélisation de l'effet des hétérogénéités du champ radiative infra-rouge sur la dynamique des nuages. *J. Rech. Atmos.*, 14:89–108.

[Yamamoto, 1962] Yamamoto, G. (1962). On the radiation chart. science reports of the tohoku university. *Ser. 5, Geophysics*, 4-1:9–23.

[Zaichik et al., 2004] Zaichik, L., Soloviev, S., Skibin, A., and Alipchenko, V. (2004). A diffusion-inertia model for predicting dispersion of low-inertia particles in turbulent flows. In *5th International Conference on Multiphase Flow - ICMF*, volume 220, pages 117–152.

[Zhang et al., 2001] Zhang, L., Gong, S., Padro, J., and Barrie, L. (2001). A size-segregated particle dry deposition scheme for an atmospheric aerosol module. *Atmos Environ*, 35:549–560.

[Zhang et al., 2014] Zhang, X., Musson-Genon, L., Dupont, E., Milliez, M., and Carissimo, B. (2014). On the influence of a simple microphysics parametrization on radiation fog modelling: a case study during parisfog. *Boundary-Layer Meteorol*, 151:293–315.