F A N T O M

Flow Analysis Numerical Techniques Organized Moduli

Ramon Codina

April 25, 2000

Overview

`Fantom` is a finite element code for the numerical simulation of incompressible flow problems. From the point of view of the physical problem, it is able to deal with nonlinear materials, thermally coupled flows, turbulent regimes and the existence of a free surface separating the fluid analyzed and air. Referring to the numerical techniques employed, both mixed interpolations (with continuous or discontinuous pressures) and stabilization techniques may be used. The time integration may be performed by several finite difference schemes.

Contents

1 Introduction

1.1 What is `Fantom` ?

`Fantom` is a Fortran 77 finite element code for the numerical simulation of incompressible flow problems. Its main features from the point of view of the simulations it can perform are:

- ✰ **Nonlinear materials**. A variety of generalized Newtonian constitutive models are implemented, such as the Power Law and Carreau's model. If needed, other constitutive laws can be easily introduced.
- ✰ **Turbulent flows.** Several turbulence models are available, including the k and k-ε models. Concerning the Reynolds stress tensor, it can be computed using the Boussinesq assumption or using an algebraic stress turbulence model.
- ✰ **Thermally coupled flows I: Boussinesq assumption.** The heat equation may be solved together with the incompressible Navier-Stokes equations and, in particular, flows driven by temperature gradients can be dealt with.
- ✰ **Thermally coupled flows II: temperature-dependent properties.** All the physical properties may depend on the temperature.
- ✰ **Flows in non-inertial frames of reference**. The frame of reference may undergo an arbitrary motion. In particular, rotating flows can be modeled.
- ✰ **Flows with a free surface**. A free surface fluid-air may exist and be tracked by the code.

From the numerical point of view, the most salient characteristics of `Fantom` are:

- ✌ **Linear, quadratic and cubic interpolations** are available for all the unknows of the code (velocity, pressure, temperature, turbulent variables).
- ✌ **Mixed velocity-pressure interpolations**, satisfying the inf-sup condition for incompressible flow, can be used for solving the Navier-Stokes equations. The pressure interpolation may be either continuous or discontinuous.
- ✌ **Penalty methods** can be employed to treat the incompressibility. These include classical penalty methods as well as augmented-Lagrangian type techniques.
- ✌ **Stabilized formulations**, both to allow equal velocity-pressure interpolation and to stabilize convection, are implemented. These are based on the algebraic version of the subgrid scale framework.
- ✌ **Shock capturing** techniques may be used in the presence of sharp gradients of the unknowns. They prevent the appearence of spurious overshoots and undershoots.

- **Several time integrators** are available, of first and second order, all based on finite difference schemes. The generalized trapezoidal rule is implemented for all the problems. For the Navier-Stokes equations, both single step and two-step schemes of second order may be used, with the possibility of treating the convective term explicitly.

An important point is that `Fantom` has *neither pre-process nor post-process*. Therefore, a pre-processor must be used to create the geometry and the boundary conditions and a post-processor is needed to visualize the results. Two companion codes, `Fantinp` and `Fantout`, are in charge of translating the data from the pre-processor to `Fantom` and from `Fantom` to the post-processor, respectively.

On the other hand, `Fantom` uses the library `MatMan` to deal with linear systems of equations. As `Fantom` itself, this library can be downloaded from my web site (http://www.rmee.upc.es/homes/codina).

1.2 Where does `Fantom` come from ? (acknowledgements)

I wrote the first version of `Fantom` in 1988, when I started to do the final year thesis of the degree of Civil Engineer at the Universitat Politècnica de Catalunya (UPC), in Barcelona. The starting point was the finite element code `Omega`, which was coordinated by Prof. M. Cervera. He actively helped me in developing `Fantom` and, in fact, suggested the name. That was my first contact with a finite element code and even the latest version of `Fantom` inherited part of the coding style of `Omega`.

By the end of 1989 I started to do the research leading to my doctoral degree, also at the UPC, under the supervision of Prof. E. Oñate. I decided to write a completely new version of `Fantom` , since the first one was excessively tight to the coding structure of solid mechanics' codes. Likewise, I wanted to incorporate many new features, such as thermally coupled flows and free surface tracking. Although most of the code was written from scratch (except of course for the routines to solve linear systems and basic element interpolation) I profited from many of the developments and stimulating ideas that Dr. M. Galindo had incorporated in `Omega`. Prof. M. Cervera also collaborated in the initial stages of this version.

The first version of `Fantom` lasted only one year, and the second one already more than three years... But that's nothing compared to the *third* version, about which this (sort of) manual is devoted. I began to write it in 1993, with the collaboration of O. Soto, who was starting his doctoral thesis at that time. It wasn't until the end on 1994 that this version was ripe enough to write a manual, but I didn't find the time until now, five years later...

By now, many people have used `Fantom.` In particular, it has been the code employed in several fluid mechanics projects undertaken by the International Center for Numerical Methods in Engineering (CIMNE), a research institute of the UPC directed by Prof. E. Oñate, where I have collaborated during these last years. Several people have introduced small modifications in the code as well, although I feel unable to give

a reliable list of them (U. Shäffer, P. Meaney and G. Houzeaux are among them). Nevertheless, none of these collaborations compare to those of the persons I've mentioned, especially Dr. O. Soto.

1.3 About this document

This document is not a *real manual*. Even though I shall refer to it as such, it lacks many of the characteristics of a true manual. It is rather a guide of how to use `Fantom` *by scientists or engineers*. Although most of the manuals for scientific codes often assume a deep theoretical background from the users, this is specially so for this one.

The bulk of this document is devoted to explain the data that has to be introduced in `Fantom's` data file. It also contains a section about the use of `Fantom` in a UNIX environment and some very simple numerical examples.

The scope of the code, both referring to the physical problems it can deal with and the numerical techniques employed, is explained just before describing a particular group data.

2 An overview of `Fantom's` data file

2.1 General remarks

In the following we shall briefly describe the instructions for the use of the code `Fantom`. It reads the data from an ascii file that must be created by the user. The main purpose of this document is to explain how to create this data file, and this is done in the following sections.

Before describing the commands of `Fantom's` data file, let us give first some general rules of how to create it. As it is explained in the following, the data file consists of several blocks. A template for all of them is presented in the corresponding section. To create the complete data file, the user must supply the values and commands indicated with lower case. All upper case words must be left as they are in the final data file. When there are several possibilities for a required command, they are written in lower case and one after the other separated by the symbol `&`. The user must introduce one of the options. Word commands are given within quotes, (`'' ''`), whereas numbers are not. When an array of numbers must be provided, with range from 1 to n, it is indicated as `(1:n)`.

The commands written between brackets [] are optional. Its meaning is explained in what follows.

First of all, and before indicating anything about the data, let us say a few things about the routine that reads these data. It has the following properties:

- ☞ Lines starting with the symbol "$" are skipped.
- ☞ The symbols "_", "=", ":" and the comma "," are also skipped.
- ☞ The symbol "/" is understood as the line continuation.
- ☞ Only the first FIVE characters of each word are read.
- ☞ It is case insensitive: lower and upper characters are not distinguished. However, a word of warning is needed: the internal translation from lower case commands to upper case MIGHT NOT work on Linux environments. In this case, the UPPER case must be used.
- ☞ Blanks are not considered.

This properties must be taken into account at the moment of reading the templates for the data file.

2.2 Blocks and sub-blocks of data

The first thing that the user must supply is the title of the problem after the word `FANTOM`. After that, there are several groups of commands. All of them have a starting sentence that identifies the group and must finish with the same sentence following the word `END_`. There are also several sub-groups of instructions that must be provided following the same syntax. The order of the commands within the groups is irrelevant.

Each group or block of commands may have sub-blocks. Again, the order of data within sub-blocks is irrelevant, but of course the command that is supposed to be within a particular block cannot be written anywhere else. To simplify the reading of the templates, blocks and sub-blocks of data are indented.

The complete list of blocks and sub-blocks of data is the following:

```
FANTOM ``title''
RUN_DATA
PHYSICAL_PROBLEM
    NAVIER_STOKES_EQUATIONS
    HEAT_EQUATION
    PROPERTIES
    FREE_SURFACE
MESH_DATA
    DIMENSIONS
    GEOMETRY
        ELEMENTS
        BOUNDARIES
        COORDINATES
        SKEW_SYSTEMS
FIXITY_CONDITIONS
    PRESCRIPTIONS
        INITIAL_&_DIRICHLET_CONDITIONS
        PERIODIC_&_SKEW_BOUNDARY_CONDITIONS
        NEUMANN_BOUNDARY_CONDITIONS
        OPERATIONS_ON_BOUNDARIES
        LOCAL_BASE_CONNEXIONS
    FUNCTION_DEFINITION
NUMERICAL_TREATMENT
    GENERAL
    NAVIER_STOKES_EQUATIONS
        SPATIAL_TREATMENT
        TEMPORAL_TREATMENT
        SOLUTION_PROCEDURE
    TURBULENCE_PARAMETERS_EQS
    HEAT_EQUATION
    FREE_SURFACE_TRACKING
OUTPUT_&_POST-PROCESS
```

Each group (or sub-group) of data is described in a section (or sub-section) of this document.

3 Title and run data

The first word `Fantom` MUST read is `FANTOM` and, right after this, the title of the problem.

The first block of data to be included after this heading is the `RUN_DATA`, which contains general information about the run. The structure of the heading and this first block of data is the following:

```
$**************************************************************************
FANTOM ``title''
$**************************************************************************
RUN_DATA
  RUN_TYPE:            ``new'' & ``restart'' & ``preliminary''
  CPU_LIMIT:             cpu
  LENGTH_OF_POINTERS:    lenpo
  CUSTOMER:            ``universal''
END_RUN_DATA
$**************************************************************************
```

The options in the `RUN_DATA` block and their meaning are:

RUN_TYPE: The possibilities for the type of run are:

- ☞ ***New***: The model to be run is new, it has not been run before.
- ☞ ***Restart***: The model has been run until a certain iteration of a certain time step, and the present run is a continuation of this.
- ☞ ***Preliminary***: The model to be run may be re-started in a future. `Fantom` has to be informed of this to create the adequate files from which the re-start run will get the information. The options ***restart*** and ***preliminary*** may be used at the same time.

CPU_LIMIT: Maximum CPU limit allowed. The run stops when the code estimates that a new iteration is not possible without trespassing this threshold. When a job is sent to a batch queue with a limited CPU time, it may be interesting to specify the CPU limit close to that of the queue, to avoid the killing of the job without any output of results.

LENGTH_OF_POINTERS: May be 4 or 8 (bytes), depending on the architecture of the computer on which `Fantom` runs.

CUSTOMER: To avoid the need for introducing all the data in some particular cases, a customer may be pre-defined. In this case, `Fantom` assumes a set of default data. For the general case, the customer ***universal*** has to be indicated.

4 Physical problem

This block of data has in turn four sub-blocks. The idea is to provide the physical setting of the problem that has to be solved. The structure of this blocks and their sub-blocks are:

```
$*************************************************************************
PHYSICAL_PROBLEM
$-------------------------------------------------------------------------
  NAVIER_STOKES_EQUATIONS
  HEAT_EQUATION
  PROPERTIES
  FREE_SURFACE
$-------------------------------------------------------------------------
END_PHYSICAL_PROBLEM
$*************************************************************************
```

The following sub-sections are concerned with the four sub-blocks of data in this group. Details of the possibilities are given in each sub-section.

4.1 Navier-Stokes equations

This section describes the possibilities available in `Fantom` concerning the Navier-Stokes equations. Since this is precisely the main problem solved by the code, this section defines its scope and limitations.

The equations describing the problem are

$$\rho\left[\partial_t \boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + 2\,\boldsymbol{\omega}\times\boldsymbol{u}\right] - 2\nabla\cdot\left[\boldsymbol{\tau}_{\mathrm{R}} + \mu\boldsymbol{\varepsilon}(\boldsymbol{u})\right] + \nabla p + \sigma\boldsymbol{u} + \rho\beta\boldsymbol{g}\vartheta = \rho\boldsymbol{f}, \quad (1)$$

$$\nabla\cdot\boldsymbol{u} = 0, \quad (2)$$

to be solved in $\Omega\times(0,t_{\mathrm{fin}})$, where $\Omega\subset\mathbb{R}^{n_{\mathrm{sd}}}$ ($n_{\mathrm{sd}}=2$ or 3 being the space dimension) is the computational domain and $[0,t_{\mathrm{fin}}]$ is the time interval to be considered. In (1)-(2), $\boldsymbol{u}$ denotes the velocity field, p is the pressure ϑ is the temperature, ρ is the density, μ is the viscosity, which may depend on the invariants of the symmetrical part of the velocity gradient $\boldsymbol{\varepsilon}(\boldsymbol{u})$, $\boldsymbol{\tau}_{\mathrm{R}}$ is the Reynolds stress tensor of turbulent flows, $\boldsymbol{\omega}$ is the velocity of rotation of the frame of reference (and thus $2\,\boldsymbol{\omega}\times\boldsymbol{u}$ is the Coriolis force), σ is the inverse of the permeability of the medium, β is the thermal expansion coefficient, $\boldsymbol{g}$ is the gravity acceleration vector and $\boldsymbol{f}$ is the vector of body forces. The density ρ is assumed constant to obtain equations (1)-(2).

In the most general case, the force vector $\boldsymbol{f}$ in (1) contains the acceleration terms coming from the temporal variation of the reference basis and the reference buoyancy forces from the Boussinesq assumption, that is

$$\boldsymbol{f} = \boldsymbol{g} + \boldsymbol{g}\beta\vartheta_0 - \boldsymbol{a}_{\mathrm{fr}} - \dot{\boldsymbol{\omega}}\times\boldsymbol{x} - \boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{x}). \quad (3)$$

In this equation, ϑ_0 is the reference temperature from which buoyancy forces are computed, $\boldsymbol{a}_{\mathrm{fr}}$ is the acceleration of the frame of the reference measured from an inertial system and expressed in the moving reference, and the vector of position in this moving reference has been denoted by $\boldsymbol{x} = (x_1, x_2, x_3) \equiv (x, y, z)$ in the 3D case. Here and below, a Cartesian coordinate system is assumed.

The rheological behavior may be that of a generalized Newtonian fluid. The options presently available are:

- **Power law**:

$$\text{If } I_2(\boldsymbol{\varepsilon}) = 0, \quad \mu = \mu_{\text{cut-off}}, \quad \text{otherwise:}$$
$$\mu = K_0 \left[4 I_2(\boldsymbol{\varepsilon})\right]^{(n-1)/2} \exp\left[\frac{b}{\vartheta}\right]. \tag{4}$$

- **Carreau's model**:

$$\text{If } I_2(\boldsymbol{\varepsilon}) = 0, \quad \mu = \mu_{\text{cut-off}}, \quad \text{otherwise:}$$
$$\mu = \mu_0 \left[1 + 4\lambda^2 I_2(\boldsymbol{\varepsilon})\right]^{(n-1)/2} \exp\left[\frac{b}{\vartheta}\right]. \tag{5}$$

- **Perzyna's mode** (with a Sellars-Tegart yield stress variation):

$$\text{If } I_2(\boldsymbol{\varepsilon}) = 0, \quad \mu = \mu_{\text{cut-off}}, \quad \text{otherwise:}$$
$$\mu = \frac{\sigma + \left[g_1 \sqrt{\left(\frac{4I_2}{3}\right)}\right]^{\frac{1}{m}}}{\sqrt{12 I_2}}, \tag{6}$$
$$\text{with} \qquad \sigma = \frac{1}{p_1} \log\left[\left(\frac{Z}{p_2}\right)^{p_3} + \sqrt{\left(\frac{Z}{p_2}\right)^{2p_3} + 1}\right] \tag{7}$$
$$\text{and} \qquad Z = \sqrt{\frac{4I_2}{3}} \exp\left[\frac{Q}{R\vartheta}\right]. \tag{8}$$

  Here, $I_2 \equiv I_2(\boldsymbol{\varepsilon})$ is the second invariant of the strain rate tensor, $I_2(\boldsymbol{\varepsilon}) := \boldsymbol{\varepsilon} : \boldsymbol{\varepsilon}/2$, the colon standing for the double contraction of second order tensors. The coefficients that have not been defined are experimental constants.

It remains to introduce the turbulence model to be used to complete the description of the Navier-Stokes equations solved. If fact, when a turbulence model is employed the resulting equations are called Reynolds averaged.

`Fantom` has two possibilities to model the Reynolds stress tensor:

- **Boussinesq assumption**: $\boldsymbol{\tau}_{\mathrm{R}} = \mu_t \boldsymbol{\varepsilon}(\boldsymbol{u})$, where $\mu_t = \rho \nu_t$ is the eddy viscosity.

- **Algebraic stress models (ASM)**: $\boldsymbol{\tau}_{\mathrm{R}}$ is an algebraic function of two parameters, k and ϕ. The former is always the turbulent kinetic energy, whereas the latter may vary according to the model.

In the first case, only an expression for μ_t is needed, whereas in the second the algebraic function needs to be specified as well. The possibilities considered by `Fantom` are:

- **Mixing length model**. In this case, only the Boussinesq assumption can be used. The model is defined by

$$\nu_t = C\ell^2\sqrt{2\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u})}, \tag{9}$$

  where C and ℓ are experimental constants.

- **Smagorinski's model**. Again, only the Boussinesq assumption can be used. The model is defined by

$$\nu_t = ch^2\sqrt{2\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u})}, \tag{10}$$

  where c is an experimental constant (usually taken close to 0.01) and h is the size of the finite element where ν_t is computed. This models depends thus on the discretization of the problem.

- k **model**. An additional differential equation for the turbulent kinetic energy, k, needs to be solved. This equation is

$$\frac{\partial k}{\partial t} + \boldsymbol{u}\cdot\nabla k = P_k - C_\mu\frac{k^{3/2}}{L} + \nabla\cdot\left(\frac{\nu_t}{\sigma_k}\nabla k\right) + \nu\Delta k, \tag{11}$$

  where C_μ, σ_k and L are experimental constants,

$$P_k = \frac{1}{\rho}\boldsymbol{\tau}_{\mathrm{R}} : \nabla\boldsymbol{u}, \tag{12}$$

  is the so called production term, and

$$\nu_t = \sqrt{k}L. \tag{13}$$

  In this case, both the Boussinesq assumption and algebraic stress models can be used. In the first case, the kinematic eddy viscosity is computed according to (13).

- k-ε **model**. Two additional differential equation have to be solved, one for the turbulent kinetic k and another for the rate of turbulent energy dissipation ε. The former is (11), whereas the latter is

$$\frac{\partial \varepsilon}{\partial t} + \boldsymbol{u} \cdot \nabla \varepsilon = C_{\varepsilon 1} \frac{\varepsilon}{k} P_k - C_{\varepsilon 2} \frac{\varepsilon^2}{k} + \nabla \cdot \left(\frac{\nu_t}{\sigma_\varepsilon} \nabla \varepsilon \right) + \nu \Delta \varepsilon, \tag{14}$$

where $C_{\varepsilon 1}$, $C_{\varepsilon 2}$ and σ_ε are experimental constants,

$$\nu_t = C_\mu \frac{k^2}{\varepsilon} \tag{15}$$

and L in (11) is given by

$$L = C_\mu \frac{k^{3/2}}{\varepsilon}. \tag{16}$$

The values normally adopted for the experimental constants of the k-ε model are given in Table 1.

C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	σ_k	σ_ε
0.09	1.44	1.92	1.0	1.3

Table 1: Constants for the k-ε model

- k-ω **model**. Two additional differential equation have to be solved, one for the turbulent kinetic k and another for an auxiliary variable ω. The former is (11), whereas the latter is

$$\frac{\partial \omega}{\partial t} + \boldsymbol{u} \cdot \nabla \omega = C_{\omega 1} \frac{\omega}{k} P_k - C_{\omega 2} \omega^2 + \nabla \cdot \left(\frac{\nu_t}{\sigma_\omega} \nabla \omega \right) + \nu \Delta \omega, \tag{17}$$

where $C_{\omega 1}$, $C_{\omega 2}$ and σ_ω are experimental constants,

$$\nu_t = \frac{k}{\omega} \tag{18}$$

and L in (11) is given by

$$L = \frac{k^{1/2}}{\omega}. \tag{19}$$

The previous variable ε is related to ω by

$$\varepsilon = C_\mu \omega k.$$

The values normally adopted for the experimental constants of the k-ω model are given in Table 2.

C_μ	$C_{\omega 1}$	$C_{\omega 2}$	σ_k	σ_ω
0.09	5/9	3/40	2.0	2.0

Table 2: Constants for the k-ω model

- **k-$k\tau$ model**. Two additional differential equation have to be solved, one for the turbulent kinetic k and another for an auxiliary variable $k\tau$. The former is (11), whereas the latter is

$$\frac{\partial(k\tau)}{\partial t} + \boldsymbol{u} \cdot \nabla(k\tau) = C_{\tau 1} \frac{(k\tau)}{k} P_k - C_{\tau 2} k + \nabla \cdot \left(\frac{\nu_t}{\sigma_\tau} \nabla(k\tau) \right) + \nu \Delta(k\tau), \quad (20)$$

where $C_{\tau 1}$, $C_{\tau 2}$ and σ_τ are experimental constants,

$$\nu_t = C_\mu k\tau \quad (21)$$

and L in (11) is given by

$$L = C_\mu \sqrt{k}\tau. \quad (22)$$

The values normally adopted for the experimental constants of the k-$k\tau$ model are given in Table 3.

C_μ	$C_{\tau 1}$	$C_{\tau 2}$	σ_k	σ_τ
0.09	0.173	0.225	1.46	10.8

Table 3: Constants for the k-$k\tau$ model

This completes the description of the possible turbulence variables, that is, the closure of the Reynolds averaged equations. As it has been mentioned, these variables can be used together with the Boussinesq assumption to compute the Reynolds stress tensor $\boldsymbol{\tau}_{\mathrm{R}}$ or an algebraic expression for it may be given. `Fantom` has two options implemented, although other models can be easily coded. These models are expressed below in terms of k and ε, although any of the previous closure models can be used. The algebraic stress models implemented in `Fantom` are:

- **An explicit Reynolds stress model**. The components of $\boldsymbol{\tau}_{\mathrm{R}}$, τ_{ij}, are given by

$$\frac{\tau_{ij}}{\rho} = \frac{6(1+\eta^2)\alpha_1 k}{3+\eta^2+6\zeta^2\eta^2+6\zeta^2} \times \left[S^*_{ij} + (S^*_{ik}W^*_{kj} + S^*_{jk}W^*_{ki}) - 2(S^*_{ik}S^*_{kj} - \frac{1}{3} S^*_{kl}S^*_{kl}\delta_{ij}) \right] \quad (23)$$

where $\alpha_1 = (C_2 - \frac{4}{3})/(C_3 - 2)$,

$$\begin{aligned}
S^*_{ij} &= \frac{1}{2} g\tau (2 - C_3) S_{ij} \\
W^*_{ij} &= \frac{1}{2} g\tau (2 - C_4) \left[W_{ij} + \left(\frac{C_4 - 4}{C_4 - 2} \right) e_{mji} \omega_m \right] \\
\eta &= (S^*_{ij} S^*_{ij}), \quad \zeta = (W^*_{ij} W^*_{ij})
\end{aligned}$$

and

$$g = \left(\frac{1}{2} C_1 + \frac{P_k}{\varepsilon} - 1 \right)^{-1}, \quad \tau = \frac{k}{\varepsilon}.$$

Here, S_{ij} and W_{ij} are the components of the symmetric and skew-symmetric part of the vlocity gradient (the former is denoted by $\boldsymbol{\varepsilon}$ above):

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \qquad W_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right).$$

In (23) P_k is the production term (12), ω_m is the m-th component of the angular velocity $\boldsymbol{\omega}$ and e_{mji} are the components of the permutation 'tensor'. The cosntants of the model are

$$C_1 = 6.80, \quad C_2 = 0.36, \quad C_3 = 1.25, \quad C_4 = 0.40. \tag{24}$$

The parameter g can also be prescribed in this model. It is normally set to $g = 0.233$. Contrary to models based on the Boussinesq assumption, this one is able to take into account the effects of rotation.

- **An implicit Reynolds stress model**. The tensor $\boldsymbol{\tau}_{\mathrm{R}}$ is given by the implicit equation:

$$\frac{\boldsymbol{\tau}_{\mathrm{R}}}{\rho} = \frac{2}{3} k \boldsymbol{I} \left[1 - \frac{(1 - c_2)\frac{P_k}{\varepsilon}}{\frac{P_k}{\varepsilon} - (1 - c_1)} \right] + \frac{(1 - c_2)}{\frac{P_k}{\varepsilon} - (1 - c_1)} \frac{k}{\varepsilon} (\boldsymbol{P}^* - \frac{2}{3} P_k \boldsymbol{I}) \tag{25}$$

where $\boldsymbol{I}$ is the identity tensor and

$$\boldsymbol{P}^* = \frac{1}{\rho} \left(\boldsymbol{\tau}_{\mathrm{R}} \cdot \nabla \boldsymbol{u} + (\boldsymbol{\tau}_{\mathrm{R}} \cdot \nabla \boldsymbol{u})^t \right). \tag{26}$$

The constants of this model are c_1 and c_2. `Fantom` computed only the deviatoric part of $\boldsymbol{\tau}_{\mathrm{R}}$, assuming that the volumetric component can be included in the pressure. This must be taken into account at the moment of interpreting the results, especially the outflow stresses.

The description of the turbulence models completes the statement of the physical problem is what concerns the Navier-Stokes equations. All the possibilities described heretofore are defined in the sub-block corresponding to the definition of the Navier-Stokes equations within the physical problem, which is:

```
$----------------------------------------------------------------------
  NAVIER_STOKES_EQUATIONS:    ``on''  & ``off'' & ``used''
    TEMPORAL_DERIVATIVE:      ``on''  & ``off''
    CONVECTIVE_TERM:          ``off'' & ``non_conservative'' &
                              ``conservative'' & ``skew_symmetric''
    VISCOUS_TERM:             ``off''         & ``laplacian_form'' &
                              ``divergence_form''
    AXES_ROTATION:            NORM: omega, OX: ox, OY: oy, OZ: oz
    ROTATION_TIME_FUNCTION:   ``parabolic'' & ``periodic'' & ``external'',
                              PARAMETERS: param(1:5)
    AXES_VELOCITY:            NORM: v, VX: vx, VY: vy, VZ: vz
    VELOCITY_TIME_FUNCTION:   ``parabolic'' & ``periodic'' & ``external'',
                              PARAMETERS: param(1:5)
    BODY_FORCES:              ``on'' & ``off'',
                              GRAVITY: g, GX: gx, GY: gy, GZ: gz
    ATMOSFERIC_STRESS:        ``on'' & ``off'',
                              REAL_DENSITY: rho1, XR: x, YR: y, ZR: z,
                              AIR_DENSITY:  rho2, XA: x, YA: y, ZA: z,
    BOUSSINESQ_COUPLING:      ``on'' & ``off'',
                              VOL_EXPANSION: beta, REF_TEMP: T_a,
                              GRAVITY: g
    LAW_VISCOSITY:            ``constant'' & ``power_law'' & ``carreau'' &
                              ``perzyna''
    TURBULENCE_MODEL :        ``laminar''   & ``mixing_length'' &
                              ``k_model''   & ``k_tau_model''   &
                              ``k_e_model'' & ``k_w_model'',
                              ``boussinesq_assumption'' &
                              ``algebraic_stress''
  END_NAVIER_STOKES_EQUATIONS
$----------------------------------------------------------------------
```

Let us explain the meaning of the commands of this sub-block. First of all, observe that the header of the block itself has three options, that correspond to the following:

- ☞ ***On***: The Navier-Stokes equations are solved in the run.
- ☞ ***Off***: The Navier-Stokes equations are neither solved nor used.
- ☞ ***Used***: Even though the Navier-Stokes equations are not solved, the velocity is used. This option is normally used when only the heat equation with a convective term needs to be solved. The program has to be informed that there is a velocity to be used, which must be coded by the user.

The options within the `NAVIER_STOKES_EQUATIONS` block and their meaning are:

TEMPORAL_DERIVATIVE: Their existence or not is controlled by

- ☞ ***On***: The temporal derivative in (1) is taken into account.
- ☞ ***Off***: $\partial_t \boldsymbol{u} \equiv \mathbf{0}$ in (1) (stationary Navier-Stokes equations). Note that this option does not mean that the rest of the problems are also stationary.

CONVECTIVE_TERM: This command determines the expression for the convective term in (1). The possibilities are:

- ☞ ***Off***: $\boldsymbol{u} \cdot \nabla \boldsymbol{u} \equiv \mathbf{0}$ in (1). This corresponds to the Stokes flow.
- ☞ ***Non_conservative***: The form $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ is used in (1).
- ☞ ***Conservative***: The form $\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = \boldsymbol{u} \cdot \nabla \boldsymbol{u} + (\nabla \cdot \boldsymbol{u})\boldsymbol{u}$ is used in (1). This coincides with $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ for the continuous problem, since $\nabla \cdot \boldsymbol{u} = 0$. However, for the discrete problem there might be some differences. The name *conservative* comes from the fact that this is the form of the convective term arising in the so called conservation form of the Navier-Stokes equations.
- ☞ ***Skew_symmetric***: The form $\boldsymbol{u} \cdot \nabla \boldsymbol{u} + \frac{1}{2}(\nabla \cdot \boldsymbol{u})\boldsymbol{u}$ is used in (1). Again, this coincides with $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ for the continuous problem. The interesting property of this expression for the convective term is that it makes the trilinear form appearing in the weak statement of the equations skew-symmetric in its last two arguments, even for non divergence free velocities.

VISCOUS_TERM: This command determines the expression for the physical viscous term in (1). The possibilities are:

- ☞ ***Off***: $2\nabla \cdot [\mu \boldsymbol{\varepsilon}(\boldsymbol{u})] \equiv \mathbf{0}$ in (1), which then is the Euler equation. Nevertheless, turbulent models are still allowed in this case.
- ☞ ***Laplacian_form***: The expression $\mu \Delta \boldsymbol{u}$ is used in (1) instead of $2\nabla \cdot [\mu \boldsymbol{\varepsilon}(\boldsymbol{u})]$. For divergence free velocities and constant viscosities we have that $\mu \Delta \boldsymbol{u} = 2\nabla \cdot [\mu \boldsymbol{\varepsilon}(\boldsymbol{u})]$. However, both expressions give rise to different natural boundary conditions. When the true stress is of interest on boundaries, the following form is to be preferred:
- ☞ ***Divergence_form***: The form $2\nabla \cdot [\mu \boldsymbol{\varepsilon}(\boldsymbol{u})]$ is used in (1).

AXES_ROTATION: The angular velocity $\boldsymbol{\omega}$ in (1) and (3) is assumed to be of the form $\boldsymbol{\omega} = \boldsymbol{\omega}_0 f(t)$, where $\boldsymbol{\omega}_0$ is a constant vector and $f(t)$ is a time dependent function. The data to be provided here are:

- ☞ **NORM**: Norm of the vector $\boldsymbol{\omega}_0$.
- ☞ **OX**: x-component of $\boldsymbol{\omega}_0$. It is divided by $|\boldsymbol{\omega}_0|$ internally by `Fantom.` Therefore, this and the following components can correspond to any vector in the direction of $\boldsymbol{\omega}_0$.
- ☞ **OY**: y-component of $\boldsymbol{\omega}_0$.
- ☞ **OZ**: z-component of $\boldsymbol{\omega}_0$.

ROTATION_TIME_FUNCTION: This determines the function $f(t)$ introduced above. The possibilities are:

- ☞ ***Parabolic***: $f(t) = A_1 t^2 + A_2 t + A_3$, for $A_4 \leq t \leq A_5$.
- ☞ ***Periodic***: $f(t) = A_1 \cos(A_2 t + A_3)$, for $A_4 \leq t \leq A_5$.
- ☞ ***External***: $f(t)$ is read from an external file. This must be coded by the user.
- ☞ **PARAMETERS**: The five parameters introduced above, A_i, $i = 1 : 5$, must be written here.

AXES_VELOCITY: The axes of the frame of reference are not only allowed to rotate, but also to have a velocity. This is described in this block. Observe that the temporal derivative of this velocity is precisely the acceleration $\boldsymbol{a}_{\mathrm{fr}}$ in (3). Similarly to the previous case, the velocity is assumed to be of the form $\boldsymbol{v}_{\mathrm{fr}} = \boldsymbol{v}_{\mathrm{fr},0} g(t)$.

- ☞ **NORM**: Norm of the vector $\boldsymbol{v}_{\mathrm{fr},0}$.
- ☞ **VX**: x-component of $\boldsymbol{v}_{\mathrm{fr},0}$. It is divided by $|\boldsymbol{v}_{\mathrm{fr},0}|$ internally by `Fantom.` Therefore, this and the following components can correspond to any vector in the direction of $\boldsymbol{v}_{\mathrm{fr},0}$.
- ☞ **VY**: y-component of $\boldsymbol{v}_{\mathrm{fr},0}$.
- ☞ **VZ**: z-component of $\boldsymbol{v}_{\mathrm{fr},0}$.

VELOCITY_TIME_FUNCTION: The description of $g(t)$ is:

- ☞ ***Parabolic***: $g(t) = B_1 t^2 + B_2 t + B_3$, for $B_4 \leq t \leq B_5$.
- ☞ ***Periodic***: $g(t) = B_1 \cos(B_2 t + B_3)$, for $B_4 \leq t \leq B_5$.
- ☞ ***External***: $g(t)$ is read from an external file. This must be coded by the user.
- ☞ **PARAMETERS**: The five parameters introduced above, B_i, $i = 1 : 5$, must be written here.

BODY_FORCES: This command defines the vector $\boldsymbol{f}$ in (3). Since the motion of the frame of reference and the Boussinesq assumption are defined in other blocks of commands, only the gravity acceleration needs to be specified.

- ☞ ***On***: $\boldsymbol{f}$ in (3) is not zero.
- ☞ ***Off***: $\boldsymbol{f} \equiv 0$ in (3). In particular, the Boussinesq coupling and the non-inertial forces are unactivated as well.
- ☞ **GRAVITY**: Norm of the vector $\boldsymbol{g}$ in (3).
- ☞ **GX**: x-component of $\boldsymbol{g}$. It is divided by $|\boldsymbol{g}|$ by `Fantom.` Therefore, this and the following components can correspond to any vector in the direction of $\boldsymbol{g}$.
- ☞ **GY**: y-component of $\boldsymbol{g}$.
- ☞ **GZ**: z-component of $\boldsymbol{g}$.

ATMOSFERIC_STRESS: Very often the computational domain Ω in fluid mechanics problems contains outflow boundaries, where boundary conditions try to emulate the far field behavior of the flow variables. If there is no gravity, a widely used outflow condition is zero stress. However, in some cases it is necessary to include gravity effects, such as in free surface flows. The zero stress condition has to be replaced by a hydrostatic traction vector. This is the option allowed by this command.

- ☞ ***On***: The hydrostatic atmosferic stress is activated.
- ☞ ***Off***: No atmosferic stress is applied (default).
- ☞ **REAL_DENSITY**: Density ρ to compute the hydrostatic traction on the fluid analyzed. For generality, it may be different from ρ in (1).
- ☞ **XR**: x-coordinate of the reference point to compute the atmosferic stress on the fluid analyzed.
- ☞ **YR**: y-coordinate of the reference point to compute the atmosferic stress on the fluid analyzed.
- ☞ **ZR**: z-coordinate of the reference point to compute the atmosferic stress on the fluid analyzed.
- ☞ **AIR_DENSITY**: Density ρ to compute the hydrostatic traction on the air when there is a free surface to be tracked.
- ☞ **XA**: x-coordinate of the reference point to compute the atmosferic stress on the air.
- ☞ **YA**: y-coordinate of the reference point to compute the atmosferic stress on the air.
- ☞ **ZA**: z-coordinate of the reference point to compute the atmosferic stress on the air.

BOUSSINESQ_COUPLING: The parameter β in (1) and (3) determines whether the Boussinesq coupling is taken into account or not. The commands and data to define it are:

- ☞ ***On***: The Boussinesq coupling is activated.
- ☞ ***Off***: The Boussinesq coupling is not taken into account.
- ☞ **VOL_EXPANSION**: Parameter β in (1) and (3).
- ☞ **REF_TEMP**: Reference temperature ϑ_0 in (3).
- ☞ **GRAVITY**: Norm of $\boldsymbol{g}$ multiplying $\beta\vartheta_0$ in (3). For generality, it may be different from the gravity norm introduced before, for example to allow to consider only the Boussinesq contribution in $\boldsymbol{f}$.

LAW_VISCOSITY: This command defines the constitutive model employed. Any generalized Newtonian fluid behavior can be easily implemented. The options currently available are:

- ☞ ***Constant***: μ is constant (Newtonian fluid).
- ☞ ***Power_law***: The Power Law model (4) is used.
- ☞ ***Carreau***: Carreau's model (5) is used.
- ☞ ***Perzyna***: Perzyna's model (6) is used.

TURBULENCE_MODEL: The turbulence model employed is defined here. Observe that TWO commands must be provided (they are separated by a comma in the above template). The first is the type of closure equations, and the second the expression adopted for the Reynolds stress tensor $\boldsymbol{\tau}_{\mathrm{R}}$. The possibilities are:

- ☞ ***Laminar***: The flow is laminar, and thus $\boldsymbol{\tau}_{\mathrm{R}} = \mathbf{0}$.
- ☞ ***Mixing_length***: Either the mixing length model (9) or Smagorinski's model (10) is used as closure. The particular model is determined by the turbulence parameters (see below).
- ☞ ***k_model***: The k model is used as closure.
- ☞ ***k_tau_model***: The k-τ model is used as closure.
- ☞ ***k_e_model***: The k-ε model is used as closure.
- ☞ ***k_w_model***: The k-ω model is used as closure.
- ☞ ***Boussinesq_assumption***: The Boussinesq assumption is employed to compute the Reynols stress tensor $\boldsymbol{\tau}_{\mathrm{R}}$.
- ☞ ***Algebraic_stress***: $\boldsymbol{\tau}_{\mathrm{R}}$ is computed from the algebraic law, either the explicit or the implicit. The particular model is determined by the turbulence parameters (see below).

4.2 Heat equation

The heat equation solved by `Fantom` is the following:

$$\rho C_p[\partial_t \vartheta + \boldsymbol{u} \cdot \nabla \vartheta] - \nabla \cdot (k \nabla \vartheta) = S, \tag{27}$$

to be solved again in $\Omega \times (0, t_{\text{fin}})$. In (27) ϑ is the temperature, k is the thermal conduction coefficient, C_p the specific heat at constant pressure (and therefore ρC_p is the heat capacity) and S is the heat source. The only type of heat source considered by `Fantom` is a constant and that coming from the Joule effect. In this case, S in (27) is given by

$$S = S_0 + 2\mu \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u}), \tag{28}$$

with S_0 a constant and the colon standing for the double contraction of second order tensors.

The block of commands describing the heat equation to be solved by `Fantom` is the following:

```
$------------------------------------------------------------------------
  HEAT_EQUATION:           ``on'' & ``off'' & ``used''
    TEMPORAL_DERIVATIVE:   ``on'' & ``off''
    CONVECTIVE_TERM:       ``on'' & ``off''
    SOURCE_TERMS:          ``on'' & ``off'', CONSTANT: & ``Joule_effect''
  END_HEAT_EQUATION
$------------------------------------------------------------------------
```

The options of the header of the block itself are:

☞ ***On***: The heat equation is solved in the run.

☞ ***Off***: The heat equation is neither solved nor used.

☞ ***Used***: Even though the heat equation is not solved, the temperature is used. This option must be used for example when the physical parameters depend on the temperature. The program has to be informed that there is a temperature to be used, which must be coded by the user.

The commands within the group are:

TEMPORAL_DERIVATIVE: Their existence or not is controlled by

☞ ***On***: The temporal derivative in (27) is taken into account.

☞ ***Off***: $\partial_t \vartheta \equiv 0$ in (27) (stationary heat equation). Note that this option does not mean that the rest of the problems are also stationary.

CONVECTIVE_TERM: This command determines the expression for the convective term in (1). The possibilities are:

- ☞ ***Off***: $\boldsymbol{u} \cdot \nabla \vartheta \equiv 0$ in (27).
- ☞ ***On***: The convective term in (27) is activated.

SOURCE_TERM: This defines the heat source to be used. The possibilities are:

- ☞ ***Off***: $S = 0$ in (27).
- ☞ ***On***: $S \neq 0$ in (27).
- ☞ **CONSTANT**: Value of S_0 in (28).
- ☞ ***Joule_effect***: The term $2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u})$ is included in (28). If this instruction is not given, S is taken constant ($S = S_0$).

4.3 Physical properties

After having defined the Navier-Stokes and the heat equations, this sub-block of data provides the physical properties and experimental constants introduced above (some of the turbulence parameters are also referred to the numerical treatment). The template of this sub-block is:

```
$-----------------------------------------------------------------------
  PROPERTIES
    DENSITY:               (...)
    VISCOSITY:             (...)
    POROSITY:              (...)
    SPECIFIC_HEAT:         (...)
    DIFFUSION:             (...)
    TURBULENCE_PARAMETERS: (...)
  END_PROPERTIES
$-----------------------------------------------------------------------
```

The symbol (...) has been used here to denote a sequence of values that depend on the models introduced before. These are:

DENSITY: Value of the fluid density ρ. Only one parameter is needed here.

VISCOSITY: Parameters to determine the viscosity. According to the model employed, they are:

- ☞ ***Constant***: (...) $= \mu$, constant of (1).
- ☞ ***Power_law***: (...) $= (K_0, n, b, \mu_{\mathrm{cut-off}})$ of (4) (4 parameters).
- ☞ ***Carreau***: (...) $= (\mu_0, \lambda, n, b, \mu_{\mathrm{cut-off}})$ of (5) (5 parameters).

☞ ***Perzyna***: (...) $= (p_1, p_2, p_3, Q, R, g_1, m, \mu_{\text{cut-off}})$ of (6) (8 parameters).

POROSITY: Coefficient σ in (1), which is the inverse of the permeability of the medium. Only one parameter is needed here.

SPECIFIC_HEAT: Coefficient C_p in (27). Only one parameter is needed here.

DIFFUSION: Coefficient k in (27). Only one parameter is needed here.

TURBULENCE_PARAMETERS: The turbulence parameters to be introduced depend on the turbulence model. In this case, they are not only physical constants of the turbulence models, but also numerical parameters and parameters to determine the boundary conditions (see below). They are:

☞ **(1)**: It depends on the closure model:

- **Mixing length model:** Constant C in (9).
- **Smagorinski's model:** Constant c in (10).
- **Rest of models:** Constant C_μ in the k-equation (11). Note that this value changes from model to model.

☞ **(2)**: It is the mixing length of the problem. In particular, it determines whether the mixing length model itself or Smagorinski's model is being used. There are three possibilities:

- $\mathbf{turbu(2) = L > 0}$. In this case, L is the mixing length. Its meaning varies according to the closure model used:
  - ***Mixing_length***: Constant L to be used in (9).
  - ***k_model***: Constant L to be used in (13).
  - ***Two-equation models***: Value of L to obtain the Dirichlet boundary condition for the second turbulence parameter (see below).
- $\mathbf{turbu(2) = 0}$. In this case, the mixing length is h, the element size. This option CAN ONLY BE USED when the closure is ***Mixing_length*** (in fact it corresponds to Smagorinski's model, in this case).
- $\mathbf{turbu(2) = -1}$. The mixing length is computed from Nikuradse's model. In this case, L is obtained from the expression

$$\frac{L}{R} = 0.14 - 0.018\left(1 - \frac{y}{R}\right)^2 - 0.06\left(1 - \frac{y}{R}\right)^4,$$

where R is a characteristic length of the domain, computed by the code as the maximum distance from the nodal points to the boundary, and y is the distance from a point to the boundary. Observe that L changes from point to point in this case.

- ☞ **(3)**: Value of the constant σ_k in (11). It is only used when the clossure is the k-model or two-equation models.
- ☞ **(4)**: Value of the constant σ_ϕ for two-equation closure models, where ϕ may be either ε, ω or τ.
- ☞ **(5)**: Value of the constant $C_{\phi,1}$ for two-equation closure models, where ϕ may be either ε, ω or τ.
- ☞ **(6)**: Value of the constant $C_{\phi,2}$ for two-equation closure models, where ϕ may be either ε, ω or τ.
- ☞ **(7)**: Parameter to fix the boundary condition to integrate up to the wall for the k-ε and k-ω clossure models. For the first model it is only a numerical threshold to fix ε, whereas for the latter it is the rugosity coefficient k_R^+ (see (37)).
- ☞ **(8)**: Value of δ, distance to the wall when a wall-law is used (see (36)).
- ☞ **(9)**: If $\delta = 0$, there is the possibility of using a fixed friction parameter given here. A stress $\boldsymbol{t} = -F^{-1}\rho|\boldsymbol{u}|\boldsymbol{u}$ is prescribed, where F is **turbu(9)**.
- ☞ **(10)**: Factor for the boundary condition for the turbulent kinetic energy k. It is prescribed to $k = c_{\mathrm{bc}}|\boldsymbol{u}|^2$ where the velocity $\boldsymbol{u}$ is given (see below). The value of c_{bc} is **turbu(10)**.
- ☞ **(11)**: It depends on the ASM model used:

  Implicit ASM: Constant c_1.

  Explicit ASM: Constant C_1 (= 6.80).
- ☞ **(12)**: It depends on the ASM model used:

  Implicit ASM: Constant c_2.

  Explicit ASM: Constant C_2 (= 0.36).
- ☞ **(13)**: It depends on the ASM model used:

  Implicit ASM: Maximum number of iterations allowed to solve (25) iteratively. This parameter is an integer, but may be introduced as a floating number.

  Explicit ASM: Constant C_3 (= 1.25).
- ☞ **(14)**: It depends on the ASM model used:

  Implicit ASM: Error to solve (25) iteratively.

  Explicit ASM: Constant C_4 (= 0.40).

☞ **(15)**: This parameter is used internally by the code to decide whether to use the explicit or the implicit ASM:

Implicit ASM: 0.

Explicit ASM: Constant g $(= 0.233)$.

☞ **(16)**: In order to improve the convergence rate of algebraic stress models, some iterations of the underlying two-equation model using the Boussinesq assumption can be performed before resorting to the ASM. These number of iterations has to be introduced here. Obviously, this parameter is an integer, but may be introduced as a floating number.

4.4 Free surface tracking

`Fantom` uses the VOF (volume of fluid) method to track the free surface between the fluid analyzed and air, whenever it exists. The basic idea of this method is to solve the Navier-Stokes equations in the whole computational domain, and assume that the definition of the position of the fluid front only affects the physical properties. Let π be any of these, i.e., density (ρ), viscosity (μ), specific heat (C_p) or conduction coefficient (k). We will have that

$$\pi(\boldsymbol{x},t) = \begin{cases} \pi_{\text{fluid}}(\boldsymbol{x},t) & \text{if } \boldsymbol{x} \in \Omega_t \\ \pi_{\text{air}} & \text{if } \boldsymbol{x} \in \Omega \setminus \Omega_t \end{cases}$$

where the domain where the fluid analyzed flows is

$$\Omega_t := \{\boldsymbol{x} \in \Omega \mid \psi(\boldsymbol{x},t) \geq \psi_c\}, \tag{29}$$

and $\psi(\boldsymbol{x},t)$ is a smooth function the isovalue ψ_c of which determines the position of the free surface. Since this is a material surface, the function ψ is the solution of the following problem:

$$\partial_t\psi + \boldsymbol{u} \cdot \nabla\psi = 0 \quad \text{in } \Omega,\ t \in (0, T_{\text{fin}}), \tag{30}$$

$$\psi = \bar{\psi} \quad \text{on } \Gamma_{\text{inf}},\ t \in (0, T_{\text{fin}}), \tag{31}$$

$$\psi(\boldsymbol{x},0) = \psi_0(\boldsymbol{x}) \quad \text{in } \Omega,$$

$$\Gamma_{\text{inf}} := \{\boldsymbol{x} \in \partial\Omega \mid \boldsymbol{u} \cdot \boldsymbol{n} < 0\}.$$

The initial condition ψ_0 must be chosen in order to define the initial position of the fluid to be analyzed. The boundary condition $\bar{\psi}$ determines whether fluid enters or not through a certain point of the inflow boundary. If it does, a value $\bar{\psi} \geq \psi_c$ is to be prescribed (for example, $\bar{\psi} = 1$); otherwise, $\bar{\psi} < \psi_c$.

The sub-block of the data file where the free surface tracking is defined has the following data:

```
$------------------------------------------------------------------------
  FREE_SURFACE:            ``on'' & ``off'' & ``used''
    TIME_EVOLUTION:        ``steady'' & ``moving''
    AIR_PROPERTIES:        DENSITY: rho, VISCOSITY: mu,  SPECIFIC_HEAT: Cp
                           DIFFUSION: k
    FILL_UP_TO:            perce (%)
    CONTINUE_AFTER_FILLED: ``yes'' & ``no''
  END_FREE_SURFACE
$------------------------------------------------------------------------
```

The options of the header of the block itself are:

☞ ***On***: The problem has a free surface to be tracked.

☞ ***Off***: No free surface exists.

☞ ***Used***: Even though the free surface is not tracked, it exists and the function ψ is used. The program has to be informed that there is a free surface to be used; the function ψ must be coded by the user.

The commands within the group are:

TIME_EVOLUTION: There are two options:

- ☞ ***Steady***: The free surface does not evolve in time.
- ☞ ***Moving***: The free surface evolves in time.

AIR_PROPERTIES: The physical properties used for 'air' (that is, the environment fluid) have to be introduced here. They are assumed to be constant:

- ☞ **DENSITY**: Air density.
- ☞ **VISCOSITY**: Air (dynamical) viscosity.
- ☞ **SPECIFIC_HEAT**: Air specific heat.
- ☞ **DIFFUSION**: Air thermal conduction (or heat diffusion).

FILL_UP_TO: A common problem of practical interest in which a free surface needs to be tracked is the filling of a cavity. If only a certain percentage of it has to be filled, this can be indicated here (**perce**). The run will stop after this percentage of the cavity has been occupied by the fluid analyzed.

CONTINUE_AFTER_FILLED: This command is only taken into account if the value **perce** introduced before is 100. There are two options:

- ☞ ***Yes***: The run continues after the domain is filled, until the final time specified by the user (this is explained later on).
- ☞ ***No***: The run stops once the cavity is completely filled up.

5 Mesh data

This section contains the definition of the domain Ω where the problem has to be solved and its discretization. Most of the data to be introduced here are standard in finite element analyses. These data is organized into two main sub-blocks as follows:

```
$**************************************************************************
MESH_DATA
$--------------------------------------------------------------------------
   DIMENSIONS
   GEOMETRY
$--------------------------------------------------------------------------
END_MESH_DATA
$**************************************************************************
```

The commands and sub-blocks of the sub-blocks `DIMENSIONS` and `GEOMETRY` are explained next.

5.1 Dimensions

This sub-block contains only commands. Its template is:

```
$--------------------------------------------------------------------------
   DIMENSIONS
     NODAL_POINTS:      npoin
     ELEMENTS:          nelem
     NODES_PER_ELEMENT: nnode,  ``brick'' & ``simplex''
     SPACE_DIMENSION:   ``two'', ``two axisymmetric'', ``three''
     BOUNDARIES:        nboun
     SKEW_SYSTEMS:      nfaco
     LOC_BASE_CHANGES:  nbcon
   END_DIMENSIONS
$--------------------------------------------------------------------------
```

The meaning of the data to be introduced is:

NODAL_POINTS: Total number of nodal points of the finite element mesh.

ELEMENTS: Total number of elements of the finite element mesh.

NODES_PER_ELEMENT: Number of nodes per element. All the elements in `Fantom` must be of the same type. After the number of nodes, one of the following words must be specified:

☞ ***brick***: The elements are bricks, that is, quadrilaterals in 2D and hexahedra in 3D.

☞ ***simplex***: The elements are simplices, that is, triangles in 2D and tetrahedra in 3D.

The possible combinations are indicated in Table 4. It is observed that `Fantom` admits standard interpolations of degree 1, 2 and 3, as well as some non-standard interpolations (bubbles, serendipid elements).

SPACE_DIMENSION: The number of space dimensions must be given here. To allow axisymmetric configurations (case in which `Fantom` works in cylindrical coordinates), these numbers must be spelled out. The possibilities are:

☞ ***two***: 2D problem.

☞ ***two axisymmetric***: Axisymmetric configuration.

☞ ***three***: 3D problem.

BOUNDARIES: Some of the edges (in 2D) or faces (in 3D) of the elements will belong to the physical boundary $\partial\Omega$. The number of these on which `Fantom` needs to operate must be given here. These operations may arise because of several reasons; they are explained later on.

SKEW_SYSTEMS: Sometimes it is interesting to express the boundary conditions for the velocity in reference systems others than the global one. The number of such systems (defined as explained later) must be given here.

LOC_BASE_CHANGES: `Fantom` has the option of computing internally a local reference system for each point in which the first vector is the normal and the other two (in 3D) are tangent to the boundary. Sometimes it is interesting to use for a node not its reference system, but that of another node. This may happen for example for corner nodes, in which the normal computed by `Fantom` is not physically relevant. The number of such reference system interchanges must be given here.

5.2 Geometry

The topology of the finite element mesh must be indicated in this sub-block, which has in turn four sub-blocks:

```
$-----------------------------------------------------------------------------
   GEOMETRY, [‘‘interpolate’’], [‘‘check’’]
     ELEMENTS
     BOUNDARIES
```

NDIME	Nodes and type	Description
2	3 Simplex	Linear triangle (P_1)
2	4 Simplex	Linear triangle with bubble
2	6 Simplex	Quadratic triangle (P_2)
2	7 Simplex	Quadratic triangle with bubble
2	10 Simplex	Cubic triangle (P_3)
2	4 Brick	Bilinear quadrilateral (Q_1)
2	5 Brick	Bilinear quadrilateral with bubble
2	8 Brick	Serendipid quadrilateral
2	9 Brick	Biquadratic quadrilateral (Q_2)
2	16 Brick	Triquadratic quadrilateral (Q_3)
3	4 Simplex	Linear tetrahedron (P_1)
3	5 Simplex	Linear tetrahedron with bubble
3	10 Simplex	Quadratic tetrahedron (P_2)
3	11 Simplex	Quadratic tetrahedron with central bubble
3	15 Simplex	Quadratic tetrahedron with central and lateral bubbles
3	20 Simplex	Cubic tetrahedron (P_3)
3	8 Brick	Trilinear hexahedron (Q_1)
3	9 Brick	Trilinear hexahedron with bubble
3	20 Brick	Serendipid hexahedron
3	27 Brick	Biquadratic hexahedron (Q_2)
3	64 Brick	Triquadratic hexahedron (Q_3)

Table 4: Element types in `Fantom`

```
    COORDINATES
    SKEW_SYSTEMS
  END_GEOMETRY
$--------------------------------------------------------------------------
```

The header of the sub-block `GEOMETRY` has two options, BOTH OPTIONAL. These are:

☞ ***interpolate***: This command is used only for higher order interpolations. If it exists, only the coordinates of the element vertices (corresponding to a linear or multilinear interpolation) must be given, the rest will be interpolated by `Fantom`.

☞ ***check***: If this command is given, `Fantom` performs some internal checks to verify that the topology of the finite element mesh is correct. This option can be used when the code stops with an error that might be due to wrong mesh data.

5.2.1 Element nodal connections

The format of this sub-group of data is:

```
ELEMENTS
  ielem   lnods(1:nnode)
END_ELEMENTS
```

It is seen that this sub-block is a list of integers with the following meaning:

☞ **ielem**: Element number.

☞ **lnods**: List of element connections, that is, list of nodes belonging to the element. `lnods(inode) = ipoin` means that node number `inode` of element `ielem` is `ipoin`.

5.2.2 Boundary nodal connections

The format of this sub-group of data is:

```
BOUNDARIES
  iboun   lnodb(1:nnodb)  [ielem]
END_BOUNDARIES
```

It contains the information about the faces (in 3D) or edges (in 2D) of the elements on which `Fantom` needs to operate (explained later on). The structure is the same as the list of element connections. The meaning of the integers to be introduced is:

☞ **iboun**: Boundary number (face or edge).

☞ **lnodb**: List of boundary connections, that is, list of nodes belonging to the boundary. `lnodb(inodb) = ipoin` means that node number `inodb` of boundary `iboun` is `ipoin`. The dimension `nnodb` is computed internally by `Fantom` from `nnode` (for example, `nnodb = 3` if `nnode = 9` in 2D, that is, the element is biquadratic).

☞ **ielem**: Element number to which the boundary belongs. Observe that this number is optional. If it is given, CPU time is saved, since otherwise `Fantom` needs to compute it internally. If the mesh generator employed is able to produce this information, it is recommended to supply it to `Fantom.`

5.2.3 Coordinates

The format of this sub-group is:

```
COORDINATES
  ipoin   coord(1:ndime)
END_COORDINATES
```

The meaning of this list of data is obvious:

☞ **ipoin**: Point number (integer).

☞ **coord**: Coordinates of `ipoin` (floats).

5.2.4 Skew systems of reference

As it has been mentioned, sometimes it is interesting to express the boundary conditions for the velocity in reference systems others than the global one. These systems are specified here. The format of this sub-group of data is:

```
SKEW_SYSTEMS
  ifaco   skcos(1:ndime,1:ndime)
END_SKEW_SYSTEMS
```

The meaning of this list of data is:

☞ **ifaco**: Number of the system of reference.

☞ **skcos**: Matrix of cosine directors of the reference system. Vectors expressed in the reference system `ifaco` will be referenced to the global (Cartesian) system when multiplied by `skcos`. This matrix must be introduced BY ROWS. Therefore, in 2D the list of numbers to be introduced is `skcos(1,1)`, `skcos(1,2)`, `skcos(2,1)`, `skcos(2,2)`.

6 Fixity conditions

This block contains the definition of the boundary conditions for the problem to be solved. It has the following two sub-blocks of data:

```
$***************************************************************************
FIXITY_CONDITIONS
$---------------------------------------------------------------------------
   PRESCRIPTIONS
   FUNCTION_DEFINITION
$---------------------------------------------------------------------------
END_FIXITY_CONDITIONS
$***************************************************************************
```

These blocks are explained after describing the boundary conditions to be applied for the different problems.

6.1 Physical boundary conditions

6.1.1 Navier-Stokes equations

The boundary condition that lead to a well-posed problem for the Navier-Stokes equations written in the $\boldsymbol{u}$-p formulation employed by `Fantom` can be obtained by integrating by parts the viscous and pressure terms in (1). If the divergence form is used for the viscous term, this leads to the boundary integral

$$\int_{\partial\Omega} \boldsymbol{v} \cdot [\boldsymbol{n} \cdot (-p\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}))] \,\mathrm{d}\,\Gamma, \tag{32}$$

where $\boldsymbol{v}$ is the velocity test function, $\boldsymbol{I}$ the identity tensor and $\boldsymbol{n}$ the unit exterior normal to $\partial\Omega$. Let us split this boundary as

$$\partial\Omega = \text{ closure of } \{\Gamma_D \cup \Gamma_N \cup \Gamma_M \cup \Gamma_O \cup \Gamma_{P_m} \cup \Gamma_{P_s}\}.$$

The boundary conditions to be prescribed in each component of this split are:

- On Γ_D: $\boldsymbol{u}(\boldsymbol{x},t) = \bar{\boldsymbol{u}}(\boldsymbol{x},t)$, given. This is the standard Dirichlet type boundary condition. Since $\boldsymbol{u}$ is a nodal variable of the problem, this condition is specified ON THE NODES of the boundary (see below). The test function $\boldsymbol{v}$ vanishes on Γ_D and therefore the contribution from this part of the boundary to (32) is zero.

- On Γ_N: $\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{n} \cdot (-p\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})) = \bar{\boldsymbol{t}}(\boldsymbol{x},t)$. This is the standard Neumann type boundary condition. The contribution from Γ_N to (32) is $\int_{\Gamma_N} \boldsymbol{v} \cdot \bar{\boldsymbol{t}} \,\mathrm{d}\,\Gamma$. This condition involves derivatives of $\boldsymbol{u}$, which may be discontinuous. It is prescribed on the element boundaries.

- On Γ_M: A combination of Dirichlet and Neumann type boundary condition is often needed, prescribing the normal component of the velocity and the tangential stresses: $\boldsymbol{u} \cdot \boldsymbol{n} = \bar{u}_n$, $\boldsymbol{n} \cdot \boldsymbol{\sigma} \cdot \boldsymbol{g}_1 = \bar{t}_1$, $\boldsymbol{n} \cdot \boldsymbol{\sigma} \cdot \boldsymbol{g}_2 = \bar{t}_2$, where ectors $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ (for the three-dimensional case) span the space tangent to Γ_M. The way `Fantom` operates is to prescribe any component of the velocity field and, for the free components, prescribe the traction. A particular case of this type of boundary is the prescription of the wall law for turbulent flows described below.

- On Γ_O: Very often the computational domain Ω has an outflow that is obtained from an artificial cut of an unbounded domain. The condition to be prescribed on such outflows is controversial. One of the possibilities is the "do-nothing" boundary condition, which consists of evaluating the integral in (32) on the outflow Γ_O. The continuous problem is ill-posed if this is done, but this option yields sometimes very good results for the discrete finite element problem.

- On $\Gamma_{P_m} \cup \Gamma_{P_s}$: Periodicity conditions. Suppose that Γ_{P_s} is obtained from Γ_{P_m} through a translation followed by a rotation of matrix $\boldsymbol{R}$. Let $\boldsymbol{x}_m$ be a point in Γ_{P_m} and $\boldsymbol{x}_s$ the image in Γ_{P_s} through the motion. The periodicity conditions between Γ_{P_m} (master face) and Γ_{P_s} (slave face) are: $\boldsymbol{u}(\boldsymbol{x}_s, t) = \boldsymbol{R}\boldsymbol{u}(\boldsymbol{x}_m, t)$ and $\boldsymbol{t}(\boldsymbol{x}_s, t) = -\boldsymbol{R}\boldsymbol{t}(\boldsymbol{x}_m, t)$, where $\boldsymbol{t}$ is the traction vector on the boundary.

The boundary conditions involving the velocity field may be specified either in the global Cartesian reference system or in a local system associated to a node under consideration.

6.1.2 Turbulence parameters

Boundary conditions have to be specied for the turbulent kinetic energy k when it is used in the closure model. Likewise, if two-equation models are employed, boundary conditions need to be given also for the second turbulent variable, ϕ. However, not all these conditions are given by the user, but they are partly selected internally by the code.

For the case in which the velocity is prescribed, k is also prescribed (Dirichlet type of boundary condition). The value of the prescription is

$$k = c_{\mathrm{bc}} u^2, \tag{33}$$

where c_{bc} is a constant and u is the Euclidean norm of the velocity. The constant c_{bc} is approximately taken between 0.003 and 0.01. When k is prescribed, so is ϕ if a two-equation model is used. The prescription for it is of the form

$$\phi = f(k, L), \tag{34}$$

where L is the mixing length and the function f depends on the model and is determined from (16) for the k-ε model, from (19) for the k-ω model, and from (22) for the k-$k\tau$ model. This length L is a result of the model in the interior of the domain, but must be given on the boundary where Dirichlet conditions are fixed so as to determine the boundary condition (34) (parameter **turbu(2)**).

Equations (33) and (34) give the boundary conditions on the part of the boundary where $\boldsymbol{u}$ is prescribed, that is, on Γ_D. On Γ_N the boundary conditions to be used are

$$\frac{\partial k}{\partial n} = 0, \quad \frac{\partial \phi}{\partial n} = 0,$$

where $\partial/\partial n$ is the normal derivative on the boundary of Ω. The same boundary condition is prescribed on Γ_M except when a wall law is prescribed there. In this case, the traction on the boundary is given by

$$\boldsymbol{t} = -\rho \frac{U_*^2}{|\boldsymbol{u}|}\boldsymbol{u}, \tag{35}$$

where ρ is the fluid density and U_* is the solution of the nonlinear equation

$$\frac{|\boldsymbol{u}|}{U_*} = \frac{1}{\kappa}\log\left(\frac{U_*\delta}{\nu}\right) + C, \tag{36}$$

with $\kappa = 0.41$ (von Kármán constant), $C = 5.5$ and where δ is the distance from the wall at which the velocity is evaluated.

When a wall law is prescribed for the velocity (that is, for the Navier-Stokes equations), k and ϕ are prescribed to

$$\begin{aligned}
k &= \frac{U_*^2}{\sqrt{C_\mu}}, \\
\varepsilon &= \frac{U_*^3}{\kappa\delta} \qquad \text{for the } k\text{-}\varepsilon \text{ model}, \\
\omega &= \frac{U_*}{\sqrt{C_\mu}\kappa\delta} \qquad \text{for the } k\text{-}\omega \text{ model}, \\
k\tau &= \frac{U_*\kappa\delta}{C_\mu} \qquad \text{for the } k\text{-}k\tau \text{ model},
\end{aligned}$$

and therefore Γ_M is a part of the boundary where also Dirichlet boundary conditions are prescribed for k and ϕ.

The use of the wall law (35) implies that the domain where the equations are solved starts at a distance δ from the boundary. However, it is also possible to integrate the turbulence equations up to the boundaries. The conditions to be prescribed are:

- $k = 0$. This case offers no difficulty.

- If the k-ε model is used, damping functions have to be introduced to be able to prescribe ε on the boundaries (low Re models). These ARE NOT implemented in `Fantom` and in this case $\varepsilon \to \infty$ as the boundary is approached. However, there is the possibility of prescribing ε to a large value, given by $\varepsilon = K\varepsilon_0$, where ε_0 is the maximum value of ε on Γ_D obtained from (34) (identified by the code) and K is a cut-off constant introduced by the user in the parameter **turbu(7)**.

- If the k-ω model is used, ω is prescribed to

$$\omega = \frac{U_*^2}{\nu} S_R$$
$$S_R := \begin{cases} \left(\frac{50}{k_R^+}\right)^2 & \text{if } k_R^+ < 25 \\ \frac{100}{k_R^+} & \text{if } k_R^+ \geq 25 \end{cases} \tag{37}$$

  where k_R^+ is a user defined parameter (**turbu(7)**) that measures the wall roughness.

- If the k-$k\tau$ model is used boundary conditions offer no problem: $k = 0$ and $k\tau = 0$.

6.1.3 Heat equation

The boundary conditions that can be prescribed for the temperature are similar to those of the Navier-Stokes equations. Let us split now the boundary as

$$\partial\Omega = \text{ closure of } \{\Gamma_{D'} \cup \Gamma_{R'} \cup \Gamma_{O'}\}.$$

The boundary conditions to be prescribed in each component of this split are:

- On $\Gamma_{D'}$: $\vartheta(\boldsymbol{x}, t) = \bar{\vartheta}(\boldsymbol{x}, t)$, given. This is the standard Dirichlet type boundary condition. Since ϑ is a nodal variable of the problem, this condition is specified ON THE NODES of the boundary (see below).

- On $\Gamma_{R'}$: $k\frac{\partial\vartheta}{\partial n} = \alpha(\vartheta - \vartheta_{\text{env}}) + q$. This is a Robin type boundary condition, a particular case of which is the standard Neumann condition (when $\alpha = 0$). In this expression, k is the thermal conduction coefficient (not to be confused with the turbulent kinetic energy!), α is the surface conduction, ϑ_{env} a reference (environment) temperature and q the prescribed heat flux. This condition involves derivatives of ϑ, which may be discontinuous. It is prescribed on the element boundaries.

- On $\Gamma_{O'}$: As for the Navier-Stokes equations, one of the possible conditions to be prescribed on outflow boundaries is the "do-nothing" condition, which consists of evaluating the integral of the heat flux $k\frac{\partial\vartheta}{\partial n}$ multiplied by the temperature test function on the outflow boundary $\Gamma_{O'}$. Again, the continuous problem is ill-posed if this is done, but this option may work for the discrete finite element problem.

6.1.4 Free surface tracking

The equation for the function that defines the position of the free surface is hyperbolic (pure convection). The problem is well posed if the unknown ψ is prescribed as indicated in (31). The physical meaning of this presciption has been already described.

6.2 Block of prescriptions

It contains the definition of the boundary conditions except for the time evolution. This sub-block has one single command and five internal groups of data, as displayed below:

```
$-------------------------------------------------------------------------
   PRESCRIPTIONS
     FIX_ONE_PRESSURE:             ''on'' & ''off''
     INITIAL_&_DIRICHLET_CONDITIONS, [''constant'']
     PERIODIC_&_SKEW_BOUNDARY_CONDITIONS
     NEUMANN_BOUNDARY_CONDITIONS, [''constant'']
     OPERATIONS_ON_BOUNDARIES
     LOCAL_BASE_CONNEXIONS
   END_PRESCRIPTIONS
$-------------------------------------------------------------------------
```

The meaning of the single command is:

FIX_ONE_PRESSURE: There are two possibilities:

- ☞ ***on***: Very often the pressure for incompressible flows is determined up to a constant. This happens for example when all the boundary conditions are of Dirichlet type. `Fantom` prescribes internally the pressure of node 1 to zero if the option ***on*** is selected. This option IS MANDATORY if there are skew systems or tangent systems, but is identified by the code otherwise, that is, if neither skew nor tangent systems exist, `Fantom` is able to decide whether it is necessary to fix one pressure or not. Note that this problem does not exist when penalty methods are used (see next section), since in this case the pressure is always uniquely determined (it has zero mean for confined flows).
- ☞ ***off***: No pressure needs to be fixed.

The blocks of data within this sub-block are described next:

6.2.1 Initial and Dirichlet conditions

This contains the codes to determine whether a node has a Dirichlet condition or not. The template is the following:

```
INITIAL_&_DIRICHLET_CONDITIONS [''constant'']
   ipoin  ix,iy,iz,it,is   vx, dx, vy, dy, vz, dz, t, dt, s, ds
END_INITIAL_&_DIRICHLET_CONDITIONS
```

The header of the block has one OPTIONAL command:

☞ ***constant***: If this command is given, it is assumed that the boundary conditions are constant in time.

The list of data to be introduced in this block is:

☞ **ipoin**: Number of nodal point. If can be preceeded by a $-$ sign. In this case, is is assumed that the boundary values for the velocity **vx, vy, vz** described next are given IN THE LOCAL FRAME OF REFERENCE computed by `Fantom.` This frame of reference consists of the normal (first component) and two vectors, in the 3D case, tangent to the boundary $\partial\Omega$. Since the user does not have control on these two vectors, tangent values **vy, vz** are not reliable. This does not happen in 2D, since the tangent to the surface is uniquely defined. According to this, the classical SLIP CONDITION for the velocity would correspond to `-ipoin 100 0.000 0.000 0.000` in a 3D problem with neither temperature nor free surface. The explanation of the codes and values used here is given right below.

☞ **ix**: Code for the boundary condition for the x-velocity. It does not need to be introduced if the Navier-Stokes equations are not ***on***. The possible values of this code are:

> **0**: The x-velocity is free.
>
> **1**: The x-velocity is prescribed (Dirichlet condition).
>
> **2**: The x-velocity is free for the air flow, but prescribed to zero for the fluid analyzed. This code is only used when there is a free surface to be tracked.
>
> **3**: A turbulent wall law is prescribed for the x-velocity (if the turbulence parameters 8 or 9 are positive), corresponding to a Dirichlet condition for the turbulence parameters (see the description of the physical problem). This code is only used when a turbulence model is activated.

A very important point is that the format of this code is INTEGER*1, and must be followed WITHOUT BLANKS by the codes described next (**iy, iz, it, is**).

☞ **iy**: Code for the boundary condition for the y-velocity. The same possibilities as for the **ix** code apply.

☞ **iy**: Code for the boundary condition for the z-velocity. The same possibilities as for the **ix** code apply. This code needs to be introduced only if the problem is 3D.

☞ **it**: Code for the boundary condition for the temperature. This code needs to be introduced only if the heat equation is solved in the run. The possibilities are:

> **0**: The temperature is free.

1: The temperature is prescribed (Dirichlet condition).

☞ **is**: Code for the boundary condition for the function ψ defining the free surface. This code needs to be introduced only if there is such a surface to be tracked. The possibilities are:

0: ψ is free. This means that the node may be wet or dry depending on whether the free surface reaches it or not.

1: Permanently wet or permanently dry node. This code must be used for example for the points where the injection of the fluid analyzed takes place.

2: Temporary dry node. This means that the node is initially dry, but when the free surface reaches it, it remains permanently wet.

☞ **vx**: x-velocity value. If **ix** $= 0$ it is taken as initial condition, if **ix** $= 1$ as Dirichlet boundary condition and is not used if **ix** $= 2$ or 3. This value is A FLOAT.

☞ **dx**: Function type for the Dirichlet boundary condition for the x-velocity component. This velocity is assumed to be of the form $v_x(t) = v_{x,0}f(t)$. The value $v_{x,0}$ is that introduced before and the type of function $f(t)$ is given here. The precise description of $f(t)$ is given in the next sub-block of data. This code is AN INTEGER and NEEDS NOT TO BE INTRODUCED if the command ***constant*** is introduced after the list heading.

☞ **vy**: y-velocity value. Same comments as for **vx** apply.

☞ **dy**: y-velocity Dirichlet function. Same comments as for **dx** apply.

☞ **vz**: y-velocity value. Same comments as for **vx** apply. Not needed if the problem is 2D.

☞ **dz**: y-velocity Dirichlet function. Same comments as for **dx** apply. Not needed if the problem is 2D.

☞ **t** : Temperature value. If **it** $= 0$ it is taken as initial condition and if **it** $= 1$ as Dirichlet boundary condition.

☞ **dt**: Function type for the Dirichlet boundary condition for the temperature. It is assumed to be of the form $\vartheta(t) = \vartheta_0 f(t)$. The value ϑ_0 is that introduced before and the type of function $f(t)$ is given here.

☞ **s** : Value for the function defining the free surface, ψ_0. If **is** $= 0$ it is taken as initial condition and if **is** $= 1$ as Dirichlet boundary condition. Recall that $\psi > \psi_0$ indicates wet node; otherwise the node is dry.

☞ **ds**: Function type for the Dirichlet boundary condition for ψ. It is assumed to be of the form $\psi(t) = \psi_0 f(t)$. The value ψ_0 is that introduced before and the type of function $f(t)$ is given here. This migth be useful to simulate the stop of fluid injection in a mould, for example.

6.2.2 Periodic conditions and skew reference systems

The information for the boundary conditions for the velocity given in systems different to the global one and the periodicity conditions is specified in the same block of data. It is a list of integers with the following structure:

```
PERIODIC_&_SKEW_BOUNDARY_CONDITIONS
   ipoin  integ
END_PERIODIC_&_SKEW_BOUNDARY_CONDITIONS
```

☞ **ipoin**: Number of nodal point.

☞ **integ**: Integer. The distinction between periodicity conditions and specification of skew systems is made according to its sign:

> > 0: The velocity conditions for `ipoin` are specified in the skew system number `integ`. The matrix that defines this skew system has been given in the sub-block `SKEW_SYSTEMS` of the block `GEOMETRY`.
>
> < 0: The point `ipoin` belongs to the slave face Γ_{P_s} and its master is −`integ`.

6.2.3 Neumann boundary conditions

The Neumann boundary conditions for the velocity and the Robin conditions for the temperature are described here. Both are defined on the *boundaries*, in the sense defined in the sub-block `BOUNDARIES` of the block `GEOMETRY`. The list of data to be introduced has the following structure:

```
NEUMANN_BOUNDARY_CONDITIONS, [``constant'']
   iboun  tx,nx ty,ny tz,nz al,na ta,nt q,nq ... (repeat until nnodb)
END_NEUMANN_BOUNDARY_CONDITIONS
```

The header of the block has one OPTIONAL command:

☞ ***constant***: If this command is given, it is assumed that the boundary conditions are constant in time.

The list of data to be introduced in this block is:

☞ **iboun**: Number of boundary.

☞ **tx**: x-component of the surface traction. This value is A FLOAT.

☞ **nx**: Function type for the Neumann boundary condition for the x-traction component. This component is assumed to satisfy (or to be approximated by) $t_x(t + \Delta t) = t_x(t) + f(t)\Delta t$. The value $t_x(0)$ is that introduced before and the type of function $f(t)$ is given here. The precise description of $f(t)$ is given in the next sub-block of data. This code is AN INTEGER and NEEDS NOT TO BE INTRODUCED if the command ***constant*** is introduced after the list heading.

☞ **ty**: y-traction value. Same comments as for **tx** apply.

☞ **ny**: y-traction Neumann function. Same comments as for **nx** apply.

☞ **tz**: z-traction value. Same comments as for **tx** apply. It does not have to be introduced if the problem is 2D.

☞ **nz**: z-traction Neumann function. Same comments as for **nx** apply. It does not have to be introduced if the problem is 2D.

☞ ... The sequence `tx,nx ty,ny tz,nz` must be introduced for all the nodes of the boundary `iboun`, corresponding to the values/functions at these nodes.

☞ **al**: Value of the coefficient α in the Robin condition for the temperature.

☞ **na**: Function type for α. This parameter is assumed to satisfy (or to be approximated by) $\alpha(t + \Delta t) = \alpha(t) + f(t)\Delta t$. The value $\alpha(0)$ is that introduced before and the type of function $f(t)$ is given here. The precise description of $f(t)$ is given in the next sub-block of data. This code is AN INTEGER and NEEDS NOT TO BE INTRODUCED if the command ***constant*** is introduced after the list heading.

☞ **ta**: Value of the reference temperature ϑ_{env} in the Robin condition for the temperature.

☞ **nt**: Function type for ϑ_{env}, analogous to `na`.

☞ **q**: Value of the heat flux q in the Robin condition for the temperature.

☞ **nq**: Function type for q, analogous to `na`.

☞ ... The sequence `al,na ta,nt q,nq` must be introduced for all the nodes of the boundary `iboun`, corresponding to the values/functions at these nodes. Observe that it is written in a DIFFERENT line than `tx,nx ty,ny tz,nz`.

6.2.4 Operations on boundaries

This sub-block of data contains additional information on the boundary conditions as well as the order to compute the force and torque exerted by the fluid on the boundary. The structure of this group is:

```
OPERATIONS_ON_BOUNDARIES
    iboun  iv  it  if  ip
END_OPERATIONS_ON_BOUNDARIES
```

The codes `iv, it, if` and `ip` may only take the values 0 and 1. The meaning of the list is the following:

☞ **iboun**: Number of boundary element.

☞ **iv**: If its value is 1, compute the integral given by (32). This means that the boundary element iboun is considered as part of the outflow boundary for the Navier-Stokes equations.

☞ **it**: If its value is 1, compute the boundary integral of the heat flux $k\frac{\partial\vartheta}{\partial n}$ multiplied by the temperature test function. This means that the boundary element iboun is considered as part of the outflow boundary for the heat equation.

☞ **if**: Add the contribution of the boundary element iboun to compute the force and torque exerted by the fluid. The latter is computed from the origin of coordinates.

☞ **ip**: Put the atmosferic stress as traction on this boundary. The data needed to compute this traction has been defined in the NAVIER_STOKES_EQUATIONS sub-block of data within the PHYSICAL_PROBLEM block.

Note that the four codes iv, it, if and ip must be supplied, even if they are meaningless for a particular run.

6.2.5 Local basis connections

Sometimes it is necessary to prescribe the velocity boundary conditions for a node in a reference system that is neither the global nor the tangent one. The general way to do this is specify the skew system if the geometry data and relate the node to the skew system in the PERIODIC_&_SKEW_BOUNDARY_CONDITIONS block. However, there is a case in which this can be simplified, and it is when the reference system for a node is the same as that of another node, which can be in particular the tangent one. This information can be provided here as follows:

```
LOCAL_BASIS_CONNECTIONS
  igive  itake
END_LOCAL_BASE_CONNEXIONS
```

The integers to be listed here are:

☞ **igive**: Node that *gives* the reference system.

☞ **itake**: Node that *takes* the reference system, i.e., whose reference system is the same as that of igive.

6.3 Block of function definition

Both in the definition of the Dirichlet and the Neumann boundary conditions the time variation has been related to a set of time dependent scalar functions. These functions are defined in this block, whose structure is:

```
$------------------------------------------------------------------------------
   FUNCTION_DEFINITION
      DIRICHLET_NUM 1: ``parabolic'' & ``periodic'', param(1:5)
      ...
      DIRICHLET_NUM n: ``parabolic'' & ``periodic'', param(1:5)
      NEUMANN_NUM 1:   ``parabolic'' & ``periodic'', param(1:5)
      ...
      NEUMANN_NUM m:   ``parabolic'' & ``periodic'', param(1:5)
   END_FUNCTIONS_DEFINITION
$------------------------------------------------------------------------------
```

There can be as many lines with `DIRICHLET_NUM` as Dirichlet functions have been introduced, and similarly for the `NEUMANN_NUM` lines. The options in both cases are:

- ***Parabolic***: $f(t) = A_3 t^2 + A_4 t + A_5$, for $A_1 \leq t \leq A_2$.
- ***Periodic***: $f(t) = A_3 \cos(A_4 t + A_5)$, for $A_1 \leq t \leq A_2$.
- **param(1:5)**: The five parameters introduced above, A_i, $i = 1 : 5$, must be written here.

Note that, even though the options are the same for the Dirichlet and the Neumann functions, they are used in a different way, as explained when describing the sub-blocks `INITIAL_&_DIRICHLET_CONDITIONS` and `NEUMANN_BOUNDARY_CONDITIONS`, respectively.

7 Numerical treatment

The block of numerical treatment has in turn several sub-blocks:

```
$**************************************************************************
NUMERICAL_TREATMENT
$--------------------------------------------------------------------------
  GENERAL
  NAVIER_STOKES_EQUATIONS
  TURBULENCE_PARAMETERS_EQS
  HEAT_EQUATION
  FREE_SURFACE_TRACKING
$--------------------------------------------------------------------------
END_NUMERICAL_TREATMENT
$**************************************************************************
```

The first sub-block corresponds to general numerical data, whereas the rest are specific of the different problems solved by `Fantom.` Each of these sub-blocks is described in the following sub-sections.

7.1 General

This block contains numerical data used for all the problems, as well as information that is not specific of any problem. The structure of this block is the following:

```
$--------------------------------------------------------------------------
  GENERAL:
    INTERPOLATION:              ``standard'' & ``hierarchic''
    INTEGRATION_RULE:           ``open''     & ``closed'',
                                QUADRATURE_POINTS: ngaus
    SMOOTHING_REQUIRED:         ``lumped''   & ``quadrature'' &
                                ``consistent''
    TIME_DATA:                  INITIAL_TIME:   t_ini,
                                FINAL_TIME:     t_fin,
                                TIME_STEP_SIZE: dtime
    ORDER_OF_BLOCK_ITERATIONS: ``momentum, energy, front_tracking''
    STAGGERED_PROBLEMS:         [momentum], [energy], [front_tracking]
    GOVERNING_PROBLEM:          ``momentum''      & ``energy'' &
                                ``front_tracking''
    OTHER_PARAMETERS:           NATURAL_LENGTH: hnatu
    FREE_PARAMETERS:            fpara(1:20)
  END_GENERAL
$--------------------------------------------------------------------------
```

The meaning of the different commands and their possibilities is described next:

INTERPOLATION: The finite element mesh has been already defined in the `GEOMETRY` block and, in particular, the number of nodes per element has been already fixed. The possibilities are given in Table 4. However, the interpolation (shape) functions may be of two types:

- ☞ ***Standard***: The degrees of freedom of the interpolation are the values of the function interpolated at the element nodes.
- ☞ ***Hierarchic***: This interpolation only makes sense for elements of degree 2 and 3. The basis for the second order interpolation is obtained by enriching the first order interpolation, and likewise the basis for the third order interpolation is obtained by enriching the second order one. The degrees of freedom of the interpolation are NOT any more the values of the function interpolated at the element nodes. Sometimes this option leads to better conditioned matrices of the final algebraic system, an important property when iterative methods are used to solve it.

INTEGRATION_RULE: This determines the numerical integration to be used to evaluate the integrals of the discrete weak form of the equations to be solved. A command and an integer must be provided:

- ☞ ***Open***: The integration points are located within the element domain
- ☞ ***Closed***: The integration points are nodes of the element.
- ☞ **QUADRATURE_POINTS**: Number of integration points. The possible integration rules available in `Fantom` are described in Tables 5 to 8.

SMOOTHING_REQUIRED: Some of the variables of interest in a flow analysis are obtained from the velocity derivatives (stresses, vorticity) and are therefore discontinuous. Likewise, the pressure interpolation may be chosen to be discontinuous by the user. For plotting purposes, it is interesting to approximate this discontinuous fields by continuous ones by projecting them onto the finite element interpolation space. This leads to a linear algebraic system whose matrix is the Gramm matrix of the finite element interpolation, $\boldsymbol{M}$. Depending on the treatment of this matrix, there are three possibilities:

- ☞ ***Lumped***: Once matrix $\boldsymbol{M}$ has been computed using the integration rule defined above for the run, it is approximated by a diagonal matrix by using the classical row sum lumping technique. Note that this option is only possible when the standard interpolation is used (not the hierarchical one) and may lead to poor results for highly distorted meshes. The next option is preferable, although it is slightly more expensive.

☞ ***Quadrature***: Regardless of the integration rule chosen by the user, matrix $\boldsymbol{M}$ is evaluated using a closed quadrature rule, with the integration points at the element nodes (for simplicial quadratic elements some of the weights are zero; to avoid zero entries in $\boldsymbol{M}$ they are replaced by the weights obtained by splitting the quadratic elements in a number of linear simplices). When the standard interpolation is used (not the hierarchical one), this leads to a diagonal matrix. This option is not allowed using the hierarchical interpolation.

☞ ***Consistent***: Matrix $\boldsymbol{M}$ is evaluated with the integration defined for the rest of the run. The L^2 projection onto the finite element space leads then to an algebraic system that is solved by the code. This option is the most expensive, but must be used when the interpolation is hierarchical.

TIME_DATA: This block contains the definition of the time interval of analysis and the time step size to be employed, which is CONSTANT during the whole calculation:

☞ **INITIAL_TIME**: The initial time `t_ini` must be introduced here.

☞ **FINAL_TIME**: The final time `t_ini` must be introduced here.

☞ **TIME_STEP_SIZE**: The time step size `dtime` is specified here. The calculation stops at time step n, where n is the integer such that n`dtime` $\geq$ `t_fin` is closest to `t_fin`.

ORDER_OF_BLOCK_ITERATIONS: `Fantom` may solve three problems, referred to as `momentum` (Navier-Stokes equations), `energy` (heat equation) and `front_tracking`, that are possibly coupled. The coupling is performed iteratively per blocks. The order of the problems within each iteration must be given here. Any combination is possible.

STAGGERED_PROBLEMS: Some of the problems to be solved may be treated staggered, that is to say, solved only once per time step. This may be done by allowing only one iteration in the numerical treatment of the problem (see below) or by explicitly saying this here.

GOVERNING_PROBLEM: Even though each problem has its convergence tolerance and number of iterations allowed within each time step, in some cases it is interesting to stop the iterative process within each time step when one of the problems (the one considered most relevant in the run)

NDIME	Integration points	Polynomial
1	1	Q1
1	2	Q3
1	3	Q5
1	4	Q7
2	$1 = 1 \times 1$	Q1
2	$4 = 2 \times 2$	Q3
2	$9 = 3 \times 3$	Q5
2	$16 = 4 \times 4$	Q7
3	$1 = 1 \times 1 \times 1$	Q1
3	$8 = 2 \times 2 \times 2$	Q3
3	$27 = 3 \times 3 \times 3$	Q5
3	$64 = 4 \times 4 \times 4$	Q7

Table 5: **Open** integration rules available in `Fantom` for **bricks**

has converged. Any of the problems solved by `Fantom` (`momentum`, `energy` and `front_tracking` may be defined as the governing one.

OTHER_PARAMETERS: This section is designed to include other relevant general numerical parameters. The only one currently existing is:

☞ ***Natural_length***: `Fantom` computes the characteristic length of the element from the length `hnatu` assigned to the reference element, which is the cube $[-1,1]^{n_{\rm sd}}$ ($n_{\rm sd}$ is the number of space dimensions) for brick elements and the standard simplex $0 \leq L_1, ..., L_{n_{\rm sd}+1} \leq 1$ (L_i are the area coordinates) for simplicial elements. Recommended values of `hnatu` are 2 for bricks and between 0.5 and 1 for simplices.

FREE_PARAMETERS: A list of up to 20 floats may be introduced here. This is designed to allow fast testing of different options (physical and/or numerical), without the need to change the routine that reads data.

7.2 Navier-Stokes equations

The Navier-Stokes equations are undoubtedly the most difficult problem solved by `Fantom.` Its numerical treatment is split into three sub-blocks of data:

```
$-----------------------------------------------------------------------------
  NAVIER_STOKES_EQUATIONS:
    SPATIAL_TREATMENT
```

`NDIME`	Integration points	Polynomial
1	2	Q1
1	3	Q3
1	4	Q3
2	$4 = 2 \times 2$	Q1
2	$9 = 3 \times 3$	Q3
2	$16 = 4 \times 4$	Q3
2	8	P2
3	$8 = 2 \times 2 \times 2$	Q1
3	$27 = 3 \times 3 \times 3$	Q3
3	$64 = 4 \times 4 \times 4$	Q3
3	20	P2

Table 6: **Closed** integration rules available in `Fantom` for **bricks**

`NDIME`	Integration points	Polynomial
2	1	P1
2	3	P2
2	4	P3
2	6	P4
2	7	P5
2	13	P9
3	1	P1
3	4	P2
3	5	P3
3	11	P4
3	14	P5

Table 7: **Open** integration rules available in `Fantom` for **simplices**

`NDIME`	Integration points	Polynomial
2	3	P1
2	4	P2
2	6	P1 (piecewise)
2	7	P3
2	10	P1 (piecewise)
3	4	P1
3	5	P2
3	10	P2
3	11	P3
3	15	P3
3	20	P2 and x^4, y^4, z^4

Table 8: **Closed** integration rules available in `Fantom` for **simplices**

```
    TEMPORAL_TREATMENT
    SOLUTION_PROCEDURE
  END_NAVIER_STOKES_EQUATIONS
$------------------------------------------------------------------------
```

These sub-blocks are described in the following. Before describing the possible options, let us present briefly the basic numerical formulation used by `Fantom` to solve equations (1)-(2).

7.2.1 Basic numerical formulation

Time discretization Let us consider now the temporal discretization of equations (1)-(2). Let $0 = t^0 < t^1 < ... < t^N = t_{\text{fin}}$ be a partition of the time interval and $\alpha \in [0, 1]$. We shall take the time step size $\Delta t := t^{n+1} - t^n$ constant for all n. Let us also introduce the notation

$$\delta f^n := f^{n+1} - f^n,$$
$$f^{n+\alpha} := \alpha f^{n+1} + (1-\alpha) f^n,$$
$$\delta_t f^n := \frac{\delta f^n}{\Delta t},$$

where f is a generic function of time and f^n denotes the value of f at time t^n or an approximation to it.

Let us assume that the force vector $\boldsymbol{f}$ is continuous in time and, for simplicity, that $\boldsymbol{\omega} = \mathbf{0}$, ν is constant, the flow is laminar, the Laplacian form is used for the viscous term, the non-conservative for the convective term, the boundary consists only of the Γ_D and Γ_N components, $\beta = 0$ and $\sigma = 0$. The time integration schemes for the Navier-Stokes equations considered by `Fantom` are the following:

- **One step approximation to the time derivative.**

  The time discretization is this case leads to the problem: from known $\boldsymbol{u}^n$, find $\boldsymbol{u}^{n+1}$ and p^{n+1} such that

$$\delta_t \boldsymbol{u}^n + C^n(\boldsymbol{u}) - \nu\Delta\boldsymbol{u}^{n+\alpha} + \nabla p^{n+1} = \boldsymbol{f}^{n+\alpha}, \tag{38}$$
$$\nabla\cdot\boldsymbol{u}^{n+1} = 0, \tag{39}$$

  where $C^n(\boldsymbol{u})$ is an approximation of the convective term.

  - *First order scheme.*

    Now $\alpha = 1$ in the previous equations. There are two possibilities:

    - Implicit convection:
      $$C^n(\boldsymbol{u}) = (\boldsymbol{u}^{n+1}\cdot\nabla)\boldsymbol{u}^{n+1}. \tag{40}$$
      The resulting scheme is unconditionally stable.
    - Explicit convection:
      $$C^n(\boldsymbol{u}) = (\boldsymbol{u}^{n}\cdot\nabla)\boldsymbol{u}^{n}. \tag{41}$$
      The resulting scheme is conditionally stable (Δt must be small enough).

  - *Second order scheme.*

    Now $\alpha = 1/2$. There are two possibilities:

    - Implicit convection:
      $$C^n(\boldsymbol{u}) = (\boldsymbol{u}^{n+1/2}\cdot\nabla)\boldsymbol{u}^{n+1/2}. \tag{42}$$
      The resulting scheme is unconditionally stable.
    - Explicit convection:
      $$C^n(\boldsymbol{u}) = \frac{1}{2}\left[3(\boldsymbol{u}^{n}\cdot\nabla)\boldsymbol{u}^{n} - (\boldsymbol{u}^{n-1}\cdot\nabla)\boldsymbol{u}^{n-1}\right]. \tag{43}$$
      The resulting scheme is conditionally stable (Δt must be small enough).

- **Two step approximation to the time derivative (Gear scheme)**.

  The scheme used in this case is:

$$\frac{1}{2\Delta t}(3\boldsymbol{u}^{n+1} - 4\boldsymbol{u}^{n+1} + \boldsymbol{u}^{n+1}) + (\boldsymbol{u}^{n+1}\cdot\nabla)\boldsymbol{u}^{n+1} - \nu\Delta\boldsymbol{u}^{n+1}$$
$$+\nabla p^{n+1} = \boldsymbol{f}^{n+1}, \tag{44}$$
$$\nabla\cdot\boldsymbol{u}^{n+1} = 0, \tag{45}$$

  It is a second order unconditionally stable scheme.

Weak form Once the time discretization has been carried out, we are left with a boundary value problem. This problem will be discretized in space using a finite element method. It is therefore necessary to obtain first the variational form of the boundary value problem. To describe the method we shall restrict ourselves to problem (38)-(39). The stationary case can be obtained setting (formally) $\Delta t = \infty$ and deleting the time superscripts. We consider here the same situation as before, but allowing a non-constant viscosity.

Consider the functional spaces

$$
\begin{aligned}
V^{n+1} &= \left\{ \boldsymbol{u}(\boldsymbol{x}) \in H^1(\Omega)^{n_{\rm sd}} \mid \ \boldsymbol{u} = \bar{\boldsymbol{u}}^{n+1} \text{ on } \Gamma_D \right\}, \\
V_0 &= \left\{ \boldsymbol{u}(\boldsymbol{x}) \in H^1(\Omega)^{n_{\rm sd}} \mid \ \boldsymbol{u} = \boldsymbol{0} \text{ on } \Gamma_D \right\}, \\
Q &= \left\{ q(\boldsymbol{x}) \in L^2(\Omega) \mid \int_\Omega q \, \mathrm{d}\Omega = 0 \text{ if } \Gamma_N = \emptyset \right\},
\end{aligned}
$$

where, as usual, $L^2(\Omega)$ denotes the space of square integrable functions in the domain Ω and $H^1(\Omega)$ the subspace of $L^2(\Omega)$ of functions with square integrable first derivatives.

Once these functional spaces have been introduced, the weak form of the problem defined by (38)-(39) reads as follows: find $\boldsymbol{u}^{n+1} \in V^{n+1}$ and $p^{n+1} \in Q$ such that

$$
\begin{aligned}
&\int_\Omega \boldsymbol{v} \cdot \left[\delta_t \boldsymbol{u}^n + (\boldsymbol{u}^{n+1} \cdot \nabla) \boldsymbol{u}^{n+1} \right] \mathrm{d}\Omega + \int_\Omega 2 \boldsymbol{\varepsilon}(\boldsymbol{v}) : \nu^{n+1} \boldsymbol{\varepsilon}(\boldsymbol{u}^{n+1}) \, \mathrm{d}\Omega - \int_\Omega p^{n+1} \nabla \cdot \boldsymbol{v} \, \mathrm{d}\Omega \\
&\qquad = \int_\Omega \boldsymbol{v} \cdot \boldsymbol{f}^{n+1} \, \mathrm{d}\Omega + \int_{\Gamma_N} \boldsymbol{v} \cdot \bar{\boldsymbol{t}}^{n+1} \, \mathrm{d}\Gamma, \qquad (46) \\
&\int_\Omega q \nabla \cdot \boldsymbol{u}^{n+1} \, \mathrm{d}\Omega = 0, \qquad (47)
\end{aligned}
$$

for all test functions $\boldsymbol{v} \in V_0$ and $q \in Q$.

Linearized equations and penalty method The final step previous to the finite element approximation of the nonlinear variational problem (46)-(47) is to linearize it. In our case, there are two sources of nonlinearity, namely, the convective term of the Navier-Stokes and the heat equations and the fact that the viscosity depends in a nonlinear way on the viscosity through a non Newtonian constitutive model.

Concerning the linearization of the convective term of the Navier-Stokes equations, both the fixed point Picard scheme and the Newton-Raphson method can be considered. Let us denote by the superscript i the iteration counter. Given an approximation $\boldsymbol{u}^{n+1,i}$ to $\boldsymbol{u}^{n+1}$, the next one is computed by approximating

$$
\begin{aligned}
&(\boldsymbol{u}^{n+1,i+1} \cdot \nabla) \boldsymbol{u}^{n+1,i+1} \approx (1 - \lambda_0)(\boldsymbol{u}^{n+1,i} \cdot \nabla) \boldsymbol{u}^{n+1,i} \\
&\lambda_0 \left\{ (\boldsymbol{u}^{n+1,i} \cdot \nabla) \boldsymbol{u}^{n+1,i+1} + \lambda_1 \left[(\boldsymbol{u}^{n+1,i+1} \cdot \nabla) \boldsymbol{u}^{n+1,i} - (\boldsymbol{u}^{n+1,i} \cdot \nabla) \boldsymbol{u}^{n+1,i} \right] \right\}, \qquad (48)
\end{aligned}
$$

where, for $\lambda_0 = 1$, $\lambda_1 = 0$ corresponds to the Picard method and $\lambda_1 = 1$ to the Newton-Raphson scheme. The former is first order, whereas the latter is second order but needs an initial guess close enough to the converged solution to converge. The case

$\lambda_0 = 0$ leads to a simple linearization that only converges for very small convective terms (small Reynolds numbers).

The linearization of the constitutive model is dealt with a Picard-like strategy, taking the viscosity evaluated with the velocity $\boldsymbol{u}^{n+1,i}$ when $\boldsymbol{u}^{n+1,i+1}$ is to be computed.

The fully linearized form of problem (46)-(47) taking $\lambda_0 = 1$ and $\lambda_1 = 0$, and using a penalty strategy, is as follows: given a guess $\boldsymbol{u}^{n+1,i}$ for $\boldsymbol{u}^{n+1}$, find $\boldsymbol{u}^{n+1,i+1} \in V^{n+1}$ and $p^{n+1,i+1} \in Q$ such that

$$\begin{aligned}
&\int_\Omega \boldsymbol{v} \cdot \left\{ \delta_t \boldsymbol{u}^{n,i+1} + (\boldsymbol{u}^{n+1,i} \cdot \nabla) \boldsymbol{u}^{n+1,i+1} \right\} \mathrm{d}\Omega \\
&\qquad + \int_\Omega 2\boldsymbol{\varepsilon}(\boldsymbol{v}) : \nu^{n+1,i} \boldsymbol{\varepsilon}(\boldsymbol{u}^{n+1,i+1}) \,\mathrm{d}\Omega - \int_\Omega p^{n+1,i+1} \nabla \cdot \boldsymbol{v} \,\mathrm{d}\Omega \\
&\qquad = \int_\Omega \boldsymbol{v} \cdot \boldsymbol{f}^{n+1} \,\mathrm{d}\Omega + \int_{\Gamma_{\mathrm{nv}}} \boldsymbol{v} \cdot \bar{\boldsymbol{t}}^{n+1} \,\mathrm{d}\Gamma,
\end{aligned} \tag{49}$$

$$\epsilon \int_\Omega q p^{n+1,i+1} \,\mathrm{d}\Omega + \int_\Omega q \nabla \cdot \boldsymbol{u}^{n+1,i+1} \,\mathrm{d}\Omega = \epsilon \int_\Omega q p^* \,\mathrm{d}\Omega, \tag{50}$$

for all test functions $\boldsymbol{v} \in V_0$ and $q \in Q$ and $\psi \in \Psi_0$. Here, $\nu^{n+1,i}$ means that the viscosity is evaluated with $\boldsymbol{u}^{n+1,i}$. Likewise,

$$\frac{\delta \boldsymbol{u}^{n,i+1}}{\Delta t} = \frac{1}{\Delta t} \left(\boldsymbol{u}^{n+1,i+1} - \boldsymbol{u}^n \right).$$

In (50), $\epsilon \geq 0$ is a (small) penalty parameter and p^* is chosen according to the following possibilities:

- *Classical penalty method*: $p^* = 0$.
- *Iterative penalty method*: $p^* = p^{n+1,i}$. The effect of the penalization disappears as the iterative process goes on.
- *Artificial compressibility*: $p^* = p^n$. Artificial compressibility method.

Finite element approximation Let $\{\Omega^e\}$ be a finite element partition of the domain Ω, with index e ranging from 1 to the number of elements n_{el}. We denote with a subscript h the finite element approximation to the unknown functions, and by $\boldsymbol{v}_h$ and q_h the velocity and pressure test functions associated to $\{\Omega^e\}$.

The finite element approximation to the functional spaces where the unknowns and the test functions belong are also characterized by a subscript h. The finite element spaces for the velocity and the pressure may be constructed using *the same* interpolation or a div-stable one, satisfying the classical inf-sup or Babuška-Brezzi condition. In the first case, a stabilized formulation needs to be used. The method used by `Fantom` is the algebraic version of the sub-grid scale concept, which allows to use equal velocity-pressure interpolation, to stabilize convective effects and to stabilize rotation.

In the most general case, the bottom line of the method is to test the residual of the momentum equation and the incompressibility equation within each element by the functions $\boldsymbol{\zeta}_u$ and ζ_p, respectively, given by:

$$\boldsymbol{\zeta}_u = \tau_u \left\{ (\boldsymbol{u}_h \cdot \nabla)\boldsymbol{v}_h + 2\ \boldsymbol{\omega} \times \boldsymbol{v}_h - \sigma \boldsymbol{v}_h + 2\nabla \cdot [\nu \boldsymbol{\varepsilon}(\boldsymbol{v}_h)] + \nabla q_h \right\}, \tag{51}$$

$$\zeta_p = \tau_p \nabla \cdot \boldsymbol{v}_h, \tag{52}$$

where the parameters τ_u and τ_p are

$$\tau_u = \left[\frac{c_1 \nu}{h^2} + \frac{c_2 |\boldsymbol{u}_h|}{h} + c_3 |\boldsymbol{\omega}| + c_4 \sigma \right]^{-1}, \tag{53}$$

$$\tau_p = c_5 \left[c_1 \nu + c_2 |\boldsymbol{u}_h| h + c_3 |\boldsymbol{\omega}| h^2 + c_4 \sigma h^2 \right], \tag{54}$$

where c_i, $i = 1, 2, 3, 4, 5$, are numerical constants, h is the element size for linear elements and half of it for quadratics and the velocity of these expressions is the last one known (previous iteration of the current time step).

Divergence of the residual stabilization (DRS) This method is intended to stabilize rotating flows, although it can be applied in general situations. It consists in adding to the original Galerkin terms a least squares form of the divergence of the residual of the momentum equation within each element.

Let $\mathcal{A}(\boldsymbol{u}_h, p_h; \boldsymbol{v}_h, q_h) - \mathcal{L}(\boldsymbol{v}_h, q_h) = 0$ be the discrete form of the problem to be solved *without* stabilization. The problem stabilized with the DRS method consists of finding a velocity $\boldsymbol{u}_h$ and a pressure p_h such that

$$\mathcal{A}(\boldsymbol{u}_h, p_h; \boldsymbol{v}_h, q_h) - \mathcal{L}(\boldsymbol{v}_h, q_h) + \mathcal{D}(\boldsymbol{u}_h, p_h; \boldsymbol{v}_h, q_h) = 0$$

for all velocity test functions $\boldsymbol{v}_h$ and pressure test functions q_h.

The stabilization term $\mathcal{D}(\boldsymbol{u}_h, p_h; \boldsymbol{v}_h, q_h)$ is a least squares form of the divergence of the momentum equations, given by

$$\mathcal{D}(\boldsymbol{u}_h, p_h; \boldsymbol{v}_h, q_h) = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \left[\tau_{\mathrm{drs}} \nabla \cdot \boldsymbol{S}_1^{\mathrm{v}}(\boldsymbol{v}_h, q_h) \left(\nabla \cdot \boldsymbol{S}_1^{\mathrm{v}}(\boldsymbol{u}_h, p_h) - \nabla \cdot \boldsymbol{f} \right) \right] \mathrm{d}\Omega$$

where $\boldsymbol{S}_1^{\mathrm{v}}(\boldsymbol{v}_h, q_h)$ is the first component of the Navier-Stokes operator with a given advection velocity and without the viscous term applied to the test functions, whereas $\boldsymbol{S}_1^{\mathrm{v}}(\boldsymbol{u}_h, p_h)$ is its counterpart applied to the velocity and pressure unknowns (observe that the divergence of the viscous term is zero for solenoidal velocities). The parameter τ_{drs} is taken for each element as

$$\tau_{\mathrm{drs}} = \delta(\mathrm{Ek}_h) \frac{h^2}{|\boldsymbol{\omega}|}, \qquad \mathrm{Ek}_h := \frac{\nu}{\omega L^2}$$

where δ is a function of the local Ekman number Ek_h whose expression is dictated by the stability analysis of the method by

$$\begin{aligned} \delta(\mathrm{Ek}_h) &= \min\left\{ \frac{C_1}{\mathrm{Ek}_h}, C_2 \right\} \\ &= \min\left\{ C_1 \frac{|\boldsymbol{\omega}| h^2}{\nu}, C_2 \right\} \end{aligned} \tag{55}$$

with C_1 and C_2 algorithmic constants.

The resulting discrete variational equation is consistent, in the sense that the exact solution of the problem satisfies it (for sufficiently smooth solutions).

It is important to remark at this point that the added term $\mathcal{D}(\boldsymbol{u}_h, p_h; \boldsymbol{v}_h, q_h)$ is not introduced to stabilize the velocity-pressure interpolation. As a consequence, we still need to use div-stable finite elements or stabilized formulations.

Boundary residual When the momentum equation is weighted element by element by the perturbation (51) there is a contribution from the gradient of the pressure test function. This contribution is

$$S_p \equiv \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_u \nabla q_h \cdot [\boldsymbol{S}_1(\boldsymbol{u}_h, p_h) - \boldsymbol{f}] \, \mathrm{d}\Omega$$

where $\boldsymbol{S}_1(\boldsymbol{u}_h, p_h)$ is the first component of the Navier-Stokes operator with a given advection velocity. When τ_u is constant, this term can be thought of as part of the weak form of (minus) the divergence of the residual tested by q_h, which would be

$$\begin{aligned} &-\sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_u q_h \nabla \cdot [\boldsymbol{S}_1(\boldsymbol{u}_h, p_h) - \boldsymbol{f}] \, \mathrm{d}\Omega \\ &\qquad = S_p - \sum_{e=1}^{n_{\mathrm{el}}} \int_{\partial\Omega^e \cap \partial\Omega} q_h \boldsymbol{n} \cdot [\boldsymbol{S}_1(\boldsymbol{u}_h, p_h) - \boldsymbol{f}] \, \mathrm{d}\Gamma \end{aligned} \tag{56}$$

In some cases it might be of interested to see what happens if the boundary term in this equation is included as stabilizing term. It is known that this prevents the appearence of spurious pressure gradients near boundaries, similar to those known to appear in classical projection schemes.

Shock capturing In some cases, the solution of the Navier-Stokes equations may exhibit sharp gradients, which may lead to localized oscillations. In order to remove them, a shock capturing (or discontinuity capturing) technique may be used.

Although this technique can be applied to any of the problems solved by `Fantom` , including the Navier-Stokes equations, it is most easily explained for the convection-diffusion-reaction equation for an unknown ϕ. For simplicity, Dirichlet boundary conditions $\phi = g$ will be considered on the whole boundary $\partial\Omega$.

The boundary value problem that we consider is

$$\mathcal{R}(\phi_h) := -\kappa \Delta\phi + \boldsymbol{u} \cdot \nabla\phi + \alpha\phi - f = 0 \quad \text{in} \ \ \Omega, \tag{57}$$

$$\phi = g \quad \text{on} \ \ \partial\Omega, \tag{58}$$

where where $\kappa > 0$ and $\alpha \geq 0$ are constants.

The basic finite element formulation for solving (57)-(58) used by `Fantom` is a stabilization method similar to that employed for the Navier-Stokes equations and described before. However, it does not preclude the presence of overshoots and undershoots in the vicinity of sharp gradients of the solution of problems with a very small diffusion term. To avoid them, the idea is to introduce a nonlinear numerical scheme in which the diffusion added is given by

$$\kappa_{\mathrm{dc}} = \frac{1}{2}\xi_{\mathrm{c}} h \frac{|\mathcal{R}(\phi_h)|}{|\nabla \phi_h|} \quad \text{for each element,}$$

where the parameter ξ_{c} is computed as

$$\xi_{\mathrm{c}} = \max\left\{0, C_{\mathrm{dc}} - \frac{2\kappa}{|\boldsymbol{u}^*|h}\right\}, \tag{59}$$

with

$$\boldsymbol{u}^* := \frac{1}{|\nabla \phi_h|^2}\left(\boldsymbol{u}\cdot\nabla\phi_h + \sigma\phi_h - f\right)\nabla\phi_h,$$

and C_{dc} a constant. From numerical experiments we have found that an effective choice is to take $C_{\mathrm{dc}} = 0.7$ for linear elements and $C_{\mathrm{dc}} = 0.35$ for quadratics.

The problem now is to decide in which directions is it necessary to introduce the diffusion κ_{dc}. It can be readily seen that the numerical diffusion inherent to the stabilized formulation acting only along the streamlines is given by

$$\kappa_{\mathrm{stab}} = \frac{1}{2}\xi h|\boldsymbol{u}|,$$

which may very well be greater than κ_{dc}. Therefore, the numerical dissipation that has to be introduced along the streamlines is only

$$\kappa_{\mathrm{sl}} = \max\{0, \kappa_{\mathrm{dc}} - \kappa_{\mathrm{stab}}\}$$

After this heuristic reasoning we are led to the introduction of a nonlinear anisotropic diffusion to the basic numerical formulation. There is also the possibility of adding the nonlinear diffusion along all directions. In any case, the final numerical method will consist in the addition of the term

$$\sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e}\left[\kappa_{\mathrm{dc}}\nabla\psi_h\cdot\nabla\phi_h + \beta_{\mathrm{dc}}(\kappa_{\mathrm{sl}} - \kappa_{\mathrm{dc}})\nabla\psi_h\cdot\left(\frac{1}{|\boldsymbol{u}|^2}\boldsymbol{u}\otimes\boldsymbol{u}\right)\cdot\nabla\phi_h\right]\mathrm{d}\Omega \tag{60}$$

where $\beta_{\mathrm{dc}} = 0$ if the shock capturing viscosity is isotropic and $\beta_{\mathrm{dc}} = 1$ if it is anisotropic.

7.2.2 Spatial treatment

The numerical formulation employed for the space approximation of the Navier-Stokes equations is described in this firs sub-block, whose template is:

```
SPATIAL_TREATMENT:
  PRESSURE_INTERPOLATION: NUM_NODES: nnods,
                          ``standard''   & ``hierarchic'',
                          ``continuous'' & ``discontinuous''
  PENALTY_STRATEGY:       ``off'' & ``classical'' & ``iterative''
                          `` artifitial_compressibility'',
                          PARAMETER: penal
  BUBBLE_ELIMINATION:     ``off'' & ``on''
  STABILITY_FUNCTIONS:    MAX_CONVECTIVE: sfnsc,
                          MAX_PRESSURE:   sfnsp,
                          MAX_ROTATION:   sfnsr,
                          MAX_POROSITY:   sfnss,
                          MAX_DIVERGENCE: sfnsd
  PERTURBATION_FUNCTION:  ``off'' & ``all'' & [convect], [viscous],
                          [pressure], [coriolis/porosity]
  DIVERGENCE_RESIDUAL:    ``off'' & ``on'', PARAMETER: sfdmo
  BOUNDARY_RESIDUAL:      ``on''  & ``off''
  SHOCK_CAPTURING:        ``off'' & ``isotropic'' & ``anisotropic'',
                          PARAMETER: shcns
END_SPATIAL_TREATMENT
```

The commands and parameters to be provided are:

PRESSURE_INTERPOLATION: This defines the interpolation used for the pressure. An integer and two commands must be provided:

- ☞ **NUM_NODES**: Number of nodes to interpolate the pressure within each element.
- ☞ ***Standard or hierarchic***: When the standard interpolation is used, the pressure degrees of freedom correspond to pressure values at the pressure nodes, whereas when the hierarchic interpolation is employed the degrees of freedom have different meaning. For discontinuous pressures, they are the pressure value at the center of the element and its derivatives, whereas for continuous pressures they correspond to the standard hierarchich enrichment.
- ☞ ***Continuous or discontinuous***: Both options are allowed. The possible combinations when div-stable velocity pressure interpolation need to be used are indicated in Table 9.

PENALTY_STRATEGY: The particular penalty strategy must be indicated here, with a command and a float. The options for the command and the meaning of the number are:

- ☞ ***Off***: No penalty strategy is used, that is, $\epsilon = 0$ in (50).

- ☞ ***Classical***: $p^* = 0$ in (50). This corresponds to the classical penalty method, with an error of order ϵ.
- ☞ ***Iterative***: $p^* = p^{n+1,i}$ in (50). The effect of the penalization disappears as the iterative procedure goes on. This is similar to the Augmented Lagrangian treatment of saddle point problems.
- ☞ ***Artifitial_compressibility***: $p^* = p^n$ in (50). The reason for this name comes from the fact that (50) can be understood as a backward Euler discretization of the equation $c^{-2}\partial_t p + \nabla \cdot \boldsymbol{u} = 0$, with an adequate identification of the parameter c^{-2}. Observe that this option only makes sense for transient problems.
- ☞ **PARAMETER**: Value of the parameter ϵ in (50). It is convenient to take it of the order of $10^{-6}\mu^{-1}$ to $10^{-9}\mu^{-1}$, where μ is a characteristic value of the viscosity used in the calculation (it must be the kinematic viscosity if the density is set to 1). If the iterative penalty method is used, values of ϵ larger than $10^{-6}\mu^{-1}$ may be also used, provided the incompressibility restriction is approximated sufficiently well.

BUBBLE_ELIMINATION: If the element has interior nodes (bubbles), not shared by other elements, it may be interesting to condense them at the element in order to reduce the size of the final algebraic system to be solved. To do it or not is decided here:

- ☞ ***Off***: No bubble elimination is done.
- ☞ ***On***: If possible, variables of interior nodes are condensed.

STABILITY_FUNCTIONS: The constants appearing in the stabilization parameters τ_u and τ_p are given here. The RECOMMENDED values of all these constants are 1. For quadratic elements, `Fantom` takes h as half the characteristic element length.

- ☞ **MAX_CONVECTIVE**: Constant c_2 in (53) divided by 2.
- ☞ **MAX_PRESSURE**: Constant c_1 in (53) divided by 4. It is referred to as the constant of the pressure term because it must be > 0 in order to stabilize the pressure, even if the rest of terms in τ_u are not present. Since it must be positive whenever a stabilization term exists, `Fantom` takes it at least equal to $\max\{c_2, c_3, c_4\}$, where c_3 and c_4 are defined next.
- ☞ **MAX_ROTATION**: Constant c_3 in (53).
- ☞ **MAX_POROSITY**: Constant c_4 in (53).
- ☞ **MAX_DIVERGENCE**: Constant c_5 in (54).

PERTURBATION_FUNCTION: This defines the perturbation functions in (51) and (52). For the sake of generality, the code allows to introduce the different terms in these functions independently. The following commands define them:

- ☞ ***Off***: $\boldsymbol{\zeta}_u = \mathbf{0}$, $\zeta_p = 0$.
- ☞ ***All***: All the terms in (51) and (52) are included. Although this is the natural choice, it is possible to explicitly indicate which terms must appear in these expressions:
- ☞ ***Convective***: The convective term appears in $\boldsymbol{\zeta}_u$ and ζ_p.
- ☞ ***Pressure***: The pressure AND THE VISCOUS terms appear in $\boldsymbol{\zeta}_u$ and ζ_p.
- ☞ ***Coriolis/porosity***: If any of these two commands is specified, BOTH the Coriolis and the porosity terms are incorporated in $\boldsymbol{\zeta}_u$ and ζ_p.

DIVERGENCE_RESIDUAL: This command controls the use of the divergence of the residual stabilization (DRS) method. It has two obvious options and a parameter:

- ☞ ***Off***: The DRS method is not used.
- ☞ ***On***: The DRS method is activated.
- ☞ **PARAMETER**: Constant C_2 in (55). The constant C_1 is taken as 0.

BOUNDARY_RESIDUAL: This command controls whether the second term in the RHS of (56) is included or not. Its only two choices are:

- ☞ ***Off***: The boundary term involving the residual IS NOT included.
- ☞ ***On***: The boundary term IS included.

SHOCK_CAPTURING: This block controls the type of shock capturing strategy used for the Navier-Stokes equations. There are three possible options and a parameter that have to be provided:

- ☞ ***Off***: No shock capturing technique is used.
- ☞ ***Isotropic***: $\beta_{\mathrm{dc}} = 0$ in (60).
- ☞ ***Anisotropic***: $\beta_{\mathrm{dc}} = 1$ in (60).
- ☞ **PARAMETER**: Value of the parameter C_{dc} in (59). Recommended values are $C_{\mathrm{dc}} = 0.7$ for linear elements and $C_{\mathrm{dc}} = 0.35$ for quadratics.

NDIME/Element shape	NNODE	NNODS	**Pressure interpolation**
2/Quad	4	1	Discontinuous (Q_1/P_0)
2/Quad	8	1	Discontinuous
2/Quad	8	4	Continuous
2/Quad	9	3	Discontinuous (Q_2/P_1)
2/Quad	9	4	Continuous (Taylor-Hood)
2/Triangle	4	3	Continuous (Mini-element)
2/Triangle	6	1	Discontinuous (P_2/P_0)
2/Triangle	6	3	Continuous (Taylor-Hood)
2/Triangle	7	3	Discontinuous (P_2^+/P_0)
3/Brick	8	1	Discontinuous (Q_1/P_0)
3/Brick	20	1	Discontinuous
3/Brick	20	8	Continuous
3/Brick	27	4	Discontinuous (Q_2/P_1)
3/Brick	27	8	Continuous (Taylor-Hood)
3/Tetrahedron	5	4	Continuous (Mini-element)
3/Tetrahedron	10	1	Discontinuous (P_2/P_0)
3/Tetrahedron	10	4	Continuous (Taylor-Hood)
3/Tetrahedron	15	4	Discontinuous (P_2^+/P_0)

Table 9: Examples of div-stable velocity-pressure interpolations allowed by `Fantom`.

7.2.3 Temporal treatment

This block contains the data that define the time integration algorithm for the Navier-Stokes equations. Obviously, it is only necessary when the transient problem is being analyzed. The template of this block is the following:

```
TEMPORAL_TREATMENT:
  WEIGHTING_OF_DERIVATIVE:  ``all''      & ``Galerkin''
  INCREMENTAL_QUOTIENT:     ``one_step'' & ``two_steps''
  If one_step:
    ORDER_OF_APPROXIMATION: order
    CONVECTION_TREATMENT:   ``implicit'' & ``explicit'',
                            SAFETY_FACTOR: safns
  EULER_TIME_STEPS:         neule
  STEADY_STATE_TOLERANCE:   tosns
END_TEMPORAL_TREATMENT
```

The commands and parameters to be provided are the following:

WEIGHTING_OF_DERIVATIVE: If the time discretization of the continuous equations is performed first, it seems natural that the function $\boldsymbol{\zeta}_u$ in (51) weights all the element residual of the equations, including the discretization of the time derivative $\delta_t \boldsymbol{u}_h^n$ (or a two-step approximation to it). However, using different reasoning (such as the characteristic Galerkin method or a discontinuous finite element discretization in time as well) one may arrive to similar schemes in which $\delta_t \boldsymbol{u}_h^n$ IS NOT weighted by $\boldsymbol{\zeta}_u$. `Fantom` has the possibility of testing both options:

- ☞ ***All***: The function $\boldsymbol{\zeta}_u$ weights also $\delta_t \boldsymbol{u}_h^n$.
- ☞ ***Galerkin***: The function $\boldsymbol{\zeta}_u$ DOES NOT weight $\delta_t \boldsymbol{u}_h^n$. This option is usually more diffusive, but might be useful if only the steady state is of interest.

INCREMENTAL_QUOTIENT: This command defines the approximation employed for the temporal derivative, $\delta_t \boldsymbol{u}_h^n$. The options are:

- ☞ ***One_step***: In this case $\delta_t \boldsymbol{u}_h^n$ is used to approximate the time derivative. The equations to be approximated are (38)-(39).
- ☞ ***Two_steps***: This corresponds to the SECOND ORDER Gear scheme (44)-(45).

ORDER_OF_APPROXIMATION: This command needs to be provided ONLY IF the option ***One_step*** has been used as approximation to the time derivative. In this case, the scheme can be of first and one second order. This INTEGER (`order` = 1 or 2) must be given here.

CONVECTION_TREATMENT: This command needs to be provided ONLY IF the option ***One_step*** has been used as approximation to the time derivative. It defines how the convective term is treated:

- ☞ ***Implicit***: The convective term is treated implicitly. If `order` = `1`, (40) is used, whereas if `order` = `2` (42) is employed.
- ☞ ***Explicit***: The convective term is treated explicitly. If `order` = `1`, (41) is used, whereas if `order` = `2` (43) is employed.
- ☞ **SAFETY_FACTOR**: Only when THE EXPLICIT option is used for the convective term, a safety factor `safns` must be supplied. The time step will be computed by the code as $\Delta t = \texttt{safns}\,\Delta t_{\text{crit}}$, where Δt_{crit} is the maximum time step allowed for stability reasons.

EULER_TIME_STEPS: Even when a second order scheme is used for the time integration, it can be convenient to perform a few time steps using the simple Euler scheme (with an implicit or explicit treatment of the convective term). This can help to damp out the high frequencies of the initial condition. The number of time steps to be advanced with the Euler scheme must be provided here.

STEADY_STATE_TOLERANCE: This parameter (`tosns`) is the tolerance to decide whether the steady state has been reached or not. It must be pointed out that the criterion employed is that $\|\boldsymbol{U}^{n+1} - \boldsymbol{U}^{n}\| \leq \texttt{tosns}\|\boldsymbol{U}^{n+1}\|$, where $\boldsymbol{U}^{n+1}$ is the vector of velocity nodal values and $\|\cdot\|$ the same norm as for checking convergence (see below). Therefore, `tosns` must be chosen by the user according to the time step employed and the characteristic time length of the problem.

7.2.4 Solution procedure

This block of data defines the linearization of the equations and the method used to deal with the final linear system to be solved. The template is:

```
SOLUTION_PROCEDURE:
  LINEARIZATION_TECHNIQUE: ``rhs'' & ``Picard'' & ``Newton_Raphson'',
                           PIC_ITERATIONS: npica,
                           REL_PARAMETER:  repns
  CONVERGENCE_STRATEGY:    ``L2'' & ``L1'' & ``L-inf'',
                           MAX_ITERATIONS: micns, TOLERANCE: tocns
  ******************
  Linear solver data
  ******************
END_SOLUTION_PROCEDURE
```

The commands and parameters to be provided are the following:

LINEARIZATION_TECHNIQUE: This command defines the linearization strategy employed for THE CONVECTIVE term of the Navier-Stokes equations. Other nonlinearities, such as those arising because of the stabilization or those due to a non Newtonian behavior, are dealt with using a simple fix point method. The options and the parameters to be provided are:

- ☞ ***Rhs***: $\lambda_0 = 0$ in (48).
- ☞ ***Picard***: $\lambda_0 = 1$ and $\lambda_1 = 0$ in (48).
- ☞ ***Newton_Raphson***: $\lambda_0 = 1$ and $\lambda_1 = 1$ in (48).
- ☞ **PIC_ITERATIONS**: The Newton-Raphson method is faster than the other two, but requires the initial guess to be close enough to the final solution to converge. In order to obtain a good initial guess, several Picard iterations can be performed before resorting to the Newton-Raphson algorithm. The number of these iterations (within each time step) must be provided here.
- ☞ **REL_PARAMETER**: In some cases, especially when convergence is hard to achieve, it can be useful to use a relaxation method, redefining $\boldsymbol{u}^{n+1,i+1} \leftarrow \gamma \boldsymbol{u}^{n+1,i+1} + (1-\gamma)\gamma \boldsymbol{u}^{n+1,i}$ once $\boldsymbol{u}^{n+1,i+1}$ has been computed. The parameter $\gamma \in (0, 1]$ must be given here.

CONVERGENCE_STRATEGY: This command defines the way to check convergence. First, the norm in which it is evaluated must be given, and then two parameters must be provided. The norms are those of ℓ^p for $p = 1$, 2 and ∞, defined for a vector $\boldsymbol{X}$ of n components as

$$\|\boldsymbol{X}\|_{\ell^p}^p = \sum_{i=1}^{n} |X_i|^p \quad \text{for } p = 1, 2, \qquad \|\boldsymbol{X}\|_{\ell^\infty} = \max_{i=1,\dots,n} \{|X_i|\}$$

- ☞ ***L2***: Convergence is checked in the discrete ℓ^2 norm for the velocity components.
- ☞ ***L1***: Convergence is checked in the discrete ℓ^1 norm for the velocity components.
- ☞ ***L-inf***: Convergence is checked in the discrete ℓ^∞ norm for the velocity components.
- ☞ **MAX_ITERATIONS**: Maximum number of iterations permitted per time step. For transient problems, the program keeps advancing in time even if convergence has not been achieved within a certain time step.

☞ **TOLERANCE**: Convergence tolerance, checked in the norm specified before.

After having defined the LINEARIZATION_TECHNIQUE and which is the CONVERGENCE_STRATEGY, the data referring to the linear solved to be used must be provided. This is the same for all the problems, and described in the last subsection of this section.

7.3 Turbulence parameters

This block contains the data for the numerical treatment of the turbulence parameters. One of them is the turbulent kinetic energy k and the second one, if it exists, depends on the two equation model employed.

The basic numerical model used to approximate the equations for the turbulence parameters is similar to that used for the Navier-Stokes equations (it is a stabilized finite element method using the backward Euler scheme for the time integration and the possibility of using a shock capturing method). It will not be described here in detail.

The template for this block of data is:

```
$----------------------------------------------------------------------------
  TURBULENCE_PARAMETERS_EQS:
    STABILITY_FUNCTION:       MAX_CONVECTIVE: sftuc
    TEMPORAL_TERM_WEIGHTING:  ``all'' & ``Galerkin''
    SHOCK_CAPTURING:          ``off'' & ``isotropic'' & ``anisotropic'',
                              PARAMETER: shctu
    K_ITERATIONS:             nkwit
    COUPLING_OF_PARAMETERS:   ``per_blocks'' & ``full''
    WALL_TREATMENT:           ``wall_integration'' &
                              ``velocity_stress''  & ``stress_velocity''
    LINEARIZATION_TECHNIQUE:  ``rhs'' & ``Picard'' & ``Newton_Raphson'',
                              PIC_ITERATIONS: npitu,
                              REL_PARAMETER:  reptu
    CONVERGENCE_STRATEGY:     OUTER_ITERATIONS: micou,
                              INNER_ITERATIONS: micin,
                              TOLERANCE:        toctu,
                              OUTPUT_STEPS:     nsctu
    ******************
    Linear solver data
    ******************
  END_TURBULENCE_PARAMETERS_EQS
$----------------------------------------------------------------------------
```

The meaning of these commands is the following:

STABILITY_FUNCTIONS: To deal with convective dominated flows, a stabilized finite element method must be used. For the turbulence parameters, the simple SUPG method is employed, consisting in using as perturbation of the test function its streamline derivative multiplied by an algorithmic parameter τ_{tur}, which is computed within each element in the classical way

$$\tau_{\text{tur}} = \frac{\xi h}{2U}, \quad \xi(\text{Pe}) = \xi_0 \min\left\{\frac{\text{Pe}}{3}, 1\right\}, \quad \text{Pe} := \frac{Uh}{2\nu_t} \tag{61}$$

where U is the norm of the mean element velocity and ν_t is the turbulent kinematic viscosity. The only parameter that the user has to specify is

- ☞ **MAX_CONVECTIVE**: Parameter ξ_0 in (61). Recommended values are 1 for linear elements and 0.5 for quadratics.

TEMPORAL_TERM_WEIGHTING: The time integration of the turbulence parameters IS ALWAYS performed using the BACKWARD EULER scheme. This can not be decided by user, but what can be chosen is whether the element residual in the stabilizing term includes or not the approximation to the temporal derivative (in a similar way as for the Navier-Stokes equations). This is specified in this command:

- ☞ ***All***: The element residual in the stabilizing term INCLUDES the approximation to the temporal derivative.
- ☞ ***Galerkin***: The element residual in the stabilizing term DOES NOT INCLUDE the approximation to the temporal derivative.

SHOCK_CAPTURING: This block controls the type of shock capturing strategy used for the turbulence parameters, in the same way as for the Navier-Stokes equations. There are three possible options and a parameter that have to be provided:

- ☞ ***Off***: No shock capturing technique is used.
- ☞ ***Isotropic***: $\beta_{\text{dc}} = 0$ in (60).
- ☞ ***Anisotropic***: $\beta_{\text{dc}} = 1$ in (60).
- ☞ **PARAMETER**: Value of the parameter C_{dc} in (59). Recommended values are $C_{\text{dc}} = 0.7$ for linear elements and $C_{\text{dc}} = 0.35$ for quadratics.

K_ITERATIONS: One of the hardest problems in the numerical approximation of turbulence models is convergence of the nonlinear scheme. Even when two equation models are employed, it may be convenient to perform some iterations of the k model, which normally converges better, and then moving to the two equation model. This number of iterations must be given here.

COUPLING_OF_PARAMETERS: When two equation models are used, `Fantom` allows to couple the two turbulence parameters in two manners:

- ☞ ***Per_blocks***: A block iterative scheme is used in the case. First, equation (11) for k is approximated and then, with the value obtained for k, the equation for the second turbulence parameter is solved.
- ☞ ***Full***: The equations for k and the second turbulence parameter are linearized and solved at the same time. In this case, the linearization employed is the NEWTON-RAPHSON method, regardless of what is specified below as linearization technique.

WALL_TREATMENT: From the physical point of view, there are two possibilities to prescribe the turbulence parameters on the walls, already described in the section on fixity conditions: the equations can be integrated up to the wall or a wall law can be used. The second option has two possible numerical treatments. Altogether, the options for the prescription of the turbulence parameters on the walls are:

- ☞ ***Wall_integration***: The equations are integrated up to the wall.
- ☞ ***Velocity_stress***: The turbulent stress is prescribed. The iterative scheme is: guess $\boldsymbol{u}$ (velocity), compute U_* (see (36)), compute and fix the stress.
- ☞ ***Stress_velocity***: The velocity is precribed. In this case, the iterative scheme is: guess the stress, compute U_*, compute and fix the velocity.

LINEARIZATION_TECHNIQUE: This command defines the linearization strategy for the turbulence parameters. The advection velocity and the turbulent kinetic energy are assumed to be given, and therefore this linearization only affects the reactive (zero order) terms. The options are:

- ☞ ***Rhs***: All the nonlinear terms are evaluated at the previous iteration.
- ☞ ***Picard***: Fixed point linearization of the nonlinear terms.
- ☞ ***Newton_Raphson***: Newton-Raphson linearization of the nonlinear terms. It is the recommended option.
- ☞ **PIC_ITERATIONS**: Number of Picard iterations before moving to the Newton-Raphson linearization.

☞ **REL_PARAMETER**: Relaxation parameter (see below).

CONVERGENCE_STRATEGY: This command defines the convergence technique employed. Within each iteration to solve the momentum equation, the turbulence parameters are computed according to the following algorithm (ϕ stands for the second turbulence parameter):

1 Compute P_k and P_ϕ (production terms)

2 Repeat until convergence for L (mixing length):

2.1 Solve for k:

2.1.1 Approximate the equation for k at iteration i

2.1.2 $k_i \leftarrow \gamma k_i + (1-\gamma) k_{i-1}$

2.2 Check convergence for k. If not, increase i and go to 2.1

2.3 Update ν_t (turbulent kinematic viscosity)

2.4 Solve for ϕ:

2.4.1 Approximate the equation for ϕ at iteration i

2.4.2 $\phi_i \leftarrow \gamma \phi_i + (1-\gamma)\phi_{i-1}$

2.5 Check convergence for ϕ. If not, increase i and go to 2.4

2.6 Compute L

2.7 Use under-relaxation for L

2.8 Update ν_t

3 Check convergence for L. If not, increase the iteration counter and go to 2.

Here, γ is the under-relaxation parameter introduced above. The value $\gamma = 0.5$ is a good choice. Observe that an under-relaxation of the mixing length L is also performed. Four parameters define this iterative algorithm:

☞ **OUTER_ITERATIONS**: Maximum number of iterations in loop 2. If it is exceeded, the program DOES NOT stop.

☞ **INNER_ITERATIONS**: Maximum number of iterations in loops 2.1 and 2.4. If it is exceeded, the program DOES NOT stop.

☞ **TOLERANCE**: Convergence tolerance, checked in the same norm as the Navier-Stokes equations.

☞ **OUTPUT_STEPS**: Frequency of output of the convergence evolution of loops 2.1 and 2.4.

As for the Navier-Stokes equations, the data for the linear system solution must be given to complete the definition of the numerical treatment of the turbulence parameters.

7.4 Heat equation

The numerical data to solve the heat equation is the following:

```
$-----------------------------------------------------------------------------
   HEAT_EQUATION:
    STABILITY_FUNCTIONS:     MAX_CONVECTIVE: sfhec
    PERTURBATION_FUNCTION:   ``off'' & ``all'' & [convective], [diffusive]
    SHOCK_CAPTURING:         ``off'' & ``isotropic'' & ``anisotropic'',
                             PARAMETER: shche
    TEMPORAL_TERM_WEIGHTING: ``all'' & ``Galerkin''
    TIME_STRATEGY:           THETA:              thehe,
                             EULER_TIME_STEPS:   neuhe,
                             STEADY_STATE_TOLER: toshe
    CONVERGENCE_STRATEGY:    ``L2'' & ``L1'' & ``L-inf'',
                             MAX_ITERATIONS: miche, TOLERANCE: toche
    ******************
    Linear solver data
    ******************
  END_HEAT_EQUATION
$-----------------------------------------------------------------------------
```

The meaning of these commands is described next:

STABILITY_FUNCTIONS: To deal with convective dominated flows, a stabilized finite element method must be used. For the heat equation, the same method as for the Navier-Stokes equations is used. The algorithmic parameter is:

$$\tau_{\text{heat}} = \xi_0 \left[\frac{4\kappa}{h^2} + \frac{2U}{h} \right]^{-1} \tag{62}$$

where U is the norm of the mean element velocity and κ is the thermal diffusivity. The only parameter that the user has to specify is

☞ **MAX_CONVECTIVE**: Parameter ξ_0 in (62). Recommended values are 1 for linear elements and 0.5 for quadratics.

PERTURBATION_FUNCTION: This defines the perturbation functions of the stabilizing term for the heat equation, ζ_t. If ψ_h is the test function, the options are:

☞ ***Off***: $\zeta_t = 0$.

☞ ***All***: $\zeta_t = \tau_{\text{heat}}[\boldsymbol{u}_h \cdot \nabla\psi_h + \kappa\Delta\psi_h]$.

☞ ***Convective***: ζ_t contains $\tau_{\text{heat}}\boldsymbol{u}_h \cdot \nabla\psi_h$.

☞ ***Diffusive***: ζ_t contains $\tau_{\text{heat}}\kappa\Delta\psi_h$.

SHOCK_CAPTURING: This block controls the type of shock capturing strategy used for the turbulence parameters, in the same way as for the Navier-Stokes equations. There are three possible options and a parameter that have to be provided:

☞ ***Off***: No shock capturing technique is used.

☞ ***Isotropic***: $\beta_{\text{dc}} = 0$ in (60).

☞ ***Anisotropic***: $\beta_{\text{dc}} = 1$ in (60).

☞ **PARAMETER**: Value of the parameter C_{dc} in (59). Recommended values are $C_{\text{dc}} = 0.7$ for linear elements and $C_{\text{dc}} = 0.35$ for quadratics.

TEMPORAL_TERM_WEIGHTING: The user can choose whether the element residual in the stabilizing term includes or not the approximation to the temporal derivative (in a similar way as for the Navier-Stokes equations). This is specified in this command:

☞ ***All***: The element residual in the stabilizing term INCLUDES the approximation to the temporal derivative.

☞ ***Galerkin***: The element residual in the stabilizing term DOES NOT INCLUDE the approximation to the temporal derivative.

TIME_STRATEGY: Several data concerning the time integration, which is performed using the trapezoidal rule, have to be introduced here:

☞ **THETA**: Parameter θ of the generalized trapezoidal rule. Values of interest are $\theta = 1$ (backward Euler scheme) and $\theta = 1/2$ (Crank Nicolson method).

☞ **EULER_TIME_STEPS**: Number of time steps to be solved with $\theta = 1$.

☞ **STEADY_STATE_TOLER**: Tolerance to check if the steady state has been reached or not.

CONVERGENCE_STRATEGY: This command defines the way to check convergence. As for the Navier-Stokes equations, the norm in which it is evaluated must be given first, and then two parameters must be provided:

☞ ***L2***: Convergence is checked in the discrete ℓ^2 norm of the temperature.

☞ ***L1***: Convergence is checked in the discrete ℓ^1 norm of the temperature.

☞ ***L-inf***: Convergence is checked in the discrete ℓ^∞ norm of the temperature.

☞ **MAX_ITERATIONS**: Maximum number of iterations permitted per time step. For transient problems, the program keeps advancing in time even if convergence has not been achieved within a certain time step.

☞ **TOLERANCE**: Convergence tolerance, checked in the norm specified before.

As for the rest of problems solved by the code, the data for the linear system solution must be given to complete the definition of the numerical treatment of the turbulence parameters.

7.5 Free surface tracking

The final problem to be considered is the tracking of the free surface. The numerical data to solve it are:

```
$-----------------------------------------------------------------------
  FREE_SURFACE_TRACKING
    CRITICAL_VALUE:        cripc
    STABILITY_FUNCTIONS:   MAX_CONVECTIVE: sfpcc
    PERTURBATION_FUNCTION: ``off'' & ``all'' & ``convective''
    TIME_STRATEGY:         THETA:              thepc,
                           EULER_TIME_STEPS:   neupc,
                           STEADY_STATE_TOLER: tospc
    CONVERGENCE_STRATEGY:  ``L2'' & ``L1'' & ``L-inf'',
                           MAX_ITERATIONS: micpc, TOLERANCE: tocpc
    ARTIFITIAL_CONTACT:    ``on'' & ``off'', CONSTANT: arcpc
    SMOOTHING:             ``on'' & ``off'', SLOPE:    slopc
    ******************
    Linear solver data
    ******************
  END_FREE_SURFACE
$-----------------------------------------------------------------------
```

The meaning of these commands is described next:

CRTICAL_VALUE: Value of ψ_c in (29) that defines the position of the free surface. If the boundary conditions are such that ψ varies between 0 and 1, an simple choice would be $\psi_c = 0.5$.

STABILITY_FUNCTIONS: Since equation (30) is purely convective, a stabilized finite element method must be used. The SUPG method is used in this case, with algorithmic parameter (computed within each element)

$$\tau_{\text{fsu}} = \frac{\xi_0 h}{2U} \tag{63}$$

where U is the norm of the mean element velocity. The only parameter that the user has to specify is

- ☞ **MAX_CONVECTIVE**: Parameter ξ_0 in (63). Recommended values are 1 for linear elements and 0.5 for quadratics.

PERTURBATION_FUNCTION: This defines the perturbation functions of the stabilizing term for the heat equation, ζ_c. If ψ_h is the test function, the options are:

- ☞ ***Off***: $\zeta_t = 0$.
- ☞ ***All or convective***: $\zeta_c = \tau_{\text{fsu}} \boldsymbol{u}_h \cdot \nabla \psi_h$

TIME_STRATEGY: Several data concerning the time integration, which is performed using the trapezoidal rule, have to be introduced here:

- ☞ **THETA**: Parameter θ of the generalized trapezoidal rule. Values of interest are $\theta = 1$ (backward Euler scheme) and $\theta = 1/2$ (Crank Nicolson method).
- ☞ **EULER_TIME_STEPS**: Number of time steps to be solved with $\theta = 1$.
- ☞ **STEADY_STATE_TOLER**: Tolerance to check if the steady state has been reached or not.

CONVERGENCE_STRATEGY: This command defines the way to check convergence. As for the Navier-Stokes equations, the norm in which it is evaluated must be given first, and then two parameters must be provided:

- ☞ ***L2***: Convergence is checked in the discrete ℓ^2 norm of the temperature.
- ☞ ***L1***: Convergence is checked in the discrete ℓ^1 norm of the temperature.
- ☞ ***L-inf***: Convergence is checked in the discrete ℓ^∞ norm of the temperature.
- ☞ **MAX_ITERATIONS**: Maximum number of iterations permitted per time step. For transient problems, the program keeps advancing in time even if convergence has not been achieved within a certain time step.

- ☞ **TOLERANCE**: Convergence tolerance, checked in the norm specified before.

ARTIFITIAL_CONTACT: If the velocity on the walls is zero (no slip condition) the fluid front can not advance there. A possible way to artificially overcome this problem is to modify the values of ψ at the boundary nodal points. `Fantom` has the following method implemented. Let ψ_m be the mean value of the function ψ for an element Ω^e adjacent to the boundary. The condition $\psi_m \geq \psi_c$ (critical value) will indicate that most of the nodes of the element have already been filled. In this situation, the value of ψ for a node located at the boundary, ψ_b, is modified as follows:

$$\psi_b \leftarrow \psi_b + \rho(\psi_m - \psi_b) \tag{64}$$

where $0 \leq \rho \leq 1$. The commands to be provided are:

- ☞ ***Off***: The artifitial contact IS NOT activated.
- ☞ ***On***: The artifitial contact is used.
- ☞ **CONSTANT**: Value of ρ in (64).

SMOOTHING: Even if the initial condition $\psi_0(\boldsymbol{x})$ is a smooth function, if the ψ is mantained unmodified over several time steps it may begin to lose its smoothness and numerical problems may be encountered. Since the only important factor is the location of the critical contour that defines the front, it is possible to smooth ψ while maintaining the position of this critical contour. This can be performed redefining the free surface function for each node of the finite element mesh according to the following expression:

$$\psi = \psi_c + \mathrm{sgn}(\psi_o - \psi_c)\ \sigma\ d \tag{65}$$

where ψ_o stands for the calculated value of ψ, σ is a given constant, d is the distance from the node under consideration to the front and $\mathrm{sgn}(\cdot)$ is the signum of the value enclosed in the brackets. The commands to be provided are:

- ☞ ***Off***: The smoothing IS NOT activated.
- ☞ ***On***: The smoothing is used.
- ☞ **SLOPE**: Value of σ in (65).

As for the rest of problems solved by the code, the data for the linear system solution must be given to complete the definition of the numerical treatment of the turbulence parameters.

7.6 Linear solver data

For all the problems to be solved, a block with the information needed to solve the linear system of equations encountered after discretization and linearization must be given. This information must be included in the block where the numerical strategy of a particular problem is defined. For the Navier-Stokes equations it must be included in the `SOLUTION_PROCEDURE` block.

`Fantom` uses the package `MatMan` (Matrix Manager) to deal with all the linear systems of equations appearing during the run. Therefore, the options of `Fantom` correspond in fact to the options of `MatMan`.

The format of the linear solver data is:

```
ALGEBRAIC_SOLVER:     ``ldu_dec''   & ``ee_cg''      & ``ee_gmres''  &
                      ``sk_cg''     & ``sk_cgnr''    & ``sk_bcg''    &
                      ``sk_dbcg''   & ``sk_stabbcg'' & ``sk_tfqmr''  &
                      ``sk_fom''    & ``sk_gmres''   & ``sk_fgmres'' &
                      ``sk_dqgmres''
                      MAX_ITER:        miter, TOLERANCE: toler,
                      KRYLOV_DIMENSION: krydi
For direct solvers:
  MATRIX_STORAGE:           ``banded'' & ``profile'' & ``sparse''
  RENUMBERING_STRATEGY:     ``on''     & ``off''
For iterative solvers:
  PRECONDITIONING_STRATEGY: ``off''    & ``diagonal'' & ``l_ilut'' &
                            ``r_ilut'' & ``l_r_ilut'',
                            FILL_IN: filin, THRESHOLD: thres
```

The meaning of the different options and parameters is the following:

ALGEBRAIC_SOLVER: This defines the type of solver to be used. There is the possibility of using a DIRECT method or a variety of ITERATIVE METHODS. The latter can be distinguished by the way in which matrix-vector products are performed. They can be done either element by element or constructing a CSR (compressed sparse row) global matrix and performing the product at the assembled level. In what follows, these two options are defined by the initials ***ee_*** and ***sk_***, respectively. These iterative methods need also some parameters to be defined:

- ☞ ***ldu_dec*** : Direct solver using a LDU decomposition.
- ☞ ***ee_cg*** : Conjugate Gradient Method with element by element matrix-vector product.
- ☞ ***ee_gmres*** : GMRES method with element by element matrix-vector product.
- ☞ ***sk_cg*** : Conjugate Gradient Method.

- ☞ ***sk_cgnr*** : Conjugate Gradient Method (Normal Residual equation).
- ☞ ***sk_bcg*** : Bi-Conjugate Gradient Method.
- ☞ ***sk_dbcg*** : BCG with partial pivoting.
- ☞ ***sk_stabbcg***: BCG stabilized.
- ☞ ***sk_tfqmr*** : Transpose-Free Quasi-Minimum Residual method.
- ☞ ***sk_fom*** : Full Orthogonalization Method.
- ☞ ***sk_gmres*** : Generalized Minimum RESidual method.
- ☞ ***sk_fgmres*** : Flexible version of GMRES method.
- ☞ ***sk_dqgmres***: Direct versions of Quasi GMRES method.
- ☞ **MAX_ITER**: Maximum number of iterations allowed for iterative methods.
- ☞ **TOLERANCE**: Convergence tolerance for iterative methods. If convergence is not achieved, the code DOES NOT stop. This convergence is always checked in the discrete ℓ^2 norm of the unknown being computed.
- ☞ **KRYLOV_DIMENSION**: Krylov dimension in the case of iterative methods based on Krylov subspace minimization (only needed for FOM, GMRES, FGMRES and DQGMRES).

MATRIX_STORAGE: This option must be provided ONLY FOR DIRECT methods. The possibilities are:

- ☞ ***Banded***: Band storage of the system matrix.
- ☞ ***Profile*** (or ***skyline)***: Profile storage of the system matrix.
- ☞ ***Sparse***: Sparse storage of the system matrix. This option is not very efficient to perform the LDU decomposition. However, IS THE ONLY storage that allows PIVOTING, which might be necessary if the Galerkin method is used to solve the Navier-Stokes equations, since in this case the submatrix that multiplies the pressure nodal values is zero in the continuity equation.

RENUMBERING_STRATEGY: This option must be provided ONLY FOR DIRECT methods. The options are:

- ☞ ***On***: The renumbering strategy is activated. Internally, `Fantom` uses a different renumbering algorithm according to storage type (banded and sparse on the one hand side and profile on the other).
- ☞ ***Off***: No renumbering is used.

PRECONDITIONING_STRATEGY: This option must be provided ONLY FOR ITERATIVE methods. The options and the parameters that some of them need are:

- ☞ ***Off***: No preconditioning is used.
- ☞ ***Diagonal***: Only a diagonal scaling is used as preconditioning. This is the recommended option, especially for the Navier-Stokes equations, since the next options are expensive and the optimal values of the parameters they need difficult to determine.
- ☞ ***l_ilut***: Left ILU (incomplete LU) preconditioning. Only allowed for SK type iterative schemes.
- ☞ ***r_ilut***: Right ILU (incomplete LU) preconditioning. Only allowed for SK type iterative schemes.
- ☞ ***l_r_ilut***: Left and right ILU (incomplete LU) preconditioning. Only allowed for SK type iterative schemes.
- ☞ **FILL_IN**: This parameter is only needed for ILU preconditioners. It is the fill-in.
- ☞ **THRESHOLD**: This parameter is only needed for ILU preconditioners. It is the threshold of the ILU methods.

According to the options described, there will be two typical situations normally specified. When direct methods are used, an example of solver data could be

```
ALGEBRAIC_SOLVER:     LDU_DEC
MATRIX_STORAGE:       PROFILE
RENUMBERING_STRATEGY: ON
```

whereas a typical situation using an iterative scheme would be:

```
ALGEBRAIC_SOLVER: SK_GMRES, MAX_ITER= 200, TOLER=1.0e-5, KRYLOV=20
PRECONDITIONING_STRATEGY: L_ILUT, FILL_IN = 5 , THRESHOLD = 1.0e-5
```

8 Output and post–process

`Fantom` writes the results of the calculation in a formatted and in a binary file (see the following section). The specification of the results that have to be written and how often this must be done is contained in the following block of data:

```
$************************************************************************
OUTPUT_&_POST-PROCESS
  OUTPUT MESH_DATA
  OUTPUT FIXITY_CONDITIONS
  OUTPUT ERRORS,             SOLUTION:
  OUTPUT VELOCITY,           STEPS:, AT_TIME:
  OUTPUT PRESSURE,           STEPS:, AT_TIME:
  OUTPUT STRESSES,           STEPS:, AT_TIME:
  OUTPUT VORTICITY,          STEPS:, AT_TIME:
  OUTPUT STREAM_LINES,       STEPS:, AT_TIME:
  OUTPUT TEMPERATURE,        STEPS:, AT_TIME:
  OUTPUT MATERIAL_FRONT,     STEPS:, AT_TIME:
  OUTPUT DENSITY             STEPS:, AT_TIME:
  OUTPUT VISCOSITY           STEPS:, AT_TIME:
  OUTPUT POROSITY            STEPS:, AT_TIME:
  POSTPROCESS VELOCITY,      STEPS:, AT_TIME:
  POSTPROCESS PRESSURE,      STEPS:, AT_TIME:
  POSTPROCESS STRESSES,      STEPS:, AT_TIME:
  POSTPROCESS VORTICITY,     STEPS:, AT_TIME:
  POSTPROCESS STREAM_LINES,  STEPS:, AT_TIME:
  POSTPROCESS TEMPERATURE,   STEPS:, AT_TIME:
  POSTPROCESS MATERIAL_FRONT, STEPS:, AT_TIME:
  POSTPROCESS DENSITY,       STEPS:, AT_TIME:
  POSTPROCESS VISCOSITY,     STEPS:, AT_TIME:
  POSTPROCESS POROSITY,      STEPS:, AT_TIME:
END_OUTPUT_&_POST-PROCESS
$************************************************************************
```

The meaning of these commands can be understood from the following rules:

☞ All the lines are OPTIONAL.

☞ All the commands starting with the word `OUTPUT` correspond to results written in the formatted file.

☞ All the commands starting with the word `POSTPROCESS` correspond to results written in the binary file.

☞ The command `OUTPUT MESH_DATA` forces the code to dump the geometry data in the results file.

☞ The command `OUTPUT FIXITY_CONDITIONS` forces the code to dump the boundary conditions in the results file.

☞ The command `OUTPUT ERRORS` forces `Fantom` to compute the error of the finite element solution of the Navier-Stokes equations with respect to an analytical solution that has to be coded by the user in the routine `exacso.f`. The integer `SOLUTION` specifies the number of exact solution chosen.

☞ For all the variables to be output, two parameters can be provided:

- `STEPS`: frequency of output in terms of time steps. This is a MANDATORY command for each variable to be written.
- `AT_TIME`: specific times at which the unknowns are written. There can be at most FIVE times. `Fantom` checks if the current time corresponds to any of these (with an error of order the time step size) and, if so, writes the varible. This command is OPTIONAL.

☞ The streamlines are computed only for 2D FLOWS in the form of streamfunction.

☞ Pressure, stresses and vorticity are computed at the nodes of the mesh. For that, the smoothing technique specified in the general numerical treatment is used (for the pressure, this is necessary only if it is discontinuous).

☞ Writing the physical properties makes sense only when they depend on the unknowns of the problem.

9 How to use `Fantom`

The executable version of `Fantom` needs a classical data file to run and creates some files with the results. Even though these may be integrated into more general pre- and post-process packages, in the following it is described how to run `Fantom` *independently* of the pre- and post-process, using only simple files.

To run `Fantom` in a more comfortable way, it is also described how to use a UNIX script that is in charge of all the management of files. The extension to other environments is, of course, also possible.

9.1 Files used by `Fantom`

First of all, let us describe the files that `Fantom` needs to run and creates. All these files have as name the model to be run followed by an extension, that is, they are of the form ***model.extension***.

All the data described in this manual must be contained in a single file:

☞ ***model.dat:*** File containing all the data. Since the geometry and the fixity conditions are very long, they may be included in other files linked to the main data file with an `INCLUDE` statement followed by the path of these files. Typically, the name of these files could be

- ***model.geo:*** *Possible* name of the geometry file.
- ***model.fix:*** *Possible* name of the file with fixity conditions.

On the other hand, `Fantom` creates several files as a result of the run. These files are:

☞ ***model.dat.bac:*** This is a copy of the main data file, copied in the directory where the results are (see the next subsection).

☞ ***model.res:*** Formatted file with the results. It contains the following information:

- Description of the problem that has been run.
- Summary of memory requirements.
- Some warnings and errors from the data file.
- Forces and torques on the boundaries (those where they have been required).
- Numerical results that have been required with the command `OUTPUT`.
- Summary of computing times.

☞ ***model.log:*** Formatted file with information about the evolution of the run. It contains the following information:

- Summary of memory requirements.
- Convergence checks for all the variables of the run.
- Summary of computing times.

☞ ***model.sol*** Formatted file with information about the solution of the linear systems of equations dealt with in the code. For each problem being solved, it gives information about:

- Permanent memory requirements once per problem.
- Temporary memory requirements every time a problem is solved.
- CPU time needed every time a problem is solved.
- Convergence evolution in the case of iterative solvers.

☞ ***model.pos:*** Binary file with the results to be post-processed. It contains the geometry and the results that have been required with the command `POSTPROCESS`. These results can be transformed into ascii data to be postprocessed by the code `Fantout`, which has to be modified to write in the format of the postprocessor to be used. This code has to take into account whether the length of the integers used in the calculation is 4 or 8. This depends on the architecture of the computer. `Fantout` is also supplied with `Fantom.`

9.2 `Fantom's` script `fantom.exe`

All the files that `Fantom` uses are better managed by using a UNIX script. In its present version (many modifications are obviously possible) this script, called `fantom.exe`, is in charge of performing the following tasks:

☞ It reads the name of the model IN THE DIRECTORY WHERE IT IS RUN.

☞ Using the name read, it creates a new directory in the path assigned by the user with the name of the model. All the results files will be placed in this directory. The path must be indicated in the file of the script itself.

☞ It assigns to `Fantom` the name of the data and results files by setting an environment variable.

☞ It allows to choose between the debugger, the optimized and the OK version of `Fantom.` The optimized version may be created with the optimization options available in the compiler used, and the OK version is any stable version renamed with the extension `.ok`. The path of the executable must be also written in the file of the script itself.

☞ It creates some addtional files in the directory of results:

- ***model.tim***: File created by the system, with the total CPU time and the errors of the run, if any.
- ***Lock.USER***: File with extension the name of the user. It prevents running a model that is being analyzed.
- ***Version.USER.VER*** Name of the version used. USER is the name of the user and VER the name of the version.

9.3 The directory `fantom.free`

`Fantom` can be downloaded from `http://www.rmee.upc.es/homes/codina`, my web site . There you will find a tar file with the directory `fantom.free`, which contains all the files needed to run `Fantom`. These are organized into the following directories:

`Executables`. Directory with *executable* files. It contains the executables of the code (all the versions), a makefile and a code to compile all the routines with a certain header (used as `makeheading` *name*). The Makefile can be modified according to the compiling options of the computer being used.

`Fantout`. Directory with the code `Fantout`. It contains the source files and a Makefile to create the executable of this code.

`Manual`. Directory with all the LaTeX files to create the present manual.

`Objects_O2`. Directory with objects created using a O2 version. This is used by the Makefile of `Fantom` (the objects are put there when the Makefile is invoked). Obviously, other options apart from this O2 may be used as well.

`Objects_g`. Directory with objects created using debugger version.

`Tests_dat`. Directory with some numerical examples, each in a different directory.

`Tests_res`. Directory with the results of the examples in `Tests_dat`.

Directories in lower case contain the source routines of the code.