1.INTRODUCTION

This document constitutes the user manual for TRUST/TrioCFD software. It supersedes the previous document.

TRUST/TrioCFD is a thermohydraulic calculation modular software package. The two currently available modules include a VDF calculation module “Finite Difference Volume” and a VEF calculation module “Finite Element Volume”.

The VDF and VEF modules are designed to process the 2D or 3D flow of Newtonian, incompressible, weakly expandable fluids the density of which is a function of a local temperature and concentration values (Boussinesq approximation).

2.DATA SET DESCRIPTION

2.1BASIC RULES

There is no line concept in TRUST.
A block may be defined using the braces:

```
{
	a block
}
```

Objects are created in the data set as follows:

*[***export***] Type identificateur*

export: if this keyword is included, *identificateur (identifier)* will have a global range, if not its range will apply to the block only (the associated object will be destroyed on exiting the block).

Type: every type of object recognised by TRUST. The list of types recognised is given in the file **hierarchie.dump**.

identificateur: the identifier of the object type *Type* created. TRUST exits in error if the identifier has already been used.

Interprete (interpretor) type objects are then used to handle the created objects with the following syntax:

Type_interprete argument

Type_interprete: any type derived from the **Interprete** (Interpretor) type recognised by TRUST. In this manual, they are written in bold.

argument: an argument may comprise one or several object identifiers and/or one or several data blocks (refer to Interpretes Généraux (General Interpretors) in 2.3).

To insert comments in the data set, use # .. # (or /* …. */) the character # must always be enclosed by blanks. Since the 1.6.1 version, comments can be inserted everywhere in the data file not only between interpretors.

Examples:

• A data set to write Ok on screen:

```
Nom un_nom                 # Creation of an object type. Name identifier un_nom #
Read un_nom Ok                    # Allocates the string "Ok" to un_nom #
Ecrire un_nom                     # Write un_nom on screen #
```

• An incorrect data set:

```
Domaine truc
...
Probleme truc                     # TRUST exits in error #
```

A possible correction:

```
{
Domaine truc
}                     # The domain truc is destroyed #
Probleme truc         # this is correct because truc is not used any more #
```

• One data set nesting another:

```
Read _file fichier_inclus ;   # you should use export in
                              the fichier_inclus to export identifiers #
```

example of the fichier_inclus file:

```
Dimension 2
export Domaine dom
export Probleme_hydraulique pb
```

Observations:

• The semi-colon is no longer an instruction separator as it was in TRIO-VF.

• The comma separates items in a list (a comma must be enclosed with spaces or a new line).

• Interpretor keywords are recognised indiscriminately whether they are written in lower and/or upper case. ***On the contrary, object names (identifiers) are recognised differently if they are written in upper or lower case.***

• Object names may not exceed 999 characters.

• ***In the following description, items (keywords or values) enclosed by [and] are facultative.***

2.2 OBJECTS

There are several object types.

Physical objects, for example:

• A block object (keyword **Pave**) is defined by its origin and dimensions (keyword **origine (origin)** and **longueurs (length))**. Discretization is given by the **nombre_de_noeuds (node number)** in each direction.

• A **Fluide_incompressible (incompressible_Fluid)** object. This type of object is defined by its physical characteristics (its dynamic viscosity μ (keyword **mu**), its density ρ (keyword **rho**), etc...)

• A **Domaine**.

More abstract object types also exist:

• A **VDF** or **VEF** according to the discretization type.

• **Schema_euler_explicite** to indicate the scheme type.

• A **Solveur_pression** to denote the pressure system solver type.

• A **Champ_Uniforme** to define, for example, the gravity field.

2.3 INTERPRETORS

Interpretors allow some operations to be carried out on objects. Currently available general interpretors include **Read**, **Read_file, Ecrire (Write), Ecrire_fichier, (Write_file), Associate**. Other interpretors shall be described further on.

2.3.1READ

The **Read** interpretor allows an object to be read (defined) in various ways:

```
Read objet
{
   ....
}
```

Read: Keyword to read the object *objet* defined between the braces.

```
Read_file nomfic ;
```

Read_file: Keyword to execute a data set given in the *nomfic* file (a space must be entered between the semi-colon and the file name).

```
Read_file objet  nomfic
```

Read _file: Keyword to read the object *objet* contained in the file *nomfic*. This is notably used when the calculation domain has already been meshed and the mesh contains the file *nomfic*, simply write (where *dom* is the name of the meshed domain):

Read _file dom nomfic

Read _file_binary objet nomfic

Read _file_binary: Keyword to read an object *objet* in the unformatted file type *nomfic*.

Lire_Tgrid domain_name filename.msh

Lire_Tgrid : Keyword to read Tgrid/Gambit mesh files. 2D (triangles or quadrangles) and 3D (tetra or hexa elements) meshes, may be read by TRUST.

Lire_Ideas domain_name filename.unv

Lire_Ideas : Keyword to read Ideas unv mesh files. 3D tetra mesh elements only may be read by TRUST.

Lire_MED [vef] [**family_names_from_group_names**|**short_family_names**] domain_name mesh_name filename.med

Lire_MED : Keyword to read MED mesh files where domain_name corresponds to the domain name, filename.med corresponds to the file (written in format MED) containing the mesh named mesh_name. Option **vef** is obsolete and is kept for backward compatibility. The option **family_names_from_group_names** uses the group names instead of the family names to detect the boundaries into a MED mesh (useful when trying to read a MED mesh file from Gmsh tool which can now read and write MED meshes). The option **short_family_names** is useful to suppress FAM_-*_ from the boundary names of the MED meshes.

Note about naming boundaries: When reading filename.med, TRUST will detect boundaries between domain (Raccord) when the name of the boundary begins by "type_raccord_". For example, a boundary named "type_raccord_wall" in filename.med will be considered by TRUST as a boundary named wall between two domains.

NB: To read several domains from a mesh issued from a MED file, use **Lire_Med** to read the mesh then use **Create_domain_from_sous_zone** keyword (see 4.3.12 chapter).
NB: If the MED file contains one or several subzone defined as a group of volumes, then **Lire_MED** will read it and will create two files *domain_name_ssz.geo* and *domain_name_ssz_par.geo* defining the subzones for sequential and/or parallel calculations. These subzones will be read in sequential in the datafile by including (after **Lire_Med** keyword) something like:
Lire_Med ....
Read_file *domain_name*_ssz.geo ;

During the parallel calculation, you will include something:
Scatter { ... }
Read_file *domain_name*_ssz_par.geo ;

2.3.2WRITE

The **Ecrire (Write)** interpretor allows an object to be written to a file or a standard outlet.

Ecrire objet

Ecrire: Keyword to write the object objet to a standard outlet.

Ecrire_fichier objet nomfic

Ecrire_fichier: Keyword to write the object objet to a file *nomfic*. Since the v1.6.3, the default format is now binary format file.

A file may be written in binary format with:

Ecrire_fichier_Bin objet nomfic

It is interesting to write in binary format big meshes for example cause it is very quick to read it compare to formatted format and it needs more than twice less memory on disc.

A file may be written in ASCII format with:

```
Ecrire_fichier_Formatte  objet  nomfic
```

```
Postraiter_domaine
{
  format name
  [binaire 0|1]
  [fichier name]
  [joints_non_postraites 0|1 ]
  [ecrire_frontiere 0|1]
  domaine name  | domaines { name1 name2 … }
}
```

To write one or more domains in a file with a specified format (MED, LML,LATA). By default, the name of the file will be datafile_name”.”format. The file name can be changed with the **fichier** option. The **ecrire_frontiere** option will write (if set to 1, the default) or not (if set to 0) the boundaries as fields into the file (it is useful to not add the boundaries when writing a domain extracted from another domain). The **joints_non_postraites** (1 by default) will not write the boundaries between the partitioned mesh. Binary (binaire 1) or ASCII (binaire 0) may be used. By default, it is 0 for LATA and only ASCII is available for LML and only binary is available for MED.

Ecrire_MED: Keyword to write a domain to MED format into a file:

```
Ecrire_MED  domain_name   filename
```

Ecrire_Champ_MED: Keyword to write a field to MED format into a file. Useful for Homard.

```
Ecrire_Champ_MED  domain_name  field_name  filename
```

2.3.3ASSOCIATE

The A**ssociate** interpretor allows one object to be associated with another.

```
Associate objet1  objet2
```

The order of the two objects in this instruction is not important.
The object *objet2* is associated to *objet*1 if this makes sense; if not either *objet1* is associated to *objet2* or the program exits in error because it cannot execute the **Associate** instruction.

For example, to calculate water flow in a pipe, a **Pb_Hydraulique** type object needs to be defined. But also a **Domaine** type object to represent the pipe, a **Schema_euler_explicite** type object for time discretization, a discretization type object (**VDF** or **VEF**) and a **Fluide_Incompressible** type object which will contain the water properties. These objects must then all be associated with the problem.

2.3.4GEOMETRIC INSTRUCTIONS

```
Dimension dim
```

This instruction is mandatory**.**

Dimension: Keyword allowing calculation dimensions to be set (2D or 3D). where *dim* is an integer set to 2 or 3.

```
Axi
```

This instruction is facultative.

Axi: This keyword allows a 3D calculation to be executed using cylindrical co-ordinates (R,θ,Z). If this instruction is not included, calculations are carried out using Cartesian co-ordinates.

Bidim_Axi

This instruction is facultative.

Bidim_Axi: Keyword allowing a 2D calculation to be executed using axisymetric co-ordinates (R, Z). If this instruction is not included, calculations are carried out using Cartesian co-ordinates.

Domaine dom

Keyword to create a domain where dom is the name.

Domaine_ALE dom

Keyword to create a domain where dom is the name with nodes at the interior of the domain are displaced in an arbitrarily prescribed way thanks to ALE description.

2.3.5 MESH

```
Mesh dom
{
    [Epsilon  ε ]
    ,
    [objet1]
    ,
    [objet2]
    .....

}
```

The **Mesh** interpretor allows a **Domain** type object *dom* to be meshed with objects *objet1*, *objet2*, etc...

Two points will be confused if the distance between them is less than **ε**. By default, ε is set to 10^{-12}. The keyword **Epsilon** allows an alternative value to be assigned to ε.

Currently, the two types of objects recognised by TRUST to mesh a domain are the **Domain** or the **Pave (block)** object. For example, to mesh a domain dom with 3 another domains domA domB and domC (it is important that boundaries are not defined on the matching edges of the domains ; notice that the interpreter **supprime_bord** allows to remove a boundary from a domain see §2.3.32):

Mailler dom { **Domain** domA , **Domain** domB , **Domain** domC }

The object **Pave** is defined as follows:

```
Pave nom_pave
{
    Origine OX OY [OZ]
    Longueurs LX LY [LZ]
    Nombre_de_noeuds NX NY [NZ]
    Facteurs [FX] [FY] [FZ]
     [Symx] [Symy] [Symz]
     [Tanh value]
    [Tanh_dilatation value]
    [Tanh_taille_premiere_maille value]
}
{
     [contact nom_cote X= X0 Y0 <= Y <= Y1 [Z0 <= Z <= Z1]]
     [contact nom_cote Y= Y0 X0 <= X <= X1 [Z0 <= Z <= Z1]]
     [contact nom_cote Z= Z0 X0 <= X <= X1 [Y0 <= Y <= Y1]]

}
```

Origine: Keyword to define the pavé (block) origin, that is to say one of the 8 block points (or 4 in a 2D system).

OX: X co-ordinates
OY: Y co-ordinates
OZ: Z co-ordinates

Longueurs: Keyword to define the block dimensions, that is to say knowing the origin, length along the axes.

LX: Length along X
LY: Length along Y
LZ: Length along Z

Facteurs: Keyword to define stretching factors for mesh discretization in each direction. This is a real number which must be positive (by default 1.0).

FX: Stretch factor along X
FY: Stretch factor along Y
FZ: Stretch factor along Z

Application to cylindrical co-ordinates:

The same **Pave** (block) object is applied to cylindrical co-ordinates (and the same keywords) **X = , Y = , Z =**) by means of two restrictions:

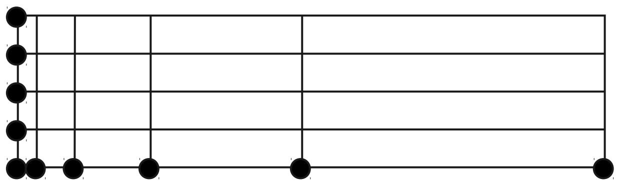
X must always correspond to R, and Y to θ.

The values entered into the block for lengths in θ or co-ordinates in θ must correspond to real values given in radians divided by 2π (so in number of turns).

For example:

Facteurs 1.2 1.0

The mesh size in the X direction increases by a factor of 1.2. In Y, the mesh will be regular. The design hereunder illustrates the example of a block where **NX**=6 and **NY**=5:



A stretching factor other than 1 allows refinement on one edge in one direction. To achieve refinement along the two edges of the block in one direction, the following keywords may be used:

Symx: Keyword to define a block mesh that is symmetrical with respect to the YZ plane (respectively straight Y in 2D) passing through the block centre.

Symy: Keyword to define a block mesh that is symmetrical with respect to the XZ plane (respectively straight X in 2D) passing through the block centre.

Symz: Keyword defining a block mesh that is symmetrical with respect to the XY plane passing through the block centre.

Tanh: Keyword to generate mesh with tanh (hyperbolic tangent) variation.

Tanh_dilatation New keyword to generate mesh with tanh (hyperbolic tangent) variation.

tanh_dilatation: The value may be -1,0,1 (0 by default):

0: coarse mesh at the middle of the channel and smaller near the walls

1: coarse mesh at the bottom of the channel and smaller near the top
-1: coarse mesh at the top of the channel and smaller near the bottom

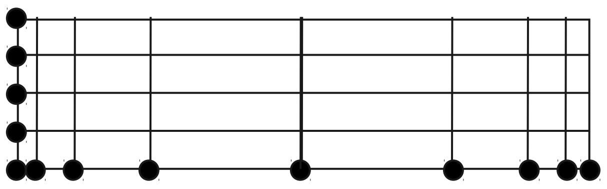
Tanh_taille_premiere_maille New keyword to generate mesh with tanh (hyperbolic tangent) variation in the Y direction.

tanh_taille_premiere_maille: size of the first cell of the mesh.

Example:
Pave Cavite
{
    Origine 0. 0. 0.
    Nombre_de_Noeuds 10 65 10
    Longueurs 6.283185307 2.0 3.141592653
    tanh_dilatation 0
    tanh_taille_premiere_maille 0.01
    }
}

Example:

The same block as previously described, where **NX**=9 and the keyword **Symx** is selected.



contact: Keyword indicating the block side type. This could be **Bord** to indicate that the block side is not in contact with another block and that limitation conditions will be applied to it, **Raccord Local Homogene (Connector)** to indicate (each of theses 3 keywords are separated by a space) that the block side is in contact with the block of another domain (case of two coupled problems), **Internes (Internal)** to indicate that the block has a set of internal faces (these faces will be duplicated automatically by the program and will be processed in a manner similar to edge faces). Two boundaries with the same limitation conditions may be given the same name (whether or not they belong to the same block).

The keyword **Internes (Internal)** must be used to execute a calculation with plates, followed by the equation of the surface area covered by the plates.

Block sides that are neither edges nor connectors are not specified. The duplicate nodes of two blocks in contact are automatically recognised and deleted.

nom_cote: name of the block side.

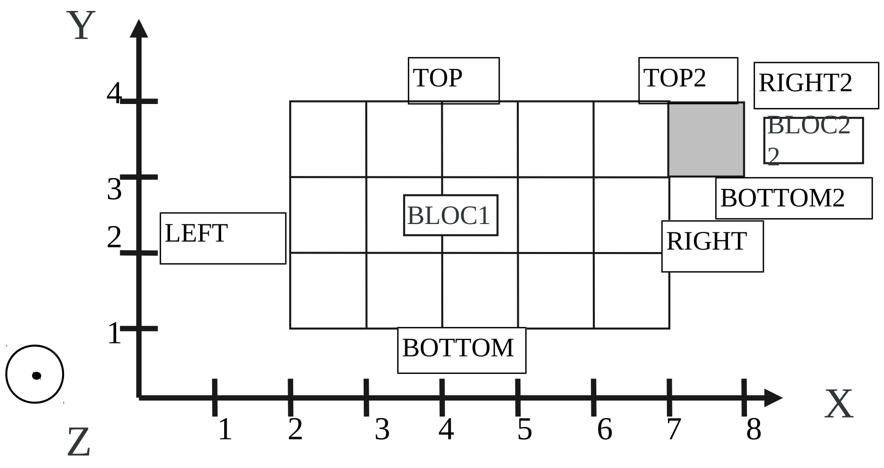
X0 Y0 Z0 X1 Y1 Z1: block side co-ordinates. ***Note spaces between the coordinate values and the keywords = and <=.***

Example (notice the comma between the description of each block **pave**):

```
pave BLOC1
{
        origine 2 1
        nombre_de_noeuds 6 4
        longueurs 5.0 3.0
}
{
        bord TOP  Y = 4  2 <= X <= 7
        bord BOTTOM  Y = 1  2 <= X <= 7
        bord LEFT  X = 2  1 <= Y <= 4
        bord RIGHT  X = 7  1 <= Y <= 3
} ,
pave BLOC2
{
        origine 7 3
        nombre_de_noeuds 2 2
        longueurs 1.0 1.0
}
{
```

bord RIGHT2 **X** = 8 3 <= **Y** <= 4
bord TOP2 **Y** = 4 7 <= **X** <= 8
bord BOTTOM2 **Y** = 3 7 <= **X** <= 8
}

2 meshed blocks will be produced with this method (note the XYZ trihedral orientation in TRUST):



Observations: The side shared by BLOCK1 and BLOCK2 will be detected but will not be specified in the definition of the two blocks.

2.3.6MAILLERPARALLEL(PARALLEL MESH)

MaillerParallel creates a parallel distributed hexaedral mesh of a parallelipipedic box. It is equivalent to creating a mesh with a single Pave, splitting it with “partition” and reloading it in parallel with “Scatter”. **MaillerParallel** only works in 3D at this time. It can also be used for a sequential computation (with all NPARTS=1)

```
MaillerParallel {
      domain       domaine_name
      nb_nodes     dimension   nX  nY  nZ
      splitting    dimension   npartsX   npartsY   npartsZ
      ghost_thickness   nghost
      [ perio_x ]
      [ perio_y ]
      [ perio_z ]
      [ function_coord_x   funcX |  file_coord_x   fileX ]
      [ function_coord_y   funcY |  file_coord_y   fileY ]
      [ function_coord_z   funcZ  |  file_coord_z   fileZ ]
      [ boundary_xmin    name_Xmin  ]
      [ boundary_xmax    name_Xmax ]
      [ boundary_ymin    name_Ymin  ]
      [ boundary_ymax    name_Ymax ]
      [ boundary_zmin    name_Zmin   ]
      [ boundary_zmax    name_Zmax  ]
}
```

domain : the name of the domain to mesh (it must be an empty domain object).

nb_nodes : dimension defines the spatial dimension (currently only dimension=3 is supported), and nX, nY and nZ defines the total number of nodes in the mesh in each direction.

splitting : dimension is the spatial dimension and npartsX, npartsY and npartsZ are the number of parts created. The product of the number of parts must be equal to the number of cpus used for the computation.

ghost_thickness : nghost is the number of ghost cells (equivalent to the epaisseur_joint parameter of partition).

perio_x, **perio_y** and **perio_z** : change the splitting method to provide a valid mesh for periodic boundary conditions.

function_coord_x|y|z : By default, the meshing algorithm creates nX|nY|nZ coordinates ranging between 0 and 1 (eg a unity size box). If **function_coord_x|y|z** is specified, it is used to transform the [0,1] segment to the coordinates of the nodes. funcX must be a function of the x variable only, funcY of y and funcZ of z.

For example:

function_coord_y (y-1)*2

will create a box with a uniform mesh over [-1,1] in the y coordinates.

file_coord_x|y|z : is specified to read in fileX|Y|Z the nX|nY|nZ floating point values used as nodes coordinates.

boundary_xmin : the name of the boundary at the minimum X direction. If it not provided, the default boundary names are xmin, xmax, ymin, ymax, zmin and zmax. If the mesh is periodic in a given direction, only the MIN boundary name is used, for both sides of the box.

2.3.7REMOVE_ELEM

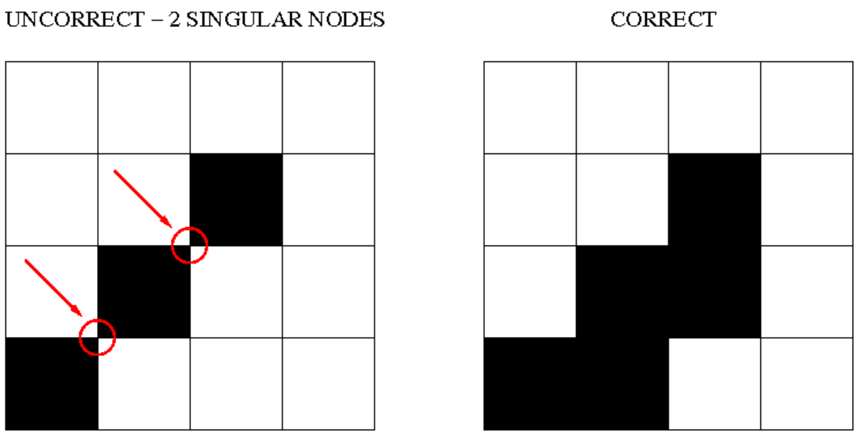
Remove_elem domain_name { **liste** integer elem0 elem1 elem2 … }

Remove_elem domain_name { **fonction** condition }

Keyword to remove element from a VDF mesh (named domaine_name), either from an explicit list of elements or from a geometric condition defined by a condition f(x,y)>0 in 2D and f(x,y,z)>0 in 3D. All the new borders generated are gathered in one boundary called : newBord (to rename it, use **RegroupeBord** keyword. To split it to different boundaries, use **DecoupeBord_Pour_Rayonnement** keyword). Example of a removed zone of radius 0.2 centered at (x,y)=(0.5,0.5):

Remove_elem dom { **fonction** 0.2*0.2-(x-0.5)^2-(y-0.5)^2>0 }

Warning : the thickness of removed zone has to be large enough to avoid singular nodes as decribed below :



2.3.8REORDONNER(REORDER)

The **Reordonner** interpretor is required sometimes for a VDF mesh which is not produced by the internal mesher.

Example where this is used:

Read _file *dom* fichier.geom

Reordonner *dom*

Observations:

This keyword is redundant when the mesh that is read is correctly sequenced in the TRUST sense. This significant mesh operation may take some time…

The message returned by TRUST is not explicit when the Reordonner (Resequence) keyword is required but not included in the data set…

2.3.9 CORRIGER_FRONTIERE_PERIODIQUE

The **Corriger_frontiere_periodique** keyword is mandatory to first define the periodic boundaries, to reorder the faces and eventually fix unaligned nodes of theses boundaries. Faces on one side of the periodic domain are put first, then the faces on the opposite side, in the same order. It must be run in sequential before mesh splitting.

```
Corriger_frontiere_periodique  {
            domaine domain_name
            bord boundary_name
           [ direction 2|3 dx dy [dz] ]
           [ fichier_post filename ]
}
```

domaine: domain_name is the name of the domain

bord: boundary_name is the name of the boundary (which must contain two opposite sides of the domain)

direction: dx dy dz defines the periodicity direction vector (a vector that points from one node on one side to the opposite node on the other side. This vector must be given if the automatic algorithm fails, that is:

- when the node coordinates are not perfectly periodic
- when the periodic direction is not aligned with the normal vector of the boundary faces

Corriger_frontiere_periodique replaces and improves the **Reordonner_faces_periodique** keyword which becomes obsolete and is kept for backward compatibility:

Reordonner_faces_periodiques DOM BORD

Is a shortcut to:

Corriger_frontiere_periodique { **domaine** DOM **bord** BORD }

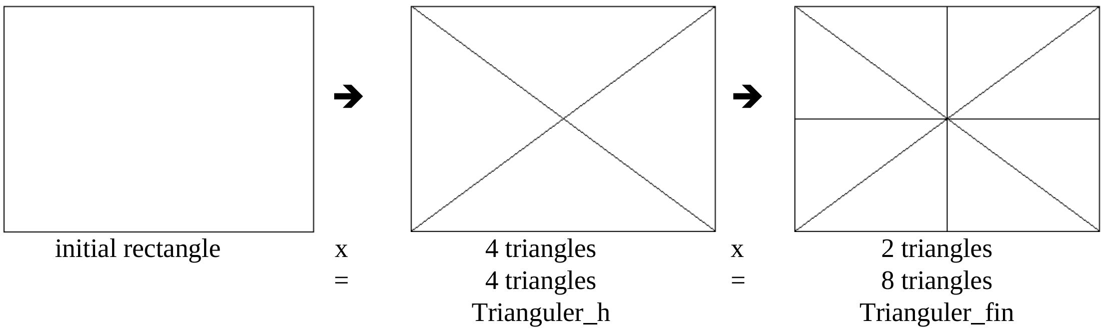
2.3.10TRIANGULATE

Trianguler_fin is the recommended option to triangulate rectangles.

```
Trianguler_fin  nom_domaine
```

Trianguler_fin : As an extension (subdivision) of **Triangulate_h** option, this one cut each initial rectangle in 8 triangles (against 4, previously). This cutting ensures :
- a correct cutting in the corners (in respect to pressure discretization PreP1B).
- a better isotropy of elements than with **Trianguler_h** option.
- a better alignment of summits (this could have a benefit effect on calculation near walls since first elements in contact with it are all contained in the same constant thickness, and, by this way, a 2D cartesian grid based on summits can be engendered and used to realise statistical analysis in plan channel configuration for instance).

Principle :



<u>Remark</u> : **Trianguler_fin** (**Trianguler_h**, respectively) is equivalent in 3D to **Tetraedriser_homogene_fin** (**Tetraedriser_homogene_compact**).

To achieve a triangular mesh from a mesh comprising rectangles (4 triangles per rectangle), the **Trianguler_H** interpretor should be used in VEF discretization.

```
Trianguler_H   nom_domaine
```

To achieve a triangular mesh from a mesh comprising rectangles (2 triangles per rectangle), the **Trianguler** interpretor should be used in VEF discretization.

```
Trianguler   nom_domaine
```

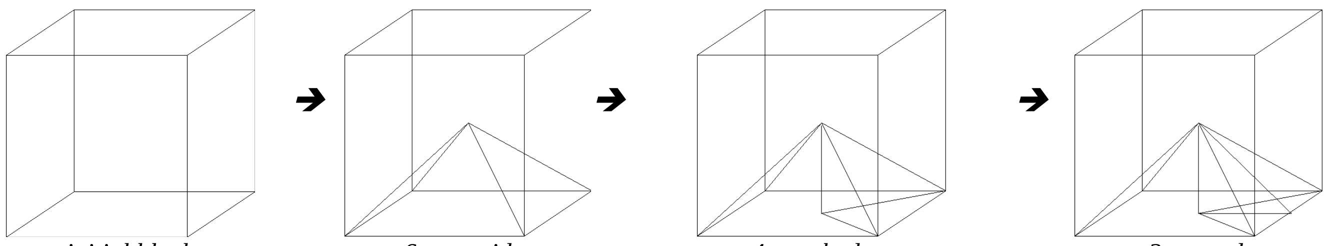
2.3.11 TETRAHEDRALISE

Tetraedriser_homogene_fin is the recommended option to tetrahedralise blocks.

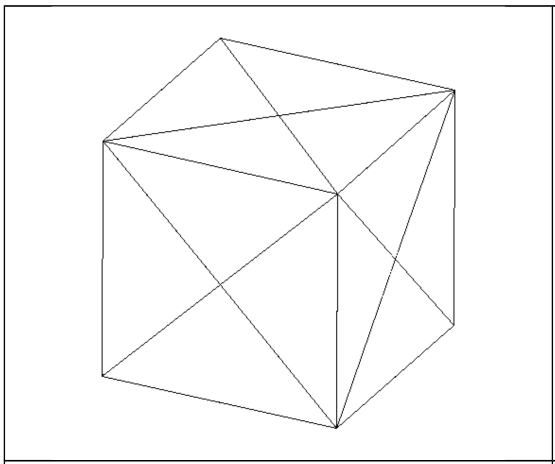
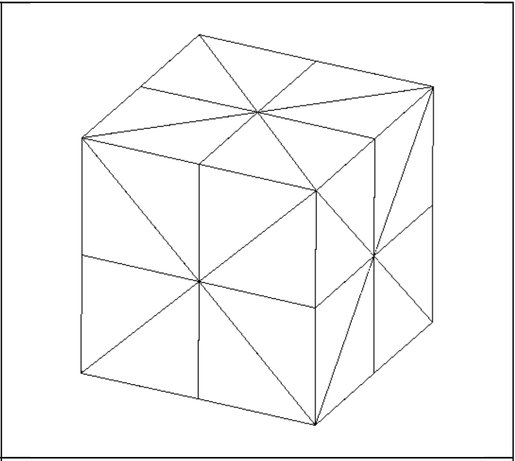
```
Tetraedriser_homogene_fin   nom_domaine
```

Tetraedriser_homogene_fin *:* As an extension (subdivision) of **Tetraedriser_homogene_compact** option, this last one cut each initial block in 48 tetrahedra (against 24, previously). This cutting ensures :
- a correct cutting in the corners (in respect to pressure discretization PreP1B).
- a better isotropy of elements than with **Tetraedriser_homogene_compact option**.
- a better alignment of summits (this could have a benefit effect on calculation near walls since first elements in contact with it are all contained in the same constant thickness and ii/ by the way, a 3D cartesian grid based on summits can be engendered and used to realise spectral analysis in HIT for instance).

<u>Principle :</u> initial block is divided in 6 pyramids, each of these being cut in 4 (**Tetraedriser_homogene_compact**), then 2 tetrahedra (**Tetraedriser_homogene_fin**).


initial block x *6 pyramids* x *4 tetrahedra* x *2 tetraedra*
= *24 tetrahedra* = *48 tetrahedra*
Tetraedriser_homogene_compact ***Tetraedriser_homogene_fin***

<u>Remark :</u> **Tetraedriser_homogene_fin** (**Tetraedriser_homogene_compact**, respectively) is equivalent in 2D to **Trianguler_fin** (**Trianguler_h**).

	
Tetraedriser_homogene_compact	***Tetraedriser_homogene_fin***

Tetraedriser_homogene_compact nom_domaine

Tetraedriser_homogene_compact : This keyword generates tetrahedral elements from cartesian or **non-cartesian** hexahedral elements. The process cut each hexahedral in 6 pyramids, each of them being cut then in 4 tetrahedral. So, in comparison with tetra_homogene, less elements (*24 instead of *40) with more homogeneous volumes are generated. Moreover, this process is done in a faster way.

Tetraedriser_homogene nom_domaine

Use the **Tetraedriser_homogene** interpretor in VEF discretization to mesh a block in tetrahedrals. Each block hexahedral is no longer divided into 5 tetrahedrals (keyword **Tetraedriser (Tetrahedralise)**), it is now broken down into 40 tetrahedrals. Thus a block defined with 11 nodes in each X, Y, Z direction will contain 10*10*10*40=40,000 tetrahedrals. This also allows problems in the mesh corners with the P1NC/P1iso/P1bulle or P1/P1 discretization items to be avoided.

Tetraedriser_par_prisme nom_domaine

Tetraedriser_par_prisme: This keyword generates 6 iso-volume tetrahedral element from primary hexahedral one (contrarily to the 5 elements ordinarily generated by tetraedriser). This element is suitable for calculation of gradients at the summit (coincident with the gravity centre of the jointed elements related with) and spectra (due to a better alignment of the points).

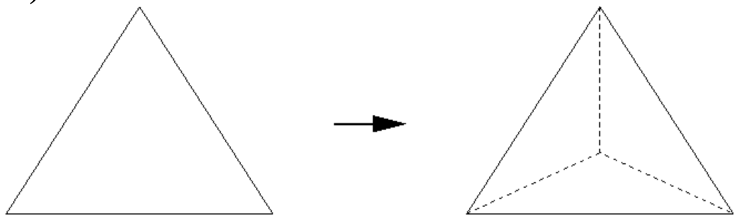
Tetraedriser nom_domaine

To achieve a tetrahedral mesh based on a mesh comprising blocks, the **Tetraedridal (Tetrahedralise)** interpretor is used in VEF discretization.

2.3.12REFINE

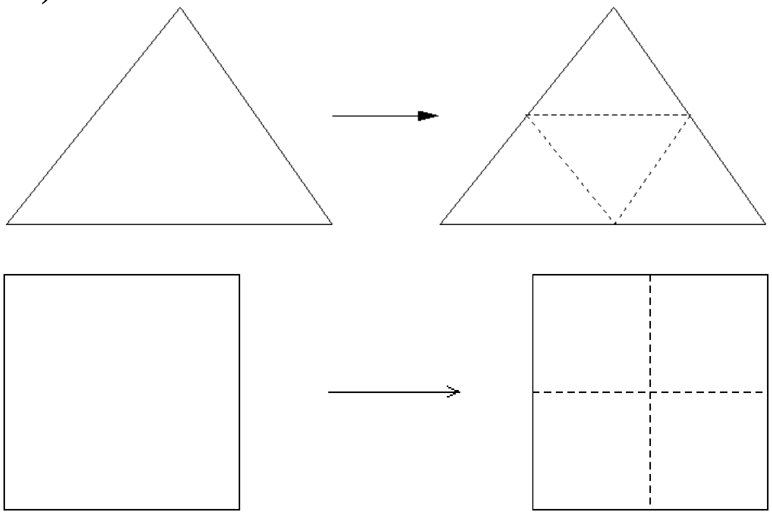
Raffiner_anisotrope domain_name

Keyword to allows to cut triangle or tetrahedra elements respectively in 3 or 4 new ones by defining a new summit located at the center of the element. Note that such a cut creates flat elements (anisotropic).



Raffiner_isotrope domain_name

Keyword to allows to cut triangles/quadrangles or tetrahedral/hexaedras elements respectively in 4 or 8 new ones by defining new summits located at the middle of edges (and center of faces and elements for quadrangles and hexaedra). Such a cut preserves the shape of original elements ("isotropic").



2.3.13EXTRUDE

Extruder { **domaine** domain_name **nb_tranches** n **direction** lx ly lz }

Extruder: Keyword to create a 3D tetrahedral/hexahedral mesh (a prism is cut in 14) from a 2D triangular/quadrangular mesh named domain_name with an extrude operation of n points in the direction (lx,ly,lz).

Extruder_en3 { **domaine** N domain_name1 domaine_name2 … domaine_nameN **nb_tranches** n **direction** lx ly lz [**nom_cl_devant** boundary_name_1] [**nom_cl_derriere** boundary_name_2] }

Extruder_en3: Keyword to create a 3D tetrahedral/hexahedral mesh (a prism is cut in 3) from a 2D triangular/quadrangular mesh named domain_name with an extrude operation of n points in the direction (lx,ly,lz). The names of the (by default, "devant" and "derriere") may be renamed by the keyword **nom_cl_devant** and **nom_cl_derriere**. If NULL is written for *boundary_name*, then no boundary condition is generated at this place.
Recommendation : to ensure conformity between meshes (in case of fluid/solid coupling) it is recommended to extrude all the domains at the same time.

Extruder_en20 { **domaine** domain_name **nb_tranches** n **direction** lx ly lz }

Extruder_en20: It does the same task as **Extruder** except a prism is cut in 20 instead of 3. Options **nom_cl_devant** and **nom_cl_derriere** are not available so the default name for the boudaries will be "devant" and "derriere". But you can change this name with the keyword **RegroupeBord**:

Extruder_en20 { **domaine** domaine_name … }
RegroupeBord domaine_name new_name_devant { devant }
RegroupeBord domaine_name new_name_derriere { derriere }

ExtrudeBord { **domaine_init** name_domain1 **direction** x y z
domaine_final name_domain2 **nom_bord** name_boundary
nb_tranches n [**hexa_old**] [**trois_tetra**] }

ExtrudeBord : Keyword dedicated to generate an extruded mesh from a boundary of a tetrahedral or an hexahedral mesh (Note that ExtrudeBord in VEF generates 3 or 14 tetrahedra from extruded prisms).

domaine_init name_domain1 : Initial domain with hexaedras or tetrahedras.
direction x y z : Directions for the extrusion.
domaine_final name_domain2 : Extruded domain.
nom_bord name_boundary : Name of the boundary of the initial domain where extrusion will be applied.
nb_tranches n : Number of elements in the extrusion direction.
hexa_old: Old algorithm for boundary extrusion from a hexahedral mesh
trois_tetra : Optional keyword to generates 3 tetraedras instead of 14 by default, from extruded prims

As **Extruder** keyword, il will create boundaries named "devant" and "derriere". If you want to create a periodic boundary named *perio*, use after the **RegroupeBord** keyword:

RegroupeBord name_domain2 *perio* { devant derriere }

This keyword can be used for example to create a periodic box extracted from a boundary of a tetrahedral or a hexaedral mesh. This periodic box may be used then to engender turbulent inlet flow condition for the main domain (see Champ_front_calc_recycl_fluct_pbperio)

In the example given below, DFLUI is the main (Hexaedra) domain and BOXPERIO is the periodic box engendered from face ENTCH of DFLUI. Extracted domain is according to the vector (-200,0.,0.) and contains 10 hexaedra in this direction:

Domaine DFLUI
Domaine BOXPERIO
Read _file DFLUI trio_DFLUI_geo.asc
ExtrudeBord { **domaine_init** DFLUI **direction** -200. 0. 0.
domaine_final BOXPERIO **nom_bord** ENTCH
nb_tranches 10 }
RegroupeBord BOXPERIO perio { devant derriere }
Ecrire_fichier BOXPERIO BOXPERIO.geom

ExtrudeParoi { **domaine** name_dom **nom_bord** name_boundary
[**epaisseur** n r1 r2 .... rn]
[**critere_absolu** 0|1]
[**projection_normale_bord** bool] }

ExtrudeParoi : Keyword dedicated in 3D (VEF) to create prismatic layer at wall. Each prism is cut in 3 tetraedra.

domaine name_dom : initial domain.
nom_bord name_boundary : Name of the (no slide) boundary for creation of prismatic layers.
epaisseur n r1 r2 .... rn : (relative or absolute) width for each layer.
critere_absolu 0|1 : relative (0, the default) or absolute (1) width for each layer.
projection_normale_bord bool : keyword to project layers on the same plane that contiguous boundaries. defaut values are : **epaisseur_relative** 1 0.5 **projection_normale_bord** 1

2.3.14CREATE_DOMAIN_FROM_SOUS_ZONE

Create_domain_from_sous_zone {
    domaine_final *domain1*
    par_sous_zone *name*
    domaine_init *domain2* }

These keyword fills the domain *domain1* with the subzone *name* from the domain *domain2*. It is very useful when meshing several mediums with **Gmsh**. Each medium will be defined as a subzone into **Gmsh**. A MED mesh file will be saved from **Gmsh** and read with **Lire_Med** keyword by the TRUST data file. And with this keyword, a domain will be created for each medium in the TRUST data file.

2.3.15EXTRACT_2D_FROM_3D

Extract_2D[**axi**]**_from_3D** *3D_domaine_name* *3D_boundary_name* *2D_domaine_name*

Extract_2D_from_3D :	Keyword to extract a 2D mesh by selecting a boundary of the 3D mesh. To generate a 2D axisymmetric mesh prefer **Extract_2Daxi_from_3D** keyword
3D_domaine_name :	Domain name of the 3D mesh
3D_boundary_name :	Boundary name. This boundary become the new 2D mesh and all the boundaries, in 3D, attached to the selected boundary, give their name to the news boundaries, in 2D.
2D_domaine_name :	Domain name of the new 2D mesh

2.3.16REORIENTER_TETRAEDRES

Reorienter_tetraedres name_domain

This keyword is mandatory for front-tracking computations with the VEF discretisation. For each tetrahedral element of the domain, it checks if it has a positive volume. If the volume (determinant of the three vectors) is negative, it swaps two nodes to reverse the orientation of this tetrahedron.

2.3.17CLEAN MESHES

```
NettoiePasNoeuds  name_domain
```

Keyword **NettoiePasNoeuds** does not delete useless nodes (nodes without elements) from a domain. Keyword **NettoieNoeuds** (suppressed useless nodes) is obsolete since the 1.4.6 version cause it is done by default now.

2.3.18ANALYSE_ANGLE

```
Analyse_angle  name_domain  nb_histo
```

Keyword **Analyse_angle** prints the histogram of the largest angle of each mesh elements of the domain named *name_domain*. nb_histo is the histogram number of bins. It is called by default during the domain discretization with nb_histo set to 18. Useful to check the number of elements with angles above 90°.

2.3.19VERIFIERCOIN

```
VerifierCoin  name_domain  { [Read_file file.decoupage_som] [expert_only] }
```

Keyword **VerifierCoin** subdivides inconsistent 2D/3D cells used with VEFPreP1B discretization. Must be used before the mesh is discretized.
The **Read_file** option can be used only if the *file.decoupage_som* was previously created by TRUST. This option, only in 2D, reverses the common face at two cells (at least one is inconsistent), through the nodes opposed. In 3D, the option has no effect.
The **expert_only** option deactivates, into the VEFPreP1B divergence operator, the test of inconsistent cells.

2.3.20PRINT MOMENTS ON BOUNDARIES

Calculer_moments nom_domaine **calcul**
Calculer_moments nom_domaine **centre_de_gravite** x y [z]

This keyword allows TRUST to calculate and print the torque (moment of force) exerted by the fluid on each boudanries in output files (.out) of the domain nom_domaine. You can either use the keyword **calcul** and the centre of gravity will be calculated or specify a specific centre with **centre_de_gravite** keyword.

2.3.21PRINT FLUX PER FACES

Imprimer_flux nom_domaine { *Bord1 Bord2 …*}

This keyword allows the flux per face at the boundaries named *Bord1*, *Bord2* of a domain to be printed. The flux are written into the .face files at a frequency defined by **dt_impr**, the evaluation printing frequency (refer to time scheme keywords). By default, flux are incorporated onto the edges before being displayed.

2.3.22PRINT FLUX PER BOUNDARY

Imprimer_flux_sum nom_domaine { *Bord1 Bord2 …*}

This keyword allows the sum of the flux per face at the boundaries named *Bord1, Bord2* of a domain defined by the user in the data set to be printed. The flux are written into the .out files at a frequency defined by **dt_impr**, the evaluation printing frequency (refer to time scheme keywords).

2.3.23GATHER BOUNDARIES

Regroupebord *domaine new_name_bord* { *Boundary1 Boundary2 Boundary3 …* }

Keyword to build one boundary *new_name_bord* with several boundaries of the domain named *domaine*.

2.3.24CONVERT BOUNDARIES

modif_bord_to_raccord *domain_name boundary_name*

Keyword to convert a boundary of *domain_name* domain of kind **Bord** to a boundary of kind **Raccord** (named *boundary_name*). It is useful when using meshes with boundaries of kind **Bord** defined and to run a coupled calculation.

2.3.25SUPPRESS INTERNEL BOUNDARIES

Remove_Invalid_Internal_Boundaries *domain_name*

Keyword to suppress an internal boundary of the *domain_name* domain. Indeed, some mesh tools may define internal boundaries (eg: for post processing task after the calculation) but TRUST does not support it yet.

2.3.26DEFINING SUB-AREAS

```
Sous_Zone nom_sous_zone
Associate nom_sous_zone nom_domaine
Read nom_sous_zone {
    bloc_lecture_sous_zone
  [ restriction nom_sous_zone2 ]
  [ union nom_sous_zone3 ]
}
```

Sous_Zone (Sub-area) is an object type describing a domain sub-set.

nom_sous_zone: Sous_Zone (Sub-area) type object identifier. This is the identifier that must be used to reference the created object type elsewhere in the data set.

A **Sous_Zone (Sub-area)** type object must be associated with a **Domaine** type object.
The **Read** interpretor is used to define the items comprising the sub-area.

Caution: The **Domain** type object *nom_domaine* must have been meshed (and triangulated or tetrahedralised in VEF) prior to carrying out the **Associate** *nom_sous_zone nom_domaine* instruction; this instruction must always be preceded by the read instruction.

Restriction : The elements of the sub-area *nom_sous_zone* must be included into the other sub-area named *nom_sous_zone2*. This keyword should be used first in the **Read** keyword.

Union : The elements of the sub-area *nom_sous_zone3* will be added to the sub-area *nom_sous_zone*. This keyword should be used last in the **Read** keyword.

bloc_lecture_sous_zone: one of the following blocks:

Fonction_sous_zone *function(x,y,z)* : Keyword to build a sub-area with the the elements included into the area defined by *function(x,y,z)*>0. See 2.4.5 how to write a function.

Rectangle Origine x0 y0 **Cotes** lx ly
Boite Origine x0 y0 z0 **Cotes** lx ly lz
The sub-area will include all the domain elements whose centre of gravity is within the Rectangle in 2D (resp. the Box in 3D).

Liste n n°1 n°i n° n
The sub-area will include n domain items, numbers No. 1 No. i No. n.

fichier *filename*
The sub-area is read into the file *filename*.

Intervalle n1 n2
The sub-area will include domain items whose number is between n1 and n2 (where n1 <= n2).

Polynomes { bloc_lecture_poly_1 **et** bloc_lecture_poly_i **et** bloc_lecture_poly_n }
Consider the surface area (or volume if referring to a 3D situation) obtained by surface intersection (or volume) defined by poly_1 => 0 ,. , poly_i => 0 , poly_n => 0. The sub-

area will include domain items whose centre of gravity is located within this surface area (or this volume).

Example:

```
Read  zone1
{
        Polynomes { 2 2 1 2 -0.33 1. et 2 2 1 2 0.66 -1. et 2 1 2 2 0. 1. et 2 1 2 2 1. -1. }
}
```

For:

x-0.33>0

0.66-x>0

y>0

1-y>0

The syntax to read a polynome is:

Polynome { dimension nx ny nx*ny c00 c01 c02 … c0(ny-1)(c10 c11 … c1(ny-1) … c(nx-1)0 …}

For $c00+c01y+c02y^2+\dots.c0(ny-1)y^{(ny-1)}+c10x+c11xy+\dots c1(ny-1)xy^{(ny-1)}+\dots+c(nx-1)x^{(nx-1)}\dots$

Couronne Origine x y [z] **ri** double **re** double

To create a "couronne" in 2D. **Origine**: the center of the circle. **ri,re**: the interior and exterior radius.

Tube Origine x y z **dir** axis **ri** double **re** double **hauteur** double

Keyword to create a tube in 3D where :

Origine: the center of the tube.

dir: direction of the main axis X, Y or Z

hauteur: the heigth of the tube

Tube_hexagonal entreplat double [**IN**|**OUT**]

Keyword to create a hexagonal tube centered at (0,0,0) and with axis along Z.

2.3.27DISCRETIZATION

A discretization object is created with the usual syntax.

```
type_discretization     dis
```

type_discretization: there are several available discretizations:
VDF : finite difference volume discretization
VEFPreP1B : finite element volume discretization ***(P1NC/P1-bubble element).*** Since the 1.5.5 version, several new discretizations are available thanks to the optional keyword **Read** :

```
VEFPreP1B     dis
Read  dis { [P0] [P1] [Pa]
            [Changement_de_base_P1Bulle 0|1]
            [Cl_pression_sommet_faible 0|1]
            [Modif_div_face_dirichlet 0|1]
}
```

P0 : Pressure nodes are added on element centres
P1 : Pressure nodes are added on vertices
Pa : Only available in 3D, pressure nodes are added on edges
Changement_de_base_P1Bulle value : This option may be used to have the P1NC/P0P1 formulation (value set to 0) or the P1NC/P1Bulle formulation (value set to 1, the default).
Cl_pression_sommet_faible value : This option is used to specify a strong formulation (value set to 0, the default) or a weak formulation (value set to 1) for an imposed pressure boundary condition. The first formulation converges quicker and is stable in general cases. The second formulation should be used if there are several outlet boundaries with Neumann condition (see *Ecoulement_Neumann* test case for example).
Modif_div_face_dirichlet value : This option (by default 0) is used to extend control volumes for the momentum equation.

By default, if the **Read** keyword is not used, the VEFPreP1B keyword is equivalent to the former VEFPreP1B formulation (v1.5.4 and sooner). P0P1 (if used with the strong formulation for imposed pressure boundary) is equivalent to VEFPreP1B but the convergence is slower. So:

VEFPreP1B dis

Is equivalent to :

VEFPreP1B dis
Read dis { **P0 P1 Changement_de_base_P1Bulle** 1 **Cl_pression_sommet_faible** 0 }

The discretization used before the 1.6.0 version by the old keyword **VEF** is now available with:

VEFPreP1B dis
Read dis { **P0** }

Cell shape	Rectangle	Rectangle	Triangle	Qaudrangle	Block	Block	Tetrahedron	Hexahedron
Coordinates	(x,y)	(r,z)	(x,y)	(x,y)	(x,y,z)	(r,θ,z)	(x,y,z)	(x,y,z)
VDF	OK	OK	■	■	OK	OK	■	■
VEFPreP1B	■	■	OK	■	■	■	OK	■

OK[(*)] means : OK on regular mesh only.

dis: the name of the created object**.**

A **Probleme (Problem)** object may be discretised according to a **VDF** or **VEF** discretization using the **Discretize** interpretor.

```
Discretize pb dis
```

The problem, *pb,* is discretised according to the *dis* discretization.

IMPORTANT: A number of objects must be already associated (a domain, time scheme, central object) prior to invoking the **Discretize** keyword. The physical properties of this central object must also have been read.

```
Discretiser_domaine dom
```

This keyword **Discretiser_domaine** is useful to discretize the domain dom (faces will be created) without defining a problem.

2.3.28ALLOCATE POROSITY

Two types of porosity, volume or surface, may be defined, the first corrects the surface of the passage offered to the fluid following a direction, the second effects the mesh volume in question.

surface porosity = (fluid surface)/(total mesh surface)
volume porosity = (fluid volume)/(total mesh volume)

The porosity can also be defined as a field. The porosity is given at each element and the porosity at each face, Psi(face), is calculated by the average of the porosities of the two neighbour elements Psi(elem1), Psi(elem2) : Psi(face)=2/(1/Psi(elem1)+1/Psi(elem2)).

Porosites_champ *nom_pb* field *field_definition*

nom_pb: name of the problem
domaine: name of the domain
field: field used to define the porosity field (**champ_fonc_xyz**, **champ_uniforme**, **champ_uniforme_morceaux**,…)

The volume porosity and surface porosity that are uniform in every direction in space on a sub-area may be defined using the **Porosites** keyword:

Porosites nom_pb nom_sous_zone **{ [volumique** val_poro_vol **]**
[surfacique 2|3 val_ poro_surf_X val_ poro_surf_ Y [val_poro_surf_Z] **] }**

nom_sous_zone: name of the sub-area to which porosity are allocated.
nom_pb: name of the problem to which the sub-area is attached.
val_poro_vol: volume porosity value.
val_poro_surf_X: surface porosity value in the X direction.
val_poro_surf_Y: surface porosity value in the Y direction.
val_poro_surf_Z: surface porosity value in the Z direction.

Observations:

- Surface porosity values must be given in every direction in space (set this value to 1 if there is no porosity).
- Prior to defining porosity, the problem must have been discretized.
- Can't be used in VEF discretization, use **Porosites_champ** instead.

2.3.29PRECISIONGEOM

Keyword to change the way floating-point number comparison is done. By default, two numbers are the same if their absolute difference is less than 1e-10. The keyword is useful to change this value:

PrecisionGeom new_value

Moreover, nodes coordinates will be written in .geom files with this same precision.

2.3.30DILATE

Keyword to multiply the whole coordinates of the geometry.

Dilate *domain_name value_of_dilatation_coefficient*

Example:
Read_file dom trio_DOM_geo_33.asc
Dilate dom 0.001

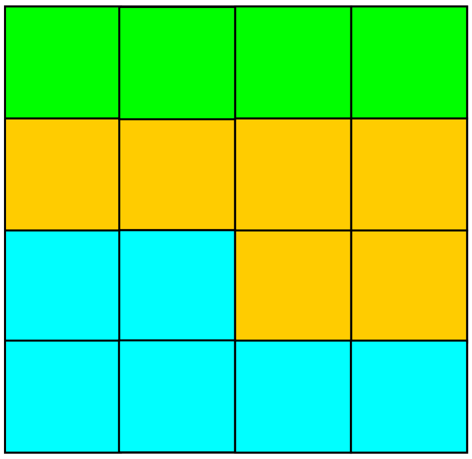
2.3.31DECOUPEBORD_POUR_RAYONNEMENT

Keyword to subdivide the external boundary of a domain in several parts (may be useful for better accuracy when using radiation model in transparent medium).

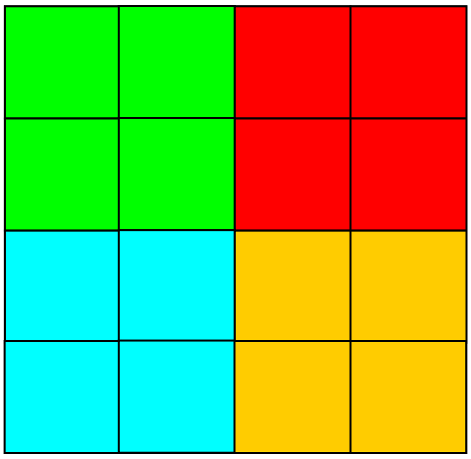
```
DecoupeBord_pour_rayonnement {
    domaine fine_domain_name
    bords_a_decouper N boundary_name1 boundary_name2 …boundary_nameN
    [ domaine_grossier coarse_domain_name
                    | nb_parts_naif  N n1 n2 … nN
              | nb_parts_geom N n1*m1 n2*m2 … nN*mM
                    | condition_geometrique N formulae1 formulae2 … formulaeN ]

    ]
    [nom_fichier_sortie filename [ binaire ]]
}
```

bords_a_decouper is the keyword to specify the boundaries of the *fine_domain_name* domain to be splitted. These boundaries will be cut according the coarse mesh defined by:
-either the keyword **domaine_grossier** (each boundary face of the coarse mesh *coarse_domain_name* will be used to group boundary faces of the fine mesh to define a new boundary). Notice that the *coarse_domain_name* domain should have the same boundaries name of the *fine_domain_name* domain.
-either by the keyword **nb_parts_naif** (each Ith boundary is splitted into nI parts)
Example: **nb_parts_naif** 1 3

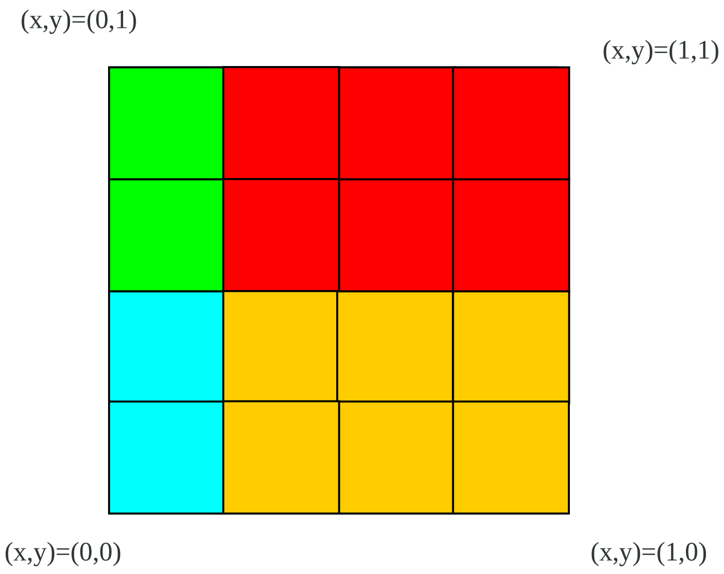


-either by the keyword **nb_parts_geom** (each Ith boundary is splitted into nI*mI parts). This keyword is only available for the VDF discretization.
Example: **nb_parts_geom** 2 2 2



-either by geometric conditions given by formulaes for each boundary with the keyword **condition_geometrique**

Example: **condition_geometrique** 1 (x>0.25)+2*(y>0.5)



A mesh file (ASCII format, except if **binaire** option is specified) named by default *fine_domain_name.newgeom* (or specified by the **nom_fichier_sortie** keyword) will be created and will contain the *fine_domain_name* domain with the splitted boundaries named *boundary_name%I* (where I is between from 0 and n-1). Furthermore, several files named *fine_domain_name.boundary_name%I* and *fine_domain_name.boundary_name_xv* will be created, containing the definition of the subdived boundaries. *fine_domain_name.newgeom* will be used to calculate view factors with **geom2ansys** script whereas only the *fine_domain_name.boundary_name_xv* files will be necessary for the radiation calculation. The file *fine_domain_name.boundary_list* will contain the list of the boundaries *boundary_name%I.*

2.3.32 SUPPRIME_BORD

Supprime_bord *domain_name { Boundary_name1 Boundary_name2 … }*

Keyword to remove boundaries (named *Boundary_name1 Boundary_name2* …) of the domain named *domain_name*.

2.3.33 ORIENTEFACESBORD

OrienteFacesBord *domain_name*

Keyword to modify the order of the boundary verteces included in a domain, such that the surface normals are outer pointing.

2.3.34 TRANSFORMER

Keyword to transform the coordinates of the geometry :

Transformer *domain_name function_for_x function_for_y [function_for_z]*

Example to rotate your mesh by a 90° rotation and to scale the z coordinates by a factor 2:
Read_file dom trio_DOM_geo_33.asc
Transformer dom -y -x 2*z

2.3.35ROTATION

Keyword to rotate the geometry of an arbitrary angle around an axis aligned with Ox, Oy or Oz axis :

Rotation *domain_name axis coord0 coord1 angle*

domain_name : name of the domain to wich the transformation is applied
axis : X, Y or Z to indicate the direction of the rotation axis
corrd0, coord1: coordinates of the center of rotation in the plane orthogonal to the rotation axis. These coordinates must be specified in the direct triad sense.
angle : angle of rotation (in degrees)

2.3.36SOLVE

Solve name_problem

Interpretor to start calculation with TRUST. The problem name_problem is solved.

2.3.37CONVERSION

Lata_To_Other format *file1 file2*

Interpretor to convert results file named *file1* written with LATA format to a file named *file2* with MED or LML format.
format: MED or LML keyword.
Warning: Fields located to faces are not supported yet.

Lata_To_MED [format] *file1 file2*

Interpretor to convert results file named *file1* to a MED file named *file2*. A data file is also created to reconvert the MED file *file2* to another format specified by the optional given format.
Warning: Fields located to faces are not supported yet.

2.3.38EXECUTE_PARALLEL

Execute_parallel { **liste_cas** N *datafile1 … datafileN* [**nb_procs** N nb1 … nbN] }

Execute_parallel: This keyword allows to run several computations in parallel on cpus allocated to TRUST. The set of cpus is split in N subsets and each subset will read and execute a different data file. *datafileX* the name of a TRUST data file without the .*d*ata extension. **nb_procs** is the number of cpus needed to run each data file. If not given, TRUST assumes that computations are sequential. Error messages usualy written to stderr and stdout are redirected to .*log* files (journaling must be activated).

2.3.39MOYENNE_VOLUMIQUE

Moyenne_volumique { **Nom_pb** *source_problem_name*
  Noms_champs *N source_field1 source_field2 ... source_fieldN*
  Nom_domaine *destination_domain_name*
  Fonction_filtre {
    type filter_type
    demie-largeur l
    [**omega** w]
    [**expression** string]
  }
  [**Localisation ELEM** | **SOM**]
  Nom_fichier_post *destination_filename*]
  [**Format_post** lata | lml]
}

This keyword should be used after **Solve** keyword. It computes the convolution product of one or more fields with a given filtering function.

Nom_pb: name of the problem where the source fields will be searched.

Noms_champs: name of the source fields (these fields must be accessible from the **post_processing**)

Nom_domaine: name of the destination domain (for example, it can be a coarser mesh, but for optimal performance in parallel, the domain should be split with the same algorithm as the computation mesh, eg, same **tranche** parameters for example)

Fonction_filter is the keyword to specify the given filter:

type filter_type : This parameter specifies the filtering function. Valid filter_type are :
Boite is a box filter, f(x,y,z)=(abs(x)<l)*(abs(y) <l)*(abs(z) <l) / (8*l^3)
Chapeau is a hat filter (product of hat filters in each direction) centered on the origin, the half-width of the filter being *l* and its integral being 1.
Quadra is a 2nd order filter
Gaussienne is a normalized gaussian filter of standard deviation sigma in each direction (all field elements outside a cubic box defined by clipping_half_width are ignored, hence, taking clipping_half_width=2.5*sigma yields an integral of 0.99 for a uniform unity field).
Parser allows a user defined function of the x,y,z variables. All elements outside a cubic box defined by clipping_half_width are ignored. The parser is much slower than the equivalent c++ coded function...

demie-largeur l : This parameter specifies the half width of the filter
[**omega** w] : This parameter must be given for the gaussienne filter. It defines the standard deviation of the gaussian filter.
[**expression** string] : This parameter must be given for the parser filter type. This expression will be interpreted by the math parser with the predefined variables "x", "y" and "z".

Localisation ELEM | SOM indicates where the convolution product should be computed: either on the elements or on the nodes of the destination domain.

Nom_fichier_post: indicates the filename where the result is written

Format_post: gives the fileformat for the result (by default : lata)

Recommandations and details:

- the filter generates also a field called *porosite* which is the result of filtering a unity field.
- the filter handles any kind of source field by evaluating the field at the center of the elements (see valeur_aux_elems() function).

- filters with a large halft-width are very slow (expect quite long computation time if the filter width is more than 20 mesh cells).
- when filtering cell centered data on a regular grid, the width of the filter will be an odd number of cells width **localisation ELEM** and an even number with **localisation SOM**.
- The filter computes for each given field the following expression: $\hat{f}(x) = \sum_{i \in E} f(y_i) \cdot g(y_i - x) \cdot V(i)$ where E is the set of elements for which the center is inside the clipping box, y_i is the coordinate of the element center (the source field is always interpolated at the center of the elements whatever its native localization) and V(i) is the volume of the element. The result is computed for each coordinate x of the destination domain (elements or nodes depending on **localisation**).
- For the **boite** filter and with **localisation elem**, the filter width should not be an exact multiple of the size of the source mesh cells (otherwise the filter might produce unpredictable results for some elements).

2.3.40EXTRAIRE_PLAN

```
Extraire_plan {
        Domaine domain_name
        Probleme pb_name
        Epaisseur float
        Origine 2|3 ox oy [oz]
        Point1 2|3 x1 y1 [z1]
        Point2 2|3 x2 y2 [z2]
        [ Point3 2|3 x3 y3 [z3] | Triangle ]
        [ via_extraire_surface
          [ inverse_condition_element ]
          [ avec_certains_bords_pour_extraire_surface n boundary1 … boundary n ]
        ]
}
```

This keyword extract a plan mesh named *domain_name* (this domain should have be declared before) from the mesh of the *pb_name* problem. The plan can be either a triangle (defined by the keywords **Origine**, **Point1**, **Point2** and **Triangle**), either a regular quadrangle (with keywords **Origine**, **Point1** and **Point2**), or either a generalized quadrangle (with keywords **Origine**, **Point1**, **Point2**, **Point3**). The keyword **Epaisseur** specifies the thickness of volume around the plan which contains the faces of the extracted mesh. The keyword **via_extraire_surface** will create a plan and use **Extraire_surface** algorithm. **Inverse_condition_element** keyword then will be used in the case where the plan is a boundary not well oriented, and **avec_certains_bords_pour_extraire_surface** is the option related to the **Extraire_surface** option named **avec_certains_bords**.

2.3.41EXTRAIRE_SURFACE

```
Extraire_surface  {
        Domaine domain_name
        Probleme pb_name
        Condition_elements f(x,y,z)
        Condition_faces g(x,y,z)
        [ avec_les_bords ]
        [ avec_certains_bords N name1 name2 … nameN ]
}
```

This keyword extract a surface mesh named *domain_name* (this domain should have be declared before) from the mesh of the *pb_name* problem. The surface mesh is defined by one or two conditions. The first condition is about elements with **Condition_elements.** For example:

Condition_elements x*x+y*y+z*z<1

Will define a surface mesh with external faces of the mesh elements inside the sphere of radius 1 located at (0,0,0). The second conditions **Condition_faces** is useful to give a restriction.

By default, the faces from the boundaries are not added to the surface mesh excepted if option **avec_les_bords** is given (all the boundaries are added), or if the option **avec_certains_bords** is used to add only some boundaries.

2.3.42EXTRAIRE_DOMAINE

```
Extraire_domaine  {
        Domaine domain_name
        Probleme pb_name
        Condition_elements f(x,y,z)
}
```

Extraire_domaine : Keyword to create a new new domain built with the domain elements of the *pb_name* problem verifying the two conditions given by **Condition_elements**. The problem *pb_name* should have been discretized.

2.3.43 INTEGRER_CHAMP_MED

```
Integrer_champ_MED  {
        Champ_med  MED_field
        Methode  [ integrale_en_z|debit_total ]
        [zmin float zmax float nb_tranche integer Fichier_sortie output_filename]
}
```

This keyword is used to calculate a flow rate from a velocity MED field read before. The method is either **debit_total** to calculate the flow rate on the whole surface, either **integrale_en_z** to calculate flow rates between z=**zmin** and z=**zmax** on **nb_tranche** surfaces. The output file indicates first the flow rate for the whole surface and then lists for each tranche : the height z, the surface average value, the surface area and the flow rate. For the debit_total method case, only one tranche is considered.

```
# z     Sum(u.dS)/Sum(dS)   Sum(dS)  Sum(u.dS)
…….
```

2.3.44 SYSTEM

```
System "unix_commands"
```

Interpretor to run Unix commands from the data file. Example:
System "echo The End | mail triou@cea.fr"

2.3.45 REDRESSER_HEXAEDRES_VDF

```
Redresser_hexaedres_VDF  domain_name
```

Keyword to convert a domain (named domain_name) with quadrilaterals/VEF hexaedras which looks like rectangles/VDF hexaedras into a domain with real rectangles/VDF hexaedras.

2.4 OBJECT FIELD DEFINITION

As they are widely used in the data set, descriptions of the various field types recognised by TRUST is given hereunder.

There are three field families:

- unknown fields; these are not mentioned in the data set
- physical parameter fields and initial condition fields
- boundary fields which are used in limitation conditions or in couplings

Two types of instructions using these fields are found in the data set:

Field creation:

In accordance with object creation syntax, a field may be created as follows:

field_type *identificateur_champ*

For example: **Champ_Uniforme** gravity (instruction No. 1)

Entering values in existing fields:

example No. 1: Values are entered for the **Champ_Uniforme** (uniform gravity) type gravity object created by instruction No. 1:

Read gravite 2 0. -9.81

example No. 2: Imagine that the **Fluide_Incompressible (Incompressible_Fluid)** which includes a **Champ_Don** object type to represent its dynamic viscosity. The read syntax will be as follows:

mu *field_type* *bloc_lecture_champ*

The **mu** identifier object already exists (it is automatically created when the **Fluide_Incompressible (Incompressible_Fluid)** type object that includes it was created). This instruction is used only to enter a value for it.

<u>example No. 3</u>: A fluid inlet with imposed speed type boundary condition is defined as follows:

Gauche **Frontiere_ouverte_vitesse_imposee** *boundary_field_type bloc_lecture_champ*.

The boundary field is specified without selecting an identifier. A value is entered in the **Champ_front (Boundary_field)** type object carried by the **Cond_lim (limitation_condition)** type object.

When you write a data set, you are not free to select the syntax, i.e., you are bound by one of the previous cases. You must select a **Champ_Don** or **Champ_front** type and correctly fill in its read block. The list of fields that may be used and the associated read blocks will be given here.

2.4.1STATIONARY FIELDS

• Champ_Uniforme (Uniform_field): field that is constant in space and stationary.

```
Champ_Uniforme  nb_comp  vrel_1...[vrel_i]
```

nb_comp: number of field components.
vrel_1...[vrel_i]: values of field components.

• Field_uniform_keps_from_ud : field which allows to impose on a domain K and EPS values derived from U velocity and D hydraulic diameter

```
Field_uniform_keps_from_ud   { U vrel  D diam }
```

vrel: this is the value of velocity specified in boundary condition.
diam: this is the value of hydraulic diameter specified in boundary condition.

• Champ_Uniforme_Morceaux: field which is partly constant in space and stationary.

```
Champ_Uniforme_Morceaux  nom_domaine  nb_comp
{ Defaut  val_def   sous_zone_1  val_1 ...  sous_zone_i  val_i }
```

nom_domaine: name of the domain to which the sub-areas belong.

nb_comp: number of field components.

By default, the value *val_def* is assigned to the field. It takes the *sous_zone_i* identifier **Sous_Zone (sub_area)** type object value, *val_i*. **Sous_Zone (sub_area)** type objects must have been previously defined if the operator wishes to use a **Champ_Uniforme_Morceaux (partly_uniform_field)** type object.

• Valeur_totale_sur_volume: Similar as **Champ_Uniforme_Morceaux** with the same syntax. Used for source terms when we want to specify a source term with a value given for the volume (eg: heat in Watts) and not a value per volume unit (eg: heat in Watts/m3).

```
Valeur_totale_sur_volume  nom_domaine  nb_comp
{ Defaut  val_def   sous_zone_1  val_1 ...  sous_zone_i  val_i }
```

• Champ_Don_lu: This field is used to read a data field (values located at the center of the cells) in a file.

```
Champ_Don_lu nom_domain  nb_comp filename
```

name_domain: name of the domain
nb_comp: number of field components
filename: name of the file. This file has the following format:

nb_val_lues	->Number of values readen in th file
Xi Yi Zi	-> Coordinates readen in the file
Ui Vi Wi	-> Value of the field

Example:
Initial_Conditions { vitesse Champ_don_lu dom 2 ftn10 }

- Champ_som_lu_VDF
- Champ_som_lu_VEF

Keywords to read in a file values located at the nodes of a mesh in VDF or VEF discretization :

```
Champ_som_lu_VDF name_domain nb_comp tolerance filename
Champ_som_lu_VEF name_domain nb_comp tolerance filename
```

name_domain: the domain name
nb_comp: value of the dimension of the field
tolerance: value of the tolerance to check the coordinates of the nodes
filename: name of the file. This file has the following format:

```
Xi Yi Zi                  -> Coordinates of the node
Ui Vi Wi                   -> Value of the field on this node
Xi+1 Yi+1 Zi+1              -> Next point
Ui+1 Vi+1 Zi+1              -> Next value
....
```

• Champ_Fonc_Reprise: This field is used to read a data field in a save file (.xyz or .sauv) at a specified time. It is very useful, for example, to run a thermohydraulic calculation with velocity initial condition read into a save file from a previous hydraulic calculation.

```
Champ_Fonc_Reprise  [xyz|formatte|binaire] filename
    problem_name field_name  [fonction n f1(val) f2(val) … fn(val)]  time | last_time
```

[xyz|formatte|binaire] : Optional keyword to specify the format of the filename (by default xyz format). If xyz format is activated, the .xyz file from the previous calculation will be given for *filename*, and if formatte or binaire is choosen, the .sauv file of the previous calculation will be specified for *filename*. In the case of a parallel calculation, if the mesh partition does not changed between the previous calculation and the next one, the binaire format should be preferred, because is faster than the xyz format.
filename: name of the save file
problem_name: name of the problem
field_name: name of the problem unknown. It may also be the temporal average of a problem unknown (like moyenne_vitesse, moyenne_temperature,…)
fonction…: Optional keyword to apply a function on the field being read in the save file (e.g. to read a temperature field in Celsius units and convert it for the calculation on Kelvin units, you will use: **fonction** 1 273.+**val**)

time: time of the saved field in the save file. If you give the keyword **last_time** instead, the last time saved in the save file will be used.

Example:
Initial_Conditions { vitesse **Champ_Fonc_Reprise** pipe.xyz pb vitesse 5.101 }

• Champ_Fonc_Med: This field is used to read a data field in a MED-format file .med at a specified time. It is very useful, for example, to restart a calculation with a new or refined geometry. The field post-processed on the new geometry at med format is used as initial condition for restarting.

```
Champ_Fonc_Med  [ last_time ] filename.med  domain_name field_name location time
```

filename: name of the .med file
domain_name: name of the domain
field_name: name of the problem unknown
location: to indicate where the field has been post-processed (**elem** or **som**)
time: time of the field in the .med file
last_time: Optional keyword to use the last time of the MED file instead of the specified time.

Example:
Initial_Conditions { temperature **Champ_Fonc_Med** pipe.med dom temperature elem 0.25 }

• Champ_init_canal_sinal : For a parabolic profile on U velocity with an unpredictable disturbance on V and W and a sinusoidal disturbance on V velocity :

```
Champ_init_canal_sinal nb_comp {  Ucent  value   h value
        ampli_bruit value   [ ampli_sin value ]  omega value dir_flow 0
              dir_wall value
              min_dir_flow value
              min_dir_wall value
        }
```

nb_comp: Number of field components.
Ucent value : Velocity value at the center of the channel.
h value : Half hength of the channel.

ampli_bruit value : Amplitude for the disturbance.
ampli_sin value : Amplitude for the sinusoidal disturbance (optional, by default equals to ucent/10).
omega value : Value of pulsation for the of the sinusoidal disturbance.

In 2D:
u=**Ucent***y(2h-y)/h/h
v=**ampli_bruit***rand+ampli_sin*sin(omega*x)
rand: unpredictable value between -1 and 1.

in 3D:
u=**Ucent***y(2h-y)/h/h
v=**ampli_bruit***rand1+ampli_sin*sin(omega*x)
w=**ampli_bruit***rand2
rand1 and rand2: unpredictables values between -1 and 1.

min_dir_wall : Keyword to define the value of the minimum coordinate in the wall direction for the initialization of the flow in a channel. Default value for dir_flow is 0.

min_dir_flow : Keyword to define the walue of the minimum coordinate in the flow direction for the initialization of the flow in a channel. Default value for dir_flow is 0.

dir_wall : Keyword to define the wall direction for the initialization of the flow in a channel.
-if dir_wall = 0, the normal to the wall is in direction
-if dir_wall = 1, the normal to the wall is in Y direction
-if dir_wall = 2, the normal to the wall is in Z direction
Default value for dir_flow is 1

dir_flow: Keyword to define the flow direction for the initialization of the flow in a channel.
-if dir_flow = 0, the flow direction is X
-if dir_flow = 1, the flow direction is Y
-if dir_flow = 2, the flow direction is Z
Default value for dir_flow is 0

2.4.2UNSTATIONNARY FIELDS

• Champ_Tabule_Temps: this type of field is constant in space and tabulated as a function of time.

```
Champ_Tabule_Temps  nb_comp { nval   tps_1....tps_nval ... vrel_1      vrel_nval }
```

nb_comp: this refers to the number of field components.
Values are entered into a table based on *nval* couples (*vrel_i*, *tps_i*). The value of the field at any time is calculated by linear interpolation from this table.

Champ_Uniforme_Morceaux_Tabule_Temps: this type of field is constant in space on one or several sub_zones and tabulated as a function of time.

```
Champ_Uniforme_Morceaux_Tabule_Temps  domaine_name nb_comp
     {
        Defaut  float(1)  … float(nb_comp)
        Sub_zone_name1 { nval   tps_1....tps_nval ... vrel_1      vrel_nval  }
        Sub_zone_name2 { nval   tps_1....tps_nval ... vrel_1      vrel_nval  }
        ….
     }
```

domaine_name: Name of the domain.
nb_comp: this refers to the number of field components.
Defaut float(1) … float(nb_comp) : Constant values for the field on elements not covered by a subzone.
Sub_zone_nameI: Name of the Ith subzone.
Values are entered into a table based on *nval* couples (*vrel_i*, *tps_i*). The value of the field at any time is calculated by linear interpolation from this table.

• Champ_Fonc_t: this type of field is constant in space and is a function of time.

Champ_Fonc_t nb_comp $f_1(t)$ … $f_{nb\_comp}(t)$

nb_comp: this refers to the number of field components. $f_i(t)$ is a time dependant function.

- Champ_Fonc_Fonction: this refers to a field that is a function of another field.

```
Champ_Fonc_Fonction  nb_comp  field   expression
```

nb_comp: this refers to the number of field components.
field: name of the field (for example: temperature)
expression: keyword to use a analytical expression like 10.*EXP(-0.1*val) where val be the keyword for the field.

- Champ_Fonc_Fonction_txyz: this refers to a field that is a function of another field and time and/or space coordinates

```
Champ_Fonc_Fonction_txyz  nb_comp  field   expression
```

nb_comp: this refers to the number of field components.
field: name of the field (for example: temperature)
expression: keyword to use a analytical expression like 10.*EXP(-0.1*val)*x*y*z+t where val be the keyword for the field.

- Champ_Fonc_Tabule: this refers to a field that is tabulated as a function of another field.

```
Champ_Fonc_Tabule nb_comp field
[ { nval  teta_1 ....teta_nval..vrel_1........vrel_nval } ]
```

nb_comp: this refers to the number of field components. Values are entered for a table based on *nval* couples (*vrel_i*, *teta_i*). The value of the tabulated field is calculated based on a given field (temperature, concentration,…) by linear interpolation from this table.

• Champ_fonc_xyz: This keyword represents a new field. It's now possible to write directly in the data file, a string representation of a function f(x,y,z).

Champ_fonc_xyz domain_name nb_comp f_1(x,y,z) … f_nbcomp(x,y,z)

f_i(x,y,z) is a string representation of a mathematical expression (see 2.4.5).

• Champ_fonc_txyz **:** This keyword defines a new type of field. It makes it possible the definition of a field that depends on the time and the space.

Champ_Fonc_txyz domain_name Nb_comp f_1(t,x,y,z) ... f_Nb_comp(t,x,y,z)

f_i(x,y,z) is a string representation of a mathematical expression (see 2.4.5).

2.4.3STATIONARY BOUNDARY FIELDS

• Champ_front_uniform: field which is constant in space and stationary

Champ_front_uniforme nb_comp vrel_1....[vrel_i]

nb_comp: this refers to the number of field components.
vrel_1...[vrel_i]: these are the values of field components.

Remark : for coupling, you can use **ch_front_input_uniforme** which is a champ_front_uniforme, which use an external value. It must be used with “Probleme.setInputField”.

• Boundary_field_uniform_keps_from_ud: field which allows to impose on a boundary K and EPS values derived from U velocity and D hydraulic diameter

```
Boundary_field_uniform_keps_from_ud   { U vrel  D diam }
```

vrel: this is the value of velocity specified in boundary condition.
diam: this is the value of hydraulic diameter specified in boundary condition.

- Champ_front_fonc_XYZ: boundary field which is not constant in space

```
Champ_front_fonc_XYZ  nb_comp f_1(x,y,z) … f_nbcomp(x,y,z)
```

f_i(x,y,z) is string representation of mathematical expression (see 2.4.5). For instance, to set the velocity :

Gauche frontiere_ouverte_vitesse_imposee **Champ_front_fonc_xyz** 2 5*y*(1-y) 0.

An example with a test :

Gauche frontiere_ouverte_vitesse_imposee **Champ_front_fonc_xyz** 2 (y>1.)*5*y*(1-y) 0.

This example fixes the velocity Vx with the function 5*y*(1-y) only if y>1.

- Champ_front_fonction: boundary field that is function of another field

```
Champ_front_fonction  nb_comp  field  expression
```

nb_comp: this refers to the number of field components.
field: name of the field (for example: temperature)
expression: keyword to use a analytical expression like 10.*EXP(-0.1*val) where val be the keyword for the field.

- Champ_front_lu: boundary field read in a file

```
Champ_front_lu  domain_name nb_comp filename
```

Champ_Front_lu : boundary field which is given from data issued from a read file. The format of this file has to be the same that the one generated by **Ecrire_fichier_xyz_valeur** (see 2.6.1).
nom_domaine : name of the domain
nb_comp : number of components
filename : path for the read file

Example for K and epsilon quantities to be defined for inlet condition in a boundary named "entree":
entree **frontiere_ouverte_K_Eps_impose Champ_Front_lu** dom *2pb_K_EPS_PERIO_1006.306198.dat*

2.4.4UNSTATIONNARY BOUNDARY FIELDS

- Champ_front_tabule: a constant field on the boundary, tabulated as a function of time

```
Champ_front_tabule  nb_comp {n
t1 t2 t3 ....tn
u1 [v1 w1 …]  u2 [v2 w2 …] u3 [v3 w3 …] … un [vn wn …]  }
```

nb_comp: refers to the number of field components.
Values are entered into a table based on *n* couples (ti, ui) if nb_comp value is 1. The value of a field at a given time is calculated by linear interpolation from this table.

- Champ_front_fonc_TXYZ: boundary field which is not constant in space and in time

```
Champ_front_fonc_TXYZ  nb_comp f_1(x,y,z,t) … f_nbcomp(x,y,z,t)
```

f_i(x,y,z,t) is string representation of mathematical expression (see 2.4.5). For instance, to set the velocity :

Gauche frontiere_ouverte_vitesse_imposee **Champ_front_fonc_txyz** 2 y*sin(t) 0

• Champ_front_bruite: a field which is variable in time and space in a random manner.

Champ_front_bruite nb_comp { [N val L val] **Moyenne** m_1.....[m_i] **Amplitude** A_1.....[A_ i]}

nb_comp: number of field components

Random noise:

If N and L are not defined, the ith component of the field varies randomly around an average value *m_i* with a maximum amplitude *A_i*.

White noise:

If N and L are defined, these two additional parameters correspond to L, the domain length and N, the number of nodes in the domain. Noise frequency will be between 2*Pi/L and 2*Pi*N/(4*L):

For example, formula for speed: u=U0(t) v=U1(t)
Uj(t)=Mj+2*Aj*bruit_blanc where bruit_blanc (white_noise) is the formula given in the mettre_a_jour (update) method of the Champ_front_bruite (noise_boundary_field) (Refer to the Ch_fr_bruite.cpp file)

• Champ_front_debit : this field is used to define a flow rate field instead of a velocity field for a Dirichlet boundary condition on Navier Stokes equation.

Champ_front_debit *type_field*

typeield : Kind of field (champ_front_uniforme, …) to define the flow rate.

• Champ_front_pression_from_u : this field is used to define a pressure field depending of a velocity field.

Champ_front_pression_from_u *f(u)*

f(u): value depending of a velocity (like *“2*u_moy^2”).*

• Champ_front_tangentiel_VEF : this field is used to define the tangential speed vector field standard at the boundary in VEF discretization.

Champ_front_tangentiel_VEF vitesse_tangentielle valeur

valeur: vector field standard [m/s]

• Boundary_field_inward : this field is used to define the normal vector field standard at the boundary in VDF or VEF discretization.

Boundary_field_inward { normal_value *f(t)* **}**

f(t): normal vector value (positive value for a vector oriented outside to inside) which can depend of the time 't'.

• Champ_front_ALE**:** Keyword to define a boundary condition on a moving boundary of a mesh.

Champ_front_ALE nb_comp val_1…[val_i]

Example:

Boundary_name frontiere_ouverte_vitesse_imposee **Champ_front_ALE** 2
20*0.3*SIN(6.28*y)*COS(20*t) 0. }

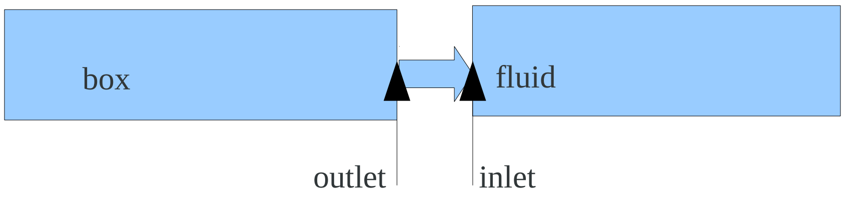
• Champ_front_calc **:** This keyword is used on a boundary to get a field from another boundary. The local and remote boundaries should have the same mesh. If not, the **Champ_front_recyclage** keyword could be used instead.

Champ_front_calc FieldProblemName FieldBoundaryName FieldName

FieldProblemName: name of the problem owning the desired field

FieldBoundaryName: boundary name of the FieldProblemName problem where the desired field values will be copied from
FieldName: name of the desired field

Example :
Read fluid
{
	...
	inlet **frontiere_ouverte_temperature_imposee**
	champ_front_calc box outlet **temperature**
	...
}



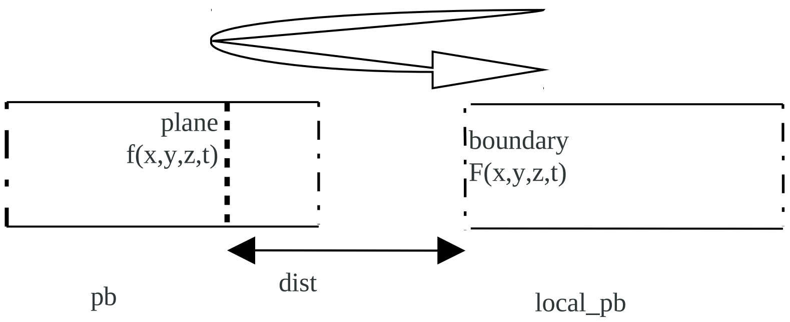
If inlet and outlet are not coincident meshes, but boundaries are coincident, you could use **champ_front_recyclage** which can work on this situation :
Read fluid
{
	...
	inlet **frontiere_ouverte_temperature_imposee**
		champ_front_recyclage { **pb_champ_evaluateur** box **temperature 1 }**
	...
}
You will notice that outlet boundary is not specified here cause the temperature field is interpolated on the nodes of the inlet boundary so outlet boundary should be located on the inlet boundary (else use **distance_plan** keyword).

- Champ_front_recyclage New keyword in the 1.6.1 version which replaces and generalizes several obsolete ones:
  - Champ_front_calc_intern
  - Champ_front_calc_recycl_fluct_pbperio
  - Champ_front_calc_recycl_champ
  - Champ_front_calc_intern_2pbs

Champ_front_calc_recycl_fluct

Champ_front_recyclage {
  pb_champ_evaluateur pb field nb_comp
  [**distance_plan** dist0 dist1 [dist2]]
  [**moyenne_imposee** methode_moy [**fichier** *file* [*second_file*]]
  [**moyenne_recyclee** methode_recyc [**fichier** *file* [*second_file*]]
  [**direction_anisotrope** 1|2|3]
  [**ampli_moyenne_imposee** 2|3 alpha(0) alpha(1) [alpha(2)]]
  [**ampli_moyenne_recyclee** 2|3 beta(0) beta(1) [beta(2)]]
  [**ampli_fluctuation** 2|3 gamma(0) gamma(1) [gamma(2)]]
}

This keyword is to use, in a general way, on a boundary of a local_pb problem, a field calculated from a linear combination of an imposed field g(x,y,z,t) with an instantaneous f(x,y,z,t) and a spatial mean field <f>(t) or a temporal mean field <f>(x,y,z) field extracted from a plane of a problem named pb (pb may be local_pb itself) :



For each component i, the field F applied on the boundary will be:

$$F_i(x,y,z,t) = \alpha_i * g_i(x,y,z,t) + \chi_i * [f_i(x,y,z,t) - \beta_i * \langle f_i \rangle]$$

The different options are :
pb_champ_evaluateur pb field nb_comp : To give the name of the pb problem, the name of the field of the problem and its number of components nb_comp.

distance_plan dist0 dist1 [dist2] : Vector which gives the distance between the boundary and the plane from where the field F will be extracted. By default, the vector is zero, that should imply the two domains have coincident boundaries.

ampli_moyenne_imposee 2|3 alpha(0) alpha(1) [alpha(2)] : α_i coefficients (by default =1)
ampli_moyenne_recyclee 2|3 beta(0) beta(1) [beta(2)] : β_i coefficients (by default =1)
ampli_fluctuation 2|3 gamma(0) gamma(1) [gamma(2)] : χ_i coefficients (by default =1)

direction_anisotrope direction : If an integer is given for direction (X:1, Y:2, Z:3, by default, direction is negative), the imposed field g will be 0 for the 2 other directions.

moyenne_imposee methode_moy : Value of the imposed g field. The methode_moy option can be :

profil [2|3] valx(x,y,z,t) valy(x,y,z,t) [valz(x,y,z,t)] : to specify analytic profile for the imposed g field.

interpolation fichier *file* : to create a imposed field built by interpolation of values read into a *file*. The imposed field is applied on the direction given by the keyword **direction_anisotrope** (the field is zero for the other directions). The format of the *file* is:

```
pos(1) val(1)
pos(2) val(2)
…
pos(N) val(N)
```

If direction given by **direction_anisotrope** is 1 (or 2 or 3), then pos will be X (or Y or Z) coordinate and val will be X value (or Y value, or Z value) of the imposed field.

connexion_approchee fichier *file* : to read the imposed field into a *file* where positions and values are given (it is not necessary that the coordinates of the points match the coordinates of the faces of the boundary, indeed, the nearest point of each face of the boundary will be used). The format of the *file* is:

```
N
x(1) y(1) [z(1)] valx(1) valy(1) [valz(1)]
x(2) y(2) [z(2)] valx(2) valy(2) [valz(2)]
…
x(N) y(N) [z(N)] valx(N) valy(N) [valz(N)]
```

connection_exacte fichier *file second_file* : to read the imposed field into two files. The first *file* contains the points coordinates (which should be the same than the coordinates of each faces of the boundary) and the *second_file* contains the mean values. The format of the first *file* is:

```
 N
 1 x(1) y(1) [z(1)]
2 x(2) y(2) [z(2)]
…
N x(N) y(N) [z(N)]
```

The format of the *second_file* is:

```
N
 1 valx(1) valy(1) [valz(1)]
2 valx(2) valy(2) [valz(2)]
…
N valx(N) valy(N) [valz(N)]
```

logarithmique diametre double **u_tau** double **visco_cin** double **direction** integer : to specify the imposed field (in this case, velocity) by an analytical logarithmic law of the wall :

g(x,y,z) = u_tau * (log(0.5*diametre*u_tau/visco_cin)/Kappa + 5.1)

With g(x,y,z)=u(x,y,z) if **direction** is set to 1 (g=v(x,y,z) if direction is set to 2, and g=w(w,y,z) if set to 3)

moyenne_recylee methode_recyc : Method used to do a spatial or a temporal averaging of f field to specify <f>. <f> can be the surface mean of f on the plane (**surface** option, see below) or it can be read from several files (for example generated by the **chmoy_faceperio** option of the **Traitement_particulier** keyword to obtain a temporal mean field). The option methode_recyc can be :

surfacique : surface mean for <f> from f values on the plane

Same options of methode_moy options but applied to read a temporal mean field <f>(x,y,z):

interpolation

connexion_approchee fichier *file*

connexion_exacte fichier *file second_file*

2.4.5SYNTAX TO DEFINE A MATHEMATICAL FUNCTION

In a mathematical function, used for example in field definition, it's possible to use the predifined function (an object parser is used to evaluate the functions) :

ABS	: absolute value function
COS	: cosinus function
SIN	: sinus function
TAN	: tan function
ATAN	: arctan function
EXP	: exponential function
LN	: neperian logaithm function
SQRT	: root mean square function
INT	: integer function
ERF	: erf function
RND(x)	: random function (values between 0 and x)
COSH	: hyperbolic cosinus function
SINH	: hyperbolic sinus function
TANH	: hyperbolic tangent function
ACOS	: inverse cosinus function
ATANH	: inverse hyperbolic tangent function
NOT(x)	: not equal to x
x_AND_y	: and function (returns 1 if x and y true else 0)
x_OR_y	: or function (returns 1 if x or y true else 0)
x_GT_y	: greater to (returns 1 if x>y else 0)
x_GE_y	: greater or equal to (returns 1 if x>=y else 0)
x_LT_y	: lesser to (returns 1 if x<y else 0)
x_LE_y	: lesser or equal to (returns 1 if x<=y else 0)
x_MIN_y	: minimum of x and y
x_MAX_y	: maximum of x and y
x_MOD_y	: modular division of x per y
x_EQ_y	: equal to (returns 1 if x=y else 0)
x_NEQ_y	: not equal to (returns 1 if x!=y else 0)

You can also use the following operations:

+	: addition
-	: substraction
/	: division
*	: multiplication
%	: modulo
$	: max
^	: power
<	: lesser than
>	: greater than
[	: less or equal to
]	: greater of equal to

You can also use the following constants:

Pi	: pi value (3,1415…)

The variables which can be used are:

x,y,z	: coordinates
t	: time

Examples:

Champ_front_fonc_txyz 2 cos(y+x^2) t+ln(y)
Champ_fonc_xyz dom 2 tanh(4*y)*(0.95+0.1*rnd(1)) 0.

Possible error:

Champ_fonc_txyz 1 cos(10*t)*(1<x<2)*(1<y<2)
Previous line is wrong. It should be written:
Champ_fonc_txyz 1 cos(10*t)*(1<x)*(x<2)*(1<y)*(y<2)

2.5 MEDIUM SPECIFICATION

There are several types of medium available. A physical value that is characteristic of a medium is always defined as follows:

name_of_the_physical_value field_type *field_description*

2.5.1 INCOMPRESSIBLE FLUID

```
Fluide_incompressible fluid
Read fluid
{
    Mu   field_type      field_description
    Rho Champ_Uniforme 1 vrel
    [ Cp Champ_Uniforme 1 vrel ]
    [ Lambda   field_type     field_description ]
    [ Beta_th   field_type     field_description ]
    [ Beta_co   field_type     field_description ]
    [ Indice   field_type     field_description ]
    [ Kappa   field_type     field_description ]
}
```

Mu: This is a keyword used to define the dynamic viscosity value (kg.m^{-1}.s^{-1}).

Rho: This is a keyword used to define the fluid density value (kg.m^{-3}).

Cp: This is a keyword used to define the specific heat value (J.kg^{-1}.K^{-1}).

Lambda: This is a keyword used to define the conductivity value (W.m^{-1}.K^{-1}).

Beta_th: This is a keyword used to define a thermal expansion value (K^{-1}).

Beta_co: This is the keyword which defines the volume expansion coefficient values in concentration

Indice: This is the keyword which defines the refractivity of fluid.

Kappa: This is the keyword which defines the absorptivity of fluid [m^{-1}].

2.5.2NON NEWTONIAN FLUID

```
Fluide_Ostwald fluid
Read fluid
{
      mu Champ_Ostwald
      K   field_type     field_description
      n   field_type     field_description
      rho Champ_Uniforme 1 vrel
   [  Cp Champ_Uniforme 1 vrel ]
   [  lambda   field_type     field_description ]
   [  Beta_th   field_type     field_description ]
   [  Beta_co   field_type     field_description ]
}
```

Fluide_Ostwald: This is the keyword used to describe non-Newtonian fluids, which are governed by Ostwald's law. The law applicable to stress tensor is:

tau=K(T)*(D:D/2)**((n-1)/2)*D

Where:
D refers to the deformation speed tensor
K refers to fluid consistency (may be a function of the temperature T)
n refers to the fluid structure index
- n=1 for a Newtonian fluid,
- n<1 for a rheofluidifier fluid
- n>1 for a rheothickening fluid

mu Champ_Ostwald: This keyword is used to define the viscosity variation law:
Mu(T)= K(T)*(D:D/2)**((n-1)/2)

K: This keyword is used to define fluid consistency

n: This keyword is used to define the fluid structure index

Rho: This keyword is used to define the fluid density value (kg.m^{-3}).

Cp: This keyword is used to define the specific heat value (J.kg^{-1}.K^{-1}).

Lambda: This keyword is used to define the conductivity value (W.m^{-1}.K^{-1}).

Beta_th: This keyword is used to define the thermal expansion value (K^{-1}).

Beta_co: This keyword is used to define the volume expansion coefficient values in concentration

2.5.3CONSTITUENT

```
Constituant  C
Read C
{
  Coefficient_diffusion   field_type    field_description
}
```

Coefficient_diffusion : This keyword is used to define the diffusion coefficient value (expressed in m^2.s^{-1}) of the constituent into the fluid. If a multi-constituent problem is being processed, the diffusion coefficients will be a vector field and each components will be the diffusion of the each constituent.

2.5.4SOLID

```
Solide solid
Read solid
{
 Rho Champ_Uniforme 1 vrel
 Cp Champ_Uniforme 1 vrel
 Lambda   field_type   field_description
}
```

Rho: This keyword is used to define the solid density value (kg.m^{-3}).

Cp: This keyword is used to define the specific heat value (J.kg^{-1}.K^{-1}).

Lambda: This keyword is used to define the conductivity value (W.m^{-1}.K^{-1}).

Observations:

• The user may simply define the properties relative to the problem.
• For the **Fluide_Incompressible (incompressible_fluid)** or **Solide (solid) type**, **Cp** and **Rho** must be **Champ_Uniforme (uniform_field)** type.

• The defined fields must have a physical value (specifically, they may not be set to zero or a negative value; if the user wishes to carry out the calculation, for example, without taking viscosity into consideration, a negligible diffusion operator should be used, but the user should not assign a value of zero to the viscosity field).
• When a thermohydraulic problem is being processed, gravity must be associated with the **Fluide_Incompressible (incompressible_fluid)** object type.

2.5.5 COMPRESSIBLE FLUID AT LOW MACH NUMBER

```
Fluide_Quasi_Compressible fluide
Read fluide
{
     mu Champ_Uniforme 1 vrel
     [ sutherland  mu0 value T0 value [Slambda value] C value ]
     lambda   field_type    field_description
     pression value
     loi_etat gaz_parfait {
        Prandtl value
        Cp value
        gamma value
        [ loi_etat Melange_gaz_parfait { Prandtl value Sc value }]
     }
     [loi_etat gaz_reel_rhoT {
           Prandtl  value
           Poly_T n+1 m+1
                a00 a01 … a0m a10 a11… a1m …an0…..anm
           Poly_rho n+1 m+1
                a00 a01 … a0m a10 a11… a1m …an0…..anm
           masse_molaire value
     }]
     [ Traitement_Pth  keyword ]
     [ Traitement_rho_gravite keyword ]
     [ temps_debut_prise_en_compte_drho_dt  value ]
     [ omega_relaxation_drho_dt  value ]
}
```

Keyword to define a gas for a calculation under a small Mach number approximation.
This gas may be a perfect gas :

mu : Dynamic viscosity mu [kg/m/s]

sutherland mu0 T0 [Slamba] C : Sutherland law for viscosity mu(T)=mu0*((T0+C)/(T+C))*(T/T0)**1.5 and (optional) for conductivity: lambda(T)=mu0*Cp/Prandtl*((T0+Slambda)/(T+Slambda))*(T/T0)**1.5
lambda : Thermal conductivity k [W/m/K]
Pression : Pressure [Pa]

For a perfect gas (**loi_etat_gaz_parfait**):
Prandtl : Prandtl number of the gas Pr=mu*Cp/k
Cp : Specific heat at constant pressure Cp [J/kg/K]
gamma : Cp/Cv with Cv specific heat at constant volume

Or a mixing or perfect gas (**loi_etat Melange_gaz_parfait**)
Prandtl : Prandtl number of the gas
Sc : Schmidt number of the gas Sc=nu/D (D: diffusion coefficient of the mixing)

Or a real gas (**loi_etat_gaz_reel**):
Poly_T : Law for the temperature [K] T(P,h)=a00+a01*P+a10*h+a11*P*h+a02*P*P+a20*h*h+….
with P pressure [hPa] and h enthalpy [J/kg]
Poly_rho : Law for the density [kg/m3]
rho(P,h)=a00+a01*P+a10*h+a11*P*h+a02*P*P+a20*h*h+….
with P pressure [hPa] and h enthalpy [J/kg]
Masse_molaire : Mass of the gas [kg/mol]

Traitement_Pth keyword : Optional keyword can be used in the description section of a quasi compressible fluid. With this keyword, it's possible to precise a particular treatment for the thermodynamic pressure Pth ; there are three possibilities:

1) with the keyword "edo" the code computes Pth solving an O.D.E. ; in this case, the mass is not strictly conserved (it is the default case for quasi compressible computation):
2) the keyword "conservation_masse" forces the conservation of the mass (closed geometry or with periodic boundaries condition).
3) the keyword "constant" makes it possible to have a constant Pth ; it's the good choice when the flow is open (e.g. with pressure boundary conditions).

Traitement_rho_gravite keyword : It may be :
1) "standard" : the gravity term is evaluted with rho*g (It is the default).
2) "moins_rho_moyen" : the gravity term is evaluated with (rho-rhomoy) *g.

temps_debut_prise_en_compte_drho_dt value : Optional option. While time<value, dRho/dt is set to zero (Rho, volumic mass). Useful for some calculation during the first time steps with big variation of temperature and volumic mass.

omega_relaxation_drho_dt value : Optional option to have a relaxed algorithm to solve the mass equation. value is used (1 per default) to specify ω:

$$\frac{\partial \rho}{\partial t}^{n+1} = \omega \frac{\rho^{n+1} - \rho^{n}}{dt} + (1-\omega)\frac{\partial \rho}{\partial t}^{n} = Div(\rho u)$$

2.6PROBLEMS

A problem is defined by creating an object and assigning the problem type that the user wishes to resolve:

•**Hydraulic problem**

Resolution of the NAVIER STOKES equations:

Pb_Hydraulique pb

•**Turbulent hydraulic problem**

Resolution of NAVIER STOKES equations with turbulence modelling:

Pb_Hydraulique_Turbulent pb

• **Thermohydraulic problem**

Resolution of coupled NAVIER STOKES/energy equations:

Pb_Thermohydraulique pb

• **Turbulent thermohydraulic problem**

Resolution of NAVIER STOKES/ energy coupled equations, with turbulence modelling.

Pb_Thermohydraulique_Turbulent pb

• **Hydraulic problem with concentration**

Resolution of NAVIER STOKES/multiple constituent transportation equations:

Pb_Hydraulique_Concentration pb

• **Turbulent hydraulic problem with concentration:**

Resolution of NAVIER STOKES/multiple constituent transportation equations

Pb_Hydraulique_Concentration_Turbulent pb

• **Thermohydraulic problem with concentration**.

Resolution of coupled NAVIER STOKES/multiple constituent transportation equations, with turbulence modelling:

Pb_Thermohydraulique_Concentration pb

• **Turbulent thermohydraulic problem with concentration**.

Resolution of coupled NAVIER STOKES/multiple constituent transportation equations, with turbulence modelling:

Pb_Thermohydraulique_Concentration_Turbulent pb

• **Conduction problem**

Resolution of the heat equation:

Pb_Conduction pb

• **Thermohydraulical problem quasi-compressible**
Resolution of thermohydraulical problem under smal Mach number:

Pb_Thermohydraulique_QC pb

• **Turbulent thermohydraulical problem quasi-compressible**

Resolution of Navier Stckes equations for a turbulent thermohydraulical problem under smal Mach number:

Pb_Thermohydraulique_Turbulent_QC pb

A problem is defined by creating an object and assigning the problem type that the user wishes to resolve:
To enter values for the problem objects created, the **Read** interpretor is used with a data block that is always structured as follows:

```
Read nom_pb
{

    bloc_lecture_equations

    [ Post_processing { ….. } ]
    [ Sauvegarde format_sauvegarde nom_fich ]
    [ Sauvegarde_simple format_sauvegarde nom_fich ]
    [ Reprise format_reprise nom_fich ]
}
```

In the following sub-chapters, the data blocks associated with each of the previously mentioned problem types will be presented. The **Post_processing**, **Sauvegardep)** and **Reprisert)** options are described in chapters 2.19, 2.17 and 2.18. Following this, this set of options will be assigned using *input_output_description* (outlet_inlet_block).

2.6.1HYDRAULIC PROBLEM

```
Read pb
{
    Navier_Stokes_Standard
    {
         Solveur_pression solveur { ….. }
       [ Dt_projection dt value ]
       [ Projection_initiale boolean ]
       [ methode_calcul_pression_initiale option ]
       [ Seuil_DivU value factor ]
        [ Solveur_bar {  } ]
         Diffusion { [dif] }
        [ uzawa value ]
         Convection { [schema] }
        [ Sources { [sou1] , [sou2] , … } ]
         Boundary_conditions { [cl_hydr1] [cl_hydr2] ….. }
        [ Initial_Conditions  { [cl_init] } ]
        [ ecrire_fichier_xyz_valeur nom_champ val_dt_impr bords integer ... ]
        [ equation_non_resolue condition(t) ]
        [ parametre_equation keyword ]
    }

    input_output_description
}
```

Navier_Stokes_Standard: This keyword is used to define NAVIER STOKES equations.

Solveur_pression: This keyword is used to define the linear pressure system resolution method. Refer to 2.10.

Dt_projection dt value: This keyword checks every period dt the equality of velocity divergence to zero. value is the criteria convergency for the solver used.

Projection_initiale boolean: Keyword to suppress, if boolean equals 0, the initial projection which checks DivU=0. By default, boolean equals 1.

methode_calcul_pression_initiale option : Keyword to select an option for the pressure calculation before the fist time step. Options are : **avec_les_cl** (default option, $\Delta P=0$ is solved with Neuman boundary conditions on pressure if any), **avec_sources** ($\Delta P=f$ is solved with Neuman boundaries conditions and f integrating the source terms of the Navier Stokes equation) and **avec_sources_et_operateurs** ($\Delta P=f$ is solved as with the previous option **avec_sources** but f integrating also some operators of the Navier Stokes equation). The two last options are useful and sometime necessary when source terms are implicited when using an implicit time scheme to solve the Navier Stokes equation.

Seuil_DivU value factor : this keyword is intended to minimise the number of iterations during the pressure system resolution. The convergence criteria during this step ("**seuil**" in solveur_pression) is dynamically adapted according to the mass conservation. At t^n , the linear system Ax=B is considered as solved if the residual $||Ax-B||<seuil(t^n)$. For t^{n+1}, the threshold value $seuil(t^{n+1})$ will be evualated as:

```
If ( |max(DivU)*dt|<value )
        Seuil(t^{n+1})= Seuil(t^n)*factor
Else
        Seuil(t^{n+1})= Seuil(t^n)*factor
Endif
```

The first parameter (**value**) is the mass evolution the user is ready to accept per timestep, and the second one (**factor**) is the factor of evolution for "seuil" (for example, 1.1, so 10% per time step). Investigations has to be lead to know more about the effects of these two last parameters on the behaviour of the simulations

Solveur_bar : This keyword is used to define when filtering operation is called (typically for **EF** convective scheme, **standard** diffusion operator and **Source_Qdm_lambdaup**). A file (solveur.bar) is then created and used for inversion procedure. Syntax is the same then for pressure solver (GCP is required for multi-processor calculations and, in a general way, for big meshes).

Diffusion: This keyword is used to specify the diffusion operator.

By default, no value is entered into the **Diffusion** { } block. But *dif* may be one of the scheme listed on 2.8.2.

Convection: Keyword to alter the convection scheme.

schema: This may be one of the scheme listed on 2.8.

Sources: This keyword is used to define the hydraulic equation source terms. Refer to 2.14.

sou: Source term definition.

Boundary_conditions: This keyword is used to define hydraulic boundary conditions. Refer to 2.13.1.

cl_hydr: Definition of a hydraulic boundary condition.

Initial_Conditions: This keyword is used to define the initial hydraulic conditions. Refer to 2.12.1.

cl_init: Defines the initial hydraulic conditions.

Ecrire_fichier_xyz_valeur: This keyword is used to write the values of a field for some boundaries in a text file with the following format:
n_valeur
x_1 y_1 [z_1] val_1
…
x_n y_n [z_n] val_n
The created files are named : pbname_fieldname_[boundaryname]_time.dat

Ecrire_fichier_xyz_valeur name_field val_dt_impr **bords** nb_bords boundary1…boundaryn

name_field: the name of the field to write (Champ_Inc, Champ_Fonc or a post_processed field)
val_dt_impr: the time period for printing in the file
bords: keyword to post-process only on some boundaries
nb_bords: number of boundaries
boundary1…boundaryn : name of the boundaries

The name of the files is *pb_name_field_name_time.dat*
Several **Ecrire_fichier_xyz_valeur** keywords may be written into an equation to write several fields. This kind of files may be read by **Champ_don_lu** or **Champ_front_lu** for example.

A binary file will be written if **Ecrire_fichier_xyz_valeur_bin** is used instead of **Ecrire_fichier_xyz_valeur** keyword.

The equation will not be solved while *condition(t)* is verified if **equation_non_resolue** keyword is used. Exemple: The Navier Stokes is not solved between time t0 and t1.

```
Navier_Sokes_Standard
{
        …
        equation_non_resolue (t>t0)*(t<t1)
}
```

Parametre_equation keyword : Keywords used to specify additional parameters for the equation. Two keywords are available for the moment: **parametre_implicite** when using an implicit time schema and **parametre_diffusion_implicite** when impliciting diffusion of the equation with the keyword **diffusion_implicite** used in an explicit time scheme. The options are listed below for each keyword:

```
Parametre_equation  parametre_implicite
{
  [ seuil_convergence_implicite float ]
  [ seuil_generation_solveur float ]
  [ seuil_verification_solveur float ]
  [ seuil_test_preliminaire_solveur float ]
  [ solveur solveur_sys_base ]
  [ resolution_explicite ]
  [ equation_frequence_resolue integer | f (t) ]
}
```

parametre_implicite: Keyword to change for this equation only the parameter of the implicit scheme used to solve the problem
seuil_convergence_implicite: Keyword to change for this equation only the value of **seuil_convergence_implicite** used in the implicit scheme

seuil_generation_solveur, seuil_verification_solveur, seuil_test_preliminaire_solveur: Keywords to change for this equation only the values of **seuil_generation_solveur** or **seuil_verification_solveur** or **seuil_test_preliminaire_solveur** used in the implicit scheme

solveur: Keyword to change for this equation only the solver used in the implicit scheme

resolution_explicite: Keyword to solve explicitly the equation whereas the scheme is an implicit scheme.
equation_frequence_resolue integer | f(t) : Keyword to specify that the equation is solved only every n time steps (n is an integer or given by a time-dependent function f(t)).

```
Parametre_equation  parametre_diffusion_implicite
{
  [ crank 0|1 ]
  [ niter_max_diffusion_implicite integer ]
  [ preconditionnement_diag 0|1 ]
  [ seuil_diffusion_implicite double ]
}
```

crank 0|1 : Use (1) or not (0, default) a Crank Nicholson method for the diffusion implicitation algorithm. Setting crank to 1 increases the order of the algorithm from 1 to 2.
niter_max_diffusion_implicite integer : Change the maximum number of iterations for the CG (Conjugate Gradient) algorithm when solving the diffusion implicitation of the equation.
preconditionnement_diag 0|1 : The CG used to solve the implicitation of the equation diffusion operator is not preconditioned by default. If this option is set to 1, a diagonal preconditionning is used. **Warning**: this option is not necessarily more efficient, depending on the treated case.
seuil_diffusion_implicite double : Change the threshold convergence value used by default for the CG resolution for the diffusion implicitation of this equation.

2.6.2TURBULENT HYDRAULIC PROBLEM

```
Read pb
{
  Navier_Stokes_Turbulent
  {
      …..
      Modele_turbulence modele { ….. }
      [ Traitement_Particulier {  kind_of_calculation } ]
  }

     input_output_description
}
```

Navier_Stokes_Turbulent: This keyword is used to define NAVIER STOKES equations as well as the associated turbulence model equations. The parameters are identical to those of **Navier_Stokes_standard** (refer to 5.4.1) with in addition:

Modele_turbulence: This keyword is used to define a turbulence model.

modele: Turbulence model selection. Refer to the chapter concerning turbulence models.

Traitement_particulier: Keyword to post-process particular values for two kinds of calculation:

1) **THI**: Keyword for a THI (Homogeneous Isotropic Turbulence) calculation:

```
Traitement_particulier {  THI {
          [init_Ec 0|1]
          [calc_spectre 0|1]
          [val_Ec double]
          [facon_init integer]
          [periode_calc_spectre double]
          [conservation_Ec]
          longueur_boite double
          [3D 0|1]
          [1D 0|1]
          [correlations 0|1]
          [champs_scalaires N field1 field2 … fieldN]
     }
}
```

init_Ec 0|1: Keyword to renormalize (1) or not (0) initial velocity so as kinetic energy equals to the value given by keyword val_Ec (default 0)
calc_spectre 0|1 : Keyword to calculate or not the spectrum of kinetic energy
val_Ec double : Keyword (VDF only) to impose a value for kinetic energy by velocity renormalization if init_Ec value is 1.
facon_init integer: Keyword (VDF only) to specify how to renormalize the initial velocity. The kinetic energy will be computed as the:

0: spatial kinetic energy
1: 1D spectral kinetic energy
3: 3D spectral kinetic energy

periode_calc_spectre double : Period of when to calculate spectrum (VEF option only)
conservation_Ec : If this keyword is used, velocity field will be changed as to have a constant kinetic energy (default 0, VEF option only).
longueur_boite double : Length of the calculation domain (VEF option only).
3D 0|1 : Calculate 3D spectrum (default 0, VEF option only).
1D 0|1 : Calculate 1D spectrum (default 0, VEF option only).
correlations 0|1 : Activate correlation calculation (default 0, VEF option only).
champs_scalaires N field1 field2 … fieldN: Add N scalar fields to the analysis (e.g. temperature) (Default, N=0, VEF option only)

Several files are created during the calculation.

2) **Canal**: Keyword for statistics on a periodic plane channel.

```
Traitement_particulier {  Canal {
        [ dt_impr_moy_spat value ]
        [ dt_impr_moy_temp value ]
        [ debut_stat value ]
        [ fin_stat value ]
        [ pulsation_w value ]
        [ nb_points_par_phase value ]
        [ reprise val_moy_temp_xxxxxx.sauv ]
        }
}
```

dt_impr_moy_spat value : Period to print the spatial average (default value is 1e6)
dt_impr_moy_temp value : Period to print the temporal average (default value is 1e6)
debut_stat value : time to start the temporal averaging (default value is 1e6)
fin_stat value : time to end the temporal averaging (default value is 1e6)

pulsation_w value : pulsation for phase averaging (in case of pulsating forcing term) (no default value)
nb_points_par_phase value : number of samples to represent phase average all along a period (no default value)
reprise val_moy_temp_xxxxxx.sauv : keyword to restart a calculation with previous average quantities

Note that for thermal and turbulent problems, averages on temperature and turbulent viscosity are automatically calculated.
To restart a calculation with phase averaging, val_moy_temp_xxxxxx.sauv_phase file is required on the directory where the job is submitted (this last file will be then automatically loaded by TRUST)

3) THI_NEW: Other keyword for a THI (Homogeneous Isotropic Turbulence) calculation

For unstructured approach, Traitement_particulier have been slightly modified. Averaging process respects better than previously the real location of the quantities (in the y-direction). Moreover, indications like friction velocity and reynolds, and evolution of the time step are calculated and written respectively in the files u_tau, reynolds_tau and dt_evol.
General syntax : Just substitute the previous keyword "nb_int" and "dir_echant" to the following ones : Ny and eps. Ny is set to initialise the dimension of tables and eps is the tolerance to check if points belongs to the same y-co-ordinate of the other points.
Note that the present post-treatment is suitable only for plane channel configuration with wall normal direction according to y.

Example:
```
Traitement_particulier
{
THI_new{
    init_Ec 1 val_Ec 1.5 facon_init 0
    calc_spectre 1
    }
}
```

A new treatment for the temperature field is available for THI computation:
THI_thermo

It offers the possibility to :
- evaluate the probability density function on temperature field,

- gives in a file the temperature field for a future spectral analysis
- monitor the evolution of the max and min temperature on the whole domain

The syntax is the same than for special treatment THI :

Traitement_particulier { THI_thermo { init_Ec 1 val_Ec 1.5 facon_init 0 calc_spectre 1 } }

Traitement_particulier { **chmoy_faceperio** { **stats** val1 val2 } }

This keyword is used to save in two files :
a) the coordinates of the points located at the periodic boundaries (**geom_face_perio**)
b) the temporal averaged velocity associated to these points (**chmoy_face_perio**) between time val1 and val2.

Il will be useful then to generate fluctuating inlet conditions.

4) **Ec**: Keyword to print total kinetic energy into the referential linked to the domain (keyword **Ec**). In the case where the domain is moving into a Galilean referential, the keyword **Ec_dans_repere_fixe** will print total kinetic energy in the Galilean referential whereas **Ec** will print the value calculated into the moving referential linked to the domain. **Periode** is the keyword to set the period of printing into the file *datafile_Ec.son* or *datafile_Ec_dans_repere_fixe.son*.

Traitement_particulier { **Ec** { **Ec|Ec_dans_repere_fixe periode** double } }

2.6.3THERMOYDRAULIC PROBLEM

```
Read pb
{
   Navier_Stokes_Standard
   {
      …..
      [ Traitement_Particulier {  kind_of_calculation } ]
   }
   Convection_diffusion_temperature
   {
        Diffusion { [dif] }
        Convection { [schema] }
      [ Sources { [sou1] [sou2] ….. } ]
        Boundary_conditions { [cl_therm1] [cl_therm2] ….. }
      [ Initial_Conditions  { [cl_init] } ]
      [ parametre_equation keyword ]
   }

   input_output_description
}
```

Navier_Stokes_Standard: This keyword is used to define NAVIER STOKES equations.

Convection_diffusion_temperature: This keyword is used to define the energy equation (temperature diffusion convection).

Diffusion: This keyword is used to specify the diffusion operator.

By default, nothing is put in the **Diffusion** { } block.
dif: The value of dif should be **Negligeable** to suppress the temperature diffusion convection equation's diffusion operator.

Convection: This keyword is used to change the convection scheme (by default, the UPWIND scheme is selected).

schema: May be set to one of the scheme listed on 2.8.

Sources: This keyword is used to define the energy equation source terms.
sou: Defines the source term.

Boundary_conditions: This keyword is used to define thermal boundary conditions. Refer to 2.13.2.

cl_therm: Defines a thermal boundary condition.

Initial_Conditions: This keyword is used to define initial thermal conditions. Refer to 2.12.2.

cl_init: Defines initial thermal conditions on the domain.

Traitement_particulier : Optional keyword to calculate some interesting values.

```
Traitement_particulier {  Temperature { Bord  boundary Direction integer } }
```

Keyword to print mass flow rate and averaged temperature on the boundary. It generates 2 external files : *RhoU_boundary* and *Tmoyen_boundary*. The first file gives the product rho*U*S at the *boundary* specified above, where U is the velocity in the direction defined by the integer value (0:X, 1:Y, 2:Z). The second one gives at the *boundary* the averaged temperature (according to Sum(rho*U*S*TdS)/Sum(rho*U*SdS)) and Tmin - Tmax for each time step.

NB : This calculation available only in VEF framework assumes that all the faces of *boundary* are on the same processor.

Parametre_equation : See 2.6.1

2.6.4TURBULENT THERMOHYDRAULIC PROBLEM

```
Read pb
{
   Navier_Stokes_Turbulent
   {
      …..
   }
 Convection_diffusion_temperature_turbulent
   {
      mêmes instructions que Convection_Diffusion_temperature
      Modele_turbulence modele { }
   }

   input_output_description

}
```

Version 1 does not feature the functionality required to process a turbulence problem in VEF discretization.

Navier_Stokes_Turbulent: This keyword is used to define NAVIER STOKES equations with turbulence modelling.

Convection_diffusion_temperature_turbulent: This keyword is used to define the energy equation (temperature diffusion convection). Parameters are identical to **Convection_diffusion_temperature** (refer to 2.6.2) with in addition:

Modele_ turbulence: This keyword is used to define a turbulence model.

modele: The turbulence model selected for the energy equation. The only currently available model is **Prandtl**.

2.6.5HYDRAULIC PROBLEM WITH CONCENTRATION

```
Read pb
{
    Navier_Stokes_Standard
    {
       …..
         [ Traitement_Particulier {  ConcMoy { ConcMoy periode double Tx1 double Tx2 double Tx3
double  } } ]
    }
    Convection_diffusion_concentration
    {
         Sources { [Source_Constituant field_type field_description] … }
         Diffusion { [dif] }
         Convection { [schema] }
         Boundary_conditions { [cl_conc1] [cl_conc2] ….. }
       [ Initial_Conditions  { [cl_init] } ]
       [ Nom_inconnue name ]
       [ parametre_equation keyword ]
    }

     input_output_description

}
```

Navier_Stokes_Standard: this keyword is used to define NAVIER STOKES equations.

Sources: This keyword is used to specify the source terms of the equation. **Source_Constituant** is a keyword to specify source rates, in [[C]/s], for each one of the nb constituents. [C] is the concentration unit.

Convection_diffusion_concentration: This keyword is used to define the constituent transportation vectorial equation (concentration diffusion convection).

Diffusion: This keyword is used to specify the diffusion operator.

dif: This is set to **Negligeable** to suppress the constituent transportation equation diffusion operator.

Convection: This keyword is used to modify the convection scheme (by default, this is set to the UPWIND scheme).

schema: This may be one of the scheme listed on 2.8.

Boundary_conditions: Keyword to define concentration boundary conditions. Refer to 2.13.3.

cl_conc: Definition of a concentration boundary condition.

Initial_Conditions: This keyword is used to define initial concentration conditions. Refer to 2.13.3.

cl_init: Definition of initial concentration conditions.

Nom_inconnue name: Keyword **Nom_inconnue** will rename the unknown of this equation with the given name. In the postprocessing part, the concentration field will be accessible with this name. This is usefull if you want to track more than one "concentration" (otherwise, only the concentration field in the first concentration equation can be accessed).

Parametre_equation : See 2.6.1

Traitement_particulier: Keyword to post-process particular values for concentration equation:
ConcMoy periode double: printing period for values in the *datafile_ConcMoy.son* file
Tx1 double: Limit 1 for concentration rate
Tx2 double: Limit 2 for concentration rate
Tx3 double: Limit 3 for concentration rate

The format file is :

```
# Time  ConcentrationRate1 ConcentrationRate2 ConcentrationRate3
0            0                  0                  0
0.1          0                  0.001              0.002
….
```

Where ConcentrationRateI is evaluated with the concentration field C(x,y,z) :
ConcentrationRateI=Sum(volume where Tx(x,y,z)<TxI)/GlobalVolume
with Tx(x,y,z)=|C(x,y,z)/MeanConcentration -1|
MeanConcentration=Sum(C(x,y,z)*volume)/GlobalVolume

2.6.6TURBULENT HYDRAULIC PROBLEM WITH CONCENTRATION

```
Read pb
{
    Navier_Stokes_Turbulent
    {
        …..
    }
    Convection_diffusion_concentration_turbulent
    {
        …
        Modele_turbulence modele { }
    }

     input_output_description

}
```

Version 1 does not feature the functionality required to process turbulence problems in VEF discretization.

Navier_Stokes_Turbulent: This keyword is used to define the NAVIER STOKES equations and the associated turbulence model equations. Refer to 2.6.4.

Convection_diffusion_concentration_turbulent: This keyword is used to define the constituent transportation equations (concentration diffusion convection). The parameters are identical to **Convection_diffusion_concentration** (refer to 2.6.5) with in addition:

Modele_turbulence: This keyword is used to define a turbulence model.

modele: Selection of the turbulence model to be used in the constituent transportation equation. The only model currently available is **Schmidt**.

2.6.7THERMOHYDRAULIC PROBLEM WITH CONCENTRATION

```
Read pb
{
    Navier_Stokes_Standard
    {
       …..
    }

    Convection_diffusion_temperature
    {
       …..
    }

    Convection_diffusion_concentration
    {
       …..
    }

     input_output_description
}
```

Navier_Stokes_Standard: This keyword is used to define NAVIER STOKES equations. Refer to 2.6.1.

Convection_diffusion_temperature: This keyword is used to define the energy equation (temperature diffusion convection). Refer to 2.6.2.

Convection_diffusion_concentration: This keyword is used to define constituent transportation equations (concentration diffusion convection). Refer to 2.6.5.

2.6.8THERMOHYDRAULIC TURBULENT PROBLEM WITH CONCENTRATION

```
Read pb
{
   Navier_Stokes_Turbulent
    {
       …..
    }

    Convection_diffusion_temperature_turbulent
    {
       …..
    }

    Convection_diffusion_concentration_turbulent
    {
       …..
    }

     input_output_description

}
```

Version 1 does not yet feature the functionality required to process a turbulence problem in VEF discretization.

Navier_Stokes_Turbulent: This keyword is used to define NAVIER STOKES equations and the associated turbulence model. Refer to 2.6.1.

Convection_diffusion_temperature_turbulent: This keyword is used to define the energy equation (temperature diffusion convection). Refer to 2.6.3.

Convection_diffusion_concentration_turbulent: This keyword is used to define the constituent transportation equations (concentration diffusion convection). Refer to 2.6.5.

2.6.9CONDUCTION PROBLEM

```
Read pb
{
   Conduction
   {
       Diffusion { [dif] }
       [ Sources { [sou1] [sou2] ….. } ]
         Boundary_conditions { [cl_therm1] [cl_therm2] ….. }
       [ Initial_Conditions { [cl_init] } ]
       [ parametre_equation keyword ]
   }

    input_output_description

}
```

Conduction: This keyword is used to define the heat equation.

Diffusion: This keyword is used to specify the diffusion operator.

dif: Set to **Negligeable** to suppress the constituent transportation equation diffusion operator.

Sources: This keyword is used to define the heat equation volume power type source terms. Refer to 2.15.

sou: Source term definition.

Boundary_conditions: This keyword is used to define the thermal boundary conditions. Refer to 2.6.4.
cl_therm: Defines the thermal boundary condition.

Initial_Conditions: This keyword is used to define initial thermal conditions. Refer to 2.12.2.
cl_init: Defines the initial thermal conditions.

Parametre_equation : See 2.6.1

2.6.10 PROBLEM FOR NAVIER STOKES EQUATIONS UNDER A SMALL MACH NUMBER APPROXIMATION

```
Pb_Thermohydraulique_QC
 Read pb
 {
        Navier_Stokes_QC { … }
        Convection_Diffusion_Chaleur_QC
        {
          diffusion { }
          convection { }
          [ mode_calcul_convection ancien
                                | divuT_moins_Tdivu
                                | divrhouT_moins_Tdivrhou ]
          sources { }
          Boundary_conditions {  }
          Initial_Conditions { }
        }
       …
 }

 Solve pb
```

The useful keywords for the solved equations are :

Navier_Stokes_QC : equation for momentum

Convection_Diffusion_Chaleur_QC : equation for energy

Mode_calcul_convection : Option to set the form of the convective operator:
divrhouT_moins_Tdivrhou (the default since 1.6.8): rho.u.gradT = div(rho.u.T)- Tdiv(rho.u.1)
divuT_moins_Tdivu : u.gradT = div(u.T) - Tdiv(u.1)
ancien : u.gradT = div(u.T) – T.div(u)

The boundary conditions are described here 2.13.1.

Keywords for the unknowns other than pressure, velocity, temperature are:
masse_volumique : density
enthalpie : enthalpy
pression : reduced pressure
pression_tot : total pressure

2.6.11 TURBULENT THERMOHYDRAULICAL PROBLEM UNDER SMALL MACH NUMBER

```
Pb_Thermohydraulique_Turbulent_QC
 Read pb
 {
        Navier_Stokes_Turbulent_QC { … }
                Convection_Diffusion_Chaleur_Turbulent_QC
               {
                        diffusion { }
                        convection { }
                        sources { }
                        boundary_conditions
                        Initial_Conditions { }

          input_output_description

 }

 Solve pb
```

New problem for Navier Stokes equations under a small Mach number approximation and with turbulence model (standard k-eps or k-eps at Low Reynolds)
New keywords for the solved equations are :
Navier_Stokes_Turbulent_QC : equation and tubulence model for momentum
Convection_Diffusion_Chaleur_Turbulent_QC : equation and turbulence model for energy
Prandtl : To give the value of Prandtl number in the Prandtl model.

Warning: Available for VDF and VEF P0/P1NC discretization only.
Low Reynolds k-eps model available in VDF discretisation only.

2.6.12DISCONTINOUS FRONT TRACKING PROBLEMS

The generic Front-Tracking problem (**Probleme_FT_Disc_gen**) in the discontinuous version differs from the rest of the TRUST code: The problem does not state the number of equations that are enclosed in the problem. Two equations are compulsory: a momentum balance equation (alias Navier-Stokes equation) and an interface tracking equation. The list of equations to be solved is declared in the beginning of the data file.

Another difference with more classical TRUST data file, lies in the fluids definition. The two-phase fluid (**Fluide_Diphasique**) is made with two usual single-phase fluids (**Fluide_Incompressible**).
These two specificities lead to the following general structure of the data file:

```
dimension 3
Probleme_FT_Disc_gen pb
Domaine DOM
Read_file DOM domain.geom
VEFPreP1B dis
Schema_Euler_explicite  sch
Read sch { ../.. }
Fluide_Incompressible liquid
Read liquid { ../.. }
Fluide_Incompressible gas
Read gas { ../.. }
Fluide_Diphasique fluids
Read fluids { ../.. }
Constituant constituant
Read constituant { ../.. }
Champ_Uniforme gravite
Read gravite 3 0. 0. -9.81
Associate fluide gravite
Navier_Stokes_FT_Disc           eq_hydraulique
Transport_Interfaces_FT_Disc       agit
Transport_Interfaces_FT_Disc       interf
Convection_Diffusion_Concentration  eq_diffusion
Associate pb eq_hydraulique
Associate pb agit
Associate pb interf
Associate pb eq_diffusion
Associate pb DOM
Associate pb sch
Associate pb fluids
Associate pb constituant
Discretize pb dis
Read pb
{
 eq_hydraulique { ../.. }
 agit         { ../.. }
 interf        { ../.. }
 eq_diffusion   { ../.. }
 liste_postraitements { ../.. }
}
Solve pb
Fin
```

In the previous example, the two-phase fluid (**Fluide_Diphasique**) is named fluids and is composed made with two usual single-phase non-compressible fluids (**Fluide_Incompressible**) named *liquid* and *gas*.
In the previous example, the Front-Tracking problem (**Probleme_FT_Disc_gen**) includes four equations:
- a momentum balance equation (**Navier_Stokes_FT_Disc**) named *eq_hydraulique*,
- a first interface tracking equation (**Transport_Interfaces_FT_Disc**) named *agit*,
- a second interface tracking equation (**Transport_Interfaces_FT_Disc**) named *interf*,
- a simple convection-diffusion equation (**Convection_Diffusion_Concentration**) of a passive scalar (a concentration in chemical specie) named *eq_diffusion*.

As the list of equations to be solved in the generic Front-Tracking problem is declared in the data file and not pre-defined in the structure of the problem, each equation has to be distinctively associated with the problem with the **Associate** keyword.

Some comments on the time scheme

At the time we write this document, the only time scheme for which a proper use has been proven is the explicit Euler scheme (**Schema_Euler_explicite**). An example of parameters for this scheme is:

```
Schema_Euler_explicite  sch
Read sch
{
        tinit 0.
        tmax 100.
        dt_min 1.e-7
        dt_max 1.e-3
        dt_impr 10.
        dt_sauv 0.2
        facsec 0.8
        seuil_statio -1
}
```

In most situations that have already been experienced, it is advisable not to let the code to select the proper time stepping, but to provide some limits. The usual **facsec** keyword with values lesser than unity can be used along with a maximum time step value defined by **dt_max**. The threshold value -1 for **seuil_statio** can be used to prevent the code from stopping during a calm period before another interesting event.

2.6.12.1Two-phase fluid description

It is necessary to first declare the two phases, and second to declare the two-fluid mixture, water and air in the following example:

```
Fluide_Incompressible liquid
Read liquid
{
        mu  Champ_Uniforme 1 0.001
        rho Champ_Uniforme 1 1000
}
Fluide_Incompressible gas
Read gas
{
        mu  Champ_Uniforme 1 0.00001
        rho Champ_Uniforme 1 1.3
}
Fluide_Diphasique fluids
Read fluids
{
        fluide0 liquid
        fluide1 gas
        sigma Champ_uniforme 1 0.07
        [chaleur_latente Champ_uniforme 1 double ]
}
```

Obviously, **rho** stands for the fluid density, **mu** for its dynamic viscosity and **sigma** for the surface tension between the two fluids. Another possible attribute of the **Fluide_Diphasique** object is **chaleur_latente** for the heat of phase-change (given to the fluid when change from phase 0 to phase 1) in diabatic liquid-vapor problems. It is a signed value for **chaleur_latente** so negative value is possible if the phase 0 is vapor and phase 1 is liquid.
The classical use of **fluide0**/**fluide1** is to choose **fluide0** for the liquid (or the denser phase) and **fluide1** for the gas, the vapor or the lighter phase. With this choice, the indicator function used in the code is equivalent to the classical void fraction: the indicator function is equal to *0* in **fluide0** and is equal to *1* in **fluide1**. The other choice of **fluide0**/**fluide1** is possible, but should be used more carefully.

The buoyancy forces

The buoyancy forces come out from the gravity acceleration and the difference of density between *fluide0* and *fluide1*. The simplest way of introducing the buoyancy forces is:

```
Champ_Uniforme gravity
Read gravity 3 0 0 -9.81
Associate fluids gravity
```

Another possible way of producing buoyancy forces lies in the forces associated to the change of frame of reference. This term is a source term, included in the momentum balance equation:

```
Read pb
{
eq_hydraulique
{
        solveur_pression GCP { ../.. }
        convection          { ../.. }
        diffusion           { ../.. }
        sources { acceleration { acceleration Champ_Fonc_t 3 0. 0. -9.81 } }
        ../..
}
```

2.6.12.2Two-phase momentum balance equation

The two-phase momentum balance equation (**Navier_Stokes_FT_Disc**) has the same attributes as single–phase equations and additional features:

```
eq_hydraulique
{
        solveur_pression GCP { precond ssor { omega 1.5 } seuil 1.e-12 impr }
        modele_turbulence model
        convection        { scheme }
        Initial_Conditions  { ../.. }
        boundary_conditions   { ../.. }
        diffusion        { scheme }
        matrice_pression_invariante
        equation_interfaces_proprietes_fluide interf
     [ equation_temperature_mpoint  temperature_equation ]
        equations_interfaces_vitesse_imposee 1 agit
        clipping_courbure_interface 10000.
        [ Terme_gravite rho_g | grad_I ]
     [ equations_concentration_source_vortex N EQ_1 ... EQ_N ]
        [ repulsion_aux_bords MINX MAXX SLOPE ]
     [ penalisation_forcage { [ pression_reference double ] } ]
}
```

The pressure solvers are common with single-phase Navier-Stokes equations. The most used choice is the **GCP** solver. Alternative choice is for instance solvers from the **Petsc** API package.
The value of **seuil** should be decreased as the mesh size is refined and the number of time steps is increased. For 1000 time steps on a 10^6-nodes domain, indicative maximum values are *1.e-6* for a 1-dm^3 domain and *1.e-9* for a 1-cm^3 domain.

Up today, two keywords are available for *model*. First, taking the keyword **nul** for *model* means you consider the flow is laminar. Another choice, for a turbulent flow, is the Wale model:

```
modele_turbulence sous_maille_wale
{
        Cw 1.e-16
        turbulence_paroi negligeable
}
```

The value of *1.e-16* for **Cw** is another way of having a negligible turbulence model. A more physical value (default) is *0.5*. However, even with no intention of having of sub-grid turbulence model, the use of **modele_turbulence** with a keyword different from **nul** allows the strain tensor τ^D to be calculated with $\left(\nabla V + {}^t\nabla V\right)$ and not only with $\left(\nabla V\right)$ alone (this has no physical ground but is specific to the way the strain tensor is calculated in TRUST). This second choice is highly recommended.

The **convection** options depend on the discretization choice. With a structured discretization (**VDF**), the most usual choice for *scheme* is **quick** and a with a non-structured discretization (**VEFPreP1B**), it is **muscl**.

The **Initial_Conditions** block is used to define the initial value of the unknown field, the velocity field (**vitesse**) in this momentum balance equation. An example is a computed velocity field (with two zero components):

```
Initial_Conditions
{
        vitesse Champ_fonc_xyz dom 3 0. 0. -32*(x-0.025)*(x-0.025)-0.02
}
```

The **boundary_conditions** block is used to specify boundary conditions.

```
boundary_conditions
{
        droite Paroi_fixe
        bas Frontiere_ouverte_vitesse_imposee champ_front_uniforme 3 0. 0. 1.
        haut Sortie_libre_rho_variable champ_front_uniforme 1 0.
        perioy Periodique
}
```

The keyword **Paroi_fixe** is used to specify zero velocity at the boundary (adherence).
The keyword **Frontiere_ouverte_vitesse_imposee** is used to specify a non-zero velocity at the boundary.
The keyword **Sortie_libre_rho_variable** is used to define an outlet boundary condition at which the pressure is defined through the given field, whereas the density of the two-phase flow may varies (value of P/ρ given in Pa/kg.m^{-3}).
The keyword **Periodique** is used to define periodic boundary condition. The syntax is the same as for single-phase problems. However, this boundary condition has not yet been used successfully with interfaces crossing through the periodic boundary…

Other available boundary condition:

```
Frontiere_ouverte_vitesse_vortex
{
   sous_zone SOUS_ZONE_NAME
   equation  EQ_NAME
   integrale_reference REF_VAL
   signe -1 | 1
   coeff_vitesse DIM vx vy [vz]
}
```

This boundary condition inherits from **Frontiere_ouverte_vitesse_imposee** and might be used to model a "spillway" (e.g. maintain a given altitude of a free surface at some point close to the boundary condition). The velocity of the fluid is a uniform field equal to the vector **coeff_vitesse** multiplied by a factor f. The indicator function of equation EQ_NAME (must be an interface transport equation) is integrated over the region SOUS_ZONE_NAME, then REF_VAL is substracted. If this value (called "factor") is of the requested sign (signe keyword), then the applied velocity is coef_vitesse*factor, otherwise the velocity is zero. SOUS_ZONE_NAME should be a small sub_zone close to the boundary condition. The amplitude of **coeff_vitesse** determines the time constant for the liquid level adjustment.

The **diffusion** block has been used for specific model in two-phase cell. However, this model is not available in the current version of TRUST: so the keyword **viscosite_fortement_variable** is not currently implemented and *scheme* keyword is limited to those available in single-phase problems.

The **matrice_pression_invariante** keyword is a shortcut to be used only when the flow is a single-phase one, with interface tracking only used for solid-fluid interfaces. In this peculiar case, the density of the fluid does not evolve during the computation and the pressure matrix does not need to be actuated at each time step.

The **equations_interfaces_vitesse_imposee** keyword is used to specify the velocity field to be used when using an interface that mimics a solid interface moving with a given solid speed of displacement. When this case is selected, the keyword sequence **Methode_transport vitesse_imposee** in the **Transport_Interfaces_FT_Disc** block will define the velocity field for the displacement of the interface. If two or more solid interfaces are defined, then the keyword **equations_interfaces_vitesse_imposee** should be used as**:**

equations_interfaces_vitesse_imposee number_of_equations equation_name1 equation_name2 …

The **equation_interfaces_proprietes_fluide** block is used for liquid-gas, liquid-vapor and fluid-fluid deformable interface, which transported at the Eulerian velocity. When this case is selected, the keyword sequence **Methode_transport vitesse_interpolee** is used in the block **Transport_Interfaces_FT_Disc** to define the velocity field for the displacement of the interface.

The **equation_temperature_mpoint** should be used in the case of liquid-vapor flow with phase-change (see the $TRUST_ROOT/doc/TRUST/ft_chgt_phase.pdf written in French for more information about the model). The name of the temperature equation, defined with the **convection_diffusion_temperature_ft_disc** keyword, should be given.

The **clipping_courbure_interface** block is used to numerically limit the values of curvature used in the momentum balance equation. Curvature is computed as usual, but values exceeding the clipping value are replaced by this threshold, before using the clipped curvature in the momentum balance. Each time a curvature value is clipped, a counter is increased by one unity and the value of the counter is written in the err file at the end of the time step. This clipping allows not reducing drastically the time stepping when a geometrical singularity occurs in the interface mesh. However, physical phenomena may be concealed with the use of such a clipping!

The **Terme_gravite** keyword changes the numerical scheme used for the gravity source term. The default is **grad_i**, which is designed to remove spurious currents around the interface. In this case, the pressure field does not contain the hydrostatic part but only a jump across the interface. This scheme seems not to work very well in vef. The **rho_g** option uses the more traditional source term, equal to rho*g in the volume. In this case, the hydrostatic pressure is visible in the pressure field and the boundary conditions in pressure must be set accordingly. This model produces spurious currents in the vicinity of the fluid-fluid interfaces and with the immersed boundary conditions.

equations_concentration_source_vortex N EQ_1 ... EQ_N : see **Source_Constituant_Vortex** keyword.

repulsion_aux_bords MINX MAXX SLOPE : This keyword is a hack to prevent bubbles (or droplets) to touch walls. The potential used to take into accound gravity is modified to provide a repulsive force located outside the region minx<=X<=maxx and minx<=Y<=maxx (these coordinates should be at a few mesh cells of the walls inside the domain). SLOPE is the gradient of the potential (2-4 times the gravity should work). Of course, this hack works for rectangular boxes only...

penalisation_forcage { [**pression_reference** double] } : This keyword is useful when a solid-fluid interface is used (see 2.6.12.4). Default is the Direct Forcing method to impose the velocity of the solid-fluid interface. With this keyword **penalisation_forcage**, user can switch to the Penalized Direct Forcing method. If the optional keyword **pression_reference** is given, the pressure is L2 penalized to the specified value.

2.6.12.3Fluid-fluid interface tracking equation

To try to be clearer, the two types of interface tracking equations are explained separately. In this section, the case of fluid-fluid interface tracking equations is described. This description stands for all cases of fluid-fluid two-phase flow. The case of solid-fluid interaction will be described in the next section.

In order to perform fluid-fluid two-phase flow (liquid-gas or liquid-liquid), **equation_interfaces_proprietes_fluide** has been declared in the Navier-Stokes equation and the interface tracking equation (**Transport_Interfaces_FT_Disc**) has to specify **vitesse_interpolee** for the **methode_transport**. Additional parameters are the remeshing parameters, initial and boundary conditions:

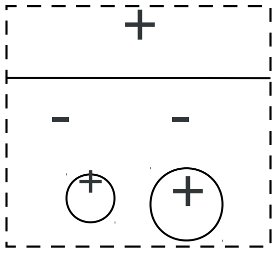
```
interf
{
        methode_transport vitesse_interpolee eq_hydraulique
        methode_interpolation_v method
        Initial_Conditions {   fichier_geom ... | fonction ... [ , fonction ... | fonction_ignorer_collision ... ] .... }
        boundary_conditions  { ../.. }
        [ maillage  { ../.. } ]
        remaillage  { … }
        [ collisions {  … }  ]
        n_iterations_distance nb
        iterations_correction_volume nb
        volume_impose_phase_1 volume
        [ Parcours_interface { [ correction_parcours_thomas ]  } ]
        [ injecteur_interfaces FILENAME ]
        [ suppression_sous_zone SOUS_ZONE_NAME ]
        [ sous_zone_volume_impose SOUS_ZONE_NAME ]
        [ interpolation_repere_local ]
}
```

In the block **methode_transport**, the keyword **vitesse_interpolee** is used to specify that the interpolation will use the velocity field of the Navier-Stokes equation named *eq_hydraulique* to compute the speed of displacement of the nodes of the interfaces.

In the block **methode_interpolation_v**, two keywords are possible for *method* to select the way the interpolation is performed. With the choice **valeur_a_elem** the speed of displacement of the nodes of the interfaces is the velocity at the center of the Eulerian element in which each node is located at the beginning of the time step. This choice is the default interpolation method. The choice **VDF_lineaire** is only available with a VDF discretization (**VDF**). In this case, the speed of displacement of the nodes of the interfaces is linearly interpolated on the 4 (in 2D) or the 6 (in 3D) Eulerian velocities closest the location of each node at the beginning of the time step. In peculiar situation, this choice may provide a better interpolated value. Of course, this choice is not available with a VEF discretization (**VEFPreP1B**).

The keyword **Initial_Conditions** is used to define the shape of the initial interfaces through the zero level-set of a **function**, or through a mesh **fichier_geom** (Refer to 2.6.12.4). Indicator function is set to *0*, that is *fluide0*, where the function is negative; indicator function is set to *1*, that is *fluide1*, where the function is positive; the interfaces are the level-set *0* of that function:

```
Initial_Conditions { fonction
(-((x-0.002)^2+(y-0.002)^2+z^2-(0.00125)^2))*((x-0.005)^2+(y-0.007)^2+z^2 (0.00150)^2))*(0.020-z))
}
```



In the above example, there are three interfaces: two bubbles in a liquid with a free surface. One bubble has a radius of *0.00125, i.e. 1.25 mm,* and its center is {*0.002, 0.002, 0.000*}. The other bubble has a radius of *0.00150, i.e. 1.5 mm,* and its center is {*0.005, 0.007, 0.000*}. The free surface is above the two bubble, at a level *z=0.02.*

Additional feature in this block concerns the keywords **ajout_phase0** and **ajout_phase1**. They can be used to simplify the composition of different interfaces. When using these keywords, the initial function defines the indicator function; **ajout_phase0** and **ajout_phase1** are used to modify this initial field. Each time **ajout_phase0** is used, the field is untouched where the function is positive whereas the indicator field is set to *0* where the function is negative. The keyword **ajout_phase1** has the symmetrical use, keeping the field value where the function is negative and setting the indicator field to *1* where the function is positive. The previous example can also be written:

```
Initial_Conditions {
        fonction z-0.020 ,
        fonction ajout_phase1 (x-0.002)^2+(y-0.002)^2+z^2-(0.00125)^2 ,
        fonction ajout_phase1 (x-0.005)^2+(y-0.007)^2+z^2-(0.00150)^2
}
```

The **boundary_conditions** block is used to specify boundary conditions. The keyword adapted to compute the indicator function boundary conditions in the case of two-phase flows is **Paroi_ft_disc**. Two kind of boundary condition may be applied : symmetry and contact angle fixed. Symmetry is equivalent to fix an angle of 90 degrees.The angle is measured between the wall and the interface in the phase 0.

```
boundary_conditions {
        cl_1 Paroi_FT_Disc symetrie
        cl_2 Paroi_FT_Disc constant Champ_Front_Uniforme 1 55
        cl_3 Paroi_FT_Disc constant Champ_Front_Fonc_xyz 1 30 +100*x
}
```

The optional **maillage** block is used to specify that we want a Gnuplot drawing of the initial mesh. There is only one keyword, *niveau_plot*, that is used only to define if a Gnuplot drawing is active (value 1) or not active (value -1). By default, skipping the block will produce non Gnuplot drawing. This option is to be used only in a debug process!
maillage niveau_plot *-1*

The **remaillage** block is used to specify the operations that are used to keep the solid interfaces in a proper condition. The following example has been successfully used for the free surface of the stirrer simulation:

```
remaillage {
        pas 0.000001
        nb_iter_remaillage 2
        critere_arete 0.35
        critere_remaillage 0.2
        pas_lissage 0.0000001
        nb_iter_barycentrage 5
        lissage_courbure_iterations 5
        lissage_courbure_coeff -0.2
      lissage_courbure_iterations_systematique N1
      lissage_courbure_iterations_si_remaillage N2
        relax_barycentrage 1
        facteur_longueur_ideale 1
        nb_iter_correction_volume 3
        seuil_dvolume_residuel 1e-12
        }
```

These parameters are described in the section following the section dedicated to the solid-fluid interface tracking equation (see paragraph 2.6.12.6).

lissage_courbure_iterations_systematique N1
lissage_courbure_iterations_si_remaillage N2

These keywords allow a finer control than the previous lissage_courbure_iterations keyword. N1 iterations are applied systematically at each timestep. N2 iterations are applied only if the local or the global remeshing effectively changes the lagrangian mesh connectivity. For proper DNS computation, N1 should be set to 0.

The **collisions** block is used to specify the operations that are used when a collision occurs between two parts of interfaces. When this occurs, it is necessary to build a new mesh that has locally a clear definition of what is inside and what is outside of the mesh.

```
collisions {
        active
        juric_pour_tout
        [ juric_local ] [ phase_continue 0 | 1 ]
        type_remaillage
            Juric { [Source_Isovaleur Indicatrice | Fonction_Distance] } | Thomas { [distance_interface_element_max N
] }
}
```

The **collisions** can either be **active** or **inactive**. If the **collisions** are **active** (highly recommended!), the keyword **juric_pour_tout** indicates that the Juric level-set reconstruction method will be used to re-create the new mesh after each coalescence or breakup. The next line (**type_remaillage**) is used to state whose field will be used for the level-set computation. Main option is **Juric**, a remeshing that is compatible with parallel computing. When using **Juric** level-set remeshing, the source field (**source_isovaleur**) that is used to compute the level-sets is then defined. It can be either the indicator function (**indicatrice**), a choice which is the default one and the most robust, or a geometrical distance computed from the mesh at the beginning of the time step (**fonction_distance**), a choice that may be more accurate in specific situations.
Type_remaillage Thomas is an enhancement of the Juric global remeshing algorithm designed to compensate for mass loss during remeshing. The mesh is always reconstructed with the indicator function (not with the distance function). After having reconstructed the mesh with the Juric algorithm, the difference between the old indicator function (before remeshing) and the new indicator function is computed. The differences occuring at a distance below or equal to *N* elements from the interface are summed up and used to move the interface in the normal direction. The displacement of the interface is such that the volume of each phase after displacement is equal to the volume of the phase before remeshing. *N* (default value 1) must be smaller than **n_iterations_distance** (suggested value: 2).
An alternate choice for the remeshing type (**type_remaillage**) is **collision_seq**, which is more complex and tries to sew the two meshes that have collided, once the collision zone has been removed. This algorithm does not work in parallel computation!
juric_local: triggers a new global remeshing algorithm to handle interface collisions: the connex component of interface where collisions occur is extracted and only this part will be remeshed with the global remeshing algorithm.
phase_continue: specifies which phase is the continuous phase that separates connex components (for suppression_sous_zone and juric_local, and maybe other algorithms that assume a dispersed flow pattern with isolated interfaces).

The **n_iterations_distance** keyword is used to specify the number or iterations requested for the smoothing process of computing the field corresponding to the signed distance to the interfaces and located at the center of the Eulerian elements. This smoothing is necessary when there are more Lagrangian nodes than Eulerian two-phase cells. The number of iterations *nb* is an integer (typical values 2,3,…).

The **iterations_correction_volume** keyword is used to specify the number or iterations requested for the correction process that can be used to keep the volume of the phases constant during the transport process. The number of iterations *nb* is an integer (typical value: 1).

The **volume_impose_phase_1** keyword is used to specify the volume of one phase to keep the volume of the phases constant during the remeshing process. It is an alternate solution to trouble in mass conservation. This option is mainly realistic when only one inclusion of phase 1 is present in the domain. In most other situations, the **iterations_correction_volume** keyword seems easier to justify. The volume *volume* to be keep is in *m*^{3} and should agree with initial condition.

Parcours_interface allows you to configure the algorithm that computes the surface mesh to volume mesh intersection. This algorithm has some serious trouble when the surface mesh points coincide with some faces of the volume mesh. Effects are visible on the indicator function, in VDF when a plane interface coincides with a volume mesh surface.
To overcome these problems, the keyword **correction_parcours_thomas** keyword can be used: it allows the algorithm to slightly move some mesh points. This algorithm, which is experimental and is NOT activated by default, triggers a correction that avoids some errors in the computation of the indicator function for surface meshes that exactly cross some eulerian mesh edges (strongly suggested !).

injecteur_interfaces FILENAME : Allows to create new interfaces at some given physical times. FILENAME file must contain ascii lines like this: 1.25 1 (x-0.2)2+(y-0.52)+(z-0.2)2-(0.052)
In this example a sphere of fluid phase 1 will be injected at time=1.25s). If the injected interface collides with an existing interface (eg, indicator function equal to injected phase at some point within the injected interface), injection is cancelled.

suppression_sous_zone SOUS_ZONE_NAME : As soon as an interface overlaps the specified region, the connex component of this interface is searched and destroyed and replaced by “phase_continue”.

interpolation_repere_local : Triggers a new transport algorithm for the interface: the velocity vector of lagrangian nodes is computed in the moving frame of reference of the center of each connex component, in such a way that relative displacements of nodes within a connex component of the lagrangian mesh are minimized, hence reducing the necessity of barycentering, smooting and local remeshing. Very efficient for bubbly flows.

2.6.12.4Solid-fluid interface tracking equation

To try to be clearer, the two types of interface tracking equations are explained separately even if it is actually coded in the same object (**Transport_Interfaces_FT_Disc**). In this section, the case of solid-fluid interface tracking equations is described with values that have already been used successfully. However, other set of choices can probably be tried, but the results are still unknown.

When used to mimic a solid-fluid immersed boundary (**equations_interfaces_vitesse_imposee** is declared in the Navier-Stokes equation), the interface tracking equation (**Transport_Interfaces_FT_Disc**) has to specify the speed of displacement of the interface, initial and boundary conditions (IBC, namely Immersed Boundary Condition), remeshing parameters:

```
agit
{
        methode_transport vitesse_imposee ../..
        Initial_Conditions { ../.. }
        boundary_conditions { ../.. }
        remaillage { ../.. }
    [ interpolation_champ_face base|lineaire { } ]
    [ nombre_facettes_retenues_par_cellule integer ]
    [ type_vitesse_imposee uniforme|analytique ]
    [ n_iterations_interpolation_ibc integer ]
    [ seuil_convergence_uzawa double ]
    [ nb_iteration_max_uzawa double ]
}
```

In the case of solid-fluid interfaces, the simplest choice in the block **methode_transport** is to use a value of **vitesse_imposee** (imposed speed of displacement) with an analytical formula, e.g:

```
methode_transport vitesse_imposee -(z-0.1)*40. 0. (x-0.1)*40.
```

In the above example, the solid interface is rotating around a Y-axis that is centered on {*0.1, 0.0, 0.1*} and has an angular velocity of 40 rad.s^{-1}.
As it is possible to compute the total fluid torque on an interface, an alternative and more physical choice may be to add an equation that will take into account a force coming, *e.g.*, from a magnetic coupling and a feedback force equivalent to the computed total fluid torque. This could lead to a richer description with a possible drift.

It is also possible to define the movement with a time-dependant law for the solid interface with the keywords **Position, Vitesse, Rotation** and **Derivee_rotation** (**verification_derivee** is a keyword to supress, which is not recommended unless necessary, the check of consistency between ug(t),vg(t),zg(t) and the time derivative of xg(t),yg(t),zg(t)).

```
Loi_horaire law
Read law
{
        Position            2|3     xg(t)   yg(t)   [zg(t)]
        Vitesse             2|3     ug(t)   vg(t)   [wg(t)]
        [Rotation           4|9     R00(t)  R01(t)  [R02(t)]
                            R10(t)  R11(t)  [R12(t)]
                            [R20(t) R21(t)  R22(t)]
        Derivee_rotation    4|9     dR00(t) dR01(t) [dR02(t)]
                                    dR10(t) dR11(t) [dR12(t)]
                                    [dR20(t)dR21(t) dR22(t)] ]
     [verification_derivee 0|1]
}
agit
{
        methode_transport loi_horaire law
        Initial_Conditions { ../.. }
        boundary_conditions { ../.. }
        remaillage { ../.. }
}
```

The keyword **Initial_Conditions** is used to define the shape of the solid interface through a mesh **fichier_geom**., or through the zero level-set of a **fonction**, e.g.:

```
Initial_Conditions { fonction -(((x-0.1))^2+((y-0.02)/0.3)^2+((z-0.1)/0.3)^2-(0.05^2)) }
```

Positive values of the function define the solid (where the velocity field is forced equal the "vitesse_imposee"), and negative values of the function define the fluid.
In the above example, the solid interface is an ellipsoid. The center of this ellipsoid is {*0.1, 0.02, 0.1*} and its half-axes are {*0.050, 0.015, 0.015*}.

Fichier_geom uses a line (in 2d calculations) or a surface (in 3d) mesh to create the initial condition for the interfaces. The mesh can be read in a .geom file (keyword **fichier_geom**) or in a domain previously created (keyword **nom_domaine**, see example below). The mesh must consist in ORIENTED segments or triangle, the normal vector of the segments or triangles (using the "right hand" rule) must point to "phase 1" (opposite to "phase 0"). Remember that for the **equations_interfaces_vitesse_imposee** keyword (immersed boundary condition), phase 1 is the solid where the velocity vector is forced, and phase 0 is the fluid. There is no check that the orientation is correct ! Incorrect orientation will produce a wrong computation of the indicator function near the interface. It is recommended to check the indicator function with the **lata_dump** keyword.

The mesh must NOT cross the boundaries of the computational domaine and it must be a CLOSED surface, but not necessarily a connex surface.

You must also tell the code where is "phase 0" and where is "phase 1" initially (phases are automatically updated as the interface moves during the computation). This can be done with the two keywords **point_phase** and **default_phase**. The code will search in the computational domain all the connex sets of mesh elements not traversed by the interface. For each set, the user must define to which phase this set must be initialized either with **point_phase**, or with **default_phase**.

*This is an experimental feature. Double check the result with the **lata_dump** keyword !*

```
Initial_Conditions {
        Fichier_geom {
                fichier_geom filename.geom | nom_domaine domaine_name
                [ point_phase 0 | 1     Xcoord  Ycoord  [Zcoord] ]
                ...
                [ default_phase 0 | 1 ]
                [ reverse_normal ]
                [ lata_dump lata_basemane ]
        }
}
```

fichier_geom *filename.geom* : Read the ascii file *filename.geom* (must be in .geom format) and use this mesh to build the interface. Use this method if the computation runs in parallel (since keyword **Read _file** will not work).

nom_domaine *domain_name* : *domain_name* must be a domain previously declared and filled in the .data file, usually with

```
Domaine CYLINDER
Lire_med CYLINDER filename.med
...
        Fichier_geom {
                nom_domaine CYLINDER  ...
```

point_phase 0 | 1 X*coord Ycoord [Zcoord]* : Tells the code that the given point (x,y,z) and the elements nearby is in the given phase (0 or 1). All elements in the same connex set of elements are given this phase number. The point must be located inside the computation domain, but not in a mesh cell crossed by the interface. You can specify several **point_phase** directives, but only one per connex set of elements.

default_phase 0 | 1 : With this keyword, the given phase will be used for all connex sets of elements that do not contain a **point_phase**. It is recommended to define a **point_phase** for the biggest connex set of elements and a **default_phase** for the other phase.

reverse_normal : You can easily build an oriented surface mesh (for example with Salome), but it is not easy to ensure that the normal vector points to "phase 1". If, once you created the mesh, you see that the normal vector points to "phase 0", this keyword will reverse all surface mesh elements to reverse the normal vector. You must check that, after correction, the normal vector is coherent with the point_phase and default_phase that you give.

lata_dump *lata_basename* : Writes a lata file containing the connex set of elements (each connex set has a different number) and the indicator function. The connex component field is equal to -1 in all cells that are crossed by the surface mesh.
Use this file to
- find coordinates where you should place **point_phase** directives,
- check that all volumes have the correct phase,
- check that the indicator function is correct in each connex set of elements and near the interface.

The **boundary_conditions** block is used to specify boundary conditions. In the case of solid interfaces, the indicator function is only important in the vicinity of these interfaces. The keyword adapted to compute the indicator function boundary conditions in that case is **Paroi_ft_disc**. Two kind of boundary condition may be applied : symmetry and contact angle fixed. Symmetry is equivalent to fix an angle of 90 degrees.The angle is measured between the wall and the interface in the phase 0.

```
boundary_conditions {
        cl_1 Paroi_FT_Disc symetrie
        cl_2 Paroi_FT_Disc constant Champ_Front_Uniforme 1 55
        cl_3 Paroi_FT_Disc constant Champ_Front_Fonc_xyz 1 30 +100*x
}
```

As for fluid-fluid interfaces (**equation_interfaces_proprietes_fluide**), the block **remaillage** is used to specify the operations that are used to keep the solid interfaces in a proper condition. The following example has been successfully used for the ellipsoid and its speed of displacement previously described.

```
remaillage
{
        pas 1e8
        nb_iter_remaillage 5
        critere_arete 0.5
        critere_remaillage 0.2
        pas_lissage -1
        nb_iter_barycentrage 5
        relax_barycentrage 1
        facteur_longueur_ideale 1
}
```

interpolation_champ_face base|lineaire { } : It is possible to compute the imposed velocity for the solid-fluid interface by direct affectation (**interpolation_scheme** would be set to **base**) or by multi-linear interpolation (**interpolation_scheme** would be set to **lineaire**). The default value is **base**.

nombre_facettes_retenues_par_cellule integer : Keyword to specify the default number (3) of facets per cell used to describe the geometry of the solid-solid interface. This number should be increased if the geometry of the solid-solid interface is complex in each cell (eulerian mesh too coarse for example).

type_vitesse_imposee uniforme|analytique : Useful only with **interpolation_champ_face** positioned to **lineaire**. Value of the keyword is **uniforme** (for an uniform solid-fluide interface's velocity, i.e. zero for instance) or **analytique** (for an analytic expression of the solid-fluide interface's velocity depending on the spatial coordinates). The default value is **uniforme**.

n_iterations_interpolation_ibc integer : Useful only with **interpolation_champ_face** positioned to **lineaire**. Set the value concerning the width of the region of the linear interpolation. For the Penalized Direct Forcing model, a value equals to 1 is enough.

seuil_convergence_uzawa double : Optional option to change the default value (10^{-8}) of the threshold convergence for the Uzawa algorithm if used in the Penalized Direct Forcing model. Sometime, the value should be decreased to insure a better convergence to force equality between sequential and parallel results.
nb_iteration_max_uzawa double : Optional option to change the default value (30) of the maximal number of iterations for the Uzawa algorithm if used in the Penalized Direct Forcing model. Sometime, the value should be increased to insure a better convergence to force equality between sequential and parallel results.

2.6.12.5Particle tracking equation

```
Navier_Stokes_FT_Disc eq_hydraulique
Associate pb eq_hydraulique
Transport_Marqueur_FT particles
Associate pb particles
Read  pb
{
  eq_hydraulique { … }
  particles
  {
    boundary_conditions  {  }
    Initial_Conditions {
          [ ensemble_points {   fichier filename   |   sous_zones N   name_ zone1 distribution … name_zoneN
distribution  } ]
         [ proprietes_particules { [ fichier filename | distribution { nb_particules nb vitesse u v [w] temperature
value masse_volumique  value  diametre  value } ] } ]
        [ t_debut_integration t_deb_integr ]
    }
    [ sources { … , … , … } ]
    [ injection {
        [ ensemble_points  { …  } ]
        [ proprietes_particules { … } ]
        [ t_debut_injection t_deb_inj ]
        [ dt_injection dt_inj ]
    } ]
    [ transformation_bulles {
        localisation N name_zone1 … name_zoneN
        diametre_min | beta_transfo diameter_size
        interface interface_name
        [ t_debut_transfo value ]
        [ dt_transfo value ]
    } ]
    [ methode_transport   vitesse_interpolee | vitesse_particules ]
    [ methode_couplage   suivi | one_way_coupling | two_way_coupling ]
    [ phase_marquee integer ]
    [ nb_iterations integer ]
    [ implicite 0|1 ]
    [ contribution_one_way 0|1 ]
  }
 liste_postraitements
 {
        Postraitement_ft_lata particles
        {
          …
            champs elements|sommets { densite_particules volume_particules }
            interfaces particles { champs sommets { vitesse volume diametre temperature masse_volumique }
        }
  }
}
```

Particles in a flow can be followed thanks to a **Transport_Marqueur_FT** equation.

The **boundary_conditions** { } block should be left empty cause the boundary conditions for this equation (and used to give the behaviour of the particles velocities at the boudaries) are the same than the boundary conditions of the hydraulic equation.

The initial conditions (**Initial_Conditions** keyword) define the initial state of the particules. The initial locations are defined with the keyword **ensemble_points**, either thanks to a file (keyword **fichier**) or with sub-zones (keyword **sous_zones**). In the first case, the format of the file *filename* is:

```
2
nb  dimension             where nb is the number of particles and dimension is 2 or 3
nb_values                 where nb_values equals to nb*dimension
x1 y1 [z1]                where xi, yi and zi in 3D are the coordinates of ith particle
..
xnb ynb [znb]
```

In the case of a location per sub-zones, the distribution of the particles can be randomized with nb the number of particles:
name_zone **aleatoire** nb
Or uniform with nbX, nbY and nbZ the number of particles in each direction :
name_zone **uniforme** nbX nbY [nbZ]

The **proprietes_particules** gives the particles properties. If the properties are non uniform, they can be read in a file with the keyword **fichier.** The format of the *filename* file is:

```
2
nb dimension              where nb is the number of particles and dimension is 2 or 3
nb_values                 where nb_values equals to nb*dimension
u1 v1 [w1]                where ui, vi and wi in 3D are the initial velocity of ith particle
…
unb vnb [wnb]
2
nb 1
nb                        where nb is the number of particles
T1                        where Ti is the initial temperature of ith particle
…
Tnb
2
nb 1
nb                        where nb is the number of particles
Rho1             where Rhoi is the initial density of ith particle
…
Rhonb
2
nb 1
nb                        where nb is the number of particles
D1                        where Di is the initial diameter of ith particle
…
Dnb
```

In the case of uniform properties for each particles, they can be given by the keyword **distribution** with:
nb_particules nb : the number of particles
vitesse u v [w] : the velocity of all the particles
temperature value : the value of the temperature for all the particles
masse_volumique value : the value of the density for all the particles
diametre value : the value of the diameter for all the particles

The beginning time for the calculation of the particles trajectories is given by the keyword **t_debut_integration.** By default, it is the value given in the time scheme with the keyword **tinit.** Before this time t_deb_integr, the particles do not move.

The **sources** terms available for this equation are: **trainee** (drag effect), **flottabilite** (buoyancy effect), **masse_ajoutee** (weight added effect), **portance** (lift effect). The last one is not available yet.

The keyword **injection** can be used to inject periodically during the calculation some other particles. The syntax for **ensemble_points** and **proprietes_particles** is the same than the initial conditions for the particles. The keyword **t_debut_injection** give the injection initial time (by default, given by **t_debut_integration**) and **dt_injection** gives the injection time period (by default given by **dt_min**).

The keyword **transformation_bulles** will activate the transformation of an inclusion (small bubbles) into a particle. **localisation** gives the sub-zones (N number of sub-zones and their names) where the transformation may happen. The diameter size for the inclusion transformation is given by either **diameter_min** option, in this case the inclusion will be suppressed for a diameter less than *diameter_size*, either by the **beta_transfo** option, in this case the inclusion will be suppressed for a diameter less than *diameter_size**cell_volume (cell_volume is the volume of the cell containing the inclusion). **interface** specifies the name of the inclusion interface and **t_debut_transfo** is the beginning time for the inclusion transformation operation (by default, it is **t_debut_integr** value) and **dt_transfo** is the period transformation (by default, it is **dt_min** value). In a two phase flow calculation, the particles will be suppressed when entring into the non marked phase (see below):

Other options for the particles:
methode_transport : Kind of transport method for the particles. With **vitesse_interpolee**, the velocity of the particles is the velocity a fluid interpolation velocity (option by default). With **vitesse_particules**, the velocity of the particules is governed by the resolution of a momentum equation for the particles.
methode_couplage : Way of coupling between the fluid and the particles. By default, (keyword **suivi**), there is no interaction between both. With **one_way_coupling** keyword, the fluid act on the particles. With **two_way_coupling** keyword, besides, particles act on the fluid.
phase_marquee integer : Phase number giving the marked phase, where the particles are located (when they leave this phase, they are suppressed). By default, for a the two phase fluide, the particles are supposed to be into the phase 0 (liquid).
nb_iterations integer : Number of sub-timesteps to solve the momentum equation for the particles (1 per default).
implicite 0|1 : Impliciting (1) or not (0) the time scheme when weight added source term is used in the momentum equation
contribution_one_way 0|1: Activate (1, default) or not (0) the fluid forces on the particles when **one_way_coupling** or **two_way_coupling** coupling method is used.

To post process the location of the particles in the flow, either a volume field for density particles (**densite_particules**) and volume particles (**volume_particles**) or point mesh (**interfaces**) visualization can be used but only with the LATA format (and VisIt) for the last one.

2.6.12.6Remeshing

The **remaillage** block only contains parameter's values. These parameters are also described in the document (in French) written by C. Poyet: *Paramètres de transport et de remaillage de l'interface Front-Tracking-Discontinu Version 1.4.7 et patchs, Août 2005.*

remaillage *{*
    pas *../..*
    pas_lissage *../..*
    nb_iter_remaillage *../..*
    nb_iter_barycentrage *../..*
    relax_barycentrage *../..*
    critere_arete *../..*
    critere_remaillage *../..*
    impr *../..*
    facteur_longueur_ideale *../..*
    nb_iter_correction_volume *../..*
    seuil_dvolume_residuel *../..*
    lissage_courbure_coeff *../..*
    lissage_courbure_iterations *../..*
    critere_longueur_fixe *../..*
}

An example of values of these parameters is given in the section "The fluid-fluid interface tracking equation", in the paragraph dedicated to the remeshing keyword (**remaillage**).

The keyword **pas** has default value *-1.*; when **pas** is set to a negative value there is no remeshing. It is the time step in second (physical time) between two operations of remeshing.

The keyword **pas_lissage** has a default value set to *-1.*; when **pas_lissage** is set to a negative value there is no smoothing of mesh. It is the time step in second (physical time) between two operations of smoothing of the mesh.

The keyword **nb_iter_remaillage** has a default value set to *0*; when **nb_iter_remaillage** is set to the zero value there is no remeshing. It is the number of iterations performed during a remeshing process.

The keyword **nb_iter_barycentrage** has a default value set to *0*; when **nb_iter_barycentrage** is set to the zero value there is no operation of "barycentrage". The "barycentrage" operation consists in moving each node of the mesh tangentially to the mesh surface and in a direction that let it closer the center of gravity of its neighbors. If **relax_barycentrage** is set to *1*, the node is move to the center of gravity. For values lower than unity, the motion is limited to the corresponding fraction. The parameter **nb_iter_barycentrage** is the number of iteration of these node displacements.

The keyword **relax_barycentrage** has a default value set to *0*; when **relax_barycentrage** is set to the zero value there is no motion of the nodes. When *0* < **relax_barycentrage** ≤ *1*, this parameter provides the relaxation ratio to be used in the "barycentrage" operation described for the keyword **nb_iter_barycentrage**.

The keyword **critere_arete** is used to compute two sub-criteria: the minimum and the maximum edge length ratios used in the process of obtaining edges of length close to **critere_longueur_fixe**. Their respective values are set to $(1\text{-}\textbf{critere\_arete})^2$ and $(1+\textbf{critere\_arete})^2$. The default values of the minimum and the maximum are set respectively to *0.5* and *1.5*. When an edge is longer than $\textbf{critere\_longueur\_fixe}*(1+\textbf{critere\_arete})^2$, the edge is cut into two pieces; when its length is smaller than $\textbf{critere\_longueur\_fixe}*(1\text{-}\textbf{critere\_arete})^2$, this edge has to be suppressed.

The keyword **critere_remaillage** was previously used to compute two sub-criteria: the minimum and the maximum length used in the process of remeshing. Their respective values are set to $(1\text{-}\textbf{critere\_remaillage})^2$ and $(1+\textbf{critere\_remaillage})^2$. The default values of the minimum and the maximum are set respectively to *0.2* and *1.7*. There are currently not used in data files.

The keyword **impr** is followed by a value that specify the printing time period given. The default value is *-1*, which means no printing.

The keyword **facteur_longueur_ideale** is used to set a ratio between edge length and the cube root of volume cell for the remeshing process. The default value is *1.0*.

The keyword **nb_iter_correction_volume** give the maximum number of iterations to be performed trying to satisfy the criterion **seuil_dvolume_residuel**. The default value is *0*, which means no iteration.

The keyword **seuil_dvolume_residuel** give the error volume (in m^3) that is accepted to stop the iterations performed to keep the volume constant during the remeshing process. The default value is *0.0*.

The keyword **lissage_courbure_coeff** is used to specify the diffusion coefficient used in the diffusion process of the curvature in the curvature smoothing process with a time step. The default value is *0.05*. That value usually provides a stable process. Too small values do not stabilize enough the interface, especially with several Lagrangian nodes per Eulerian cell. Too high values induce an additional macroscopic smoothing of the interface that should physically come from the surface tension and not from this numerical smoothing.

The keyword **lissage_courbure_iterations** is used to specify the number of iterations to perform the curvature smoothing process. The default value is *1*.

The keyword **critere_longueur_fixe** is used to specify the ideal edge length for a remeshing process. The default value is *-1.*, which means that the remeshing does not try to have all edge lengths to tend towards a given value.

2.6.12.7Concentration equation on a two phase-flow with interface tracking

The **Convection_diffusion_concentration_ft_disc** allows to take into account the interface and prevents the scalar from diffusing through the interface.

```
Probleme_FT_Disc_gen   pb
Convection_diffusion_concentration_FT_disc  concentration_equation
Associate  pb concentration_equation
…
Read   pb
{
   …
   concentration_equation
  {
     ... (parameters for the classic Convection_Diffusion_Concentration, see 2.6.5)
      equation_interface eq_name
      phase 0 | 1
      option RIEN | RAMASSE_MIETTES_SIMPLE
      constante_cinetique  VAL
      equations_source_chimie N EQ_NAME_1 ... EQ_NAME_N
      constante_cinetique_nu_t VAL
       equation_nu_t  EQ_NAME
```

```
        zone_sortie  SOUS_ZONE_NAME
        [ Sources {  Source_Constituant_Vortex { … } } ]
    }
}
```

equation_interface: this is the name of the interface tracking equation to watch. The scalar will not diffuse through the interface of this equation.

phase 0|1: tells whether the scalar must be confined in phase 0 or in phase 1

option: Experimental features used to prevent the concentration to leak through the interface between phases due to numerical diffusion.

RIEN: do nothing

RAMASSE_MIETTES_SIMPLE: at each timestep, this algorithm takes all the mass located in the opposite phase and spreads it uniformly in the given phase.

constante_cinetique VAL : experimental, documentation to be written

equations_source_chimie N EQ_NAME_1 ... EQ_NAME_N : experimental, documentation to be written

constante_cinetique_nu_t VAL : experimental, documentation to be written

equation_nu_t EQ_NAME : experimental, documentation to be written

zone_sortie SOUS_ZONE_NAME : artificial source term that drops the concentration to zero within the specified sub-zone (see file _bilan.out below)

Available source term for Convection_Diffusion_Concentration_FT_Disc :

```
Source_Constituant_Vortex
{
 rayon_spot RADIUS
 integrale INTEGRAL
 debit FLOW
 senseur_interface  {
   equation_interface EQ_NAME
   segment_senseur_1 DIM CoordX CoordY [ CoordZ ]
   segment_senseur_2 DIM CoordX CoordY [ CoordZ ]
   nb_points_tests N_POINTS
 }
 delta_spot DIM Dx Dy [ Dz ]
}
```

This is a dynamic source term of concentration designed to simulate injection at a free surface: injects a gaussian spot of concentration of given INTEGRAL value (INTEGRAL is a flux, the integral of the source term injected during a timestep is INTEGRAL multiplied by the time step). RADIUS characterizes the radius of the gaussian spot. FLOW is the fluid flow injected in the Navier-Stokes equation (equal to the integral of a source of divergence velocity) in m3/s. **senseur_interface** describes a sensor to tell where the injection will take place. EQ_NAME is the name of an interface tracking equation (describes the free surface). **segment_senseur_1** and **segment_senseur_2** describe a segment that crosses the interface. The position of the free surface will be checked on N_POINTS points on this segment starting from **segment_senseur_1**. If a phase interface is found, the position of the injected spot will be located at this position, plus the **delta_spot** vector. For proper conservation of the injected species, the **delta_spot** vector should direct the injection at least at RADIUS distance below the free surface.

Notice: the source term will by taken into account by the **Navier_Stokes_FT_Disc** equation to modify the divergence of the velocity at the injection point only if **equation_concentration_source_vortex** N EQ1 ... EQN keyword is also added to the Navier_Stokes equation. EQ1..EQN are the names of the concentration equations containing source terms of velocity divergence.

Content of the file *EQ_NAME_bilan.out* created when using this equation : every timestep, it writes one line:
column 1: time
column 2: integral of concentration in phase 0
column 3: integral of concentration in phase 1
column 4: integral of zone_sortie source term during the timestep
column 5: equal to column 4 divided by timestep (eg flux of concentration)

2.6.12.8Temperature equation on a single phase flow with interface tracking

The **Convection_diffusion_temperature** is the keyword to add the temperature equation for a single phase flow with solid-fluid interfaces for example.

```
Probleme_FT_Disc_gen  pb
Convection_diffusion_temperature   temperature_equation
Associate  pb  temperature_equation
…
```

```
Read  pb {
  …
  temperature_equation
  {
    ... (parameters for the classic Convection_Diffusion_Temperature)
          [ penalisation_L2_FTD {   solid_fluid_interface_name1 1 specified_temperature
                                     solid_fluid_interface_name2 1 specified_temperature … } ]
  }
}
```

The optional keyword **penalisation_L2_FTD** is to activate or not (the default is Direct Forcing method) the Penalized Direct Forcing method to impose the specified temperature on the solid-fluid interface.

2.6.12.9Temperature equation on a two-phase flow with interface tracking

The **Convection_diffusion_temperature_ft_disc** is the keyword (**Convection_diffusion_temperature** will return an error) to add the temperature equation for one phase of a front tracking calculation (temperature values for the other phase will not be realistic). A model with two temperature equations (one for each phase will be introduced in future releases). Futhermore, the temperature of the other phase will be set to the saturation temperature, and is a constant (0°C) for the moment.

```
Probleme_FT_Disc_gen  pb
Convection_diffusion_temperature_FT_Disc   temperature_equation
Associate  pb  temperature_equation
…
Read  pb {
  …
  temperature_equation
  {
    ... (parameters for the classic Convection_Diffusion_Temperature)
   equation_navier_stokes  name
   equation_interface  name
   phase 0 | 1
   stencil_width  N
   [ maintien_temperature  SOUS_ZONE_NAME VALUE ]
  }
}
```

equation_navier_stokes name : The name of the Navier Stokes equation of the problem should be given.
equation_interface name : The name of the interface equation should be given.
phase 0 | 1 : Phase in which the temperature equation will be solved. The temperature, which may be postprocessed with the keyword **temperature_EquationName**, in the orther phase may be negative: the code only computes the temperature field in the specified phase. The other phase is supposed to physically stay at saturation temperature. The code uses a ghost fluid numerical method to work on a smooth temperature field at the interface. In the opposite phase (1-X) the temperature will therefore be extrapolated in the vicinity of the interface and have the opposite sign, saturation temperature is zero by convention).
stencil_witdth N : distance in mesh elements over which the temperature field should be extrapolated in the opposite phase.
maintien_temperature SOUS_ZONE_NAME VALUE : experimental, this acts as a dynamic source term that heats or cools the fluid to maintain the average temperature to VALUE within the specified region. At this time, this is done by multiplying the temperature within the SOUS_ZONE by an appropriate uniform value at each timestep. This feature might be implemented in a separate source term in the future.

2.6.12.10Post processing

The block **liste_postraitements** defines the output files to be written during the computation. The output format is **lata** in order to use OpenDX to draw the results. The block **liste_postraitements** can be divided in one or several sub-blocks that can be written at different frequencies and in different directories. Attention! The directory **lata** used in this example should be created before running the computation or the **lata** files will be lost!

The general structure of the **liste_postraitements** block is:

```
liste_postraitements
{
        Postraitement_ft_lata post1 { ../.. }
        Postraitement_ft_lata post2 { ../.. }
...
}
```

Each **Postraitement_ft_lata** has the same general structure:

Postraitement_ft_lata *post1*
{
	dt_post *string*
	[**nom_fichier** *lata/post*]
	[**format** *format*]
	[**fichiers_multiples**]
	champs *location { fields_list }*
	interfaces *interf {* **champs** *location { fields_list } }*
	skip_header
	print
}

The option **dt_post** is the same than the **Champs** option **dt_post** defined at the paragraph 2.19.4.

The optional keyword **nom_fichier** is used to specify the sub-directory and the root of all the post-processing files. The default value is the name of the data file.
In the example, the code will write all files in a subdirectory named *lata* that should be in the directory of the data file. All files will have the prefix *post1*. For instance, the initial interface will be post-processed in the file post1.lata.INDICATRICE_INTERF.I.ELEM.DOM.pb.0.000000, the interface at the first time step could be post1.lata.INDICATRICE_INTERF.I.ELEM.DOM.pb.0.010000, and so on.

The optional keyword **format** can be followed (*format*) by either **binaire** or **ascii**. The first choice is more compact and is actually dedicated to using OpenDX, whereas the second one can be browsed with any textbrowser. The default value is **ascii**.

The optional keyword **fichiers_multiples** is used in parallel computing to split the post-processing into one file for each processor. When this keyword is not present (default), a single file is constructed by collecting data from all the cpus.

The keyword **champs** can be followed (*location*) by either **sommets**, **elements** or **faces**, and a list of fields (*fields_list*) to be post-processed in these positions. Of course, this means that the field values are to be post-processed respectively at the summits, the center of the volume elements or the faces of the elements. When the field is not stored in these positions, the post-processed values are interpolated within the closest neighbors. Example, for the pressure and velocity: **champs sommets** *{* **pression vitesse** *}*

A special case concerns the indicator functions. To be able to deal with data files that involve more than one interface, a suffix is used to specify which couple {interface, indicator function} is concerned. The suffix is the name of the interface (**Transport_Interfaces_FT_Disc**) declared in the problem description. The suffix is concatenated with the **indicatrice_**. E.g:

champs elements
{
	indicatrice_*interf*
	indicatrice_*agit*
	pression
	concentration
}

The keyword **interfaces** is followed by the name of an interface (**Transport_Interfaces_FT_Disc**). In the block under brackets, are defined the fields to be post-processed on the interfaces. E.g:

interfaces *interf {* **champs sommets** *{* **courbure** *} }*

From the structure of the code, the *location* of the post-processing on interfaces can either be on **sommets** (nodes of the Lagrangian mesh), or on **elements** (center of the triangles of the Lagrangian mesh).

Today, these features are still limited to two physical quantities that can be processed on the **sommets** of the interfaces: **courbure** (curvature) and **vitesse** (speed of displacement). Additional integer parameters may be post-processed, mainly for debugging or to illustrate the parallel computing by domain decomposition: **pe** (index of the processor that is responsible of this part of the interface at the current time step), **pe_local** (index of the processor that is currently writing the information on this part of the interface at the current time step) and **numero** (index of the part of the interface in the table of the current processor).

The keyword **skip_header** is used to prevent the post-processing to write header in each file. This can be used in case of restart of a computation or when the post-processing is written in a file created by another post-processing.

The keyword **print** is used to enable the printing of post-processing comments in the *err* file.

The following example has been used to deal with two different interfaces (*interf* and *agit*) in the stirrer and free surface simulation. *interf* corresponds to the free surface whereas *agit* corresponds to the stirrer solid-fluid interface:

```
liste_postraitements
{
        Postraitement_ft_lata post1
        {
                dt_post 0.01
                nom_fichier lata/post1
                format binaire
                print
                champs sommets { vitesse }
                champs elements {
                        distance_interface_elem_interf
                        distance_interface_elem_agit
                        indicatrice_interf
                        concentration }
                interfaces interf { champs sommets { courbure } }
        }
        Postraitement_ft_lata post2
        {
                dt_post 0.01
                nom_fichier lata/post2
                format binaire
                print
                interfaces agit { champs sommets { pe } }
        }
}
```

2.6.13PHASE FIELD PROBLEM

Complete description of the Phase Field model for incompressible and immiscible fluids can be found into this PDF file: $TRUST_ROOT/doc/TRUST/phase_field_non_miscible_manuel.pdf

```
Read  pb {
    Navier_Stokes_Phase_Field {
        Solveur_Pression …
        Convection { … }
        Approximation_de_Boussinesq oui|non
        Viscosite_dynamique_constante oui|non
        Diffusion { … }
        Sources { Source_Qdm_Phase_Field { Forme_du_terme_source integer }
        Gravite n x y [z]
        Initial_Conditions { … }
        boundary_conditions { … }
    }
    Convection_Diffusion_Phase_Field {
       Convection { … }
       Diffusion { … }
       Sources { Source_Con_Phase_Field {
                      Temps_d_affichage value
                      Alpha value Beta value
                      Kappa value Kappa_variable oui|non
                      Moyenne_de_kappa      string
                      Multiplicateur_de_kappa    value
                      Couplage_NS_CH          string
                      Implicitation_CH         oui|non
                      Gmres_non_lineaire        oui|non
                      Seuil_cv_iterations_ptfixe value Seuil_residu_ptfixe value
                      Seuil_residu_gmresnl       value
                      Dimension_espace_de_krylov  integer
                      Nb_iterations_gmresnl      integer
                      Residu_min_gmresnl  value Residu_max_gmresnl value
                    }
               }
               mu_1  value  mu_2  value rho_1  value  rho_2  value
               Potentiel_chimique_generalise     string
               boundary_conditions { … }
               Initial_Conditions  { Concentration … }
    }
    Post_processing { Fields dt_post value {
      Concentration
      Potentiel_chimique_generalise  … } }
}
```

Navier_Stokes_Phase_Field : Keyword to define the Navier Stokes equation for the Phase Field problem.

Approximation_de_Boussinesq oui|non : To use or not the Boussinesq approximation.
Viscosite_dynamique_constante oui|non : To use or not a viscosity which will depends on "concentration" C (in fact, C is the unknown of Cahn-Hilliard equation).
Gravite n x y [z] : Keyword to define gravity in the case Boussinesq approximation is not used.
Source_Qdm_Phase_Field { **forme_du_terme_source** integer } : Keyword to define the capillary force into the Navier Stokes equation for the Phase Field problem. The kind of the source term is given by integer (1,2,3 or 4).

Convection_Diffusion_Phase_Field: Keyword to define the Cahn-Hilliard equation of the Phase Field problem. The unknown of this equation is the "concentration" C.
Source_Con_Phase_Field : Keyword to define the source term of the Cahn-Hilliard equation.
Temps_d_affichage value : Time during the caracteristics of the problem are shown before calculation.
Alpha value : To define the internal capillary coefficient α
Beta value : To define the parameter β of the model
Kappa value : To define the mobility coefficient κ_0
Kappa_variable oui|non : To define a mobility which depends on "concentration" C
Moyenne_de_kappa string : To define how mobility κ is calculated on faces of the mesh according to cell-centered values (string is arithmetique|harmonique|geometrique)
Multiplicateur_de_kappa value : To define the the parameter a of the mobility expression when mobility depends on C.
Couplage_NS_CH string : Evaluating time choosen for the term source calculation into the Navier Stokes equation (string is mutilde(n+1/2)|mutilde(n), in order to be conservative, the first choice seems better)
Implicitation_CH oui|non : To define if the Cahn-Hilliard will be solved using a implicit algorithm or not
Gmres_non_lineaire oui|non : To define the algorithm to solve Cahn-Hilliard equation:(oui: Newton-Krylov method, non: fixed point method)

To define options of the fixed point method:
Seuil_cv_iterations_ptfixe value : the convergence threshold
Seuil_residu_ptfixe value : the threshold for the matrix inversion used in the method

To define options of the Newton-Krylov method:
Seuil_residu_gmresnl value : the convergence threshold
Dimension_espace_de_krylov integer : the vector numbers used in the method
Nb_iterations_gmresnl integer : the maximal iterations
Residu_min_gmresnl value : the minimal convergence threshold
Residu_max_gmresnl value : the maximal convergence threshold

mu_1 value : To define the dynamic viscosity of the first phase
mu_2 value : To define the dynamic viscosity of the second phase
rho_1 value : To define the density of the first phase
rho_2 value : To define the density of the second phase
Potentiel_chimique_generalise string : To define (string set to avec_energie_cinetique) or not (string set to sans_energie_cinetique) if the Cahn-Hilliard equation contains the cinetic energy term

Concentration : Keyword to postprocess the unknown C of the Cahn-Hilliard equation
Potentiel_chimique_generalise : Keyword to postprocess the field mutilde

2.6.14PROBLEM WITH PASSIVE SCALARS

```
Problem pb
Read  pb {
        Navier_Stokes_Standard { … }
        Convection_Diffusion_Temperature
        {
                Convection { … }
                Diffusion { … }
                Sources { … }
                boundary_conditions { … }
                Initial_Conditions { temperature … }
        }
        Equations_Scalaires_Passifs
        {
                Convection_Diffusion_Temperature
                {
                        Convection { … }
                        Diffusion { … }
                        Sources { … }
                        boundary_conditions { … }
                        Initial_Conditions { temperature0 … }
                }
                Convection_Diffusion_Temperature
                {
                        Convection { … }
                        Diffusion { … }
                        Sources { … }
                        boundary_conditions { … }
                        Initial_Conditions { temperature1 … }
                }
                Convection_Diffusion_Temperature
                {
                        Convection { … }
                        Diffusion { … }
                        Sources { … }
                        boundary_conditions { … }
                        Initial_Conditions { temperature2 … }
                }
                ….
        }
        Post_processing {
                Champs dt_post value
                {
                        Temperature
                        Temperature0
                        Temperature1
                        Temperature2
                        …
                }
        }
}
```

Problem is a keyword to create a classical problem with a scalar transport equation (e.g: temperature or concentration) and an additional set of passive scalars (e.g: temperature or concentration) equations. The list of keywords available for *Problem* are :

Pb_Hydraulique_Concentration_Scalaires_Passifs
Pb_Hydraulique_Concentration_Turbulent_Scalaires_Passifs
Pb_Thermohydraulique_Scalaires_Passifs
Pb_Thermohydraulique_Turbulent_Scalaires_Passifs
Pb_Thermohydraulique_Concentration_Scalaires_Passifs
Pb_Thermohydraulique_Concentration_Turbulent_Scalaires_Passifs
Pb_Thermohydraulique_QC_fraction_massique*
Pb_Thermohydraulique_Turbulent_QC_fraction_massique*
…
***** For these two last problems, hydraulic and energy equations are solved and a list of passive scalar equations may be added.

The unknowns of the passive scalar equation number N are named **temperatureN** or **concentrationN** or **fraction_massiqueN**. This keyword is used to define initial conditions and the post processing fields. This kind of problem is very useful to test in only one data file (and then only one calculation) different schemes or different boundary conditions for the scalar transport equation.

2.6.15 PROBLEM WITH TRANSPORT OF CHEMICAL SPECIES

```
Probleme_FT_Disc_gen pb
Chimie model
Read  model {
    Reactions {
           {
                reactifs formulae
                produits formulae
                constante_taux_reaction double
            [ contre_reaction double ]
                coefficients_activites { Specie_1,1 double  … Specie_N1,1 double }
                enthalpie_reaction 0
           } ,
           ....
           {
                reactifs formulae
                produits formulae
                constante_taux_reaction double
                [ contre_reaction double ]
                coefficients_activites { Specie_1,R double  … Specie_NR,R double }
                enthalpie_reaction 0
           }
    }
    [modele_micro_melange 0|1]
    [constante_modele_micro_melange double]
    [espece_en_competition_micro_melange specie]
}
Associate pb model
Convection_Diffusion_Concentration  Specie1
......
Convection_Diffusion_Concentration  SpecieN
Read  pb {
        …
           Specie1
         {
                       diffusion { }
                       convection { … }
                       nom_inconnue Specie1
                       boundary_conditions { … }
                       Initial_Conditions { Specie1 … }
                       masse_molaire double
           }
         …
        SpecieN {
           ….
         }
}
```

The keyword **Chimie** is used to define the list of reactions thanks to the keyword **Reactions**. In each reaction r (r=1 to R) where N_r species are used, the following keywords are defined:

-the reactant species (**reactifs** keyword) and their stoichiometric coefficients defined in a formulae, for example 6*Hp+5*Im+IO3m where Hp, Im and IO3m are 3 species

-the product species (**produits** keyword) and their stoichiometric coefficients defined in a formulae, for example 3*I2+3*H2O where I2 and H20 are 2 other species

-the forward rate constant for the reaction (**constante_taux_reaction** keyword in $[s^{-1}]$) $k_{f,r}$

-the optional equilibirum constant $K_r=k_{b,r}/k_{f,r}$ (with $k_{b,r}$ the backward rate constant for the reaction $[s^{-1}]$), defined by the **contre_reaction** keyword. This should be used for a reversible reaction (by default, it is a non-reversible reaction with $K_r=0$)

-the rate exponent $A_{j,r}$ for each specie j in the reaction (**coefficients_activites** keyword)

-the enthalpy generated by the reaction (**enthalpie_reaction** keyword). For the moment, only 0 for **enthalpie_reaction** value is possible, that means there is no heat source term into the energy equation caused by the species reaction.

The kinetic of the reaction r is defined by: $\omega_r = k_{f,r}\left(\prod_{j=1}^{N_r}[C_{j,r}]^{A_{j,r}} - K_r\prod_{j=1}^{N_r}[C_{j,r}]^{A_{j,r}}\right)$ where $C_{j,r}$ is the species molar concentration of the reaction.

A turbulent micromixing model can also be activated to change the kinetic, several optional keywords are available :

modele_micro_melange 1 : activate the model (by default 0)

constante_modele_micro_melange double : specify the constant of the model

espece_en_competition_micro_melange specie : keyword to exclude a specie from

To know more on this micromixing model which, one will look at the $TRUST_ROOT/src/ThHyd/Chimie/Chimie.cpp source file.

The transport of the chemical species are then specified by the **Convection_Diffusion_Concentration** keyword for the N species. **Nom_inconnue** defines the name of the field concentration and **masse_molaire** the molar mass for the transported specie.

Example:

```
Read  la_chimie
{
        modele_micro_melange 1
        constante_modele_micro_melange 1e-5
        reactions
        {
                {
                        reactifs H2BO3m+Hp

                        produits H3BO3
                        constante_taux_reaction 1.e11
                        coefficients_activites { H2BO3m 1 Hp 1 }
                        enthalpie_reaction 0.
                } ,
                {
                        reactifs 6*Hp+5*Im+IO3m
                        produits 3*I2+3*H2O
                        constante_taux_reaction 5.8e7
                        coefficients_activites { Hp 2 Im 2 IO3m 1 }
                        enthalpie_reaction 0.
                } ,
                {
                        reactifs Im+I2
                        produits I3m
                        constante_taux_reaction 5.6e9
                        contre_reaction 786.
                        coefficients_activites { Im 1 I2 1 I3m 1 }
                        enthalpie_reaction 0.
                }
        }
}
```

2.7 COUPLINGS

Probleme_Couple *nom_pb_couple*

This instruction causes a **Probleme_Couple** type object to be created. This type of object has an associated problem list, that is, the coupling of n problems among them may be processed. Coupling between these problems is carried out explicitly via conditions at particular contact limits.

Each problem may be associated either with the **Associate** keyword or with the **Read /groupes** keywords:
Probleme_Couple pbc
Associate pbc pb1
Associate pbc pb2
Associate pbc pb3
Associate pbc pb4
Or:
Probleme_Couple pbc
Read pbc { **groupes** { { pb1 , pb2 } , { pb3 , pb4 } } }

The difference is that in the first case, the four problems exchange values then calculate their timestep, rather in the second case, the same strategy is used for all the problems listed inside one group, but the second group of problem exchange values with the first group of problems after the first group did its timestep. So, the first case may then also be written like this:
Probleme_Couple pbc
Read pbc { **groupes** { { pb1 , pb2 , pb3 , pb4 } } }

There is a physical medium per problem (however, the same physical medium could be common to several problems). Each problem is resolved in a domain.

Warning : Presently, coupling requires coincident meshes. In case of non-coincident meshes, boundary condition "**paroi_contact**" in VEF returns error message (see **paroi_contact** for correcting procedure).

2.7.1THERMOHYDRAULIC RADIATION COUPLING

```
Pb_Thermohydraulique Pb_fluide
Pb_Conduction Pb_solide
Pb_Couple_Rayonnement Pb_couple
…
Modele_Rayonnement_Milieu_Transparent mod
Read  mod {
      nom_pb_rayonnant        problem_name
      fichier_fij             file_name
      fichier_face_rayo       file_name
      [fichier_matrice | fichier_matrice_binaire  file_name]
}

Associate pb_couple mod

Read  pb_fluide { … }
Read  pb_solide { … }
…
Solve pb_couple
```

Pb_Couple_Rayonnement: This keyword is used to define a problem coupling several other problems to which radiation coupling is added.

Modele_Rayonnement_Milieu_Transparent *mod***:** This refers to the keyword and name of the wall thermal radiation model for a transparent gas and resolving a radiation-conduction-thermohydraulics coupled problem in VDF or VEF.

Read *mod*: Keyword to read the *mod* radiation model. The syntax of this radiation model has changed for the 1.5.6 version. Previous syntax is still recognized. Here is the new one:

nom_pb_rayonnant problem_name : problem_name is the name of the radiating fluid problem

fichier_fij *file_name* : file_name is the name of the file which contains the shape factor matrix between all the faces.

fichier_face_rayo *file_name* : file_name is the name of the file which contains the radiating faces characteristics (area, emission value ...)

fichier_matrice|fichier_matrice_binaire *file_name* : file_name is the name of the ASCII (or binary) file which contains the inverted shape factor matrix. It is an optional keyword, if not defined, the inverted shape factor matrix will be calculated and written in a file.

The two first files can be generated by a preprocessor, they allow the radiating face characteristics to be entered (set of faces considered to be uniform with respect to radiation for emission value, flux, etc.) and the form factors for these various faces. These files have the following format:

File on radiating faces:

```
N M                            -> N nombre de faces rayonnantes (=bords) et
                               (N is the number of radiating faces (=edges) and
                            -> M nombre de faces rayonnantes a emissivitée non nulle
                                        M equals the number of non-zero emission radiating faces
Nom(i) S(i) E(i)             -> Nom du bord i, surface du bord i, valeur de
                               (Name of the edge i, surface area of the edge i)
                               -> l'émissivité (comprise entre 0 et 1) (emission value (between 0 an 1))
Exemple:
13 4
Gauche  50.0 0.0
Droit1  50.0 0.5
Bas     10.0 0.0
Haut    10.0 0.0
Arriere  5.0 0.0
Avant    5.0 0.0
Droit2  30.0 0.5
Bas1    40.0 0.0
Haut1   20.0 0.0
Avant1  20.0 0.0
Arriere1 20.0 0.0
Entree  20.0 0.5
Sortie  20.0 0.5
```

File on form factors:

```
N      -> Nombre de faces rayonnantes (Number of radiating faces)
Fij    -> Matrice des facteurs de formes avec i,j entre 1 et N (Matrix of form factors where i, j between 1 and N)
Example:
13
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.24 0.20 0.10 0.10 0.10 0.10 0.16
0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.40 0.00 0.00 0.00 0.00 0.00 0.20 0.10 0.10 0.10 0.10 0.00
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.00 0.15 0.10 0.10 0.15 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.30 0.00 0.10 0.10 0.00 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.20 0.10 0.00 0.10 0.10 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.20 0.10 0.10 0.00 0.10 0.10
0.00 0.25 0.00 0.00 0.00 0.00 0.15 0.30 0.00 0.10 0.10 0.00 0.10
```

```
0.00 0.40 0.00 0.00 0.00 0.00 0.00 0.20 0.10 0.10 0.10 0.10 0.00
```

Caution:

a) The radiation model's precision is decided by the user when he/she names the domain edges. In fact, a radiating face is recognised by the preprocessor as the set of domain edges faces bearing the same name. Thus, if the user subdivides the edge into two edges which are named differently, he/she thus creates two radiating faces instead of one.
b) The form factors are entered by the user, the preprocessor carries out no calculations other than checking preservation relationships on form factors.

c) The fluid is considered to be a transparent gas.

Associate: This keyword is used to associate the radiation model to the problem.

Solve: This keyword is used to resolve the problem coupled to radiation.

2.7.2THERMOHYDRAULIC PROBLEM WITH RADIATION MODEL FOR SEMI TRANSPARENT GAS

```
Fluide_Incompressible fluide
Read  fluide { … }
Pb_Thermohydraulique Pb_fluide
Pb_Couple_Rayo_Semi_Transp Pb_couple
…
Modele_Rayo_Semi_Transp  mod
Read  mod {
        Eq_rayo_semi_transp {
                solveur solveur
                boundary_conditions
                {
                        Name_boundary Boundary_condition_type A value emissivite field_type field_description
                        ……
                }
        }
        Post_processing { …  }
}
Associate mod fluide
Associate pb_couple fluide
# The model should be associated to the coupling problem BEFORE the time scheme #
Associate pb_couple mod

Read  pb_fluide
{
        Navier_Stokes_Standard { …. }
        Convection_Diffusion_Temperature
        {
                diffusion { }
                convection { … }
                Initial_Conditions { … }
                sources { Source_rayo_semi_transp }
                boundary_conditions { … }
        }
}
…
Solve pb_couple
```

Pb_Couple_Rayo_Semi_Transp : This keyword is used to define a problem coupling several other problems to which radiation coupling is added.

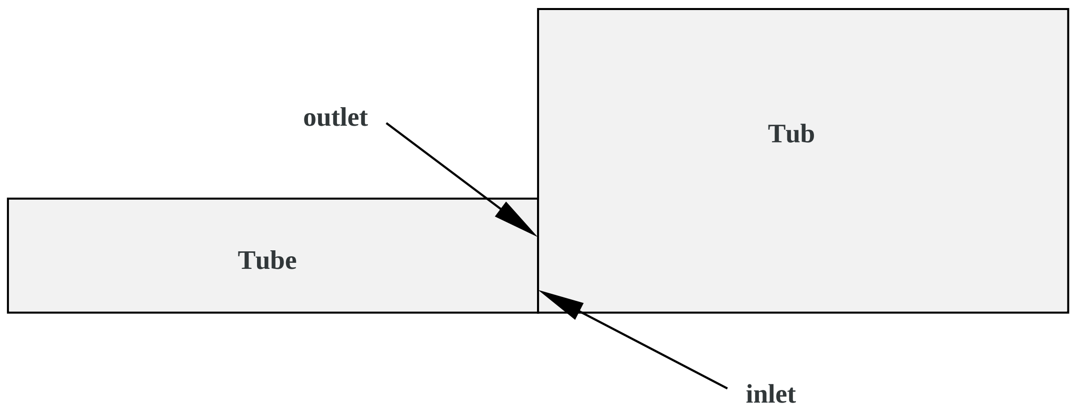
Source_rayo_semi_transp : Radiative term source in energy equation.

Modele_Rayo_Semi_Transp : Keyword to define the radiation model for semi transparent gas
Eq_rayo_semi_transp : Irradiancy G equation. Radiative flux equals -grad(G)/3/kappa
Post_processing : The model is a problem with the usual definition of the fields being postprocessed, here the **irradiance** field.
solveur : Keyword to define the solver of the irradiancy equation
Boundary_condition_type :
Flux_radiatif_VDF : Boundary condition for radiation equation in VDF.
Flux_radiatif_VEF : Boundary condition for radiation equation in VEF.
A : Constant in boundary condition for irradiancy (sqrt(3) for half-infinite domain or 2 in closed domain)
emissivite : Wall emissivity, value between 0 and 1.

Warning:
Calculation with semi transparent gas model may lead to divergence when high temperature differences are used. Indeed, the calculation of the stability time step of the equation does not take in account the source term. In semi transparent gas model, energy equation source term depends strongly of temperature via irradiance and stability is not guaranteed by the calculated time step. Reducing the **facsec** of the time scheme is a good tip to reach convergence when divergence is encountered.

2.7.3OTHER COUPLINGS

TRUST allows other couplings to be performed. Examples allowed by the structure are given here.



```
Dimension 2
Domaine tuyau
Domaine cuve
Read _file  tuyau  geom1.doc          # the tube object is read in the geom1.doc file #
Read _file  cuve geom2.doc            # the tub object is read in the geom2.doc file #
Schema_Euler_Explicite sch
Read _file  sch  sch_tps.doc
VDF dis
Fluide_Incompressible  fluide
Read _file  fluide  fluide.doc
Pb_Hydraulique pb1                    # The Pb_Hydraulique type object pb1 is created#
Pb_Hydraulique pb2                    # The Pb_Hydraulique type object, pb2 is created#
Associate pb1 tuyau
Associate pb2 cuve
Associate pb1 fluide
Associate pb2 fluide
Probleme_Couple pb_couplage         # Create the pb_couplage object; problems must be
                                     associated to the Probleme_Couple type  object
```

```
                                        before applying the other instructions #
Associate pb_couplage pb1               # Association of the pb1 object to the pb_couplage
                                        object #
Associate  pb_couplage pb2              # Association of the pb2 object to the pb_couplage
                                        object #
Associate pb_couplage sch
Discretize pb_couplage dis
# the tube object contains an edge called outlet
the tub object contains an edge called inlet
The two edges are identical from a geometric point of view and coupling is achieved by
means of this edge #
Read  pb1
{
        Navier_Stokes_std
        {
                …....................
                boundary_conditions {
                        …....................
                        sortie   Frontiere_ouverte_pression_imposee
                        Champ_front_recyclage {
                                pb_champ_evaluateur pb2  pression 1
                        }
                }
        }
}

Read  pb2
{
        Navier_Stokes_std
        {
                …....................
```

```
                  Conditions_aux_limites {
                        …....................
                        entree  Frontiere_ouverte_vitesse_imposee
                        Champ_front_recyclage {
                              pb_champ_evaluateur pb1 vitesse 2
                        }
                  }
            }
      }

      Solve pb_couplage
```

2.8 SPATIAL DISCRETIZATION

2.8.1 CONVECTIVE SCHEMES

Scheme availability:

Scheme name	*Keyword VDF*	*Keyword VEF*
No scheme	Negligeable	Negligeable
Upwind generic formulation	■	Generic
Upwind	Amont	Amont
Quick-Sharp	Quick	Kquick
Center (order 2)*	Centre	Centre
Center (order 4)*	Centre4	Centre4
Muscl	■	Muscl
DI_L2	■	DI_L2
ALE	■	ALE
EF_stab	■	EF_stab
EF	■	EF

(*): **Warning**: the centered schemes are unstable under some conditions.

The keyword **Negligeable** suppresses the Navier Stokes convection operator.

EF_stab : Keyword for a VEF convective scheme.

EF_stab { [**TdivU**] [**alpha** factor] [**volumes_etendus**] [**volumes_non_etendus**] [**old**] [**test**]
[**amont_sous_zone** sub_zone_name]
[**alpha_sous_zone** N sub_zone_name_1 alpha_1 … sub_zone_name_N alpha_N]
}

The options of the keyword are :
TdivU : To have the convective operator calculated as div(TU)-TdivU(=UgradT).

alpha double : To weight the scheme centering with the factor double (between 0 (full centered) and 1 (mix between upwind and centered), by default 1).
volumes_etendus : Option for the scheme to use the extended volumes (default, yes).
volumes_non_etendus : Option for the scheme to not use the extended volumes (default, no).

old : To use old version of EF_stab scheme (default no).
test : Developer option to compare old and new version of EF_stab
amont_sous_zone sz_name : Option to degenerate EF_stab scheme into Amont (upwind) scheme in the sub zone of name sz_name. The sub zone may be located arbitrarily in the domain but the more often this option will be activated in a zone where EF_stab scheme generates instabilities as for free outlet for example.
alpha_sous_zone N sub_zone_name_1 alpha_1 …. sub_zone_name_N alpha_N : Option to change locally the **alpha** value on N sub-zones named sub_zone_name_I. Generally, it is used to prevent from a local divergence by increasing locally the **alpha** parameter.

Generic scheme [limiter] [order] **:** Keyword for generic calling of upwind and muscl convective scheme in VEF discretization. For muscl scheme, limiters and order for fluxes calculations have to be specified. The available limiters are : **minmod** - **vanleer** -**vanalbada** - **chakravarthy** - **superbee**, and the order of accuracy is 1 or 2. Note that **chakravarthy** is a non-symmetric limiter and **superbee** may engender results out of physical limits. By consequence, these two limiters are not recommended.

Examples:

convection { **generic amont** }
convection { **generic muscl minmod** 1 }
convection { **generic muscl vanleer** 2 }

In case of results out of physical limits with muscl scheme (due for instance to strong non-conformal velocity flow field), user can redefine in data file a lower order and a smoother limiter, as :
convection { **generic muscl minmod** 1 }

Amont: Keyword for upwind scheme in VEF discretization equivalent to **generic amont** for the 1.5 version or later. The previous upwind scheme can be used with the obsolete in future **amont_old** keyword.

Muscl: Keyword for muscl scheme in VEF discretization equivalent to **generic muscl vanleer 2** for the 1.5 version or later. The previous muscl scheme can be used with the obsolete in future **muscl_old** keyword.

ALE { scheme } : Keyword to use a convective scheme for ALE method.

Example: See the test case ALE_membrane

```
Navier_Stokes_standard
{
        solveur_pression GCP { ... }
        convection { ALE { amont } }
        diffusion { }
        Initial_Conditions { ... }
        boundary_conditions  {
                    Bord1 frontiere_ouverte_vitesse_imposee
                          Champ_front_ALE 2 20*0.3*SIN(6.28*y)*COS(20*t) 0.
         }
}
```

EF : For VEF calculations, a centred convective scheme based on Finite Elements formulation can be called through the following data:

Convection { **EF transportant_bar** val **transporte_bar** val **antisym** val **filtrer_resu** val }

This scheme is 2nd order accuracy (and get better the property of kinetic energy conservation). Due to possible problems of instabilities phenomena, this scheme has to be coupled with stabilisation process (see **Source_Qdm_lambdaup**)

For parameterised studies, following keywords (admitting Boolean values 0 or 1) can be specified.

transportant_bar 1 refers to filtered transporting velocity (P1-conform)
transporte_bar 1 refers to filtered transported velocity (P1-conform)
antisym 1 adjoins anti-symmetric part for preserving kinetic energy
filtrer_resu 1 filters all the convective fluxes contribution

In the aim not to specify these keywords, **defaut_bar** can be used :

Convection { **EF** defaut_bar } , equivalent to :
convection { **EF** transportant_bar 0 transporte_bar 1 filtrer_resu 1 antisym 1 }

These two last data are equivalent from a theoretical point of view in variationnal writing to : ½[(u. grad ub , vb) – (u. grad vb, ub)], where vb corresponds to the filtered reference test functions.

Remark:
This class requires to define a filtering operator : see **solveur_bar**

2.8.2DIFFUSIVE SCHEME

Several possibilities are available to take in count or not the diffusivity:
Diffusion: This keyword is used to specify the diffusion operator.

```
Diffusion { [keyword] }
```

Several possible uses:
Diffusion { } : the standard diffusive scheme used is an order 2 scheme.

Diffusion { stab { [**standard** integer] [**info** integer] [**new_jacobian** integer]
[**nu** integer] [**nut** integer] [**nu_transp** integer] [**nut_transp** integer] } }
A keyword allowing consistent and stable calculations even in case of obtuse angle meshes.

Several options ara available for general flow:
standard integer : to recover the same results as calculations made by standard laminar diffusion operator. However, no stabilization technique is used and calculations may be unstable when working with obtuse angle meshes (by default 0)
info integer : developer option to get the stabilizing ratio (by default 0)
new_jacobian integer : when implicit time schemes are used, this option defines a new jacobian that may be more suitable to get stationary solutions (by default 0)
Several options are available for turbulent flow:
nu 1 (respectively **nut** 1) takes the molecular viscosity (resp. eddy viscosity) into account in the velocity gradient part of the diffusion expression (by default **nu**=1 and **nut**=1)
nu_transp 1 (respectively **nut_transp** 1) takes the molecular viscosity (resp. eddy viscosity) into account in the transposed velocity gradient part of the diffusion expression (by default **nu_transp**=0 and **nut_transp**=1)

Diffusion { negligeable } : the diffusivity will not taken in count exactly as if the equation has no diffusive operator.

Diffusion { implicite Solveur kind_of_solver **{** options_for_solver **} }** : To have diffusive implicitation, it use **Uzawa** algorithm. Very useful when viscosity has large variations.

Uzawa: Keyword to set the convergency of the Uzawa algorithm if Implicite Solveur keyword has been set in **Diffusion**.

Example:
```
Read  pb
{
      Navier_Stokes_standard
      {
            solveur_pression GCP { … }
            convection { amont }
            diffusion { implicite solveur cholesky { impr } }
            uzawa 1.e-8
            Initial_Conditions { … }
            boundary_conditions { … }
      }
      Post_processing { … }
```

Diffusion { **P1NCP1B** { [**alphaE** integer] [**alphaS** integer] [**alphaA** integer] [**test**] [**decentrage** integer] [**epsilon** double] } }
A keyword intended for conduction calculations to improve the default diffusion scheme when used with VEFPre1B discretization.

alphaE integer: to add (integer=1) or suppress (integer=0) the P0 part of the operator (by default 1)
alphaS integer: to add (integer=1) or suppress (integer=0) the P1 part of the operator (by default 1)
alphaA integer: to add (integer=1) or suppress (integer=0) the P2 part of the operator (option not coded yet so by default 0)
test : developer option to compare explicit and implicit operators
decentrage integer: to ensure the positivity of the operator (by default 1)
epsilon double : to weight the P0 part of the operator (between 0, full P1 discretization, and 1, full P0 discretization, default 1e-3)

Diffusion { standard grad_Ubar value **nu** value **nut** value **nu_transp** value **nut_transp** value **filtrer_resu** value } : A new keyword, intended for LES calculations, has been developed to optimise and parameterise each term of the diffusion operator.

For parameterised studies, following keywords (admitting Boolean values 0 or 1) can be specified.

grad_Ubar 1 makes the gradient calculated through the filtered values of velocity (P1-conform).
nu 1 (respectively **nut** 1) takes the molecular viscosity (eddy viscosity) into account in the velocity gradient part of the diffusion expression.

nu_transp 1 (respectively **nut_transp** 1) takes the molecular viscosity (eddy viscosity) into account according in the TRANSPOSED velocity gradient part of the diffusion expression.
filtrer_resu 1 allows to filter the resulting diffusive fluxes contribution.

In the aim not to specify these keywords, **defaut_bar** can be used :

diffusion { **standard** defaut_bar } , equivalent to :
diffusion { **standard** grad_Ubar 1 nu 1 nut 1 nu_transp 1 nut_transp 1 filtrer_resu 1 }

Remark:

1. This class requires to define a filtering operator : see **solveur_bar**
2. The former (original) version: diffusion { } -which omitted some of the term of the diffusion operator- can be recovered by using the following parameters in the new class :
diffusion { **standard** grad_Ubar 0 nu 1 nut 1 nu_transp 0 nut_transp 1 filtrer_resu 0}.

2.9TIME SCHEMES

type_schema sch

type_schema: scheme type in time used.
sch: object identifier

The available types are explicit schemes:

Schema_Euler_explicite
Schema_Adams_Bashforth_order_2
Schema_Adams_Bashforth_order_3
Runge_Kutta_Rationnel_ordre_2
Runge_Kutta_ordre_3
Runge_Kutta_ordre_4_D3P
Schema_Predictor_Corrector
Sch_CN_iteratif
Sch_CN_EX_iteratif
Schema_Phase_Field
RK3_FT

And also implicit schemes:

Schema_Euler_implicite
Schema_Adams_Moulton_order_2
Schema_Adams_Moulton_order_3
Schema_Backward_Differentiation_order_2
Schema_Backward_Differentiation_order_3

Example:
Runge_Kutta_ordre_3 *sch*

The time scheme parameters are then read.
The read block that follows is similar for all scheme types.

```
Read  sch
{
    [tinit vrel]

    [tmax vrel]

    [tcpumax vrel]

    [nb_pas_dt_max integer]

    [dt_min vrel]

    [dt_max vrel]

    [dt_start ….]

    [dt_impr vrel]

    [precision_impr integer]

    [dt_sauv vrel]

    [seuil_statio vrel]

    [facsec vrel]

    [facsec_max double]
    [facsec_evol_facteur double]
    [max_iter_implicite int]
    [Solveur solver {
         [seuil_convergence_implicite vrel]
         [no_qdm]
         [solveur solver]
         [seuil_generation_solveur vrel]
         [seuil_test_preliminaire_solveur vrel]
         [seuil_verification_solveur vrel]
         [relax_pression vrel]
         [nb_corrections_max int]
         }
     ]
  [diffusion_implicite integer ]
  [seuil_diffusion_implicite vrel]
  [impr_diffusion_implicite int]
  [niter_max_diffusion_implicite ivalue]
  [periode_sauvegarde_securite_en_heures ivalue]
  [no_check_disk_space]
}
```

tinit vrel: This is a keyword and the value of the initial calculation time (0 by default).

tmax vrel: This is an optional keyword and the time during which the calculation was stopped (10^{30}s by default).

tcpumax vrel : Optional CPU time limit (must be specified in hours) for which the calculation is stopped (10^{30}s by default).

nb_pas_dt_max integer: This is a keyword and the maximum number of calculation time steps.

dt_min vrel: This is a keyword and the minimum calculation time step (10^{-16}s by default).

dt_max vrel: This keyword gives the maximum calculation time step (10^{30}s by default).

dt_start: This keyword allows to specify the way to define the time step when (re)starting a calculation.

dt_start dt_min : the first iteration is based on dt_min
dt_start dt_calc : the time step at first iteration is calculated in agreement with CFL condition.
dt_start dt_fixe value : the first time step is fixed by the user (recommended when restarting calculation with Crank Nicholson temporal scheme to ensure continuity).

By default, the first iteration is based on dt_calc.

```
Schema_Euler_Explicite sch
Read  sch
{
      tinit 0.563
      tmax 1.
      dt_min 0.00001
      dt_max 0.2
      dt_start dt_fixe  0.000154
      dt_impr 0.001
         …
}
```

dt_impr vrel: This is a keyword and scheme parameter printing time step in time (10^{30}s by default). The time steps and the flux balances are printed (incorporated onto every side of processed domains) into the **.out** file.

precision_impr integer: Optional keyword to define the digit number for flux values printed into .out files (by default 3).

dt_sauv vrel: This is a keyword and holds the save time step value (10^{30}s by default). Every dt_sauv, fields are saved in the **.sauv** file.

seuil_statio vrel: This is a keyword and holds the value of the convergence threshold (10^{-12} by default). Problems using this type of time scheme converge when the derivatives dG_i/dt of all the unknown transported values G_i have a combined absolute value less than this value. This is the keyword used to set the permanent rating threshold.

facsec vrel: This is a keyword and the value assigned to the safety factor for the time step (1. by default). The time step calculated is multiplied by the safety factor. **The first thing to try when a calculation does not converge with an explicit time scheme is to reduce the facsec to 0.5.**
Warning: Some schemes needs a facsec lower than 1 (0.5 is a good start), for example **Schema_Adams_Bashforth_order_3**

Solveur *solver*: This keyword is used to designate the solver selected in the situation where the time scheme is an implicit scheme (see list page 145). *solver* is the name of the solver that allows equation diffusion and convection operators to be set as implicit terms. Keywords corresponding to this functionality are **Simple** (SIMPLE type algorithm), **Simpler** (SIMPLER type algorithm) for incompressible systems, **Piso** (**P**ressure **I**mplicit with **S**plit **O**perator), and **Implicite** (similar to PISO, but as it looks like a simplified solver, it will use fewer timesteps. But it may run faster because the pressure matrix is not re-assembled and thus provides CPU gains.

Advice: Since the 1.6.0 version, we recommend to use first the **Implicite** or **Simple**, then **Piso**, and at least **Simpler**. Because the two first give a fastest convergence (several times) than **Piso** and the **Simpler** has not been validated. It seems also than **Implicite** and **Piso** schemes give better results than the **Simple** scheme when the flow is not fully stationary. Thus, if the solution obtained with **Simple** is not stationary, it is recommended to switch to **Piso** or **Implicite** scheme.

seuil_convergence_implicite : Keyword to set the value of the convergence criteria for the resolution of the implicit system build by solving several times per time step the Navier_Stokes equation and the scalar equations if any. This value MUST be used when a coupling between problems is considered (should be set to a value typically of 0.1 or 0.01).

no_qdm : Optional keyword to not solve the impulsion equation (and turbulence models of these equation)

solveur solver : **solveur** is an optional keyword to specify a method (different from the default one, **Gmres** with diagonal preconditioning) to solve the linear system for implicitation.

<u>Advice:</u>
A good strategy (best CPU results) for the choice of the solver is to specify a **GMRES** method (and diagonal preconditioning) with a very low convergence threshold but limit to a maximum of 5 iterations (it converges generally quicky in few iterations):
solveur gmres { **diag seuil** 1e-30 **nb_it_max** 5 **impr** }
And in a first approach, to not use the following thresholds:

seuil_generation_solveur *vrel*: Option to create a **GMRES** solver and use *vrel* as the convergence threshold (implicit linear system Ax=B will be solved if residual error ||Ax-B|| is lesser than *vrel*)

seuil_verification_solveur *vrel* : Option to check if residual error ||Ax-B|| is lesser than *vrel* after the implicit linear system Ax=B has been solved.

seuil_test_preliminaire_solveur *vrel* : Option to decide if the implicit linear system Ax=B should be solved by checking if the residual error ||Ax-B|| is bigger than *vrel*.

<u>NB:</u>
seuil_convergence_solveur *vrel* option becomes obsolete since the 1.6.2 version. In the past, the same value *vrel* was used for the 3 last thresholds.

facsec_max double: Maximum ratio allowed between time step and stability time returned by CFL condition. The initial ratio given by **facsec** keyword is changed during the calculation with the implicit scheme but it couldn't be higher than **facsec_max** value.

Warning: Some implicit schemes do not permit high facsec_max, example **Schema_Adams_Moulton_order_3** needs facsec=facsec_max=1.

Advice:
The calculation may start with a **facsec** specified by the user and increased by the algorithm up to the **facsec_max** limit. But the user can also choose to specify a constant facsec (**facsec_max** will be set to **facsec** value then). Faster convergence has been seen and depends on the kind of calculation:
-Hydraulic only or thermal hydraulic with forced convection and low coupling between velocity and temperature (Boussinesq value β low), **facsec** between 20-30
-Thermal hydraulic with forced convection and strong coupling between velocity and temperature (Boussinesq value β high), **facsec** between 90-100
-Thermohydralic with natural convection, **facsec** around 300
-Conduction only, **facsec** can be set to a very high value (10^8) as if the scheme was unconditionally stable
These values can also be used as rule of thumb for initial **facsec** with a **facsec_max** limit higher.

max_iter_implicite int: Maximum number of iterations allowed for the implicit algorithm (by default 200).
relax_pression vrel: Value between 0 and 1 (by default 1), this keyword is used only by the SIMPLE algorithm for relaxing the increment of pressure.
nb_corrections_max int : Maximum number of corrections performed by the **PISO** algorithm to achieve the projection of the velocity field. The algorithm may perform less corrections then nb_corrections_max if the accuracy of the projection is sufficient. (By default nb_corrections_max is set to 21).

diffusion_implicite *integer*: This keyword is used to make the diffusion term in the Navier Stokes equation implicit (in this case, *integer* should be set to1). The stability time step is then only based on the convection time step (dt=facsec*dt_convection). Thus, in some circumstances, an important gain is achieved with respect to the time step (large diffusion with respect to convection on tightened meshes). *Caution*: It is however recommended that the user should avoid exceeding the calculation convection time step by selecting a facsec that is too large. Start with a facsec of 1 and then increase this gradually if you wish to accelerate calculation. In addition, for a natural convection calculation with a zero initial speed, in the first time step, the convection time is infinite and therefore dt=facsec*dt_max.

seuil_diffusion_implicite: This keyword changes the default value (1e-6) of convergency criteria for the resolution by conjugate gradient used for **implicit diffusion**.

impr_diffusion_implicite 0|1**:** Unactivate (default) or not the printing of the convergence during the resolution of the conjugate gradient.

niter_max_diffusion_implicite: This keyword changes the default value (number of unknowns) of the maximal iterations number in the conjugate gradient method used for **implicit diffusion**.

periode_sauvegarde_securite_en_heures: This keyword is used to change the default period (10 hours) between the save of the fields in .sauv file.

no_check_disk_space : This keyword disables the check of the available amount of disk space during the calculation.

Note:
The new scheme **Schema_Predictor_Corrector** scheme (2nd order) is more accurate and economic than MacCormack scheme. It gives best results with a second ordre convective scheme like quick, centre (VDF).

Example: (See test case Pred_Cor_VEF)

```
Schema_Predictor_Corrector sch
Read  sch {
        tinit 0.
        tmax 2.
        dt_min 1.e-5
        dt_max 1.
        dt_impr 1.e-4
        dt_sauv 100
        seuil_statio 1.e-12
  }
```

Sch_CN_iteratif
This keyword describes a Crank-Nicholson method of second order accuracy. A mid-point rule formulation is used (Euler-centered scheme).

The basic scheme is: u(t+1) = u(t) + du/dt(t+1/2)*dt.

The estimation of the time derivative du/dt at the level (t+1/2) is obtained either by iterative process. The time derivative du/dt at the level (t+1/2) is calculated iteratively with a simple under-relaxations method. Since the method is implicit, neither the cfl nor the fourier stability criteria must be respected. The time step is calculated in a way that the iterative procedure converges with the less iterations as possible.

Parameters *(and values taken by defaut):*

niter_min: minimal number of p-iterations to satisfy convergence criteria (*2*)

niter_max : number of maximum p-iterations allowed to satisfy convergence criteria (*6*)

niter_avg: threshold of p-iterations (*3*). If the number of p-iterations is greater than **niter_avg,** facsec is reduced, if lesser than **niter_avg**, facsec is increased (but limited by the **facsec_max** value).

facsec_max : maximum ratio allowed between dynamical time step returned by iterative process and stability time returned by CFL condition (*2*).

seuil: criteria for ending iterative process (Max(|| u(p) – u(p-1)||/Max || u(p) ||) < **seuil**) (*0.001*)

Sch_CN_EX_iteratif

This keyword also describes a Crank-Nicholson method of second order accuracy but here, for scalars, because of instablities encountered when dt>dt_CFL, the Crank Nicholson scheme is not applied to scalar quantities. Scalars are treated according to Euler-Explicite scheme at the end of the CN treatment for velocity flow fields (by doing p Euler explicite under-iterations at dt<=dt_CFL).

Parameters are the sames (but default values may change) compare to the **Sch_CN_iterative** scheme plus a relaxation keyword*:*

niter_min: (*2*)

niter_max : (*6*)

niter_avg: (*3*)

facsec_max : (*20*)

seuil: (*0.05*)

omega: relaxation factor (*0.1*)

Remark: for stationary or RANS calculations, no limitation can be given for time step through high value of **facsec_max** parameter (for instance: **facsec_max** 1000). In counterpart, for LES calculations, high values of **facsec_max** may engender numerical instabilities.

Schema_Phase_Field

Keyword for the only available Scheme for time discretization of the Phase Field problem. This keyword has two mandatory options:

Schema_CH scheme { } : Time scheme for the Cahn-Hilliard equation.

Schema_NS scheme { } : Time scheme for the Navier-Stokes equation.

RK3_FT

Keyword for Runge Kutta time scheme for Front_Tracking calculation. Validated en tested only for the two following cases : problem with hydraulic and one interface equation, problem with hydraulic equation, one interface equation and one concentration equation.

2.10 PRESSURE SOLVERS

Solveur_pression type_solveur *solver_description*

Solveur_pression: This keyword is used to indicate the choice of pressure solver.

Three algorithms are possible for the pressure solver.

2.10.1 PRECONDITIONED CONJUGATED GRADIENT

```
Solveur_pression GCP { [ [precond_nul] precond type_precond { [ omega omega ] } ]
  [ seuil seuil ]
  [ impr | quiet ]
  [ optimized ]
}
```

Where:
seuil *seuil* : corresponds to the conjugated gradient convergence value. The method stops to iterate when the Euclidean residue standard ||Ax-B|| is less than this value.

precond_nul : Keyword to not use a preconditioning method.

precond : is a keyword used to define system preconditioning in order to accelerate resolution by the conjugated gradient. For example, a preconditioning **ssor** with a overrelaxation factor *omega* (between 1 and 2, optimal value around 1.5-1.6). Many parallel preconditioning methods are not equivalent to their sequential counterpart, and you should therefore expect differences, especially when you select a high value of the final residue ("seuil"). The result depends on the number of cpus and on the mesh splitting. It is sometimes useful to run the solver with no preconditioning at all. In particular:
- when the solver does not converge during initial projection,
- when comparing sequential and parallel computations.
With no preconditioning, except in some particular cases (no open boundary), the sequential and the parallel computations should provide exactly the same results within fpu accuracy. If not, there might be a coding error or the system of equations is singular.

impr is the keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).

quiet is a keyword which is used to not displaying any outputs of the solver.

optimized : This keyword triggers a memory and network optimized algorithms useful for strong scaling (when computing less than 100 000 elements per processor). The matrix and the vectors are duplicated, common items removed and only virtual items really used in the matrix are exchanged.
Warning: this is experimental and known to fail in some VEF computations (L2 projection step will not converge). Works well in VDF.

Observations:
Use the pressure solver **Arakawa_P1** with VEF_P1_P1 discretization in order to avoid the appearance of parasite pressure.
Example of using the **Arakawa_P1** solver:
solveur_pression Arakawa_P1 { **omega** 1.5 **seuil** 1.e-12 **impr epsilon** 0. }

The value of epsilon should be included between 0 and 1. If epsilon = 0, no stabilisation is detected.

2.10.2SOLVERS FROM PETSC API

Solveur_pression Petsc *Solver* { **precond** *Precond*
[**seuil** *seuil* | **nb_it_max** *integer*]
[**impr** | **quiet**]
[**save_matrix**|**read_matrix**]
}

Solver : Several solvers through PETSc API are available :

GCP : Conjugate Gradient

PIPECG : Pipelined Conjugate Gradient (possible reduced CPU cost during massive parallel calculation due to a single non-blocking reduction per iteration, if TRUST is built with a MPI-3 implementation).

GMRES : Generalized Minimal Residual

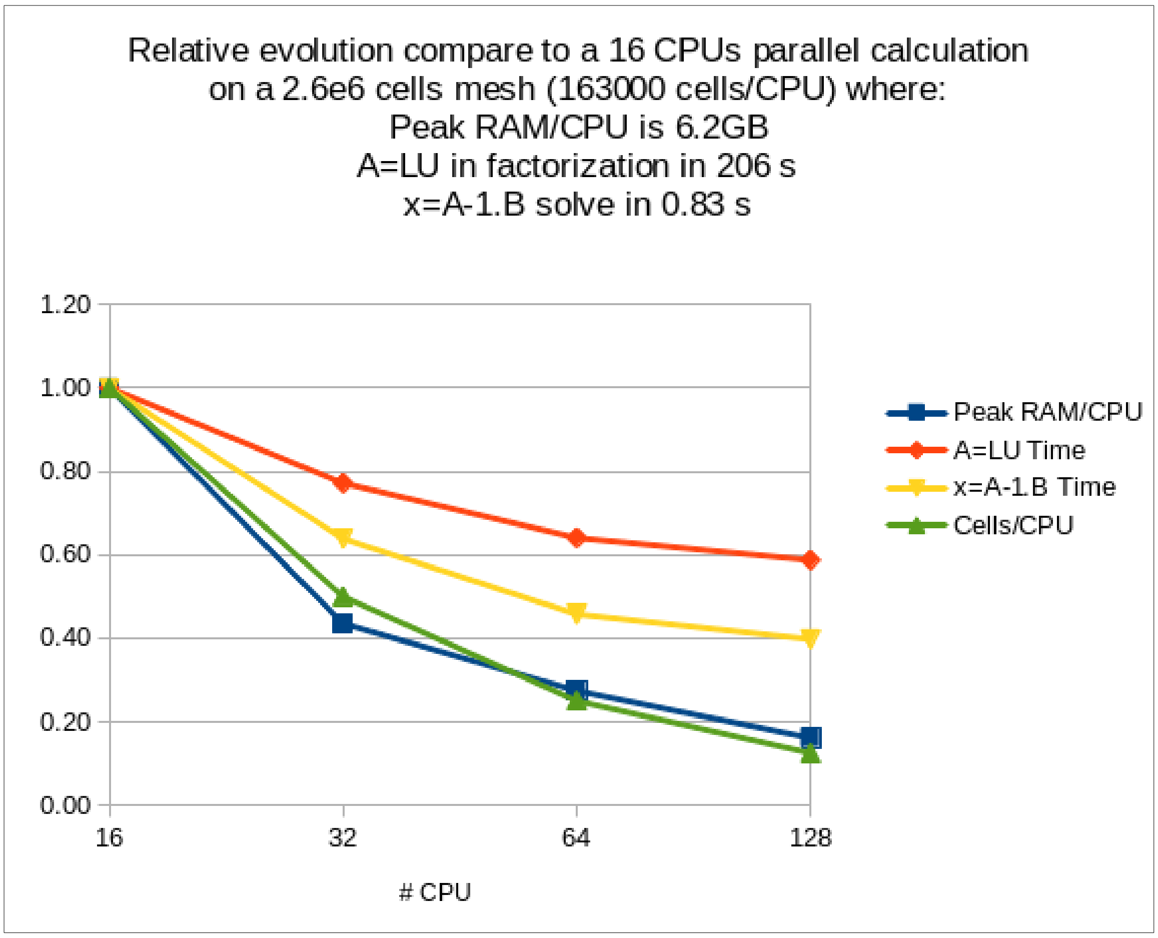
BICGSTAB : Stabilized Bi-Conjugate Gradient

IBICGSTAB : Improved version of previous one for massive parallel computations (only a single global reduction operation instead of the usual 3 or 4).

CHOLESKY : Parallelized version of Cholesky from MUMPS library. This solver accepts since the 1.6.7 version an option to select a different ordering than the automatic selected one by MUMPS (and printed by using the **impr** option). The possible choices are **Metis | Scotch | PT-Scotch | Parmetis**. The two last options can't only be used during a parallel calculation, whereas the two first are available for sequential or parallel calculations. It seems that the CPU cost of A=LU factorization but also of the backward/forward elimination steps may sometimes be reduced by selecting a different ordering than the default one. Notice that this solver requires a huge amont of memory compared to iterative methods. To know how many RAM you will need by core, then use the **impr** option to have detailled informations during the analysis phase and before the factorisation phase (in the following output, you will learn that the largest memory is taken by the 0th CPU with 108MB):

```
...
 ** Rank of proc needing largest memory in IC facto        :        0
 ** Estimated corresponding MBYTES for IC facto            :      108
...
```

Thanks to the following graph, you read that in order to solve for instance a flow on a mesh with 2.6e6 cells, you will need to run a parallel calculation on 32 CPUs if you have cluster nodes with only 4GB/core (6.2GB*0.42~2.6GB) :



CHOLESKY_OUT_OF_CORE : Same as the previous one but with a written LU decomposition of disc (save RAM memory but add an extra CPU cost during Ax=B solve)

CHOLESKY_SUPERLU : Parallelized Cholesky from SUPERLU_DIST library (less CPU and RAM efficient than the previous one)

CHOLESKY_PASTIX : Parallelized Cholesky from PASTIX library

CHOLESKY_UMFPACK : Sequential Cholesky from UMFPACK library (seems fast).

LU: Same as Cholesky but for a non symmetric matrix.

CLI { string } : Command Line Interface. Should be used only by advanced users, to access the whole solver/preconditioners from the PETSC API. To find all the available options, run your calculation with the -ksp_view -help options:

```
trust datafile [N]  –ksp_view –help
…
Preconditioner (PC) Options -------------------------------------------------
  -pc_type Preconditioner:(one of) none jacobi pbjacobi bjacobi sor lu shell mg
```

eisenstat ilu icc cholesky asm ksp composite redundant nn mat fieldsplit galerkin openmp spai hypre tfs (PCSetType)

HYPRE preconditioner options

-pc_hypre_type <pilut> (choose one of) pilut parasails boomeramg euclid

HYPRE ParaSails Options

-pc_hypre_parasails_nlevels <1>: Number of number of levels (None)

-pc_hypre_parasails_thresh <0.1>: Threshold (None)

-pc_hypre_parasails_filter <0.1>: filter (None)

-pc_hypre_parasails_loadbal <0>: Load balance (None)

-pc_hypre_parasails_logging: <FALSE> Print info to screen (None)

-pc_hypre_parasails_reuse: <FALSE> Reuse nonzero pattern in preconditioner (None)

-pc_hypre_parasails_sym <nonsymmetric> (choose one of) nonsymmetric SPD nonsymmetric,SPD

Krylov Method (KSP) Options ---

-ksp_type Krylov method:(one of) cg cgne stcg gltr richardson chebychev gmres tcqmr

bcgs bcgsl cgs tfqmr cr lsqr preonly qcg bicg fgmres minres symmlq lgmres lcd (KSPSetType)

-ksp_max_it <10000>: Maximum number of iterations (KSPSetTolerances)

-ksp_rtol <0>: Relative decrease in residual norm (KSPSetTolerances)

-ksp_atol <1e-12>: Absolute value of residual norm (KSPSetTolerances)

-ksp_divtol <10000>: Residual norm increase cause divergence (KSPSetTolerances)

-ksp_converged_use_initial_residual_norm: Use initial residual residual norm for computing relative convergence

-ksp_monitor_singular_value <stdout>: Monitor singular values (KSPMonitorSet)

-ksp_monitor_short <stdout>: Monitor preconditioned residual norm with fewer digits (KSPMonitorSet)

-ksp_monitor_draw: Monitor graphically preconditioned residual norm (KSPMonitorSet)

-ksp_monitor_draw_true_residual: Monitor graphically true residual norm (KSPMonitorSet)

Example to use the multigrid method as a solver, not only as a preconditioner:
Solveur_pression Petsc CLI { -ksp_type richardson -pc_type hypre -pc_hypre_type boomeramg -ksp_atol 1.e-7 }

Precond : Several preconditioners are available :

NULL { } : No preconditioner used

ILU { **level** k }: Parallel incomplete LU factorization (PILU(k) algorithm from Euclid Hypre library). The integer k is the factorization level (default value, 1).

BLOCK_JACOBI_ICC { **level** k **ordering natural | rcm** } : Incomplete Cholesky factorization for symmetric matrix with the PETSc implementation. The integer k is the factorization level (default value, 1). In parallel, the factorization is done by block (one per processor by default). The ordering of the local matrix is **natural** by default, but **rcm** ordering, which reduces the bandwith of the local matrix, may interestingly improves the quality of the decomposition and reduces the number of iterations. This precondtioner converges generally with more iterations than the parallel version ILU from Hypre, but will be much more faster.

SSOR { **omega** double } : Symmetric Successive Over Relaxation algorithm. **omega** (default value, 1.5) defines the relaxation factor.

EISENTAT { **omega** double } : SSOR version with Eisenstat trick which reduces the number of computations and thus CPU cost

SPAI { **level** nlevels **epsilon** thresh } : Spai Approximate Inverse algorithm from Parasails Hypre library. Two parameters are available, nlevels and thresh.

PILUT { **level** k **epsilon** thresh }: Dual Threashold Incomplete LU factorization. The integer k is the factorization level and **epsilon** is the drop tolerance.

DIAG { } : Diagonal (Jacobi) preconditioner.

BOOMERAMG { } : Multigrid preconditioner (no option is available yet, look at CLI command and Petsc documentation to try other options).

seuil corresponds to the iterative solver convergence value. The iterative solver converges when the Euclidean residue standard ||Ax-B|| is less than the value *seuil*.

nb_it_max integer : In order to specify a given number of iterations instead of a condition on the residue with the keyword **seuil**. May be useful when defining a PETSc solver for the implicit time scheme where convergence is very fast: 5 or less iterations seems enough.

impr is the keyword which is used to request display of the Euclidean residue standard each time this iterates through the conjugated gradient (display to the standard outlet).

quiet is a keyword which is used to not displaying any outputs of the solver.

save_matrix|read_matrix are the keywords to save|read into a file the constant matrix A of the linear system Ax=B solved (eg: matrix from the pressure linear system for an incompressible flow). It is useful when you want to minimize the MPI communications on massive parallel calculation. Indeed, in VEF discretization, the overlapping width (generaly 2, specified with the **largeur_joint** option in the partition keyword **partition**) can be reduced to 1, once the matrix has been properly assembled and saved. The cost of the MPI communications in TRUST itself (not in PETSc) will be reduced with length messages divided by 2. So the strategy is:
I) Partition your VEF mesh with a **largeur_joint** value of 2

II) Run your parallel calculation on 0 time step, to build and save the matrix with the **save_matrix** option. A file named *Matrix_NBROWS_rows_NCPUS_cpus.petsc* will be saved to the disc (where NBROWS is the number of rows of the matrix and NCPUS the number of CPUs used).
III) Partition your VEF mesh with a **largeur_joint** value of 1
IV) Run your parallel calculation completly now and substitute the **save_matrix** option by the **read_matrix** option. Some interesting gains have been noticed when the cost of linear system solve with PETSc is small compared to all the other operations.

TIPS:
A) Solver for symmetric linear systems (e.g: Pressure system from Navier Stokes equation):
-The **CHOLESKY** parallel solver is from MUMPS library. It offers better performance than all others solvers if you have enough RAM for your calculation. A parallel calculation on a cluster with 4GBytes on each processor, 40000 cells/processor seems the upper limit. Seems to be very slow to initialize above 500 cpus/cores.

-When running a parallel calculation with a high number of cpus/cores (typically more than 500) where preconditioner scalabilty is the key for CPU performance, consider **BICGSTAB** with **BLOCK_JACOBI_ICC(1)** as preconditioner or if not converges, **GCP** with **BLOCK_JACOBI_ICC(1)** as preconditioner.

-For other situations, the first choice should be **GCP/SSOR**. In order to fine tune the solver choice, each one of the previous list should be considered. Indeed, the CPU speed of a solver depends of a lot of parameters. You may give a try to the **OPTIMAL** solver to help you to find the fastest solver on your study.

B) Solver for non symmetric linear systems (e.g.: Implicit schemes):
The **BICGSTAB/DIAG** solver seems to offer the best performances.

Additional information is available into the PETSC documentation available there: **$TRUST_ROOT/lib/src/LIBPETSC/petsc/*/docs/manual.pdf**

2.10.3CHOLESKY DIRECT METHOD

Solveur_pression Cholesky { [**impr | quiet**] }

Where:
impr is a keyword which may be used to print the resolution time.

quiet is a keyword which is used to not displaying the outputs of the solver.

The Cholesky implementation in TRUST is not parallel and will become obsolete. Consider **Petsc Cholesky** keywords for a parallel calculation.

2.11 OTHER SOLVERS

We may use also methods for non symmetric linear systems:

2.11.1 PETSC API SOLVERS

```
Petsc Solver { … }
```

Solver may be **GMRES** or **BICGSTAB**. Look at 2.10.2 to the see the options.

2.11.2 GMRES METHOD

```
Gmres {
  seuil double
  [ diag ]
  [ impr | quiet ]
  [ nb_it_max integer ]
  [ controle_residu 0|1 ]
}
```

Where:

seuil double: This keyword is used to define the convergence threshold.
impr is an optional keyword which may be used to print the convergence.
quiet is a keyword which is used to not displaying any outputs of the solver.
diag is an optional keyword to use diagonal preconditioning instead of Pilut one which is not parallel.
nb_it_max is an optional keyword to set the maximum iterations number for the Gmres.
controle_residu is an optional Boolean (by default 0). If set to 1, the convergence occurs if the residu suddenly increases.

2.11.3GEN METHOD

```
Gen {
  seuil double
  solv_elem bicgstab
  precond precond
  [ impr | quiet ]
  [ save_matrice ]
}
```

Where:

seuil double: This keyword is used to define the convergence threshold.

impr is an optional keyword which may be used to print the convergence.

quiet is a keyword which is used to not displaying any outputs of the solver.

solv_elem is the keyword to specify the solver used with the method (**BICGSTAB** is the solver to use if **Gmres** solver fails to converge with the implicit schemes).

precond precond **:** To specify the preconditionner of the solver given with the previous keyword **solv_elem**. ILU is the recommended one when using **BICGSTAB** solver:

ILU { **type** m **filling** n }

With m=1,2,3 (default 2), and n filling of the ILU method (by default 1). For example: **ILU** { **type** 2 **filling** 20 }

save_matrice is an optional keyword to save the matrix in a file.

2.11.4OPTIMAL

```
Optimal {
  seuil val
  [ save_matrice ]
  [ frequence_recalc double ]
  [ nom_fichier_solveur file ]
  [ fichier_solveur_non_recree ]
  [ impr | quiet ]
}
```

Optimal is a solver which tests several solvers of the previous list to choose the fastest one for the considered linear system. Options:

seuil val : Convergence threshold

save_matrice : Keyword to save the linear system (A, x, B) into a file

frequence_recalc double : Keyword to set a time step period (by default, 100) for re-checking the fatest solver

nom_fichier_solveur file : To specify the file containing the list of the tested solvers

fichier_solveur_non_recree : Keyword to avoid the creation of the file containing the list

impr : To print the convergency of the fastest solver.

quiet is a keyword which is used to not displaying any outputs of the solver.

Another keyword is available to test solvers:

```
Test_solveur {
  [ fichier_secmem file ]
  [ fichier_matrice file ]
  [ fichier_solution file ]
  [ nb_test int ]
  [ impr | impr ]
  [ solveur string ]
  [ fichier_solveur  file ]
  [ genere_fichier_solveur  double ]
  [ seuil_verification  precision ]
  [ pas_de_solution_initiale ]
  [ ascii ]
}
```

fichier_secmem file : Filename containing the second member B

fichier_matrice file : Filename containing the matrix A

fichier_solution file : Filename containing the solution x

nb_test int : Number of tests to measure the time resolution (one preconditionnement)

impr: To print the convergence solver

quiet: keyword which is used to not displaying any outputs of the solver.

solveur string : To specify a solver

fichier_solveur file : To specify a file containing a list of solvers

genere_fichier_solveur double : To create a file of the solver with a threshold convergence

seuil_verification precision : Check if the solution satisfy ||Ax-B||<precision
pas_de_solution_initiale : Resolution isn’t initialized with the solution x
ascii : Ascii files

2.12 INITIAL CONDITIONS

2.12.1 SPEEDS

```
Vitesse   field_type   bloc_lecture_champ
```

Vitesse: This keyword is used to define the initial speed values.

field_type : Type of initial speed field.

2.12.2 TEMPERATURE

```
Temperature   field_type   bloc_lecture_champ
```

Temperature: This keyword is used to define the initial temperature values.

field_type : The initial temperature field type.

The initial temperature is given in °C (or K).

2.12.3 TURBULENT VALUES

```
[ K_eps   field_type   bloc_lecture_champ ]
[ Flux_Chaleur_Turbulente   field_type   bloc_lecture_champ ]
[ Fluctu_Temperature   field_type   bloc_lecture_champ ]
```

K_eps: This keyword is used to define the initial kinetic energy values and the turbulent dissipation rate. The initial turbulent kinetic energy is given in $m^2.s^{-2}$ and the initial turbulent dissipation rate is given in $m^2.s^{-3}$.

Flux_Chaleur_Turbulente: This keyword is used to define the initial turbulent heat flux vector values. It is expressed in [mK/s].

Fluctu_Temperature: This keyword is used to define the initial value vector {temperature fluctuation, fluctuation dissipation rate }. This value is expressed in [K^2, K^2].

field_type : Initial value field type.

2.13 BOUNDARY CONDITIONS

It is important to specify here that TRUST will not accept any boundary conditions by default.

2.13.1 HYDRAULIC BOUNDARY CONDITIONS

[Bord **Frontiere_ouverte_vitesse_imposee** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_rho_u_impose** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_pression_imposee** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_gradient_pression_impose** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_gradient_pression_libre_VEFPreP1B** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_gradient_pression_impose_VEFPreP1B** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_pression_imposee_Orlansky**]
[Bord **Paroi_fixe**]
[Bord **Paroi_decalee_Robin** { **delta** value }]
[Bord **Paroi_defilante** boundary_field_type *bloc_lecture_champ*]
[Bord **Symetrie**]
[Bord **Periodique**]
[Bord **Paroi_Knudsen_non_negligeable**] field_type_front *bloc_lecture_champ*
[Bord **Paroi_rugueuse** { **erugu** value }]

Bord: name of the edge where the boundary condition applies.

boundary_field_type: boundary field type.

Direction: to be selected along X, Y or Z

Frontiere_ouverte_vitesse_imposee: This keyword is used to designate a condition of imposed speed at an open boundary called *bord*.

The imposed speed field at the inlet is vectorial and the imposed speed values are expressed in m.s^{-1}.

Frontiere_ouverte_rho_u_impose: This keyword is used to designate a condition of imposed mass rate at an open boundary called *bord*. The imposed mass rate field at the inlet is vectorial and the imposed speed values are expressed in kg.s^{-1}. This boundary condition can be used only with the Quasi compressible model (see 2.6.10).

Frontiere_ouverte_pression_imposee: This keyword is used to designate an imposed pressure condition at the open boundary called *bord*. The imposed pressure field is expressed in Pa.

Frontiere_ouverte_gradient_pression_impose: Keyword used to designate a normal imposed pressure gradient condition on the open boundary called *bord*.
This boundary condition may be only used in VDF discretization. The imposed ∂P/∂n value is expressed in Pa.m^{-1}.

Frontiere_ouverte_pression_imposee_Orlansky: This boundary condition may only be used with VDF discretization **(*)**.

(*) Caution: There is no reference for pressure for theses boundary conditions so it is better to add pressure condition (with **Frontiere_ouverte_pression_imposee**) on one or two cells (for symmetry in a channel) of the boundary where Orlansky conditions are imposed.

Frontiere_ouverte_gradient_pression_libre_VEFPreP1B

Keyword for an oulet boundary condition in VEF P1B/P1NC like Orlansky.

Example:
Sortie **frontiere_ouverte_gradient_pression_libre_VEFPreP1B** Champ_front_uniforme 1 0.

Frontiere_ouverte_gradient_pression_impose_VEFPreP1B

Keyword for an oulet boundary condition in VEF P1B/P1NC on the gradiant of the pressure.

Example:
Sortie **frontiere_ouverte_gradient_pression_impose_VEFPreP1B** Champ_front_uniforme 1 0.

Paroi_fixe: Keyword used to designate a situation of adherence to the wall called *bord* (normal and tangential speed at the edge is zero).

Paroi_decalee_Robin: This keyword is used to designate a Robin boundary condition (a.u+b.du/dn=c) associated with the Pironneau methodology for the wall laws. The value of given by the **delta** option is the distance between the mesh (where symmetry boundary condition is applied) and the fictious wall. This boundary condition needs the definition of the dedicated source terms (**Source_Robin or Source_Robin_Scalaire**) according the equations used.

Paroi_defilante: This keyword is used to designate a condition where tangential speed is imposed on the wall called *bord*. If the speed set by the user is not tangential, projection is used.

Symetrie: This keyword is used to designate a symmetry condition concerning the speed at the boundary called *bord* (normal speed at the edge equal to zero and tangential speed gradient at the edge equal to zero).

Periodique: This keyword is used to indicate the fact that the horizontal speed inlet values are the same as the outlet speed values, at every moment. As regards meshing, the inlet and outlet edges bear the same name.

Paroi_Knudsen_non_negligeable

New boundary condition for number of Knudsen (Kn) above 0.001 where slip-flow condition appears: the velocity near the wall depends on the shear stress :
Kn=l/L with l is the mean-free-path of the molecules and L a characteristic length scale.

U(y=0)-Uwall=k(dU/dY)

Where k is a coefficient given by several laws:
Mawxell: k=(2-s)*l/s
Bestok&Karniadakis:k=(2-s)/s*L*Kn/(1+Kn)
Xue&Fan:k=(2-s)/s*L*tanh(Kn)
s is a value between 0 and 2 named accomodation coefficient. s=1 seems a good value.

Example:
boundary_conditions {
…………………..
    Bord1 **Paroi_Knudsen_non_negligeable** vitesse_paroi Champ_front_uniforme 3 0. 0. 0. k Champ_front_uniforme 1 0.1
}
Warning:

The keyword is available for VDF calculation only for the moment.

Paroi_rugueuse

New wall boundary condition for turbulent calculation to change the roughness constant value *Erugu* of a wall (default value 9.11). This keyword change the law for a smooth wall. It adds a constant which depends of a dimensionless roughness height:

$k_s^+ = \frac{u_* k_s}{\nu}$ (where k_s is the equivalent sand-grain roughness height)

We have:

$$u^+ = \frac{1}{\kappa} \ln\left(\frac{3.93}{k_s^+} E y^+\right)$$

This law may be compared to the law for a smooth wall :

$$u^+ = \frac{1}{\kappa} \ln(E y^+)$$

With K=0.415 (Von Karman constant), E = 9.11 (*Erugu* value for a smooth law). To deal with this model an atmospheric boundary layer with a velocity profile :

$U(y) = \frac{u_*}{\kappa} \ln\left(\frac{y}{y_0}\right)$, (where y_0 is the aerodynamic roughness length)

You may use the law with:

$Erugu = \frac{\nu}{y_0 \cdot u_*}$ (where $k_s = 3.93 E y_0$)

2.13.2THERMAL BOUNDARY CONDITIONS

Boundary conditions that are not specific to discretization:

[Bord **Frontiere_ouverte_temperature_imposee** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_temperature_imposee_rayo_semi_transp** boundary_field_type *bloc_lecture_champ*]

[Bord **Frontiere_ouverte T_ext** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_rayo_semi_transp T_Ext** boundary_field_type *bloc_lecture_champ*]
[Bord **Frontiere_ouverte_rayo_transp T_Ext** boundary_field_type *bloc_lecture_champ*]

[Bord **Symetrie**]
[Bord **Periodique**]

[Bord **Paroi_adiabatique**]
[Bord **Paroi_decalee_Robin** { **delta** value }]
[Bord **Paroi_flux_impose** boundary_field_type *bloc_lecture_champ*]
[Bord **Paroi_temperature_imposee** boundary_field_type *bloc_lecture_champ*]
[Bord **Paroi_echange_externe_impose H_imp** boundary_field_type *bloc_lecture_champ* **T_ext** boundary_field_type *bloc_lecture_champ*]
[Bord **Paroi_contact** problem_name Bord]
[Bord **Paroi_contact_fictif** problem_name Bord thermal_conductivity thickness]

Bord: name of the edge where the boundary condition applies.

boundary_field_type: boundary field type.

Frontiere_ouverte_temperature_imposee: This keyword is used to set an imposed temperature condition at the open boundary called *bord* (in the case of fluid inlet). This condition must be associated with an imposed inlet speed condition. The imposed temperature value is expressed in °C or K.

Frontiere_ouverte_temperature_imposee_rayo_semi_transp: Keyword to apply the same condition for a radiation problem with semi transparent gas.

Frontiere_ouverte: This keyword is used to set a boundary outlet temperature condition on the boundary called *bord* (diffusion flux zero). This condition must be associated with a boundary outlet hydraulic condition.

Frontiere_ouverte_rayo_semi_transp: Keyword to apply the same condition for a radiation problem with semi transparent gas.

Frontiere_ouverte_rayo_transp: Keyword to apply the same condition for a radiation problem with transparent gas.

T_ext: This keyword is used to define the temperature at the boundary.

Symetrie: This keyword is used to set a symmetry condition on temperature on the boundary named *bord*.

Periodique: This keyword is used to set a periodic condition on temperature. The two edges dealing with this periodic condition bear the same name.

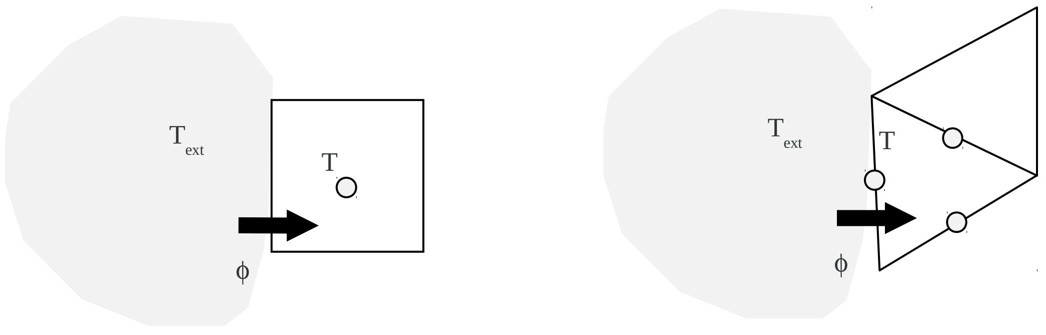
Paroi_adiabatique: This keyword is used to refer to a normal zero flux condition at the wall called *bord* .

Paroi_decalee_Robin: This keyword is identical to the one described here: 2.13.1.

Paroi_flux_impose: This keyword is used to refer to a normal flux condition at the wall called *bord*. The surface area of the flux (W.m^{-2}) is imposed at the boundary according to the following convention: <u>a positive flux is a flux that enters into the domain according to convention</u>.

Paroi_temperature_imposee: This keyword is used to refer to an imposed temperature condition at the wall called *bord*.

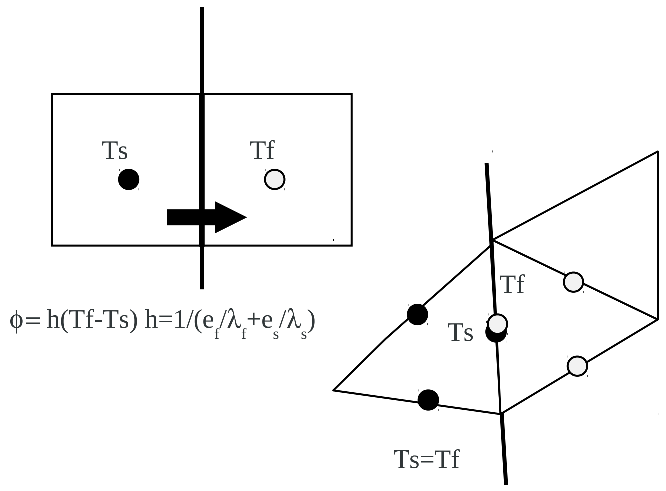
Paroi_echange_externe_impose: This keyword is used to set an external type exchange condition with a heat exchange coefficient and an imposed external temperature.



ϕ = h (T -T_{ext}) where 1/h = 1/h_{imp} + e/λ in VDF

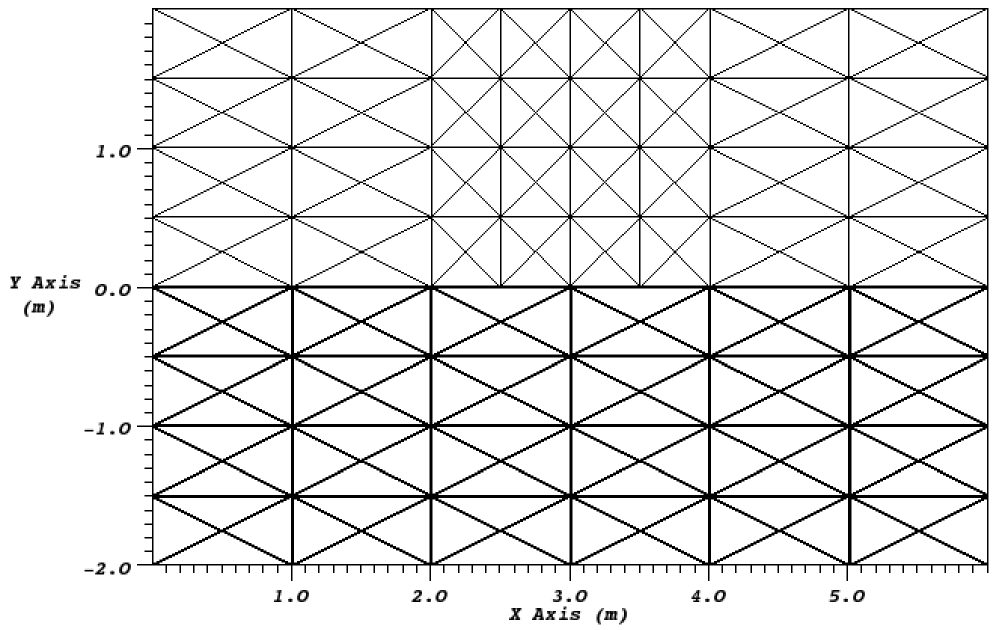
ϕ = h_{imp} (T -T_{ext}) in VEF

Paroi_contact : This keyword is used to set a thermal condition between two domains. Important: the name of the boundaries in the two domains should be the same. (Warning: there is also an old limitation not yet fixed on the sequential algorithm in VDF to detect the matching faces on the two boundaries: faces should be ordered in the same way). The kind of condition depends on the discretization. In VDF, it is a heat exchange condition, and in VEF, a temperature condition:

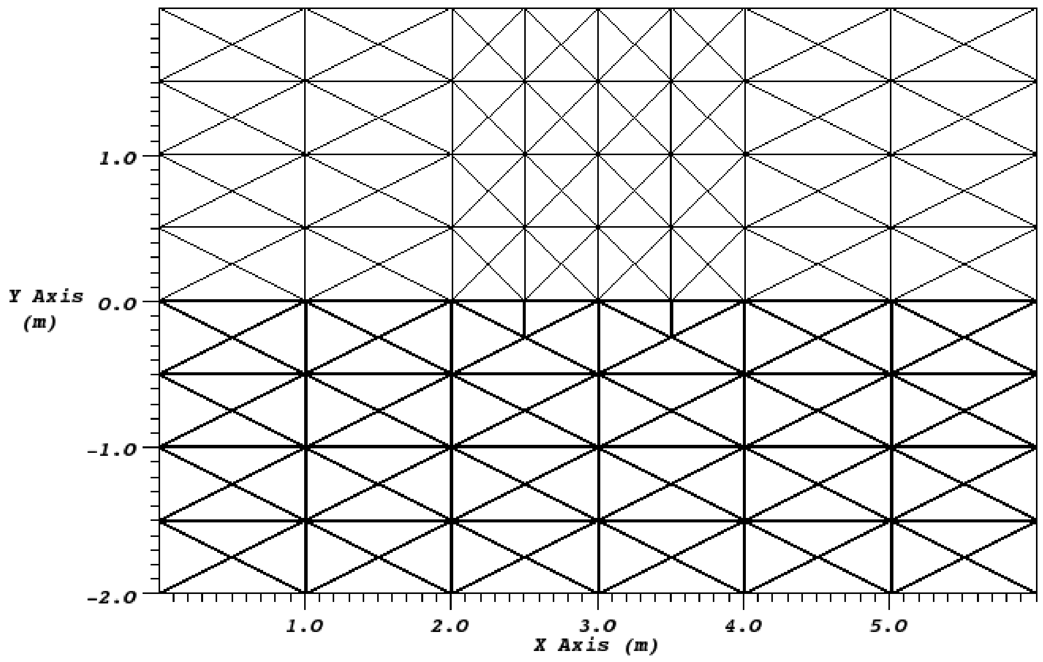


Such a coupling requires coincident meshes for the moment. In case of non-coincident meshes, run is stopped and two external files are automatically generated in VEF (*connectivity_failed_boundary_name* and *connectivity_failed_pb_name.med*). In 2D, the keyword **Decouper_bord_coincident** associated to the *connectivity_failed_boundary_name* file allows to generate a new coincident mesh.

Example :
dimension 2
Domaine solide
Read _file solide *solide.geom*
Decouper_bord_coincident solide boundary_name
Ecrire_fichier solide *new_solide.geom*



The mesh before (solide in the *solide.geom* file)



The mesh after (solide in the *new_solide.geom* file)

In 3D, for a first preliminary cut domain with HOMARD (fluid for instance), the second problem associated to *pb_name* (solide in a fluid/solid coupling problem) has to be submitted to HOMARD cutting procedure with *connectivity_failed_pb_name.med*.

Such a procedure works as while the primary refined mesh (fluid in our example) impacts the fluid/solid interface with a compact shape as described below (values 2 or 4 indicates the number of division from primary faces obtained in fluid domain at the interface after HOMARD cutting):

2-2-2-2-2-2 2-4-4-4-4-4-2 2-4-4-4-4-2 2-2-2 2-2-2-2-2 2-4-2 2-2 OK	2-2 2-2-2 2-4-2 2-2 2-2 2-2 NOT OK

Paroi_contact_fictif : This keyword is derivated from **paroi_contact** and is especially dedicated to compute coupled fluid/solid/fluid problem in case of thin material. Thanks to this option, solid is considered as a fictitious media (no mesh, no domain associated), and coupling is performed by considering instantaneous thermal equilibrium in it (for the moment).
problem_name : name of the problem
Bord : boundary name of the remote problem which should be the same than the local name
thermal_conductivity : thermal conductivity
thickness : thickness of the fictitious media

Boundary conditions specific toVDF discretization:

[Bord **Paroi_echange_global_impose** **H_imp** boundary_field_type *bloc_lecture_champ* **T_ext** boundary_field_type *bloc_lecture_champ*]

[Bord **Paroi_Echange_externe_impose_rayo_semi_transp** **H_imp** boundary_field_type *bloc_lecture_champ* **T_ext** boundary_field_type *bloc_lecture_champ*]

[Bord **Paroi_Echange_contact_VDF** pb2 Bord2 **temperature** val_h_contact]
[Bord **Echange_contact_rayo_transp_VDF** pb2 Bord2 **temperature** temp]
[Bord **Paroi_echange_contact_correlation_VDF {** **dir** integer **Tinf** double **Tsup** double
lambda function **rho** function **Cp** double **mu** function **debit** double **Dh** double **dt_impr** double
Nu function **volume** function **[Reprise_correlation] }**]

Paroi_echange_global_impose: This keyword is used to set a global type exchange condition (internal) that is to say that diffusion on the first fluid mesh is not taken into consideration.
H_imp: This is a keyword used to define the global exchange coefficient value. The global exchange coefficient value is expressed in W.m^{-2}.K^{-1}.
T_ext: This is a keyword used to define the external temperature value. The external temperature value is expressed in °C or K.

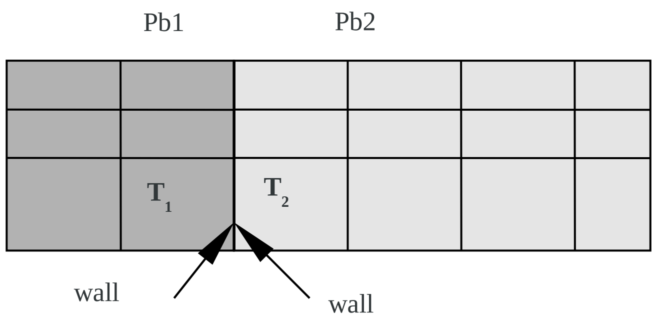
$$\phi = h_{imp} (T - T_{ext})$$

Paroi_Echange_externe_impose_rayo_semi_transp: This keyword is used to set the same condition but for a coupled problem with radiation in semi transparent gas.
H_imp: This is a keyword used to define the external exchange coefficient value. The external exchange coefficient value is expressed in W.m^{-2}.K^{-1}.
T_ext: This is a keyword used to define the external temperature value. *vrel2*: external temperature value (°C or K).

In the case of a coupling, one of the following boundary condition types must be used:

Paroi_echange_contact_VDF to model the heat flux between two problems. Important: the name of the boundaries in the two problems should be the same.

An example of using this boundary condition:



The following instruction will be found in the pb1 read block:

wall **Paroi_Echange_contact_VDF** pb2 wall **temperature** val_h_contact

The following instruction will be found in the pb2 read block

wall **Paroi_Echange_contact_VDF** pb1 wall **temperature** val_h_contact

val_h_contact: this corresponds to the value assigned to a coefficient (expressed in $W.K^{-1}m^{-2}$) that characterises the contact between the two mediums. In order to model perfect contact, *val_h_contact* must be taken to be infinite. This value must obviously be the same in both the pb1 and pb2 blocks.

The surface thermal flux exchanged between the two mediums is represented by:

$\phi = h (T_1-T_2)$ where $1/ h = d_1 / \lambda_1 + 1/ \text{val\_h\_contact} + d_2 / \lambda_2$

where d_i: distance between the node where T_i and the wall is found.

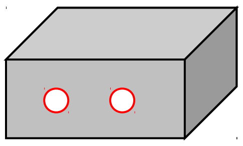
Echange_Contact_Rayo_Transp_VDF: This keyword is used to set an exchange boundary condition in VDF between the fluid and the solid for a problem coupled with radiation. Without radiation, it is the equivalent of the Paroi_Echange_contact_VDF exchange condition. Refer to the definition of the latter for identical syntax.

Bord1, *Bord2*: Names of the edges in contact.

Pb: Name of the opposed problem of which Bord2 (edge2) is one of the domain boundaries.

temperature *val_h_contact*: Keyword used to specify the value of a coefficient (expressed in W.K^{-1}m^{-2}) which characterises contact between the two mediums. To model perfect contact, *val_h_contact* must be taken to be infinite. This value must obviously be the same in both the pb1 and pb2 blocks

Paroi_echange_contact_correlation_VDF : This keyword is used to define a thermal hydraulical 1D model which will apply to a boundary of 2D or 3D domain.
Example: Conduction will be calculated in the 3D gray domain, whereas 1D model will apply in the channel. The boundary condition applying on to the red boundary are defined with the following parameters:



dir integer : Direction (0 : axis X, 1 : axis Y, 2 : Axis Z) of the 1D model
Tinf double : Inlet fluid temperature of the 1D model (°C or K)
Tsup double : Outlet fluid temperature of the 1D model (°C or K)
lambda function : Thermal conductivity of the fluid (W.m^{-1}.K^{-1})
rho function : Mass density of the fluid (kg.m^{-3}) which may be a function of the temperature T
Cp double : Calorific capacity value at a constant pressure of the fluid (J.kg^{-1}.K^{-1})
mu function : Dynamic viscosity of the fluid (kg.m^{-1}.s^{-1}) which may be a function of the temperature T
debit double : Surface flow rate (kg.s^{-1}.m^{-2}) of the fluid into the channel
Dh double : Hydraulic diameter (m) of the channel
dt_impr double : Printing period in *name_of_data_file_time.dat* files of the 1D model results
Nu function : Nusselt number which may be a function of the Reynolds number (Re) and the Prandtl number (Pr)
volume function : Exact volume of the 1D domain (m3) which may be a function of the hydraulic diameter (Dh) and the lateral surface (S) of the meshed boundary
Reprise_correlation Optional keyword in the case of a restarting calculation with this correlation

Warning: For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche.

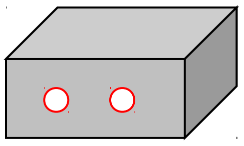
Example:

```
        INTERFACE   Paroi_Echange_contact_Correlation_VDF
          {
                    dir 2
                    Tinf 1180
                    Tsup 751
                    lambda 2.774e-3*T^0.701
                    rho 90e5/(2077.22*T+90e5*(9.5e-4+9.5e-4/(1-3.4e-2*T)+2.74e-3/(1+9.4e-4*T)))
                    Cp 5193
                    mu 3.953e-7*T^0.687
                    debit -109.5
                    Dh 0.016
                    dt_impr 0.1
                    Nu 0.023*Re^0.8*Pr^(1./3.)
                    volume S*Dh*3.1415927/16
          }
```

Boundary conditions specific toVEF discretization:

[Bord **Paroi_echange_contact_correlation_VEF { dir** integer **Tinf** double **Tsup** double
lambda function **rho** function **Cp** double **mu** function **debit** double **Dh** function **dt_impr** double
Nu function **surface** function **N** integer **xinf** double **xsup** double **[Reprise_correlation] }**]

Paroi_echange_contact_correlation_VEF : This keyword is used to define a thermal hydraulical 1D model which will apply to a boundary of 2D or 3D domain.
It has the same options of **Paroi_echange_contact_correlation_VDF** keyword minus the **volume** option and plus the following options:



Dh function : Hydraulic diameter may be a function f(x) with x position along the 1D axis (xinf <= x <= xsup)
surface function : Section surface of the channel which may be function f(Dh,x) of the hydraulic diameter (Dh) and x position along the 1D axis (xinf <= x <= xsup)
N integer: Number of 1D cells of the 1D mesh
xinf double: Position of the inlet of the 1D mesh on the axis direction
xsup double: Position of the outlet of the 1D mesh on the axis direction
Reprise_correlation Optional keyword in the case of a restarting calculation whith this correlation

Warning: For parallel calculation, the only possible partition will be according the axis of the model with the keyword Tranche_geom.

Example:

```
        INTERFACE   Paroi_Echange_contact_Correlation_VDF
```

```
{
        dir 2
        Tinf 1180
        Tsup 751
        lambda 2.774e-3*T^0.701
        rho 90e5/(2077.22*T+90e5*(9.5e-4+9.5e-4/(1-3.4e-2*T)+2.74e-3/(1+9.4e-4*T)))
        Cp 5193
        mu 3.953e-7*T^0.687
        debit -109.5
        Dh 0.016
        dt_impr 0.1
        Nu 0.023*Re^0.8*Pr^(1./3.)
        Surface  3.1415/4*Dh*Dh
        N 20
        xinf 0.
        xsup 0.8
}
```

2.13.3BOUNDARY CONDITIONS IN CONCENTRATION

[Bord **Frontiere_ouverte_concentration_imposee** boundary_field_type *bloc_lecture_champ_front*]
[Bord **Frontiere_ouverte C_ext** boundary_field_type *bloc_lecture_champ_front*]
[Bord **Paroi**]
[Bord **Paroi_flux_impose** boundary_field_type *bloc_lecture_champ_front*]
[Bord **Symetrie**]
[Bord **Periodique**]

Bord: name of the edge where the boundary condition is applied.

Frontiere_ouverte_concentration_imposee: Keyword used to set an imposed concentration condition at an open boundary called *bord* (situation corresponding to a fluid inlet). This condition must be associated with an imposed inlet speed condition.

Frontiere_ouverte: This keyword is used to refer to a boundary outlet condition on the boundary called *bord* (zero diffusion flux). This condition must be associated with a boundary outlet hydraulic condition.
C_ext: This keyword is used to describe concentration at a boundary.

Paroi: This keyword is used to refer to an impermeability condition at a wall called *bord* (standard flux zero). This condition must be associated with a wall type hydraulic condition.

Paroi_flux_impose: This keyword is used to set a flux boundary condition. If U is the unit of the concentration C, the flux value (D*gradC.n) is given in $ms^{-1}U$ and should be a positive quantity if flux is oriented outside to inside the domain.

Symetrie: This is a keyword used to refer to a symmetrical condition applied to constituent concentration at the boundary called *bord*.

Periodique: This keyword is used to set a periodic condition on temperature. The two edges dealing with this periodic condition bear the same name.

2.13.4BOUNDARY CONDITIONS FOR TURBULENCE

[Bord **Frontiere_ouverte_K_Eps_impose** boundary_field_type *bloc_lecture_champ_front*]
[Bord **Frontiere_ouverte K_Eps_ext** boundary_field_type *bloc_lecture_champ_front*]
[Bord **Paroi**]
[Bord **Symetrie**]
[Bord **Periodique**]
[Bord **Frontiere_ouverte_Fluctu_Temperature_imposee** boundary_field_type *bloc_lecture_champ_front*]
[Bord **Frontiere_ouverte Fluctu_Temperature_ext** boundary_field_type *bloc_lecture_champ_front*]
[Bord **Frontiere_ouverte_Flux_Chaleur_Turbulente_imposee** boundary_field_type *bloc_lecture_champ_front*]
[Bord **Frontiere_ouverte Flux_Chaleur_Turb_ext** boundary_field_type *bloc_lecture_champ_front*]

Bord: name of the edge where the boundary condition applies.

Frontiere_ouverte_K_eps_impose: Keyword used to refer to a turbulence condition imposed on an open boundary called Bord (this situation corresponds to a fluid inlet). This condition must be associated with an imposed inlet speed condition.

Frontiere_ouverte: Keyword used to refer to a boundary outlet condition on the boundary called Bord (zero diffusion flux). This condition must be associated with a boundary outlet hydraulic condition.

K_Eps_ext: This is a keyword used to define the kinetic energy and turbulent dissipation rate for the boundary.

The kinetic energy is expressed in $m^2.s^{-2}$.

The turbulent dissipation rate is expressed in $m^2.s^{-3}$.

Paroi: This is a keyword used to refer to a zero flux condition at the wall called Bord (ε null and k standard flux). This condition must be associated with a paroi (wall) type hydraulic condition. Caution: this keyword should not be confused with the wall laws which are applicable to static walls when no turbulence condition is applied to them.

Symetrie: This keyword is used to refer to a symmetry condition for k and ε on the boundary called Bord.

Periodique: This keyword is used to set a periodic boundary condition for k and ε on the boundary called Bord.

2.14 HYDRAULIC SOURCE TERMS

To introduce a source term into an equation, add the following line into the block defining the equation. The list of source keyword is described below.

Sources { source_keyword }

To introduce several source terms into the same equation, the blocks corresponding to the various terms need to be separated by a comma:

Sources { source_keyword1 , source_keyword2 , … }

2.14.1 PRESSURE LOSS TYPE SOURCE TERMS (VDF DISCRETIZATION)

Perte_Charge_Reguliere type_perte_charge *bloc_definition_pertes_charges*

Perte_Charge_Reguliere: source term modelling the presence of a bundle of tubes in a flow.

type_perte_charge: there are two types of options available: **Longitudinale** or **Transversale**: the first may be used to define pressure loss in the direction of the tube bundle and the second to define the pressure loss in the direction perpendicular to the tube bundle.

The two types of pressure loss definition blocks are as follows:

Perte_Charge_Reguliere **Longitudinale** direction_application valeur_diametre_hydraulique **A** val **B** val nom_sous_zone *direction_application*: keyword which may be selected from among **X**, **Y** or **Z**.

valeur_diamètre_hydraulique: tube bundle hydraulic diameter value. This value is expressed in m.

A val **B** val: These keywords are used to set law coefficient values for the coefficient of regular pressure losses which are written as follows:

$$\Lambda=A.Re^{-B}$$

nom_sous_zone: name of the sub area occupied by the tube bundle. A **Sous_Zone** (Sub-area) type object called *nom_sous_zone* (sub_area_name) should have been previously created (refer to 2.3.23).

```
Perte_Charge_Reguliere Transversale
direction_application  valeur_pas_faisceau  d valeur_d  A  val  B  val  nom_sous_zone
```

direction_application: keyword which may be selected from among **X**, **Y** or **Z**.

valeur_pas_faisceau: value of the tube bundle step.

valeur_d: value of the tube external diameter

A val **B** val: These keywords are used to set the law coefficient values for the coefficient of regular pressure losses which is written as follows:

$$\Lambda=A.Re^{-B}$$

nom_sous_zone: name of the sub-area occupied by the tube bundle. A **Sous_Zone** (Sub-area) type object called *nom_sous_zone* (sub_area_name) should have been previously created (refer to 0).

2.14.2PRESSURE LOSS TYPE SOURCE TERMS (VEF DISCRETIZATION)

```
Perte_Charge_Directionnelle { diam_hydr  field_type
            lambda  function
      direction  field_type
            [ sous_zone  name ] }
```

Perte_Charge_Directionnelle: Keyword for directional pressure loss.
diam_hydr field_type : Hydraulic diameter value.
lambda function : Function for loss coefficient which may be Reynolds dependant (Ex: 64/Re)
direction field_type : Field which indicates the direction of the pressure losse.
sous_zone name : Optional sub-zone where pressure loss applies.

```
Perte_Charge_Isotrope { diam_hydr  field_type
            lambda  function
            [ sous_zone  name ] }
```

Perte_Charge_Isotrope : Keyword for isotropic pressure loss. Same parameters as **Perte_Charge_Directionnelle** except **direction** keyword.

```
Perte_Charge_Anisotrope { diam_hydr  field_type
            lambda  function
          lambda_ortho function
          direction  field_type
            [ sous_zone  name ] }
```

Perte_Charge_Anisotrope : Keyword for anisotropic pressure loss. Same parameters as **Perte_Charge_Directionnelle** plus:
lambda_ortho function: Function for loss coefficient in transverse direction which may be Reynolds dependant (Ex: 64/Re)

```
Perte_Charge_Circulaire {
        diam_hydr  field_type
        dima_hydr_ortho field_type
            lambda  function
         lambda_ortho function
         direction  field_type
            [ sous_zone  name ] }
```

Perte_Charge_Circulaire : Keyword as anisotropic pressure loss (**Perte_Charge_Anisotrope**) but with 3 Reynolds numbers:

$$\mathrm{Re}\_tot = \frac{\|U\| D}{\nu}$$

$$Re\_long = \frac{U.nD}{\nu}$$

$$Re\_ortho = \frac{\|U - U.nn\| Do}{\nu}$$

Defined thanks:

U: Velocity vector

n : Vector direction of the pressure loss given by the **direction** option

D : Hydraulic diameter given the **diam_hydr** option

Do : Transverse hydraulic diameter given the **diam_hydr_ortho** option

ν: Kinematic viscosity

lambda function : Function f(Re_tot, Re_long, t, x, y, z) for loss coefficient in the longitudinal direction

lambda_ortho function: Function f(Re_tot, Re_ortho, t, x, y, z) for loss coefficient in transverse direction

2.14.3PRESSURE LOSS TYPE SOURCE TERMS (VDF OR VEF DISCRETIZATIONS)

Perte_Charge_Singuliere KX | KY | KZ coefficient_value { [X | Y | Z = location *subzone_name]* | Surface *dom* }

Perte_Charge_Singuliere: source term that is used to model a pressure loss over a surface area (transition through a grid, sudden enlargement).

The surface can be defined:

- either by the faces of elements located on the intersection of a subzone named *subzone_name* and a X,Y, or Z plane located at X,Y or Z = location.
- or by the faces of the domain 2D named *dom*. This option is only available in 3D and for VEF discretization. The surface *dom* may be an inner surface extracted via Extraire_Surface keyword or may be a domain read in Med file.

KX, **KY** or **KZ** keyword specify the directional pressure loss coefficient_value for respectively a X, Y or Z direction.

Example : **sources** { **Perte_Charge_Singuliere KX** 0.5 { **X =** 0.35 sous_zone_toto } }

2.14.4MOMENTUM SOURCE TERMS

Source_Qdm field_type *field_description*

Momentum source term in the Navier Stokes equation.

Canal_perio { **bord** boundary_name [**h** value] [**coeff** value] [**debit_impose** double] }

Momentum source term to maintain flow rate :

Canal_perio: Keyword for the source term.
bord boundary_name : The name of the (periodic) boundary normal to the flow direction.
h value: Half heigth of the channel. Optional.
coeff value: Damping coefficient (optional, default value is 10).
debit_impose double : Optional option to specify the aimed flow rate Q(0). If not used, Q(0) is computed by the code after the projection phase, where velocity initial conditions are slighlty changed to verify incompressibility.

The expression of the source term is:

S(t) = (2*(Q(0) - Q(t))-(Q(0)-Q(t-dt))/(coeff*dt*area)

Where:
coeff=damping coefficient
area=area of the periodic boundary
Q(t)=flow rate at time t
dt=time step

Three files will be created during calculation on a datafile named DataFile.data. The first file contains the flow rate evolution. The second file is useful for restarting a calculation with the flow rate of the previous stopped calculation, and the last one contains the pressure gradient evolution:
-DataFile_Channel_Flow_Rate_ProblemName_BoundaryName
-DataFile_Channel_Flow_Rate_repr_ProblemName_BoundaryName
-DataFile_Pressure_Gradient_ProblemName_BoundaryName

Sources_Qdm_lambdaup { **lambda** value
[**lambda_min** value]
[**lambda_max** value]
[**ubar_umprim_cible** value] }

This source term is a dissipative term which is intended to minimise the energy associated to non-conform scales u' (responsible for spurious oscillations in some cases). The equation for these scales can be seen as:
du'/dt= -lambda. u' + grad P'
where -lambda. u' represents the dissipative term, with lambda = a/Delta t
Optional values **lambda_main** and **lambda_max** give the minimal and maximal value for lambda whereas **ubar_umprim_cible** is a threshold in the lambda algorithm calculation (by default 0.1).

For Crank-Nicholson temporal scheme, recommended value for a is 2.

Sources { **Source_Qdm_lambdaup** { **lambda** 2. } }

Remark:
This method requires to define a filtering operator : see **solveur_bar**

Source_Robin N boundary_name_1 ... boundary_name_N

This source term should be used when a **Paroi_decalee_Robin** boundary condition is set in a hydraulic equation. The source term will be applied on the N specified boundaries. To post-process the values of tauw, u_tau and Reynolds_tau into the files tauw_robin.dat, *reynolds_tau_robin.dat* and *u_tau_robin.dat*, you must add a block "Traitement_particulier { canal { } }" see 2.6.2.

Acceleration { [**vitesse** time_field] **acceleration** time_field **omega** time_field **domegadt** time_field **centre_rotation** time_field [**option terme_complet**|**coriolis_seul**|**entrainement_seul**] **}**

Momentum source term to take in account the forces due to rotation or translation of a non Galilean referential R' (centre 0') into the Galilean referential R (centre 0).

acceleration time_field: Keyword for the acceleration of the referential R' into the R referential (d^2OO'/dt^2 term [$m.s^{-2}$]). time_field is a time dependant field (eg: **Champ_Fonc_t**).

vitesse time_field: Optional keyword for the velocity of the referential R' into the R referential (dOO'/dt term [m.s^{-1}]). The velocity is mandatory when you want to print the total cinetic energy into the non-mobile Galilean referential R (see **Ec_dans_repere_fixe** keyword).
omega time_field: Keyword for a rotation of the referential R' into the R referential [rad.s^{-1}]. time_field is a 3D time dependant field specified for example by a **Champ_Fonc_t** keyword. The time_field field should have 3 components even in 2D (In 2D: 0 0 omega).
domegadt time_field: Keyword to define the time derivative of the previous rotation [rad.s^{-2}]. Should be zero if the rotation is constant. The time_field field should have 3 components even in 2D (In 2D: 0 0 domegadt).
centre_rotation time_field: Keyword to specify the centre of rotation (expressed in R' coordinates) of R' into R (if the domain rotates with the R' referential, the centre of rotation is 0'=(0,0,0)). The time_field should have 2 or 3 components according the dimension 2 or 3.
option terme_complet|coriolis_seul|entrainement_seul : Optional keyword to specify the kind of calculation. **terme_complet** (default option) will calculate both the Coriolis and centrifugal forces, **coriolis_seul** will calculate the first one only, **entrainement_seul** will calculate the second one only.

The source term can be reported in results files with the **Acceleration_terme_source** keyword.

2.14.5PORIUS MEDIA SOURCE TERMS

Darcy source term with constant permeability :

```
Darcy { modele_K   K_constant  { valeur value } }
```

Darcy source term with Ergun's law permeability:

```
Darcy { porosite value modele_K  ErgunDarcy  { diametre value } }
```

This keyword is used for calculation in a porius media with source term of Darcy -nu/K*V. This keyword must be used with a **permeability model**. For the moment there are two models :permeability constant or Ergun's law. Darcy source term is **available for quasi compressible** calculation.A new keyword is aded for porosity (**porosite**)

Forcheimer source term with Ergun's law :

```
Forchheimer { porosite value Cf value modele_K ErgunForchheimer { diametre value } }
```

Forcheimer source term with the constant law :

```
Forchheimer { Cf  value  modele_K  K_constant  { valeur value } }
```

This keyword makes it possible to add the source term of Forchheimer -Cf/sqrt(K)*V2 in the Navier Stokes equations. Like the term of Darcy, we must precise a permeability model : constant or Ergun's law. Moreover we can give the constant Cf : by default its value is 1. Forchheimer source term is **available also for quasi compressible** calculation. A new keyword is aded for porosity (**porosite**)

2.14.6BOUSSINESQ TYPE SOURCE TERMS

```
Boussinesq_temperature { T0 vrel  [ verif_boussinesq 0|1 ] }
```

Boussinesq_temperature: Keyword used to describe a source term that couples the movement quantity equation and energy equation with the Boussinesq hypothesis.

T0: Keyword used to describe the reference temperature.

vrel: reference temperature value (°C or K). It can also be a time dependant function since the 1.6.6 version.

verif_boussinesq: Optional keyword to check (1) or not (0) the reference temperature in comparison with the mean temperature value in the domain. It is set to 1 by default.

```
Boussinesq_concentration { C0  N  C0(1) .... C0(N)  [ verif_boussinesq 0|1] }
```

Boussinesq_concentration: Keyword used to describe a source term that couples the movement quantity equation and constituent transportation equation with the Boussinesq hypothesis

C0: Keyword used to describe the reference concentration.
N : Number of constituents
C0(i) : Values for reference concentration for each constituent (may be time dependant since the 1.6.8 version).

verif_boussinesq: Optional keyword to check (1) or not (0) the reference concentration in comparison with the mean concentration value in the domain. It is set to 1 by default.

2.14.7CORIOLIS

Coriolis { **omega** value }

Keyword for a Coriolis term in hydraulic equation.

Example: (See also the test case Coriolis)
Sources { **Coriolis** { **omega** 2 0.1 } }

Warning: Only available in VDF.

2.15SCALAR SOURCE TERMS

Source_Th_TdivU

This term source is dedicated for any scalar (called "T") transportation. Coupled with upwind ("amont") or muscl scheme, this term gives for final expression of convection : div(U.T)-T.div(U)=U.grad(T)
This ensures, in incompressible flow when divergence free is badly resolved, to stay in a better way in the physical boundaries.

Warning: Only available in VEF discretization.

2.15.1THERMAL SOURCE TERMS

Puissance_thermique field_type *bloc_lecture_champ*

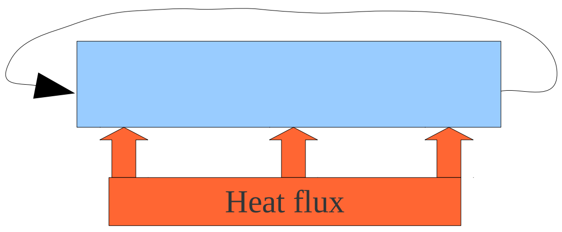
Puissance_thermique: This keyword is used to define a source term corresponding to a volume power release in the energy equation.

field_type : thermal power field type. To impose a volume power on a domain sub-area, the **Champ_Uniforme_Morceaux (partly_uniform_field)** type must be used.

Warning: The volume thermal power is expressed in W.m^{-3} in 3D. It is a power per volume unit (in a porous media, it is a power per fluid volume unit).

Canal_perio { **bord** boundary_name }

Energy source term to add in a periodic channel with heat flux boundary conditions:



Canal_perio: Keyword for the source term.
bord boundary_name : The name of the (periodic) boundary normal to the flow direction.

The expression of the implemented source term is:

S(x,y,z,t) = -V(x,y,z,t)*ImposedHeatFlux/(ρ*Cp*Volume*ChannelBulkVelocity)

Where:
V(x,y,z,t)=velocity according to the periodic direction
Volume=Volume of the fluid
ImposedHeatFlux=Heat flux imposed on the periodic channel walls
ChannelBulkVelocity= Bulk velocity=Flow rate / area of the periodic boundary
Warning: Available in VEF only in the 1.6.8 version.

```
Source_Robin_Scalaire  N boundary_name_1   temp_wall_value1 ...
                         boundary_name_N   temp_wall_valueN
                         dt_impr
```

This source term should be used when a **Paroi_decalee_Robin** boundary condition is set in a an energy equation. The source term will be applied on the N specified boundaries. The values temp_wall_valueI are the temperature specified on the Ith boundary. The last value dt_impr is a printing period which is mandatory to specify in the data file but has no effect yet.

2.15.2 GENERIC SOURCE TERM

```
Source_Generique   field_type   bloc_lecture_champ
```

Source_Generique : This keyword is used to define a source term depending on some discrete fields of the problem and (or) analytic expression. It is expressed by the way of a generic field usually used for post-processing.

field_type : generic field type (see §2.19.3).

2.16 TURBULENCE MODELS

The turbulence models described hereunder may only be used in discretization. A turbulence model is selected in the hydraulic equation. For scalar convection diffusion equations coupled with the hydraulic equation, a turbulence model is selected as a function of that which was selected for the hydraulic equation.

2.16.1 MODELS FOR NAVIER STOKES EQUATIONS

2.16.1.1 SUB-GRID SCALE MODELS

```
Modele_turbulence model
{
    [ Cs valeur ]
    [ longueur_maille Characteristic_length ]
    Turbulence_paroi  law…
    [ Correction_visco_turb_pour_controle_pas_de_temps ]
   [ Correction_visco_turb_pour_controle_pas_de_temps_parametre value ]
}
```

Turbulence_paroi: This keyword is used to define the wall turbulence model equations. Refer to 2.16.3.1.

Cs value: This is an optional keyword and the value is used to set the constant used in the model. (This is currently only valid for Smagorinsky models and it is set to 0.18 by default.)

Correction_visco_turb_pour_controle_pas_de_temps : Keyword to set a limitation to low time steps due to high values of turbulent vicosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the **corr_visco_turb** field which is the correction of turbulent viscosity: it should be 1. on the whole domain.

Correction_visco_turb_pour_controle_pas_de_temps_parametre value : Keyword as **Correction_visco_turb_pour_controle_pas_de_temps** to set a limitation to high values of turbulent viscosity. The specified value is the desired ratio between diffusive time step and convective time step. The value should be greater than 0 and lesser or equal to 1. If set to 1, it is equivalent to the **Correction_visco_turb_pour_controle_pas_de_temps** keyword.

Characteristic_length: different ways to calculate the characteristic length may be specified :

volume : (by default) characteristic length is based on the cubic square of volume cells. (To avoid discontinuities of this quantity in VEF from a cell to another, a smoothing procedure is applied)
volume_sans_lissage : for VEF only - the same as previously without smoothing procedure
Scotti : **volume** * Scotti's correction to take into account the stretching of the cell in case of an-isotropic meshes.
arete : for VEF only - characteristic length relies on the max edge (+ smoothing procedure)

The model keyword may be:

Sous_maille: This keyword is entered to use a structure sub-grid function model.
Sous_maille_selectif: This keyword is entered to use the selective structure sub-grid function model. This model is derived from the previous model: the only difference is that a filter is applied to the structure function.

The two last keywords for LES models has new options:
formulation_a_nb_points 4 dir1 dir2 : The structure fonction is calculated on four points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homegeneity planes. Example for channel flows, planes parallel to the walls.
formulation_a_nb_points 6 : By default, the structure fonction is calculated on six points.

Sous_maille_axi: This keyword is entered to indicate usage of the structure sub-grid function turbulence model available in cylindrical co-ordinates.

Sous_maille_Smago: This keyword is used to indicate that the Smagorinsky sub-grid turbulence model should be used.

Nut=Cs*Cs*ℓ*ℓ*sqrt(2*S*S)   (Cs=0.18 by default)

Sous_maille_smago_dyn: This keyword is used to indicate that the dynamic sub-grid model should be used (available in VDF discretization only). Options are available :

```
Modele_turbulence Sous_maille_smago_dyn
{
    stabilise
    [ 6_points ]
    [ plans_paralleles nb_points integer ]
    [ moy_euler ]
    [ moy_lagrange ]
    Turbulence_paroi  law…
    [ Correction_visco_turb_pour_controle_pas_de_temps ]
}
```

Sous_maille_smago_filtre: This keyword is used to indicate that the Smagorinsky sub-grid turbulence model should be used with low-filter.

Sous_maille_selectif_mod: Keyword with this model (in VDF only).

Sous_maille_1elt_selectif_mod: Keyword for VEF calculation with this model.
THI ki kc: For homogeneous isotropic turbulence (THI), two integers ki and kc are needed in VDF (not in VEF).
Canal h dir_faces_paroi: For a channel flow, the half width h and the orientation of the wall dir_faces_paroi are needed.
formulation_a_nb_points 4 dir1 dir2: The structure fonction is calculated on four points and we should add the 2 directions (0:OX, 1:OY, 2:OZ) constituting the homegeneity planes. Example for channel flows, planes parallel to the walls.

Example:
```
Read  pb
{
      Navier_Stokes_Turbulent {
            solveur_pression GCP { ... }
            convection { Centre }
            diffusion { }
            Initial_Conditions { ... }
            boundary_conditions { ... }
            Sources { ... }
            Modele_turbulence sous_maille_selectif_mod {
                  Turbulence_paroi negligeable
                  THI 2 4
            }
            Traitement_particulier { ... }
      }
}
```

Sous_maille_wale

The WALE model is a new sub-grid scale model for eddy-viscosity in LES that has the following properties :
 it goes naturally to 0 at the wall (it doesn't need any information on the wall position or geometry)
- it has the proper wall scaling in o(y3) in the vicinity of the wall
- it reproduces correctly the laminar to turbulent transition.

The unique parameter of this sgs model is the value of the constant, Cw, whose default value 0.5.

Example:

```
Modele_turbulence sous_maille_wale
{
        turbulence_paroi negligeable
        cw 0.5
        Correction_visco_turb_pour_controle_pas_de_temps
}
```

Availability as a function of discretization is as follows:

Model	*VDF*	*VEF*
Sous_maille	YES	YES
Sous_maille_selectif	YES	YES
Sous_maille_axi	YES	NO
Sous_maille_DSGS	YES	NO
Sous_maille_Smago	YES	YES
Sous_maille_Smago_filtre	YES	YES
Sous_maille_selectif_mod	YES	NO
Sous_maille_1elt_selectif_mod	NO	YES
Sous_maille_wale	YES	YES

2.16.1.2THE MIXING LENGTH MODEL

```
Modele_turbulence  Longueur_Melange
{
    Turbulence_paroi  law
    [ Fichier ] domainname_Wall_length.xyz
    [ dmax value ]
    [ Canalx height ] [Tuyaux|Tuyauy|Tuyauz diameter ]
    [ Correction_visco_turb_pour_controle_pas_de_temps ]
    [ Fichier_ecriture_K_Eps filename.med ]
}
```

Longueur_Melange : (following keywords are available in VEF only). This model is based on mixing length modelling. For a non academic configuration (see below), formulation used in the code can be expressed basically as :

nu_t=(Kappa.y)^2.dU/dy

Till a maximum distance (**dmax**) set by the user in the data file, "y" is set equal to the distance from the wall (dist_w) calculated previously and saved a file *domainname_Wall_length.xyz*". [see Distance_paroi keyword]

Then (from y=dmax), "y" decreases as an exponential function :
y=dmax*exp[-2.*(dist_w-dmax)/dmax]

Example:
Modele_turbulence Longueur_Melange

```
{
                turbulence_paroi loi_standard_hydr dt_impr_ustar 0.00001
                dmax 0.3  fichier dom_Wall_length.xyz
 }
```

In some cases (academic configurations like pipe, channel, or, experimental ones), it is recommended to use the following data :

Canalx [height] : plane channel according to Ox direction (for the moment, formulation in the code relies on fixed heigh : H=2)

Tuyaux|Tuyauz|Tuyauz [diameter] : pipe according to Ox,Oy or Oz direction (for the moment, formulation in the code relies on fixed diameter : D=2)

Correction_visco_turb_pour_controle_pas_de_temps : Keyword to set a limitation to low time steps due to high values of turbulent vicosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the **corr_visco_turb** field which is the correction of turbulent viscosity: it should be 1. on the whole domain.

Fichier_ecriture_K_eps : When a restart with k-epsilon model is envisaged, this keyword allows to generate external MED-format file with evaluation of k and epsilon quantities (based on eddy turbulent viscosity and turbulent characteristic length returned by mixing length model). The frequency of the MED file print is set equal to **dt_impr_ustar**. Moreover, k-eps MED field is automatically saved at the last time step. MED file is then used for the restarting K-Epsilon calculation with the **Champ_Fonc_Med** keyword as explained in the 2.16.1.3 section.

Distance_Paroi domain_name nb_boundaries boundary1 boundary2 … format

Distance_paroi : This keyword generates external file "domainname_Wall_length.xyz" devoted for instance, for mixing length modelling [see Longueur_Melange]. In this file, are saved the coordinates of each element (center of gravity) of domain_name domain and minimum distance between this point and boundaries (specified boundary1,…) that user specifies in data file (typically, those which are associated to walls). Value for format may be binaire (a binary file domainname_Wall_length.xyz is written) or formatte (moreover, a formatted file domainname_Wall_length_formatted.xyz is written).

Example :

{

```
        dimension 3
        Pb_Hydraulique_Turbulent pb
        Domaine dom
        Read _file file.geo ;
        Tetraedriser_homogene_compact dom
        Distance_paroi dom 3 paroi1 paroi2 paroi3 binaire
        Fin
}
```

Where value 3 and names paroi1, paroi2, paroi3 designate respectively the number and the name of the boundaries from which minimum distance is calculated. A field Distance_paroi is available to post process the distance to the wall:

```
Post_processing {
                Fields dt_post 50. { Distance_paroi elem }
        }
```

2.16.1.3THE K-EPSILON MODEL

```
Modele_turbulence K_epsilon
{
   [ Cmu val ]
   Transport_K_Epsilon
   {
       Diffusion { [dif] }
       Convection { [schema] }
       [ Sources   {
                   Source_Transport_K_Eps { C1_eps  val C2_eps  val }
                  | Source_Transport_K_Eps_anisotherme { C1_eps val C2_eps val C3_eps val }
                  | Source_Transport_K_Eps_aniso_concen { C1_eps val C2_eps val C3_eps val }
                  | Source_Transport_K_Eps_aniso_therm_concen { C1_eps val C2_eps val C3_eps val }
       } ]
       boundary_conditions { [cl_turb1] [cl_turb2] ..... }
       [ Initial_Conditions { [cl_init] } ]
       [ parametre_equation keyword ]
   }
  [ Prandtl_K val ] [ Prandtl_Eps val ]
  [ Correction_visco_turb_pour_controle_pas_de_temps ]
   Turbulence_paroi …
}
```

Cmu :Keyword to modify the Cmu constant of k-eps model : Nut=Cmu*k*k/eps
Default value is 0.09

K_Epsilon: This keyword is selected to indicate that the turbulence model (k-ε) should be used.

Transport_K_Epsilon: This keyword is used to define the (k-ε) transportation equation.

Diffusion: This keyword is used to set the diffusion operator.

dif: This should be set to **Negligeable** to suppress the k and ε transportation equation's diffusion operator.

Convection: This keyword is used to alter the convection scheme (by default, the UPWIND scheme is selected).

schema: This may be set to **Amont** or **Quick**. Enter the first keyword to select an UPWIND type scheme, the second keyword to select a QUICK-FRAM type scheme.

Prandtl_K : Keyword to change the Pr_k value (default 1.0).

Prandtl_Eps: Keyword to change the Pr_ε value (default 1.3).

Source_Transport_K_Eps: This keyword is used to alter the source term constants in the standard k-eps model epsilon transportation equation. By default, these constants are set to:
C1_eps=1.44
C2_eps=1.92

Source_Transport_K_Eps_anisotherme | **Source_Transport_K_Eps_aniso_concen** | **Source_Transport_K_Eps_aniso_therm_concen** : This keywords are used to modify the source term constants in the anisotherm | aniso-concentration | anisotherm and aniso-concentration k-eps model epsilon transportation equation. By default, these constants are set to:
C1_eps=1.44
C2_eps=1.92
C3_eps=1.0

Boundary_conditions: This keyword is used to set the turbulence boundary conditions. Refer to 2.13.4.
cl_turb: Used to set a turbulence boundary conditions.

Initial_Conditions: These keywords are used to set initial turbulence conditions. Refer to 2.12.3.
cl_init: Defines an initial turbulence condition on a boundary. To restart from a previous mixing length calculation, an external MED-format file containing reconstructed K and Epsilon quantities can be read (see 2.16.1.2 section) thanks to the **Champ_fonc_MED** keyword (see more details for this keyword in the 2.4.1 section). Example:

Initial_Conditions { **K_Eps Champ_Fonc_MED** [**last_time**] *filename.med* domain_name **K_Eps_from_nut elem** time }

Where time is the save time of the MED fields K and Epsilon. For a practical use, last physical time can be simply loaded threw **last_time** keyword (the specified time is then unused).

Turbulence_paroi: This keyword is used to select the wall turbulence model. Refer to 2.16.3.

Correction_visco_turb_pour_controle_pas_de_temps : Keyword to set a limitation to low time steps due to high values of turbulent vicosity. The limit for turbulent viscosity is calculated so that diffusive time-step is equal or higher than convective time-step. For a stationary flow, the correction for turbulent viscosity should apply only during the first time steps and not when permanent state is reached. To check that, we could post process the **corr_visco_turb** field which is the correction of turbulent viscosity: it should be 1. on the whole domain.

Parametre_equation : See 2.6.1

Warning: When used with the Quasi-compressible model, k and ε should be viewed as ρk and ρε when defining initial and boundary conditions or when visualizing values for k and ε. This bug will be fixed in a future version.

2.16.1.4THE K_EPSILON_V2 MODEL

```
Modele_turbulence K_epsilon_V2
{
   Transport_K_Epsilon_V2 { … }
   Transport_V2 { … }
   EqnF22 { Solveur solver_kind }
   …
}
```

K_Epsilon_V2 Keyword to refer to a turbulence model available in VDF discretization

This model is a variant of the K-Epsilon turbulence model called K-Eps-V2. A transport equation for V2 is added to calculate turbulent viscosity (Nut=CmuV2).

Other new keywords:
Transport_K_Epsilon_V2 : Transport equation for K-Eps
Transport_V2 : Transport equation for V2.
EqnF22 : Elliptic equation to calculate the V2 tranport source term (solver like GMRES is needed)
V2 : New unknown field.

Example:

```
modele_turbulence K_Epsilon_V2 {
        Transport_K_Epsilon_V2
        {
                convection { amont }
                diffusion { }
                boundary_conditions {
                        bas paroi
                        haut paroi
                        obst paroi
                        entree frontiere_ouverte_K_eps_impose Champ_Front_Uniforme 2 1.e-2 1.e-3
                        sortie frontiere_ouverte K_EPS_EXT Champ_Front_Uniforme 2 0. 0.
                }
                Initial_Conditions { k_eps Champ_Uniforme 2 1.e-3 1.e-3 }
        }
        Transport_V2
         {
                convection { amont }
                diffusion { }
                boundary_conditions {
                        bas   paroi
                        haut   paroi
                        obst  paroi
                        entree frontiere_ouverte_K_eps_impose Champ_Front_Uniforme 1 1.e-3
                        sortie frontiere_ouverte K_EPS_EXT Champ_Front_Uniforme 1 0.
                }
                Initial_Conditions { V2 Champ_Uniforme 1 1.e-6 }
        }
        EqnF22 { Solveur Gmres { } }
}
```
Warning:
This model, only available in VDF discretization, is not tested.

2.16.1.5TURBULENCE MODEL K_EPSILON AT TWO LAYERS

```
Modele_turbulence K_epsilon_2_Couches
{
   Transport_K_KEpsilon {
        …
        Nb_couches integer
        [impr]
        [Y*_switch integer]
        [Nut_Switch integer]
        [Conv_forcee]
        [Conv_nat]
   }
   Turbulence_paroi  law_of_the_wall
   …
}
```

This turbulence model at two layers for the hydraulic equation is a variant of the K-Epsilon turbulence model.

Transport_K_KEpsilon : Transport equation for K and Epsilon.
Nb_couches : Maximal number of meshes for the first layer.
Impr : Optional keyword for output of the mesh numer between the two layers.
Y*_switch : Optional keyword to modify the default value (160) of the y* switch between the two layers.
Nut_Switch : Optional keyword to modify the default value (30) of the turbulent viscosity between the two layers.
Conv_forcee : Optional keyword to apply the forced convection laws inside the first layer.
Conv_nat : Optional keyword to apply the natural convection laws inside the first layer (default).

Two keywords for the standard law of the wall:
loi_paroi_2_couches for a hydraulic problem.
loi_paroi_2_couches_scalaire for a thermohydraulic problem

The boundary conditions are the same than the K-Epsilon model.

Example: (See also the test case Cavite_2couches)

```
Navier_Stokes_turbulent
{
      solveur_pression GCP { ... }
      convection { ... }
      diffusion { }
```

```
        sources { boussinesq_temperature { T0 20. } }
        Initial_Conditions { ... }
        boundary_conditions { ... }
        modele_turbulence K_Epsilon_2_couches {
                Transport_K_KEpsilon
                {
                        Nb_couches 10
                        Impr
                        convection { amont }
                        diffusion { }
                        boundary_conditions {
                                plaq_bas   paroi
                                plaq_haut   paroi
                                plaq_gauche  paroi
                                plaq_droit  paroi
                        }
                        Initial_Conditions { k_eps Champ_Uniforme 2 1.e-3 1.e-3 }
                }
                Turbulence_Paroi loi_paroi_2_couches
                dt_impr_ustar 10
        }
}
```

Warning:
Model only available in VDF discretization.

2.16.1.6LOW REYNOLDS MODEL ⟹ DISABLED MODEL SINCE V1.7.2

```
Modele_turbulence K_Epsilon_Bas_Reynolds
{
  Transport_K_Epsilon_Bas_Reynolds {
        Diffusion { [dif] }
        Convection { [schema] }
         [ Sources { Source_Transport_K_Eps_Bas_Reynolds { C1_eps val C2_eps val } } ]
        Boundary_conditions { [cl_turb1] [cl_turb2] ..... }
        [ Initial_Conditions  { [cl_init] } ]
   }
   Modele_fonc_Bas_Reynolds modele { }
}
```

K_Epsilon_Bas_Reynolds: This keyword is selected to indicate that the bas Reynolds k-ε turbulence model should be used. Caution: this model is only available in the VDF module.

Transport_K_Epsilon_Bas_Reynolds: This keyword is used to define the bas Reynolds k-ε transportation equation.

Diffusion: This keyword is used to specify the diffusion operator.

Convection: This keyword is used to change the convection scheme.

Source_Transport_K_Eps_Bas_Reynolds C1_eps C2_eps: Keywords used to modify the source term constants in the model's epsilon transportation equation. By default, these constants are set to:
C1_eps=1.55
C2_eps=2.

Boundary_conditions: This keyword is used to define turbulence boundary conditions. Refer to 2.13.4.

cl_turb: Sets a turbulence boundary condition.

Initial_Conditions: Keyword used to define initial turbulence conditions. Refer to 2.12.3.

cl_init: Sets an initial turbulence condition at a boundary.

Modele_fonc_Bas_Reynolds:*model* : This keyword is used to set the bas Reynolds model used. Currently, two models are available for VDF and VEF discretizations.

model : **Launder_Sharma** for Launder-Sharma model or **Jones_Launder** for Jones-Launder model.
When Launder Sharma's model is used, one must specify the correct constants C1 and C2 for K_eps transport equation source termes (C1 = 1.44 and C2 = 1.92) :

sources { **source_transport_K_Eps_bas_Reynolds** { **C1_eps** 1.44 **C2_eps** 1.92 } }

2.16.1.7LOW REYNOLDS FOR FLOW WITH NATURAL CONVECTION ⟹ DISABLED MODEL SINCE V1.7.2

This turbulence model for the temperature equation may be used at
low Reynolds for flow with natural convection. The other keywords are:
Transport_Fluctuation_Temperature_W_Bas_Re : Transport equation for the temperature fluctuation.
Modele_Fonc_Bas_Reynolds_Thermique : Choice of the coefficient (Jones Lauder).

As the model for hydraulic equation, boundary conditions for the transport equation of the

fluctuations are :
Frontiere_ouverte_Fluctu_Temperature_imposee : inlet boundary condition
Fluctu_Temperature_ext : outlet boundary condition

Example: (See also the test case Nagano_WBasRe)

```
Convection_Diffusion_Temperature_Turbulent
{
    diffusion { }
    convection { ... }
    boundary_conditions { ... }
    Initial_Conditions { Temperature Champ_Uniforme 1 16. }
    modele_turbulence Fluctuation_Temperature_W_Bas_Re {
        Transport_Fluctuation_Temperature_W_Bas_Re
        {
         diffusion { }
         convection { amont }
         boundary_conditions {
         plaque Paroi_fixe
         loin Frontiere_ouverte_Fluctu_Temperature_imposee Champ_Front_Uniforme 2 0.1 0.1
         planche Frontiere_ouverte Fluctu_Temperature_ext Champ_Front_Uniforme 2 0. 0.
         plafond Frontiere_ouverte Fluctu_Temperature_ext  Champ_Front_Uniforme 2 0. 0.
         }
         Initial_Conditions {
             Fluctu_Temperature Champ_Uniforme 2 1. 1.
         }
       }
       Modele_Fonc_Bas_Reynolds_Thermique Jones_Launder { }
    }
}
```

Warning:
Model only available in VDF discretization.

2.16.1.8SPECIFIED MODEL

```
Modele_turbulence Combinaison
{
  [nb_var integer var1 var2 …]
  fonction string
  Turbulence_paroi …
}
```

This keyword specify a turbulent viscosity model where the turbulent viscosity is user-defined.

nb_var integer …**:** Optional number and names of variables which will be used in the turbulent viscosity definition (by default 0)
fonction string: Fonction for turbulent viscosity. X,Y,Z and variables defined previously can be used.
Turbulence_paroi… : This keyword is used to select the wall turbulence model. Refer to 2.16.3.

2.16.2SCALAR EQUATION MODELS

2.16.2.1THE PRANDTL (SCHMIDT) MODEL

For the scalar equations, only the model based on Reynolds analogy is available.

If **K_Epsilon** was selected in the hydraulic equation, **Prandtl** must be selected for the convection-diffusion temperature equation coupled to the hydraulic equation and **Schmidt** for the concentration equations.

The syntax to use these turbulence models is as follows:

```
 Modele_turbulence Prandtl | Schmidt
{
        Turbulence_paroi law
        [ dt_impr_nusselt  value ]
        [ Prdt | ScTurb double | Prandt_turbulent_fonction_nu_t_alpha string ]
}
```

Turbulence_paroi law : A scalar wall law should be specified. Refer to 2.16.3.2.

Prdt|ScTurb: Keywords to modify the constant of the model. Default value is 0.9 for turbulent Prandtl number ($\alpha_t=\nu_t/P_{rt}$) and 0.7 for the turbulent Schmidt number ($D_t=\nu_t /S_{ct}$).

Prandt_turbulent_fonction_nu_t_alpha : Optional keyword to specify turbulent diffusivity (by default, $\alpha_t=\nu_t/Pr_t$) with another formulae, for example: $\alpha_t=\nu_t^2/(0,7\alpha+0,85\nu_t)$ with the string **nu_t*nu_t**/(0,7***alpha**+0,85***nu_t**) where alpha (α) is the thermal diffusivity.

dt_impr_nusselt : Keyword to print local values of Nusselt number and temperature near a wall during a turbulent calculation. The values will be printed in the "_Nusselt.face" file each dt_impr_nusselt time period.

The local Nusselt expression is as follows : Nu =((lambda+lambda_t)/lambda)*d_wall/d_eq where d_wall is the distance from the first mesh to the wall and d_eq is given by the wall law. This option also gives the value of d_eq, h=(lambda+lambda_t)/d_eq and the fluid temperature of the first mesh near the wall.
For the Neumann boundary conditions (flux_impose), the "equivalent" wall temperature given by the wall law is also printed (Tparoi equiv.) preceded for VEF calculation by the edge temperature "T face de bord".

2.16.2.2DYNAMIC SUBGRID SCALE MODEL

```
Modele_turbulence Sous_maille_dyn
{
    [ dynamique_y2 integer ]
    [ stabilise
    [ 6_points ]
    [ plans_paralleles nb_points integer ]
    [ moy_euler ]
    [ moy_lagrange ] ]
    Turbulence_paroi  law…
}
```

Warning: Available in VDF only. Not coded in VEF yet.

2.16.2.3THERMAL FLUCTUATION TURBULENCE MODEL

```
 Modele_turbulence Fluctuation_Temperature
 {
 Transport_Fluctuation_Temperature {
        Diffusion { [dif] }
        Convection { [schema] }
      [ Sources { Source_Transport_Fluctuation_Temperature { Ca val Cb val Cc val Cd val } } ]
        Boundary_conditions { [cl_turb1] [cl_turb2] ..... }
      [ Initial_Conditions { [cl_init] } ]
    }
 Transport_Flux_Chaleur_Turbulente {
        Diffusion { [dif] }
        Convection { [schema] }
         [ Sources { Source_Transport_Flux_Chaleur_Turbulente { C1_teta val C2_teta val C3_teta
         val } } ]
        Boundary_conditions { [cl_turb1] [cl_turb2] ..... }
      [ Initial_Conditions  { [cl_init] } ]
    }
```

Fluctuation_Temperature: This is a keyword used to select a model for thermal fluctuations should a turbulent thermohydraulic problem occur. This model resolves two new equations (keywords **Transport_Fluctuation_Temperature** and **Transport_Flux_Chaleur_Turbulente**) and uses specific boundary conditions. The first equation deals with thermal fluctuation (T'^2) variance transportation and the thermal fluctuation dissipation rate (new field Fluctu_Temperature of the T'^2,Eps_T' components), the second deals with transportation of 3 turbulent heat flux components (new field Flux_Chaleur_Turbulente belonging to the uT',vT',wT' components).

Diffusion: This keyword is used to specify an equation diffusion operator.

Convection: This keyword is used to alter the equation convection scheme.

Source_Transport_Fluctuation_Temperature Ca Cb Cc Cd: These keywords are used to modify the source term constants in the temperature fluctuation transportation equation in the thermal fluctuation model. By default, these constants are set to:
Ca=0.8
Cb=2.0
Cc=1.96
Cd=0.8

Source_Transport_Flux_Chaleur_Turbulente C1_teta C2_teta C3_teta: These keywords are used to alter the source term constants in the turbulent heat flux transportation equation in the thermal fluctuation model. By default, these constants are set to:
C1_teta=5.
C2_teta=0.5
C3_teta=0.33

Boundary_conditions: These keywords are used to set the turbulence boundary conditions. Refer to 2.13.4.

cl_turb: Sets a turbulence boundary condition.

Initial_Conditions: This keyword is used to define the initial turbulence conditions. Refer to 2.12.3.

cl_init: Sets an initial turbulence condition at the boundary.

This model features the following keyword that may be used to post-process the fields (refer to 2.12.3):

Variance_Temperature: This keyword is used to post-process the temperature fluctuation variation (T'^2) during a k-eps calculation with a turbulence model for thermal fluctuations.

Taux_Dissipation_Temperature: This keyword is used to post-process the temperature fluctuation dissipation rate during a k-eps calculation with a turbulence model for thermal fluctuations.

Flux_Chaleur_Turbulente: This keyword is used to post-process turbulence heat flux components (uT', vT', wT') during a k-eps calculation with a turbulence model for thermal fluctuations.

2.16.3WALL LAWS

```
Turbulence_paroi loi
  [dt_impr_ustar periode ]
  [dt_impr_ustar_mean_only {
          dt_impr periode
          [boundaries nb_boundaries  boundary_name1 boundary_name2 ... ]
  }]
  [nut_max value]
  [eps_min value]
  [k_min value]
```

Turbulence_paroi: This keyword is used to set the wall law model.

dt_impr_ustar: This keyword is used to print the values (U +, d+, u*) obtained with the wall laws into a file named *datafile_ProblemName_Ustar.face* and *periode* refers to the printing period, this value is expressed in seconds.

dt_impr_ustar_mean_only: This keyword is used to print the mean values of u* (obtained with the wall laws) on each boundary, into a file named *datafile_ProblemName_Ustar_mean_only.out. periode* refers to the printing period, this value is expressed in seconds. If you don't use the optional keyword **boundaries**, all the boundaries will be considered. If you use it, you must specify *nb_boundaries* which is the number of boundaries on which you want to calculate the mean values of u*, then you have to specify their names.

Keywords to set a limitation to low or high turbulent values for K-Eps models :
nut_max : upper limitation of turbulent viscosity (default value 1.e8).
eps_min : lower limitation of epsilon (default value 1.e-10).
k_min: lower limiation of k (default value 1.e-10).

loi: The law selected for wall turbulence. It depends of the equation :

2.16.3.1MOMEMTUM EQUATIONS

- **Loi_standard_hydr** (or **Loi_standard_hydr_3couches**) : Keyword for the logarithmic wall law. **Loi_standard_hydr** refers to first cell rank eddy-viscosity defined from continuous analytical functions, whereas **Loi_standard_hydr_3couches** from functions separataly defined for each sub-layer

- **Loi_expert_hydr** { … } : This keyword is similar to the previous keyword **Loi_standard_hydr** but has several additional options into brackets :

  Kappa value: The value of κ can be changed from the default one (0.415)
  Erugu value: The value of E can be changed from the default one for a smooth wall (9.11). It is also possible to change the value for one boundary wall only with **paroi_rugueuse** keyword.
  A_plus value: A+ value can can be changed from the default one (26.0)

  More options for **loi_expert_hydr** keyword are available for VEF discretization:
  u_star_impose value : The value of the friction velocity (u*) is not calculated but given by the user.
  methode_calcul_face_keps_impose option : The available options select the algorithm to apply K and Eps boundaries condition (the algorithms differ according to the faces).
    toutes_les_faces_accrochees : Default option in 2D (the algorithm is the same than the algorithm used in **Loi_standard_hydr**)
    que_les_faces_des_elts_dirichlet : Default option in 3D (another algorithm where less faces are concerned when applying K-Eps boundary condition)

- **Paroi_TBLE { N** value [**kappa** value] [**facteur** value] [**modele_visco** filename] [**stats** value value] **}**

Keyword for the Thin Boundary Layer Equation wall-model (a more complete description of the model can be found into this PDF file). The wall shear stress is evaluated thanks to boundary layer equations applied in a one-dimensional fine grid in the near-wall region. The options are:

N value: Number of nodes in the TBLE grid (mandatory option).

kappa value **:** Optional option to change the default 0.415 value for kappa?

facteur value: Stretching ratio for the TBLE grid (to refine, the TBLE factor must be greater than 1)

modele_visco filename: File name containing the description of the eddy viscosity model

stats values: Statistics of the TBLE velocity and turbulent viscosity profiles. 2 values are required : the starting time and ending time of the statistics computation.

- **Utau_imp**: Keyword to impose the friction velocity on the wall with a turbulence model for thermohydraulic problems. There are two possibilities to use this keyword :
  1. we can impose directly the value of the friction velocity u_star.

     Example :
     ```
     modele_turbulence sous_longueur_melange
     {
             Cs 0.01
             turbulence_paroi UTAU_IMP { u_tau Champ_uniforme 1 0.1 }
     }
     ```
  2. we can also give the friction coefficient **lambda_c** and hydraulic diameter **diam_hydr**. **Lambda_c** can be function of the spatial coordinates x,y,z, the Reynolds number **Re**, and the diameter hydraulic **Dh**. So, TRUST determines the friction velocity by :

     u_star = U*sqrt(lambda_c/8)

     Example:
     ```
     modele_turbulence longueur_melange
     {
             turbulence_paroi UTAU_IMP
             {
                     diam_hydr Champ_uniforme 1 2
                     lambda_c 0.02
             }
     }
     ```

- **Negligeable** : This keyword is used to suppress the calculation of a law of the wall with a turbulence model. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall (tau_tan /rho= nu dU/dy).
  Warning: This keyword is not available for k-epsilon models. In that case you must choose a wall law.

Other available laws:

- **Loi_Ciofalo_hydr**
- **Loi_WW_hydr**

Warning:
Only **Loi_WW_hydr** laws have been qualified on channel calculation.
These keywords are only available for a LES calculation.

2.16.3.2SCALAR EQUATIONS

- **Loi_standard_hydr_scalaire :** Keyword for the law of the wall.

- **Loi_expert_scalaire** { … } **:** Keyword similar to keyword **Loi_standard_hydr_scalaire** but with additional option into brackets :
  calcul_ldp_en_flux_impose value : By default (value set to 0), the law of the wall is not applied for a wall with a Neumann condition. With value set to 1, the law is applied even on a wall with Neumann condition.
  Prdt_sur_kappa value : This option is to change the default value of 2.12 in the scalable wall function.

- **Loi_Paroi_Nu_Impose :** Keyword, it is possible to impose Nusselt numbers on the wall for the thermohydraulic problems. To use this option, it is necessary to give in the data file the value of the hydraulic diameter and the expression of the Nusselt number. This expression can be a function of x, y, z, Re (Reynolds number), Pr (Prandtl number)
  Example :
  ```
  Turbulence_paroi Loi_Paroi_Nu_Impose
  {
          nusselt 0.023*Re^0.8*Pr^(1./3.)
          diam_hydr champ_uniforme 1 9e-3
  }
  ```
  In this example, the Nusselt expression is the Colburn correlation.

- **Loi_ODVM** { **N** value **Gamma** value **Stats** value_t0 value_dt **Check_files** }
  Thermal wall-function based on the simultaneous 1D resolution of a turbulent thermal boundary-layer and a variance transport equation, adapted to conjugate heat-transfer problems with fluid/solid thermal interaction (where a specific boundary condition should be used : **Paroi_Echange_Contact_OVDM_VDF**). This law is also available with isothermal walls.
  N value: number of points per face in the 1D uniform meshes. N should be choosen in order to have the first point situated near $\Delta y^{+}=1/3$.
  Gamma value: Smoothing parameter of the signal between 10e-5 (no smoothing) and 10e-1 (high averaging).

Stats value_t0 value_dt: Only for plane channel flow, it gives mean and root mean square profiles in the fine meshes, since value_t0 and every value_dt seconds. The values are printed into files named *ODVM_fields*.dat*.
Check_files: It gives for one boundary face a historical view of local instantaneous and filtered values, as well as the calculated variance profiles from the resolution of the equation. The printed values are into the file *Suivi_ndeb.dat*.

- **Paroi_TBLE_scal { N** value [**Prandtl** value] [**facteur** value] [**modele_visco** filename] [**Nb_comp** value] [**stats** value value] **}**
  Keyword for the Thin Boundary Layer Equation thermal wall-model.
  Prandtl value : Option to change the default value (1.0) of turbulent Prandtl number.
  See **Paroi_TBLE** for the other options.

- **Negligeable_scalaire :** Keyword to suppress the calculation of a law of the wall with a turbulence model for thermohydraulic problems. The wall stress is directly calculated with the derivative of the velocity, in the direction perpendicular to the wall.

2.17 SAVING A PROBLEM

```
Sauvegarde format_sauvegarde  nom_fichier
```

```
Sauvegarde_simple format_sauvegarde  nom_fichier
```

Sauvegarde: Keyword used when calculation results are to be backed up.
Sauvegarde_simple: Same keyword than **Sauvegarde** except, the last time step only is saved.

format_sauvegarde: thress keywords may be used: **binaire** (binary format) or **formatte** (ASCII format) or **xyz** (multi-processor/multi-physics format).

The results are saved to the *nom_fichier* file according to a frequency set by **dt_sauv** (refer to time schemes 2.9). The file contains all the information saved over time.

If this instruction is not entered, results are saved only upon calculation completion in the file *nom_du_cas.sauv*.
When a coupling is performed, the backup-recovery file name must be well specified for each problem. In this case, you must save to different files and correctly specify these files when restarting the calculation.

2.18 RESTARTING A PROBLEM

Reprise|Resume_last_time format_reprise nom_fichier

Reprise: This keyword is used to restart a calculation at the **tinit** time with the fields stored into the *nom_fichier* file.

Resume_last_time does the same thing, but will restart the calculation at the last time found in the file ('tinit' is set to last time of saved files).

format_reprise: there are three keywords available: **binaire (**binary format**)**, **formatte (**formatted format**),** or **xyz.**. The calculation is restarted based on the *nom_fichier* file. If **xyz** is entered, the *nom_fichier* file should be the *.xyz* file created by the previous calculation. With this file, it is possible to restart a parallel calculation on P cpus, whereas the previous calculation has been run on N (N<>P) cpus. By default, a .xyz file is created at the end of the calculation. To save space disc, you can prevent TRUST from writing this **.xyz** file, thanks to a line "**EcritureLectureSpecial** value" (with 0 as value) located in the data file just before the **Solve** keyword.

2.19PROBLEM POST-PROCESSING

Several keywords can be used to create a postprocessing block, into a problem. First, you can create a single postprocessing task (**Post_processing** keyword). Generally, in this block, results will be printed with a specified format at a specified time period.

```
Post_processing {
    Postraitement_definition
}
```

But you can also create a list of postprocessing with **Post_processings** keyword (named with *Post_name1*, *Post_name2*, etc…), in order to print results to several formats or with different time periods, or into different results files:

```
Post_processings {
    Post_name1  { Postraitement_definition }
    Post_name2  { Postraitement_definition }
    …
}
```

The postraitement_definition has the following syntax :

[**Probes** {
    [nom_sonde [type] field_name **Periode** dts **Points**
        position_like nom_sonde | n x1 y1 [z1] x2 y2 [z2] .... xn yn [zn]]
    [nom_sonde [type] field_name **Periode** dts **Segment**
        position_like nom_sonde | ns x1 y1 [z1] x2 y2 [z2]]
    [nom_sonde [type] field_name **Periode** dts **Segmentpoints**
        position_like nom_sonde | ns x1 y1 [z1] x2 y2 [z2] .... xn yn [zn]]
    [nom_sonde [type] field_name **Periode** dts **Plan**
        position_like nom_sonde | ns1 ns2 x1 y1 [z1] x2 y2 [z2] x3 y3 [z3]]
    [nom_sonde [type] field_name **Periode** dts **Volume**
        position_like nom_sonde | ns1 ns2 ns3 x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4]
    [nom_sonde [type] field_name **Periode** dts **Circle**
        position_like nom_sonde | n x0 y0 [z0 dir] r teta1 teta2]
    [nom_sonde [type] field_name **Periode** dts **Numero_elem_sur_maitre** integer
}]

[**Definition_champs** {
    [field_name_post **refChamp** { … }]
    [field_name_post **Interpolation** { … }]
    [field_name_post **Gradient** { … }]
    [field_name_post **Divergence** { … }]
    [field_name_post **Moyenne** { … }]
    [field_name_post **Ecart_Type** { … }]
    [field_name_post **Correlation** { … }]
    [field_name_post **Transformation** { … }]
    [field_name_post **Extraction** { … }]
    [field_name_post **Reduction_0D** { … }]
    [field_name_post **Morceau_Equation** { … }]
    [field_name_post **Predefini** { … }]
    [field_name_post **Tparoi_VEF** { … }]
}]

[**Fichier** filename] [**Format lml**|**lata**|**lata_v1**|**lata_v2**|**med**] [**Domaine** domaine_name]

[**Fields** [**formatte**|**binaire**] **dt_post** string | **nb_pas_dt_post** integer {
    [field_name] [localisation]
    …
}]

[**Statistiques Dt_post** dtst {
    t_deb value **t_fin** value
    [*stat* field_name [second_field_name]] [localisation]
    …
}]

[**Statistiques_en_serie Dt_integr** dtst {
    t_deb value **t_fin** value
    [*stat* field_name] [localisation]
    …
}]

Where :
Probes is a keyword to define probes postprocessing (1D plots). See 2.19.2
Definitions_champs is a keyword to create new fields for postprocessing. See 2.19.3
Format, Fichier, Domaine, Fields, Statistiques, Statistiques_en_serie are keywords related to field 2D/3D postprocessing. See 2.19.4 and 2.19.5

field_name is the name of the field being postprocessed and the next paragraph gives details about the different fields.

2.19.1POST-PROCESSING FIELD NAMES

The fields which may currently be post processed are:

Physical values	*Keyword for field_name*	*Unit*
Speed	**Vitesse**	$m.s^{-1}$
Kinetic energy	**Energie_cinetique**	$m^2.s^{-2}$
Vorticity	**Vorticite**	s^{-1}
Pressure in incompressible flow (=P/ρ+gz). For Front Tracking probleme (=P+ρgz)	**Pression** $^{(***)}$	$Pa.m^3.kg^{-1}$ or Pa
Pressure in incompressible flow (=P+ρgz)	**Pression_pa**	Pa
Pressure in compressible flow	**Pression**	Pa
Totale pressure (when quasi compressible model is used)=Pth+P	**Pression_tot**	Pa
Pressure gradient (=grad(P/ρ+gz))	**Gradient_pression**	$m.s^{-2}$
Temperature	**Temperature**	°C or K
Phase temperature of a two phases flow	**Temperature_EquationName**	°C or K
Mass transfer rate between two phases	**Temperature_mpoint**	$kg.m^{-2}.s^{-1}$
Temperature variance	**Variance_Temperature**	K^2
Temperature dissipation rate	**Taux_Dissipation_Temperature**	$K^2.s^{-1}$
Temperature gradient	**Gradient_temperature**	$K.m^{-1}$
Heat exchange coefficient	**H_echange_Tref** $^{(**)}$	$W.m^{-2}.K^{-1}$
Turbulent heat flux	**Flux_Chaleur_Turbulente**	$m.K.s^{-1}$
Turbulent viscosity	**Viscosite_turbulente**	$m^2.s^{-1}$

Turbulent dynamic viscosity (when quasi compressible model is used)	**Viscosite_dynamique_turbulente**	kg.m.s^{-1}
Turbulent kinetic energy	**K**	m^2.s^{-2}
Turbulent dissipation rate	**Eps**	m^3.s^{-1}
Constituent concentration	**Concentration**	
Component velocity along X	**VitesseX**	m.s^{-1}
Component velocity along Y	**VitesseY**	m.s^{-1}
Component velocity along Z	**VitesseZ**	m.s^{-1}
Mass balance on each cell	**Divergence_U**	m^3.s^{-1}
Irradiancy	**Irradiance**	W.m^{-2}
Q-criteria	**Critere_Q**	s^{-1}
Distance to the wall Y+=yU*/ν (only computed on boundaries of wall type)	**Y_plus**	dimensionless
Friction velocity	**U_star**	m.s^{-1}
Cell volumes	**Volume_maille**	M^3
Chemical potential	**Potentiel_Chimique_Generalise**	
Source term in non Galinean referential	**Acceleration_terme_source**	m.s^{-2}
Stability time steps	**Pas_de_temps**	S
Boundary fluxes	**Flux_bords**	
Volumetric porosity	**Porosite_volumique**	dimensionless
Distance to the wall	**Distance_Paroi** [(*)]	M
Volumic thermal power	**Puissance_volumique**	W.m^{-3}
Local shear strain rate defined as $\sqrt{(2S_{ij}S_{ij})}$	**Taux_cisaillement**	s^{-1}
Cell Courant number (VDF only)	**Courant_maille**	dimensionless
Cell Reynolds number (VDF only)	**Reynolds_maille**	dimensionless

[(*)]: **distance_paroi** is a field which can be used only if the mixing length model (see 2.16.1.2) is used in the data file.

[(**)]: **Tref** indicates the value of a reference temperature and must be specified by the user. For example, **H_echange_293** is the keyword to use for Tref=293K.

[(***)] : The post-processed pressure is the pressure divided by the fluid's density (P/rho+gz) on incompressible laminar calculation. For turbulent, pressure is P/rho+gz+2/3*k cause the turbulent kinetic energy is in the pressure gradient.

Note 0: Since the 1.4.8 version, statistical fields can be plotted with probes with the keyword "**operator_field_name**" like for example, **Moyenne_Vitesse** or **Ecart_Type_Pression** or **Correlation_Vitesse_Vitesse**. For that, it is mandatory to have the statistical calculation of this fields defined with the keyword **Statistiques**.

Note 1: Since the 1.5.3 version, physical properties (conductivity, diffusivity,…) can also been interrogated. The name of the fields and components available for post-processing is displayed in the error file after the following message: "Lecture des champs a postraiter". Of course, this list depends of the problem being solved.

For example, the Poiseuille_VDF test case provides the following fields or components:

...
Lecture des champs a postraiter
Milieu_base : 1 masse_volumique
Fluide_Incompressible : 2 viscosite_cinematique viscosite_dynamique
Equation_base : 1 volume_maille
Operateur_base : 0
Operateur_base : 0
Navier_Stokes_std : 13 divergence_U gradient_pressionY gradient_pressionX gradient_pression pression_pa pression vitesseY vitesseX vitesse y_plus porosite_volumique critere_Q vorticite
...

2.19.2POST-PROCESSING BY PROBE

Probes refer to sensors that allow a value or several points of the domain to be monitored over time. The probes may be a set of points defined one by one (keyword **Points**) or a set of points evenly distributed over a straight segment (keyword **Segment**) or arranged according to a layout (keyword **Plan**) or according to a parallelepiped (keyword **Volume**)

The fields allow all the values of a physical value on the domain to be known at several moments in time.

Probes: This keyword is used to define the probes.

nom_sonde: This is the name of the file suffix in which the values taken over time will be saved. The complete file name is *nom_sonde.son.*

type: Option to change the positions of the probes. Several options are available:
grav : each probe is moved to the nearest cell center of the mesh
som : each probe is moved to the nearest vertex of the mesh
nodes : each probe is moved to the nearest face center of the mesh
chsom : Only available for P1NC sampled field. The values of the probes are calculated according to P1-Conform corresponding field.

field_name: name of the sampled field.

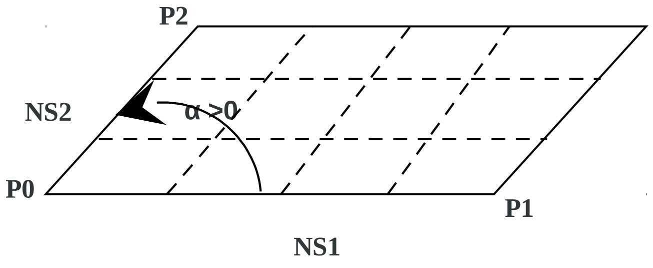
Periode: This keyword is used to set the sampled field measurement frequency. Every *dts* seconds, the field value calculated at the previous time step is written to the *nom_sonde.son* file.
dts: period value(s).

Points: This keyword is used to define the number of probe points. The field *field_name* is sampled at *n* points in the domain.
n: number of probe points.
xi yi zi: probe measurement point co-ordinates. If the point does not coincide with a calculation node, the measurement is extrapolated linearly according to neighbouring node values.

Segment: This keyword is used to define the number of probe segment points. The *field_name* field is sampled at *ns* points of the segment, evenly distributed.
ns: number of probe fields defined on the segment.
x1 y1 z1 x2 y2 z2: co-ordinates of the 2 outer probe segment points. If the point does not coincide with a calculation node, the measurement is linearly extrapolated according to neighbouring node values.

Segmentpoints: This keyword is used to define a probe segment from specifics points. The field_name field is sampled at ns specifics points.
ns: number of specifics points.
xi yi zi: co-ordinates of the specifics points. If the point does not coincide with a calculation node, the measurement is linearly extrapolated according to neighbouring node values.

Plan: Keyword used to set the number of probe layout points.



x1 y1 z1 x2 y2 z2 x3 y3 z3: co-ordinates of the 3 points that define the angle. This angle should be positive.
The keyword **Plan** (layout) file format is type .lml, the others (Point and Segment) are arranged in columns.
Observations: the probe co-ordinates should be given in Cartesian co-ordinates (X Y Z, including axisymmetric.

Volume: This is a keyword used to define the probe volume in a parallelepiped passing through 4 points A, B, C, D, and the number of probes in each direction. For example:

```
Probes  {
        Sonde_P pression periode 0.01 volume 5 3 3   0. 0. 0.   5. 0. 0.   2. 2. 0.   0. 0. 2.
}
```

Circle: This is a keyword to define several probes located on a circle of radius r and centered at point x0,y0,z0. dir is an integer which gives the axis normal to the circle plane (0:x axis, 1:y axis, 2:z axis). The n probes are between teta1 and teta2 (angles given in degrees).

Position_like nom_sonde: Keyword to define a probe at the same position of another probe named nom_sonde.

Numero_elem_sur_maitre integer : Keyword to define a probe on the mesh element integer. Useful when using min/max probes.

To not have interpolations on your post-processed fields, use in **VDF** :

	Names	Trio_U keywords	Where is it calculated in VDF ?	Recommended keyword in VDF	
				for probes (*.son)	for fields (*.lata)
Unknowns	Pressure	***pression***	gravity center of the element	***grav***	***elem***
	Velocity	***vitesse***	center of the faces	***nodes***	***faces***
	Temperature	***temperature***	gravity center of the element	***grav***	***elem***
Physical caracteristics	Density rho	***masse_volumique***	gravity center of the element	***grav***	***elem***
	Cinematic viscosity nu	***viscosite_cinematique***	gravity center of the element	***grav***	***elem***
	Dynamic viscosity mu	***viscosite_dynamique***	gravity center of the element	***grav***	***elem***
Turbulence	k	***k***	gravity center of the element	***grav***	***elem***
	eps	***eps***	gravity center of the element	***grav***	***elem***
	y+	***y_plus***	gravity center of the element	***grav***	***elem***
	u*	***u_star***	center of the faces	***nodes***	***faces***
	Turbulent viscosity	***viscosite_turbulente***	gravity center of the element	***grav***	***elem***

To not have interpolations on your post-processed fields, use in **VEF** :

	Names	Trio_U keywords	Where is it calculated in VEF ?	Recommended keyword in VEF	
				for probes (*.son)	for fields (*.lata)
Unknowns	Pressure	***pression***	P0: gravity center of the element	***grav***	***elem***
			P1: vertexes	***som***	***som***
			Pa: center of the faces (only for 3D)	***nodes***	***faces***
	Velocity	***vitesse***	center of the faces	***nodes***	***faces***
	Temperature	***temperature***	center of the faces	***nodes***	***faces***
Physical caracteristics	Density rho	***masse_volumique***	gravity center of the element	***grav***	***elem***
	Cinematic viscosity nu	***viscosite_cinematique***	gravity center of the element	***grav***	***elem***
	Dynamic viscosity mu	***viscosite_dynamique***	gravity center of the element	***grav***	***elem***
Turbulence	k	***k***	center of the faces	***nodes***	***faces***
	eps	***eps***	center of the faces	***nodes***	***faces***
	y+	***y_plus***	gravity center of the element	***grav***	***elem***
	u*	***u_star***	center of the faces	***nodes***	***faces***
	Turbulent viscosity	***viscosite_turbulente***	gravity center of the element	***grav***	***elem***

2.19.3 ADVANCED FIELD POST-PROCESSING

```
Definition_champs {
    field_name_post  field_type { … }
   …
}
```

Definition_champs: Keyword to create new or more complex field for advanced postprocessing. *field_name_post* is the name of the new created field. *field_type* is one of the following possible type (**refChamp**, **Interpolation**, **Gradient**,…) :

```
field_name_post refChamp { Pb_champ nom_pb field_name }
```

nom_pb is the problem name and *field_name* is the selected field name

```
field_name_post Interpolation { [domaine nom_dom ]
                                localisation type_loc
                                [methode type_method]
                                source field_type { … }
                              }
```

This keyword creates a field which is an interpolation of the field given by the keyword **source.** *nom_dom* is the domain name where the interpolation is done (by default, the calculation domain) *type_loc* indicate where is done the interpolation (« elem » for element or « som » for node). The optional keyword **methode** is limited to **calculer_champ_post** for the moment.

```
field_name_post Gradient { source  field_type { … } }
field_name_post Divergence { source field_type { … } }
```

These keywords enable to calculate gradient or divergency of a given field.

```
 field_name_post Moyenne {
           t_deb val1 t_fin val2 source  field_type {…}
                     [moyenne_convergee  Champ_fonc_reprise file.xyz pb_name Moyenne_field
last_time]
 }
field_name_post Ecart_Type {
           t_deb val1 t_fin val2 source field_type {…}
}
field_name_post Correlation {
           t_deb val1 t_fin val2 sources {  field_type { … } ,  field_type { … } }
}
```

These keywords enable to create more statistic fields (see 2.19.5). The option **moyenne_convergee** allows to read a converged time averaged field in a .xyz file in order to calculate, when restarting the calculation, the statistics fields (rms, correlation) which depend on this average. In that case, the time averaged field is not updated during the restarting calculation. In this case, the time averaged field must be fully converged to avoid errors when calculating high order statistics.
Warning : a correlation between two fields that are not calculated at the same discretisation location, takes very much time ! For example, a correlation between the velocity and the temperature in VDF which are respectively calculated at the faces and at the nodes/elements, is really expensive.

```
field_name_post Transformation {
        methode  norme
                   | produit_scalaire
                   | composante numero integer
                   | formule expression 1 f (x,y,z,t,)
                   | vecteur expression N f1(x,y,z,t) … fN(x,y,z,t)
        [localisation loc]
        [source field_type { … } | sources { field_type { … } , field_type { … } , …}]
 }
```

This keyword is used to create a field with a transformation.
methode norme : will calculate the norm of a vector given by a **source** field specified by *field_type.*
methode produit_scalaire : will calculate the dot product of two vectors given by two **sources** fields

methode composante numero integer : will create a field by extracting the integer component of a field given by a **source** field
methode formule expression 1 : will create a field located to elements using one expression with x,y,z,t parameters and field names given by a **source** field or several **sources** fields. This field will be a scalar or a vector field according to the fields used in the expression.
methode vecteur expression N f1(x,y,z,t) … fN(x,y,z,t) **:** will create a scalar (N=1) or vector field (N>1) located to elements by defining its N components with N expressions with x,y,z,t parameters and field names given by a **source** field or several **sources** fields.

field_name_post **Extraction** { **domaine** *nom_dom* **nom_frontiere** *nom_fr*
[**methode** [**trace** | **champ_frontiere**]]
source *field_type* { … }
}

This keyword is used to create a surface field (values at the boundary) of a volume field
-*nom_dom* name of a surface domain which should has been created before
-*nom_fr* boundary name of the volume domaine where the values of the volume field will be picked
-*type_methode* name of the extraction method (**trace** by_default, the field on the surface will be calculated from the volume field or **champ_frontiere,** the boundary conditions of the volume field will be used)

field_name_post **Reduction_0D** { [**methode** *type_methode*]
source *field_type* { … }
}

These keyword is used to calculate the min, max, or mean value of a field.

-*type_methode* name of the reduction method (**min, max, somme** for the sum, **somme_ponderee** for a weighted sum (integral), **norme_L2** for the L2 norm, **moyenne** for a mean and **moyenne_ponderee** for a mean ponderated by integration volumes, e.g: cell volumes for temperature or pressure in VDF, volumes around faces for velocity and temperature in VEF)

```
field_name_post Morceau_Equation {
        type piece_type
        [numero 0 | 1]
        option option_type [ compo num_compo ]
        source  field_type { … }
}
```

These keyword is used to calculate a field related to a piece of equation. For the moment, *piece_type* can only be **operateur** for equation operators. **numero** will be 0 (diffusive operator), 1 (convective operator), 2 (gradient operator), 3 (divergence operator). **option** (option_type) is limited for the moment to **stability** (for time steps) or **flux_bords** (for boundary fluxes, in this case **compo** permits to specify the number component of the boundary flux choosen). The keyword **source** will be used to specify the equation. The problem name and the unknown of the equation (temperature, vitesse for example) should be given:
Source refChamp { **Pb_Champ** problem_name unknown_field_of_equation }

```
field_name_post Operateur_Eqn  {
        numero_source int
        numero_op int
        sans_solveur_masse 0 | 1
        source  field_type { … }
}
```

These keyword is used also to calculate a field related to a piece of equation, either an operator (**numero_op** option, 0 for diffusive operator, 1 for convective operator) or a source term (**numero_source** option, the integer will specify the rank of the source term in the equation sources list). The field calculated will be returned either multiplied by the reverse matrix mass (**sans_solveur_masse** set to 1) or not (**sans_solveur_masse** set to 0, the default). The keyword **source** will be used to specify the equation. The problem name and the unknown of the equation (temperature, vitesse for example) should be given:
Source refChamp { **Pb_Champ** problem_name unknown_field_of_equation }

```
field_name_post  Predefini { Pb_Champ nom_pb field_name } }
```

These keyword is used to post process predefined postprocessing fields. For the moment, only kinetic energy (**energie_cinetique** keyword to use for *field_name*) is available.

```
field_name_post Tparoi_VEF {
        Source refChamp { Pb_Champ nom_pb field_name }
}
```

These keyword is used to post process (only for VEF discretization) the temperature field with a slight difference on boundaries with Neumann condition where law of the wall is applied on the temperature field. *nom_pb* is the problem name and *field_name* is the selected field name. A keyword (**temperature_physique**) is available to post process this field without using **Definition_champs**.

Remarks:

I) In the previous examples, if the source field specified with the **source** keyword is already a new post field named *name_of_champ_post_field*, you should use **source_reference** *name_of_champ_post_field* instead of **source** *field_type* { … } or **sources_reference** { name1 name2 ... nameN } if you have N fields.

II) It is possible to create an alias for a source field with the **nom_source** keyword:
field_name field_type { **source** *field_type* { **nom_source** nom} }
By default, the name of source field is given according to the *field_type*:

refChamp:	fieldname_**natif_**domain
Interpolation:	sourcename_localization_domainInterpolation
Moyenne:	**Moyenne_**sourcename
Ecart_Type:	**Ecart_Type_**sourcename
Correlation:	**Correlation_**firstsourcename_secondsourcename
Gradient :	**Gradient_**sourcename
Divergence :	**Divergence_**sourcename
Transformation:	**Combinaison_**sourcename
Extraction:	**Extraction_**sourcename
Reduction_0D:	**Reduction_0D_**sourcename
Tparoi_VEF:	**Tparoi_VEF_**sourcename

III) The components of a field is obtained by adding the number of the component (0 for the first component, 1 for the second one,…). Example:

```
Definition_champs
{
        Pressure_gradient gradient { source refchamp { pb_champ pb pression } }
}
Fields dt_post 1.1
{
        Gradient_pression0 elem # dp/dx #
        Gradient_pression1 elem  # dp/dy #
}
```

IV) The oldier syntax for the field type remains understood. The corresponding types are :

last syntax	*old syntax*
refChamp	(->**Champ_Post_refChamp**)
Interpolation	(->**Champ_Post_Interpolation**)
Moyenne	(->**Champ_Post_Statistiques_Moyenne**)
Ecart_Type	(->**Champ_Post_Statistiques_Ecart_Type**)
Correlation	(->**Champ_Post_Statistiques_Correlation**)
Gradient	(->**Champ_Post_Operateur_Gradient**)
Divergence	(->**Champ_Post_Operateur_Divergence**)
Transformation	(->**Champ_Post_Transformation**)
Extraction	(->**Champ_Post_Extraction**)
Reduction_0D	(->**Champ_Post_Reduction_0D**)
Tparoi_VEF	(->**Champ_Post_Tparoi_VEF**)

V) It is recommended to build a complex field in a one way process. For example, to define the L2 norm error of velocity compare to an analytical solution, you will define something like:

```
# Define the L2 error #
Definition_champs {
        Error  reduction_0D
         {
                methode norme_L2 source Transformation
                {
                        methode formule expression 1 velocity-solution
                        sources {
                                refChamp { Pb_champ pb vitesse nom_source velocity } ,
                                Transformation
                                {
                                        methode vecteur
                                        expression 2 x*y x+y nom_source solution
                                }
                        }
                }
        }
}
# Write the L2 error like a probe in a file #
Probes {  file_error   Error  periode  0.0005  numero_elem_sur_maitre  0 }
```

Another example :

```
# Calculate circonferential velocity W from velocity components Ux and Uy #
Definition_champs
{
        W Transformation
        {
                methode formule expression 1 (Ux*cos(atan(x/y))-Uy*sin(atan(x/y)))
                sources {
                        Transformation
                        {
                                methode composante numero 0
                                source refchamp { Pb_champ pb vitesse } nom_source Ux } ,
                                Transformation
                                {
                                        methode composante numero 1
                                        source refchamp { Pb_champ pb vitesse }
                                        nom_source Uy
                                }
                        }
                }
        }
```

Another example :

```
# Calculate X component of the pressure force on a sub-boundary named ring #
Domaine ring
Extraire_surface {
       Domaine ring Probleme pb
       Condition_faces (z+2)*(z+1)*(x^2+y^2-0.51)>0 avec_certains_bords 1 Cylindre
 }
...
Read pb {
...
        Definition_champs
        {
                FPx Reduction_0D
                {
                        methode somme source Interpolation
                        {
                                domaine ring localisation elem
                                source Morceau_equation
                                {
                                        type operateur numero 2
                                option flux_bords compo 0
                                source refChamp { Pb_champ pb vitesse }
                            }
                        }
                    }
            }
            Probes { filename   FPx   periode 0.005 numero_elem_sur_maitre 0 }
    }
```

2.19.4GENERAL FIELD POST-PROCESSING

The parameters are:

[**Fichier** *filename*]

The name of the result file will be build with *filename* plus the format name choosen. (example: *channel.lata* if *channel.data* is the data file and LATA the results format). By default, *filename* is the name of the data file. In the case, where **Post_processings** keyword is used (and **Fichier** keyword not specified), *filename* is by default the name of the data file plus the name of the postprocessing block plus the format name choosen (example: *channel_Post_name1.lata*)

[**Format** format]

This optional parameter specifies the format of the output file. The basename used for the output file is the basename of the data file. For the format parameter, choices are **lml**, **lata**, **lata_v1, lata_v2, med** A short description of each format can be found below. The default value is **lml**. The recommended format is **lata**.

[**Domaine** domain_name]

This optional parameter specifies the domain on which the data should be interpolated before it is written in the output file. The default is to write the data on the domain of the current problem (no interpolation).

Fields [**formatte**|**binaire**] **dt_post** string | **nb_pasdt_post** integer { ... }

This parameter specifies which fields should be written. The string given after **dt_post** keyword is the minimum time elapsed in seconds of physical simulation time between two post-processing, it may be a real value or a time expression like 2*exp(-t) if it we want the a decreasing period. It is also possible to specify this period as a number of time steps, thanks to the **nb_pas_dt_post** keyword. A post-processing is always forced at the end of the computation. The optional keywords **formatte** (ASCII) or **binaire** (Binary) are only applicable for the lml and lata format. Binary format is recommended since it is more compact and much faster to read and to write. The default is ASCII output (for lml format) and binary output (for lata format) and time_interval=0 (post-process all computed timesteps)

field_name [localisation]
You can specify as many fields as you want. field_name is the name of a field (example: vitesse), a component of a field (example: vitesse_x), or a post-processing field previously defined in a definition_champs block (the valid fields are the same as for probes, see 2.19.2).
The optional localisation keyword can be equal to **elem** (the post-processed field will be interpolated at the center of the elements of the chosen domain, if it is not already a P0 field), **som** (interpolation on the vertices of the domain), or **faces** (works only with the **lata** format and for fields discretized at the faces of the domain: velocity field in VDF and VEF, temperature field in VEF, ... the field is not interpolated and it is written "as is". This option uses a lot more disk space than the other options and it shows the "non conformity" of the velocity field). The default value for localisation is som. You might want to force smooth results or reduce the amount of data being written with the som option (in vef, fields processed wih som are much smaller), or you might want to get the most detailed representation of the computed field and use the native localisation of the field (watch for "discretisation" messages in the error file).

Format: Optional keyword (set to lml format by default) used to define the file format to which the fields will be written. There are currently four available formats:

lml:
Keyword used to select standard result post-processing. This post-processing results in a *nom_du_cas*.lml file. If the binary option was not requested for post-processing, an ASCII file is produced (refer also to the example in 5.3).

OBSERVATION: currently all the integers need to be written in FORTRAN format fp.q or Ep.q and not Dp.q

nom_code	character string: name of the code used

version	character string: code version
date	3 integers: dd,mm,yy day month year (2 figures per integer)
nom_problème	character string characterising the problem to be processed (may not include blank characters)
comment	remarks (without blank characters)
format	keyword, may be FORMAT or BINAIRE
GRILLE	keyword
nom_grille	character string: grid name
dim_grille	integer: problem dimension (2D 3D)
nb_noeud	integer: number of grid nodes
xi yi zi	co-ordinates of the nodes where i = 1 to nb_noeud (node_nb)
TOPOLOGIE	keyword
nom_topologie	character string characterising topology
nom_ grille	name of the grid to which this topology is related
MAILLE	keyword
nb_maille	integer: mesh number

for each mesh:

type_maille	element type character string: surface elements: POLY4 to POLY8 volume elements: TETRA4 PRISM6 VOXEL8

et

ie1 ie2 .. ien	integers: list of nodes comprising the mesh
FACE	keyword
nb_face	integer: number of faces

for each face:

type_face	face type character string: linear face : LINE2 surface area face: POLY3 to POLY8 and
if1 if2 .. ifm	list of nodes comprising the face
je1 je2	list of elements touching the face
TEMPS	keyword present at each time step
val_temps	time value at the time step in question
CHAMPPOINT	keyword
nom_champ	character string characterising the field
nom_topologie	name of the topology on which the field is defined in points
temps	time value
nom_var	name of the field variable
nb_comp	number of field component(s)
unité	character string specifying the variable unit
type_var	character string characterising the type of variable discretization (P1, P2 ..)
nb_points	number of given points
n³ noeud et valeur du champ	list of data i = 1,nb_points (point_nb)
CHAMPFACE	keyword
nom_champ	character string characterising the field
nom_topologie	name of the topology on which the field is defined by faces
temps	time value
nom_var	name of the field variable
nb_comp	number of field component(s)
unité	character string specifying the variable unit
type_var	character string characterising the variable discretization type (P1, P2, ...)
nb_faces	number of faces on which the field is given
n³ face et valeur du champ	list of data i = 1,nb_faces (face_nb)
CHAMPMAILLE	keyword
nom_champ	character string characterising the field

nom_topologie	name of the topology on which the field is defined by meshes
temps	time value
nom_var	character string characterising the variable
nb_comp	number of field component(s)
unité	character string specifying the variable unit
type_var	character string characterising the variable discretization type (P1, P2, ...)
nb_mailles	number of meshes on which the field is given
n³ maille et valeur du champ	list of data i = 1,nb_mailles (mesh_nb)
FIN	keyword which must complete the graphic file

lata, lata_v1, lata_v2:

Theses keywords (several versions of the format are available, version 1 with **lata_v1** keyword or version 2 with **lata_v2** keyword, the lata keyword is by default setting the version 2 format since the 1.6.4 version) are used to specify a result post-processing format that is broken down into several files. The domain name must also be indicated (see example). This post-processing generates the following files:

- A *nom_du_cas*.lata file containing the post-processing file index
- The *nom_du_cas*.lata.champ.type.domaine.probleme.temps files containing the fields at a given time for the problem domain where:

champ = pressure, speed, temperature, ...
type = som, elem
domaine = domain name
probleme = problem name
temps = multiple points in time of dt_post

med:

Keyword used to write a Med format file (**M**odélisation **E**change **D**onnées). The binary file generated is *nom_du_cas_000n*.med (n is the number of the writing process) but a file *nom_du_cas*.med is also created for the user.

Format	*Usable viewing tools*	*File size for a field backup of over a million meshes*	*Real number precision in the files*
Lml	**Data Vizualiser** (program not included in the package) **Avs Express** (program not included in the package) **Ensight** (program not included in the package) when the lml2ensight interface located in the ENSIGHT directory of the TRUST distribution is used	12 Mb	Double
Lata	**Avs Express** (program not included with the package)	4 Mb	Single

2.19.5FIELD GENERAL POST-PROCESSING FOR STATISTICS

Statistiques: This keyword is used to set the statistics.

Dt_post: This keyword is used to set the calculated statistics write period.
dts: frequency value.
t_deb value: Start of integration time
t_fin value: End of integration time

stat: Set to **Moyenne (average)** to calculate the average of the field *nom_champ* (field name) over time or **Ecart_type (std_deviation)** to calculate the standard deviation (statistic rms) of the field *nom_champ (field_name)* or **Correlation** to calculate the correlation between the two fields *nom_champ* and *second_nom_champ.*

nom_champ: name of the field on which statistical analysis will be performed. Possible keywords are **Vitesse (speed)**, **Pression (pressure)**, **Temperature**, **Concentration,…**

localisation: localisation of post-processed field values (**elem** or **som**).

Example:
```
Statistiques Dt_post dtst {
            t_deb 0.1  t_fin 0.12
          Moyenne Pression
          Ecart_type Pression
}
```

It will write every **dt_post** the mean and standard deviation value:

$t <= t\_deb:$

$\overline{P(t)} = 0$

$< P(t) >= 0$

$t > t\_deb:$

$$\overline{P(t)} = \frac{1}{t - t\_deb} \int_{t\_deb}^{t} P(t)dt$$

$$< P(t) >= \sqrt{\frac{1}{t - t\_deb} \int_{t\_deb}^{t} \left[\overline{P(t)} - P(t)\right]^2 dt}$$

Statistiques_en_serie: This keyword is used to set the statistics. Average on **dt_integr** time interval is post-processed every **dt_integr** seconds

dt_integr value : Period of integration and write period.

stat: Set to **Moyenne (average)** to calculate the average of the field *nom_champ* (field name) over time or **Ecart_type (std_deviation)** to calculate the standard deviation (statistic rms) of the field *nom_champ (field_name).*

nom_champ: name of the field on which statistical analysis will be performed. Possible keywords are **Vitesse (speed)**, **Pression (pressure)**, **Temperature**, **Concentration,…**

localisation: localisation of post-processed field values (**elem** or **som**).

Example:

```
Statistiques_en_serie Dt_integr dtst {
        Moyenne Pression
}
```

Will calculate and write every dtst seconds the mean value:

$$(n+1)dt\_\text{int}egr > t > n * dt\_\text{int}egr, \overline{P(t)} = \frac{1}{t - n * dt\_\text{int}egr} \int_{t\_n*dt\_\text{int}egr}^{t} P(t)dt$$

2.20 PROBLEM RESOLUTION

The **Solve** interpretor allows a previously defined problem to be resolved.

```
Solve pb
```

Solve: Keyword to resolve a problem *pb*

2.21PARALLEL CALCULATION

You need several keywords to run a parallel calculation. First, you will run in sequential mode a data file where you will partition your mesh thanks to the **partition** instruction. Then you will run in parallel mode your complete data file where you will read the partitioned mesh thanks to **Scatter** keyword.

2.21.1PARTITION

The following keywrd is used for parallel calculation to cut a domain for each processor. By default, these keyword is commented in the reference test cases.

```
Partition DOMAIN_OBJECT_NAME
{
        [ periodique N BOUNDARY_NAME_1 BOUNDARY_NAME_2 ... ]
        partition_tool ALGORITHM_NAME { OPTIONS }
        [ larg_joint THICKNESS ]
        [ reorder 0|1 ]
        [ zones_name BASENAME ]
        [ ecrire_decoupage PARTITION_FILENAME ]
        [ ecrire_lata LATAFILE_BASENAME ]
        [ nb_parts_tot N ]
        [ formatte ]
}
```

DOMAIN_OBJECT_NAME : the name of the domain object to cut.
periodique N BOUNDARY_NAME_1 BOUNDARY_NAME_2 ... : N is the number of boundary names given. Periodic boundaries must be declared by this method. The partitionning algorithm will ensure that facing nodes and faces in the periodic boundaries are located on the same processor.
partition_tool ALGORITHM_NAME { OPTIONS } : Defines the partitionning algorithm (the effective C++ object used is "Partitionneur_ALGORITHM_NAME"). Valid algorithms and options are :

```
partition_tool Metis {
        nb_parts N
        [ use_weights ]
        [ pmetis | kmetis ]
      [ nb_essais N ]
}
```

Metis is an external partitionning library. It is a general algorithm that will generate a partition of the domain.
N is the number of non empty parts that must be generated (generally equal to the number of cpus in the parallel run).
If **use_weights** is specified, weighting of the element-element links in the graph is used to force metis to keep opposite periodic elements on the same processor. This option can slightly improve the partitionning quality but it consumes more memory and takes more time. It is not mandatory since a correction algorithm is always applied afterwards to ensure a correct partitionning for periodic boundaries.
The default values are "**pmetis**", default parameters are automatically chosen by Metis. "**kmetis**" is faster than "**pmetis**" option but the last option produces better partitioning quality. In both cases, the partitioning quality may be slightly improved by increasing the "**nb_essais**" option (by default N=1). It will compute N partitions and will keep the best one (smallest edge cut number). But this option is CPU expensive, taking N=10 will multiply the CPU cost of partitioning by 10.
Experiments show that only marginal improvements can be obtained with non default parameters.

```
partition_tool Tranche {
        tranches nx ny [nz]
}
```

This algorithm will create a geometrical partitionning by slicing the mesh in the two or three axis directions, based on the geometric center of each mesh element. nz must be given if dimension=3. Each slice contains the same number of elements (slices don't have the same geometrical width, and for VDF meshes, slice boundaries are generally not flat except if the number of mesh elements in each direction is an exact multiple of the number of slices). First, nx slices in the X direction are created, then each slice is split in ny slices in the Y direction, and finally, each part is split in nz slices in the Z direction. The resulting number of parts is nx*ny*nz.
If one particular direction has been declared periodic, the default slicing (0, 1, 2, ..., n-1)
is replaced by (0, 1, 2, ... n-1, 0), each of the two "0" slices having twice less elements than the other slices.

```
partition_tool Sous_Zones {
        [ sous_zones N SUBZONE_NAME_1 SUBZONE_NAME_2 ... ]
}
```

This algorithm will create one part for each specified subzone. All elements contained in the first subzone are put in the first part, all remaining elements contained in the second subzone in the second part, etc...
If all elements of the domain are contained in the specified subzones, then N parts are created, otherwise, a supplemental part is created with the remaining elements.
If no subzone is specified, all subzones defined in the domain are used to split the mesh.

```
partition_tool Partition {
        domaine DOMAINE_NAME
}
```

This algorithm re-use the partition of the domain named DOMAINE_NAME. It is useful to partition for example a post processing domain. The partition should match with the calculation domain.

```
partition_tool Fichier_Decoupage {
        fichier FILENAME
        [ corriger_partition ]
}
```

This algorithm reads an array of integer values on the disc, one value for each mesh element. Each value is interpreted as the target part number n>=0 for this element. The number of parts created is the highest value in the array plus one. Empty parts can be created if some values are not present in the array.
The file format is ASCII, and contains space, tab or carriage-return separated integer values. The first value is the number nb_elem of elements in the domain, followed by nb_elem integer values (positive or zero).
Contrary to other partitioning algorithms, no correction is applied by default to the partition (eg. element 0 on processor 0 and corrections for periodic boundaries). If "**corriger_partition**" is specified, these corrections are applied.

larg_joint THICKNESS : This keyword specifies the thickness of the virtual ghost zone (data known by one processor though not owned by it). The default value is 1 and is generally correct for all algorithms except the QUICK convection scheme that require a thickness of 2. Since the 1.5.5

version, the VEF discretization imply also a thickness of 2 (except VEF P0). Any non-zero positive value can be used, but the amount of data to store and exchange between cpus grows quickly with the thickness.

reorder 0|1 : If this option is set to 1 (0 by default), the partition is renumbered in order that the processes which communicate the most are nearer on the network. This may slighlty improves parallel performance.

zones_name BASENAME : It is the base name of the .Zone files written on disc. If this keyword is not specified, the geometry is not written on disc (you might just want to generate a "**ecrire_decoupage**" or "**ecrire_lata**").

ecrire_decoupage FILENAME : After having called the partitionning algorithm, the resulting partition is written on disc in the specified filename. See also partition_tool Fichier_Decoupage. This keyword is useful to change the partition numbers. First, you write the partition into a file with the option **ecrire_decoupage**. This file contains the zone number for each element's mesh. Then you can easily permute zone numbers in this file. Then read the new partition to create the .Zones files with the **Fichier_Decoupage** keyword.

ecrire_lata FILENAME : After having called the partitionning algorithm, a .lata file is written, containing the partitionning for visualization purposes. You can check the generated partition.

nb_parts_tot N : Keyword to generates N .Zone files, instead of the default number M obtained after the partitionning algorithm. N must be greater or equal to M. This option might be used to perform coupled parallel computations. Supplemental empty zones from M to N-1 are created. This keyword is used when you want to run a parallel calculation on several domains with for example, 2 cpus on a first domain and 10 on the second domain because the first domain is very small compare to second one. You will write **Nb_parts** 2 and **Nb_parts_tot** 10 for the first domain and **Nb_parts** 10 for the second domain.

formatte : These keyword specify ASCII format for the .Zone files. "**binaire**" is the default and recommended format, but is not actually portable. You must generate the .Zone file on the same computer architecture (big-endian or little endian) than the one used to run the parallel computation. In "**ascii**" (synonym for "**formatte**"), some precision might be lost in the node coordinates.

Restrictions for periodic boundary conditions:

Before the 1.4.8 version, periodic boundaries should be on the same processor. So the partitioning should be appropriate. Since the 1.4.8 version, the rule is: periodic faces should be on the same processor. Examples of good partitioning:

```
 WWWWWWWWW          WWWWWWWWW
P       0        P      P     |       |      P
P---------------------P      P  0   |   1   |   0   P
P       1        P      P     |       |      P
 WWWWWWWWW          WWWWWWWWW
```

P: periodic face
W: wall face

The following partitionning will not run. Normally, it will never happen with **Metis** or **Tranche** algorithms if **periodique** keyword is used to define the periodic boundaries.

```
 WWWWWWWWW          WWWWWWWWW
P        |        P      P              1    P
P    0   |    1   P      P----------|          P
P        |        P      P     0    |-----------P
 WWWWWWWWW          WWWWWWWWW
```

2.21.2SCATTER

Keyword to read a partionned mesh during a parallel calculation.

```
Scatter name.Zones domain_name
```

Scatter name.Zones domain_name: Keyword to read the partitions of the domain domain_name in the files called name_0001.Zones to name_000n.Zones. The files are by default in binary format since the 1.4.8 version. To read formatted .Zones files from an older version, use the **ScatterFormatte** keyword:

```
ScatterFormatte name.Zones domain_name
```

```
ScatterMED  domain_name  file.med
```

This keyword will read the partition of the domain_name domain into a the MED format files file.med created by Medsplitter.

2.21.3MPIRUN

Command line to run a parallel calculation. First the data file (ex: study.data) must contain directive **Partition** to partition the domain and the TRUST binary must be ran in sequential mode.

$ $exec study 1>out 2>err

Then, once the files containing the partitions are generated, change the data file to add the **Scatter** directive to read theses files. Then, we can run TRUST in parallel mode thanks to mpirun:

$ **mpirun -np** n $exec study n 1>out 2>err

n is the number of cpus which should match the number of partitions.

2.22 TOOLS

2.22.1 POST PROCESSING

2.22.1.1 Lata2dx

lata2dx is a external tool, which can be used with command lines, to convert LATA or LML files to LATA, OPENDX or PRM files. The source files are located in $TRUST_ROOT/Outils/lata2dx/lata2dx. The LATA plugin ($TRUST_ROOT/Outils/VisIt /plugins/lata) used to import LATA or LML files into VisIt is built with some classes of the lata2dx tool.

The tool lata2dx is compiled during TRUST build process. The binary lata2dx is located into $TRUST_ROOT/exec

How to use lata2dx is given by running lata2dx:

```
veymont.intra.cea.fr:/work/triou > lata2dx
Usage : lata2dx input_file_name
 [timestep=n]
 [domain=name]
 [component=label]
 [[binary|ascii]] [bigendian|littleendian]] [[int32|int64]] [[real32|real64]]
   [[binout|asciiout]] [[bigendianout|littleendianout]] [[int32out|int64out]] [[real32out|
real64out]]
 [forcegroup]
 [regularize=tolerance [invalidate]]
 [reconnect=tolerance]
 [verbosity=n]
[fortranblocs=no]
…
```

So we will not describe all the options and will just give some few examples. By default, lata2dx converts a LATA file into a OPENDX file on the standard output.

To convert a binary LATA file to an ASCII LATA file:
lata2dx input.lata writelata_convert=output.lata asciiout fortranblocs=no
To convert a LML file to an LATA file:
lata2dx input.lml writelata_convert=output.lata
To select a mesh at several timesteps and several fields:
lata2dx input.lata writelata_convert=output.lata timestep=N1 timestep=N2 ... domain=MESH1 component=FIELD1 component=FIELD2
To calculate a time average:
lata2dx input.lata writelata=avg.lata timeaverage=rectangles rms_fluctuations
Each fiel dis replaced by its time average into the input file input.lata. rms_fluctuatuions adds new fields rms_fluct_XXX for each field XXX.
To build a new LATA file with a reconnect partitioned mesh in a parallel calculation:
lata2dx input.lata writelata=output.lata reconnect=epsilon
Where epsilon (~1.e-7*biggest size of the mesh) is the biggest length to considerate two points as separated.
…

2.22.2KEYWORD USEFUL FOR DEBUGGING

2.22.2.1Debog

Keyword to debug some differences between two TRUST versions on a same data file.

Debog problem_name file_to_write_domain file_to_write_faces error mode_debog

problem_name : Name of the problem to debug
file_to_write_domain : Name of the file where domain will be written in sequential calculation
file_to_write_faces : Name of the file where faces will be written in sequential calculation
error : Minimal value (by default 1.e-20) for the differences between the two codes
mode_debog : By default -1 (nothing is written in the different files), you will set 0 for the run with the first code, and 1 for the run with the second code.

If you want to compare the results of the same code in sequential and parallel calculation, first run (mode_debog=0) in sequential mode (the files file_to_write_domain an file_to_write_faces will be written first) then the second run in parallel calculation (mode_debog=1).

During the first run (mode_debog=0), it prints into the file DEBOG, values at different points of the code thanks to the C++ instruction call.
see for example in Noyau/Resoudre.cpp file the instruction:
Debog::verifier(msg,value);

Where msg is a string and value may be a double, integer or array.

During the second run (mode_debog=1), it prints into a file Err_Debog.dbg the same messages than in the DEBOG file and checks if the differences between results from the two codes are less than error. If not, it prints Ok else show the differences and the lines where it occured.

Example:
dimension 2
Pb_Thermohydraulique pb
...
Discretize pb dis
Debog pb seq faces 1.e-6 0
Read pb { ... }
Solve pb

3.FILES EXAMPLES

3.1MESH FILES

The following is an example of a commented TRUST mesh file:

```
ENVEL     <- Nom du domaine (domain name)
2      <- Nombre de valeurs a lire ensuite (number of values to be then read)
60 3     <- Nombre de sommets et dimension (number of peaks and dimension)
180     <- Nombre de valeurs a lire ensuite (number of values to be then read
.0 .0 .0    <- Liste des coordonnees x y z des sommets (list of peak x y z co-ordinates)
.0 .0 .5
.0 .0 1.0
.0 .0 1.5
.0 .0 2.0
.0 .0 2.5
.0 .0 3.0
.0 .5 .0
.0 .5 .5
.0 .5 1.0
.0 .5 1.5
.0 .5 2.0
.0 .5 2.5
.0 .5 3.0
.0 1.0 .0
.0 1.0 .5
.0 1.0 1.0
.0 1.0 1.5
.0 1.0 2.0
.0 1.0 2.5
.0 1.0 3.0
.5 .0 .0
.5 .0 .5
.5 .0 1.0
.5 .0 1.5
.5 .0 2.0
.5 .0 2.5
.5 .0 3.0
.5 .5 .0
.5 .5 .5
.5 .5 1.0
.5 .5 1.5
.5 .5 2.0
.5 .5 2.5
.5 .5 3.0
.5 1.0 .0
.5 1.0 .5
.5 1.0 1.0
.5 1.0 1.5
.5 1.0 2.0
.5 1.0 2.5
.5 1.0 3.0
1.0 .0 .0
1.0 .0 .5
1.0 .0 1.0
1.0 .0 1.5
```

```
 1.0 .0 2.0
 1.0 .0 2.5
 1.0 .5 .0
 1.0 .5 .5
 1.0 .5 1.0
 1.0 .5 1.5
 1.0 .5 2.0
 1.0 .5 2.5
 1.0 1.0 .0
 1.0 1.0 .5
 1.0 1.0 1.0
 1.0 1.0 1.5
 1.0 1.0 2.0
 1.0 1.0 2.5

 {      <- Debut de la definition des zones du domaine (start of domain area definition)
 VOLUME1    <- Nom de la zone (area name)
 Hexaedre    <- Type de l'element (HEXAEDRE,TETRAEDRE,RECTANGLE,TRIANGLE) (element type
(HEXAGON,TETRAHEDRAL,RECTANGLE,TRIANGLE)
 2      <- Nombre de valeurs a lire ensuite (number of values to be then read)
 22 8      <- Nombre de mailles et nombre de sommets par maille (number of meshes and number of peaks per mesh)
 176      <- Nombre de valeurs a lire ensuite (number of values to be then read)
 0 1 7 8 21 22 28 29 <- Liste des sommets de chaque maille (list of peaks for each mesh)
 1 2 8 9 22 23 29 30 <- A noter que la numerotation des sommets demarre de 0 (it should be noted that peak numbering starts at 0)
 2 3 9 10 23 24 30 31
 3 4 10 11 24 25 31 32
 4 5 11 12 25 26 32 33
 5 6 12 13 26 27 33 34
 7 8 14 15 28 29 35 36
 8 9 15 16 29 30 36 37
 9 10 16 17 30 31 37 38
 10 11 17 18 31 32 38 39
 11 12 18 19 32 33 39 40
 12 13 19 20 33 34 40 41
 21 22 28 29 42 43 48 49
 22 23 29 30 43 44 49 50
 23 24 30 31 44 45 50 51
 24 25 31 32 45 46 51 52
 25 26 32 33 46 47 52 53
 28 29 35 36 48 49 54 55
 29 30 36 37 49 50 55 56
 30 31 37 38 50 51 56 57
 31 32 38 39 51 52 57 58
 32 33 39 40 52 53 58 59

 {          <- Debut de la definition des bords de la zone (start of area edge definition)
 LAT      <- Nom du bord (edge name)
  QUADRANGLE_3D <- Type des elements de bords (QUADRANGLE_3D,TRIANGLE_3D,SEGMENT_2D) (type of edge elements)
(QUADRANGLE_3D,TRIANGLE_3D,SEGMENT_2D)
 2          <- Nombre de valeurs a lire ensuite (number of valules to be then read)
 32 4       <- Nombre de faces de bord et nombre de sommets par face (number of edge faces and number of peaks per face)
 128        <- Nombre de valeurs a lire ensuite (number of values to be then read)
 0 1 7 8  <- Liste des sommets de chaque face de bord (list of peaks for each face of the edge)
 0 7 21 28
 1 2 8 9
 2 3 9 10
 3 4 10 11
```

```
4 5 11 12
5 6 12 13
26 27 33 34
6 13 27 34
7 8 14 15
7 14 28 35
8 9 15 16
9 10 16 17
10 11 17 18
11 12 18 19
12 13 19 20
33 34 40 41
13 20 34 41
42 43 48 49
21 28 42 48
43 44 49 50
44 45 50 51
45 46 51 52
46 47 52 53
26 33 47 53
48 49 54 55
28 35 48 54
49 50 55 56
50 51 56 57
51 52 57 58
52 53 58 59
33 40 53 59

2       <- Nombre de valeurs a lire ensuite (number of values to be then read)
32 2    <- Nombre de faces de bord et nombre de sommets par face (number of edge faces and number of peaks per face)
64      <- Nombre de valeurs a lire ensuite (number of values to be then read)
-1 -1   <- Numeros des mailles de chaque cote de la face (numbers of   meshes on each side of the face)
-1 -1   <- Necessaire meme si cela n'est pas encore utilise par TRUST (necessary, even if it is not yet used by TRUST)
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
```

```
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
```

```
,      <- Virgule pour separer la definition des bords (comma to separate edge definition
NOR
QUADRANGLE_3D
2
11 4
44
14 15 35 36
15 16 36 37
16 17 37 38
17 18 38 39
18 19 39 40
19 20 40 41
35 36 54 55
36 37 55 56
37 38 56 57
38 39 57 58
39 40 58 59

2
11 2
22
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1

,
SUD
QUADRANGLE_3D
2
11 4
44
0 1 21 22
1 2 22 23
2 3 23 24
3 4 24 25
4 5 25 26
```

```
5 6 26 27
21 22 42 43
22 23 43 44
23 24 44 45
24 25 45 46
25 26 46 47

2
11 2
22
-1 -1
```

```
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1

}       <- Fin de la definition des bords de la zone (end of area edge definition)
vide    <- Necessaire meme si cela n'est pas encore utilise par TRUST (this is required, even though it is not yet used by TRUST)
vide    <- Idem (ditto)
vide    <- Idem (ditto)
}       <- Fin de la definition des zones du domaine (end of domain area definition)
vide    <- Idem (ditto)
```

3.2 DATA SET FILES

For examples of data set files, see under a TRUST distribution in either the **Tests_reference** directory (simple test cases) or **Validation** directory (more complicated test cases).

3.3 RESULT FILE

```
If the binary option was not requested for postprocessing, an ASCII file
nom_du_cas.lml is obtained, which has the following format:

nom_code             character string: name of the code used
version              character string: code version
date                 3 integers: dd,mm,yy day month year with 2
                     figures per integer
nom_probleme         character string characterising the problem
                     to be processed (without spaces)
comment              remarks (without blank characters)

format               keyword FORMAT or BINAIRE

GRILLE               keyword

nom_grille           character string: grid name
dim_grille           integer: problem dimension (2D, 3D)
nb_noeud             integer: number of nodes in the grid

xi yi zi             node co-ordinates for i = 1 to nb_nœud (node number)

TOPOLOGIE            keyword

nom_topologie        character string characterising the topology
nom_grille           name of the grid related to this topology

MAILLE               keyword

nb_maille            integer: number of meshes

   for each mesh:
type_maille          element type character string:
              surface elements: POLY4 a POLY8
              volume elements:          TETRA4
                                PRISM6
                                VOXEL8
  and

ie1 ie2 .. ien       integers: list of nodes comprising the mesh

FACE                 keyword

nb_face              integer:   number of faces

      for each face:
type_face            type face character string
```

	linear faces: LINE2
	surface area faces: POLY3 a POLY8
and	
if1 if2 .. ifm	list of nodes comprising the face
je1 je2	list of elements touching the face
TEMPS	keyword present at each time step
val_temps	time value at the time step in question
CHAMPPOINT	keyword
nom_champ	character string characterising the field
nom_topologie	name of the topology on which the field is defined in points
temps	time value
nom_var	name of the field variable
nb_comp	number of field components
unite	character string specifying the variable unit
type_var	character string characterising the variable discretization type P1, P2, etc.)
nb_points	number of given points
n^3 noeud et valeur du champ	list of data i = 1,nb_points (number of points)
CHAMPFACE	keyword
nom_champ	character string characterising the field
nom_topologie	name of the topology on which the field is defined in faces
temps	time value
nom_var	name of the field variable
nb_comp	number of field components
unite	character string specifying the variable unit
type_var	character string characterising the variable discretization type (P1, P2, etc.)
nb_faces	number of faces on which the field is given
n^3 face et valeur du champ	list of data i = 1,nb_faces (number of faces)
CHAMPMAILLE	keyword
nom_champ	character string characterising the field
nom_topologie	name of the topology on which the field is defined in meshes
temps	time value
nom_var	character string characterising variable
nb_comp	number of field components
unite	character string specifying the variable unit
type_var	character string characterising the variable discretization type (P1, P2, etc.)
nb_mailles	number of meshes on which the field is given
n^3 maille et valeur du champ	list of data i = 1,nb_mailles (number of meshes)
FIN	keyword which should end the graph file

An example of a lml file, example.lml:

```
Trio_U  Version1 01/09/96
exemple
Trio_U
 GRILLE
Grille_dom   3  18
0.00000000e+00  0.00000000e+00  0.00000000e+00
5.00000000e-02  0.00000000e+00  0.00000000e+00
1.00000000e-01  0.00000000e+00  0.00000000e+00
0.00000000e+00  5.00000000e-02  0.00000000e+00
5.00000000e-02  5.00000000e-02  0.00000000e+00
1.00000000e-01  5.00000000e-02  0.00000000e+00
0.00000000e+00  1.00000000e-01  0.00000000e+00
5.00000000e-02  1.00000000e-01  0.00000000e+00
1.00000000e-01  1.00000000e-01  0.00000000e+00
0.00000000e+00  0.00000000e+00  1.00000000e+00
5.00000000e-02  0.00000000e+00  1.00000000e+00
1.00000000e-01  0.00000000e+00  1.00000000e+00
0.00000000e+00  5.00000000e-02  1.00000000e+00
5.00000000e-02  5.00000000e-02  1.00000000e+00
1.00000000e-01  5.00000000e-02  1.00000000e+00
0.00000000e+00  1.00000000e-01  1.00000000e+00
5.00000000e-02  1.00000000e-01  1.00000000e+00
1.00000000e-01  1.00000000e-01  1.00000000e+00
TOPOLOGIE
Topologie_Cavite   Grille_dom
MAILLE
4
VOXEL8   1  2  4  5  10  11  13  14
VOXEL8   2  3  5  6  11  12  14  15
VOXEL8   4  5  7  8  13  14  16  17
VOXEL8   5  6  8  9  14  15  17  18
FACE
0
TEMPS 0.00000000e+00
CHAMPPOINT pression_som_dom   Topologie_Cavite   0.00000000e+00
pression_som_dom   1  Pa.m3/kg
  type0  18
1  0.00000000e+00
2  0.00000000e+00
3  0.00000000e+00
4  0.00000000e+00
5  0.00000000e+00
6  0.00000000e+00
7  0.00000000e+00
8  0.00000000e+00
9  0.00000000e+00
10  0.00000000e+00
11  0.00000000e+00
12  0.00000000e+00
13  0.00000000e+00
14  0.00000000e+00
15  0.00000000e+00
16  0.00000000e+00
17  0.00000000e+00
18  0.00000000e+00
CHAMPMAILLE vitesseX_elem_dom   Topologie_Cavite   0.00000000e+00
vitesseX_elem_dom   1  m/s
  type0  4
1  0.00000000e+00
2  0.00000000e+00
```

```
3  0.00000000e+00
4  0.00000000e+00
CHAMPMAILLE vitesseY_elem_dom   Topologie_Cavite   0.00000000e+00
vitesseY_elem_dom   1  m/s
  type0  4
1  0.00000000e+00
2  0.00000000e+00
3  0.00000000e+00
4  0.00000000e+00
CHAMPPOINT vitesse_som_dom   Topologie_Cavite   0.00000000e+00
vitesse_som_dom   3  m/s
  type1  18
1  0.00000000e+00  0.00000000e+00  0.00000000e+00
2  0.00000000e+00  0.00000000e+00  0.00000000e+00
3  0.00000000e+00  0.00000000e+00  0.00000000e+00
4  0.00000000e+00  0.00000000e+00  0.00000000e+00
5  0.00000000e+00  0.00000000e+00  0.00000000e+00
6  0.00000000e+00  0.00000000e+00  0.00000000e+00
7  0.00000000e+00  0.00000000e+00  0.00000000e+00
8  0.00000000e+00  0.00000000e+00  0.00000000e+00
9  0.00000000e+00  0.00000000e+00  0.00000000e+00
10  0.00000000e+00  0.00000000e+00  0.00000000e+00
11  0.00000000e+00  0.00000000e+00  0.00000000e+00
12  0.00000000e+00  0.00000000e+00  0.00000000e+00
13  0.00000000e+00  0.00000000e+00  0.00000000e+00
14  0.00000000e+00  0.00000000e+00  0.00000000e+00
15  0.00000000e+00  0.00000000e+00  0.00000000e+00
16  0.00000000e+00  0.00000000e+00  0.00000000e+00
17  0.00000000e+00  0.00000000e+00  0.00000000e+00
18  0.00000000e+00  0.00000000e+00  0.00000000e+00
TEMPS 2.00000000e-02
CHAMPPOINT pression_som_dom   Topologie_Cavite   2.00000000e-02
pression_som_dom   1  Pa.m3/kg
  type0  18
1  -3.72315262e-07
2  0.00000000e+00
3  3.72315262e-07
4  0.00000000e+00
5  0.00000000e+00
6  0.00000000e+00
7  3.72315262e-07
8  0.00000000e+00
9  -3.72315262e-07
10  -3.72315262e-07
11  0.00000000e+00
12  3.72315262e-07
13  0.00000000e+00
14  0.00000000e+00
15  0.00000000e+00
16  3.72315262e-07
17  0.00000000e+00
18  -3.72315262e-07
CHAMPMAILLE vitesseX_elem_dom   Topologie_Cavite   2.00000000e-02
vitesseX_elem_dom   1  m/s
  type0  4
1  0.00000000e+00
2  -1.48926105e-07
3  0.00000000e+00
4  1.48926105e-07
CHAMPMAILLE vitesseY_elem_dom   Topologie_Cavite   2.00000000e-02
vitesseY_elem_dom   1  m/s
```

```
  type0  4
1  0.00000000e+00
2  0.00000000e+00
3  1.48926105e-07
4  -1.48926105e-07
CHAMPPOINT vitesse_som_dom   Topologie_Cavite   2.00000000e-02
vitesse_som_dom   3  m/s
  type1  18
1  0.00000000e+00  0.00000000e+00  0.00000000e+00
2  0.00000000e+00  0.00000000e+00  0.00000000e+00
3  0.00000000e+00  0.00000000e+00  0.00000000e+00
4  0.00000000e+00  0.00000000e+00  0.00000000e+00
5  0.00000000e+00  0.00000000e+00  0.00000000e+00
6  0.00000000e+00  0.00000000e+00  0.00000000e+00
7  0.00000000e+00  0.00000000e+00  0.00000000e+00
8  0.00000000e+00  0.00000000e+00  0.00000000e+00
9  0.00000000e+00  0.00000000e+00  0.00000000e+00
10  0.00000000e+00  0.00000000e+00  0.00000000e+00
11  0.00000000e+00  0.00000000e+00  0.00000000e+00
12  0.00000000e+00  0.00000000e+00  0.00000000e+00
13  0.00000000e+00  0.00000000e+00  0.00000000e+00
14  0.00000000e+00  0.00000000e+00  0.00000000e+00
15  0.00000000e+00  0.00000000e+00  0.00000000e+00
16  0.00000000e+00  0.00000000e+00  0.00000000e+00
17  0.00000000e+00  0.00000000e+00  0.00000000e+00
18  0.00000000e+00  0.00000000e+00  0.00000000e+00
FIN
```

4.PUBLICATIONS

Notes:

STR/LML/92136
"Projet TRIO Unitaire – Premières spécifications théoriques" (Unit TRIO project - first theoretical specifications)
O. CUETO, J.P. MAGNAUD, M.VILLAND

STR/LML/93-183
"Développement de TRIO-U: planning prévisionnel" (TRIO-U development: forecast scheduling)
M. FARVACQUE, J.C. MICAELLI

STR/LMTL/96-20
"TRIO-U: Document de conception TRIO-U Version1" (TRIO U: TRIO-U Version1 design documentation)
M. FARVACQUE, O. CUETO, Ph. EMONOT

STR/LMTL/96-21
"TRIO-U: Manuel d'utilisation" (TRIO U: User manual)
P. LEDAC

STR/LMTL/96-36
"TRIO-U: Manuel informatique" (TRIO U: Computer manual)
M. FARVACQUE

STR/LMTL/96-88
"TRIO-U: User manual"
O. CUETO, Ph. EMONOT , P. LEDAC

SMTH/LATA/97-001
F. Barré, D. Laurence
"Etude d'opportunité – Modélisation des écoulements turbulents" (Opportunity analysis - modelling of turbulent flow)

SMTH/LATA/97-003
B Bollini, Y. Hascoët
"Conception et développement d'une IHM pour le code de thermohydraulique TRIO-U" (design and development of an GUI for the TRIO-U thermohydraulic code)
Work placement report

SMTH/LATA/97-006
ASilveira Neto, Ph. Emonot
"Simulation numérique fine des écoulements turbulents diphasiques non miscibles" (fine digital simulation of non-miscible disphase turbuent flow)

SMTH/LATA/97-009
B Piuze, Ph. Emonot
"Conception et développement d'une IHM pour TRIO-U" (design and development of an GUI for TRIO-U)
Work placement report

SMTH/LATA/97-010

TRIO-U Work Group
"Plan de développement de TRIO-U version 2 – Objectifs, contenu du noyau de la version 2, architecture logiciel, co-développement" (TRIO-U version 2 development schedule - goals, content of version 2 core, software architecture, co-development)

SMTH/LATA/97-012
Barsamian, O. Cueto, Ph. Emonot
"Application of the dynamic subgrid scale model to TRIO-U"
Technical report

SMTH/LATA/97-013
C. Calvin, P. Ledac
"Mesures de performances de TRIO-U sur machines scalaires et parallèles" (measurement of TRIO-U performance on scalar and parallel machines)

SMTH/LATA/97-014
TRIO-U Work Group
"Cahier des charges de l'audit de la version 1 de TRIO-U" (TRIO-U version 1 audit specification)

SMTH/LATA/97-015
C. Calvin, M. Cordebard
"Intégration d'un découpeur de domaines dans le logiciel de calcul TRIO-U" (incorporation of a domain partitionner into the TRIO-U calculation software)
Work replacement report

SMTH/LATA/97-018
C. Calvin, Ph. Emonot
"The Parallelism in the TRIO-Unitaire Project"
Communication presented at NURETH'8, Japan, 30/09/97-04/10/97

SMTH/LATA/97-021
ASilveira Neto, Ph. Emonot
"The front-tracking method for interface transport"
Communication presented at "European two-phase flow group meeting", Brussels 6-7/061997

SMTH/LATA/97-023
Barsamian, O. Cueto, Ph. Emonot
"Application of the dynamic subgrid scale model to TRIO-U"
(Further information to note 97-12)

SMTH/LATA/97-024
C. Calvin, Ph. Emonot
"The TRIO-Unitaire Project: a parallel CFD 3-dimensional code"
Communication presented at ISCOPE'97, USA, 08-11/12/97

SMTH/LATA/97-026
F. Barré, I. Toumi
"Module diphasique tridimensionnel avec approche moyennée de TRIO-U: cahier des charges" (tridimensioned two phase module with the TRIO-U averaging method: specifications)

SMTH/LATA/97-028
O. Cueto, C. Calvin, Ph. Emonot

"Principes généraux de la structure logicielle de la version 1 de TRIO-U" (main principles of TRIO-U version 1 software structure)

SMTH/LATA/98-30
F. Barré, D. de Crécy, D. Bestion, Ph Emonot, AForestier, J. Gauvain, J.P. Magnaud, I. Toumi, M. Grandotto, N. Thuy
"Organisation du projet TRIO-U" (TRIO-U project organisation)

SMTH/LATA/98-33
C.Calvin
"Manuel utilisateur de Trio_U parallèle et environnement d'utilisation sur CRAY T3E" (operation environment on CRAY T3E and parallel Trio_U user manual)

SMTH/LATA/98-41
P. Barron, C. Dumas, C.Calvin
"Introduction de méthodes de type multi-grilles dans le code TRIO-U" (Introduction of multi-grid type methods in TRIO-U code)
Work replacement report

SMTH/LATA/98-37
BMenant
"Analyse à l'aide de TRIO-U du fonctionnement thermohydraulique actuel de CASCAD" (analysis of current CASCAD thermohydraulic operation using TRIO-U)

SMTH/LATA/98-42
BMenant
"Mise en oeuvre de TRIO-U en vue de l'analyse du fonctionnement thermohydraulique actuel de CASCAD" (implementation of TRIO-U with the aim of analysing current CASCAD thermohydraulic operation)

SMTH/LATA/98-45
BMenant
"Application de TRIO-U à l'étude du transit d'un bouchon d'eau claire dans un circuit primaire de REP" (application of TRIO-U to the study of the transition of a plug of clear water through a primary REP system)

SMTH/LATA/98-46
KLatour, O. Cueto
"Introduction de la discrétisation P1-P1 dans le logiciel PRICELES" (introduction of P1-P1 discretization in PRICELES software)
Work replacement report

SMTH/LATA/98-50
U. Bieder, Ph. Emonot, D. Laurence
"PRICELES. Summary of the numerical scheme"

SMTH/LATA/99-56
C. Ackermann
"Modélisation sous-maille dans le logiciel CEA/EDF PRICELES – 1ère partie: tests de validation en maillages structurés" (sub-grid modelling in the CEA/EDF PRICELES software - 1st section: validation tests in structured meshes)

SMTH/LATA/99-73
U. Bieder
"PRICELES. Tests of the numerical scheme"

SMTH/LATA/99-64
C.Calvin, Ph Emonot
"Etude préliminaire sur l'utilisation de la STL dans TRIO-U V2" (preliminary study concerning the use of STL in TRIO-U V2)

SMTH/LATA/99-65
C.Calvin
"Document de conception détaillée du noyau de la version 1 de TRIO-U" (detailed design document concerning the core of TRIO-U version 1)

SMTH/LATA/99-66
C.Calvin
"Document de conception détaillée de la version 1 de TRIO-U: introduction" (TRIO-U version 1 detailed design document: introduction)

SMTH/LATA/99-67
C.Calvin
"Document de conception détaillée du module géométrie de la version 1 de TRIO-U" (TRIO-U version 1 geometry module detailed design document)

SMTH/LATA/99-73
U. Bieder
"PRICELES. Large Eddy Simulation of the very near wake of a circular cylinder"

SMTH/LATA/99-74
O. Cueto, G. Fauchet
"Un premier résultat du module diphasique de TRIO-U" (first result for the TRIO-U two phase module)

Publications:

C. Calvin, Ph. Emonot
"The TRIO-U project: a parallel CFD 3-dimensional code"
ISCOPE'97, USA, 08-11/12/97

C. Calvin, Ph. Emonot
"The Parallelism in the TRIO-Unitaire Project"
NURETH'8, Japan, 30/09/97-04/10/97

M. Farvacque, O. Cueto, F. Barré, Ph. Emonot
"TRIO-U: a new generation of thermalhydraulics computer code"
NURETH'8, Japan, 30/09/97-04/10/97

C. Calvin
"Large thermalhydraulic 3D simulations using TRIO-U code on CRAY T3E"
3rd European SGI/CRAY MPP Workshop, Paris, France, 11-12/09/07

C. Ackerman
"Modèles sous-maille pour la thermohydaulique des réacteurs" (sub-grid model for reactor thermohydraulics)
Séminaire du Centre de physiques des Houches sur les écoulements turbulents complexes, (Houche physics centre symposium on complex turbulent flow), France, 04-07/05/99

U. Bieder, C. Calvin, Ph Emonot

"Industrial application of Large Eddy Simulations: validation of a new numerical scheme"
8th International Symposium on CFD, Brême, Germany, 5-10/09/99

O. Cueto
"Module diphasique de TRIO-U: la méthode ICE" (TRIO-U two phase module: the ICE method)
Atelier sur les schémas de flux pour la simulation numérique des écoulements diphasiques (workshop concerning flux schemes for digital simulation of two phase flows), Cargèse, France, 22-24/09/99

I. Toumi, A Kumbaro, H. Paillère, F. Barré, O. Cueto
"Numerical methods and physical models for two-phase flow simulations in the TRIO-U code"
9th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH'9), San Francisco, USA, 3-8/10/99

C. Calvin
"TRIO-U: le code avancé de mécanique des fluides du CEA/DRN" (TRIO-U: CEA/DRN's advanced fluid mechanics code)
4th Convention of CEA partners, Grenoble, France, 16/11/99

F. Barré, D. Laurence
"Priceles, une plate-forme avancée pour la simulation de la turbulence" (Priceles, an advanced platform for turbulence simulation)
4th Convention of CEA partners, Grenoble, France, 16/11/99

C. Calvin
"TRIO-U: le code avancé de mécanique des fluides du CEA/DRN" (TRIO-U: CEA/DRN's advanced fluid mechanical code)
4th Convention of CEA partners, Grenoble, France, 16/11/99

U. Bieder, C. Calvin, Ph. Emonot.
"PRICELES: AParallel CFD 3-Dimensional Code for Industrial Large Eddy Simulations"
Parallel CFD 2000, Trodheim-Norway, 2000.

U. Bieder, C. Calvin, Ph. Emonot.
"PRICELES: An Object Oriented Code for Industrial Large Eddy Simulations"
CFD2K, Montréal-Canada, 2000.

5.FRENCH-ENGLISH DICTIONNARY FOR TRUST KEYWORDS

Although most keywords in TRUST have English counterparts with a similar spelling, there are some exceptions and users not so familiar with English would anyway not find straightforward to guess the English translation thus the meaning of a TRUST keyword.
These pages are intended to help non French speaking TRUST users to get familiar with the keywords used throughout the input data file of TRUST.

Easy ones:

Fortunately the most numerous, just a couple of letters at the end change between English and French

- Probleme, Domaine, Origine, Limite, Frontiere, Initiale, Molaire, Uniforme = problem, domain, origin, limit, frontier, initial, molar, uniform
- Objet = object
- Schema = scheme
- Pave = pave (to pave the floor with stones, tiles, cobbles…)
- Homogene = homogeneous
- Thermohydraulique, Volumique, Surfacique, Hyperbolique, periodique, adiabatique = thermal-hydraulic, volumic, surfasic, hyperbolic, periodic, adiabatic
- Tabule = tabulated
- Tangentiel = tangential
- Solide, fluid = solid, fluid
- Porosite, diffusivite, viscosite = porosity, diffusivity, viscosity

A bit harder to guess… ?

- Calculer (abbreviated as “calc”)= to calculate, to compute
- Solveur = solver
- Nombre = number
- Facteur = factor
- Sous-zone = sub-zone
- Champ = field (of a variable)
- Morceau = a piece, a chunk
- Echange = exchange

Names:

- Temps = time
- Vitesse = velocity
- Pression = pressure
- Chaleur = heat
- Paroi = wall
- Fichier = file
- Sonde = probe
- Amont = upwind
- Moyenne = average
- Ecart_type = root mean square
- Longueur = length
- Noeud = node (of a grid)
- Bord = edge
- Chapeau = hat (cosine shaped)
- Tourbillon = vortex

- Bruit = noise
- Perte de charge = pressure loss, head loss

Verbs:

- Ecrire = to write
- Trianguler = to mesh a 2D surface with triangles
- Tetraedriser = to mesh a 3D volume with tetrahedrons
- Imprimer = to print
- Sauvegarde = the action of saving the job results
- Reprise = the action of restarting a job from previously saved results

Adjectives:

- Parfait = perfect(for a gas)
- Impose(e) = imposed
- Ouvert(e) = open
- Defilante = moving
- Bas = low

6.TEST CASES INDEX

Test cases can be found in the $TRUST_ROOT/tests directory.

You have access to useful resources in the $TRUST_ROOT/index.html file with your favourite browner (eg: firefox). To find test case examples containing a particular keyword thanks to the Keywords link ($TRUST_ROOT/tests/Reference/index_keywords_tests.html):

```
   firefox $TRUST_ROOT/index.html &
or trust -index &
```

7.KEYWORD INDEX