TRUST Generic Guide V1.7.5

Support team: triou@cea.fr

Link to: **Project Reference Manual**

June 7, 2017

Contents

Chapter 1

Introduction

This document constitutes the generic guide for **TRUST software** and his **Baltik projects**.

TRUST is a thermohydraulic software package for CFD simulations for incompressible monophasic/diphasic flow.

You can create new project based on **TRUST** plateform. Theses projects are named **"BALTIK"** projects.

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 validated 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).

1.1 Before TRUST: a modular software named Trio_U

TRUST was born from the cutting in two pieces of **Trio_U** software. **Trio_U** was a software brick based on the **Kernel** brick (which contains the equations, space discretizations, numerical schemes, parallelism...) and used by other CEA applications (cf Figure 1.1).

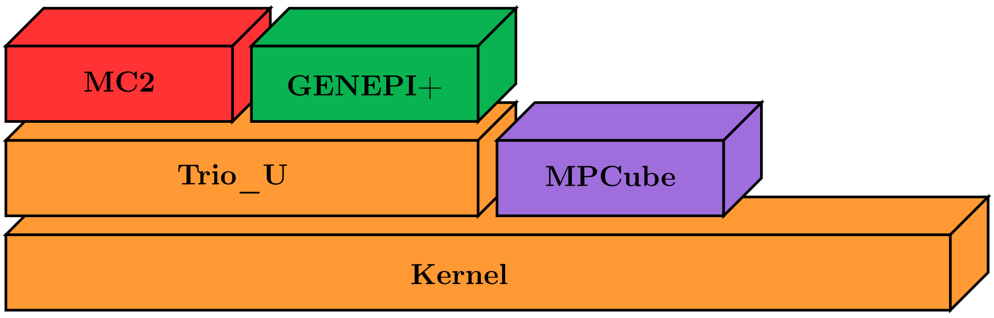


Figure 1.1: Trio_U: brick software

We could create new projects based on Kernel brick or **Trio_U** brick. Theses projects were named **"BALTIK"** projects: "**B**uild an **A**pplication **L**inked to **T**r**I**o_U **K**ernel".

In 2015, **Trio_U** was divided in two parts: **TRUST** and **TrioCFD**.

- **TRUST** is a new platform, its name means: "**TR**io_**U** **S**oftware for **T**hermohydraulics",
- **TrioCFD** is a BALTIK project based on **TRUST**, which contains the following models: FT, Radiation, LES, zoom...

Here is the structure of **TRUST** platform (cf Figure 1.2):

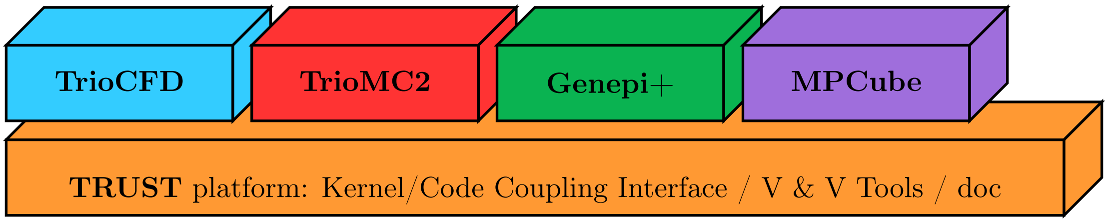


Figure 1.2: TRUST platform & his BALTIKs

Note that: **Trio_U** = **TRUST** + **TrioCFD**.

1.2 Short history

TRUST is developed at the CEA/DEN/DANS/DM2S/STMF service. The project starts in 1994 and improved versions were built ever since:

- 1994: start of the project Trio_U
- 01/1997: v1.0 (VDF only)
- 06/1998: v1.1 (VEF version)
- 04/2000: v1.2 (parallel version)
- 07/2001: v1.3 (radiation model)
- 11/2002: v1.4 (new LES turbulence models)
- 02/2006: v1.5 (VDF/VEF Front Tracking)
- 10/2009: v1.6 (data structure revamped)
- 06/2015: v1.7 (cut into **TRUST** and **TrioCFD** + switch to open source)

1.3 Data file

To launch a calculation with **TRUST**, you need to write a "data file" which is an input file for **TRUST** and will contain all the informations about your simulation. Data files are written following some rules as shown below. But their language is not a programming language, users can't make loops or switch...

Note that:

- lines between # ... # and /* ... */ are comments,
- in that document, words in **bold** are **TRUST** keywords, you can highlight them in your file editor with the command line (details in section 1.4):
  `> trust -config nedit|vim|emacs`
- braces "{ }" are elements that **TRUST** reads and interprets, so don't forget them and put space before and after them,
- elements between bracket "[ ]" are optional.

1.3.1 Data file example: base blocks

Here is the template of a basic sequential data file:

```
# Dimension 2D or 3D #
Dimension 2
```

```
# Problem definition #
Pb_hydraulique my_problem
```

```
# Domain definition #
Domaine my_domain
```

```
# Mesh #
Read_file my_mesh.geo ;
```

```
# For parallel calculation only! #
# For the first run: partitioning step #
# Partition my_domain
{
      Partition_tool partitioner_name { option1 option2 ... }
      Larg_joint 2
      zones_name DOM
      ...
}
End #
```

```
# For parallel calculation only! #
# For the second run: read of the sub-domains #
# Scatter DOM.Zones my_domain #
```

```
# Discretization on hexa or tetra mesh #
VDF my_discretization
```

```
# Time scheme explicit or implicit #
Scheme_euler_explicit my_scheme
Read my_scheme
{
      # Initial time #
      # Time step #
      # Output criteria #
      # Stop Criteria #
}
```

```
# Physical characteristics of medium #
Fluide_Incompressible my_medium
Read my_medium
{
      ...
}
```

```
# Gravity vector definition #
Uniform_field my_gravity
Read my_gravity 2 0 -9.81
```

```
# Association between the different objects #
Associate my_problem my_domain
Associate my_problem my_scheme
Associate my_problem my_medium
Associate my_medium my_gravity
```

```
# Discretization of the problem #
Discretize my_problem my_discretization
```

```
# New domains for post-treatment #
# By default each boundary condition of the domain is already extracted
  with names such as "my_dom"_boundaries_"my_BC" #
Domaine plane
extraire_surface
{
        domaine plane
        probleme my_probleme
        condition_elements (x>0.5)
        condition_faces (1)
}
```

```
# Problem description #
Read my_problem
{
```

```
        # hydraulic problem #
        Navier_Stokes_standard
        {
                # Choice of the pressure matrix solver #
                Solveur_Pression solver { ... }
                # Diffusion operator #
                Diffusion { ... }
                # Convection operator #
                Convection { ... }
                # Sources #
                Sources { ... }
                # Initial conditions #
                Initial_conditions { ... }
                # Boundary conditions #
                Boundary_conditions { ... }
        }
```

```
    # Post_processing description #
    /* To know domains that can be treated directly, search in .err
       output file: "Creating a surface domain named" */
    /* To know fields that can be treated directly, search in .err
       output file: "Reading of fields to be postprocessed" */
    Post_processing
    {
        # Definition of new fields #
        Definition_Champs { ... }

        # Probes #
        Probes { ... }

        # Fields #
        # format default value: lml #
        # select 'lata' for VisIt tool or 'MED' for Salomé #
        format lata
        fields dt_post 1. { ... }

        # Statistical fields #
        Statistiques dt_post 1. { ... }
    }
```

```
# Saving and restarting process #
[sauvegarde binaire datafile.sauv]
[resume_last_time binaire datafile.sauv]
```

```
# End of the problem description block #
}
```

```
# The problem is solved with #
Solve my_problem
```

```
# Not necessary keyword to finish #
End
```

1.3.2 Basic rules

There is no line concept in **TRUST**.

Data files uses blocks. They may be defined using the braces:

```
{
        a block
}
```

1.3.3 Objects notion

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 be applied to the block only (the associated object will be destroyed on exiting the block).

- *Type*: must be a type of object recognised by **TRUST**, correspond to the C++ classes. The list of recognised types is given in the file hierarchie.dump.
- *identificateur*: the identifier of the object type *Type* created, correspond to an instancy of the C++ class *Type*. **TRUST** exits in error if the identifier has already been used.

There are several object types. Physical objects, for example:

- 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,
- A **Scheme_euler_explicit** to indicate the time scheme type,
- A **Solveur_pression** to denote the pressure system solver type,
- A **Uniform_field** to define, for example, the gravity field.

1.3.4 Interpretor notion

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**. You can highlight them in your file editor with the command (details in section 1.4):
  `> trust -config nedit|vim|emacs`
- *argument*: an argument may comprise one or several object identifiers and/or one or several data blocks.

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**.

1.3.5 Example

A data set to write Ok on screen:

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

1.3.6 Important remarks

1. To insert comments in the data set, use # .. # (or /* ... */), the character # must always be enclosed by blanks.
2. The comma separates items in a list (a comma must be enclosed with spaces or a new line).
3. Interpretor keywords are recognised indiscriminately whether they are written in lower and/or upper case.
4. **On the contrary, object names (identifiers) are recognised differently if they are written in upper or lower case.**
5. In the following description, items (keywords or values) enclosed by [and] are optional.

1.4 Running a data file

To use **TRUST**, your shell must be "bash". So ensure you are in the right shell:

```
> echo $0
/bin/bash
```

To run your data file, you must initialize the TRUST environment using the following command:

```
> source $my_path_to_TRUST_installation/env_TRUST.sh
source $my_path_to_TRUST_installation/env/env_TRUST.sh
TRUST vX.Y.Z support : triou@cea.fr
Loading personal configuration /$path_to_my_home_directory/.perso_TRUST.env
```

1.4.1 Sequential calculation

You can run your sequential calculation:

```
> cd $my_test_directory
> trust [-evol] my_data_file
```

where "trust" command call the "trust" script. You can have the list of his options with:

```
> trust -help
```

or

```
> trust -h
```

Here is a panel of available options:

```
Usage: trust [option] datafile [nb_cpus] [1>file.out] [2>file.err]
Where option may be:
-help|-h                : List options.
-index                  : Access to the TRUST ressource index.
-doc                    : Access to the TRUST manual (Generic Guide).
-config nedit|vim|emacs : Configure nedit or vim or emacs with TRUST keywords.
-mesh                   : Visualize the mesh(es) contained in the data file.
-evol                   : Monitor the TRUST calculation (new IHM).
-wiz                    : Create a TRUST datafile from a MED file (new IHM).
-prm                    : Write a prm file.
-clean                  : Clean the current directory from all the generated
                          files by TRUST.
-copy                   : Copy the test case datafile from the TRUST database
```

```
                               under the present directory.
-check all|testcase|list : Check the non regression of all the test cases or a
                           single test case or a list of tests cases specified
                           in a file.
-check function|class|method::class : Check the non regression of a list of
                           tests cases covering a function, a class or a
                           class method.
-gdb                     : Run under gdb debugger.
-valgrind                : Run under valgrind.
-valgrind_strict         : Run under valgrind with no suppressions.
-create_sub_file         : Create a submission file only.
-prod                    : Create a submission file and submit the job on the
                           main production class with exclusive resource.
-help_trust              : Print options of
                           TRUST_EXECUTABLE [CASE[.data]] [options].
```

1.4.2 Parallel calculation

To run a parallel calculation, you must do two runs:

- the first one, to partition and create your 'n' sub-domains (two methods: "By hand" method below and "Assisted" method cf parts 7.3.1 & 7.3.2),
- the second one, to read your 'n' sub-domains and run the calculation on 'n' processors.

We will explain here how to do such work:

- **Partitioning: "By hand" method**
  You have to make two data files:
  - *BuildMeshes.data* and
  - *Calculation.data*.

  The *BuildMeshes.data* file only contains the same informations as the begining of the sequential data file and partitioning informations. This file will create the sub-domains (cf .Zones files).

BuildMeshes.data

Dimension 2
Domaine *my_domain*
 	
# BEGIN MESH # 	
Read_file *my_mesh.geo* ;
# END MESH # 	
 	
# BEGIN PARTITION # 	
Partition *my_domain*
{
    Partition_tool *partitioner_name* { *option1 option2* ... }
    Larg_joint *2*
    zones_name *DOM*
    ...
}
End
# END PARTITION # 	

Run the *BuildMeshes.data* with **TRUST**:

```
> trust BuildMeshes
```

You may have obtained files named *DOM_000n*.**Zones** which contains the 'n' sub-domains.

- **Read the sub-domains**
  To see your sub-domains, you can run:

  ```
  > trust -mesh Calculation
  ```

  In the *Calculation.data* file contains the domain definition, the block which will read the sub-domains and the problem definition (as in sequential calculation).

  Calculation.data

  ```
  Dimension 2

  Domaine my_domain

  # BEGIN SCATTER #
  Scatter DOM.Zones my_domain
  # END SCATTER #

  VDF my_discretization

  Scheme_euler_explicit my_scheme
  Read my_scheme { ... }

  Fluide_Incompressible my_medium
  Read my_medium { ... }

  Uniform_field my_gravity
  Read my_gravity 2 0 -9.81

  Associate my_problem my_domain
  Associate my_problem my_scheme
  Associate my_problem my_medium
  Associate my_medium my_gravity

  Discretize my_problem my_discretization

  Read my_problem { ... }

  Solve my_problem

  End
  ```

  Run the *Calculation.data* file with **TRUST**:

  ```
  > trust Calculation procs_number
  ```

  This will read your *DOM_000n*.**Zones** files. You can see the documentation of the **"scatter"** keyword in this part of the Project Reference Manual.

For more information, you can go to see this exercise in the **TRUST** tutorial.

1.5 Visualization

To learn how to use the "**-evol**" option, you can go to see the first exercise of the **TRUST** tutorial: Obstacle.

Chapter 2

Important references

For details and practices, see:

- the **TRUST** tutorial,
- the **TRUST** & **TrioCFD** user slides,
- **TRUST** & **TrioCFD** development slides,
- the Project Reference Manual.

Other references:

- Methodology for incompressible single phase flow (Models_Equations_TRUST.pdf),
- Trio_U code validation data base & best practice guidelines (Best_Practice_TRUST.pdf),
- Organisation of TrioCFD validation data base (HowTo_Validation.pdf),
- To access **TRUST** test case list:

```
> trust -index
```

For further information, you can see the **TRUST** web site: http://www-trio-u.cea.fr.

Remember that you can attend to:

- a user training session (2 days),
- a developper training session (2 days) or
- a dedicated session (one day)

provided by the support team. To request session, send an email to triou@cea.fr.

Chapter 3

Data setting

We will now explain how to fill a data file. First you must specify some basic informations like the dimension of your domain, his name, the problem type... To define the problem dimension, we use the following keyword:

Dimension *2* or **Dimension** *3*

3.1 Problems

You have to define the problem type that you wish to solve.

Pb_ ***type*** *my_problem*

Here are some of the available problem types:

- for incompressible flow: **Pb_**[**Thermo**]**hydraulique**[**_Concentration**][**_Turbulent**],
- for quasi-compressible flow: **Pb_Thermohydraulique**[**_Turbulent**]**_QC**,
- for solid: **Pb_Conduction**,
- you can find all problem types in the Reference Manual.

where:

- **hydraulique**: means that we will solve Navier Stokes equations without energy equation,
- **Thermo**: means that we will solve Navier Stokes equations with energy equation,
- **Concentration**: that we will solve multiple constituent transportation equations,
- **Turbulent**: that we will simulate a turbulent flow and specify a turbulent model (RANS or LES),
- **Conduction**: resolution of the heat equation,
- **QC**: Navier Stokes equations with energy equation for quasi-compressible fluid under low Mach numbers,

3.2 Domain definition

To define the domain, you must name it. This is done thanks to the following block:

Domaine *my_domain*

Then you must add your mesh to your simulation.

3.3 Mesh

Notice the presence of the tags:

```
# BEGIN MESH #
...
# END MESH #
```

in the data file of section 1.3.1. This is necessary for parallel calculation (see section 7).

3.3.1 Allowed meshes

TRUST allows:

- quadrangular or triangular undeformed meshing for 2D cases (Figure 3.1),

Figure 3.1: 2D allowed elements

- hexahedral or tetrahedral undeformed meshing for 3D cases (Figure 3.2).

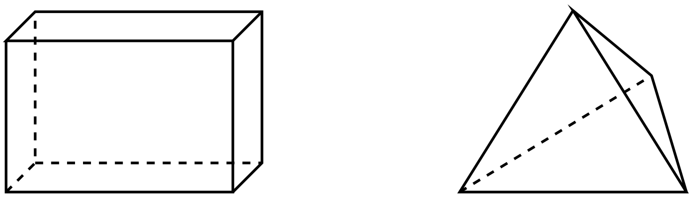


Figure 3.2: 3D allowed elements

Be carefull non standard and hybrid meshing are not supported! (cf Figure 3.3)

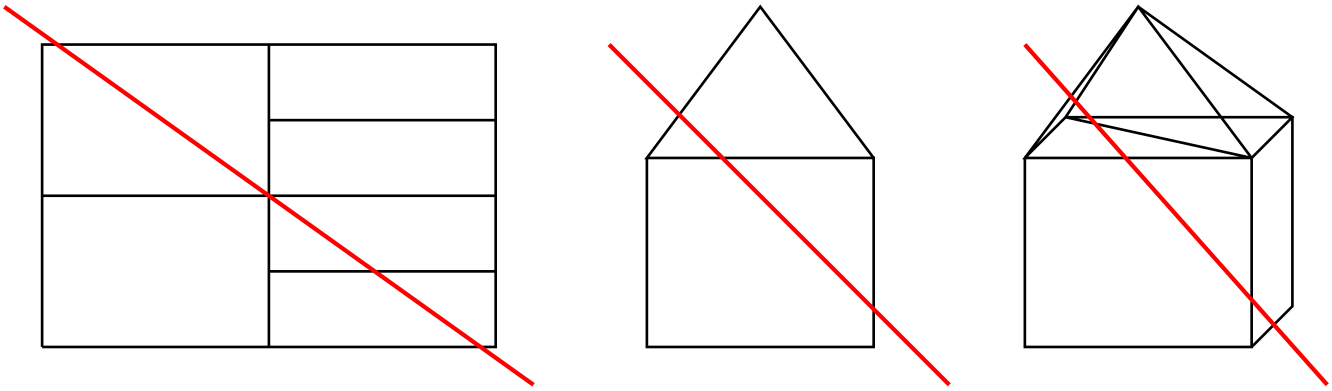


Figure 3.3: Prohibited meshes

3.3.2 Import a mesh file

If your mesh was generated with an external tool like Salomé (open source software), ICEM (commercial software), Gmsh (open source software, included in **TRUST** package) or Cast3M (CEA software), then you must use one of the following keyword into your data file:

- **Read_MED** for a MED file from Salomé, Gmsh,... ,
- **Read_File** for a binary mesh file from ICEM,
- for another format, see the Project Reference Manual.

If you want to learn how to make a mesh with Salomé or Gmsh and read it with **TRUST**, you can look at the exercises of the **TRUST** tutorial: here for Salomé and here for Gmsh.

3.3.3 Quickly create a mesh

Here is an example of a simple geometry (of non complex channel type) using the internal tool of **TRUST**:

```
Mailler my_domain
{
      /* Define the domain with one cavity */
      /* cavity 1m*2m with 5*22 cells */
      Pave box
      {
            Origine 0. 0.
            Longueurs 1 2
            /* Cartesian grid */
            Nombre_de_Noeuds 6 23
            /* Uniform mesh */
            Facteurs 1. 1.
      }
      {
            /* Definition and names of boundary conditions */
            bord Inlet    X = 0. 0. <= Y <= 2.
            bord Outlet   X = 1. 0. <= Y <= 2.
            bord Upper    Y = 2. 0. <= X <= 1.
            bord Lower    Y = 0. 0. <= X <= 1.
      }
}
```

To use this mesh in your data file, you just have to add the previous block in your data file or save it in a file named for example "*my_mesh.geo*" and add the line:

```
Read_file my_mesh.geo ;
```

Do not forget the semi-colonn at the end of the line!

3.3.4 Transform mesh within the data file

You can also make transformations on your mesh after the **"Mailler"** or **"Read_*"** command, using the following keywords:

- **Trianguler** to triangulate your 2D cells and create an unstructured mesh.

- **Tetraedriser** to tetrahedralise 3D cells and create an unstructured mesh.
- **Raffiner_anisotrope/Raffiner_isotrope** to triangulate/tetrahedralise elements of an untructured mesh.
- **ExtrudeBord** 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.
- **RegroupeBord** to build a new boundary with several boundaries of the domain.
- **Transformer** to transform the coordinates of the geometry.
- for other commands, see the section interprete of the Project Reference Manual.

Note that theses mesh modifications work on all mesh types (i.e. also for ***.geo** or ***.bin** or ***.med** files).

3.3.5 Test your mesh

The keyword **Discretiser_domaine** is useful to discretize the domain (faces will be created) without defining a problem. Indeed, you can create a minimal data file, post-process your mesh in lata format (for example) and visualize it with VisIt.

Note that you must name all the boundaries!

Here is an example of this kind of data file:

my_data_file.data

```
dimension 3
Domaine my_domain
Mailler my_domain
{
        Pave box
        {
              Origine 0. 0. 0.
              Longueurs 1 2 1
              Nombre_de_Noeuds 6 23 6
              Facteurs 1. 1. 1.
        }
        {
              bord Inlet    X = 0. 0. <= Y <= 2. 0. <= Z <= 1.
              bord Outlet   X = 1. 0. <= Y <= 2. 0. <= Z <= 1.
              bord Upper    Y = 2. 0. <= X <= 1. 0. <= Z <= 1.
              bord Lower    Y = 0. 0. <= X <= 1. 0. <= Z <= 1.
              bord Front    Z = 0. 0. <= X <= 1. 0. <= Y <= 2.
              bord Back     Z = 1. 0. <= X <= 1. 0. <= Y <= 2.
        }
}
discretiser_domaine my_domain
postraiter_domaine { domaine my_domain format lata }
End
```

To use it, launch in a bash terminal:

```
# if not already done
> source $my_path_to_TRUST_installation/env_TRUST.sh
# then
```

```
> trust my_data_file
> visit -o my_data_file.lata &
```

To see how to use VisIt, look at the first **TRUST** tutorial exercise: Obstacle.

If you want to learn how to make a mesh with Salomé or Gmsh and read it with **TRUST**, you can look at the exercises of the **TRUST** tutorial: here for Salomé and here for Gmsh.

3.4 Discretization

You have to specify the discretization type which can be **VDF**, **EF** or **VEFPreP1B**.

In **VDF** discretization, the locations of the unknown are drawn in the Figure 3.4.

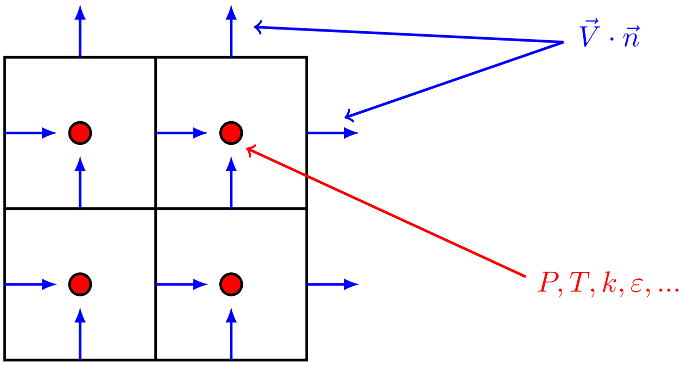


Figure 3.4: VDF unknown localisations

For **VEFPreP1B**, the locations of the unknown are drawn in the Figure 3.5.

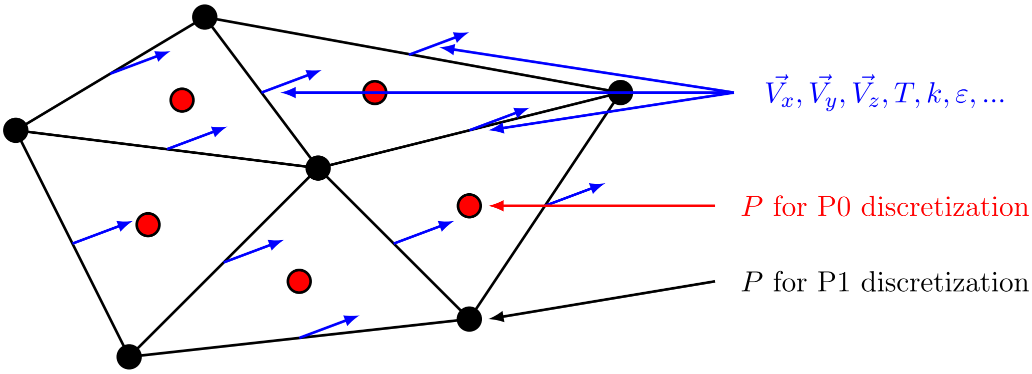


Figure 3.5: VEF unknown localisations in 2D

In 3D for the pressure, we can also use the P0+P1+Pa discretization for flow with a strong source term and a low velocity field. In this case P0+P1 pressure gradient has trouble to match the source term so we use P0+P1+Pa discretization (cf Figure 3.6).
To specify the wanted discretization, you have to add the following block to your data file:

Discretization_type *my_discretization* **[Read** *my_discretization { ... }*]

You can add parameters to your discretization with the optional keyword **Read** (see **VEFPreP1B discretization**).

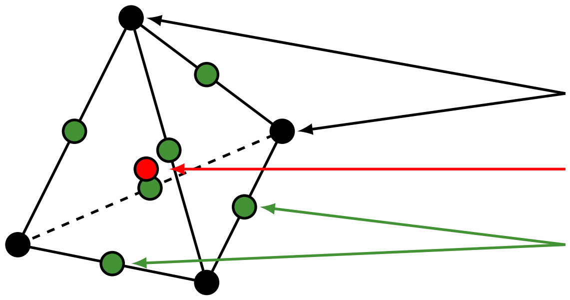


Figure 3.6: VEF pressure localisation in 3D

On the TRUST website, you can find information about:

- **VDF** discretization in the PhD thesis of A. Chatelain,
- **VEFPreP1B** discretization (Crouzet-Raviart elements) in the PhD thesis of T. Fortin and PhD thesis of S. Heib.

3.5 Time schemes

Now you can choose your time scheme to solve your problem. For this you must specify the time scheme type wanted and give it a name. then you have to specify its parameters by filling the associated **"Read"** block.

```
Scheme_type my_time_scheme
Read my_time_scheme { ... }
```

3.5.1 Some available time schemes

Here are some available types of explicit schemes:

- **Scheme_Euler_explicit**,
- **Schema_Adams_Bashforth_order_2**,
- **Runge_Kutta_ordre_3**,

And also some available types of implicit schemes:

- **Scheme_Euler_implicit**,
- **Schema_Adams_Moulton_order_3**.

For other scheme, see this section of the Reference Manual.

Note that you can use semi-implicit schemes activating the **diffusion_implicite** keyword in your explicit time scheme.

3.5.2 Calculation stopping condition

You must specify at least one stopping condition for you simulation. It can be:

- the final time: **tmax**
- the maximal allowed cpu time: **tcpumax**
- the number of time step: **nb_pas_dt_max**

- the convergency treshold: **seuil_statio**

Note that if the time step reaches the minimal time step **dt_min**, **TRUST** will stop the calculation.

If you want to stop properly your running calculation (i.e. with all saves), you may use the *my_data_file*.stop file (cf section 5.2). When the simulation is running, you can see the "**0**" value in that file.

To stop it, put a "**1**" instead of the "**0**", save the file and at the next iteration the calculation will stop properly.

When you don't change any thing in that file, at the end of the calculation, you can see that it is writen "**Finished correctly**".

3.6 Medium/Type of fluide

To specify the medium or fluid, you must add the following block.

```
Fluid_type my_medium
Read my_medium { ... }
```

Fluid_type can be one of the following:

- **Fluide_incompressible**
- **Fluide_quasi_compressible**
- **Solide**
- for other types and more information see Project Reference Manual.

If you want to use more than one medium, you can add an other blocks for each medium or fluid.

3.7 Add gravity

If needed, you can add a gravity term to your simulation. This is done by adding a uniform field, no matter his name. For example in 2D:

```
# Gavity vector definition #
Uniform_field my_gravity
Read my_gravity 2 0 -9.81
```

3.8 Objects association and discretization

3.8.1 Association

Until now, we have created some objects, now we must associate them together. For this, we must use the **Associate** interpretor:

```
# Association between the different objects #
Associate my_problem my_domain
Associate my_problem my_time_scheme
Associate my_problem my_medium
[Associate my_medium my_gravity]
```

3.8.2 Discretization

Then you must discretize your domain using the **Discretize** interpretor:

Discretize *my_problem my_discretization*

The problem *my_problem* is discretized according to the *my_discretization* 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.

Note that when the discretization step succeeds, the mesh is validated by the code.

At this level of your data file, you can visualize your mesh with the "**-mesh**" option of the trust script, it will directly open your mesh with VisIt.

```
# if not already done
> source $my_path_to_TRUST_installation/env_TRUST.sh
# then
> trust -mesh my_data_file
```

It will only run the mesh and stop, the problem will not be solved.

Chapter 4

Problem definition

4.1 Set of equations

In function of your choice of problem type, you will have a different set of equations.

4.1.1 Incompressible problems

TRUST solves Navier-Stokes equations with/without the heat equation for an incompressible fluid:

$$\begin{cases} \nabla \cdot \vec{u} = 0 \\ \dfrac{\partial \vec{u}}{\partial t} + \nabla \cdot (\vec{u} \otimes \vec{u}) = \nabla \cdot (\nu \nabla \vec{u}) - \nabla P^* \\ \dfrac{\partial T}{\partial t} + \vec{u} \nabla T = \nabla \cdot (\alpha \nabla T) + \dfrac{Q}{\rho C_p} \end{cases}$$

where: $P^* = \dfrac{P}{\rho} + gz$, Q is the heat source term, and:

- ρ: density,
- μ: dynamic viscosity,
- $\nu = \dfrac{\mu}{\rho}$: cinematic viscosity,
- $\vec{g} = gz$: gravity vector in cartesian coordinates,
- $\alpha = \dfrac{\lambda}{\rho C_p}$: thermal diffusivity.
- C_p: specific heat capacity at constant pressure,
- λ: thermal conductivity,

Note that red terms are convective terms and blue terms are diffusive terms.

```
Pb_Thermohydraulique_Concentration_Turbulent   my_problem
...
Read my_problem
{
      # Navier Stokes equations with/without turbulent model #
                              _Turbulent
      Navier_Stokes_Standard
      {
            Solveur_Pression my_solver { ... }
            Diffusion { ... }
            Convection { ... }
            Initial_conditions { ... }
            Boundary_conditions { ... }
            Modele_turbulence modele { ... }
            Sources { ... }
            ...
      }
      # Energy equation with/without turbulent model #
      Convection_Diffusion_Temperature_Turbulent
      {
            Diffusion { ... }
            Convection { ... }
            Initial_conditions { ... }
            Boundary_conditions { ... }
            Modele_turbulence Prandtl { ... }
            Sources { ... }
            ...
      }
      # Constituent transportation equations with/without turbulent model #
      Convection_Diffusion_Concentration_Turbulent
      {
            Diffusion { ... }
            Convection { ... }
            Initial_conditions { ... }
            Boundary_conditions { ... }
            Modele_turbulence Schmidt { ... }
            Sources { ... }
            ...
      }
}
```

For documentation, see:

Thermo	hydraulique	Concentration	Turbulent	Reference Manual
	Pb_hydraulique			doc
	Pb_hydraulique	_Concentration		doc
	Pb_hydraulique		_Turbulent	doc
	Pb_hydraulique	_Concentration	_Turbulent	doc
Pb_Thermo	hydraulique			doc
Pb_Thermo	hydraulique	_Concentration		doc
Pb_Thermo	hydraulique		_Turbulent	doc
Pb_Thermo	hydraulique	_Concentration	_Turbulent	doc

4.1.2 Quasi-compressible problem

TRUST solves Navier-Stokes equations with/without heat equation for quasi-compressible fluid:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \\ \dfrac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot (\rho \vec{u} \vec{u}) = \nabla \cdot (\mu \nabla \vec{u}) - \nabla P - \rho \vec{g} \\ \rho C_p \left(\dfrac{\partial T}{\partial t} + \vec{u} \nabla T \right) = \nabla \cdot (\lambda \nabla T) + \dfrac{dP_0}{dt} + Q \end{cases}$$

where: $P_0 = \rho R T$, Q is a heat source term, and:

- ρ: density,
- μ: dynamic viscosity,
- $\vec{g} = gz$: gravity vector in cartesian coordinates,
- C_p: specific heat capacity at constant pressure,
- λ: thermal conductivity.

Note that red terms are convective terms and blue terms are diffusive terms.

```
Pb_Thermohydraulique_Turbulent_QC my_problem
...
Read my_problem
{
	# Navier Stokes equations for quasi-compressible fluid under #
	# low Mach numbers with/without turbulent model #
	Navier_Stokes_Turbulent_QC
	{
		Solveur_Pression my_solver { ... }
		Diffusion { ... }
		Convection { ... }
		Initial_conditions { ... }
		Boundary_conditions { ... }
		Modele_turbulence modele { ... }
		Sources { ... }
		...
	}
	# Energy equation for quasi-compressible fluid under low Mach #
	# numbers with/without turbulent model #
	Convection_Diffusion_Chaleur_Turbulent
	{
		Diffusion { ... }
		Convection { ... }
		Initial_conditions { ... }
		Boundary_conditions { ... }
		Modele_turbulence Prandtl { ... }
		Sources { ... }
		...
	}
}
```

For more information on QC problem, go there and for turbulent QC problem, go there.

4.1.3 Conduction problem

For this kind of problem, **TRUST** solves the heat equation:

$$\rho C_p \frac{\partial T}{\partial t} = \nabla \cdot (\lambda \nabla T) + Q$$

where:

- ρ: density,
- C_p: specific heat capacity at constant pressure,
- λ: thermal conductivity,
- Q is a heat source term.

Note that red terms are convective terms and blue terms are diffusive terms.

In your data file, you will have:

```
Pb_Conduction my_problem
...
Read my_problem
{
      # Resolution of the heat equation #
      Conduction
      {
            Diffusion { ... }
            Convection { ... }
            Initial_conditions { ... }
            Boundary_conditions { ... }
            Sources { ... }
            ...
      }
}
```

For more information, see the Project Reference Manual.

4.1.4 Coupled problems

With **TRUST**, we can couple problems. We will explain here the method for two problems but you can couple as many problems as you want.

To couple two problems, we define two problems *my_problem_1* and *my_problem_2* each one associated to a separate domain *my_domain_1* and *my_domain_2*, and to a separate medium *my_medium_1* and *my_medium_2* (associated or not to the gravity).

```
Dimension 2

Pb_ThermoHydraulique_Turbulent my_problem_1
Pb_ThermoHydraulique_Turbulent my_problem_2

Domaine my_domain_1
Read_file my_mesh_1.geo ;

Domaine my_domain_2
Read_file my_mesh_2.geo ;

Fluide_Incompressible my_medium_1
Read my_medium_1 { ... }

Fluide_Incompressible my_medium_2
Read my_medium_2 { ... }

Associate my_problem_1 my_domain_1
Associate my_problem_1 my_medium_1

Associate my_problem_2 my_domain_2
Associate my_problem_2 my_medium_2
```

Then we define a coupled problem associated to a single time scheme like for example:

```
Probleme_Couple my_coupled_problem

VEFPreP1B my_discretization

Scheme_euler_explicit my_scheme
Read my_scheme { ... }

Associate my_coupled_problem my_problem_1
Associate my_coupled_problem my_problem_2
Associate my_coupled_problem my_scheme
```

Then we discretize and solve everything:

```
Discretize my_coupled_problem my_discretization
Read my_problem_1 { ... }
Read my_problem_2 { ... }
Solve my_coupled_problem
End
```

You can see the documentation of this kind of problem in the Project Reference Manual.

4.1.5 Other problems

TRUST can also solve the following types of problems:

- Resolution of NAVIER STOKES/energy/multiple constituent transportation equations, with the additional passive scalar equations, and
- describe the chemical reactions.

4.2 Pressure solvers

Then you may indicate the choice of pressure solver (cf Project Reference Manual) using the following syntax:

Solveur_pression *my_solver* `{ ... }`

The *my_solver* may be:

- **GCP**,
- **Petsc** *Petsc_solver_name*,
- **Cholesky**,
- **Gmres**,
- **Gen**,
- **Optimal**.

Reminder: in CFD, a separate solver is used to solve the pressure. For more details, you can have a look at the section "Time and space schemes" of the the **TRUST** & **TrioCFD** user slides.

4.3 Convection

There is no default convective scheme so you must choose one convection scheme:

convection `{` *convective_scheme* `}`

You can use the following convective scheme, following the recommendations of the user training session (cf section "Time and space schemes" of the the **TRUST** & **TrioCFD** user slides and the section "Recommendations for schemes") following your discretization type:

- **Amont**
- **Muscl**
- **EF_stab**
- for more, see the Project Reference Manual.

Note that there is no default convective scheme and if you don't want convection in your problem, you may use:

convection { negligeable }

4.4 Diffusion

For the diffusive scheme, it is the same syntax:

diffusion `{ [`*diffusive_scheme*`] }`

You can choose your scheme with the help of the Project Reference Manual.

Note that if you don't specify any diffusive scheme, the code automatically uses the standard diffusive scheme of order 2. If you don't want diffusion in your problem, you may use:

diffusion { negligeable }

4.5 Initial conditions

For each equation, you **must** set initial conditions:

initial_conditions { ... }

To see the syntax of each available initial condition: cf Project Reference Manual. Here are the most used initial conditions:

- **Velocity** field_type *bloc_lecture_champ*
- **Temperature** field_type *bloc_lecture_champ*
- **K_eps** field_type *bloc_lecture_champ*

We list here some "field_type":

- **Uniform_Field**: for a uniform field,
- **Champ_Fonc_Med**: to read a data field in a MED-format file .med at a specified time,
- **Champ_Fonc_txyz**: for a field which depends on time and space,
- **Champ_Fonc_Fonction_txyz**: for a field which is a function of another field and time and/or space coordinates,
- **Champ_Fonc_Reprise**: to read a data field in a saved file (.xyz or .sauv) at a specified time.
- refer to the Project Reference Manual.

4.6 Boundary conditions

Then you may specify your boundary conditions like:

boundary_conditions { ... }

It is important to specify here that **TRUST will not accept any boundary conditions by default.**

You can find help for boundary conditions in the Project Reference Manual. Here is a list of the most used boundary conditions:

- **Bord Frontiere_ouverte_vitesse_imposee** boundary_field_type *bloc_lecture_champ*
- **Bord Frontiere_ouverte_pression_imposee** boundary_field_type *bloc_lecture_champ*
- **Bord Paroi_fixe**
- **Bord Symetrie**
- **Bord Periodique**
- **Bord Frontiere_ouverte_temperature_imposee** boundary_field_type *bloc_lecture_champ*
- **Bord Frontiere_ouverte T_ext** boundary_field_type *bloc_lecture_champ*
- **Bord Paroi_adiabatique**
- **Bord Paroi_flux_impose** boundary_field_type *bloc_lecture_champ*
- for more, see the Project Reference Manual.

To choose your "boundary_field_type" parameters, refer to the Project Reference Manual.

4.7 Turbulent model

User can add a turbulent model to his simulation using the keyword:

```
Modele_turbulence my_model { ... }
```

where *my_model* can be:

- **Longueur_Melange**: RANS model based on mixing length modelling,
- **Sous_maille**: LES model which uses a structure sub-grid function model,
- **K_epsilon**: for RANS turbulence model (k-ε),
- for more, see the Project Reference Manual.

4.8 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 , ...}
```

- **Perte_Charge_Reguliere** type_perte_charge bloc_definition_pertes_charges
- **Perte_Charge_Singuliere** **KX** | **KY** | **KZ** coefficient_value { ... }
- **Canal_perio** { ... }
- **Boussinesq_temperature** { ... }
- **Boussinesq_concentration** { ... }
- **Puissance_thermique** field_type bloc_lecture_champ
- documentation for hydraulic source terms and for scalar source terms.

4.9 Post-process

Before post-processing fields, during a run, **TRUST** creates several files which contain informations about the calculation, the convergence, flux, balances... (see part 6.1 for more informations).

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 postprocessings with **Post_processings** keyword (named with Post_name1, Post_name2, etc...), in order to print results into several formats or with different time periods, or into different results files:

```
Post_processings
{
        Post_name1 { Postraitement_definition }
        Post_name2 { Postraitement_definition }
        ...
}
```

4.9.1 Field names

- **Existing & predefined fields**

  You can post-process predefined fields and already existing fields. Here is a list of post-processable fields, but it is not the only ones.

Physical values	Keyword for field_name	Unit
Speed	**Vitesse** or **Velocity**	$m.s^{-1}$
Kinetic energy per elements ($0.5\rho\|\|u_i\|\|^2$)	**Energie_cinetique_elem**	$kg.m^{-1}.s^{-2}$
Total kinetic energy $\left(\frac{\sum_{i=1}^{nb\_elem} 0.5\rho\|\|u_i\|\|^2 vol_i}{\sum_{i=1}^{nb\_elem} vol_i}\right)$	**Energie_cinetique_totale**	$kg.m^{-1}.s^{-2}$
Vorticity	**Vorticite**	s^{-1}
Pressure in incompressible flow ($P/\rho + gz$) For Front Tracking probleme ($P + \rho gz$)	**Pression** [1]	$Pa.m^3.kg^{-1}$ or Pa
Pressure in incompressible flow (P+ρgz)	**Pression_pa** or **Pressure**	Pa
Pressure in compressible flow	**Pression**	Pa
Hydrostatic pressure (ρgz)	**Pression_hydrostatique**	Pa
Totale pressure (when quasi compressible model is used)=Pth+P	**Pression_tot**	Pa
Pressure gradient ($\nabla(P/\rho + gz)$)	**Gradient_pression**	$m.s^{-2}$
Temperature	**Temperature**	^{o}C or K
Phase temperature of a two phases flow	**Temperature_EquationName**	^{o}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** [2]	$W.m^{-2}.K^{-1}$
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**	$m^3.s^{-1}$
Turbulent quantities K and Epsilon	**K_Eps**	($m^2.s^{-2}$,$m^3.s^{-1}$)

... continued on next page ...

[1] 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.

[2] 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.

Physical values	Keyword for field_name	Unit
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 $Y^+ = yU/\nu$ (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** [3]	m
Volumic thermal power	**Puissance_volumique**	$W.m^{-3}$
Local shear strain rate defined as $\sqrt{(2SijSij)}$	**Taux_cisaillement**	s^{-1}
Cell Courant number (VDF only)	**Courant_maille**	dimensionless
Cell Reynolds number (VDF only)	**Reynolds_maille**	dimensionless

Remark: Physical properties (conductivity, diffusivity,...) can also be interrogated.

The name of the fields and components available for post-processing is displayed in the error file after the following message: "Reading of fields to be postprocessed". Of course, this list depends of the problem being solved.

For more informations, you can see the Project Reference Manual.

- **Creating new fields**

  The **Definition_champs** keyword is used to create new or more complex fields for advanced postprocessing.

  Definition_champs { *field_name_post* ***field_type*** { ... } }

  field_name_post is the name of the new created field and ***field_type*** is one of the following possible type:

  - **refChamp**
  - **Reduction_0D** using for example the **min**, **max** or **somme** methods.
  - **Transformation**
  - for details and other keywords, see the Project Reference Manual.

  Note that you can combine several ***field_type*** keywords to create your field and then use your new fields to create other ones.

  Here is an example of new field named *max_temperature*:

[3] distance_paroi is a field which can be used only if the mixing length model (see 2.15.1.2) is used in the data file.

```
Read my_problem {
    ...
    Postraitement {
        Definition_champs {
            # Creation of a 0D field: maximal temperature of the domain #
            max_temperature Reduction_0D {
                methode max
                source refChamp { Pb_champ my_problem temperature }
            }
        }
        Probes {
            # Print max(temperature) into the datafile_TMAX.son file #
            tmax max_temperature periode 0.01 point 1 0. 0.
        }
        Champs dt_post 1.0 { ... }
    }
}
```

You can find other examples in the the **TRUST & TrioCFD** user slides in the section "Post processing description".

4.9.2 Post-processing blocks

There are three method to post-process in **TRUST**: using probes, fields or making statistics.

- **Probes**
  Probes refer to sensors that allow a value or several points of the domain to be monitored over time. The probes are a set of points defined:
  - one by one: **Points** keyword or
  - by a set of points evenly distributed over a straight segment: **Segment** keyword or
  - arranged according to a layout: **Plan** keyword or
  - arranged according to a parallelepiped: **Volume** keyword.

  Here is an example of 2D **Probes** block:

```
Probes {
    pressure_probe [loc] pressure Periode 0.5 Points 3 1. 0. 1. 1. 1. 2.
    velocity_probe [loc] velocity Periode 0.5 Segment 10 1. 0. 1. 4.
}
```

  where the use of *"loc"* option allow to specify the wanted localisation of the probes. The available values are **"grav"** for gravity center of the element, **"nodes"** for faces and **"som"** for vertices. There is not default location. If the point does not coincide with a calculation node, the value is extrapolated linearly according to neighbouring node values.

  For complete syntax, see the Project Reference Manual, also for all options.

- **Fields**
  This keyword allows to post-process fields on the whole domain, specifying the name of the backup file, its format, the post-process time step and the name (and localisation) of the post-processed fields.

  Here is an example of **Fields** block:

```
Fichier results
Format lata
Fields dt_post 1. {
        velocity [faces] [som] [elem]
        pressure [elem] [som]
        temperature [elem] [som]
}
```

where **"faces"** , **"elem"** and **"som"** are keywords allowed to specify the localisation of the field.

Note that when you don't specify the localisation of the field, the default value is **"som"** for values at the vertices. So fields are post-processed at the vertices of the mesh.

To visualize your post-processed fields, you can use open source softwares like: VisIt (included in **TRUST** package) for lata files, or for med files: Salomé or Paraview.

You can see the complete syntax and all options in the Project Reference Manual.

- **Statistics**
  Using this keyword, you will compute statistics on your unknows. You must specify the begining and ending time for the statistics, the post-process time step, the statistic method, the name (and localisation) of your post-processed field.
  Here is an example of **Statistiques** block:

```
Statistiques dt_post 0.1 {
        t_deb 1. t_fin 5.
        moyenne velocity [faces] [elem] [som]
        ecart_type pressure [elem] [som]
        correlation pressure velocity [elem] [som]
}
```

This block will write at every **dt_post** the average of the velocity $\overline{V(t)}$:

$$\overline{V(t)} = \begin{cases} 0 & \text{, for } t \leq t\_deb \\ \frac{1}{t-t\_deb}\int_{t\_deb}^{t} V(t)dt & \text{, for } t\_deb < t \leq t\_fin \\ \frac{1}{t\_fin-t\_deb}\int_{t\_deb}^{t\_fin} V(t)dt & \text{, for } t > t\_fin \end{cases}$$

the standard deviation of the pressure $\langle P(t) \rangle$:

$$\langle P(t) \rangle = \begin{cases} 0 & \text{, for } t \leq t\_deb \\ \frac{1}{t-t\_deb}\sqrt{\int_{t\_deb}^{t} \left[P(t) - \overline{P(t)}\right]^2 dt} & \text{, for } t\_deb < t \leq t\_fin \\ \frac{1}{t\_fin-t\_deb}\sqrt{\int_{t\_deb}^{t\_fin} \left[P(t) - \overline{P(t)}\right]^2 dt} & \text{, for } t > t\_fin \end{cases}$$

and correlation between the pressure and the velocity $\langle P(t).V(t) \rangle$ like:

$$\langle P(t).V(t) \rangle = \begin{cases} 0 & \text{, for } t \leq t\_deb \\ \frac{1}{t-t\_deb}\int_{t\_deb}^{t} \left[P(t) - \overline{P(t)}\right] \cdot \left[V(t) - \overline{V(t)}\right] dt & \text{, for } t\_deb < t \leq t\_fin \\ \frac{1}{t\_fin-t\_deb}\int_{t\_deb}^{t\_fin} \left[P(t) - \overline{P(t)}\right] \cdot \left[V(t) - \overline{V(t)}\right] dt & \text{, for } t > t\_fin \end{cases}$$

Remark: 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**.

For the complete syntax, see the Project Reference Manual here, and for all options see the **TRUST** Reference Manual.

4.9.3 Post-process localisation

You can use localisation keywords to specify where you want to post-process your fields in order to avoid interpolations on your post-processed fields.

For the **VDF** discretization, you can see the Figure 3.4 and here is a table with a reminder of the computation location of the fields in the **VDF** discretization:

Names	Keyword	Localisation
Pressure	**pressure**	element gravity center
Velocity	**velocity**	faces center
Temperature	**temperature**	element gravity center
Density ρ	**masse_volumique**	element gravity center
Cinematic viscosity ν	**viscosite_cinematique**	element gravity center
Dynamic viscosity μ	**viscosite_dynamique**	element gravity center
K	**k**	element gravity center
eps	**eps**	element gravity center
y^+	**y_plus**	element gravity center
u^*	**u_star**	faces center
Turbulent viscosity	**viscosite_turbulente**	element gravity center

For the **VEFPreP1B** discretization, you can see the Figure 3.5 and 3.6. Here is a table with a reminder of the computation location of the fields in **VEFPreP1B** discretization:

Names	Keyword	Localisation
Pressure	**pressure**	element gravity center vertices
Velocity	**velocity**	faces center
Temperature	**temperature**	faces center
Density ρ	**masse_volumique**	element gravity center
Cinematic viscosity ν	**viscosite_cinematique**	element gravity center
Dynamic viscosity μ	**viscosite_dynamique**	element gravity center
K	**k**	faces center
eps	**eps**	faces center
y^+	**y_plus**	element gravity center
u^*	**u_star**	faces center
Turbulent viscosity	**viscosite_turbulente**	element gravity center

Be careful, if you are in P0+P1 discretization (default option) and you post-process the pressure field at the element (or at the vertices), you will have an **interpolation** because the field is computed at the element **and** at the vertices.

Chapter 5

End of the data file

5.1 Solve

Now that you have finished to specify all your computation parameters, you may add the **"Solve"** keyword at the end of your data file, in order to solve your problem. You may also add the **"End"** keyword to specify the end of your data file.

Solve *my_problem*   **[End]**

For more details, see the Project Reference Manual.

You can see methods to run your data file in section 1.4.

5.2 Stop a running calculation

Your calculation will automatically stops if it has reached:

- the end of the calculation time,
- the maximal allowed cpu time,
- the maximal number of iterations or
- the threshold of convergence.

You may use the *my_ data_file*.**stop** file, if you want to stop properly your running calculation (i.e. with all saves).

When the simulation is running, you can see the "**0**" value in that file. To stop it, put a "**1**" instead of the "**0**" and at the next iteration the calculation will stop properly. When you don't change any thing to that file, at the end of the calculation, you can see that it is writen "**Finished correctly**".

5.3 Save

TRUST makes automatic backups during the calculation. The unknowns (velocity, temperature,...) are saved in:

- one **.xyz** file, happening:
  - at the end of the calculation,
  - but, user may disable it with the specific keyword "**EcritureLectureSpecial 0**" added just before the "**Solve**" keyword.

- one (or several in case of parallel calculation) **.sauv** files, happening:
  - at the start of the calculation,
  - at the end of the calculation,
  - each 23 hours of CPU, to change it, uses **"periode_sauvegarde_securite_en_heure"** keyword (default value 23 hours),
  - user may also specify a time physical period with **"dt_sauv"** keyword,
  - periodically using **"tcpumax"** keyword for which calculation stops after the specified time (default value 10^{30}), use it for calculation on CCRT/TGCC and CINES clusters for example.

By default, the name for the **.sauv** files is **"filename_problemname.sauv"** for sequential calculation, **"filename_problemname_000n.sauv"** for parallel calculation (one per process). The format of theses files is binary and the files are completed during successive saves.

You can change the behaviour using the following keywords just before the **solve** instruction:

sauvegarde	**binaire|xyz**	*filename*.sauv | *filename*.xyz

with **"xyz"**: the **.xyz** file is written instead of the **.sauv** files.

Note that, you can use **"sauvegarde_simple"** instead of **"sauvegarde"** where the .sauv or .xyz file is deleted before saves, in order to keep disk space:

sauvegarde_simple	**binaire|xyz**	*filename*.sauv | *filename*.xyz

For more details, see the Project Reference Manual.

5.4 Restart

To restart your calculation, you may:

- change your initial time, the new initial time will be the real final calculation time of the previous calculation (cf .err file),
- change your final calculation time to the new wanted value and
- add the following block just before the **"Solve"** keyword:

reprise	**binaire|xyz**	*filename*.sauv | *filename*.xyz

You can restart your calculation:

- from .sauv file(s) (one file per process): you can only restart the calculation with the **same number of equations** on **the same number of processes**,
- or from a .xyz file: here you can restart your calculation by **changing the number of equations solved** and/or with a **different number of processes**.

Instead of **"reprise"** keyword, you can use **"resume_last_time"** where **tinit** is automatically set to the last time of saved files (but you may change **tmax**):

resume_last_time	**binaire|xyz**	*filename*.sauv | *filename*.xyz

For examples, see the **TRUST** tutorial and the Project Reference Manual.

Note that you can run a calculation with initial condition read into a save file (.xyz or .sauv) from a previous calculation using **Champ_Fonc_reprise** or read a into a MED file with **Champ_Fonc_MED**.

Chapter 6

Post-processing

6.1 Output files

After running, you will find different files in your directory. Here is a short explaination of what you will find in each type of file in function of his extension.

Even if you don't post-process anything, you will have output files which are listed here:

File	Contents
my_data_file.**dt_ev**	Time steps, facsec, equation residuals
my_data_file.**stop**	Stop file ('0', '1' or 'Finished correctly')
my_data_file.**log**	Journal logging
my_data_file.**TU**	CPU performances
*my_data_file*_**detail.TU**	Statistics of execution
my_data_file_problem_name.**sauv** or **.xyz** or *specified_name*.**sauv** or **.xyz**	Saving 2D/3D results for restart (binary files)

and the listing of boundary fluxes where:

- *my_data_file*_**Contrainte_visqueuse.out** correspond to the friction drag exerted by the fluid,
- *my_data_file*_**Convection_qdm.out** contains the momentum flow rate,
- *my_data_file*_**Debit.out** is the volumetric flow rate,
- *my_data_file*_**Force_pression.out** correspond to the pressure drag exerted by the fluid.

If you add post-processings in your data files, you will find:

File	Contents
my_data_file.**sons**	1D probes list
my_data_file_probe_name.**son**	1D results with probes
my_data_file_probe_name.**plan**	3D results with probes
my_data_file.**lml** *(default format)* *my_data_file*.**lata** *(with all *.lata.* files)* *my_data_file*.**med** or *specified_name*.**lml** or **.lata** or **.med**	2D/3D results

The sceen outputs are automatically redirected in *my_data_file*.**out** and *my_data_file*.**err** files if you run a parallel calculation or if you use the "**-evol**" option of the "trust" script.
Else you can redirect them with the following command:

```
# if not already done
> source $my_path_to_TRUST_installation/env_TRUST.sh
# then
> trust my_data_file 1>file_out.out 2>file_err.err
```

In the .out file, you will find the listing of physical infos with mass balance and in the .err file, the listing of warnings, errors and domain infos.

6.2 Tools

To open your 3D results in **lata** format, you can use VisIt which is an open source software included in **TRUST** package. For that you may "source" **TRUST** environment and launch VisIt:

```
# if not already done
> source $my_path_to_TRUST_installation/env_TRUST.sh
# then
> visit -o my_data_file.lata &
```

To learn how to use it, you can try the **TRUST** tutorial exercise Obstacle.

To open your 3D results in **med** format, you can also use VisIt or Salomé or Paraview.

Here are some actions that you can perform when you simulation is finished:

- To visualize the positions of your probes in function of the 2D/3D mesh, you can open your .son files at the same time of the .lata file in VisIt.
- If you need more probes, you can create them with VisIt (if you have post-processed the good fields) or with MEDCoupling.
- You can use the option "**-evol**" of the trust script, like:

  ```
  trust -evol my_data_file
  ```

  and acces to the probes or open VisIt for 2D/3D visualizations via this tool.

Chapter 7

Parallel calculation

TRUST is a plateform which allow to make parallel calculation following some basic rules:

- **S**ingle **P**rogram, **M**ultiple **D**ata model: tasks are split up and run simultaneously on multiple processors with different input in order to obtain results faster,
- messages exchange by **M**essage **P**assing **I**nterface,
- from a PC to a massively parallel computer, with shared or distributed memory.

7.1 Basic notions

To make a parallel calculation, you have to partition your domain. Each sub-domain will be treated by one processor. In order to have good performances, ensure you that:

- sub-domains have almost the same number of cells,
- joint lengths (boundaries between sub-domains) are minimal,

7.2 Performances

You have to choose a number of processors which is in agreement with the number of cells. So, you can evaluate your speed-up (sequential time/parallel time which must be as close as possible of the number of processors) or efficiency (=1/SpeedUp) to choose the right number of processors.

From our experience, for good performance with **TRUST**, each processor has to treat between 20000 and 30000 cells.

7.3 Partitioning

To run a parallel calculation, you must:

- make some modifications on your *my_data_file.data* file,
- do two runs:
  - the first one, to partitioning and create your 'n' sub-domains (two methods will by presented),
  - the second one, to read your 'n' sub-domains and run the calculation on 'n' processors.

7.3.1 The different blocks

Different blocks appear in the data file.

- **Modifications on the mesh block**
  First you may add the tags "# BEGIN MESH #" and "# END MESH #" before and after your mesh block, for example:

```
# BEGIN MESH #
Read_file my_mesh.geo ;
[Trianguler_h my_domain]
# END MESH #
```

  You can refer to section 3.3 to have more informations.

- **Adding a partitioning block**
  You may now add the partitioning block which contains the cutting instruction, after your mesh block:

```
# BEGIN PARTITION
Partition my_domain
{
        Partition_tool partitioner_name { option1 option2 ... }
        Larg_joint 2
        zones_name DOM
        ...
}
End
END PARTITION #
```

  Where *partitioner_name* is the name of the chosen partitioner, one of **METIS**, **Tranche**, **Sous_Zones**, **Partition** or **Fichier_Decoupage** (cf section 7.3.3).
  The **"Larg_joint"** keyword allows to specify the overlapping width value. You can see the documentation of this part in the Project Reference Manual.

  Note the "End" before the last line. It will be useful for the cutting step.

  This block will made the partitioning of your domain into the specified number of sub-domains during the partitioning run.

- **Adding a block to read the sub-domains**
  At last, you will add a block which will be activated during the parallel calculation and will allow to read the sub-domains:

```
# BEGIN SCATTER
Scatter DOM.Zones my_domain
END SCATTER #
```

7.3.2 Partitionning: "Assisted" method

Here we will use the script `make_PAR.data` to make the partitioning step. We consider that you have correctly add the "#" in your *my_data_file.data* file with the partitioning block and cutting block.

Be careful with the hashtags "#", they are interpreted by the script!

To automatically perform the partitioning step and obtain the parallel data file, you have to run:

```
> make_PAR.data my_data_file [parts_number]
```

Note that here `parts_number` is the number of sub-domains created but it is also the number of processors which will be used.

This command creates:

- a *SEQ_my_data_file.data* file which is a backup file of *my_data_file.data* the sequential data file,
- a *DEC_my_data_file.data* file which is the first data file to be run to make the partitioning. It is immediately run by the command line **make_PAR.data** to create a partition, located in the *DOM_000n*.**Zones** files.
  Note that the code stops reading this file at the **"End"** keyword just before the "# END PARTITION #" block.
- a *PAR_my_data_file.data* file which is the data file for the parallel calculation. It reads the *DOM_000n*.**Zones** files through the instruction **"Scatter"**.
  Note that the meshing and cut of the mesh are commented here.

To see your sub-domains, you can run:

```
> trust -mesh PAR_my_data_file
```

For more information, you can go to see this exercise in the **TRUST** tutorial.

7.3.3 TRUST available partitioning tools

In **TRUST**, you can make partitioning with:

- the external partitionning library **"METIS"** (open source). It is a general algorithm that will generate a partition of the domain (cf Project Reference Manual),

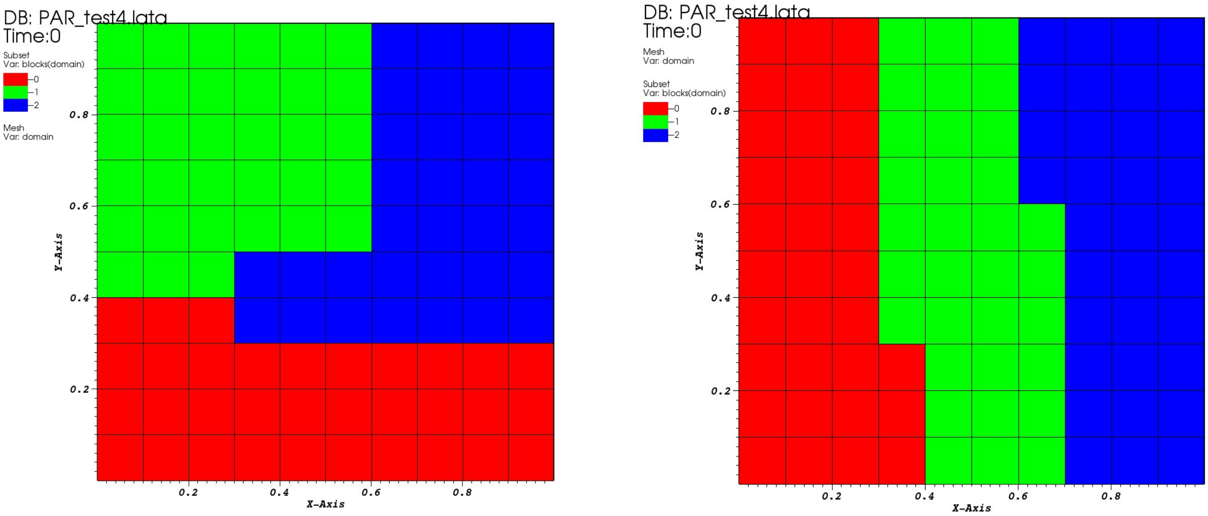
```
partition_tool Metis {
        nb_parts N
        [use_weights]
        [pmetis | kmetis]
        [nb_essais N]
}
```

- internal **TRUST** partitioning tool: **Tranche** which make parts by cutting the domain following x, y and/or z directions.

```
partition_tool Tranche {
        tranches nx ny [nz]
}
```

- other internal partitioning tools.

The Figure 7.1 is an example of what you can obtain by cutting a 1m x 1m square, divided in three parts using **METIS** and the same square divided in three slices following the x direction with **Tranche**.



METIS

Tranche

Figure 7.1: Partitioning tools

You can see the documentation of this part in the Project Reference Manual.

7.3.4 Overlapping width value

To make the partitioning, you will have to specify the overlapping width value. This value correspond to the thickness of the virtual ghost zone (data known by one processor though not owned by it) i.e. the number of vertices or elements on the remote sub-domain known by the local sub-domain (cf Figure 7.2).

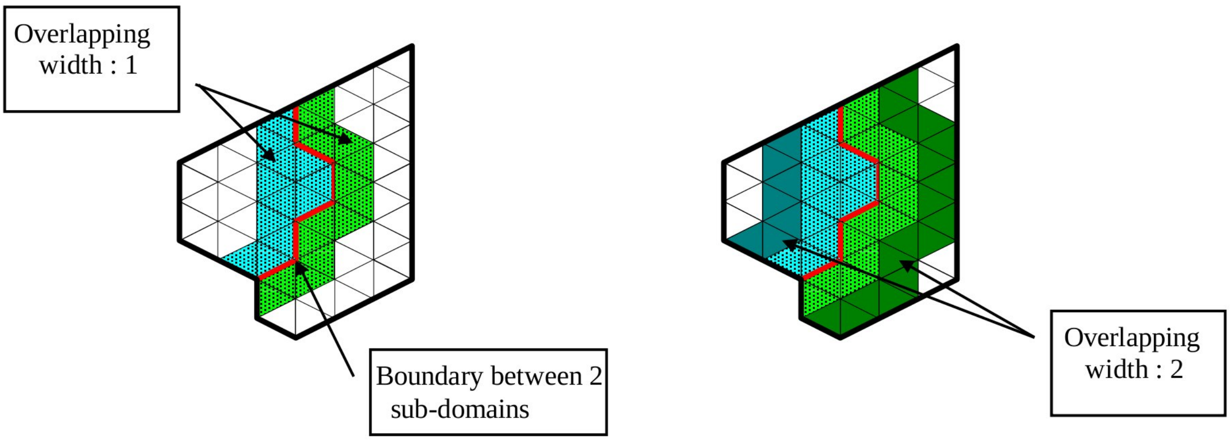


Figure 7.2: Overlapping width

This value depends on the space scheme orders:

- 1 if 1-2nd order,
- 2 if 3-4th order.

Note that in general, you will use "2"!

7.4 Running a parallel calculation

7.4.1 On a PC

To launch the calculation, you have to run the calculation by the usual command completed by the number of processors needed:

```
> trust my_parallel_data_file procs_number
```

and *procs_number* is the number of processors used. In fact it is the same as the number of sub-domains.

You can see the **TRUST** & **TrioCFD** user slides in the "Parallel calculation" section for more informations and those two exercise of the **TRUST** tutorial: exercise 1 and exercise 2.

7.4.2 On a cluster

You must submit your job in a queue system. For this, you must have a submission file. **TRUST** can create a submission file for you **on clusters on which the support team has done installations**. To create this file, run:

```
> trust -create_sub_file my_parallel_data_file
```

You obtain a file named **"sub_file"**, you can open it and verify/change values(for example the name of the job, the name of the exe, ...).

Then you must submit you calculation with:

```
> sbatch sub_file
```

or

```
> ccc_msub sub_file
```

following the queue system of the cluster.

You can see the **TRUST** & **TrioCFD** user slides in the "Parallel calculation" section for more informations and this exercise of the **TRUST** tutorial.

7.5 Visualization

To visualize your probes, you can use the CurvePlot tool, with the command line:

```
> trust -evol my_parallel_data_file
```

or use Gnuplot or any software which reads values in columns in a file.

There are three ways to visualize your parallel results with VisIt:

- HPCDrive on CCRT/TGCC clusters: opens a deported graphic session on dedicated nodes with more memory (on curie cluster: HPCDrive),
- local mode: copy your results from the cluster to your local computer and open it with a local parallel version of VisIt with:

  ```
  > visit -np 4 &
  ```

You can have a look at the **TRUST** & **TrioCFD** user slides in the "Parallel calculation description" section.

7.6 Useful informations

7.6.1 Modify the mesh

If you want to modify your mesh, you have two possibilities:

- modify the *my_ data_file.data* file and run:

  ```
  > make_PAR.data my_data_file [parts_number]
  ```

  Be carefull it will erase the *SEQ_ my_ data_file.data*, *DEC_ my_ data_file.data* and *PAR_ my_ data_file.data* files and create new ones.
  Then it will run the new *DEC_ my_ data_file.data* file which gives your new *DOM_ 000n*.**Zones** files.

- modify the meshing part of file *DEC_ my_ data_file.data* and run it with:

  ```
  > trust DEC_my_data_file
  ```

Then run the parallel calculation normally, on the new *DOM_ 000n*.**Zones** files.

```
> trust PAR_my_data_file procs_number
```

7.6.2 Modify calculation parameters

If you want to modify the calculation parameters, you can modify:

- the file *my_ data_file.data* and run:

  ```
  > make_PAR.data data_file_name [parts_number]
  ```

  But it will erase the *SEQ_ my_ data_file.data*, *DEC_ my_ data_file.data* and *PAR_ my_ data_file.data* files and create new ones.
  Then it will run the new *DEC_ my_ data_file.data* file which gives your new *DOM_ 000n*.**Zones** files.
  Note that in that case, you don't need to re-create the mesh so you can use the second point below:

- modify the *PAR_ my_ data_file.data* file without running `make_PAR.data` script.

Then run the *PAR_ my_ data_file.data* file with:

```
> trust PAR_my_data_file procs_number
```

Note that if after a certain time, you want to reopen an old case and understand want you did in it without any doubts, you may create two files by hands:

- one "BuildMeshes.data" file only for the mesh and the cut of the mesh, and
- one "calculation.data" file for the parallel calculation.

You will run it like:

```
> trust BuildMeshes
> trust calculation processors_number
```

For this point, you can have a look at this exercise of the **TRUST** tutorial.