LAPLACIAN

CEA-EDF-INRIA summer school in numerical analysis
CDO schemes in
code_saturne

June, 2023

---

1 Introduction

This test case is a quick start tutorial. It corresponds to one of the simplest verification test case allowing one to check the consistency of a space discretization scheme. This will enable each participant to check the installation process/environment settings are correctly set, to get familiar with the code_saturne environment and perform a first post-processing.

1.1 Installation of the container

First, here is shortly described how the singularity container file has to be installed. This step is mandatory before continuing the tutorial. More details are available here:

`https://github.com/npaster/summer_school_2023`

One assumes that the `salome_meca_2022.1.0_lgpl_summer.sif` singularity file has already been downloaded on your laptop and the application *singularity* is installed on your system. If the container has not been installed, write the following command lines:

In a terminal

```
cd /where/you/want/to/store/data/
mkdir -p SUMMER_SCHOOL
cd SUMMER_SCHOOL
mv /path/to/salome_meca_2022.1.0_lgpl_summer.sif .
singularity run --app install salome_meca_2022.1.0_lgpl_summer.sif
```

At this stage, the container is installed. Then, write the following command to execute a shell in the container environment (this command may take time).

In a terminal

```
./salome_meca_2022.1.0_lgpl_summer --shell
Singularity> source /opt/public/scibian9_mpi.sh
```

1.2 Data for this tutorial

Data for this tutorial are available in the archive `CDO_TUTORIAL_00.tar.xz` in the Github repository:

`https://github.com/npaster/summer_school_2023/tree/main/TP1`

Now, to retrieve the set of data files, please write:

In a terminal

```
Singularity> mkdir -p CDO
Singularity> mv /path/to/CDO_TUTORIAL_00.tar.xz CDO/
Singularity> cd CDO
Singularity> unxz CDO_TUTORIAL_00.tar.xz
Singularity> tar -xvf CDO_TUTORIAL_00.tar
cd 00_LAPLACIAN/
```

2 Definition of the test case

The computational domain is a unit cube denoted by Ω, an open bounded connected polyhedral subset of $\mathbb{R}^3$ (all computations are performed in three dimension). $\partial\Omega = \overline{\Omega} \setminus \Omega$ denotes the boundary of the domain. The problem at stake solves:

$$\left\{ \begin{array}{rcll} -\Delta Y & = & 0 & \text{in } \Omega \\ Y & = & Y_D & \text{on } \mathcal{D} \subset \partial\Omega \\ -\underline{\nabla}(Y) \cdot \underline{n} & = & \Phi_N & \text{on } \mathcal{N} \end{array} \right. \tag{1}$$

where Y is the scalar-valued field defined on the domain Ω and belongs to $H^1(\Omega)$, $\underline{n}$ denotes the unit outward normal vector to $\partial\Omega$.

The boundary conditions on the field Y are composed of Dirichlet conditions on $\mathcal{D}$ corresponding to the two surfaces $x = 0$ and $x = 1$. Elsewhere, a homogeneous Neumann boundary conditions on $\mathcal{N}$ is considered as depicted in Figure 1.

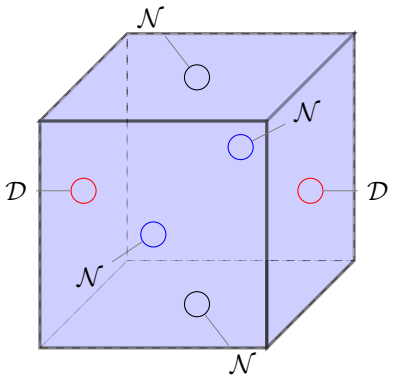

Figure 1: Set of boundary conditions.

So, one ends up with a 1D problem where the exact solution Y_e is defined by:

$$Y_e = x \tag{2}$$

3 Methodology

The case settings are located in the directory `00_LAPLACIAN` (a code_saturne study). The case `REFERENCE_SETTINGS` inside this study is ready to run. The case `MY_FIRST_SETTINGS` is a copy of the reference case. The initial structure of the study is depicted in Figure 2.

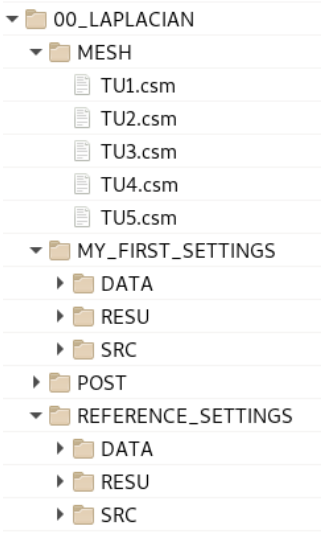

Figure 2: Structure of the case `00_LAPLACIAN`

3.1 Check the correct installation of code_saturne

In a terminal

```
code_saturne update -c REFERENCE_SETTINGS -c MY_FIRST_SETTINGS
cd REFERENCE_SETTINGS/
code_saturne run
```

The `update` command updates pathes stored in a `CASE` directory. This will generate a result directory named by default with the convention `YYMMDD-HHMM` inside the `RESU` directory. This results directory contains several files related to logging information or post-processing and a copy of the setup (`setup.xml` file for the setup relying on the GUI and `src/cs_user*.c` files for the user-defined source code.

3.1.1 Post-processing

A visualization of the results is possible using `paraview` for instance.

In a terminal

```
cd RESU/20230626*
paraview postprocessing/RESULTS_FLUID_DOMAIN.case &
```

Then, click on the 

button.

3.1.2 Analysis

Among the *log* files:

- `listing` or `run_solver.log` is the main log file with information about the computation and the computed solution.
- `performance.log` is the log file dedicated to the timer information, the linear algebra, the I/O and parallelism performances.
- `setup.log` is a summary of (nearly) all options set for the computation.

Variable information. Open the `listing` file, the following lines should be written in the section related to the *vaiable information*

```
================================================================================
#      Solve steady-state problem(s)
================================================================================

  ** Field values on vertices
     ------------------------

   field             minimum         maximum        set mean
-  ---------  --------------  --------------  --------------
v  potential               0               1             0.5
```

giving information on the "potential" field which is the variable associated to the equation "Laplacian".

Settings information. Open the `setup.log` file with a text editor to know what are the default `CDO` settings. There are several sections related to the space discretization, the boundary conditions or the linear algebra. Here are some examples of information available inside this file.

```
### Laplacian | High-level settings
  * Laplacian | Type: User-defined
  * Laplacian | Terms: unsteady:*False*, convection:*False*, diffusion:*True*
  * Laplacian | Terms: curl-curl:*False*, grad-div:*False*
  * Laplacian | Terms: reaction:*False*, source term:*False*, force internal values: *False*
  * Laplacian | Space scheme:       CDO vertex-based
  * Laplacian | Space poly degree:  0
  * Laplacian | Verbosity:          1
```

To know which term is used in the equation called "Laplacian"

```
### Laplacian | Boundary condition settings
  * Laplacian | Boundary conditions | Default: Homogeneous Neumann
  * Laplacian | Boundary conditions | Enforcement: weak using an algebraic manipulation
  * Laplacian | Boundary conditions | Number of definitions: 2
```

To know how the Dirichlet boundary conditions are enforced and how many boundary conditions are set (in addition to the default boundary condition).

```
### Laplacian | Diffusion term settings
  * Laplacian | Diffusion property: unity

        Diffusion Hodge op.  | Type: EpFd
        Diffusion Hodge op.  | Algo: Orthogonal Consistency/Stabilization (OCS)
        Diffusion Hodge op.  | Algo.Coef: 6.667e-01
        Diffusion Hodge op.  | Associated property: unity
        Diffusion Hodge op.  | Property inversion: *False*
```

To know how the discrete Hodge operator related to the diffusion term is precisely defined.

3.2 Make variants of the initial test case

Go to the case `MY_FIRST_SETTINGS`

3.2.1 Refined the mesh

- Write the following command in order to open the GUI of code_saturne. Then, open the page *Mesh* in the left panel and edit the number of cells in the x-direction as shown in Figure 3.

In a terminal

```
cd DATA
./code_saturne &
```

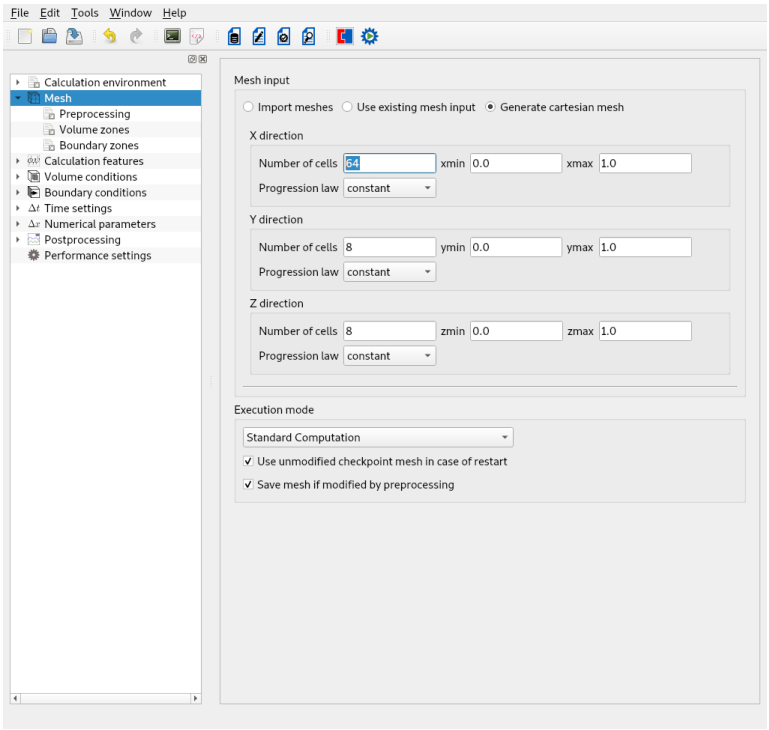

Figure 3: Page for the modification of the Cartesian mesh. Modify only the x-direction since the problem is only 1D.

- To launch the computation, either proceed as before (`code_saturne run` in a terminal) or click on the button in the toolbar and then click on the *Save case and run calculation* button as shown in Figure 4.
- Proceed as before to visualize the computed solution with `paraview` knowing that the result directory is located in `MY_FIRST_SETTINGS/RESU/20230626*`

3.2.2 Use a predefined tetrahedral mesh

Meshes already preprocessed with code_saturne are named `*.csm` (code_saturne mesh) and are ready to be used. To change the current settings related to the Cartesian mesh generated on-the-fly by a tetrahedral mesh, one has to:

- Open the code_saturne's GUI and edit the *Mesh* page as depicted in the Figure 5
- Generate the two boundary zones by editing the page *Mesh/Boundary zones* as depicted in Figure 6.
- Launch the computation as previously detailed (button)
- Visualize the computed solution with `paraview`. A predefined post-processing can be loaded using `File > Load State File.. > Choose the file in 00_LAPLACIAN/POST/post.pvsm` and then proceed as depicted in Figure 7 (select your last results directory under `MY_FIRST_SETTINGS/RESU/`)

3.2.3 Heterogeneous diffusivity

To conclude this tutorial, one considers a diffusivity property λ which is heterogeneous. λ is constant and equal to 3 in the volume corresponding to $x < 0.5$ whereas λ is equal to 1 in the volume corresponding to $x \geq 0.5$. One first create a new case from an existing one. One assumes that the current directory is `00_LAPLACIAN`.

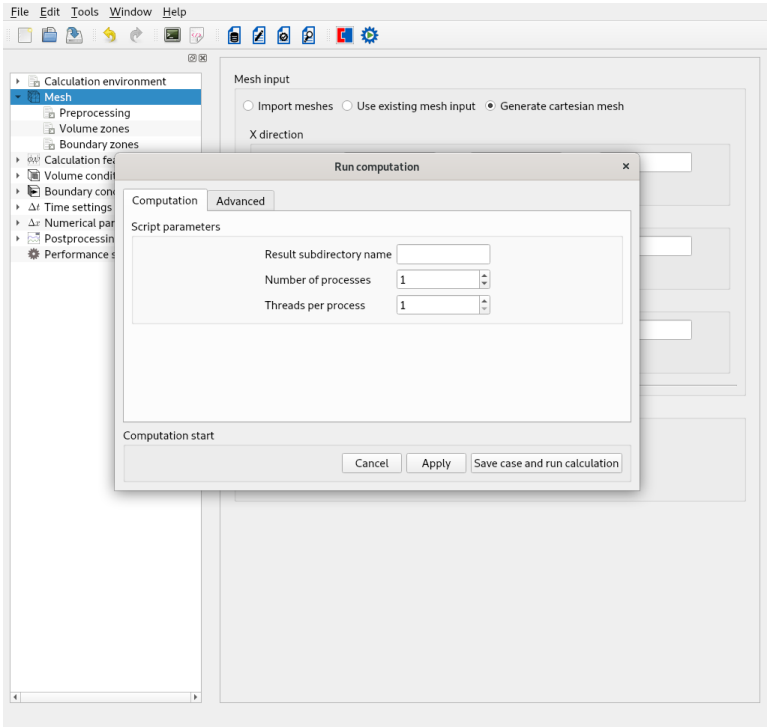

Figure 4: Page for launching a computation. Click on the *Save case and run calculation* button.

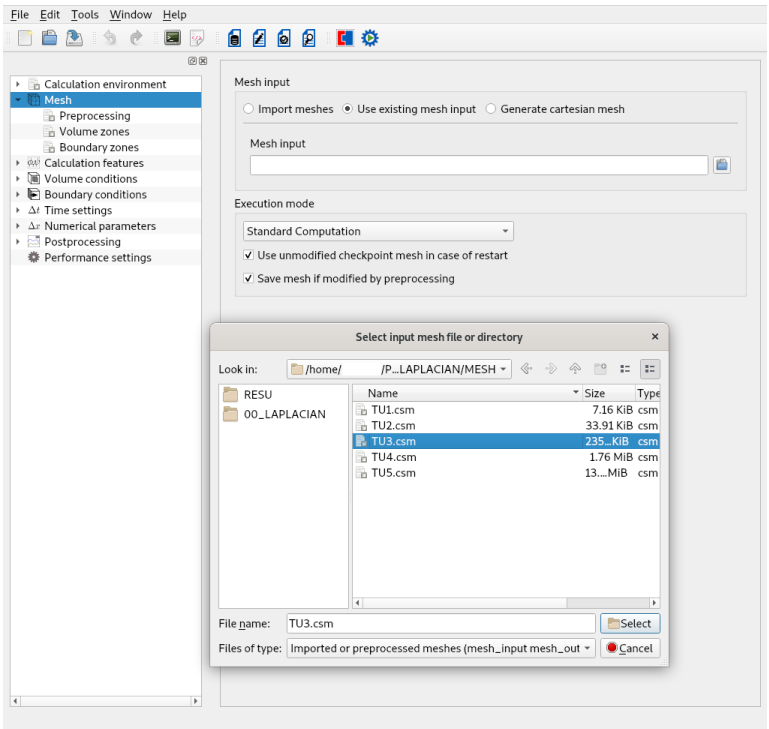

Figure 5: Choose a code_saturne mesh (use an existing mesh input)

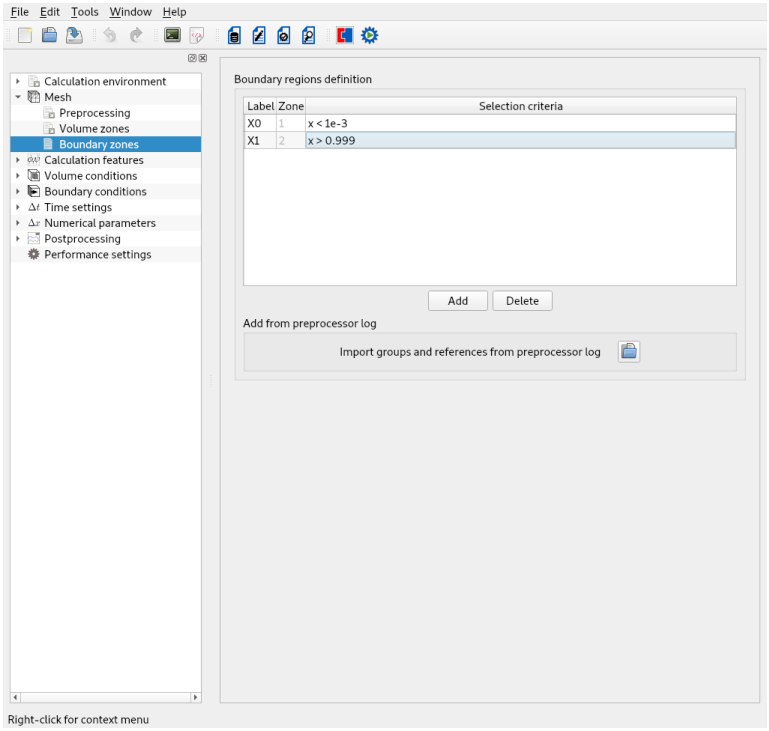

Figure 6: Edit the "selection criteria" column to define the two boundary zones X0 and X1

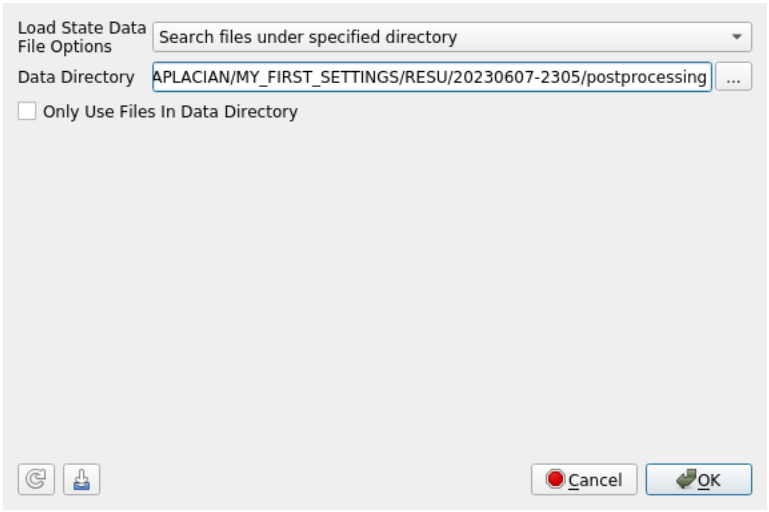

Figure 7: Options to set when loading a paraview state file `*.pvsm`

In a terminal

```
code_saturne create -c HETEROGENEOUS_SETTINGS --copy-from MY_FIRST_SETTINGS
cd HETEROGENEOUS_SETTINGS/DATA
./code_saturne &
```

Here are the different steps to follow:

- Add two new volume zones by editing the page *Mesh/Volume zones* as depicted in Figure 8.
- Save the GUI state and open the `../SRC/cs_user_parameters.c` user source file.
- Edit the function `cs_user_model()` in order to add a new (isotropic) property named "diffusivity" as follows (do not hesitate to copy/paste the following blocks):

C code

```
/* Add a new property named "diffusivity" */

cs_property_add("diffusivity", CS_PROPERTY_ISO);
```

- Edit the function `cs_user_finalize_setup()` and, especially the part related to the diffusion term, in order 1 (to) define the property "diffusivity" and (2) to associate this property to the equation "Laplacian" as follows (the part related to the boundary conditions is unchanged):

C code

```
/* Define the property in the two volume zones */

cs_property_t  *pty = cs_property_by_name("diffusivity");

cs_property_def_iso_by_value(pty,
                             "Left",
                             3);

cs_property_def_iso_by_value(pty,
                             "Right",
                             1);

/* Associate the previous property to the "Laplacian"  equation */

cs_equation_add_diffusion(eqp, pty);
```

If the variable `eqp` is not already defined in the function, add this line at the top of the function:

C code

```
  cs_equation_param_t  *eqp = cs_equation_param_by_name("Laplacian");
```

- Save the file `cs_user_parameters.c` and launch the computation

In a terminal

```
code_saturne run
cd ../RESU/20230626*
```

- Postprocess the result with `paraview` using the same state file in `00_LAPLACIAN/POST/post.pvsm`. You should obtain the same view as the one depicted in Figure 9. The computed solution is piece-wise linear and the value at $x = 0.5$ should be equal to $\frac{\lambda_{\text{Right}}}{\lambda_{\text{Right}}+\lambda_{\text{Left}}} = \frac{1}{4}$.

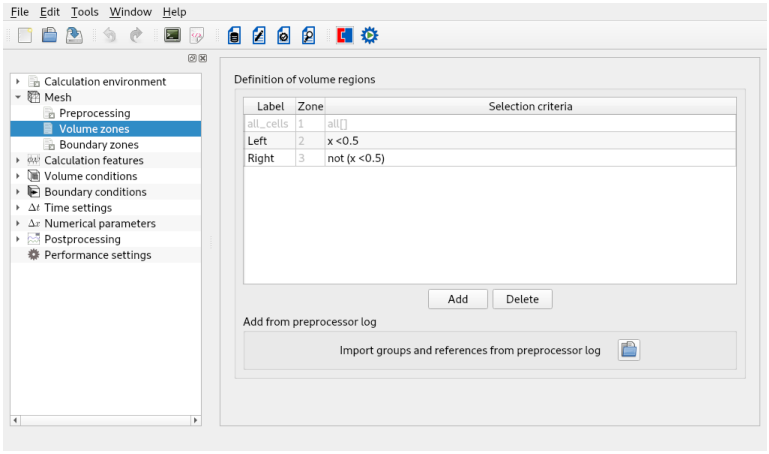

Figure 8: Edit the "selection criteria" column to define the two volume zones Left and Right

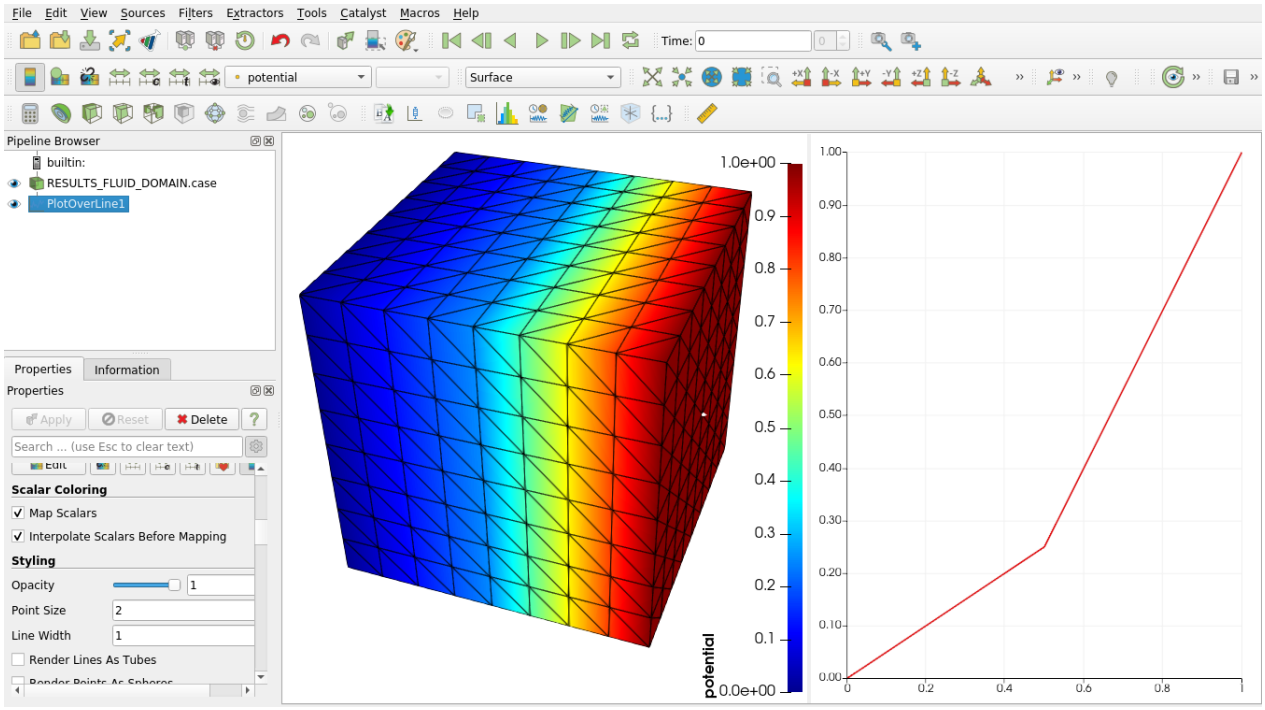

Figure 9: Resulting view in `paraview`.

3.2.4 Extra: Change the space discretization scheme

- Edit the `../SRC/cs_user_parameters.c` user source file and in particular, the function `cs_user_parameters()` as follows:

C code

```
/* Retrieve the set of equation parameters and then associate the
   previous property to this equation */

cs_equation_param_t  *eqp = cs_equation_param_by_name("Laplacian");

cs_equation_param_set(eqp, CS_EQKEY_SPACE_SCHEME, "cdo_fb");
```

- Available choices for a scalar-valued Laplacian problem are: "cdo_vb" (the default value), "cdo_fb", "cdo_vcb",

It is possible to add a prefix to the result directory so that the name is more meaningful. Launch code_saturne as follows:

In a terminal

```
code_saturne run --id-prefix=CDOFB.
```

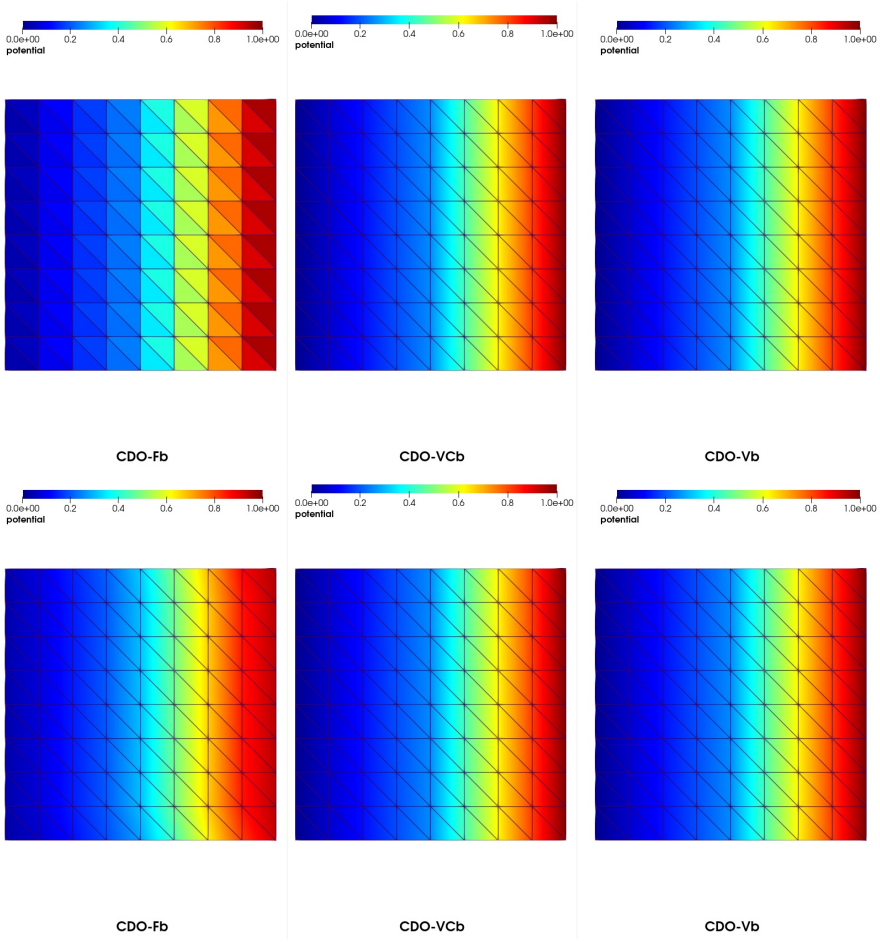

Figure 10: Comparison between `CDO-Fb`, `CDO-VCb` and `CDO-Vb` schemes. `CDO-Fb` schemes with cell values (Top) and with a `Cell Data to Point Data` filter in `paraview`.