ANISOTROPIC DIFFUSION

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

June, 2023

---

1 Introduction

This test case is a benchmark created for the Finite Volume for Complex Applications (FVCA) conference [1]. The aim of this benchmark was to compare different space discretizations on the resolution of an anisotropic diffusion problem with general/polyhedral meshes. One assumes that the introductory tutorial `00_LAPLACIAN` has already been done to get familiar with the code_saturne's environment.

A reference setting is given with the possibility to check the order convergence. Different numerical options may be tested to observe their influence on the accuracy and efficiency of the numerical scheme. Moreover, several mesh sequences are available from uniform Cartesian meshes to highly distorted meshes to assess the influence of the mesh quality on the numerical solution.

1.1 Data for this tutorial

Data for this tutorial are available in the archive `CDO_TUTORIAL_01.tar.xz` in the Github repository:

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

Now, to retrieve the set of data files in the directory `CDO` (created in the previous tutorial), please write:

In a terminal

```
Singularity> cd /path/to/CDO
Singularity> mv /path/to/CDO_TUTORIAL_01.tar.xz .
Singularity> unxz CDO_TUTORIAL_01.tar.xz
Singularity> tar -xvf CDO_TUTORIAL_01.tar
cd 01_3D_ANISOTROPIC_DIFFUSION
```

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:

$$\left\{ \begin{array}{rcll} -\mathrm{div}\left(\underline{\underline{K}} \cdot \nabla Y\right) & = & f & \text{on } \Omega \\ Y & = & Y_d & \text{on } \partial\Omega \end{array} \right. \tag{1}$$

where $Y \in H^1(\Omega)$ is the scalar-valued variable defined on the domain Ω and the tensor $\underline{\underline{K}}$ is uniform in Ω and equal to

$$\underline{\underline{K}} = \begin{pmatrix} 1 & 0.5 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix}.$$

The boundary conditions on the variable Y are Dirichlet boundary conditions on the whole boundary. The source term f is defined to comply with the exact solution Y_e given by

$$Y_e(\underline{x}) = 1 + \sin(\pi x) \sin\left(\pi\left(y + \frac{1}{2}\right)\right) \sin\left(\pi\left(z + \frac{1}{3}\right)\right) \tag{2}$$

This solution is bounded between $0 \leq Y_e \leq 2$ in the computational domain.

3 Mesh sequences

In this tutorial, we will consider several mesh sequences built for the FVCA6 benchmark: from uniform Cartesian hexahedral meshes (`Hexa`), free tetrahedral meshes (`Tetra`), prism meshes with triangle bases (`PrT`) and prism meshes with general basis yielding polyhedral meshes (`PrG`), polyhedral mesh sequence with hanging nodes (`CB`) and highly distorted hexahedral mesh sequence named Kershaw (`Ker`). The `Ker` and `Tetra` mesh sequences are not uniformly refined so that the convergence order may variate between two successive meshes. Therefore, one also considers the `TetU` mesh sequence which is a uniformly refined tetrahedral mesh sequence. An example of a mesh for each mesh sequence is depicted in Figure 1 (except for the uniform Cartesian mesh whose shape is obvious).

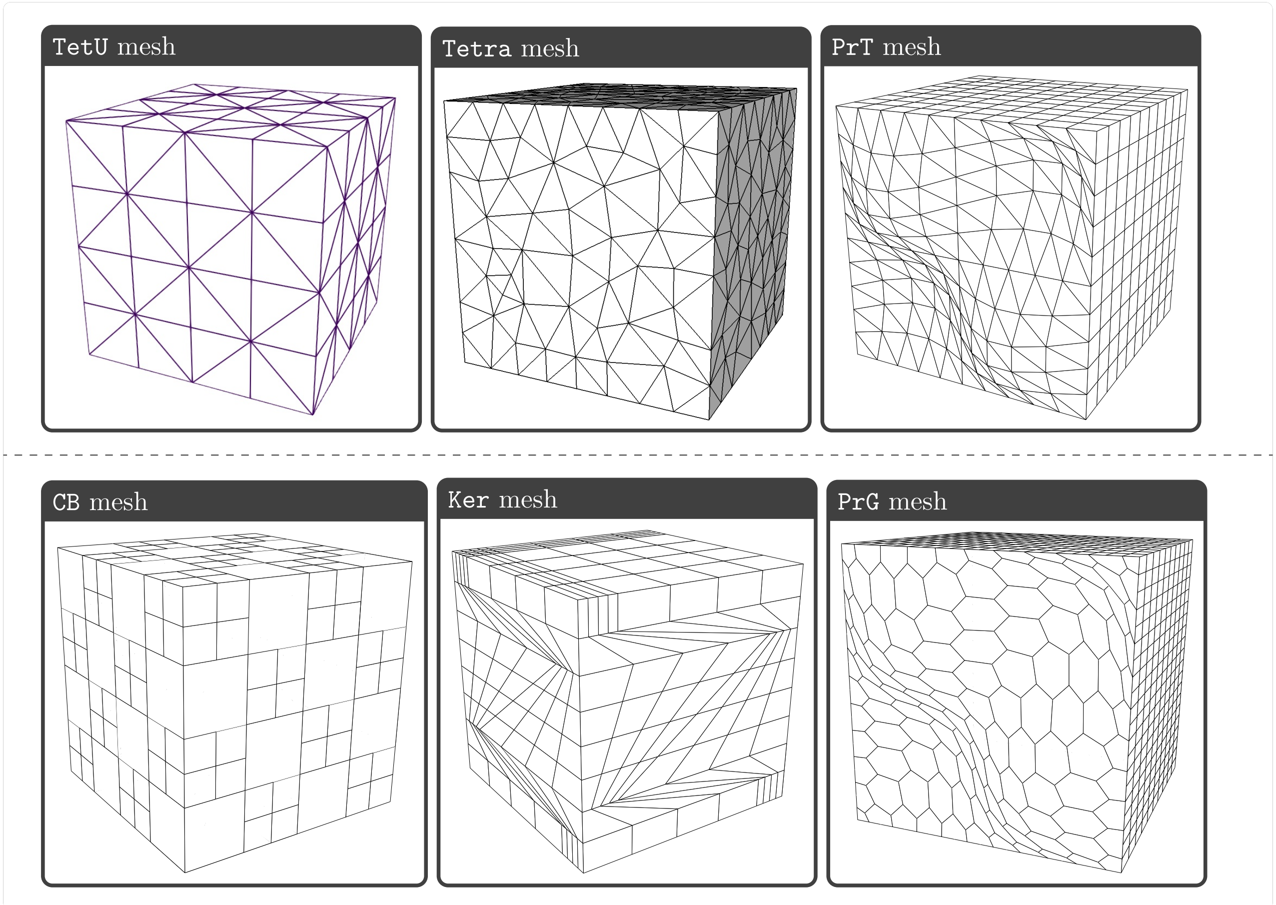

Figure 1: An overview of mesh sequences.

4 Error norms

4.1 Tested space discretizations

The three following families of space discretizations will be tested in this tutorial. According to the discretization scheme, degrees of freedom (DoFs) are located at different locations.

- `CDO-Vb` family corresponds to schemes with DoFs at vertices. V denotes the set of mesh vertices and $\#V$ represents the number of mesh vertices. The numerical solution $\mathbf{Y} \in \mathcal{V} \equiv \mathbb{R}^{\#V}$.
- `CDO-VCb` family corresponds to schemes with DoFs at vertices and cells. C denotes the set of mesh cells and $\#C$ represents the number of mesh cells. The numerical solution $\mathbf{Y} \in \mathcal{V} \times \mathcal{C} \equiv \mathbb{R}^{\#V+\#C}$.
- `CDO-Fb` family corresponds to schemes with DoFs at faces and cells. F denotes the set of mesh faces and $\#F$ represents the number of mesh faces. The numerical solution $\mathbf{Y} \in \mathcal{F} \times \mathcal{C} \equiv \mathbb{R}^{\#F+\#C}$.

4.2 Definition of the discrete error norms

Several discrete error norms which are computed by the functions defined in in `cs_user_extra_operations.c` are detailed in what follows.

- A relative discrete L_2-like error norm on the variable at cells

$$\mathsf{E}_2^{\mathsf{C}}(\mathbf{Y}) = \sqrt{\frac{\sum_{c \in C} |c| \left(Y_e(\underline{x}_c) - \mathbf{Y}_c\right)^2}{\sum_{c \in C} |c| Y_e(\underline{x}_c)^2}}$$

  where $\underline{x}_c$ is the cell center of the cell c among the set of cells C, $|c|$ is its volume and $\mathbf{Y}_c$ is the value of the discrete variable in the cell c. For `CDO-Vb` schemes, the value at the cell center is interpolated from the knowledge of the numerical solution at vertices.

  - For `CDO-Vb` and `CDO-VCb`, a variant at vertices is also considered:

$$\mathsf{E}_2^{\mathsf{V}}(\mathbf{Y}) = \sqrt{\frac{\sum_{c \in C} \sum_{v \in V_c} |\mathfrak{p}_{v,c}| \left(Y_e(\underline{x}_v) - \mathbf{Y}_v\right)^2}{\sum_{c \in C} \sum_{v \in V_c} |\mathfrak{p}_{v,c}| Y_e(\underline{x}_v)^2}}$$

    where $\underline{x}_v$ is the location of the mesh vertex $v \in V$, $|\mathfrak{p}_{v,c}|$ is the volume of dual cell $\tilde{c}(v)$ intersecting the (primal) cell c ($\mathfrak{p}_{v,c} = c \cap \tilde{c}(v)$); see Figure 2. $\mathbf{Y}_v$ is the value of the discrete variable in the vertex v. V_c denotes the set of vertices belonging to the cell c.

  - For `CDO-Fb`, a variant at faces is also considered:

$$\mathsf{E}_2^{\mathsf{F}}(\mathbf{Y}) = \sqrt{\frac{\sum_{c \in C} \sum_{f \in F_c} |\mathfrak{p}_{f,c}| \left(Y_e(\underline{x}_f) - \mathbf{Y}_f\right)^2}{\sum_{c \in C} \sum_{f \in F_c} |\mathfrak{p}_{f,c}| Y_e(\underline{x}_f)^2}}$$

    where $\underline{x}_f$ is the location of the face barycenter of the face $f \in F$, $|\mathfrak{p}_{f,c}|$ is the volume of pyramid of base f and apex $\underline{x}_c$; see Figure 2. $\mathbf{Y}_f$ is the value of the discrete variable on the face f. F_c denotes the set of faces belonging to the cell c.

- A discrete L_∞-like error norm on the variable at cells

$$\mathsf{E}_\infty^{\mathsf{C}}(\mathbf{Y}) = \max_{c \in C} \left(Y_e(\underline{x}_c) - \mathbf{Y}_c\right)$$

  - For `CDO-Vb` and `CDO-VCb` schemes, a variant at vertices is also considered:

$$\mathsf{E}_\infty^{\mathsf{V}}(\mathbf{Y}) = \max_{v \in V} \left(Y_e(\underline{x}_v) - \mathbf{Y}_v\right)$$

– For `CDO-Fb` schemes, a variant at faces is also considered:

$$\mathsf{E}^{\mathsf{F}}_{\infty}(\mathbf{Y}) = \max_{f \in F} \left(Y_e(\underline{x}_f) - \mathbf{Y}_f \right)$$

- a relative discrete L_2-like norm on the gradient in cells

$$\mathsf{E}^{\mathsf{C}}_{2}(\nabla \mathbf{Y}) = \sqrt{\frac{\displaystyle\sum_{c \in C} |c| \, |\underline{\nabla} Y_e(\underline{x}_c) - \underline{\nabla}_c \mathbf{Y}|^2}{\displaystyle\sum_{c \in C} |c| |\underline{\nabla} Y_e(\underline{x}_c)|^2}}$$

where $\underline{\nabla}_c \mathbf{Y}$ is a constant vector-valued quantity defining the discrete cell gradient in the cell c. The way to reconstruct this quantity depends on the space discretization.

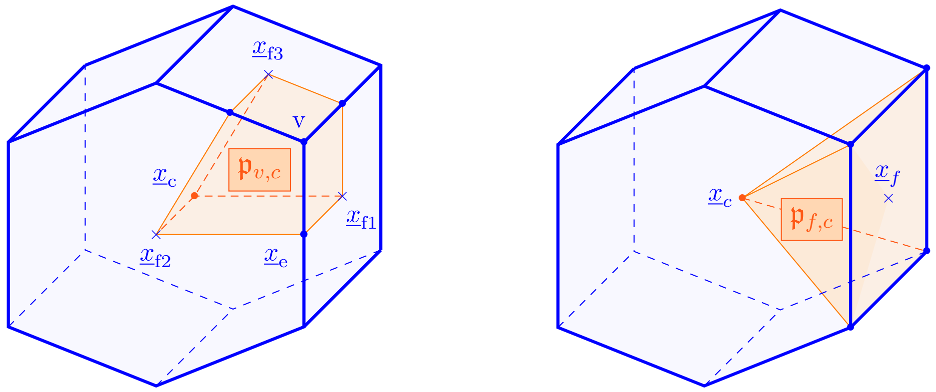

Figure 2: Example of a cell partition based on vertices ($\mathfrak{p}_{v,c}$) on the left or based on faces ($\mathfrak{p}_{f,c}$) on the right.

Convergence rates. The order of convergence (OoC) between two successive meshes $\mathtt{k}$ and $\mathtt{k}-1$ in a mesh sequence is defined as follows:

$$\mathsf{OoC}(\mathtt{k}) = -3 \frac{\log\left(\frac{\mathsf{E}^*_*(\mathbf{Y}_\mathtt{k})}{\mathsf{E}^*_*(\mathbf{Y}_{\mathtt{k}-1})}\right)}{\log\left(\frac{\mathtt{nsys}(\mathtt{k})}{\mathtt{nsys}(\mathtt{k}-1)}\right)}$$

where E^*_* is one of the error norms previously introduced and $\mathtt{nsys}(\mathtt{k})$ is the number of unknowns in the linear system associated to the mesh $\mathtt{k}$ in a mesh sequence and $\mathbf{Y}_\mathtt{k}$ is the numerical solution for the mesh $\mathtt{k}$.

5 First computation

5.1 Case settings

The case settings are all located in the directory `01_3D_ANISOTROPIC_DIFFUSION` (a code_saturne study). The case `REFERENCE_SETTINGS` inside this study is ready to run. Before running the first calculation, one gives some details on the case settings and its translation in terms of code_saturne's functions to call. In the `SRC` directory, there are

- the user-defined source file `cs_user_parameters.c` with functions
  – `cs_user_model()` where the equation called "FVCA6" associated to the variable named "scalar1" to solve is added

C code

```
cs_equation_add_user("FVCA6",        /* equation name */
                     "scalar1",      /* associated variable field name */
                     1,              /* dimension of the unknown */
                     CS_PARAM_BC_HMG_DIRICHLET); /* default boundary */
```

along with the property related to the diffusion term called “conductivity”

C code

```
cs_property_add("conductivity",      /* property name */
                CS_PROPERTY_ANISO);  /* type of material property */
```

- cs_user_parameters() where the default numerical options can be overwritten. The function used to do this task is cs_equation_param_set. For instance, the space discretization can be modified using

C code

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

where eqp is declared at the beginning of the function cs_user_parameters() as follows:

C code

```
  cs_equation_param_t  *eqp = cs_equation_param_by_name("FVCA6");
```

- cs_user_finalize_setup() where one details how
    * the property is defined (after having retrieved the pointer to the structure)

    C code

    ```
    cs_property_t  *conductivity = cs_property_by_name("conductivity");

    cs_property_def_aniso_by_value(conductivity, /* property structure */
                                   "cells",      /* volume zone name */
                                   k_tensor);    /* values */
    ```

    * the boundary conditions are set on all boundary faces thanks to a function called *get_sol* allowing one to get the exact solution at each point (x, y, z) of the computational domain. The detailed definition of the function *get_sol* is available in the same user-defined source file.

    C code

    ```
    cs_equation_add_bc_by_analytic(eqp,
                                   CS_PARAM_BC_DIRICHLET,
                                   "boundary_faces", /* zone name */
                                   get_sol,          /* function */
                                   NULL);            /* optional structure */
    ```

    * the source term is set in all cells thanks to a function called *get_source_term* allowing one to get the exact solution at each point (x, y, z) of the computational domain. The detailed definition of the function *get_source_term* is available in the same user-defined source file.

    C code

    ```
    cs_xdef_t  *st = cs_equation_add_source_term_by_analytic(eqp,
                                                             "cells",
                                                             get_source_term,
                                                             NULL);
    ```

- the user-defined source file cs_user_extra_operations.c allowing one to compute error norms and global quantities of interest to analyze the numerical solution. The source code inside this file corresponds to an advanced usage. There is no need to modify this file during this tutorial.

5.2 Run

To run a first calculation with default mesh (a Cartesian `8x8x8` mesh generated on-the-fly), write

In a terminal

```
cd SUMMER_SCHOOL/CDO/01_3D_ANISOTROPIC_DIFFUSION
code_saturne update -c REFERENCE_SETTINGS
cd REFERENCE_SETTINGS/
code_saturne run
```

5.2.1 Analysis

To visualize the results:

In a terminal

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

As detailed in the `00_LAPLACIAN` tutorial you can load a state file to display predefined post-processings with vertex-based schemes (`01_3D_ANISOTROPIC_DIFFUSION/POST/post_vb_schemes.pvsm`). An illustration of this predefined post-processings is given in Figure 3.

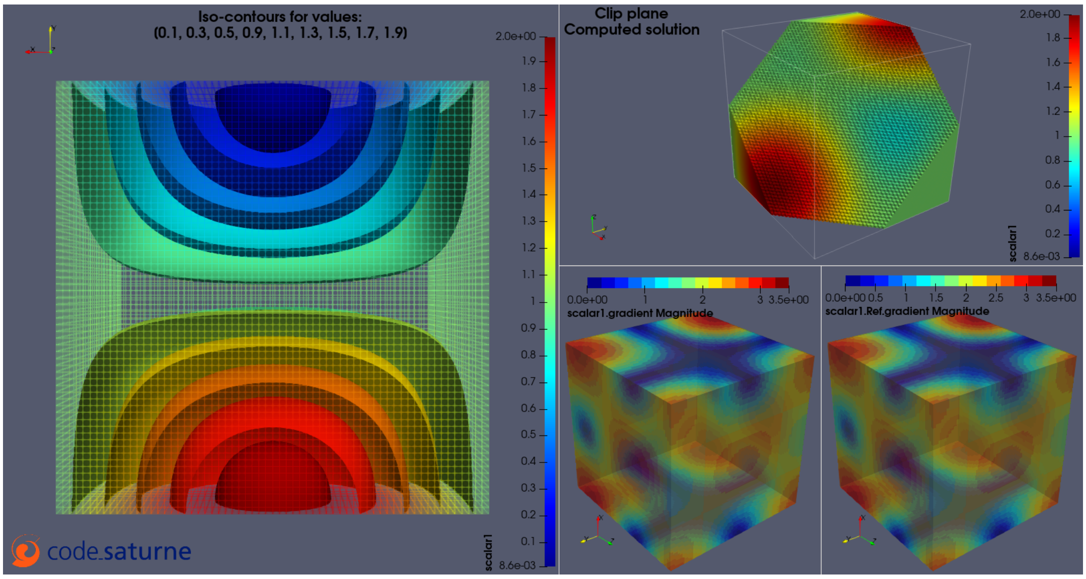

Figure 3: Predefined post-processings for vertex-based schemes

Besides the log files automatically generated by code_saturne, the `FVCA6.log` file gathers the values of several error norms computed in extra-operations (user-defined `cs_user_extra_operations.c`). These quantities will be useful for the evaluation of the order of convergence. For instance, you should find the following values in this file.

Kind of information gathered in the file FVCA6.log (`8x8x8 Cartesian mesh`)

```
-cvg- Err.C.Max          4.982388e-02
-cvg- ErrL2.C            1.725575e-02
-cvg- ErrInf.Grad.C      3.497488e-02
-cvg- ErrL2.Grad.C       4.472543e-02
-cvg- ErrEne.C           3.750978e-02
```

6 First convergence study

6.1 Case settings

The starting point is to duplicate the `REFERENCE_SETTINGS` case. From the directory `01_3D_ANISOTROPIC_DIFFUSION` write

In a terminal

```
code_saturne create -c CDOVB.DGA --copy-from REFERENCE_SETTINGS
cd CDOVB.DGA/DATA
./code_saturne
```

The directory `01_3D_ANISOTROPIC_DIFFUSION` should be as in Figure 4. Then, update the *Mesh* page as illustrated in Figure 5 and save & quit the GUI.

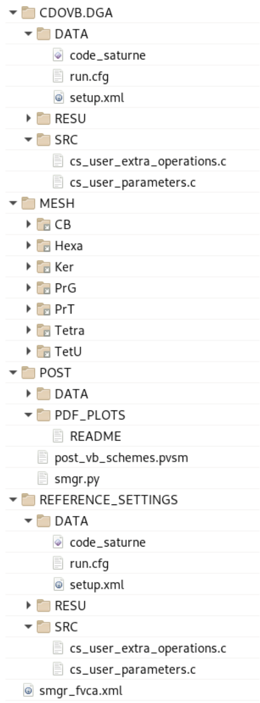

Figure 4: State of the directory `01_3D_ANISOTROPIC_DIFFUSION`

Then, you are ready to run a first computation with a polyhedral mesh.

In a terminal

```
code_saturne run --id-prefix=CB2.
```

You can check the results in `../RESU/CB2.20230626*`

6.2 Convergence study

To perform a convergence study, we will use the *studymanager* capabilities of code_saturne. One first generates a *xml* file specifying all needed computations (a run for each mesh of each mesh sequence) and all associated post-processings to do. From the directory `01_3D_ANISOTROPIC_DIFFUSION` write

In a terminal

```
./POST/smgr.py -g
code_saturne smgr -f smgr_fvca.xml -r -p --dest ../01_3D_ANISOTROPIC_DIFFUSION_RUNS
```

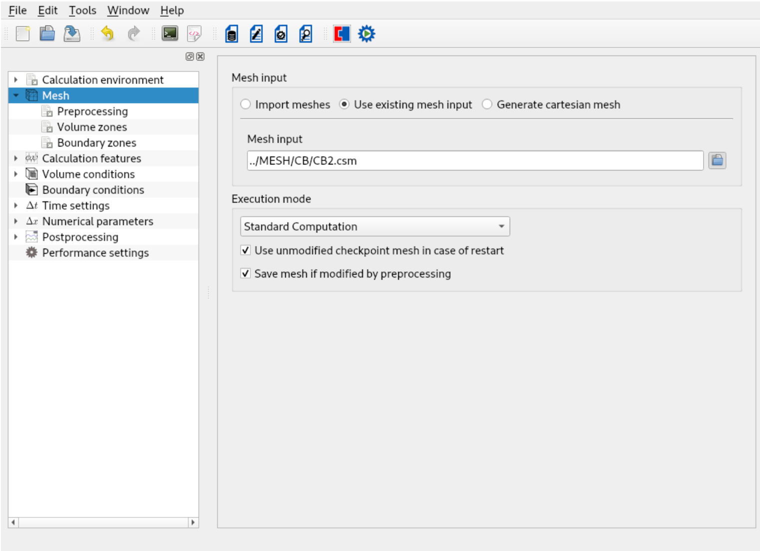

Figure 5: Use a preprocessed mesh called 'CB2.csm' which is in the directory `01_3D_ANISOTROPIC_DIFFUSION/MESH/CB/`

The first command generates the file *smgr_fvca.xml* storing all actions that the study manager has to do.

Remark 1 (Overwrite)**.** *To overwrite a previous destination directory, please add the* `--rm` *option in the previous command line. To get more details on the available options with the studymanager tool, write* `code_saturne smgr --help`*.*

After all computations are completed, you could open the PDF files generated automatically in the directory `../01_3D_ANISOTROPIC_DIFFUSION_RUNS/POST/PDF_PLOTS` A PDF gathering all plots is also available in `../01_3D_ANISOTROPIC_DIFFUSION_RUNS/synthesis.pdf`. You should get similar results to the one depicted below in Figure 6.

7 Other convergence studies

Several options may be modified and one can compare the accuracy and the order of convergence by adding new cases. For instance, from the root directory of the study `01_3D_ANISOTROPIC_DIFFUSION`

In a terminal

```
code_saturne create -c CDOFB.GCR --copy-from CDOVB.DGA
```

GCR means Generalized Crouzeix–Raviart. This is a variant of discrete Hodge operator. Then, edit the user-defined `CDOFB.GCR/SRC/cs_user_parameters.c` file with these new values as follows:

C code

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

The next step is the edition of the file `smgr.py` around lines 143

Python code

```
f_cases.append({'name':"CDOFB.GCR", 'tag':"fb", 'ls':"--"})
```

to activate the new case. The following lines will generate a new `smgr_fvca.xml` file gathering more calculations to do and additional post-processings and then launch a set of calculations (if a result directory is already defined then the calculation is skipped).

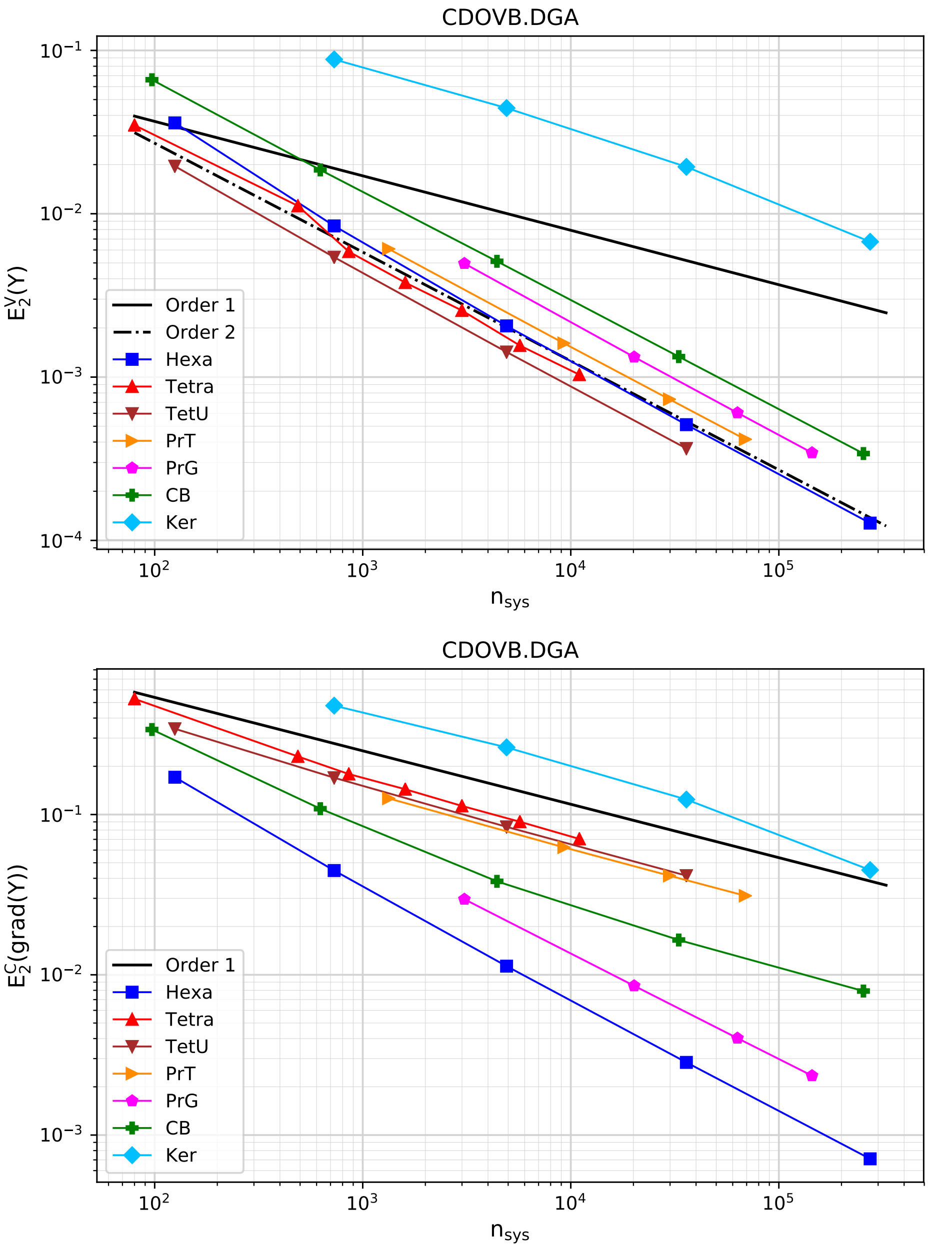

Figure 6: Convergence plots for the case `CDOVB_DGA`. Top: discrete L_2 error norm on the potential. Bottom: discrete L_2 error norm on the gradient of the potential.

In a terminal

```
./POST/smgr.py -g
code_saturne smgr -f ./smgr_fvca.xml -r -p --dest ../01_3D_ANISOTROPIC_DIFFUSION_RUNS/
```

You can repeat this procedure (edit the `smgr.py`, generate a new `smgr_fvca.xml` file and launch the computations and post-processings) to test and compare several variants of CDO schemes for instance

- `CDOVB.WBS`: `CDO-Vb` scheme with a *Whitney Barycentric Subdivision* (WBS) algorithm to build the discrete Hodge operator related to the diffusion term.

  C code

  ```c
  cs_equation_param_set(eqp, CS_EQKEY_SPACE_SCHEME, "cdo_vb");
  cs_equation_param_set(eqp, CS_EQKEY_HODGE_DIFF_ALGO, "wbs");
  ```

  Do not forget to remove the line setting the key `CS_EQKEY_HODGE_DIFF_COEF`.

- `CDOVCB`: a `CDO-VCb` scheme (the only choice in this case is the WBS algorithm)

  C code

  ```c
  cs_equation_param_set(eqp, CS_EQKEY_SPACE_SCHEME, "cdo_vcb");
  cs_equation_param_set(eqp, CS_EQKEY_HODGE_DIFF_ALGO, "wbs");
  ```

  Do not forget to remove the line setting the key `CS_EQKEY_HODGE_DIFF_COEF`.

- `CDOFB.SUSHI`: a `CDO-Fb` scheme with the SUSHI variant.

  C code

  ```c
  cs_equation_param_set(eqp, CS_EQKEY_SPACE_SCHEME, "cdo_fb");
  cs_equation_param_set(eqp, CS_EQKEY_HODGE_DIFF_ALGO, "cost");
  cs_equation_param_set(eqp, CS_EQKEY_HODGE_DIFF_COEF, "sushi");
  ```

Many other variants are available if one modifies the type of algorithm to build the discrete Hodge operator, the way to enforce the Dirichlet boundary conditions or the iterative solver for instance.

References

[1] Eymard R., Henry G., Herbin R., Hubert F., Klöfkorn R. and Manzini G. (2011) *3D Benchmark on Discretization Schemes for Anisotropic Diffusion Problems on General Grids*, FVCA 6 conference proceedings, Springer.