STEADY STOKES

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

June, 2023

---

1 Introduction

The aim of this benchmark was to compare different space discretizations on the resolution of a Stokes 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. The reading of the `01_ANISOTROPIC_DIFFUSION` tutorial may also be useful but is not mandatory.

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_02.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_02.tar.xz .
Singularity> unxz CDO_TUTORIAL_02.tar.xz
Singularity> tar -xvf CDO_TUTORIAL_02.tar
cd 02_3D_STEADY_STOKES
```

2 Definition of the test case

The computational domain is a unit cube denoted by Ω, an open bounded connected polyhedral subset of $\mathbb{R}^3$. $\partial\Omega = \overline{\Omega} \setminus \Omega$ denotes the boundary of the domain. The problem at stake is: Find $(\underline{u}, p) \in [H_1(\Omega)]^3 \times L_2(\Omega)$ s.t.

$$\left\{\begin{array}{rcll} -\mu\underline{\Delta}\left(\underline{u}\right) + \underline{\nabla}(p) &=& \underline{f} & \text{on } \Omega \\ \operatorname{div}\underline{u} &=& 0 & \text{on } \Omega \\ \int_\Omega p &=& 0 & \\ \underline{u} &=& \underline{u}_d & \text{on } \partial\Omega \end{array}\right. \tag{1}$$

where $\underline{u}$ is the velocity and p the pressure. For the sake of simplicity, the laminar viscosity $\mu = 1$. The boundary conditions are Dirichlet boundary conditions on the velocity on the whole boundary. The source

term $\underline{f}$ is defined to comply with the exact solutions $(\underline{u}_e, p_e)$ given by

$$\underline{u}_e(\underline{x}) := \begin{cases} \frac{1}{2}\sin(2\pi x)\cos(2\pi y)\cos(2\pi z) \\ \frac{1}{2}\cos(2\pi x)\sin(2\pi y)\cos(2\pi z) \\ -\cos(2\pi x)\cos(2\pi y)\sin(2\pi z) \end{cases} \tag{2}$$

and

$$p_e(\underline{x}) = \sin(2\pi x)\sin(2\pi y)\sin(2\pi z) \tag{3}$$

3 Mesh sequences

In this tutorial, we will consider several mesh sequences built for the FVCA6 benchmark [1]: 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).

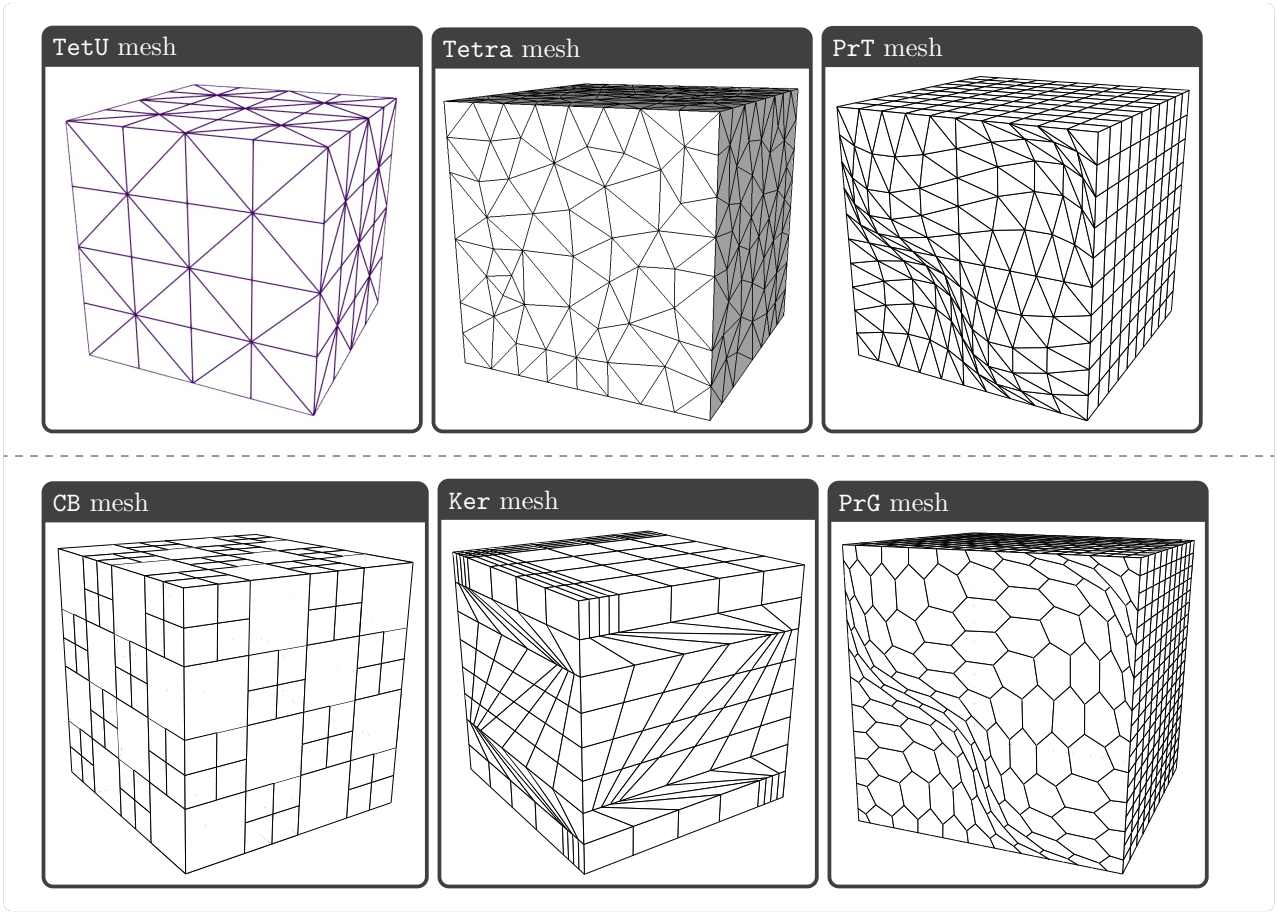

Figure 1: An overview of mesh sequences.

4 Error norms

`CDO-Fb` schemes are considered with a vector-valued velocity field located at faces and cells (defined as the mean-value over the geometric entity) and a scalar valued pressure field located at cells (defined as the mean-

value over a cell). C (resp. F) denotes the set of mesh cells (resp. faces) and $\#C$ (resp. $\#F$) represents the number of mesh cells (resp. faces). The numerical solution $\mathbf{u} \in \widehat{\underline{\mathcal{F}}} \times \underline{\mathcal{C}} \equiv \mathbb{R}^{3(\#F+\#C)}$ and $\mathbf{p} \in \mathcal{C} \equiv \mathbb{R}^{\#C}$.

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

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

  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. A variant at faces is also considered for the velocity

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

  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. $\underline{\mathbf{u}}_f$ is the velocity vector 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

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

  A variant at faces is also considered:

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

  For vector-valued quantities, the module of the component-wise difference is used.

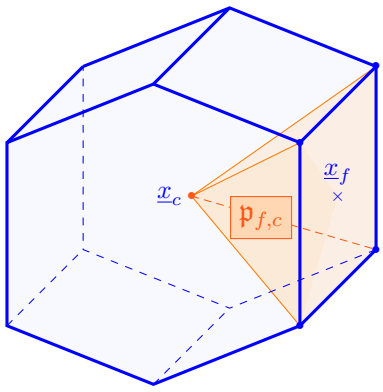

Figure 2: Example of a cell partition based on faces ($\mathfrak{p}_{f,c}$).

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

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

where E_*^* is one of the error norms previously introduced and `nsys(k)` is the number of unknowns in the linear system associated to the mesh `k` in a mesh sequence and $\mathbf{Y}_\mathtt{k}$ is the numerical solution for the mesh `k`. Since a *static condensation* is used, the size of the saddle-point system to solve is equal to $3\#F + \#C$.

5 First computation

5.0.1 Case settings

The case settings are all located in the directory `02_3D_STEADY_STOKES` (a code_saturne study). The first case `REFERENCE_SETTINGS` inside this study is ready to run. Before running the 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 Navier–Stokes is activated and set to a steady-state Stokes problem. This induces the creation of an equation called *momentum* along with two variables: the velocity and the pressure. Moreover, one requests to postprocess the divergence of the velocity field.

    C code

    ```
    cs_navsto_system_activate(domain->boundaries,
                              /* Model: */
                              CS_NAVSTO_MODEL_STOKES,
                              CS_NAVSTO_MODEL_STEADY,
                              /* Velocity-pressure coupling */
                              CS_NAVSTO_COUPLING_MONOLITHIC,
                              /* Postprocessing flag */
                              CS_NAVSTO_POST_VELOCITY_DIVERGENCE   |
                              0);
    ```

  - `cs_user_parameters()` where the default numerical options associated to the momentum equation can be overwritten. The function used to do this task is `cs_equation_param_set`. More options like the way to solve the saddle-point system arising from the `CDO-Fb` schemes can also be specified at this stage using `cs_navsto_param_set`. The solver used to invert the velocity block in case of strategy relying on the resolution (more or less accurate according to the kind of operation: solver or preconditionner) can also be specified at this stage.

    C code

    ```
    cs_navsto_param_set(nsp, CS_NSKEY_SLES_STRATEGY, "sgs_schur_gcr");
    cs_equation_param_set(mom_eqp, CS_EQKEY_ITSOL, "fcg");
    ```

    where `mom_eqp` and `nsp` are declared at the beginning of the function `cs_user_parameters()` as follows:

    C code

    ```
    cs_navsto_param_t  *nsp = cs_navsto_system_get_param();

    cs_equation_param_t  *mom_eqp = cs_equation_param_by_name("momentum");
    ```

  - `cs_user_finalize_setup()` where one details how
    - the Dirichlet boundary conditions are specified

      C code

      ```
      cs_equation_add_bc_by_analytic(mom_eqp,
                                     CS_PARAM_BC_DIRICHLET,
                                     "boundary_faces",   // zone name
                                     _get_velocity_sol,  // function
                                     NULL);              // input struct.
      ```

The boundary conditions are set on all boundary faces thanks to a function called *_get_velocity_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_velocity_sol* is available in the same user-defined source file.

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

C code

```
cs_navsto_param_t  *nsp = cs_navsto_system_get_param();

cs_navsto_add_source_term_by_analytic(nsp,
                                      "cells",          // zone name
                                      _get_source_term, // function
                                      NULL);            // input struct.
```

- 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.

5.1 Run

To run a first calculation with the default mesh (`CB4` mesh), write

In a terminal

```
cd SUMMER_SCHOOL/CDO/02_3D_STEADY_STOKES
code_saturne update -c REFERENCE_SETTINGS
cd REFERENCE_SETTINGS/
code_saturne run
```

5.2 Analysis

To visualize the results:

In a terminal

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

An example of postprocessing is illustrated in Figure 3. Besides the log files automatically generated by code_saturne, the `Stokes.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 Stokes.log (`CB4` mesh)

```
 -cvg- VelC.Err.Linf       5.322279e-02
 -cvg- VelC.Err.L2         4.521899e-02
...
 -cvg- PreC.Err.Linf       3.166802e-01
 -cvg- PreC.Err.L2         2.544240e-01
```

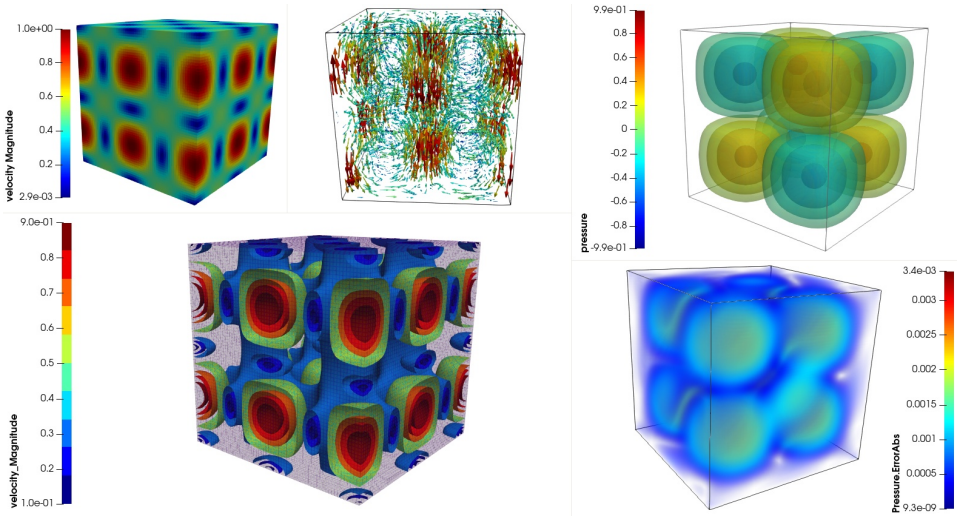

Figure 3: Example of postprocessing with paraview

6 First 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 `02_3D_STEADY_STOKES` write

In a terminal

```
./POST/smgr.py -g
code_saturne smgr -f smgr_stokes.xml -r -p --dest ../02_3D_STEADY_STOKES_RUNS
```

The first command generates the file *smgr_stokes.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 detail on the available options with the studymanager tool, write* `code_saturne smgr --help`*.*

Remark 2 (Parallel computing)**.** *To perform a parallel computation of each case, please add the* `--n-procs` *with a number of ranks in the previous command line (*2 *or* 3 *for instance according to your laptop).*

After all computations are completed, you could open the PDF files generated automatically in the directory `../02_3D_STEADY_STOKES_RUNS/POST/PDF_PLOTS` A PDF gathering all plots is also available in `../02_3D_STEADY_STOKES_RUNS/synthesis.pdf`. You should get similar results to the one depicted below in Figure 4.

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.

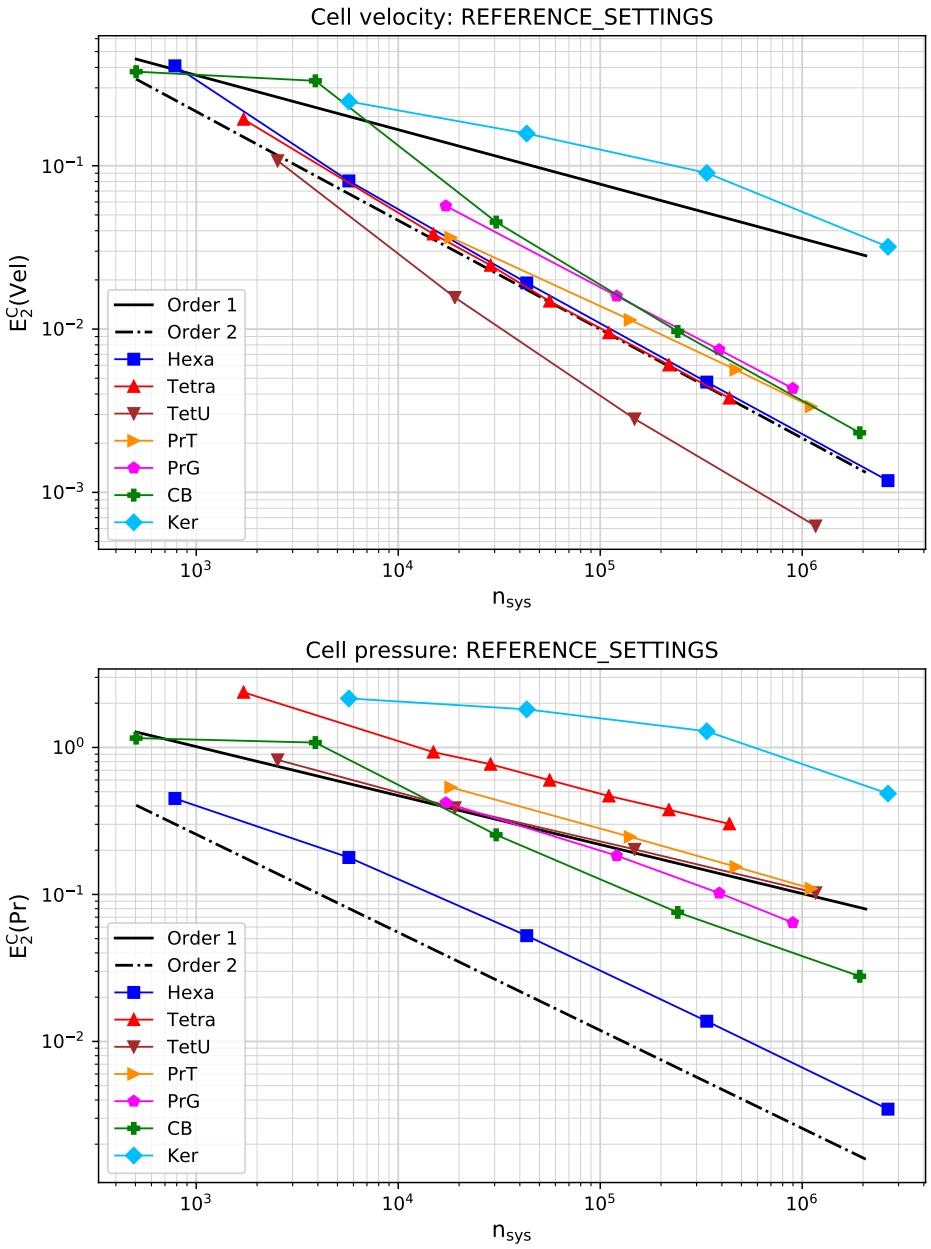

Figure 4: Convergence plots for the velocity (top) and the pressure (bottom) for the degrees of freedom defined at cells. A first order convergence rate is expected for the pressure and a second order convergence rate is expected for the velcoity.