SOFTWARE QUALITY ASSURANCE AND V&V PROCEDURE DURING THE DEVELOPMENT STAGES OF EDF CFD TOOL CODE_SATURNE: FOCUS ON VERIFICATION TESTS

J. Fontaine, M. Ferrand and E. Le Coupanec
EDF, R&D, Fluid Mechanics, Energy and Environment Dept.
6, Quai Watier, 78401 Chatou Cedex, France
jacques-j.fontaine@edf.fr

ABSTRACT

The present paper gives the state of the Software Quality Assurance (SQA) practices and the unitary Verification and Validation (V&V) procedures of the free Computational Fluid Dynamics (CFD) software *Code_Saturne*. The strict definitions of verification and validation are detailed. The autovnv framework embedded in *Code_Saturne* and dedicated to automate the V&V procedure is presented. A focus on the verification methodology is presented and particularly the computation of the time and space discretization errors. At the end, three verification test cases from *Code_Saturne* 4.0 autovnv are detailed.

KEYWORDS
Software Quality Assurance, V&V, CFD, *Code_Saturne*, MMS, MES

1. INTRODUCTION

The paper shows some technical aspects of the Verification and Validation (V&V) procedure of EDF open source CFD software *Code_Saturne* (Archambeau et al. [1], http://code-saturne.org).

Code_Saturne is an in-house EDF application designed to solve Navier-Stokes equations in the cases of 2D, 2D axi-symmetric and 3D flows. Its main module intends to simulate the flows which may be steady or unsteady, laminar or turbulent, incompressible or dilatable, isothermal or not. Scalars and turbulent fluctuations of scalars can be taken into account. The code includes specific modules, referred to as "specific physics", for the treatment of Lagrangian particle tracking, semi-transparent radiative transfer, gas combustion, pulverized coal combustion, fire modeling, magneto-hydrodynamics (Joule effect and electric arcs), compressible flows, severe accidents and groundwater flows. It can also be coupled with SYRTHES[1] code for conjugate heat transfers with or without radiative effects. *Code_Saturne* is a massively parallel application based on a hybrid MPI/OpenMP parallelization (P. Trespeuch et al. [2]). The code is highly scalable and demonstrates an efficient speed-up at very high core counts, e.g. from 32,768 to 65,536 cores. (Moulinec et al. [3]). *Code_Saturne* is a free software under the terms of the GNU General Public License as published by the Free Software Foundation; sources and documentation of the code are available on the dedicated web site.

Elementary verification and validation work is one of the main parts of the *Code_Saturne* project at EDF R&D. V&V procedures are built to measure correctness and accuracy of the mathematical and physical models. This work is a prerequisite for the integral validation (e.g. Barthet et al. [4]) and is crucial to enhance simulation credibility.

[1] http://syrthes.org

First, the paper presents the Software Quality Assurance practices during the development of *Code_Saturne* and focuses on technical points related to the verification procedure; then three examples of verification cases from *Code_Saturne* V&V database are detailed and the results obtained with *Code_Saturne* 4.0 are presented.

2. VERIFICATION AND VALIDATION DURING THE DEVELOPMENT PROCESS OF CODE_SATURNE: TERMINOLOGY AND METHODOLOGY

The definitions of verification and validation are based on those that can be found in Oberkampf et al. [5].
The verification means establishing that the mathematical models implemented are a faithful representation of the original theoretical models (code verification) and ensuring that the utilized mathematical models are adapted to the numerical problem (solution verification).
The validation (or model validation) is the process of determining whether a model is an accurate representation of a real-life configuration.
Code_Saturne V&V procedure aims at evaluating whether the code capabilities behave as expected and if the implemented mathematical and physical models are of good quality on a selected range of test cases. Therefore, academic as well as industrial test cases have been selected to cover our needs with the following requirements:

- represent at least once the different flow regimes (laminar, turbulent, stratified, …);
- depict at least once the specific physics of interest (coal combustion, conjugate heat transfer, atmospheric flow, …);
- evaluate discretization schemes and numerical options;
- test the code solution on classical industrial geometries (U-bend, T-Junction, …).

The evaluation is done by comparisons between the obtained numerical solution and an exact solution (MES[2]), a manufactured solution (MMS[3]), experimental results or in some particular cases, other numerical solutions (usually DNS[4] computations). Whenever it is possible, the level of error is quantified with a numerical criterion (discrete L_2 error for instance, see section 2.2.).

2.1. Software Quality Assurance (SQA) During the Development Process of *Code_Saturne*

The development process of *Code_Saturne* follows strict predefined rules. The main objective is to minimize programming errors during the implementation of the computational models. The next subsections give a few elements about the SQA activities in the development process of the code.

2.1.1. Co-development, Subversion code repository and documentation

In order to minimize misunderstanding during the code development or its use, several documentation guides are provided. The installation guide gives instructions for complex installations (on big clusters), the user manual lists the calculation options and the physical models, the theory guide describes numerical and physical modeling aspects, the developer guide gives coding rules and the source code DOXYGEN[5] documentation makes easier the documentation of function prototypes and the exploration of the code through a web browser.

[2] Method of Exact Solution.

[3] Method of Manufactured Solution.

[4] Direct Numerical Simulation.

[5] www.doxygen.org

Co-development and tracking of the source code modifications are simplified with the use of a Subversion code repository.
Only a short list of developers can add new code modifications on the development version (trunk) of *Code_Saturne*. The developments provided by other developers (for instance from the open source community) are added to the trunk version if the following criteria are fulfilled:

- the developers followed the rules detailed in the *Code_Saturne* developer guide;
- non-regression tests have been successful;
- DOXYGEN, user, and theory documentations have been updated;
- if a new model has been developed, at least one well documented test case is provided and included in the V&V procedure.

2.1.2. Code versioning policy

As for all software, different kinds of versions are released (see Figure 1). The main difference is between the fully validated, long-term support versions (x.0) and the intermediate versions (x.1, x.2, x.3, …).

The x.0 versions are released every 2 years and are maintained for 4 years. They have to go through the full V&V process and are mainly intended to be used by the engineering of EDF. The intermediate versions are released every 6 months and maintained for 1 year. They are provided to the users who want to try or validate the latest functionalities of the code. Patched versions may be released with .z extension to adapt the code portability or to solve unexpected bugs.

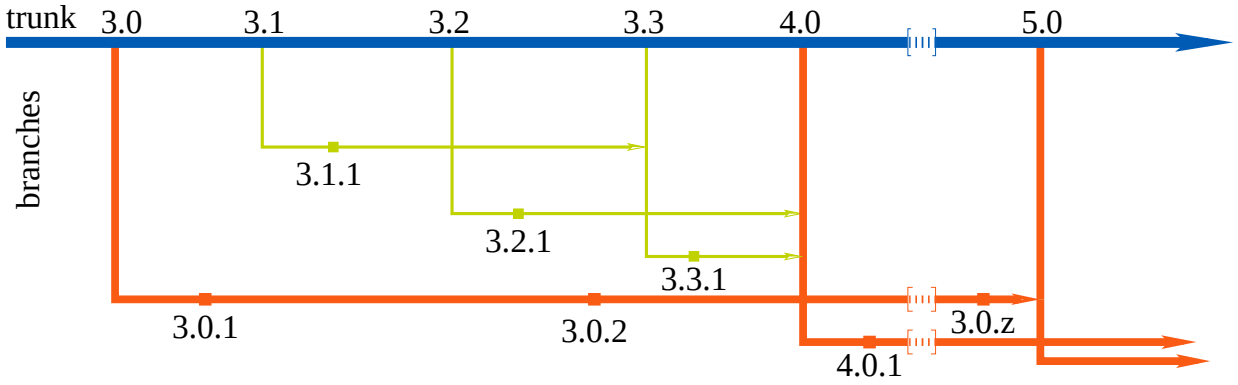


Figure 1. Versioning policy of *Code_Saturne*.
Trunk (or development) versions in blue, intermediate versions in green, long-term support versions in orange.

2.1.3. Autovnv tool, V&V Subversion repository and daily non-regression testing

Autovnv (for automatic validation and verification) is a tool developed as a part of *Code_Saturne*, providing a framework for the automation of the computations launch and the post-processing of results, as for example comparing 1D profiles from different computations to available experimental results. Furthermore, autovnv is able to bitwise compare a new result file to a previous one; this functionality allows performing non-regression verification tests when the code undergoes a refactoring.
The autovnv tool is non-intrusive: a standard setup is established and a XML file containing commands for each *Code_Saturne* test case is added to control automatic runs and the post-processing. Note that autovnv does not need a specific installation and the entire framework is embedded in the source code.

The reader willing to have a review of all the capabilities and parameters of the autovnv framework is invited to read the autovnv manual[6].
All the test cases of *Code_Saturne* V&V database have been setup according to the predefined autovnv format. Computations and post-processing are entirely repeatable. The V&V database is versioned with a Subversion repository allowing a better traceability and avoiding human errors in the setup and post-processing stages.
In addition to daily automatic compilations on different environments (different compilers and options, with/without MPI, with/without OpenMP, …), the autovnv allows performing daily "minimal" tests on all V&V test cases (only 10 outer iterations are run and the check is performed at the last one). If an error is detected in at least one test case, new modifications in the code repository are not allowed except for solving the encountered problem. When the bug is fixed, restriction on code modification is lifted.

At the time of the writing, about 250 validation cases and 560 verification cases are daily run.

2.2. Verification Methodology of the Numerical Schemes with MES or MMS

The present section introduces technical aspects utilized in the validation process of *Code_Saturne*. The formula to compute discretization errors are detailed and a methodology to obtain an estimator of the convergence ratio is proposed.
($\underline{U}_e$, P_e) is the exact solution (non equal to zero) of Navier-Stokes equations ($\underline{U}$ *and P* variables) obtained with MES or MMS. If a transport equation is introduced for a scalar quantity (Y variable), Y_e stands for the exact solution. ($\underline{U}_\Sigma$, P_Σ, Y_Σ) is the numerical solution computed by *Code_Saturne* with a given discretization Σ (in time and in space).
The discretization error $E_2^\Sigma(\underline{U})$ (L_2) of the velocity vector is defined as:

$$E^{\Sigma}{}_2(\underline{U}) = \sqrt{\frac{\sum_i \left\| \underline{U}_e(i) - \underline{U}_i \right\|_2^2 \left|\Omega_i\right|}{\sum_i \left\| \underline{U}_e(i) \right\|_2^2 \left|\Omega_i\right|}},$$

where $\left|\Omega_i\right|$ is the volume of a computational cell i, $\underline{U}_e(i)$ and $\underline{U}_i$ the values of the exact and numerical solutions in cell i, respectively. The discretization error on a scalar quantity Y is defined as:

$$E_2^{\Sigma}(Y) = \sqrt{\frac{\sum_i \left[Y_e(i) - Y_i \right]^2 \left|\Omega_i\right|}{\sum_i Y_e(i)^2 \left|\Omega_i\right|}}.$$

When the pressure field is defined to within a constant, the discretization error for the pressure is computed with a correction and is defined as:

$$E_2^{\Sigma}(P) = \sqrt{\frac{\sum_i \left[P_e(i) - P_e^{mean} + P^{mean} - P_i \right]^2 \left|\Omega_i\right|}{\sum_i P_e(i)^2 \left|\Omega_i\right|}},$$

[6] http://code-saturne.org/cms/documentation

where:

$$P_e^{mean} = \sum_i P_e(i)\left|\Omega_i\right| \frac{1}{\sum_i \left|\Omega_i\right|},$$

$$P^{mean} = \sum_i P_i \left|\Omega_i\right| \frac{1}{\sum_i \left|\Omega_i\right|}.$$

The discretization error is calculated at a given physical time (unsteady problems) or after the convergence in time (steady problems). The two cases are detailed in the next subsections.

2.2.1. Steady state Navier-Stokes problems

In *Code_Saturne*, a steady numerical solution may be obtained after the time convergence of an unsteady algorithm (or pseudo-steady). The problem is iteratively solved to get the steady solution, hence a stopping criterion is needed to interrupt the computation. A steady problem is supposed converged in time when for each variable X of ($\underline{U}$, P, Y), the following criterion is verified:

$$D(X) = \sqrt{\frac{\sum_i \Delta t_i^{-2} \left\| X_i^t - X_i^{t-1} \right\|_2^2 \left|\Omega_i\right|}{\sum_i \left\| X_i \right\|_2^2 \left|\Omega_i\right|}} < \varepsilon,$$

where X^t_i is the value of the discrete field X (can be a vector field) in cell i at the time step t (or pseudo time step), and Δt_i is the value of the local time step of cell i.

The value which one has to take for ε is not straightforward and depends on the considered problem and the chosen numerical options.

When the time convergence is obtained and the space refinement tends to 0, one wants to verify that:

- the numerical solution converges towards the analytical solution;
- the local numerical convergence ratio converges towards the theoretical convergence ratio.

One defines the convergence ratio for the variable X between two homogeneous space refinement levels h and $a*h$ (where a is a strictly positive real) as:

$$\hat{p}^{a,h}(X) = \frac{\ln\left[E_2^{a*h}(X) / E_2^h(X) \right]}{\ln\left[a \right]}.$$

The « ^ » distinguishes the local numerical convergence ratio from the theoretical one. The goal is to verify that:

$$\hat{p}^{a,h}(X) \xrightarrow[h\to 0]{} p(X),$$

where $p(X)$ is the theoretical convergence ratio.

First, it is required to prove that the local numerical convergence ratio is in the asymptotic convergence zone. For this purpose, the local convergence ratio has to be calculated on at least 3 refinement levels. One has also to verify that the local convergence ratio is constant. Finally, the converged local convergence ratio can be compared to the theoretical one.

2.2.2. Unsteady Navier-Stokes problems

In case of unsteady problems, the discretization error is computed at a given physical time τ. The discretization error includes time and space discretization errors. The global convergence ratio could be computed using the previous methodology. In this case, the space-time discretization level Σ of parameters ($h, \Delta t$) tends to 0 homogeneously.

The main disadvantage of this methodology is that it is impossible to distinguish the time error from the space one. In order to estimate space and time errors separately, one proposes the methodology used in the verification report of Fire Dynamics Simulator (McDermot et al. [6]) and Thétis (Thétis [7]). First, the discretization error is computed with a fixed time step (the smallest one) at different space discretization levels. One assumes the existence of a zone where the error coming from space discretization becomes negligible compared to the one due to time discretization (space saturation zone). Finally, the time convergence ratio is calculated in the space saturation zone. One can proceed in a similar way to estimate space convergence ratio (in the time saturation zone).

3. EXAMPLES OF CODE_SATURNE VERIFICATION CASES

3.1. Verification of Space Discretization Operators

3.1.1. Basic verification of the diffusion operator

The elliptic problem being considered here is the heterogeneous anisotropic diffusion. The main objective is to verify *Code_Saturne* solvers for heterogeneous diffusion problems (Ferrand et al. [8]). As several elliptic problems are solved in the code, the present test is applied to various steps in the core solver (diffusion part of a scalar transport equation with Generalized Gradient Diffusion Hypothesis (GGDH) (Dehoux et al. [9]), projection step of the predictor-corrector Navier-Stokes solver in presence of pressure losses, ...). The presented benchmark case is taken from FVCA-6 conference (Eymard et al. [10]).
The general Poisson equation which is studied reads:

$$\begin{cases} -\mathrm{div}\left(\underline{\underline{K}} \cdot \underline{\nabla} Y\right) = f & \text{on } \Omega \\ Y = Y_e & \text{on } \partial\Omega \end{cases}$$

where $\Omega = [0,1]^3$, $\partial\Omega$ is its boundary, Y a scalar field defined on the domain Ω,, $\underline{\underline{K}}$ the diffusivity tensor field, assumed to be symmetric positive definite and f a source term.
MMS is utilized in the three following test cases. If Y_e stands for the imposed analytical solution, the source term is given by $f = -\mathrm{div}\left(\underline{\underline{K}}.\underline{\nabla} Y_e\right)$ and the boundary condition by the value of this analytical solution at the boundary. One gives in the following the analytical expressions of $\underline{\underline{K}}$ and Y_d in the different cases.
The first analytical test case (case 1) is defined with:

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

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

The second analytical test case (case 2) is defined as follows:

$$\underline{\underline{K}} = \begin{pmatrix} 1+y^2+z^2 & -xy & -xz \\ -xy & 1+x^2+z^2 & -yz \\ -xz & -yz & 1+x^2+y^2 \end{pmatrix}$$

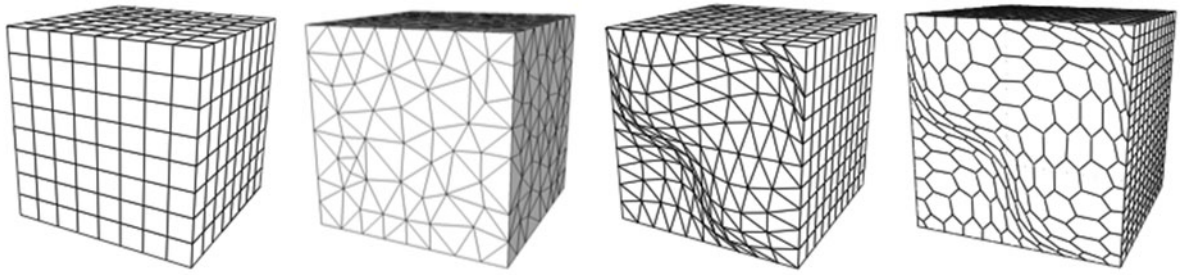
$$Y_e = x^3y^2z + x\sin\left(2\pi xy\right)\sin\left(2\pi xz\right)\sin\left(2\pi z\right)$$

This case is representative of the applications covered by the algorithm solving the diffusive term of a scalar transport equation with a heterogeneous and anisotropic tensor $\underline{\underline{K}}$, but not discontinuous.

The last analytical test case (case 3) presented here is:

$$\underline{\underline{K}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1000 \end{pmatrix}, Y_e\left(\underline{x}\right) = \sin\left(2\pi x\right)\sin\left(2\pi y\right)\sin\left(2\pi z\right).$$

All the these cases are run on hexahedral meshes (HEX), tetrahedral meshes (TET), prism meshes with triangle bases (BLS) and prism meshes with general bases (DBLS) of the FVCA-6 benchmark mesh database (see Eymard et al. [10]). Figure 2 gives an overview of the meshes.



(a) HEX mesh. (b) TET mesh. (c) BLS mesh. (d) DBLS mesh.

Figure 2. An overview of mesh types.

Figures 3 to 5 show the results of the discrete L^2 errors for the variable *Y* and its gradient, with all the meshes and for the three analytical test cases. As expected, a second order convergence rate on the variable and a first order one on its gradient are obtained, respectively. One may notice that the results do not exhibit perfectly a second order with BLS on case 3. This is due to high non-orthogonal faces induced by the stretching of the tetrahedral computational cells. A hyper-convergence is also observed for the gradient of the scalar quantity with HEX and DBLS meshes on all the cases, this is probably due to the good properties induced by hexahedral and polyhedral elements (Moulinec et al. [11]).

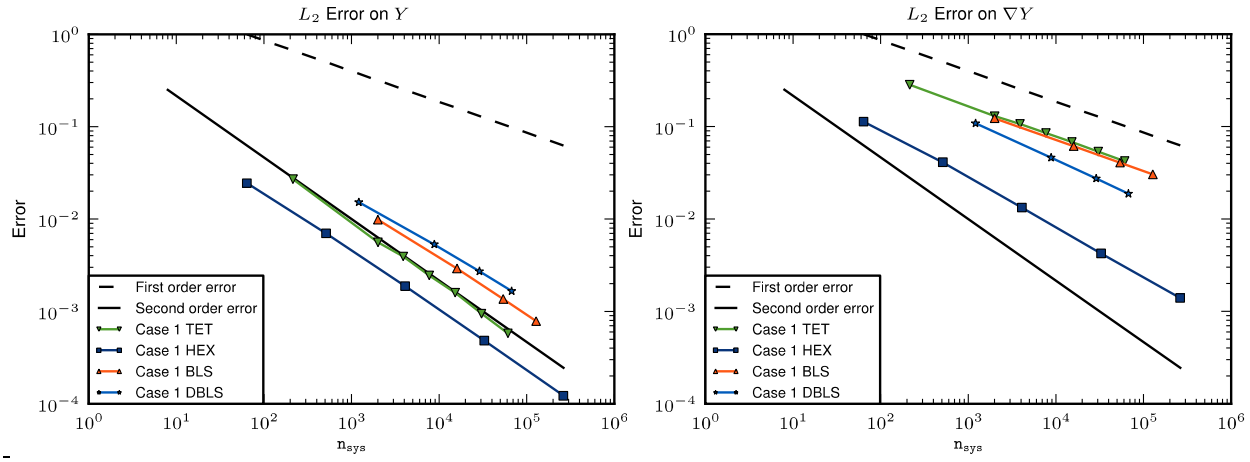


Figure 3. L^2 discrete error, as a function of the total number of unknowns (number of cells), in logarithmic scale for all the mesh series for case 1.

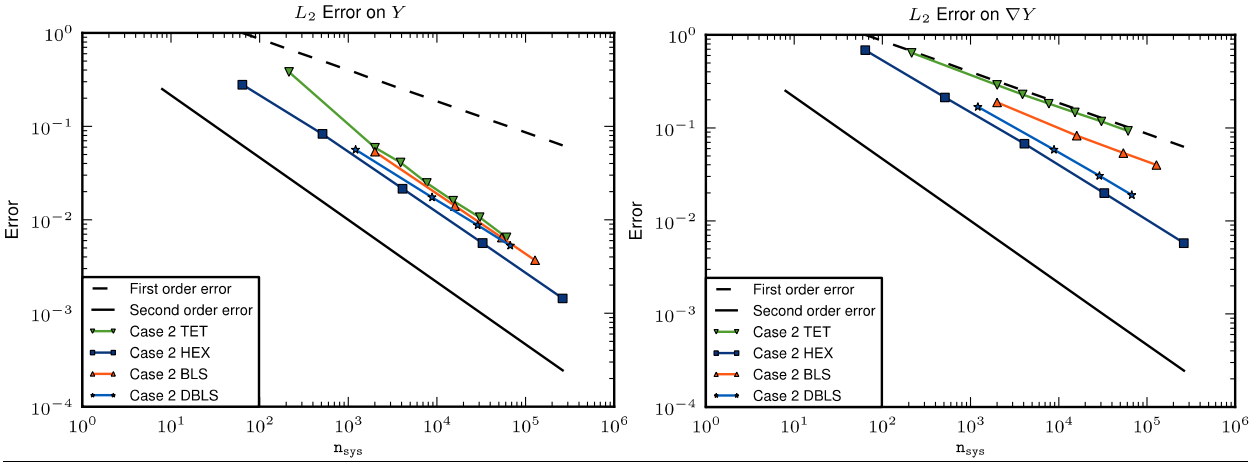


Figure 4. L^2 discrete error, as a function of the total number of unknowns (number of cells), in logarithmic scale for all the mesh series for case 2.

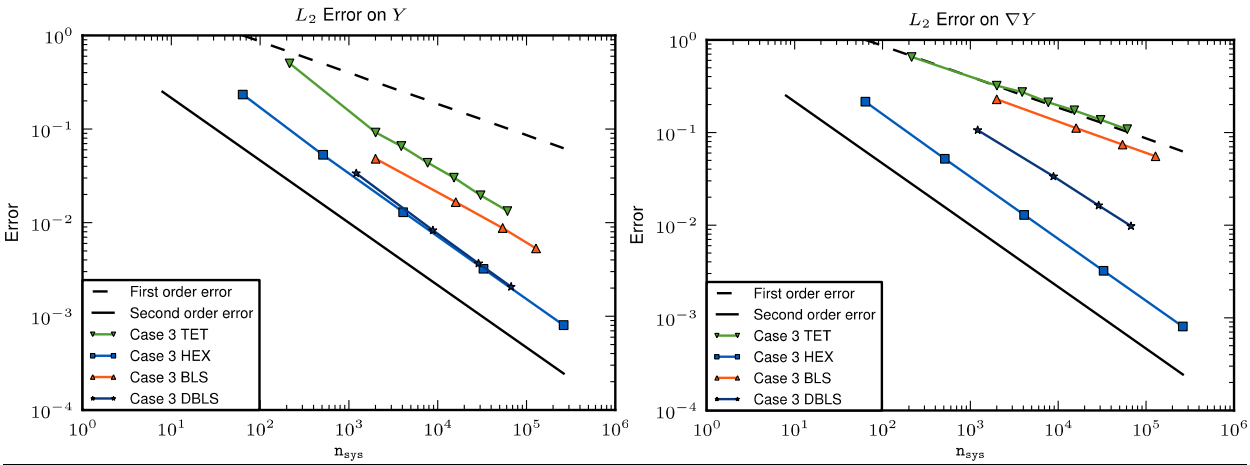


Figure 5. L^2 discrete error, as a function of the total number of unknowns (number of cells), in logarithmic scale for all the mesh series for case 3.

3.1.2 Verification of steady state flow with a variable density

The Navier-Stokes problem being considered in the present section is 2D, steady and with a variable density. The temperature equation is also solved. This verification case is from Shunn et al. [12]. The main objective is to verify in *Code_Saturne* the use of a variable density and the discretization in space (convection and diffusion operators). MMS is again applied.

All the variables underscore *e* stand for the explicit analytical solutions used in the MMS.
A zero divergence mass flux is chosen:

$$\rho_e \underline{U}_e(x,y) = \begin{bmatrix} [1-\cos(2\pi x)]\sin(2\pi y) \\ \sin(2\pi x)[\cos(2\pi y)-1] \end{bmatrix}$$

The pressure and scalar fields are given by the following equations:

$$\begin{cases} P_e(x,y) = \sin(\pi x + \pi/2)\sin(\pi y + \pi/2) \\ T_e(x,y) = \frac{1}{2}(\sin(2\pi x)\sin(2\pi y)+1) \end{cases}$$

Finally, the following equation of state is given for the density:

$$\rho_e = \left(\frac{T_e}{\rho_1} + \frac{1-T_e}{\rho_0} \right)^{-1}$$

where ρ_0 and ρ_1 are the extreme densities of the fluid at T_e = 0 and 1, respectively.
The different source terms are then deduced from the governing equations (in the following, the unsteady term vanishes):

$$S_U = \rho_e \frac{\partial U_e}{\partial t} + \rho U_e \frac{\partial U_e}{\partial x} + \rho V_e \frac{\partial U_e}{\partial y} + \frac{\partial P_e}{\partial x} - \mu\left(\frac{\partial^2 U_e}{\partial x^2} + \frac{\partial^2 U_e}{\partial y^2} \right) - \frac{1}{3}\left(\frac{\partial^2 U_e}{\partial x^2} + \frac{\partial^2 V_e}{\partial x \partial y} \right)$$

$$S_V = \rho_e \frac{\partial V_e}{\partial t} + \rho U_e \frac{\partial V_e}{\partial x} + \rho V_e \frac{\partial V_e}{\partial y} + \frac{\partial P_e}{\partial x} - \mu\left(\frac{\partial^2 V_e}{\partial x^2} + \frac{\partial^2 V_e}{\partial y^2} \right) - \frac{1}{3}\left(\frac{\partial^2 V_e}{\partial x^2} + \frac{\partial^2 U_e}{\partial x \partial y} \right)$$

$$S_T = \rho_e \frac{\partial T_e}{\partial t} + \rho U_e \frac{\partial T_e}{\partial x} + \rho V_e \frac{\partial T_e}{\partial y} - K\left(\frac{\partial^2 T_e}{\partial x^2} + \frac{\partial^2 T_e}{\partial y^2} \right)$$

The domain is a square perpendicular to the z-direction. Its dimensions are 1 x 1.
Several meshes have been used to check the convergence. The number of elements varies from 32 x 32 to 256 x 256, with either hexahedral or triangular prism cells. The 32 x 32 hexahedral mesh is shown in Figure 6, along with the 32 x 32 prismatic one.

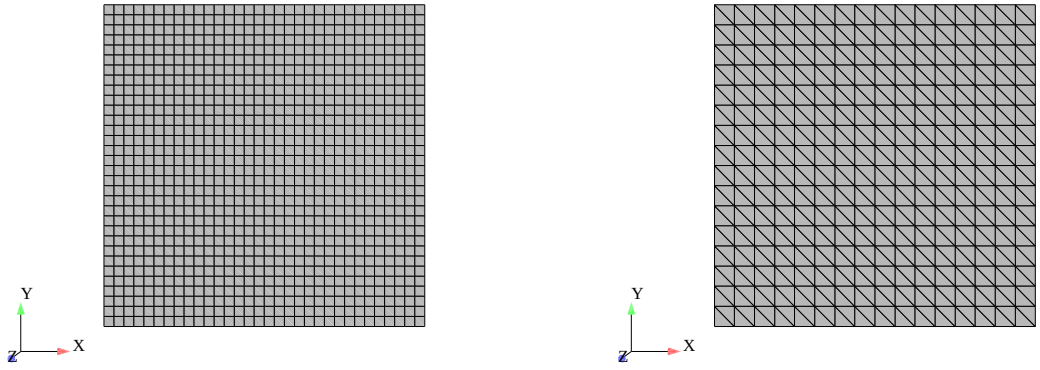


Figure 6. 32 x 32 meshes, with either hexahedral (left) or prismatic (right) elements.

The flow is considered to be steady and laminar at a Reynolds number equal to 1. The Prandtl number is set to 0.71 and the Rayleigh number to 5000. The density ratio $\omega = \frac{\rho_1}{\rho_0}$ is equal to 0.1, with $\rho_0 = 1$. Gravity is not taken into account.

Three different convection schemes are tested: a pure upwind scheme, a pure centered scheme (without slope test) and a SOLU (Second order Linear Upwind) scheme. The steady state is obtained after convergence of the unsteady solver. The time convergence is reached when $D < 10^{-10}$ for all the variables. Expected convergence ratios are obtained, first order with the upwind scheme (Figure 7) and second order with the centered and the SOLU discretization (Figure 8).

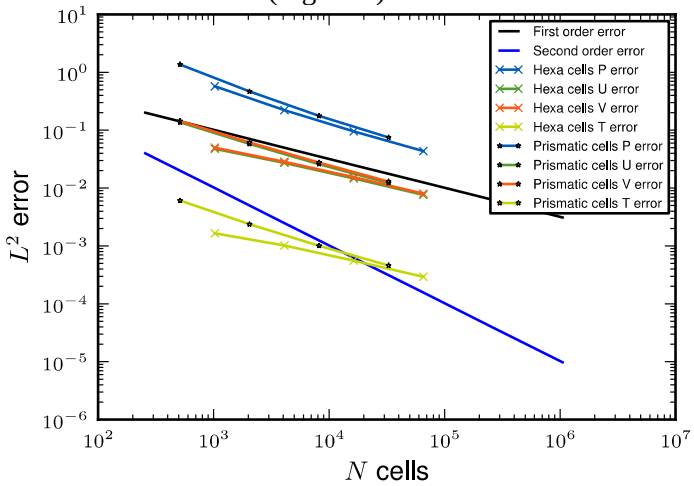


Figure 7. L^2 discrete error, as a function of the total number of unknowns (number of cells), in logarithmic scale for all the mesh series with upwind scheme for convection term.

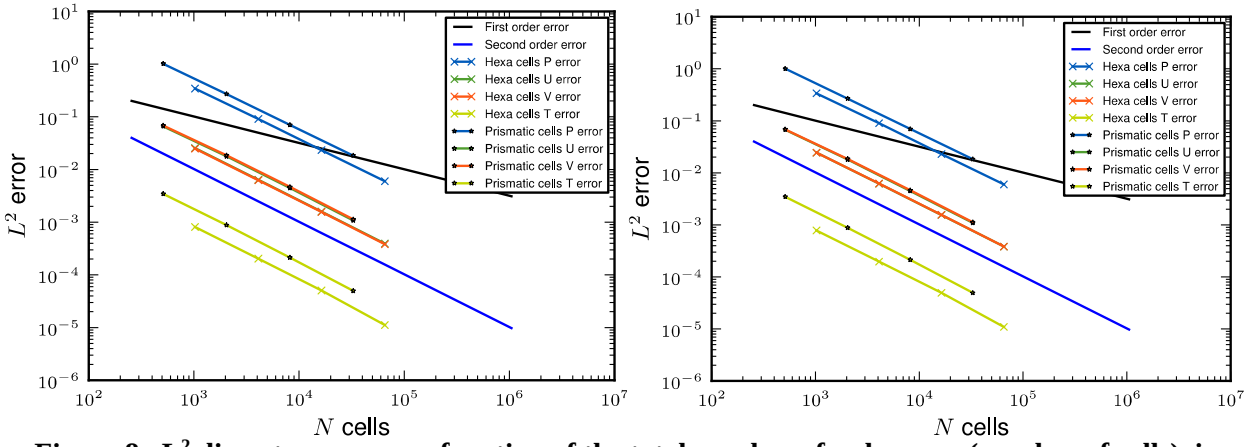


Figure 8. L^2 discrete error, as a function of the total number of unknowns (number of cells), in logarithmic scale for all the mesh series with centered (left) and SOLU (right) schemes for convection term.

3.2. Verification of Algorithms in Time

Taylor-Green vortices (see for example [13]) represent decaying vortices over time and exhibit an analytical solution which is appropriate to perform a MES. This flow has exact theoretical solutions considering 2D incompressible Navier-Stokes equations. The main objective is to verify in *Code_Saturne* the velocity/pressure coupling algorithms and the discretization in time.

A particular solution of the Navier-Stokes problem on the domain [0,1] x [0,1] with a molecular kinematic viscosity equal to 1/*Re* and periodic boundary conditions in the two space directions is given by:

$$\begin{aligned} U_e(x,y,t) &= \left(\sin(x)\cos(y)\right)\exp(-\frac{2}{Re}t) \\ V_e(x,y,t) &= -\left(\cos(x)\sin(y)\right)\exp(-\frac{2}{Re}t) \\ P_e(x,y,t) &= \frac{\rho}{4}\left(\cos(2x)+\cos(2y)\right)\exp(-\frac{4}{Re}t) \end{aligned}$$

For the Cartesian grid used in the present test case, initial velocity and pressure at cell i follow:

$$\begin{aligned} U_e(i) &= \frac{1}{|\Omega_i|}\int_{\Omega_i}\sin(x)\cos(y)\mathrm{d}\Omega \\ &= \frac{\Delta z}{|\Omega_i|}\left(\cos\left(x_i-\frac{\Delta x}{2}\right)-\cos\left(x_i+\frac{\Delta x}{2}\right)\right)\left(\sin\left(y_i+\frac{\Delta y}{2}\right)-\sin\left(y_i-\frac{\Delta y}{2}\right)\right) \\ V_e(i) &= \frac{1}{|\Omega_i|}\int_{\Omega_i}-\cos(x)\sin(y)\mathrm{d}\Omega \\ &= \frac{\Delta z}{|\Omega_i|}\left(\sin\left(x_i+\frac{\Delta x}{2}\right)-\sin\left(x_i-\frac{\Delta x}{2}\right)\right)\left(\cos\left(y_i+\frac{\Delta y}{2}\right)-\cos\left(y_i-\frac{\Delta y}{2}\right)\right) \\ P_e(i) &= \frac{1}{|\Omega_i|}\int_{\Omega_i}\frac{\rho}{4}\left(\cos(2x)+\cos(2y)\right)\mathrm{d}\Omega \\ &= \frac{\Delta z}{|\Omega_i|}\frac{\rho}{2\times 4}\left(\left(\sin\left(2x_i+\Delta x\right)-\sin\left(2x_i-\Delta x\right)\right)+\left(\sin\left(2y_i+\Delta y\right)-\sin\left(2y_i-\Delta y\right)\right)\right) \end{aligned}$$

with Ω_i is the cell volume and Δx (resp. Δy) the cell size along x-axis (resp. y-axis).
Moreover, initial velocity and pressure at a face f follow:

$$\begin{aligned} (\rho U)_e(f) &= \rho\frac{1}{|S_f|}\int_{S_f}\sin(x)\cos(y)\mathrm{d}S \\ &= \rho\frac{\Delta z}{|S_f|}\sin(x_f)\left(\sin(y_f+\frac{\Delta y}{2})-\sin(y_f-\frac{\Delta y}{2})\right) \\ (\rho V)_e(f) &= \rho\frac{1}{|S_f|}\int_{S_f}-\cos(x)\sin(y)\mathrm{d}S \\ &= \rho\frac{\Delta z}{|S_f|}\sin(y_f)\left(\sin(x_f+\frac{\Delta x}{2})-\sin(x_f-\frac{\Delta x}{2})\right) \end{aligned}$$

with S_f being the face area and Δx (resp. Δy) the face size along x-axis (resp. y-axis).
The velocity and pressure are then given by the following analytical formulas:

$$U_e(x,y,t) = U_e(x,y,0)\exp(-\frac{2}{Re}t)$$

$$V_e(x,y,t) = V_e(x,y,0)\exp(-\frac{2}{Re}t)$$

$$P_e(x,y,t) = P_e(x,y,0)\exp(-\frac{4}{Re}t)$$

The fluid domain is 2π x 2π. The mesh is hexahedral and the number of cells is 1024 x 1024. Other tests showed that this mesh is enough to be in the space saturation zone. Periodic conditions are imposed at the boundary of the domain in x and y directions, respectively. Density is set to 1 and the Reynolds number is equal to $10/3$. The time step is taken between $\Delta t = 2^{-5}$ and $\Delta t = 2$. Spatial discretization is centered without a slope test and Rhie and Chow interpolation is disabled. Convergence is tested with implicit Euler, double backward implicit Euler and Crank-Nicolson time schemes. Inner subiterations are also introduced ("nterup" in Figure 9).

In the following, the errors are considered at time *t = 4*, which corresponds roughly to a decrease of 90% of the velocity magnitude.

Taylor-Green vortices exhibit a particular situation where convection and pressure gradient terms are in balance. Hence, time evolution of velocity follows:

$$\frac{\partial \underline{U}}{\partial t} = \frac{1}{Re}\Delta \underline{U} = -\frac{2}{Re}\underline{U}$$

Hereafter, the theoretical evolution of $\underline{U}$ assuming perfect solving of laplacian on a cartesian grid and exact respect of equilibrium between convection and pressure gradient is developed for the different time schemes. These results give the theoretical convergence of the different time schemes when error in time dominates.

Implicit Euler:

$$\frac{\underline{U}^{n+1} - \underline{U}^{n}}{\Delta t} = \frac{1}{Re}\Delta \underline{U}^{n+1}$$

$$\underline{U}^{n} = \left(\frac{1}{1+\frac{2}{Re}\Delta t}\right)^{n}\underline{U}^{0}$$

Second order backward differentiation (double backward Euler):

$$\frac{\underline{U}^{n+1} - \frac{4}{3}\underline{U}^{n} + \frac{1}{3}\underline{U}^{n-1}}{\Delta t} = \frac{2}{3}\frac{1}{Re}\Delta \underline{U}^{n+1}$$

$$\underline{U}^{n+1} = \frac{4\underline{U}^{n} - \underline{U}^{n-1}}{3+\frac{4}{Re}\Delta t}$$

Second order Crank-Nicolson:

$$\frac{\underline{U}^{n+1} - \underline{U}^{n}}{\Delta t} = \frac{1}{2Re}\left(\Delta \underline{U}^{n+1} + \Delta \underline{U}^{n}\right)$$

$$\underline{U}^{n} = \left(\frac{1-\frac{\Delta t}{Re}}{1+\frac{\Delta t}{Re}}\right)^{n}\underline{U}^{0}$$

Figure 9 shows the results obtained with the different time schemes and with and without inner iterations on pressure –velocity coupling. It is clear that the expected orders are obtained with the Euler time stepping and Crank-Nicolson approaches while using the inner iterations for pressure/velocity coupling.

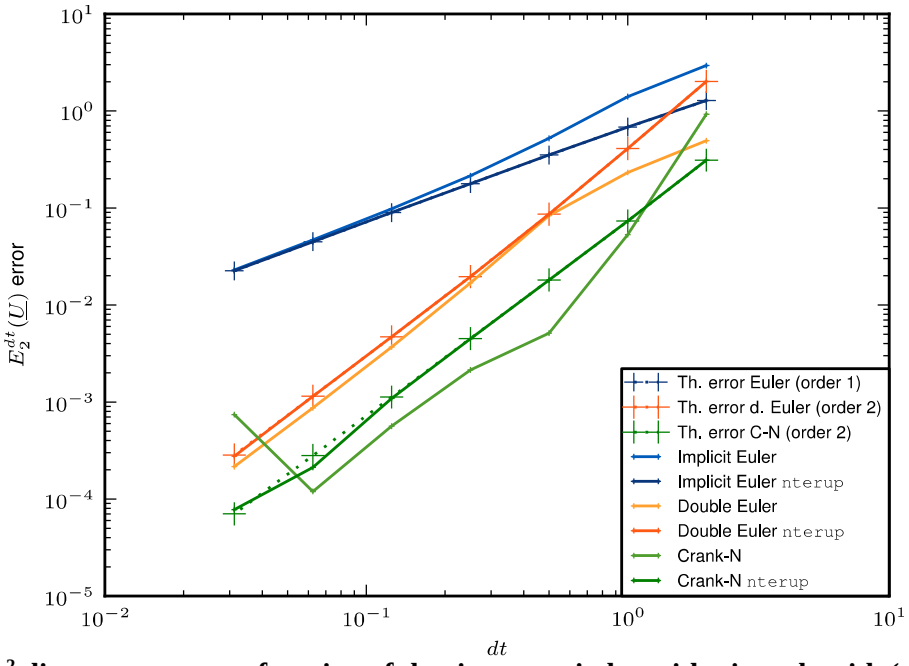


Figure 9. L^2 discrete error as a function of the time step, in logarithmic scale with (nterup) and without inner iterations on Navier-Stokes system.

4. CONCLUSIONS

An overview of Software Quality Assurance activities for the development process of *Code_Saturne* has been presented. Several technical aspects of *Code_Saturne* verification procedure such as the definition of the discretization errors, the computation of convergence ratios, the mesh topologies used in the tests and some test-cases based on MES or MMS have been described. The results obtained on this verification cases show that the numerical models implemented in *Code_Saturne* 4.0 are a faithful representation of mathematical models theory.

ACKNOWLEDGMENTS

The authors would like to thank Y. Fournier, T. Moy, B. Sapa, F. Nmira and R. Camy for their contribution in the V&V process and S. Benhamadouche for the careful proofreading of the present paper.

REFERENCES

1. F. Archambau, N. Mehitoua and M. Sakiz, “Code Saturne: A finite volume Code for the computation of turbulent incompressible flows,” *International Journal on Finite Volume* (2004).
2. P. Trespeuch, Y. Fournier, C. Evangelinos, P. Vezolle, “Mesh Renumbering Methods and Performance with OpenMP/MPI in Code Saturne,” *Proceedings of the Fourth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, Civil-Comp Press, Stirlingshire, UK, Paper 30, (2015).
3. C. Moulinec, A.G. Sunderland, P. Kabelikova, A. Ronovsky, V. Vondrak, A. Turk, C. Aykanat and C. Theodosiou, “Optimisation of Code_Saturne for Petascale Simulations,” *Partnership for Advanced Computing in Europe*, (2014).
4. A. Barthet, B. Gaudron and D. Alvarez, “Code_Saturne integral validation on a ROCOM test,” *The 15th Internationnal Topical Meeting on Nuclear Thermal – Hydraulics, NURETH-15*, Pisa, Italy, May 12-17, (2013).

5. W.-L. Oberkampf, and C.-J. Roy. “Verification and Validation in Scientific Computing,” *Cambridge University Press*, (2010).
6. R. McDermott, K. McGrattan, S. Hostikka, and J. Floyd. “Fire Dynamics Simulator (Version 5) Technical Reference Guide Volume 2: Verification,” NIST, (2010).
7. Thétis, “Cahier de Validation,” http://thetis.enscbp.fr, (2010).
8. M. Ferrand, J. Fontaine and O. Angelini, “An Anisotropic Diffusion Finite Volume Algorithm Using a Small Stencil”, *Finite Volumes for Complex Applications VII-Elliptic, Parabolic and Hyperbolic Problems*, pp. 577-585 (2014).
9. F. Dehoux, Y. Lecocq, S. Benhamadouche, R. Manceau and L.-E. Brizzi, “Algebraic Modeling of the Turbulent Heat Fluxes Using the Elliptic Blending Approach - Application to Forced and Mixed Convection Regimes,” *Flow, turbulence and combustion*, **88**, pp. 77-100 (2012).
10. R. Eymard, G. Henry, R. Herbin, F. Hubert, R. Klöfkorn, G. Manzini, “3D Benchmark on Discretization Schemes for Anisotropic Diffusion Problems on General Grids,” *FVCA 6 conference proceedings*, Springer, (2011).
11. C. Moulinec, S. Benhamadouche, D. Laurence, M. Perić. « LES in a U-Bend Pipe Meshed by Polyhedral Cells », *Engineering Turbulence Modelling and Experiments 6*, pages 237-246, Sardinia, 23-25 May (2005).
12. L. Shunn, F. Ham and P. Moin, “Verification of variable-density flow solvers using manufactured solutions,” *Journal of Computational Physics*, **231**(9), pp. 3801–3827, (2012).
13. S. Benhamadouche, “Large-Eddy Simulation with the unstructured collocated arrangement”, PhD thesis, The University of Manchester (2006).