Chair of Scientific Computing
TUM School of Computation, Information and Technology
Technical University of Munich

Time stepping review of open-source solvers

Guided research

Marc Amorós ✉

✉ marc.amoros@tum.de

Advisor: M.Sc. Benjamin Rodenberg
Supervisor: Prof. Dr. Hans-Joachim Bungartz

1 Introduction

- Introduction to coupling simulations (FSI) and to the preCICE library.
- Some brief motivation of performing a convergence study on the known solvers.
- Talk somehow about higer timestepping schemes, and why/when is good to use them. Should you use a higher order timestepping scheme if it doesn't give good results? (No bc it is slower)
- Say why we chose this two open source solvers.
- Explain difference between preCICEv2 and preCICEv3.

2 OpenFOAM

- Small introduction to OpenFOAM.
- Explain time stepping schemes available in the solver, and their orders. Mainly talk about Euler method, and Crank Nikolson, as they are the two cases we used.
- Talk about the script created to automatize the procedure of running these simulations.

OpenFOAM is an open-source computational fluid dynamics (CFD) software package widely used for simulating and analyzing complex fluid flow problems. Its solver modules employ finite volume methods to numerically solve the Navier-Stokes equations, making it a versatile tool for simulating fluid dynamics in various engineering and scientific applications. In this solver we can find various time stepping schemes, and we focused in two for our analysis. The first one is the Euler implicit scheme, as is usually the default one. Given the following partial differential equation:

$$\frac{\partial u}{\partial t} = F(u, x, t) \tag{1}$$

The Euler implicit scheme would discretize it as follows:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = {F_i}^{n+1}(u, x, t) \tag{2}$$

This is a first order method, that is quite stable, reason why it is usually the default choice. For the purpouse of this study, we also chose a second order scheme, this being the Crank Nikolson method [1]. This scheme is a combination of an explicit and an implicit Euler step, leading to a second order convergence in time. This method would discretize the previous PDE as:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2}\left[{F_i}^n(u, x, t) + {F_i}^{n+1}(u, x, t)\right] \tag{3}$$

OpenFOAM uses a sligtly different version of this method, by introducing a blending coefficient θ between the Euler implicit method and the Crank Nikolson method. If $\theta = 0$ then we obtain the implicit Euler method, and if $\theta = 1$ then it's Crank Nikolson. For stability, the value $\theta = 0.9$ is recomended in their documentation.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\theta}{2} {F_i}^n(u, x, t) + \left(1 - \frac{\theta}{2}\right) {F_i}^{n+1}(u, x, t) \tag{4}$$

TODO:(include this?) Diffusion example of Crank Nikolson.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{2\,(\Delta x)^2} + \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{2\,(\Delta x)^2} \tag{5}$$

2.1 Solver parameters

- Talk about the important parameters in the configuration, to obtain accurately enough results, as those where quite time consuming to find. For example, foamToVTK is not accurate enough, and was misleading at the beginning. Also mention the solver used.

2.2 Case study: Taylor–Green vortex

To study the convergence behaviour of the OpenFOAM, we focused on the Taylor-Green vortex [2, 3], a standard setup in CFD for the validation of fluid flow solvers, given that an analytical solution of the case is known. In a 2D, this solution can be obtained by the formulas:

$$u(x, y, t) = -\cos(x)\sin(y)e^{-2\nu t} \tag{6}$$

$$v(x, y, t) = \sin(x)\cos(y)e^{-2\nu t} \tag{7}$$

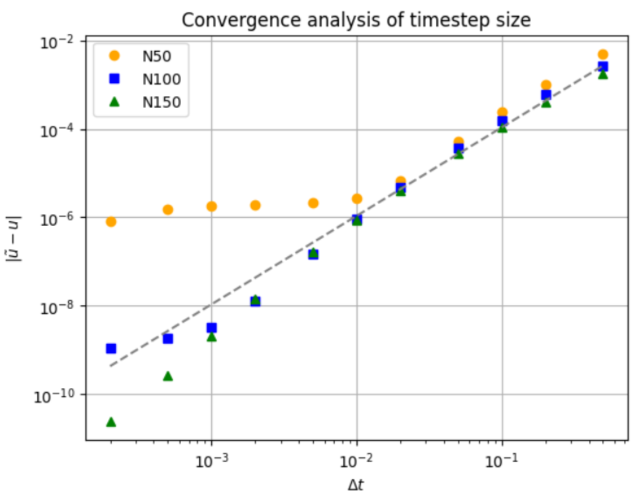
$$p(x, y, t) = -\frac{1}{4}\left[\cos(2x) + \sin(2y)\right] e^{-2\nu t} \tag{8}$$

where u and v are the horizontal and vertical velocities respectively, p is the pressure and ν is the viscosity of the fluid. This solution holds for a square domain of size 2π. We implemented a program that computes the initial velocity for this setup, and writes it into the OpenFOAM configuration. We also wrote a script to automatize the configuration and execution of different setups with varying parameters, so we can perform several experiments automatically. To observe the behaviour of the error, we did several executions of our setup case, fixing all the parameters (grid size, initial velocity, solver tolerances etc.) and changing the time-step size. On our analysis, we mainly focused on the velocity profile.

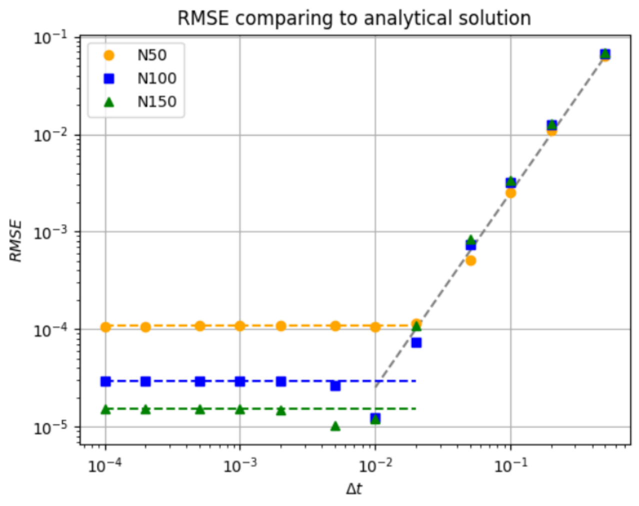
In a simulation, there are several elements that contribute to the error ε_u. In this study, we were only interested in the error contribution of the time discretization scheme $\varepsilon_{\Delta t}$ to verify the order of the scheme. We assume that the error is formed by $\varepsilon_u = \varepsilon_{\Delta t} + \varepsilon_{\Delta x} + \varepsilon_{\text{num}}$, where $\varepsilon_{\Delta x}$ is the spatial discretization error, and ε_{num} is the error introduced by numerical errors, and other factors. We know that $\varepsilon_{\Delta x}$ is related to the grid size, so we can assume that is constant among the experiments with the same grid size.

There are several possible approaches to study the error. Our strategy was to choose a position cell (i, j) and compare the values of the cell in this position for the different samples. We define as the reference sample $\tilde{u}$, obtained by running the simulation with a $\Delta t = 10^{-5}$. We computed the absolute difference between every sample and the chosen reference solution $|u - \tilde{u}|$ and we plotted them in Figure 1a, for three different grid sizes. As we assumed that $\varepsilon_{\Delta x}$ is constant among the samples with the same grid size, in this plot we obtain $\varepsilon_{\Delta t} + \varepsilon_{\text{num}}$, allowing us to extract conclusions form the convergence behaviour of $\varepsilon_{\Delta t}$. We can observe how the error decreases when the timestep gets smaller, proportionally to $O(\Delta t^2)$, until a point where the error flattens. Our assumption is that this happens when $\varepsilon_{\Delta t} < \varepsilon_{\text{num}}$, and after a certain point this ε_{num} dominates the entire error. It is also remarkable to see how this point of flattening happens on smaller timesteps for increasing resolutions of the domain.

As me mentioned before, the Taylor–Green vortex scenario has an analytical solution u^*, what allows us to obtain the exact error ε_u of the solutions we obtained, as those are going to be of the form $u = u^* + \varepsilon_u$. To



(a)



(b)

Figure 1 Caption for both figures

compute this error ε_u, we compute the root-mean-square error (RMSE) of the the velocity flow field, compared to the analytical solution as follows:

$$\text{RMSE} = \sqrt{\sum_{(i,j)} (u^*_{ij} - u_{ij})^2} \tag{9}$$

This is being plotted in Figure 1b, where once again we can observe the error decreasing proportionally to $O(\Delta t^2)$, showing once again a second order convergence in time. In this Figure is very visible the flattening of the error, and how it happens in different points for different resolutions of the domain. This is given that, in this case, the flattening occurs when $\varepsilon_{\Delta t} < \varepsilon_{\Delta x}$, as this time the spatial error is included. This plot also clearly shows how this spatial error decreases for higher domain resolutions, and gives a clear idea of the magnitude of this $\varepsilon_{\Delta x}$ in these scenarios.

3 Calculix

- Small introduction to Calculix.

3.1 Solver parameters

- Talk about the supported time stepping scheme (only one, but higher order). In this case the parametrization is quite simpler, so I would also summarize it here.

3.2 Case study: perpendicular flap

- Talk about the simulation that we performed, a perpendicular flap versus a constant force.
- Mention the scripts created to automatize this task, and to obtain the tip point value from the results.
- Comment on Figure 2, that shows a higher order (between 1 and 2) of convergence. Mention how we compute the error, and how for values <1e-4 the outputed value is the same, meaning either that for lower timesteps the spatial discretization error governs, or that the solver can't achieve better accuracy bc of how values are stored (max of 6 decimal values, more likely given that the results are the exact same).

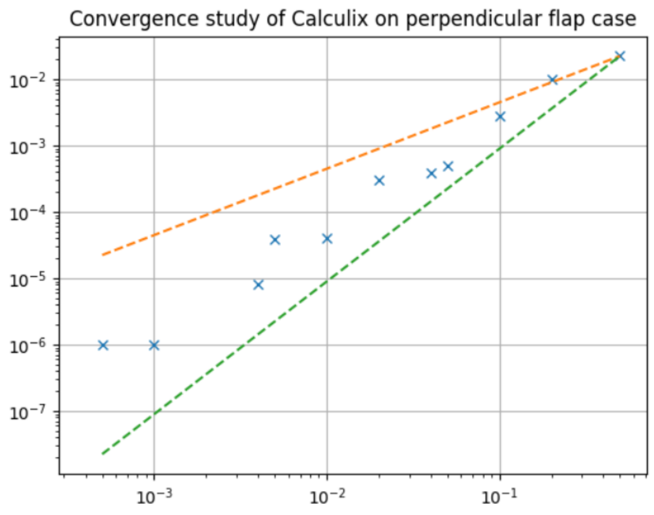


Figure 2 TODO: improve this figure, by adding legend and nicer colors.

4 Coupling the two solvers

- Talk about what a FSI is in general. Talk more specifically about the perpendicular flap case study, based on the preCICE tutorials.
- Talk about the automatization of this, using scripts.
- Supported time stepping schemes, difference between v2 and v3.

4.1 Simulation parameters

- Talk about the parameters of the two solvers (mainly the same as the previous simulations, except change of openFAOM solver).
- Mention the possible preCICE parameters (window-size, ...).

4.2 Case study: FSI - perpendicular flap

- Comment on Figure 3. Show how First order convergence seems to be working, but higher order performs poorly. This is due to an error on the openFAOM adapter, which only supports Euler timestepping.
- Crank Nikolson needs two evaluations per time step, or reuse buffered data of previous timesteps (what is doing openFAOM, most likely, TODO: check this.).
- Give reasoning why this is not working, give some clues what should be done to actually improve it.

5 OpenFOAM Adapter

- Maybe explain a bit how it interacts with the solver, quite documented already by Adapter documentation, and by Article of Gerasimos et al..
- Mention what should be fixed, maybe propose a prototype?
- Mention how should be tested, with a fake-fluid setup for example. Then also test with the same setup to see if it is viable.

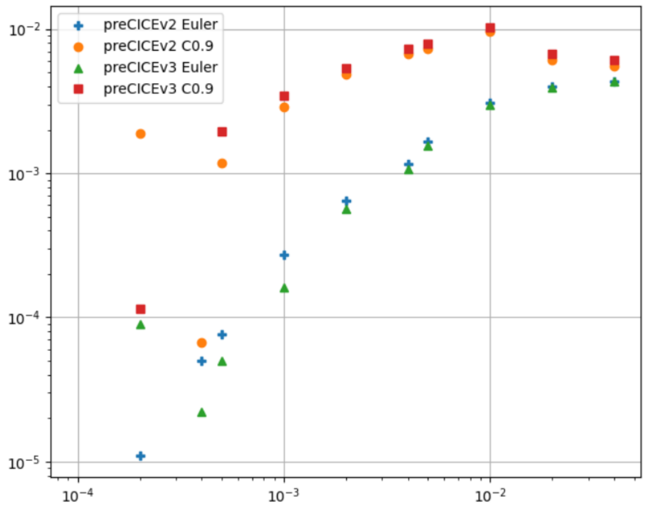


Figure 3 TODO: add title and maybe convergence lines

6 Conclusions and future work

- Talk about the convergence conclusions of each of the solvers. Mention the obtained results with the preCICE couplings.
- Give directions on what to fix of the OpenFOAM adapter, and what to be tested after the fixing implementation.

References

[1] John Crank and Phyllis Nicolson. “A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type”. In: *Mathematical proceedings of the Cambridge philosophical society*. Vol. 43. 1. Cambridge University Press. 1947, pp. 50–67.

[2] Geoffrey Ingram Taylor and Albert Edward Green. “Mechanism of the production of small eddies from large ones”. In: *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences* 158.895 (1937), pp. 499–521.

[3] Alexandre Joel Chorin. “Numerical solution of the Navier-Stokes equations”. In: *Mathematics of computation* 22.104 (1968), pp. 745–762.