

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

Abstract — The accurate numerical simulation of physical phenomena is crucial in various scientific and engineering disciplines, and open-source solvers offer flexibility and accessibility to researchers and practitioners. A vital aspect of these solvers is the order of the time-stepping scheme employed to evolve the solution over time, as it significantly affects the error of the obtained solutions. This project investigates and verifies the order of different time-stepping methods implemented on various solvers. We focused on higher-order methods, which give better accuracy and help reduce the computational costs of obtaining a precise enough solution. To study the order of fluid-structure interaction (FSI) simulations using preCICE, we chose OpenFOAM and Calculix as the studied solvers, which are compatible with this coupling library.

We verified the expected error convergence of second-order methods on both solvers using single-solver simulations and confirmed the correct convergence of first-order methods on FSI simulations. Moreover, we tested higher-order time-stepping schemes on FSI simulations and found an incoherent behaviour of the error. We followed by ruling out possible causes of this poor convergence and proposing possible solutions, making this work valuable for future projects aiming to obtain accurate FSI simulations in the preCICE environment.

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 higher 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 these two open source solvers.
- Explain difference between preCICEv2 and preCICEv3.
- All the code accessible at repo: https://github.com/atmarc/guided_research/.

2 OpenFOAM

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 section we will explain the convergence analysis performed to verify that it has a higher order convergence in time.

2.1 Time stepping schemes

This solver offers 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:

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

The Euler implicit scheme would discretize it as follows:

$$\frac{u^{n+1} - u^n}{\Delta t} = F(u^{n+1}, t^{n+1}) \quad (2)$$

This is a first order method, that is quite stable, reason why it is usually the default choice. For the purpose 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:

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{1}{2} [F(u^n, t^n) + F(u^{n+1}, t^{n+1})] \quad (3)$$

OpenFOAM uses a slightly 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 recommended in their documentation.

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{\theta}{2} F(u^n, t^n) + \left(1 - \frac{\theta}{2}\right) F(u^{n+1}, t^{n+1}) \quad (4)$$

2.2 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.

2.3 Convergence study

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

$$u(x, y, t) = -\cos(x) \sin(y) e^{-2\nu t} \quad (5)$$

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

$$p(x, y, t) = -\frac{1}{4} [\cos(2x) + \sin(2y)] e^{-2\nu t} \quad (7)$$

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 . In Figure 1 we can see an example of the computed initial velocities.

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 2a, 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

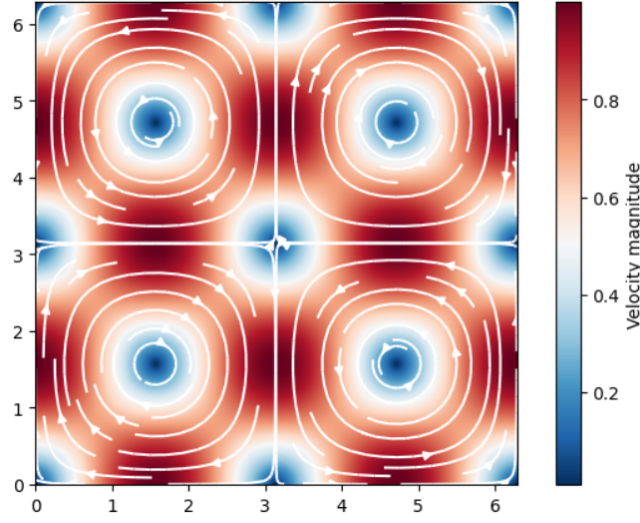


Figure 1

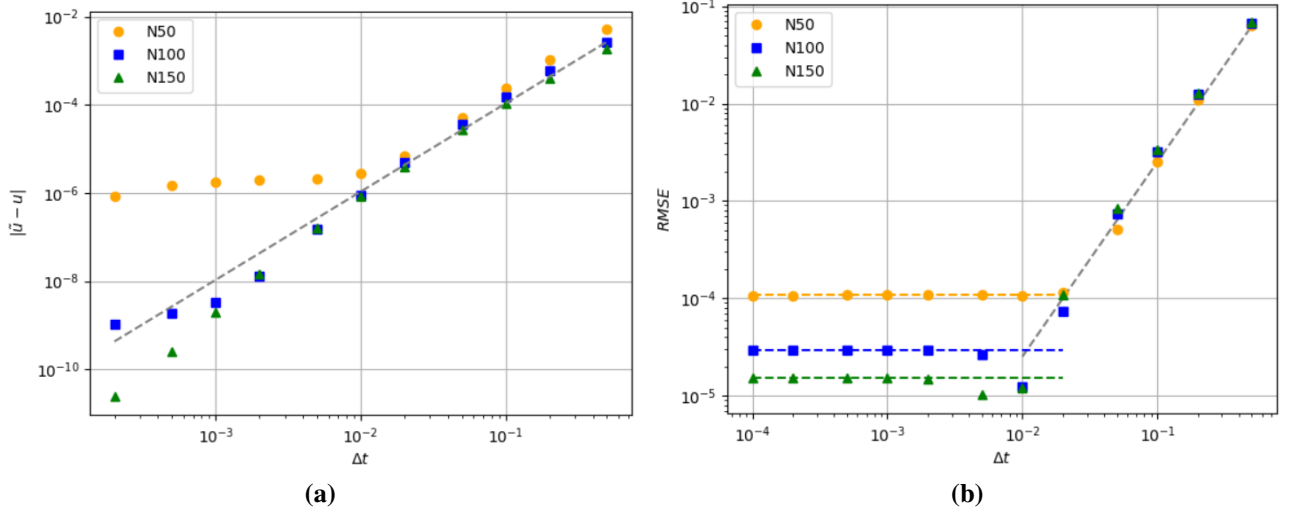


Figure 2 TODO: Caption for both figures

$\varepsilon_{\Delta t} + \varepsilon_{\text{num}}$, allowing us to extract conclusions from 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 we 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 compute this error ε_u , we compute the root-mean-square error (RMSE) of the velocity flow field, compared to the analytical solution as follows:

$$\text{RMSE} = \sqrt{\sum_{(i,j)} (u_{ij}^* - u_{ij})^2} \quad (8)$$

This is being plotted in Figure 2b, 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

CalculiX is an open-source finite element software suite primarily used for solving structural analysis problems. It is designed to simulate the behavior of mechanical and structural systems subjected to various loading conditions. CalculiX provides capabilities for linear and nonlinear static, dynamic, and thermal analyses. It supports a variety of element types, boundary conditions, and material models, making it suitable for a wide range of engineering simulations. In this section, we will overview the timestepping scheme implemented in the solver, and present the convergence study we performed, that displays a higher order convergence.

3.1 Time stepping scheme

The only time-stepping scheme implemented in CalculiX is the α -method [4]. The solver allows the user to select between an implicit or an explicit version of it, and allows to control the α parameter. Moreover, one can define a fixed timestep size, using the DIRECT clause. To get an idea of how this method works, we can start with a given material point with displacement \mathbf{u} , velocity \mathbf{v} and acceleration \mathbf{a} . We know that the acceleration and velocity are related as such $\mathbf{a} = \dot{\mathbf{v}}$, and this yields to the following formula to obtain the next timestep values:

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \int_{t^n}^{t^{n+1}} \mathbf{a}(\xi) d\xi \quad (9)$$

The integral on the right-hand side can be approximated by a linear combination of \mathbf{a}^n and \mathbf{a}^{n+1} :

$$\mathbf{a}(\xi) \approx (1 - \gamma)\mathbf{a}^n + \gamma\mathbf{a}^{n+1} \quad (10)$$

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \left[(1 - \gamma)\mathbf{a}^n + \gamma\mathbf{a}^{n+1} \right] \quad (11)$$

A similar reasoning can be applied to \mathbf{u} , as $\dot{\mathbf{u}} = \mathbf{v}$, hence:

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \int_{t^n}^{t^{n+1}} \mathbf{v}(\eta) d\eta = \mathbf{u}^n + \Delta t \mathbf{v}^n + \int_{t^n}^{t^{n+1}} \int_{t^n}^{\eta} \mathbf{a}(\xi) d\xi d\eta \quad (12)$$

Assuming again that we can approximate \mathbf{a} by a linear combination of \mathbf{a}^n and \mathbf{a}^{n+1} in the interval $[t^n, t^{n+1}]$, we can compute the new displacement \mathbf{u}^{n+1} as:

$$\mathbf{a}(\xi) \approx (1 - 2\beta)\mathbf{a}^n + 2\beta\mathbf{a}^{n+1} \quad (13)$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{v}^n + \frac{1}{2}(\Delta t)^2 \left[(1 - 2\beta)\mathbf{a}^n + 2\beta\mathbf{a}^{n+1} \right] \quad (14)$$

Notice how the linear combinations can be different, so $2\beta \neq \gamma$. This is the basic setup of the α -method, which is proven to be second-order accurate and unconditionally stable for $\alpha \in [-1/3, 0]$, if γ and β satisfy that [5]:

$$\beta = \frac{1}{4}(1 - \alpha)^2 \quad (15)$$

$$\gamma = \frac{1}{2} - \alpha \quad (16)$$

This α parameter controls the high frequency dissipation, and in CalculiX the value set by default is $\alpha = -0.05$.

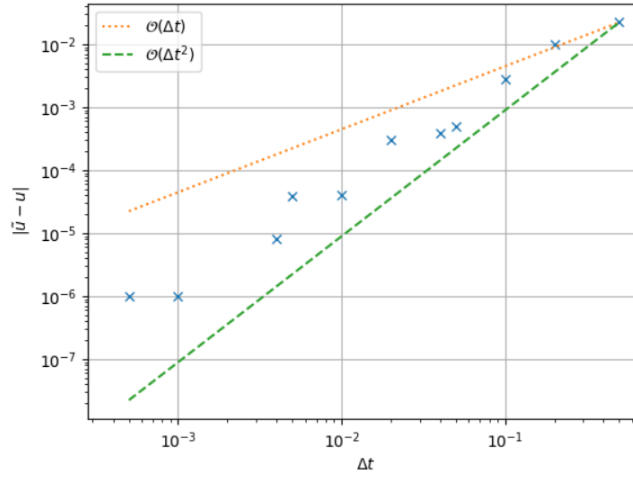


Figure 3 TODO: .

3.2 Convergence study

In this case, we simulated a solid elastic flap fixed to the floor. A constant force is applied perpendicular to the flap, which is initially resting, which makes it oscillate due to its elasticity. This scenario can be seen in further detail in the solid part of the preCICE perpendicular-flap tutorial [6] or in the code repository of this project.

Given the oscillatory behaviour of the flap, we measured the displacement of the flap's tip over time and used this value for the convergence study. As before, we implemented a script to automatize the execution of the simulations and the post-processing of the output data. We defined a reference solution \tilde{u} as with the OpenFOAM study, which is the one obtained with a $\Delta t = 4 \times 10^{-4}$. This value was chosen given that for smaller values of Δt , the solution is exactly the same as \tilde{u} . This can be due to another error contribution governing the total error after this point, such as the spatial discretization error. Then, we plotted the absolute difference to the reference solution $|u - \tilde{u}|$ to observe how the error behaves relative to this solution. One can see in Figure 3 how the error decreases faster than $O(\Delta t)$. With the obtained results, it is hard to argue if it follows a second-order convergence, but it is higher-order.

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 (window-size).
- Talk about the parameters of the two solvers (mainly the same as the previous simulations, except change of openFOAM solver).
- Mention the possible preCICE parameters (coupling scheme?, ...).
- Comment on Figure 5. 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.
- Give reasoning why this is not working, give some clues what should be done to actually improve it.

We have observed on the previous sections how a higher order convergence is achievable with the OpenFOAM and CalculiX solvers. In this section, we will test them in a coupled simulation making use of the preCICE coupling library. To do so, we make use of the perpendicular flap setup that can be found on the example tutorials of the library [6]. It consists of two components, a two-dimensional fluid flowing through a channel,

and a solid, elastic flap fixed to the floor of this channel. The fluid and solid parts of the simulation are computed by OpenFOAM and CalculiX respectively, communicating between them with the help of preCICE. An example of this can be seen in Figure 4.

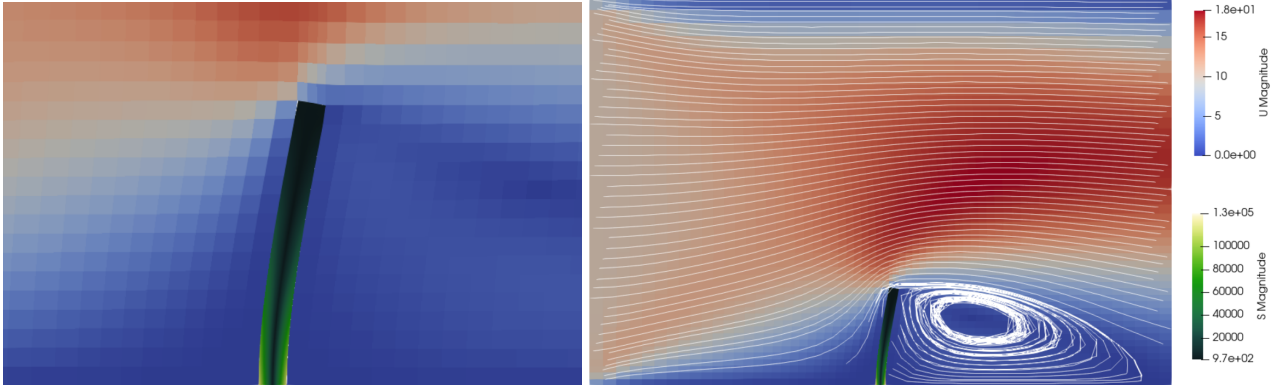


Figure 4 Example solution of the FSI simulation. The left image shows the zoomed in flap, where the stress of the material can be seen in a green scale. On the right image, we observe the whole domain, where the velocity field magnitude can be seen.

The flap oscillates due to the fluid pressure building up on its surface during a limited period, given by its initial resting position. The oscillations dampen until the entire system reaches a steady state. We will quantify the error of the simulation by measuring the displacement of the tip of the flap at time $t = 1s$, which is still on a oscillatory behaviour. We wrote specific scripts once again to automatize this task, utilizing very similar parameters than in the single case analysis previous tasks.

We ran multiple simulations setting the same timestep size on both solvers, and on the preCICE configuration. We also used the two introduced time stepping schemes for the fluid solver, to observe the difference on their error convergence. Moreover, we ran tests with the stable version 2 and the newest version 3 of preCICE, to observe if the update of the library added or decreased the overall error. In Figure 5 we can observe the results

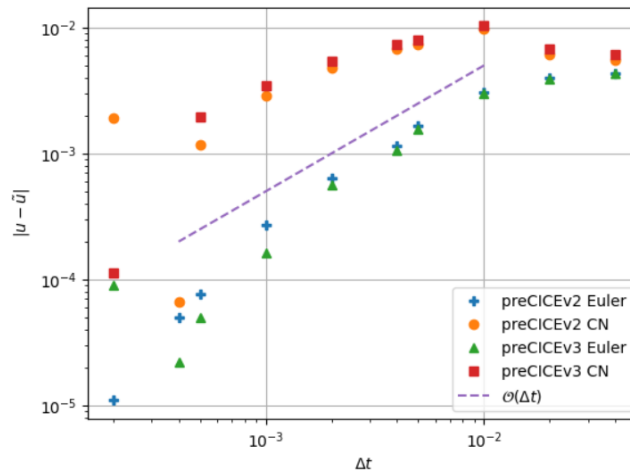


Figure 5 TODO: add title and maybe convergence lines

of this experiments. One can observe how using the Euler method, we can achieve a first order convergence of the error. On the other side, we see how the Crank Nikolson method performs poorly in this scenario, giving worst convergence. It is relevant to mention how the different versions of the code seem to give different results, even though it does not affect the apparent convergence.

preCICE is a complex environment with many moving many pieces. Each solver has its own adapter, that enables it to be a participant in a partitioned multi-physics simulation. This translates in many possible sources of error.

5 Fake Fluid / verify CalculiX adapter

- To verify that the CalculiX adapter works fine with new version, we test, convergence with a fake fluid.
- It shows same convergence as single CalculiX analysis Fig 4.

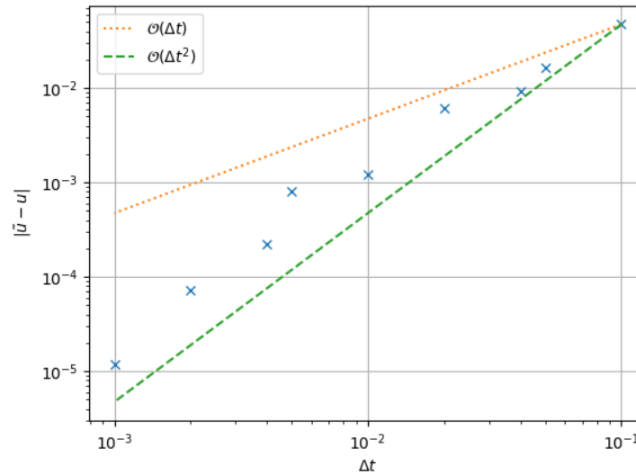


Figure 6 TODO:

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.
- [4] Guido Dhondt. “Calculix crunchix user’s manual version 2.12”. In: (2023), p. 300.
- [5] Guido Dhondt. *The finite element method for three-dimensional thermomechanical applications*. John Wiley & Sons, 2004.
- [6] *preCICE Tutorials: Perpendicular flap*. <https://precice.org/tutorials-perpendicular-flap.html>. Accessed: 8th March 2024.