A direction-splitting Navier–Stokes solver on co-located grids

A. Chiarini, M. Quadrio, F. Auteri[1]

Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano.

Abstract

We introduce a new finite-difference solver for the incompressible Navier–Stokes equations that exploits the direction-splitting method proposed by Guermond and Minev in 2010, but is formulated on a co-located grid. The main ingredients of the new solver are: i) the direction-splitting approach adopted for both the momentum and the pressure equations; and ii) the co-located grid approach. The solver is parallelised by the Schur-complement method, and achieves very high performance levels on thousands of processors. Several test cases are proposed to assess the accuracy and efficiency of the method.

Keywords:

1. Introduction

As the power of supercomputers increases, new problems become tractable that were out of reach only a few years ago. A typical application requiring an extensive computational time is the investigation of the stability properties of 3D laminar flows, the so-called tri-global stability analysis [1]. For this kind of flows, the methods relying on the explicit construction of the tangent matrix, to be used in Newton's method to compute the base flow and in the calculation of the direct and adjoint eigenvectors, are usually too expensive in terms of memory and computing time, especially when the problem at hand lacks symmetries that can be leveraged to reduce its computational complexity [2]. For this reason, the most common approach to tri-global stability analysis is based on time-dependent solvers [3, 4, 5]. Also in this case, however, the high computational cost of this kind of calculations severely limits the kind of problems within reach. Other interesting and expensive problems arise in the numerical simulation of complex fluid-structure interaction problems typical of bio-medical applications [6] or mixing problems in microfluidics [7].

For such calculations, spectral-element methods are often used [4, 8, 9]. These methods provide better geometrical flexibility while still retaining high accuracy, and scale very well on massively parallel computers. Well tested open-source solvers are available online, such as Nek5000 [10] or Nektar++ [11]. Unfortunately, their computational complexity is quite high, they are limited to relatively simple geometries and lose part of their accuracy in presence of corners or singularities in the domain [12]. Moreover, their efficiency can degrade quite a lot in

[1]Corresponding author.

Email address: franco.auteri@polimi.it (F. Auteri)

the case of moving bodies or fluid-structure interaction problems. This is also the case for finite element methods, even when XFEM or similar techniques are used to take immersed bodies into account [13, 14]. The fastest solvers are those based on the finite-difference discretisation [15, 16], even if quite recently, Lattice Boltzmann Methods are joining finite differences as a fast discretisation method for the incompressible Navier–Stokes equations [17]. Finite differences seem to provide the best trade-off between accuracy and efficiency when problems with moving bodies are tackled with the immersed boundary method [18].

Several fast finite-difference Navier–Stokes solvers have been proposed in the past. Most of them rely on fast Poisson solvers [19, 15, 16] to solve the pressure equation. Such solvers exploit a spectral decomposition of the Laplacian operator and Fast Fourier Transforms (FFTs) in planes — where a uniform Cartesian grid has to be used — to reduce the solution of the problem to a set of tridiagonal linear systems for the third direction. While this approach leads to solvers with a suboptimal computational complexity, since the cost of fast Poisson solvers grows at least like $N^3 \log N$, it can leverage highly optimised libraries for FFT [20], thus leading to very efficient solvers. However, when the number of unknowns becomes very high, as is the case for very demanding present and future simulations, the uniform grid and the $\log N$ term become increasingly penalising constraints. To obtain an optimal scalable solver, multigrid has usually been the solver of choice for the pressure Poisson equation [21]. Unfortunately, the multigrid method has its drawbacks: it is an iterative scheme which may require complicate algorithmic constructions that limit the performance with respect to fast Poisson solvers [22], and it may suffer from the presence of nonuniform grids if *ad hoc* smoothers are not employed. Quite recently, Guermond and Minev [23, 24] proposed a new fractional-step method that requires only the solution of tridiagonal linear systems in solving for both the velocity and the pressure. This method actually extends the use of direction splitting [25], similar to ADI [26], to the pressure step, thus improving the efficiency and scalability of the solver. Since the method is direct, neither iterations nor special data structures are necessary. Moreover, using the Schur-complement method [27, 28, 29], high levels of parallelism can be achieved as well as an optimal computational complexity.

All of the aforementioned discretisation methods, with the exception of spectral methods in three-periodic boxes, suffer from the problem that a compatibility condition — the well known LBB condition [30, 31, 32] — must be satisfied by the spatial discretisation of the velocity field and that of the pressure field to obtain a stable discretisation. Incompatible discretisations will allow spurious pressure or velocity modes to appear and will often completely destroy the solution or, in any case, prevent the numerical solution from converging to the exact one [33]. For this reason, finite difference solvers are frequently formulated on staggered grids, where four different grids are employed for the three velocity components and pressure. When dealing with complicated geometries, or multiscale phenomena, that require local refinement, the staggered arrangement is quite inconvenient. Moreover, in the immersed-boundary method, interpolation stencils that are different for each grid must be computed. In fluid-structure interaction problems, this implies a four-fold increase of the computing time needed for the interpolations in 3D, a significant penalty especially for interpolation techniques that use extended stencils [34, 35]. Also, when the grid is refined, the staggering must be taken into account and complicated interpolation stencils may be required [36]. For this reasons, the use of co-located grids where all variables are stored in the same grid point can lead to a distinct performance advantage.

Unfortunately, co-located grids do not satisfy the LBB condition, and are prone to the insurgence of pressure checkerboard modes [37]. Several techniques have been proposed in the past to eliminate such spurious pressure modes. They can be classified in two broad groups: stabilisation methods [38, 39], where a special interpolation is used in the calculation of the divergence operator, and filtering methods [40]. While filtering methods are in principle an elegant way to solve the problem, without introducing any stabilisation, since they work by filtering spurious modes out of the solution, their use is practical only when uniform cartesian grids are employed. Indeed, the idea behind filtering methods is to remove the unphysical oscillations by subtracting them from the solution. Since the spurious pressure modes belong to the null space of the discrete Stokes operator, the solution is first orthogonally projected on this null space, and the projection is then subtracted from the solution. Regrettably, an orthonormal basis for the null space can be obtained trivially only for equispaced orthogonal Cartesian grids [40]; for nonuniform grids, it must be computed numerically by solving an eigenvalue problem. For what concerns stabilisation methods, the most renowned method was proposed by Rhie & Chow [38]. It was formulated as an interpolation method to be used in the calculation of the divergence of the momentum equation, to obtain a Poisson equation for the pressure. Indeed, the method can be reformulated as the addition of a stabilisation term, arising from the discretisation of a fourth-order elliptic operator, to the right-hand side of the Poisson equation for the pressure [37]. In [41], the authors showed that such a stabilisation term can be actually multiplied by a coefficient lower than one, and still a stable formulation is obtained.

The aim of the present work is to introduce a new, fast, parallel solver for the incompressible Navier–Stokes equations that hinges on the direction-splitting, fractional-step method proposed by Guermond and Minev [23, 24] and uses second-order finite differences on a co-located grid. The resulting direction-splitting formulation is not prone to checker-board pressure modes, so that no stabilisation term is required. A simple and fast algorithm is thus obtained that only requires the solution of 1D tridiagonal linear systems. Such a solver is expected to be faster than its staggered counterpart, and can be used to implement fast immersed-boundary solvers aimed at solving triglobal stability problems [2] and fluid-structure interaction problems typical of biological and bio-medical applications [6, 42].

The paper is organised as follows. After this introduction, in the next section the key elements of the solver are presented: the adopted time discretisation, the fractional-step algorithm, the formulation of the non-linear term and the imposition of boundary conditions. The third section describes the parallelisation. In the fourth section, we report some results to show the accuracy and efficiency of the method: thorough convergence studies have been carried out to check that the expected convergence rate was achieved, and several well-documented test cases have been reproduced to check the solution accuracy for different sets of boundary conditions. A few concluding remarks are drawn in the last section.

2. Methods

2.1. Mathematical framework

We consider the time dependent, dimensionless incompressible Navier–Stokes equations solved in a three-dimensional volume $\Omega = (0, L_x) \times (0, L_y) \times (0, L_z)$:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\,\boldsymbol{u} - \dfrac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \boldsymbol{\nabla} p = \boldsymbol{f} & \text{in } \Omega \times (0, \mathrm{T}), \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 & \text{in } \Omega \times (0, \mathrm{T}), \\ \boldsymbol{u}|_{\partial\Omega} = \boldsymbol{a} & \text{in } \partial\Omega \times (0, \mathrm{T}), \\ \boldsymbol{u}|_{t=0} = \boldsymbol{u}_0 & \text{in } \Omega, \end{cases} \tag{1}$$

where $\boldsymbol{u}$ is the velocity, p the reduced pressure, $\boldsymbol{f}$ a body force, $\boldsymbol{a}$ contains the boundary data and $\boldsymbol{u}_0$ is the initial condition. The initial and boundary conditions need to satisfy the well known compatibility conditions [43].

An approach often used to solve the Navier–Stokes equations is the incremental pressure-correction scheme, which can be written as the following singular perturbation problem:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}_\epsilon}{\partial t} + (\boldsymbol{u}_\epsilon \cdot \boldsymbol{\nabla})\,\boldsymbol{u}_\epsilon - \dfrac{1}{\mathrm{Re}}\nabla^2 \boldsymbol{u}_\epsilon + \boldsymbol{\nabla} p_\epsilon = \boldsymbol{f} & \boldsymbol{u}_\epsilon|_{\partial\Omega\times(0,\mathrm{T})} = \boldsymbol{a},\ \ \boldsymbol{u}_\epsilon|_{t=0} = \boldsymbol{u}_0, \\ -\Delta t \nabla^2 \phi_\epsilon + \boldsymbol{\nabla}\cdot \boldsymbol{u}_\epsilon = 0 & \dfrac{\partial \phi_\epsilon}{\partial n}|_{\partial\Omega\times(0,\mathrm{T})} = 0, \\ \Delta t \dfrac{\partial p_\epsilon}{\partial t} = \phi_\epsilon - \dfrac{\chi}{\mathrm{Re}}\boldsymbol{\nabla}\cdot \boldsymbol{u}_\epsilon & p_\epsilon|_{t=0} = p_0, \end{cases} \tag{2}$$

where $\epsilon := \Delta t$ is the perturbation parameter and $\chi \in [0,1]$. Guermond and Shen [44] found that $\boldsymbol{u}_\epsilon$ is a $O(\Delta t^2)$ perturbation of $\boldsymbol{u}$ in the $\boldsymbol{L}^2$-norm for all $\chi \in [0,1]$. $\chi = 0$ corresponds to the standard form of the projection method, while $\chi = 1$ corresponds to the rotational form. Intermediate values of χ are possible.

Recently, Guermond and Minev [23] showed that the same stability properties and error estimates of the aforementioned singular perturbation problem can be obtained by substituting the Laplacian operator in the second equation with a more general operator, that will be called A, provided the bilinear form induced by A, $a(p,q) := \int_\Omega pAq$, is symmetric and $||\nabla q||^2_{L^2} \leq a(q,q)$. Therefore, we have

$$\begin{cases} \dfrac{\partial \boldsymbol{u}_\epsilon}{\partial t} + (\boldsymbol{u}_\epsilon \cdot \boldsymbol{\nabla})\,\boldsymbol{u}_\epsilon - \dfrac{1}{\mathrm{Re}}\nabla^2 \boldsymbol{u}_\epsilon + \boldsymbol{\nabla} p_\epsilon = \boldsymbol{f} & \boldsymbol{u}_\epsilon|_{\partial\Omega\times(0,\mathrm{T})} = \boldsymbol{a},\ \ \boldsymbol{u}_\epsilon|_{t=0} = \boldsymbol{u}_0, \\ \Delta t A \phi_\epsilon + \boldsymbol{\nabla}\cdot \boldsymbol{u}_\epsilon = 0 & \dfrac{\partial \phi_\epsilon}{\partial n}|_{\partial\Omega\times(0,\mathrm{T})} = 0, \\ \Delta t \dfrac{\partial p_\epsilon}{\partial t} = \phi_\epsilon - \dfrac{\chi}{\mathrm{Re}}\boldsymbol{\nabla}\cdot \boldsymbol{u}_\epsilon & p_\epsilon|_{t=0} = p_0, \end{cases} \tag{3}$$

that coincides with the projection method if $A = -\nabla^2$. It is interesting to notice that for transient flow, the computed velocity field is not divergence free, owing to the second and third equations. However, $\boldsymbol{u}_\epsilon$ is a second-order-in-time approximation of the exact velocity field.

Let the Crank–Nicolson time scheme be used for advancing the solution in time, with the leap-frog strategy for the pressure. As a result, the discrete expression of the pressure-correction scheme reads:

$$\begin{cases} \dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} - \dfrac{1}{2Re}\nabla^2\left(\boldsymbol{u}^{n+1} + \boldsymbol{u}^n\right) = \boldsymbol{f}^{n+1/2} - \boldsymbol{\nabla} p^{*,n+1/2} - \mathbf{nl}(\boldsymbol{u}^{*,n+1/2}), \\ A\phi^{n+1/2} = -\dfrac{1}{\Delta t}\boldsymbol{\nabla}\cdot \boldsymbol{u}^{n+1}, \\ p^{n+1/2} = p^{n-1/2} + \phi^{n+1/2} - \dfrac{\chi}{2\mathrm{Re}}\boldsymbol{\nabla}\cdot(\boldsymbol{u}^{n+1} + \boldsymbol{u}^n), \end{cases} \tag{4}$$

where $\mathbf{nl}$ denotes the nonlinear term, the superscripts n and $n+1$ indicate two successive time steps for velocity, $n + 1/2$ the intermediate time step that is used for pressure and for the right-hand side. In particular, $\boldsymbol{u}^{*,n+1/2} = (3\boldsymbol{u}^n - \boldsymbol{u}^{n-1})/2$ is a second-order extrapolation of the velocity field from the time steps $n-1$ and n.

2.2. *Algorithm*

In this section we present the solution algorithm that is needed to advance the discrete solution in time. The solution algorithm exploits direction splitting [25] for the three directions for both the velocity, as already done in the past, see e.g. [45, 46], and the pressure equation, as first proposed by Guermond and Minev [23, 24], by suitably selecting the A operator in equation (4).

2.2.1. *Pressure predictor*

The first step of the algorithm consists in computing the pressure predictor $p^{*,1/2}$. The algorithm is initialised by setting $p^{*,1/2} = p^{*,-1/2}$, and for $n > 0$ we set

$$p^{*,n+1/2} = p^{*,n-1/2} + \phi^{n-1/2}. \tag{5}$$

2.2.2. *Velocity update*

In the second step, the velocity field is updated by solving the momentum equation. This amounts, by virtue of the direction splitting technique, to the solution of a set of tridiagonal linear systems in the three directions:

$$\begin{cases} \dfrac{\boldsymbol{\xi}^{n+1} - \boldsymbol{u}^n}{\Delta t} = \dfrac{1}{\text{Re}} \nabla^2 \boldsymbol{u}^n + \boldsymbol{f}^{n+1/2} - \boldsymbol{\nabla} p^{*,n+1/2} - \mathbf{nl}(\boldsymbol{u}^{*,n+1/2}), \\ \dfrac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\xi}^{n+1}}{\Delta t} - \dfrac{1}{2Re} \dfrac{\partial^2}{\partial x^2} \left(\boldsymbol{\eta}^{n+1} - \boldsymbol{u}^n \right) = 0, \\ \dfrac{\boldsymbol{\zeta}^{n+1} - \boldsymbol{\eta}^{n+1}}{\Delta t} - \dfrac{1}{2Re} \dfrac{\partial^2}{\partial y^2} \left(\boldsymbol{\zeta}^{n+1} - \boldsymbol{u}^n \right) = 0, \\ \dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{\zeta}^{n+1}}{\Delta t} - \dfrac{1}{2Re} \dfrac{\partial^2}{\partial z^2} \left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n \right) = 0. \end{cases} \tag{6}$$

This formulation can be manipulated to make the computation of $\boldsymbol{u}^{n+1}$ easier and more efficient. It reads:

$$\begin{cases} \left(\boldsymbol{\eta}^{n+1} - \boldsymbol{u}^n \right) - \dfrac{\Delta t}{2\text{Re}} \dfrac{\partial^2}{\partial x^2} \left(\boldsymbol{\eta}^{n+1} - \boldsymbol{u}^n \right) & = \Delta t \boldsymbol{f}^{n+1/2} - \Delta t \boldsymbol{\nabla} p^{*,n+1/2} - \Delta t\, \mathbf{nl}(\boldsymbol{u}^{*,n+1/2}), \\ \left(\boldsymbol{\zeta}^{n+1} - \boldsymbol{u}^n \right) - \dfrac{\Delta t}{2\text{Re}} \dfrac{\partial^2}{\partial y^2} \left(\boldsymbol{\zeta}^{n+1} - \boldsymbol{u}^n \right) & = \left(\boldsymbol{\eta}^{n+1} - \boldsymbol{u}^n \right), \\ \left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n \right) - \dfrac{\Delta t}{2\text{Re}} \dfrac{\partial^2}{\partial z^2} \left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n \right) & = \left(\boldsymbol{\zeta}^{n+1} - \boldsymbol{u}^n \right). \end{cases} \tag{7}$$

These expressions show that updating the velocity field amounts to the solution of tridiagonal linear systems only.

2.2.3. *Penalty step*

In the third step, $\phi^{n+1/2}$ is computed. In this algorithm, as suggested in [23, 24], we use again the direction-splitting operator. The second equation of the system (4) reads:

$$A p^{n+1/2} = -\frac{1}{\Delta t} \boldsymbol{\nabla} \cdot \boldsymbol{u}^{n+1}. \tag{8}$$

The choice of the actual expression for the operator A is key to increasing the performance of the algorithm. As in [23, 24], in this work we used the following expression for $A := (1 - \partial^2/\partial x^2)(1 -$

$\partial^2/\partial y^2)(1-\partial^2/\partial z^2)$, that satisfies the necessary conditions for the stability and convergence of the method. We thus obtain that $\phi^{n+1/2}$ can be computed by solving the following equation cascade:

$$\begin{cases} \psi - \dfrac{\partial^2 \psi}{\partial x^2} = -\dfrac{1}{\Delta t}\boldsymbol{\nabla}\cdot\boldsymbol{u}^{n+1}, & \dfrac{\partial \psi}{\partial x}|_{x=0,1} = 0, \\ \varphi - \dfrac{\partial^2 \varphi}{\partial y^2} = \psi, & \dfrac{\partial \varphi}{\partial y}|_{y=0,1} = 0, \\ \phi^{n+1/2} - \dfrac{\partial^2 \phi^{n+1/2}}{\partial z^2} = \varphi, & \dfrac{\partial \phi^{n+1/2}}{\partial z}|_{y=0,1} = 0. \end{cases} \tag{9}$$

As for the velocity update, computing $\phi^{n+1/2}$ too amounts to solve tridiagonal linear systems only.

We observe that, despite the chosen form of A leads to an operator whose kernel is void, in principle this does not ensure that spurious pressure modes cannot arise in the solution, as clearly shown in [33]. For some schemes, spurious pressure modes arise when very small time steps are considered, since the constant in the LBB condition can go to zero with the time step. For other schemes, the instability is not observed or it is only observed for very long simulations toward a steady state, see [36] and the references therein. For the proposed scheme, we did a lot of testing to ascertain the presence of spurious pressure modes, partially reported in section 5. We tried unsteady simulations with very small time steps with respect to the spatial discretisation, as in [33], and long simulations leading to a steady state. In both cases, we did not observe spurious pressure modes, either in the pressure fields or in the pressure-error field, when available. We also compared the solution obtained by the described algorithm with that obtained by adding a stabilisation term to the pressure equation according to [41]. The obtained pressure fields were almost identical. We are aware that numerical tests are not equivalent to a rigorous mathematical proof, therefore we do not claim that the proposed method is always free from any pressure spurious mode. Nonetheless, we can say that the proposed method is very robust since no instabilities were observed in a wide range of simulations with very different time and space resolutions.

2.2.4. Pressure update

In the last step of the algorithm, the pressure is updated as follows:

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1/2} - \chi\nu\boldsymbol{\nabla}\cdot\left(\frac{1}{2}\left(\boldsymbol{u}^{n+1} + \boldsymbol{u}^{n}\right)\right). \tag{10}$$

2.3. Treatment of the non-linear term

The formulation of the non-linear term has been chosen to ensure energy conservation. Indeed, it is well known that several different forms of the nonlinear term can be chosen, which are equivalent in the continuum setting but differ once discretised [47]. Such formulations differ essentially by a term multiplied by the divergence of the velocity, and are therefore equivalent only when the velocity field is divergence-free at machine precision. When the velocity field is only approximately divergence-free, as in the present case, the different forms of the nonlinear term are no longer equivalent and, in general, do not satisfy energy conservation, as it is the case in the continuum. Failing to conserve energy can lead to an explosive instability, especially at relatively high Reynolds numbers and when the smaller scales of the flow are not sufficiently resolved. We write the nonlinear term so as to conserve energy as suggested in [48]. For simplicity, we show the discretisation for the $2D$ case on a uniform grid, but the strategy can be easily extended to the three-dimensional case. The conservative discrete weak formulation is:

$$
\begin{aligned}
\boldsymbol{b}_h(\boldsymbol{u}_h, \boldsymbol{\phi}_h, \boldsymbol{\xi}_h) = \sum_{i=1}^{M} \sum_{j=1}^{N} \{ & [\Delta y u_{i+(1/2)j} \phi_{i+(1/2)j} \xi_{ij} - \Delta y u_{i-(1/2)j} \phi_{i-(1/2)j} \xi_{ij} \\
& + \Delta x v_{ij+(1/2)} \phi_{ij+(1/2)} \xi_{ij} - \Delta x v_{ij-(1/2)} \phi_{ij-(1/2)} \xi_{ij}] \\
& - \frac{1}{2} \Delta x \Delta y (\frac{u_{i+(1/2)j} - u_{i-(1/2)j}}{\Delta x} + \frac{v_{ij+(1/2)} - v_{ij-(1/2)}}{\Delta y}) \phi_{ij} \xi_{ij} \}
\end{aligned} \tag{11}
$$

where $\boldsymbol{u}_{i+(1/2)j}$ denotes the velocity at the mid point between $\boldsymbol{x}_{ij}$ and $\boldsymbol{x}_{i+1j}$ and its second order approximation is:

$$
\boldsymbol{u}_{i+(1/2)j} = \frac{\boldsymbol{u}_{i+1j} + \boldsymbol{u}_{ij}}{2}. \tag{12}
$$

The last term of (11) can be recognised to be the divergence of the velocity. This term is supposed to be zero when the end-of-step velocity is used and it is divergence-free, not in our case. Therefore, this term is retained to ensure conservation. The proof of the conservation properties of this formulation of the nonlinear term is very similar to what is done in [49], and it is not repeated here for conciseness. As a result, the conservative formulation of the non-linear term in 3D reads:

$$
\begin{aligned}
&(\boldsymbol{u} \cdot \boldsymbol{\nabla})\, u_l|_{x_i, y_j, z_k} \sim \\
&\frac{1}{4\Delta x} \left[u_l(x_{i+1}, y_j, z_k) \left(u(x_{i+1}, y_j, z_k) + u(x_i, y_j, z_k) \right) - u_l(x_{i-1}, y_j, z_k) \left(u(x_{i-1}, y_j, z_k) + u(x_i, y_j, z_k) \right) \right] + \\
&\frac{1}{4\Delta y} \left[u_l(x_i, y_{j+1}, z_k) \left(v(x_i, y_{j+1}, z_k) + v(x_i, y_j, z_k) \right) - u_l(x_i, y_{j-1}, z_k) \left(v(x_i, y_{j-1}, z_k) + v(x_i, y_j, z_k) \right) \right] + \\
&\frac{1}{4\Delta z} \left[u_l(x_i, y_j, z_{k+1}) \left(w(x_i, y_j, z_{k+1}) + w(x_i, y_j, z_k) \right) - u_l(x_i, y_j, z_{k-1}) \left(w(x_i, y_j . z_{k-1}) + w(x_i, y_j, z_k) \right) \right].
\end{aligned} \tag{13}
$$

The numerical tests confirmed that this formulation does not lead to numerical instabilities as previously observed for the convective form.

3. An additional temporal scheme

This section describes the formulation of the algorithm when a different, more efficient time-advancement scheme is used instead of Crank–Nicolson's, characterised by a larger stability margin in terms of the Courant-Friedrichs-Lewy condition, thus allowing larger time steps. Specifically, we describe the implementation of a popular scheme introduced in [50], hereafter called RK-RAI3. It is a partially implicit method, where the viscous terms are dealt with by a second-order-accurate Crank–Nicolson method, whereas a third-order Runge-Kutta scheme is employed for the convective terms. The single time step Δt is thus divided into three substeps, each one corresponding to an advancement of $\frac{2}{\alpha_i \Delta t}$. After the time discretisation and by using again the leap-frog strategy for the pressure, the Navier–Stokes equations read:

$$\begin{cases} \dfrac{\alpha_i}{2}\dfrac{\boldsymbol{u}_i^n - \boldsymbol{u}_{i-1}^n}{\Delta t} - \dfrac{1}{2Re}\nabla^2\left(\boldsymbol{u}_i^n + \boldsymbol{u}_{i-1}^n\right) \\ \qquad\qquad = \boldsymbol{f}_{i-1/2}^n - \boldsymbol{\nabla} p_{*,i-1/2}^n - \gamma_i\left(\boldsymbol{u}_{i-1}^n\cdot\boldsymbol{\nabla}\right)\boldsymbol{u}_{i-1}^n - \delta_i\left(\boldsymbol{u}_{i-2}^n\cdot\boldsymbol{\nabla}\right)\boldsymbol{u}_{i-2}^n, \\ A\phi_{i-1/2}^n = -\dfrac{\alpha_i}{2\Delta t}\boldsymbol{\nabla}\cdot\boldsymbol{u}_i^n, \\ p_{i-1/2}^n = p_{i-3/2}^n + \phi_{i-1/2}^n - \dfrac{\chi}{2\mathrm{Re}}\boldsymbol{\nabla}\cdot(\boldsymbol{u}_i^n + \boldsymbol{u}_{i-1}^n), \\ \text{b.c.} \\ \text{i.c.} \end{cases} \tag{14}$$

Here the superscript n refers to the time step, while the subscript i refers to the three substeps needed to advance the solution by one time step Δt, and α_i, γ_i and δ_i are the following coefficients:

$$\begin{array}{lll} \alpha_1 = 120/32, & \alpha_2 = 120/8, & \alpha_3 = 120/20, \\ \gamma_1 = 1, & \gamma_2 = 25/8, & \gamma_3 = 45/20, \\ \delta_1 = 0, & \delta_2 = -17/8, & \delta_3 = -25/20. \end{array} \tag{15}$$

$p_{*,i-1/2}^n$ denotes the pressure predictor, described later.

Now, similarly to what is done for the Crank-Nicolson case, we briefly show the algorithm to be used at each time step if the RK-RAI3 time scheme is employed.

3.0.1. Pressure predictor

$$p_{*,i-1/2}^n = p_{i-3/2}^n + \frac{\alpha_i}{\alpha_{i-1}}\phi_{i-3/2}^n. \tag{16}$$

3.0.2. Velocity update

$$\begin{cases} \dfrac{\alpha_i}{\Delta t}\left(\boldsymbol{\eta}_i^n - \tilde{\boldsymbol{u}}_{i-1}^n\right) - \dfrac{1}{Re}\dfrac{\partial^2}{\partial x^2}\left(\boldsymbol{\eta}_i^n - \tilde{\boldsymbol{u}}_{i-1}^n\right) & = \boldsymbol{rhs}, \\ \dfrac{\alpha_i}{\Delta t}\left(\boldsymbol{\zeta}_i^n - \tilde{\boldsymbol{u}}_{i-1}^n\right) - \dfrac{1}{Re}\dfrac{\partial^2}{\partial y^2}\left(\boldsymbol{\zeta}_i^n - \tilde{\boldsymbol{u}}_{i-1}^n\right) & = \left(\boldsymbol{\eta}_i^n - \tilde{\boldsymbol{u}}_{i-1}^n\right), \\ \dfrac{\alpha_i}{\Delta t}\left(\tilde{\boldsymbol{u}}_i^n - \tilde{\boldsymbol{u}}_{i-1}^n\right) - \dfrac{1}{Re}\dfrac{\partial^2}{\partial z^2}\left(\tilde{\boldsymbol{u}}_i^n - \tilde{\boldsymbol{u}}_{i-1}^n\right) & = \left(\boldsymbol{\zeta}_i^n - \tilde{\boldsymbol{u}}_{i-1}^n\right), \end{cases} \tag{17}$$

where

$$\boldsymbol{rhs} = \frac{2}{Re}\nabla^2\tilde{\boldsymbol{u}}_{i-1}^n - 2\boldsymbol{\nabla} p_{*,i-1/2}^n + 2\boldsymbol{f}_{i-1/2}^n - 2\gamma_i\left(\boldsymbol{u}_{i-1}^n\cdot\boldsymbol{\nabla}\right)\boldsymbol{u}_{i-1}^n - 2\delta_i\left(\boldsymbol{u}_{i-2}^n\cdot\boldsymbol{\nabla}\right)\boldsymbol{u}_{i-2}^n.$$

3.0.3. Penalty step

$$\begin{cases} \psi_{i-1/2}^n - \dfrac{\partial^2\psi_{i-1/2}^n}{\partial x^2} & = -\dfrac{\alpha_i}{2\Delta t}\boldsymbol{\nabla}\cdot\tilde{\boldsymbol{u}}_{i-1}^n \\ \varphi_{i-1/2}^n - \dfrac{\partial^2\varphi_{i-1/2}^n}{\partial y^2} & = \psi_{i-1/2}^n, \\ \phi_{i-1/2}^n - \dfrac{\partial^2\phi_{i-1/2}^n}{\partial z^2} & = \varphi_{i-1/2}^n. \end{cases} \tag{18}$$

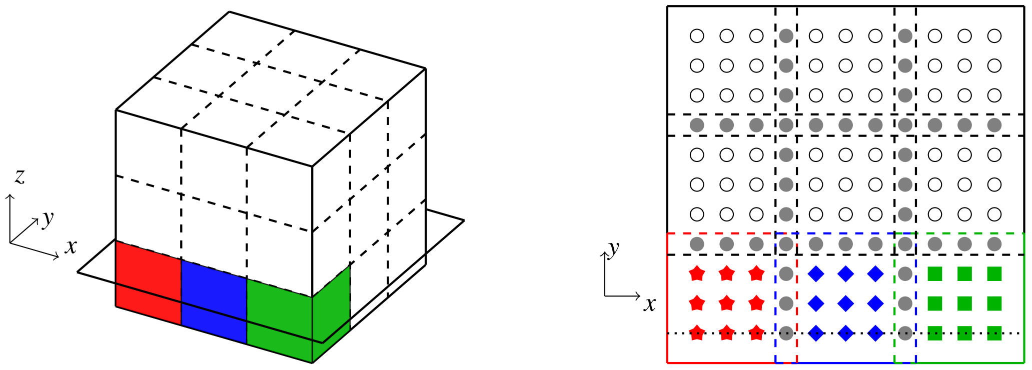

Figure 1: Domain partitioning for parallel implementation. Left: block decomposition of the three-dimensional domain. Right: grid decomposition for parallel execution on the plane highlighted in the left panel; grey circles denote shared interface grid points; red stars, blue diamonds and green squares denote the internal grid points of the blocks highlighted by the same colour code in the left panel; white circles denote the internal grid points of the other blocks. The dotted line highlights the line in the x direction considered in §4.1.

3.0.4. Pressure update

$$p^n_{i-1/2} = p^n_{i-3/2} + \phi^n_{i-1/2} - \frac{\chi}{2\mathrm{Re}} \boldsymbol{\nabla} \cdot (\boldsymbol{u}^n_i + \boldsymbol{u}^n_{i-1}). \tag{19}$$

4. Parallel implementation

The present algorithm has been implemented using central finite differences for the first- and second-order derivatives. The parallel implementation is based on a Cartesian block decomposition of the domain (as shown in figure 1), and uses the Message Passing Interface (MPI) library for maximum portability. Each processor deals with a different block.

The grid points in each block are divided into internal points x_i and shared interface points x_s. For internal points, the right-hand side of the linear systems can be assembled locally to the processor since their stencil is within the same block. In contrast, for the shared interface points, assembling the right-hand side requires some sort of communication as such points are placed on the interfaces between the blocks and their stencil is shared between consecutive processors. Two processors dealing with two consecutive blocks share the interface points placed on the face they have in common — see the right panel of figure 1 — and both of them solve the linear systems for such points in the two directions parallel to the face; in the following we show that this small overhead is necessary to highly reduce the communication among processors. The overall number of shared interfaces along one direction $N_{s,i}$ (with $i = 1, 2, 3$ denoting the direction) depends on the boundary conditions. Indeed, in case of a periodic direction — where the first and last block can be thought as consecutive — the number of interfaces is equal to the number of blocks $N_{p,i}$ in that direction while to $N_{p,i} - 1$ in all the other cases.

The solver is parallelised keeping the communication among processors to a minimum. Communication is needed only for the solution of the linear systems and the assembling of the right-hand sides at the interface points.

4.1. *Parallelisation of the solution of the linear systems*

The spatial discretisation we used reduces each one-dimensional linear problem to a tridiagonal linear system, that can be easily solved using the Schur complement method [27, 28, 29], adopting the same strategy proposed in [24]. To explain the procedure, we consider only a line in the x direction for brevity, along which the solution is split across N_{px} processors (see the right panel of figure 1 as an example); for each processor the unknowns u are divided in internal unknowns u_i and shared interface unknowns u_s. The unknowns and equations of the tridiagonal system associated with the considered line can be reordered so as to separate the internal unknowns from those on the shared interfaces:

$$\begin{bmatrix} A_{ii} & A_{is} \\ A_{si} & A_{ss} \end{bmatrix} \begin{bmatrix} u_i \\ u_s \end{bmatrix} = \begin{bmatrix} f_i \\ f_s \end{bmatrix}. \tag{20}$$

The reordered linear system is no longer tridiagonal, clearly. Considering for example the case in figure 1 with $N_{px} = 3$ and $N_{sx} = 2$ — corresponding to non-periodic boundary conditions — the matrices have the following pattern:

$$A_{ii} = \begin{bmatrix} \star & \star & & & & & & & \\ \star & \star & \star & & & & & & \\ & \star & \star & & & & & & \\ & & & \blacklozenge & \blacklozenge & & & & \\ & & & \blacklozenge & \blacklozenge & \blacklozenge & & & \\ & & & & \blacklozenge & \blacklozenge & & & \\ & & & & & & \blacksquare & \blacksquare & \\ & & & & & & \blacksquare & \blacksquare & \blacksquare \\ & & & & & & & \blacksquare & \blacksquare \end{bmatrix} \quad A_{is} = \begin{bmatrix} & \\ & \\ \star & \\ \blacklozenge & \\ & \\ & \blacklozenge \\ & \blacksquare \\ & \\ & \end{bmatrix} \quad A_{si}^T = \begin{bmatrix} & \\ & \\ \bullet & \\ \bullet & \\ & \\ & \bullet \\ & \bullet \\ & \\ & \end{bmatrix} \quad A_{ss} = \begin{bmatrix} \bullet & \\ & \bullet \end{bmatrix}. \tag{21}$$

As in figure 1, red stars, blue diamonds and green squares refer to the internal grid point of the three processors, whereas grey circles indicate the shared interface grid points. As shown in the previous equation, A_{ii} is tridiagonal and A_{ss} is diagonal, while both A_{is} and A_{si} are extremely sparse.

Using block Gaussian elimination, the reordered linear system can be reduced to triangular form. Then, the solution reduces to the successive solution of the following linear systems

$$S u_s = (A_{ss} - A_{si} A_{ii}^{-1} A_{is}) u_s = f_s - A_{si} A_{ii}^{-1} f_i, \tag{22}$$

$$A_{ii} u_i = f_i - A_{is} u_s, \tag{23}$$

S is the Schur complement, and in the above example with $N_{px} = 3$ and $N_{sx} = 2$ its pattern is

$$S = \begin{bmatrix} \bullet & \bullet \\ \bullet & \bullet \end{bmatrix}. \tag{24}$$

By virtue of the sparse pattern of the matrix blocks, the Schur complement S is also tridiagonal for the present discretisation, although this is not evident in the present example with two shared interface unknowns only.

As a result, the equations for the shared interface points (22) and for the internal points (23) become decoupled. Furthermore, the equations for the internal points of different blocks are decoupled too. Indeed, owing to the tridiagonality of the original linear system and the locality of the finite-difference operators, A_{ii} is a block-diagonal matrix (see equation (21)). Therefore, in the present algorithm, only tridiagonal linear system must be solved.

The tridiagonal systems (22) and (23) are solved by Thomas' algorithm [51] which is equivalent to a banded LU decomposition without pivoting [52] when the coefficients are saved in the forward step. This is convenient in the present case, since the matrices do not vary in time, and can therefore be factored once and for all at the beginning of the computation. The internal linear systems have to be solved twice, the first time to compute the right-hand side of the linear systems for the shared interface unknowns, right-hand side in (22), the second time to compute the internal ones by solving (23).

The procedure is explained in more detail in algorithm 1. The tridiagonal matrix $S :=$

Algorithm 1: Procedure performed by each processor to solve tridiagonal linear systems in parallel. Black circles denote local computations, while the red square denotes computations that need communication between processors.

```
Preprocessing
    ○ Assemble the tridiagonal matrix $S := (A_{ss} - A_{si}A_{ii}^{-1}A_{is})$
Runtime
    ○ Compute right-hand side for internal points $f_i$
    ○ Compute processor's contribution to the right-hand side of the equation for the shared
      interface unknowns: $f_s - A_{si}A_{ii}^{-1}f_i$
    □ Assemble via communication all processors' contributions to $f_s - A_{si}A_{ii}^{-1}f_i$
    ○ Solve for shared interface unknowns $u_s$
    ○ Compute $f_i - A_{is}u_s$
    ○ Solve for internal unknowns $u_i$
```

$(A_{ss} - A_{si}A_{ii}^{-1}A_{is})$ of size $N_{sx} \times N_{sx}$ is computed in the preprocessing stage and stored on each processor. Then for each time step, after computing the right-hand sides for the internal grid points f_i, each processor computes the contribution to the the right-hand sides of the equations for the shared interface unknowns, $f_s - A_{si}A_{ii}^{-1}f_i$, associated with its portion of the stencil; then all the contributions are assembled on each processor via communication. Once the linear system (22) is assembled, it can be solved on each processor to minimise the required communication during time advancement. Indeed, for the usual fluid dynamic problems, the size of the Schur complement is much lower than the dimension of the internal problems: $N_s \ll N_i$. Finally, each processor evaluates locally its internal unknowns without the need of communication by solving the tridiagonal system (23), in which the equations for internal points of different blocks are decoupled, as said above.

With this procedure, the communication is kept to a minimum. It is only needed when assembling the right-hand side of the equations for the shared interface points.

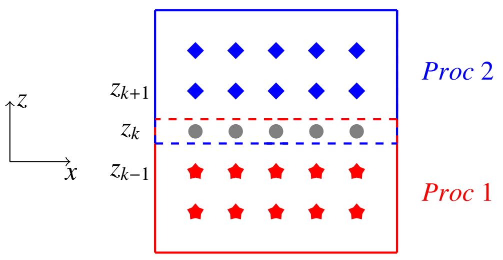

Figure 2: Grid decomposition for parallel execution in the $x - z$ plane. The domain is divided into two blocks in the z direction. Red stars denote the internal grid points for block/processor 1; blue diamonds denote the internal grid points for block/processor 2; grey circles denote the shared interface grid points.

4.2. Parallel treatment of the right-hand side

For simplicity, in this section we consider the domain to be decomposed into two blocks in the z direction; see figure 2. As said above, at each time step the following linear systems, dealing with the x, y and z directions, have to be solved for both the momentum and the pressure steps:

$$\begin{cases} A_x\psi = f, \\ \text{b.c.} \end{cases} \qquad \begin{cases} A_y\varphi = \psi, \\ \text{b.c.} \end{cases} \qquad \begin{cases} A_z\phi = \varphi. \\ \text{b.c.} \end{cases} \tag{25}$$

However, at the shared interface points (the grey points in figure 2) the right-hand side f_i for the system dealing with the x direction cannot be fully computed in any of the two processors sharing the interface, since the needed stencil is distributed across the two processors. Indeed, as long as we want to minimise communication, the communication should only occur when strictly necessary, i.e. when assembling the right-hand side of the Schur complement, in this case when solving the z direction. For this reason, we avoid the communication of the missing information necessary to build the right-hand side of the aforementioned linear systems on the two processors. As it will become clear in a moment, this is not necessary. We show as an example the component of the pressure gradient in the z direction evaluated on a homogeneous grid, i.e.

$$\frac{\partial p_k^n}{\partial z} := \underbrace{-\frac{p_{k-1}^n}{\delta z}}_{Processor\ 1} + \underbrace{\frac{p_{k+1}^n}{\delta z}}_{Processor\ 2} \ ; \tag{26}$$

where the index k denotes a shared interface point. Without any communication, the first term of the right-hand side can be evaluated only by processor 1, whereas the second one only by processor 2.

We therefore leverage the linearity of the equation and split the solution at the shared interface points among the processors. In detail, we write:

$$f_i = f_{i,1} + f_{i,2} \tag{27}$$

where $f_{i,1}$ and $f_{i,2}$ denote the contributions to f_i evaluated with the portion of the stencil available on the two processors, respectively. In doing this one has to pay attention not to consider the contribution from the shared interface point twice. Considering the above example, the first term of equation (26) goes into $f_{i,1}$, whereas the second term into $f_{i,2}$.

Therefore, when we solve for the shared interface points along the x and y directions, the two processors compute:

$$\begin{cases} A_x\psi_{i,1} = f_{i,1}, \\ \text{b.c.}|_1, \end{cases} \qquad \begin{cases} A_x\psi_{i,2} = f_{i,2}, \\ \text{b.c.}|_2, \end{cases} \tag{28}$$

$$\begin{cases} A_y\varphi_{i,1} = \psi_{i,1}, \\ \text{b.c.}|_1, \end{cases} \qquad \begin{cases} A_y\varphi_{i,2} = \psi_{i,2}, \\ \text{b.c.}|_2, \end{cases} \tag{29}$$

respectively, where $\text{b.c.}|_1 + \text{b.c.}|_2 = \text{b.c}$; this requirement can be met, for instance, by imposing the actual boundary conditions when solving on one processor and homogeneous conditions on the other one.

It turns out that, once the previous linear systems have been solved for the shared interface points along the x and y directions, the two processors computed

$$\varphi_{i,1} = A_y^{-1}A_x^{-1}f_{i,1}$$

and

$$\varphi_{i,2} = A_y^{-1}A_x^{-1}f_{i,2},$$

respectively. Finally, when the z direction is considered, which is normal to the interface in this example, the Schur complement is used. As already mentioned, at this stage the right-hand side of the equations for the interfaces is built assembling the contributions from the different processors. Therefore, the shared interface unknown results in

$$\phi_i = A_z^{-1}\left(\varphi_{i,1} + \varphi_{i,2}\right) = A_z^{-1}A_y^{-1}A_x^{-1}\left(f_{i,1} + f_{i,2}\right) = A_z^{-1}A_y^{-1}A_x^{-1}f_i. \tag{30}$$

Therefore, this procedure requires to solve the tridiagonal systems for the shared interface points on both processors sharing them, which is a small overhead necessary to limit the inter-processor communication to the construction of the right-hand side of (22).

5. Numerical Results

In this Section, we assess the accuracy and efficiency of the method with some example problems. First, the spatial and temporal convergence of the scheme is investigated for a manufactured solution of the Navier–Stokes equations, i.e. a solution obtained by introducing a suitable body force in the right-hand side of the equations [53]. Then, three classic, well-documented test cases are presented to describe the capabilities of the algorithm: one with two periodic directions, one with a single periodic direction, and two with no periodic directions: the driven cavity problem (internal flow) and the flow around a rectangular plate (external flow).

5.1. Convergence tests

In order to investigate temporal and spatial convergence, the algorithm has been tested numerically on the following manufactured solution of the Navier–Stokes equations:

$$u = \sin(x)\cos(t+y)\sin(z), \quad v = \cos(x)\sin(t+y)\sin(z), \quad w = 2\cos(x)\cos(t+y)\cos(z),$$
$$p = \frac{3}{\text{Re}}\cos(x)\cos(t+y)\cos(z). \tag{31}$$

The body-force field $\boldsymbol{f}$ that closes the momentum equation for this manufactured solution reads:

$$f_x = \frac{\sin(x)}{\mathrm{Re}}\Big[\frac{1}{2}\left(\mathrm{Re}\cos(x)\left(1 + 2\cos\left(2(t+y)\right) + \cos(2z)\right)\right) \\ - 3\cos(t+y)\left(\cos(z) - \sin(z)\right) - \mathrm{Re}\sin(t+y)\sin(z)\Big],$$

$$f_y = \frac{1}{4\,\mathrm{Re}}\Big[\mathrm{Re}\ \cos^2(x)(3+\cos(2z))\sin(2(t+y)) - 2Re\ \sin^2(x)\sin(2(t+y))sin^2(z) \\ + 4\cos(x)\Big(-3\cos(z)\sin(t+y) + (\mathrm{Re}\ \cos(t+y) + 3\sin(t+y))\sin(z)\Big)\Big],$$

$$f_z = \frac{1}{\mathrm{Re}}\Big[\cos(x)\Big(-2\mathrm{Re}\ \cos(z)\sin(t+y) + \cos(t+y)(6\cos(z) - 3\sin(z))\Big) \\ - \frac{1}{2}\Big(\mathrm{Re}\ \cos^2(x)(3+\cos(2(t+y)))\sin(2z)\Big) - \mathrm{Re}\ \cos^2(t+y)\sin^2(x)\sin(2z)\Big].$$

The problem is solved in the cubic domain $\Omega = (0,6)\times(0,6)\times(0,6)$ for $0 \le t \le T = 1$ and a Reynolds number $\mathrm{Re} = 1$. Dirichlet boundary conditions are used for the velocity and are given by the exact solution calculated on the boundary. The initial condition is the exact solution at $t = 0$. Temporal and spatial accuracies of the algorithm are tested on five uniform grids of varying size, i.e. 10^3, 20^3, 40^3, 80^3 and 160^3 points, with seven values of the time step, i.e. $\Delta t_n = 0.1 \times 2^{-n}$, where n is an integer in the interval $[0,6]$.

The L^2-norm of the velocity and pressure error is plotted in figure 3 and 4 for $\chi = 0$ and $\chi = 1$, respectively, and a comparison with the staggered algorithm presented in [24] is also reported, for the same size of the grid and time step. The left panels concern the spatial convergence; here the time step is fixed at $\Delta t = 0.003125$. They show that both the velocity and the pressure approximations are second-order accurate, as expected. In the right panels, the time accuracy is considered; here the grid size is fixed at the smallest value $\Delta x = 0.0375$, to keep the spatial error below the temporal error, at least for the largest time steps considered. The results show that both approximations, for velocity and pressure, are second-order accurate. Interestingly, the convergence curves show that the pressure approximation converges faster for $\chi = 0$. The velocity and pressure approximations seem to be slightly more accurate when the staggered version of the algorithm is used.

We tested the presence of spurious pressure modes for very small time steps. We advanced the solution in time for 200 time steps according to what has been proposed in [33]. The pressure-error field, reported in figure 5 for three different values of Δt, shows no evidence of spurious pressure modes.

5.2. *Scalability*

The scalability of the computer code implementing the method has been tested using the previously described manufactured solution. The simulations have been run on the GALILEO supercomputer at CINECA. It features 1022 36-core computing nodes connected through an Intel OmniPath (100Gb/s) high-performance network. Each node contains two 18-cores Intel Xeon E5-2697 v4 (Broadwell) at 2.30 GHz. All the computing nodes have 128 GB of memory. In figure 6 we show the scalability tests of the presented code. The left panel shows the strong scalability test. It consists in fixing the overall number of grid points and measuring the CPU time as the number of processing cores N_p increases ($N_p = 16, 32, 64, 128, 256, 512, 1024$ is considered). The figure shows a better-than-ideal behaviour. Indeed, The CPU time per timestep and grid point goes from $5.78 \cdot 10^{-7}$ s for 16 processors to $4.24 \cdot 10^{-7}$ s for 1024 processors,

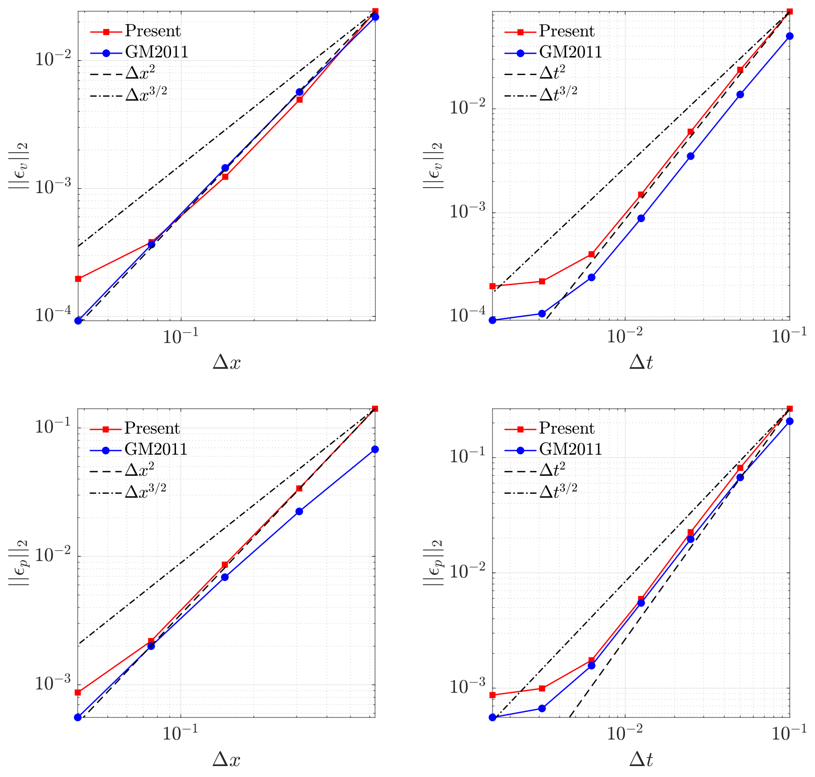

Figure 3: Standard form, i.e. ($\chi = 0$). L^2-norm of the error at $T = 1$ on uniform grids with respect to the manufactured solution given in eq. (31) for Re = 1, compared with the results obtained using the code of Ref. [24]. Top: velocity; bottom: pressure. Left: spatial convergence; right: temporal convergence.

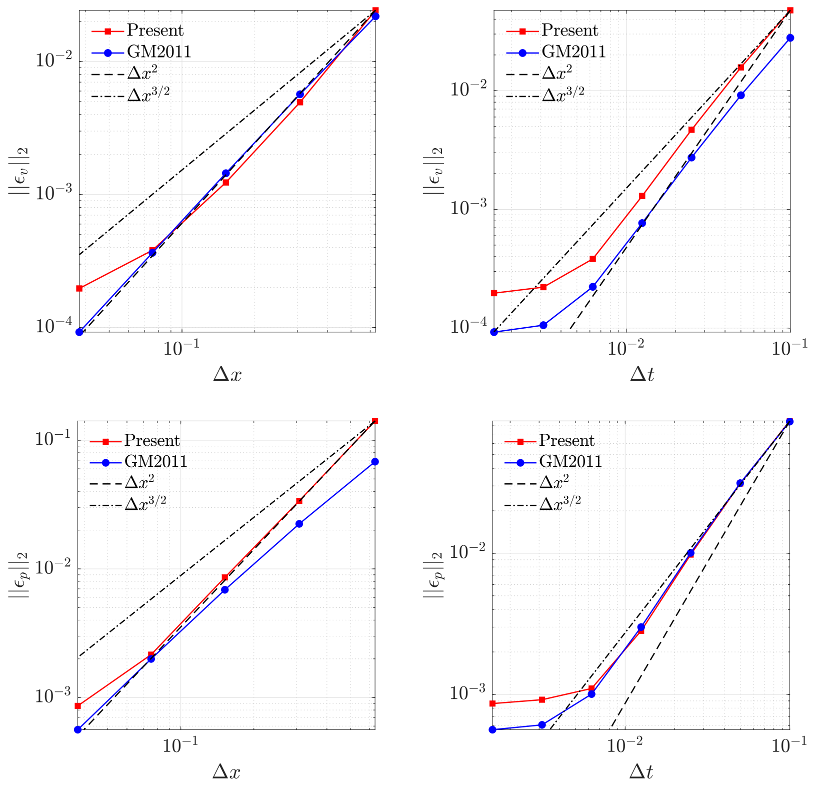

Figure 4: Rotational form, i.e. (χ = 1). L^2-norm of the error at $T = 1$ on uniform grids with respect to the manufactured solution given in eq. (31) for Re = 1, compared with the results obtained using the code of Ref. [24]. Top: velocity; bottom: pressure. Left: spatial convergence; right: temporal convergence.

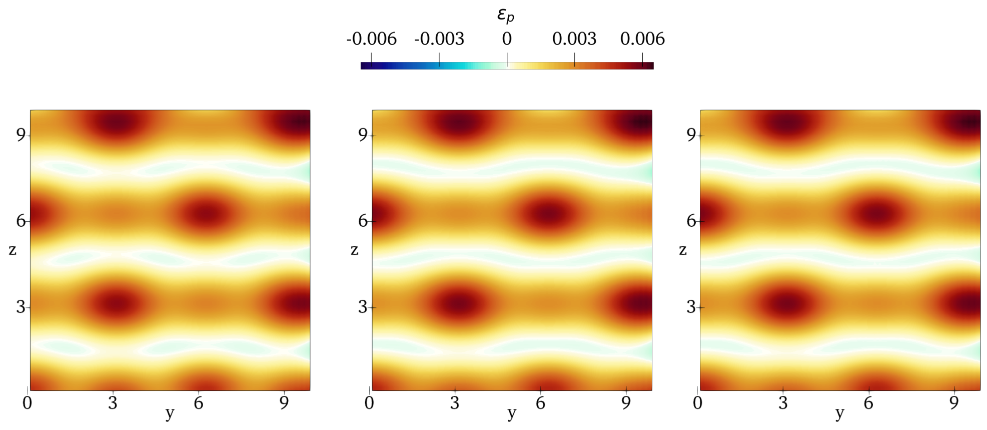

Figure 5: Pressure error in the $X = 5$ plane for the problem with manufactured solution (31) after 200 time steps. For this simulation the domain is $\Omega = (0,10) \times (0,10) \times (0,10)$ with $\Delta x = \Delta y = \Delta z = 0.1$; the Reynolds number is set to $Re = 1$. Left: $\Delta t = 10^{-4}$; centre: $\Delta t = 10^{-5}$; right: $\Delta t = 10^{-6}$

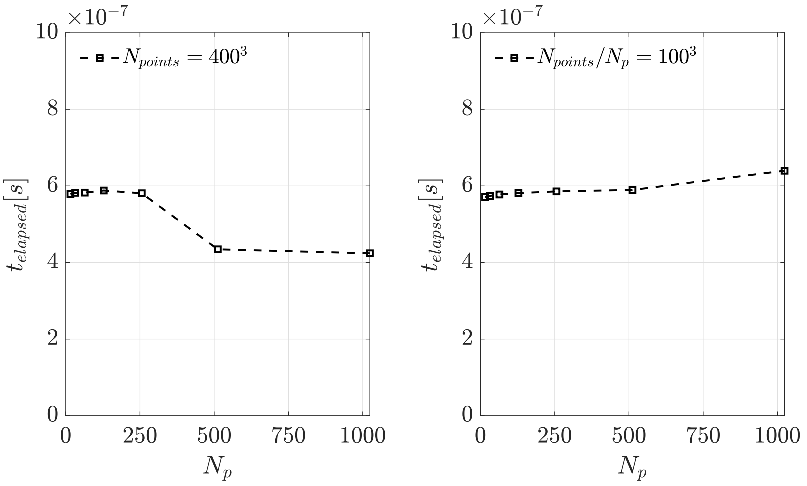

Figure 6: Scalability tests: CPU time per timestep and grid point versus the number N_p of parallel processors. Left: strong scalability test, with total number of grid points fixed to $N_{points} = 400^3$. Right: weak scalability test, with $N_{points}/N_p = 100^3$.

with a decrease for $N_p > 256$. A preliminary analysis of the code performance suggests that the code is memory-bound, i.e. the performance is limited by the memory bandwidth [54]. Indeed, a high speed-up is observed using a single core per node, irrespective of the number of nodes. This observation explains the present results for strong scalability, since for smaller local problems a larger portion of the data fits into higher-level cache memory, thus increasing the available memory bandwidth for data transfer.

The right panel, instead, shows the results for a weak scalability test. In this case the number of grid points per processor is fixed. The figure shows that the CPU time per timestep divided per overall number of grid points goes from $5.707 \cdot 10^{-7}$ s for 16 processors to $6.397 \cdot 10^{-7}$ for 1024 processors. This is a very good result, considering that the code, although carefully written, still has to undergo a detailed optimization of the parallel communication, by for instance overlapping computation and communication.

5.3. Comparison with established benchmarks

In this section we compare the accuracy of our code with that of well established results.

5.3.1. 3D driven cavity

The first validation test consists in solving the three-dimensional lid-driven cavity problem in the domain $\Omega = (0,1) \times (0,1) \times (-1,1)$ (in the x, y and z direction, respectively) at $\text{Re} = V_w h/\nu = 1000$, where V_w denotes the driving lid velocity, h the length of the cavity edge in the x and y directions, ν the kinematic viscosity of the fluid. This test has been chosen since it is an established test case and a benchmark solution obtained with the staggered version of the present algorithm [55] is available online. Here, quantities are made dimensionless with V_w and h. The driving lid is the side wall at $x = 1$, where $\boldsymbol{u} = (0, 1, 0)$ is the imposed velocity, while homogeneous Dirichlet boundary conditions are imposed on all the other walls. Two grids consisting of $200 \times 200 \times 400$ and $400 \times 400 \times 800$ in the x, y, z directions, respectively, have been

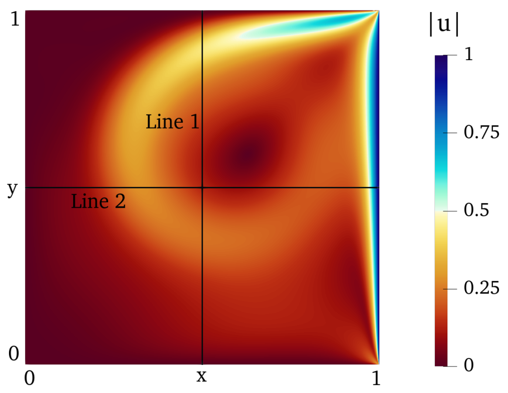

Figure 7: Three-dimensional driven cavity: velocity magnitude in the $z = 0$ plane at time $t = 8$, Re = 1000.

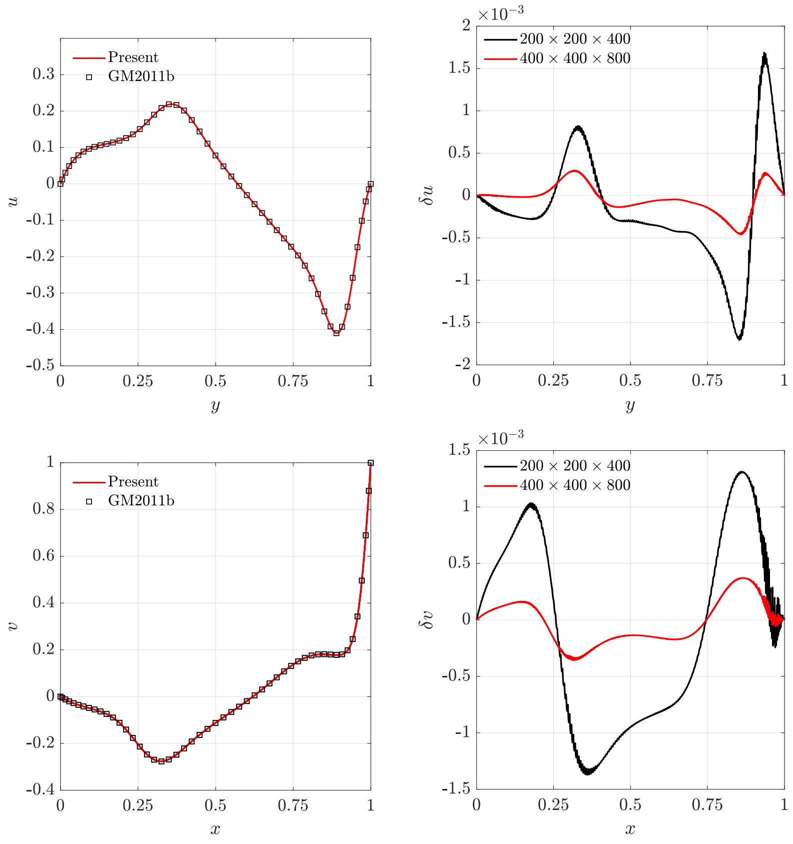

Figure 8: Three-dimensional driven cavity: solution at $t = 8$, Re = 1000. Top: x-component of the velocity along the vertical line in the $z = 0$ plane passing through the point $(x, y, z) = (0.5, 0.5, 0)$, see line 1 in figure 7. Bottom: y-component of the velocity along the horizontal lines in the $z = 0$ plane passing through the point $(x, y, z) = (0.5, 0.5, 0)$, see line 2 in figure 7. Left: velocity components. Right: difference between the present results and that reported in [55].

used to show the convergence. As in [55], if N_i denotes the number of points in the i–direction, the coordinates of the grid points are defined as:

$$C_i = 1 + (2/L_i)^{\frac{1}{2}}, \tag{32}$$

$$X_k = \frac{k-1}{N_i - 1} L_i \quad k = 1, .., N_i, \tag{33}$$

$$x_k = \begin{cases} C_i X_k^{\frac{3}{2}} (1 + X_k^{\frac{1}{2}})^{-1} & \text{if } X_k \leq L_i/2, \\ L_i - C_i (L_i - X_k)^{\frac{3}{2}} (1 + (L_i - X_k)^{\frac{1}{2}})^{-1} & \text{otherwise,} \end{cases} \tag{34}$$

to properly resolve the boundary without excessive stretching of the grid cells. The simulation has been performed using the Crank–Nicolson time scheme with an average CFL ~ 0.4, slightly below the stability limit of this scheme. The two spatial discretisations have been used alternatively to verify the accuracy of both near their stability limit.

Figure 7 is a colour plot of the magnitude of the velocity vector in the $z = 0$ plane at $t = 8$. The figure portraits a qualitatively correct solution: one clearly appreciates the high-speed region near the moving lid, as well as a high-speed region in the shear layer produced near the downstream corner of the lid. The figure also shows the main vortex that is forming near the centre of the cavity and the low-speed region near the side opposite to the moving lid. Figure 8 shows the u and v velocity components along the vertical and horizontal lines in the plane $z = 0$ passing through the point $x = 0.5$, $y = 0.5$ and $z = 0$ (see lines 1 and 2 in figure 7) at time $t = 8$. This allows a quantitative comparison with the results by Guermond and Minev contained in Ref. [55], which are also available online. Figure 8 shows that they are in very good agreement; the differences among the two simulations, shown in the right column, are in average less than one percent for both u and v. These differences become smaller if the finer grid is used, the ratio between the differences being consistent with the expected second-order convergence.

5.3.2. 3D Periodic driven cavity

The second validation test case consists in solving the three-dimensional flow in a square lid driven cavity in $\Omega = (-0.5, 0.5) \times (-0.5, 0.5) \times (-0.5, 0.5)$, with the moving wall placed at $x = 0$ and the periodic boundary conditions in the z direction. With respect to the previous one, this test case involves periodic boundary conditions and a steady-state solution. We compare the solution obtained with the present numerical method with the reference solution reported in [56] that has been obtained with a different numerical approach, a projection time-advancement scheme and a Chebyshev spectral method for the spatial discretisation with subtraction of the singularity to improve the accuracy. As in [56], the Reynolds number $Re = V_w h/\nu$ is again set to 1000, and the grid consists of $96 \times 96 \times 96$ points. Along the x directions the grid points are distributed as:

$$x_i = -\frac{1}{2} \cos\left(\frac{(i-1)\pi}{N_x - 1}\right) \quad i = 1, .., N_x, \tag{35}$$

where N_x denotes the number of points. The same grid is used also in the y direction. In the z direction an uniform grid is used, with the points distributed as:

$$z_k = \frac{1}{2}\left(\frac{2(k-1)}{N_z - 2} - 2\right) \quad k = 1, .., N_z. \tag{36}$$

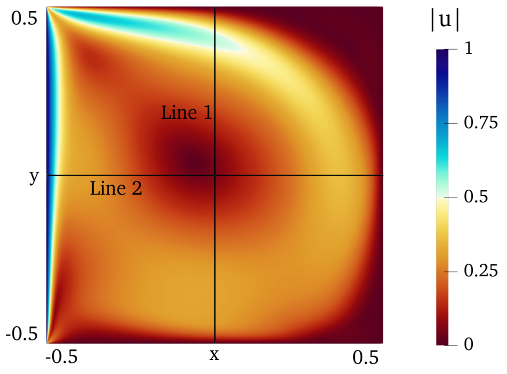

Figure 9: Three-dimensional periodic driven cavity test. Velocity magnitude of the steady solution in the $z = 0$ plane for Re = 1000.

The simulation is advanced with the RK-RAI3 time scheme described in the appendix with an average $CFL \sim 1$ until the steady-state solution is reached. As a criterion to ensure convergence to the steady solution, the following condition is used:

$$\frac{\max|u_i(\boldsymbol{x},t) - u_i(\boldsymbol{x},t-\Delta t)|}{\Delta t} < 10^{-7}. \tag{37}$$

Figure 9 shows the magnitude of the velocity in the $z = 0$ plane. The picture is qualitatively similar to that shown for the start-up of the 3D cavity, with two regions of high speed and the vortex. In this steady case, the vortex is more extended, and the recirculation regions in the corners opposite to the moving lid can be recognised. A separation region forms also on the lower horizontal side, not far from the intersection with the moving lid. To validate the results, figure 10 plots u, v and p along the horizontal and vertical lines in the $z = 0$ plane passing through $x = 0$ and $y = 0$; see figure 9, with superimposed the results taken directly form the paper of [56] shown with open symbols. Again, a complete agreement is found as the two results perfectly overlap.

5.3.3. Optimal perturbations in plane Couette flow

A further validation test involves two periodic directions. It concerns the amplification of optimal transient perturbations in plane Couette flow. For this purpose the work of [57] is considered as a reference and the main results replicated. The authors investigated the transition scenario in a Couette flow in which the transient growth mechanism is initiated by four decaying modes, showing that if their initial structure corresponds to counter-rotating vortex pairs, they are sufficient to capture the transient growth mechanism. Indeed, the energy growth based on non-linear DNS simulations for small initial amplitudes of the disturbances is in very good agreement with the linear analytical one based of the first four even modes.

In this test we replicate such work. We compute the energy growth with our program and we compare it with the analytical solution of the linearised Navier–Stokes equations based on the following velocity field:

$$\boldsymbol{u}(y,z,t) = \boldsymbol{U}_b(y) + \boldsymbol{u}_1(y,z,t) \tag{38}$$

where x, y and z denote the streamwise, wall-normal and spanwise directions; $\boldsymbol{U}_b = (0, y, 0)$ and

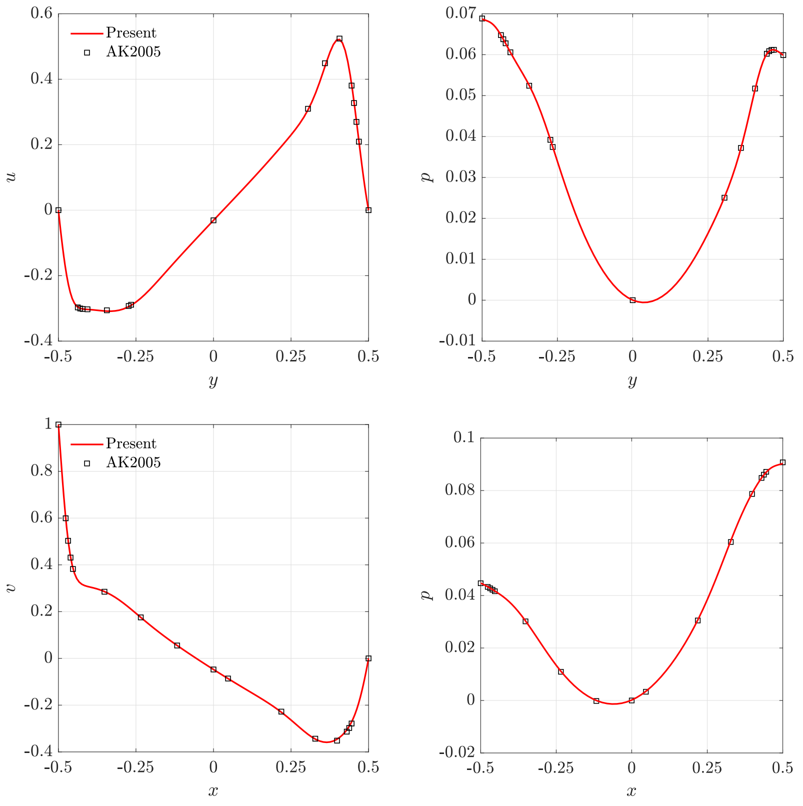

Figure 10: Three-dimensional periodic driven cavity test, steady state at Re = 1000. Top: x-component of the velocity (left) and pressure (right) along the vertical line in the $z = 0$ plane passing through the point $(x, y, z) = (0.5, 0.5, 0)$ (see line 1 in figure 9). Bottom: y-component of the velocity (left) and pressure (right) along the horizontal lines in the $z = 0$ plane passing through the point $(x, y, z) = (0.5, 0.5, 0)$ (see line 2 in figure 9). Results, computed at time $t = 8$, are compared with data in [56], denoted with white squares.

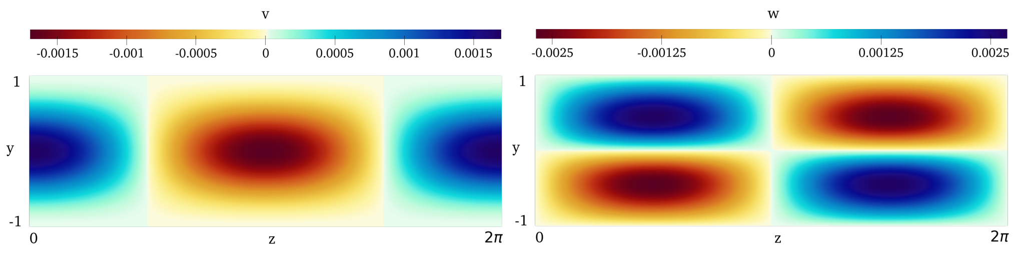

Figure 11: Two-dimensional visualisation of the initial cross-stream velocity field, consisting on a counter rotating vortex pair superimposed to the laminar Couette base flow. Left: wall-normal velocity component; right: spanwise velocity component.

$\boldsymbol{u}_1$ as:

$$\begin{aligned} u_1(y,z,t) =& \Omega \mathrm{Re} \mathrm{A}_0 [\mathrm{A}_1 \mathrm{F}_{sq1}(y) \mathrm{e}^{-i\omega_{sq1}t} + \mathrm{F}_{os1}(y)\mathrm{e}^{-i\omega_{os1}t} \\ &+ \mathrm{A}_2 \mathrm{F}_{sq2}(y) \mathrm{e}^{-i\omega_{sq2}t} + \mathrm{A}_3 \mathrm{F}_{os2}(y) \mathrm{e}^{-i\omega_{os2}t}] \cos(\beta z) \end{aligned} \tag{39}$$

$$v_1(y,z,t) = \mathrm{A}_0[\mathrm{V}_{os1}(y)\mathrm{e}^{-i\omega_{os1}t} + \mathrm{A}_3 \mathrm{V}_{os2}(y)\mathrm{e}^{-i\omega_{os2}t}]\cos(\beta z) \tag{40}$$

$$w_1(y,z,t) = -\mathrm{A}_0[\mathrm{W}_{os1}(y)\mathrm{e}^{-i\omega_{os1}t} + \mathrm{A}_3 \mathrm{W}_{os2}(y)\mathrm{e}^{-i\omega_{os2}t}]\sin(\beta z) \tag{41}$$

where the expressions of ω, F, V and W and the values of A_1, A_2 and A_3 are given in [57].

The domain is $\Omega = (0, 2\pi) \times (-1, 1) \times (0, 2\pi)$ and the grid consists on $96 \times 96 \times 96$ points uniformly placed along the periodic directions and with a hyperbolic-tangent distribution along the y direction to provide enhanced resolution the near-wall regions:

$$y_i = y_{\min} + \frac{y_{\max} - y_{\min}}{2} \left[\frac{\tanh\left(a\left(\frac{2i}{N_y} - 1\right)\right)}{\tanh(a)} + \frac{y_{\max} - y_{\min}}{2} \right] \quad i = 1, N_y,$$

where $a = 1.2$, $y_{\min} = -1$, $y_{\max} = 1$ and N_y denotes the number of points in the wall-normal direction. The Reynolds number $Re = U_w h/\nu$ (where U_w is the velocity of each wall and h the half height of the channel) is set to 1000. The initial condition at $t = 0$ consists on a counter rotating vortex pair superimposed on the laminar Couette solution, with the perturbation field $\boldsymbol{u}_1$ defined with $\beta = 1$, see figure 11. Three values of A_0 have been considered: $\mathrm{A}_0 = 0.0001$, $\mathrm{A}_0 = 0.001$ and $\mathrm{A}_0 = 0.002$. The simulation is advanced for $T = 400h/U_w$ with the Crank–Nicolson time scheme with an average $CFL \sim 0.1$. We have used the same averaged CFL as in [57]. Figure 12 compares the energy growth measured by the present code with the analytical curve, shown with open symbols. Our simulation is in very good agreement with the results reported in [57] (see figure 3 in their paper): the DNS with the smallest A_0 follows the predicted curve. This is because the reference solution is obtained for the linearised equations, and therefore it is exact in the limit of infinitesimal perturbations. For this reason a small gap between the DNS and the predicted results can be observed close to the maximum of the curve. Such difference, also visible in the pseudo-spectral DNS results reported in [57], decreases as the amplitude A_0 is decreased as expected, see the right panel in figure 12.

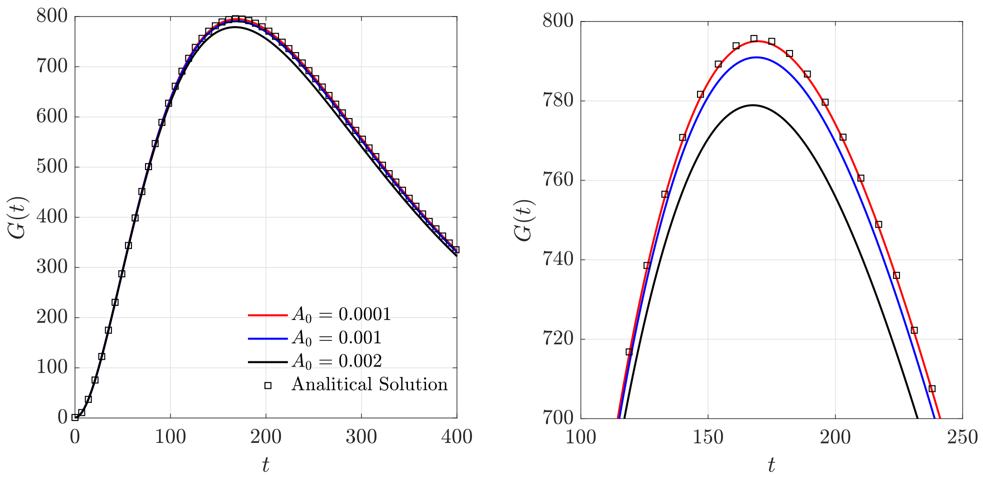

Figure 12: Couette flow test. Energy growth curve for $Re = 1000$ and $\beta = 1$ Comparison between the analytical curve based on four even modes as in [57] and our non-linear DNS results. Left: complete curve; right: zoom of the maximum.

5.4. Wake instability of a 1:6 aspect ratio flat-plate oriented perpendicularly to the incoming flow

The last validation consists in investigating the destabilisation of the three-dimensional steady wake developing behind a thin rectangular plate oriented perpendicularly to the uniform incoming flow. Here, x is the streamwise direction, whereas y and z denote the two cross-stream directions. The thickness-to-width ratio of the plate is $L_x/L_y = 1/6$, while the length to width ratio in $L_z/L_y = 6$. The work of Marquet and Larsson [2], where this problem is addressed via linear stability analysis and finite elements, is taken as reference. The presence of the flat plate is dealt with by an explicit immersed-boundary method [58]. At each time iteration, after the viscous step and before the pressure step, the velocity is linearly interpolated on the points nearest to the boundary. The Reynolds number based on the undisturbed incoming flow and on L_y, $Re = U_\infty L_y/\nu$, is set to 60, which is just above the first critical Reynolds number for such flow configuration according to [2]. The size of the computational domain is $-15 \le x \le 20$, $-20 \le y \le 20$ and $-20 \le z \le 20$ with the geometric centre of the plate placed at $(x, y, z) = (0, 0, 0)$. The grid consists of $272 \times 376 \times 448$ points in the three directions, with 50, 100 and 200 points over the plate sides in the x, y and z directions, respectively. A geometric progression is adopted to distribute the points:

$$
\begin{aligned}
x_i &= x_{i-1} + k_x^{i-1}\Delta x_1\\
y_j &= y_{j-1} + k_y^{j-1}\Delta y_1\\
z_k &= z_{k-1} + k_z^{k-1}\Delta k_1
\end{aligned}
$$

where $\Delta x_1 = 0.00335$, $\Delta y_1 = 0.00494$ and $\Delta z_1 = 0.0032751$ indicate the x- y- and z-size of the cells at the corners of the plate in the three directions; k_x is 1 for $0 \le x \le 1/6$ and 1.05 otherwise; $k_y = 1.04$; k_z is 1.03 for $-3 \le z \le 3$ and 1.04 otherwise. The simulation is advanced with the Crank–Nicolson time scheme for a time interval of $630L_y/U_\infty$; the time step is set to $\Delta t = 0.00125$ which corresponds to a $CFL \approx 0.5$. The boundary conditions of the problem are

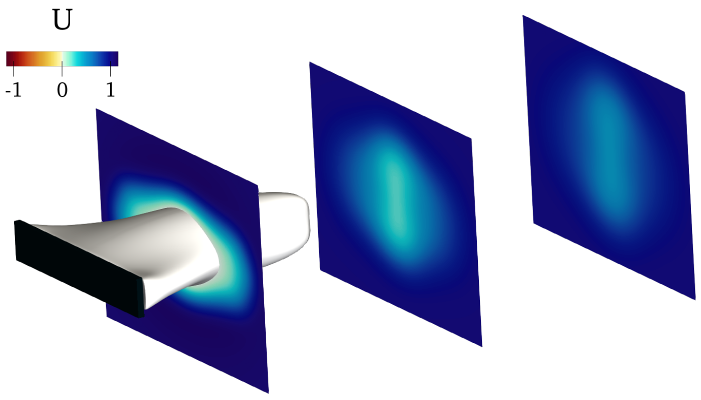

Figure 13: Three-dimensional visualisation of the steady three-dimensional at $Re = 60$. The white isosurface denotes $U = 0$ and highlights the recirculating region behind the plate. The color contours show the streamwise velocity at different stations down the wake.

the no-slip condition on the surface of the plate together with unperturbed velocity $(U_\infty, 0, 0)$ at the inlet, symmetry conditions at the far field in the y and z directions and a homogeneous Neumann condition for the velocity at the outlet.

In order to represent the shape of the unstable mode, the steady three-dimensional velocity field has been subtracted from an instantaneous velocity field of the unsteady nonlinear simulation. To compute such steady-state solution, corresponding to the baseflow in the linear stability analysis [2], we carried out an additional simulation inhibiting the onset of the instability by imposing the symmetry of the flow with respect to the two symmetry planes of the plate aligned with the free-stream velocity. Therefore, the simulation is carried out considering only $1/4$ of the overall computational domain, i.e. $-15 \leq x \leq 20$, $0 \leq y \leq 20$ and $0 \leq z \leq 20$, using symmetry boundary conditions on the planes $y = 0$ and $z = 0$. This simulation is carried out with the same parameters described above and is advanced in time until the steady state is reached, i.e. when the difference of the velocity between two successive iterations is $\leq 10^{-5}$. Figure 13 shows the three-dimensional base flow obtained after mirroring it with respect to the two symmetry planes. The white isosurface indicates $U = 0$ and highlights the recirculation region downstream the plate, whereas the contour of U is shown in different stations downstream the place. The figure closely recalls what found in [2]; indeed, both the $U = 0$ isosurface and the evolution of U in the wake have the same spatial evolution. To be more quantitative we found that the length of the recirculation region is $L_b \approx 6.34$ that is very close to the value of $L_b = 6.33$ found in [2], the difference being about 0.15%.

The simulation carried out on the complete domain reveals that at this Reynolds number the wake is unstable: an unstable mode is found to break the top/bottom symmetry. Figure 14 shows the power spectral density of the v velocity component computed at the point $(x, y, z) = (0.2189, 0.3458, 1.7400)$ via Fourier Transform. There is a clear spike, which reveals that the frequency of the unstable mode is $f \approx 0.0835$. This is again very close to the value $f = 0.084$ reported in [2]. The spatial distribution of the unstable mode is depicted in figure 15 with isosurfaces of positive (red) and negative (blu) streamwise velocity. As in [2], spatial structures of alternating sign are observed to develop in the x direction in both the shear layers. Such structures have opposite sign in the two shear layers as can be seen looking at the $y - x$ plane.

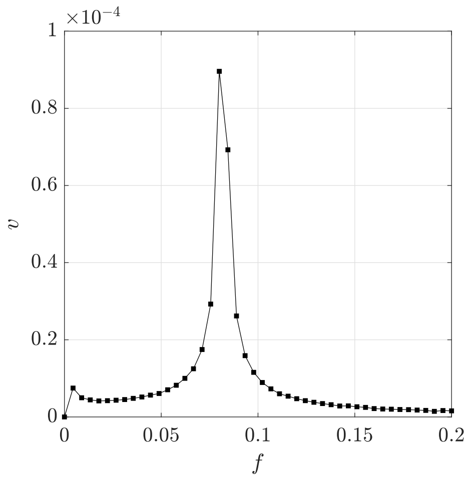

Figure 14: Power spectral density of the v velocity component computed at $(x, y, z) = (0.2189, 0.3458, 1.7400)$.

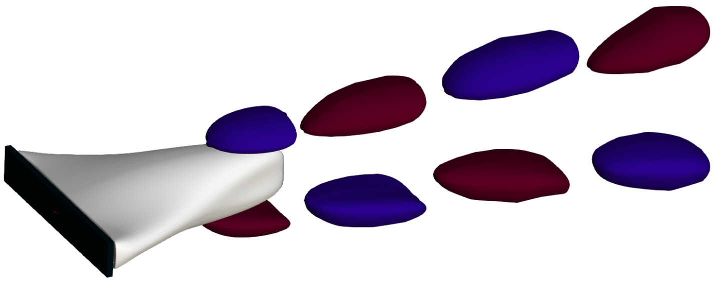

Figure 15: Three-dimensional visualisation of the unstable mode for $Re = 60$. Isosurfaces of positive (red) and negative (blu) streamwise velocity are shown in red and blue respectively. The isosurface of zero streamwise steady velocity is displayed in white.

6. Conclusions

We have presented an original finite-difference solver for the incompressible Navier–Stokes equations, based on a direction-splitting fractional step method to advance the equations in time. It extends the approach proposed by Guermond & Minev [24], with the important difference that a co-located grid is used instead of a staggered one. The main characteristics of the solver are: (i) the use of the direction-splitting technique for both the viscous and the pressure steps of the algorithm; (ii) the use of the Schur-complement method for the parallelisation.

The solver is extremely efficient from the computational viewpoint. Indeed, combining the direction-splitting technique with the discretization of derivatives through second-order central differences leads to a computer code whose kernel consists in solving tridiagonal linear systems, which takes place via the Thomas algorithm. The Schur-complement method allows a massive and effective parallelisation of such solutions and of the code as a whole; we have demonstrated very good parallel performance on thousands of processors, measuring a CPU time equal or less than 6×10^{-7} seconds per time step and per point, with both weak and strong scaling.

The most notable feature of this solver is the use of a co-located grid combined with direction splitting. This is an important step forward in view of its use combined with the immersed-boundary technique, to deal for example with situations with moving boundaries of complex shape, as in many fluid-structure interaction problems. In this case a co-located grid reduces the required number of interpolations in comparison with the staggered case, further contributing to the computational efficiency of the solver, while decreasing its algorithmic complexity. A common disadvantage of using co-located grids is the need to annihilate the spurious modes of the pressure. In the present solver, by virtue of the properties of the operator used in the penalty step where the pressure is computed, no spurious pressure modes are observed, and there is no need to add a stabilisation term in the formulation. Like other co-located methods, the present approach conserves energy only in the limit of Δt and Δx approaching zero. Therefore, the space and time resolution required for the direct numerical simulation of turbulent flows is higher than in standard solvers, making the present approach less interesting for this kind of flows.

The second-order convergence in space and time have been assessed by using the solver on a manufactured solution of the Navier–Stokes equations. Several tests have also been presented to check the accuracy of the code with different sets of boundary conditions and flow phenomena, including steady and time-dependent, internal and external flows .

References

[1] V. Theofilis, Advances in global linear instability analysis of nonparallel and three-dimensional flows, Prog Aerospace Sci 39 (4) (2003) 249–315.

[2] O. Marquet, M. Larsson, Global wake instabilities of low aspect-ratio flat-plates, Eur J Mech B Fluids 49 (2015) 400–412.

[3] L. S. Tuckerman, D. Barkley, Bifurcation analysis for timesteppers, in: E. Doedel, L. S. Tuckerman (Eds.), Numerical Methods for Bifurcation Problems and Large-Scale Dynamical Systems, Springer New York, New York, NY, 2000, pp. 453–466.

[4] V. Citro, F. Giannetti, P. Luchini, F. Auteri, Global stability and sensitivity analysis of boundary-layer flows past a hemispherical roughness element, Phys Fluids 27 (8) (2015) 084110.

[5] J.-C. Loiseau, M. A. Bucci, S. Cherubini, J.-C. Robinet, Time-stepping and Krylov methods for large-scale instability problems, in: A. Gelfgat (Ed.), Computational Modelling of Bifurcations and Instabilities in Fluid Dynamics, Springer International Publishing, Cham, 2019, pp. 33–73.

[6] M. D. de Tullio, A. Cristallo, E. Balaras, R. Verzicco, Direct numerical simulation of the pulsatile flow through an aortic bileaflet mechanical heart valve, J Fluid Mech 622 (2009) 259–290.

[7] A. Fani, S. Camarri, M. V. Salvetti, Investigation of the steady engulfment regime in a three-dimensional T-mixer, Phys Fluids 25 (6) (2013) 064102.

[8] H. Xu, S. M. Mughal, E. R. Gowree, C. J. Atkin, S. J. Sherwin, Destabilisation and modification of Tollmien–Schlichting disturbances by a three-dimensional surface indentation, J Fluid Mech 819 (2017) 592–620.

[9] M. A. Bucci, D. K. Puckert, C. Andriano, J.-C. Loiseau, S. Cherubini, J.-C. Robinet, U. Rist, Roughness-induced transition by quasi-resonance of a varicose global mode, J Fluid Mech 836 (2018) 167–191.

[10] P. F. Fischer, An overlapping schwarz method for spectral element solution of the incompressible Navier–Stokes equations, J Comp Phys 133 (1) (1997) 84–101.

[11] G. Karniadakis, S. Sherwin, Spectral/Hp Element Methods for Computational Fluid Dynamics, 2nd Edition, Oxford University Press, 2005.

[12] C. Canuto, M. Hussaini, A. Quarteroni, T. Zang, Spectral Methods in Fluid Dynamics, Springer-Verlag, Berlin Heidelberg, 1988.

[13] A. Gerstenberger, W. Wall, An Extended Finite Element Method/Lagrange multiplier based approach for fluid-structure interaction, Comput Methods Appl Mech Eng 197 (19) (2008) 1699–1714.

[14] S. Court, M. Fournié, A fictitious domain finite element method for simulations of fluid–structure interactions: The Navier–Stokes equations coupled with a moving solid, J Fluids Struct 55 (2015) 398–408. doi:10.1016/j.jfluidstructs.2015.03.013.

[15] X. Zhu, E. Phillips, V. Spandan, J. Donners, G. Ruetsch, J. Romero, R. Ostilla-Mónico, Y. Yang, D. Lohse, R. Verzicco, M. Fatica, R. J. Stevens, AFiD-GPU: A versatile Navier–Stokes solver for wall-bounded turbulent flows on GPU clusters, Comput Phys Commun 229 (2018) 199–210.

[16] P. Costa, A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows, Comput Math Appl 76 (8) (2018) 1853–1862.

[17] S. Chen, G. Doolen, Lattice Boltzmann method for fluid flows, Annu Rev Fluid Mech 30 (1) (1998) 329–364.

[18] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu Rev Fluid Mech 37 (1) (2005) 239–261.

[19] P. Orlandi, Fluid Flow Phenomena – a Numerical Toolkit, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[20] M. Frigo, S. Johnson, The design and implementation of FFTW3, Proceedings of the IEEE 93 (2) (2005) 216–231.

[21] M. Ertl, J. Reutzsch, A. Naegel, G. Wittum, B. Weigand, Towards the implementation of a new multigrid solver in the DNS code FS3D for simulations of shear-thinning jet break-up at higher reynolds numbers, in: High Performance Computing in Science and Engineering' 17: Transactions of the High Performance Computing Center, Stuttgart (HLRS) 2017, 2018, pp. 269–287.

[22] A. Gholami, D. Malhotra, H. Sundar, G. Biros, FFT, FMM, or multigrid? A comparative study of state-of-the-art poisson solvers for uniform and nonuniform grids in the unit cube, SIAM J Sci Comput 38 (3) (2016) C280–C306.

[23] J. Guermond, P. Minev, A new class of fractional step techniques for the incompressible Navier–Stokes equations using direction splitting, C R Math 348 (2010) 581–585.

[24] J. Guermond, P. Minev, A new class of massively parallel direction spliting for the incompressible Navier–Stokes equations, Comput Methods Appl Mech Eng 200 (2011) 2083–2093.

[25] J. Douglas, Alternating direction methods for three space variables, Numer Math 4 (1) (1962) 41–63. doi:10.1007/BF01386295.

[26] D. W. Peaceman, H. H. Rachford, Jr., The numerical solution of parabolic and elliptic differential equations, J Soc Indust Appl Math 3 (1) (1955) 28–41.

[27] W. Couzy, M. Deville, A fast Schur complement method for the spectral element discretization of the incompressible Navier–Stokes equations, J Comp Phys 116 (1) (1995) 135–142.

[28] S. Kocak, H. U. Akay, Parallel Schur complement method for large-scale systems on distributed memory computers, Appl Math Model 25 (10) (2001) 873–886.

[29] B. E. Moutafis, C. K. Filelis-Papadopoulos, G. A. Gravvanis, Parallel Schur complement techniques based on multiprojection methods, SIAM J Sci Comput 40 (4) (2018) C634–C654.

[30] O. Ladyzhenskaya, The Mathematical Theory of Viscous Incompressible Flow, Vol. 2 of Mathematics and Its Applications, Gordon and Breach Science Publishers, New York, NY, USA, 1969.

[31] I. Babuška, The finite element method with lagrangian multipliers, Numer Math 20 (1972) 179–192.

[32] F. Brezzi, On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers, Esaim Math Model Numer Anal 8 (R2) (1974) 129–151.

[33] J.-L. Guermond, L. Quartapelle, On stability and convergence of projection methods based on pressure Poisson equation, Int J Numer Methods Fluids 26 (9) (1998) 1039–1053.

[34] M. Vanella, E. Balaras, A moving-least-squares reconstruction for embedded-boundary formulations, J Comp Phys 228 (18) (2009) 6617–6628.

[35] M. de Tullio, G. Pascazio, A moving-least-squares immersed boundary method for simulating the fluid–structure interaction of elastic bodies with arbitrary thickness, J Comp Phys 325 (2016) 201–225.

[36] S. Popinet, Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries, J

Comp Phys 190 (2) (2003) 572–600.

[37] J. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, Springer-Verlag, Berlin Heidelberg, 2000.

[38] C. Rhie, W. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, AIAA J 21 (1983) 1525–1532.

[39] Y. Zang, R. L. Street, J. R. Koseff, A non-staggered grid, fractional step method for time-dependent incompressible Navier–Stokes equations in curvilinear coordinates, J Comp Phys 114 (1) (1994) 18–33.

[40] Shashank, J. Larsson, G. Iaccarino, A co-located incompressible Navier–Stokes solver with exact mass, momentum and kinetic energy conservation in the inviscid limit, J Comp Phys 229 (12) (2010) 4425–4430.

[41] S. Faure, J. Laminie, R. Temam, Colocated Finite Volume Schemes for Fluid Flows, Commun Comput Phys (2008) 25.

[42] P. Khongar, J. Pralits, P. Soleri, M. Romano, R. Repetto, A study of the mechanical forces on aphakic iris-fixated intraocular lenses, J Biomech Eng 140 (11) (Aug. 2018).

[43] L. Quartapelle, Numerical Solution of the Incompressible Navier–Stokes Equations, Birkhäuser, Basel, CH, 1993.

[44] J. Guermond, J. Shen, Velocity-correction projection methods for incompressible flows, SIAM J Numer Anal 41 (2003) 112–134.

[45] C. Hamakiotes, S. Berger, Fully developed pulsatile flow in a curved pipe, J Fluid Mech 195 (1988) 23–55.

[46] M. Braza, The 3D transition to turbulence in wake flows by means of direct numerical simulation, Flow Turbul Combust 63 (2000) 315–341.

[47] P. Gresho, Incompressible fluid dynamics: Some fundamental formulation issues, Annu Rev Fluid Mech 23 (1) (1991) 413–453.

[48] R. Temam, Navier-Stokes Equations: Theory and Numerical Analysis, no. 2 in Studies in Mathematics and Its Applications, North Holland, Amsterdam, 1979.

[49] S. Faure, Stability of a colocated finite volume scheme for the Navier–Stokes equations, Num Meth Partial Differential Eq 21 (2005) 242–271.

[50] M. Rai, P. Moin, Direct simulations of turbulent flow using finite-difference schemes, J Comp Phys 96 (1991) 15.

[51] L. Thomas, Elliptic problems in linear differential equations over a network, Tech. rep., Watson Science Computer Laboratory Report, Columbia University, New York, NY, USA (1949).

[52] G. Golub, C. Van Loan, Matrix Computations, 4th Edition, The Johns Hopkins University Press, Baltimore, MD, USA, 2013.

[53] P. J. Roache, Code verification by the method of manufactured solutions, J Fluids Eng 124 (1) (2001) 4–10.

[54] V. Eijkhout, Introduction to High Performance Scientific Computing, 2nd Edition, Lulu. com, 2016.

[55] J.-L. Guermond, P. Minev, Start-up flow in a three-dimensional lid-driven cavity by means of a massively parallel direction splitting algorithm, Int J Num Meth Fluids 68 (2012) 856–871.

[56] S. Albensoeder, H. Kuhlmann, Accurate three-dimensional lid-driven cavity flow, J Comp Phys 206 (2005) 536–558.

[57] M. Karp, J. Cohen, Tracking stages of transition in Couette flow analytically, J Fluid Mech 748 (2014) 896–931.

[58] E. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J Comp Phys 161 (1) (2000) 35–60.