Efficient Semi-Implicit Time-Stepping Schemes for Incompressible Flows

Kak Choon Loy

Thesis submitted to the Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Mathematics

[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The Ph.D. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

The development of numerical methods for the incompressible Navier-Stokes equations received much attention in the past 50 years. Finite element methods emerged given their robustness and reliability. In our work, we choose the $\mathbb{P}_2$-$\mathbb{P}_1$ finite element for space approximation which gives 2^{nd}-order accuracy for velocity and 1^{st}-order accuracy for pressure. Our research focuses on the development of several high-order semi-implicit time-stepping methods to compute unsteady flows. The methods investigated include backward difference formulae (SBDF) and defect correction strategy (DC). Using the defect correction strategy, we investigate two variants, the first one being based on high-order artificial compressibility and bootstrapping strategy proposed by Guermond and Minev (GM) and the other being a combination of GM methods with sequential regularization method (GM-SRM). Both GM and GM-SRM methods avoid solving saddle point problems as for SBDF and DC methods. This approach reduces the complexity of the linear systems at the expense that many smaller linear systems need to be solved. Next, we proposed several numerical improvements in terms of better approximations of the nonlinear advection term and high-order initialization for all methods. To further minimize the complexity of the resulting linear systems, we developed several new variants of **grad**-div splitting algorithms besides the one studied by Guermond and Minev. Splitting algorithm allows us to handle larger flow problems. We showed that our new methods are capable of reproducing flow characteristics (e.g., lift and drag parameters and Strouhal numbers) published in the literature for 2D lid-driven cavity and 2D flow around the cylinder. SBDF methods with **grad**-div stabilization terms are found to be very stable, accurate and efficient when computing flows with high Reynolds numbers. Lastly, we showcased the robustness of our methods to carry 3D computations.

In loving memory of

Mong Tee Lim, the sacrifice that you have made for your children, grandchildren and great-grandchildren is immeasurable.

Philippe Paradis, you were a very close friend of mine. Your presence had made my five-year stay in Ottawa more than enjoyable. I fondly remember that you were a brilliant, determined and inspiring mathematician who has a great potential to change the history of machine learning.

You will always be in our hearts...

Dedications

To my parents, Siew Wah Loy and Wai Wah Lee; to my dearest wife, Nyuk Sian Chong; to my siblings, Kak Wai and Yip Chun. To my parents-in-law, Yap Kiew Chong and Heng Hui Liaw; siblings-in-law, Nyuk Ling, Nyuk Yuin, Tze Huat and Tze Lung. Thank you for your undivided attention, most of all, your unconditional love and support since the first day of my PhD journey.

This humble work is the expression of my love to all of you.

Acknowledgment

I personally believe than even the lengthiest acknowledgment written would not suffice to thank every possible person, in many ways, who has helped me to make this thesis a reality.

First and foremost, I would like to thank my supervisor, Dr. Yves Bourgault for the knowledge he shares, his endless patience, his continuous guidance and his unconditional support–both moral and financial. He gave me a lot of great opportunities to participate in several prestigious workshops and conferences from one end to the other of Canada. I regard Dr. Yves as close friend of mine and someone I can easily talk to.

To my thesis committee members; the external examiner Dr. André Garon and the internal examiners Dr. Benoit Dionne, Dr. Victor LeBlanc and Dr. Emmanuel Lorin, thank you very much for your comments and valuable feedback. Without your contributions, this thesis could not be possibly improved to nearly perfect condition.

A special appreciation goes to the Director of Graduate Program, Dr. Benoit Dionne for his academic guidance. To us, he is the "law and order" of graduate regulations, all technicalities and important dates that all graduate students must follow. I also acknowledge his contribution as a web administrator (intranet) which serves as a one-stop information center. Without this useful website, all graduate students will have no sense of direction.

I must not forget to acknowledge my two sponsors: Universiti Malaysia Terengganu (also my current employer) and the Ministry of Education of Malaysia for the scholarship given. Without their financial and various administration support, my Ph.D studies at the University of Ottawa could not have been possible.

My deepest gratitute to many professors for the knowledge they had shared when I was taking course with them (for my Ph.D pre-requisite). They are Dr. David Amundsen, Dr. Emmanuel Lorin, Dr. Arian Novruzi, Dr. Natalia Stepanova, Dr. Ioana Ada Cosma and Dr. Raluca Balan.

Special thanks to the administrative staff of the Department of Mathematics and

Statistics Dr. Rafal Kulik (current Director), Dr. Monica Nevins (past Director), Dr. Mayor Alvo (past Director), Mayada El Maolouf, Chantal Giroux, Diane Demers, Carolynne Roy, Michelle Lukaszczyk, Jarnick Rainville and Rolland Fillion. They have strived to make the department the most conducive place to work. I am really thankful for their welfare, management and technical support given since my first day in the department.

There are several people in my life that I remember fondly who are my past mentors Prof. Dr. Abu Osman bin Md. Tap, Prof. Dr. Mohd Lokman bin Husain, Prof. Dr. Hock Lye Koh, Prof. Dr. Fredolin Tangang, Assoc. Prof. Dr. Juneng Liew and Prof. Prakash Sinha. Their past guidance had made me strong and shaped the person I am now. Thank you all!

Last but not least, I would like to thank many important individuals; for your great concern, constant motivation and kind support especially Mohamed Amari, Azidah Buang, Micheal Mcleod, Thomas Roy, Emily Campling, Jarno van der Kolk, Jeff Musgrave, Nguyen Thu Huong, Sana Keita, Edward Boey, Saint Cyr Koyaguerebo Ime, Diane Fokoue, Nicole Claudette and Yue Dong. To anyone I have missed out, please accept my apologies.

Contents

List of Figures

List of Tables

List of Abbreviations

a.e.	almost everywhere
ALM	Augmented Lagrangian Method
BDF	Backward Difference Formulae
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrich–Lewy
CNAB	Crank–Nicolson and Adam–Bashford
CPU	Central Processing Unit
DNS	Direct Numerical Simulation
FFT	Fast Fourier Transform
GALS	Galerkin Least-Square
GMRes	Generalized Minimal Residual
HIPS	Hierarchical Iterative Parallel Solver
ILU	Incomplete Lower-Upper Factorization
ILUT	Incomplete Lower-Upper Factorization with Threshold
IMEX	IMplicit-EXplicit
LES	Large Eddy Simulation
LBB	Ladyzhenskaya–Babuška–Brezzi
MOL	Method of Lines
MUMPS	MUltifrontal Massively Parallel Sparse direct Solver
NSE	Navier–Stokes Equations
ODE	Ordinary Differential Equation
PCG	Preconditioned Conjugate Gradient
PDE	Partial Differential Equation
PML	Perfectly-Matched-Layer
PSPG	Pressure Stabilized Petrov–Galerkin
RAM	Random Access Memory
Re	Reynolds number
SPD	Symmetric Positive Definite
Str	Strouhal number
SUPG	Streamline Upwind Petrov–Galerkin
UMFPACK	Unsymmetric MultiFrontal method

Chapter 1

Introduction: Modeling of Incompressible Flows

Numerical approximations for incompressible flows are one of the most studied topic in the realm of applied mathematics. The dynamic motion of fluids is usually governed by a set of partial differential equations (PDEs). In simplified problems, these PDEs can possibly be solved using analytical methods. In most flow problems, analytical solutions cannot be obtained because of the complexity of geometries and boundary conditions, as well as the presence of nonlinearity in the PDE. These flows can only be approximated using numerical methods, aiming for the best achievable accuracy. This chapter has demonstrated several preliminary computations to solve incompressible Navier–Stokes equations for both steady and non-steady flows. This is done using mixed finite element methods for space and various methods of lines for time approximations. From there, we will highlight issues pertaining to numerical accuracy, stability, convergence and efficiency. Finally, we suggest the best methods to compute solutions of steady and unsteady problems. This preliminary study subsequently provides motivation for the development of time-stepping methods which is the main contribution of this thesis. Before reaching the core discussion of this chapter, we provide a short discussion on the Stokes equations and steady Navier–Stokes equations which we feel are both important and useful to study such flows.

1.1 Stokes Equations

Stokes equations have been used to model creeping flows; i.e., laminar flows where inertial effects are negligible. These equations are named after George Gabriel Stokes,

the same mathematician who derived Stokes' law in 1851. The equations are

$$\begin{aligned} -\nu\Delta\mathbf{u} + \nabla p &= \mathbf{f}, \quad && \text{in } \Omega, \\ \nabla \cdot \mathbf{u} &= 0, && \text{in } \Omega, \\ \mathbf{u} &= 0, && \text{on } \Gamma. \end{aligned} \tag{1.1.1}$$

The first equation is called the momentum equation and the second is the equation for mass conservation, which are solved simultaneously in a connected domain Ω. For simplicity, we define homogeneous Dirichlet boundary condition along the boundary Γ of Ω. For three-dimensional flows, the velocity $\mathbf{u} = (u, v, w)$ and external volumic force $\mathbf{f} = (f_1, f_2, f_3)$ are vector fields while the pressure p is a scalar function. The parameter $\nu > 0$ is known as the viscosity coefficient which is assumed to be a fixed real number. When modeling flow of fluid having high viscosity $\nu = \mathcal{O}(1)$ is typically chosen. For mass conservation in the domain Ω, we assumed that the flow is incompressible hence the divergence of the velocity is equal to zero.

To solve (1.1.1), we used a well-known numerical technique known as the finite element methods. Finite element methods have been introduced in the early 1950s and are known to be indispensable for many problems arising in engineering, physics and mathematics. Because of their robustness to handle physical problems with complex geometries, finite element methods are capable of solving complex PDEs. Finite element methods are based on weak formulation of the PDE. After multiplying by a test function in a proper functional space and using Green's formula, the weak formulation for the Stokes equations is expressed as follows: Find the solution $\mathbf{u} \in V$ and $p \in Q$ such that

$$\begin{aligned} \int_\Omega \nu\nabla\mathbf{u} \cdot \nabla\mathbf{v}\,\mathrm{d}x - \int_\Omega p\nabla \cdot \mathbf{v}\mathrm{d}x &= \int_\Omega \mathbf{f} \cdot \mathbf{v}\,\mathrm{d}x, && \forall \mathbf{v} \in V, \\ -\int_\Omega q\nabla \cdot \mathbf{u}\,\mathrm{d}x &= 0, && \forall q \in Q. \end{aligned} \tag{1.1.2}$$

where V is the usual Sobolev functional space $[H_0^1(\Omega)]^d$ (i.e., $d = 2, 3$ is the dimension), the space consisting of all functions in $L^2(\Omega)$ whose first derivative (in the sense of distribution) is in $L^2(\Omega)$ and which vanishes a.e. on the boundary Γ. We also define $Q = L_0^2(\Omega)$, the space of L^2-functions with zero mean in Ω.

Introducing finite dimensional subspaces $V_h \subset V$ and $Q_h \subset Q$, the discrete weak formulation for Stokes equations is: Find the solution $\mathbf{u}_h \in V_h$ and $p_h \in Q_h$ such that

$$\begin{aligned} \int_\Omega \nu\nabla\mathbf{u}_h \cdot \nabla\mathbf{v}_h\,\mathrm{d}x - \int_\Omega p_h\nabla \cdot \mathbf{v}_h\,\mathrm{d}x, &= \int_\Omega \mathbf{f} \cdot \mathbf{v}_h\,\mathrm{d}x, && \forall \mathbf{v}_h \in V_h, \\ -\int_\Omega q_h\nabla \cdot \mathbf{u}_h\,\mathrm{d}x &= 0, && \forall q_h \in Q_h. \end{aligned} \tag{1.1.3}$$

To solve (1.1.3), mixed finite element methods are used. For instance, a "compatible" pair of finite element spaces for $\mathbf{u}_h$ and p_h is chosen to avoid spurious pressure modes. Spurious pressure modes show up in the form of "checkerboard" solutions on the pressure p_h, in particular when an "unhealthy" pair of finite elements is chosen (see [6, 22]). For example, this occurs when the same degree of piecewise polynomials are used for velocity and pressure on the triangular elements; i.e., $\mathbb{P}_k$-$\mathbb{P}_k$ for velocity-pressure on the rectangular elements, $\mathbb{Q}_k$-$\mathbb{Q}_k$, where $k \geq 0$. The pair of finite element spaces has to fulfill Ladyzhenskaya–Babŭska–Brezzi (LBB) or discrete inf-sup condition: There exists $\beta > 0$ independent of the grid size h such that

$$\inf_{q_h \in Q_h} \sup_{\mathbf{v}_h \in V_h} \frac{\int_\Omega q_h \nabla \cdot \mathbf{v}_h \, \mathrm{d}x}{\|q_h\|_{Q_h} \|\mathbf{v}_h\|_{V_h}} \geq \beta.$$

Besides the discrete inf-sup condition, the discrete space V_h for velocity must be richer than the discrete space Q_h for pressure to avoid a "locking phenomena" where the only incompressible velocity field $\mathbf{u}_h$ is the null field. Here, we choose the Taylor–Hood or $\mathbb{P}_2$-$\mathbb{P}_1$ elements which form a compatible pair to solve (1.1.3). Taylor–Hood elements are continuous piecewise Lagrange polynomials of degree 2 and 1 for the velocity and pressure, respectively. This finite element is popular since it satisfies the inf-sup condition and the implementation is straighforward. It is also fairly accurate, especially the velocity which has 2^{nd}-order accuracy. The pressure is only 1^{st}-order accurate but this is expected (pressure is less regular) and often sufficient. A detailed discussion on the compatibility of $\mathbb{P}_2$-$\mathbb{P}_1$ elements and the formal proof of the inf-sup condition can be found in [6, 7, 22, 48]. Along this direction, we solve (1.1.3) by considering the following discrete finite element spaces

$$\begin{aligned} V_h &= \{\mathbf{v}_h \in [\mathcal{C}^0(\overline{\Omega})]^d \mid \mathbf{v}_{h|K} \in [\mathbb{P}_2]^d, \forall K \in \mathcal{T}_h\}, \\ Q_h &= \{q_h \in \mathcal{C}^0(\overline{\Omega}) \mid q_{h|K} \in \mathbb{P}_1, \forall K \in \mathcal{T}_h\}. \end{aligned} \tag{1.1.4}$$

$\mathcal{T}_h$ is a regular triangulation of the domain (see Appendix B.1) using conforming mesh and we have

$$\overline{\Omega} = \bigcup_{K \in \mathcal{T}_h} K.$$

Each element K has the degrees of freedom (d.o.f.) shown on Figure 1.1 for velocity (left) and pressure (right). The mesh size h is defined as

$$h = \max_{K \in \mathcal{T}_h} \{\mathrm{diam}(K)\} \tag{1.1.5}$$

Figure 1.1: $\mathbb{P}_2$-$\mathbb{P}_1$ Finite Element

Solving (1.1.2) can be understood as minimizing the energy of the system over all $\mathbf{v} \in V$ subject to divergence free constraints. The pressure $p \in Q$ is then regarded as a Lagrange multiplier. In other words, any mixed finite element discretization of Stokes equations (1.1.3) produces a linear system of the form

$$\begin{bmatrix} A_h & B_h^T \\ B_h & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h \\ p_h \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \tag{1.1.6}$$

where the block matrix, $A_h \in \mathbb{R}^{n\times n}, B_h \in \mathbb{R}^{m\times n}, \mathbf{u}_h \in \mathbb{R}^n, p_h \in \mathbb{R}^m$ and $\mathbf{f} \in \mathbb{R}^n$. Here, m and n are the number of unknowns for p_h and $\mathbf{u}_h$, respectively. The matrix in (1.1.6) is symmetric but indefinite (has both positive and negative eigenvalues) with a null block on the lower diagonal. This type of linear system is called a "saddle point problem".

Computational challenges occur while solving saddle point problems. Direct linear solvers can be easily overwhelmed whenever finer meshes or 3D flows are involved since $\mathbf{u}_h$ and p_h are both coupled, and Gaussian elimination leads to a lot of fill-in of the space matrix. Moreover, the computational cost in terms of number of floating point operations could be as high as $O((n+m)^3)$ and memory requirements become prohibitive. On the other hand, the convergence behaviour of iterative solvers is usually not good since the global matrix is indefinite [4]. To mitigate this problem, iterative solvers with specially constructed preconditioner are required to accelerate the convergence [5]. Nonetheless for most of smaller incompressible flow problems, the linear system can be best solved using direct method (e.g., using unsymmetric multi-frontal methods, UMFPACK [11]).

We recall that for Stokes equations the discrete operator B_h^T corresponds to the gradient operator for p_h and furthermore $p_h \in Q_h \subset L_0^2(\Omega)$. This implies that $\mathrm{Ker}(B_h^T) \neq \{0\}$ if only a Dirichlet boundary condition on the velocity $\mathbf{u}_h$ is involved and if Q_h is taken as a subspace of $L^2(\Omega)$ rather than of $L_0^2(\Omega)$ (i.e., zero-average pressure is not enforced). The global linear system (1.1.6) does not yield a unique solution since the solution p_h is unique up to an additive constant. Hence, the global matrix is singular which causes technical difficulties to solve the linear system. When solving Stokes equations involving Neumann boundary conditions however, the uniqueness

of p or p_h is guaranteed. To address this issue, several numerical techniques are proposed, as explained in the following section.

1.2 Coupled Methods

In problem (1.1.3), the variables $\mathbf{u}_h$ and p_h are coupled, and uncoupling is hardly achievable for most pair of compatible finite elements. The resulting linear system is either solved in a coupled fashion or by iterating between momentum (to obtain $\mathbf{u}_h$) and continuity equations (to obtain p_h). In both cases, non-uniqueness of the pressure p_h (resulting from Dirichlet boundary condition on the velocity $\mathbf{u}_h$) requires special care. We first present two coupled methods to address the lack of uniqueness of the pressure, known as the penalization and zero mean methods.

1.2.1 Penalization Method

As mentioned earlier, solving linear system (1.1.6) is not always possible since the global matrix is singular. Temam [65] proposed to add a penalization term ϵp to the continuity equation such that $\nabla \cdot \mathbf{u} + \epsilon p = 0$. The penalization parameter $\epsilon > 0$ is an arbitrary small real number. With this pertubation term, the global matrix has full rank and $\mathrm{Ker}(B_h^T) = \{0\}$. The linear system can then be solved for any right hand side and reads as follows:

$$\begin{bmatrix} A_h & B_h^T \\ B_h & -\epsilon\mathcal{M} \end{bmatrix} \begin{bmatrix} \mathbf{u}_h \\ p_h \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \tag{1.2.1}$$

where $\mathcal{M}$ is a mass matrix (eventually lumped). By using penalization method, we expect that the solutions are slightly perturbed from the actual solution with a discrepancy that depends on the parameter ϵ. In practice, the penalization parameter ϵ is set to the lowest limit to obtain a good solution. On the other hand, ϵ should not be too small to prevent over-penalization which eventually "destroys" the solution. The optimal penalization parameter may depend on the smallest element size h_{min} such that $\epsilon = \mathcal{O}(h_{min}^{-1})$. To control the mesh dependence in our numerical computations, we chose $\epsilon = \frac{\bar{\epsilon}}{h_{min}}$ where $\bar{\epsilon} = 1 \times 10^{-8}$ is typically fixed.

1.2.2 Zero Mean Method

We recall that the space for the pressure $L_0^2(\Omega)$ is isomorphic to the quotient space $L^2(\Omega)/\mathbb{R}$, since the pressure is set up to an arbitrary constant. This is equivalent to finding the pressure p within the functional space $L^2(\Omega)$ such that p satisfies a zero

mean condition over the domain; i.e., $\int_\Omega p \, \mathrm{d}x = 0$. This can be done by introducing a Lagrange multiplier, $\mu \in \mathbb{R}$, to enforce the zero average and modifying problem (1.1.2) as follows: Seek $(\mathbf{u}, p, \lambda) \in [H_0^1(\Omega)]^d \times L^2(\Omega) \times \mathbb{R}$ such that

$$\begin{aligned} \int_\Omega \nu \nabla \mathbf{u} \cdot \nabla \mathbf{v} \, \mathrm{d}x - \int_\Omega p \nabla \cdot \mathbf{v} \, \mathrm{d}x &= \int_\Omega \mathbf{f} \cdot \mathbf{v} \, \mathrm{d}x, && \text{for all } \mathbf{v} \in [H_0^1(\Omega)]^d, \\ - \int_\Omega q \nabla \cdot \mathbf{u} \, \mathrm{d}x - \int_\Omega \mu q \, \mathrm{d}x &= 0, && \text{for all } q \in L^2(\Omega), \\ - \int_\Omega \lambda p \, \mathrm{d}x &= 0, && \text{for all } \lambda \in \mathbb{R}. \end{aligned} \tag{1.2.2}$$

Here, one can see that the second constraint appears as the third equations in (1.2.2). Albeit being able to produce satisfactory results, this method has a clear drawback. Besides producing yet another saddle point problem, the linear system is larger by only one unknown and one equation than the linear system for the penalization method, however the extra row in the global matrix has non-zero coefficients for all velocity degrees of freedom. The resulting matrix has much larger bandwidth. However, an advantage of using this method is that there is no additional parameter required when solving (1.2.2).

1.3 Uncoupled methods

Uncoupled methods are sometimes confused with splitting methods. There are two different types of principles behind these methods. Uncoupled methods aim at breaking down the linear system such that two or more variables are solved independently, while splitting methods consist in the reduction of the main PDE problem into a series of smaller PDE problems based upon the operators in the main PDE (e.g., Stokes, Laplacian, nonlinear advection, etc). We shall introduce splitting methods which are meant to solve nonstationary Navier–Stokes equations in later sections. We now consider one uncoupled method to reduce the saddle point linear system (1.1.6) to two smaller linear systems with symmetric positive definite (SPD) matrices. This can be done by the addition of a penalization term or by breaking the linear system into subsystems, for instance one linear system for $\mathbf{u}$ and one for p.

1.3.1 Augmented Lagrangian Method (ALM)

This uncoupled method is similar to the penalization method where the term 'ϵp' is added to the mass conservation in (1.1.1). For ALM, the term $\epsilon \frac{\partial p}{\partial s}$ known as the artificial-compressibility term is added to the continuity equation in Stokes equations,

where $\frac{\partial p}{\partial s}$ is the derivative of the pressure with respect to the pseudo-time s. This gives the following system of PDEs

$$\begin{aligned} -\nu\Delta\mathbf{u} + \nabla p &= \mathbf{f}, \quad && \text{in } \Omega, \\ \epsilon\frac{\partial p}{\partial s} + \nabla\cdot\mathbf{u} &= 0, && \text{in } \Omega, \\ \mathbf{u} &= 0, && \text{on } \Gamma. \end{aligned} \tag{1.3.1}$$

As for the penalization method, the so-called artificial-compressibility term $\epsilon\frac{\partial p}{\partial s}$ enforces the uniqueness of pressure at every time s (see [56, 65]). Letting the pseudo-time $s \to +\infty$ and assuming that $\frac{\partial p}{\partial s} \to 0$, we recover a solution of Stokes equations. In order to solve (1.3.1) the discretization of the artificial-compressibility term by a 1^{st}-order backward-Euler scheme gives

$$\begin{aligned} -\nu\Delta\mathbf{u}^{r+1} + \nabla p^{r+1} &= \mathbf{f}, \quad && \text{in } \Omega, \\ p^{r+1} = p^r - \lambda\nabla\cdot\mathbf{u}^{r+1} &= 0, && \text{in } \Omega, \\ \mathbf{u}^{r+1} &= 0, && \text{on } \Gamma, \end{aligned} \tag{1.3.2}$$

where $r \in \mathbb{N}$, $\lambda = \frac{\Delta s}{\epsilon}$ is fixed and known as the iteration parameter. By eliminating the pressure term p^{r+1} in momentum equation using the one from the continuity equation in (1.3.2), the saddle point problem results in two uncoupled problems, each with a SPD matrix. This technique gives rise to the augmented Lagrangian method (ALM), in which we solve (1.3.2) iteratively using proper functional spaces for velocity and pressure, respectively [15, 48]. Given $\mathbf{f} \in L^2(\Omega)$ and the initial pressure $p^0 = p(0) \in Q$, $\forall s \in \mathbb{N}^+$, we seek the solution $\mathbf{u}^{s+1} \in V$ and $p^{s+1} \in Q$ such that

$$\begin{aligned} \int_\Omega \nu\nabla\mathbf{u}^{s+1}\cdot\nabla\mathbf{v}\,\mathrm{d}x + \int_\Omega \lambda(\nabla\cdot\mathbf{u}^{s+1})(\nabla\cdot\mathbf{v})\,\mathrm{d}x - \int_\Omega p^s\nabla\cdot\mathbf{v}\,\mathrm{d}x &= \int_\Omega \mathbf{f}\mathbf{v}\,\mathrm{d}x, \quad \forall\mathbf{v}\in V, \\ \int_\Omega p^{s+1}q\,\mathrm{d}x = \int_\Omega p^s q\,\mathrm{d}x - \int_\Omega \lambda(\nabla\cdot\mathbf{u}^{s+1})q\,\mathrm{d}x, &\quad \forall q \in Q. \end{aligned} \tag{1.3.3}$$

The first equation (momentum) in (1.3.3) is used to compute $\mathbf{u}^{s+1}$ for a given $\mathbf{p}^s$, while the second equation (continuity) updates the pressure, $p^s \to p^{s+1}$. Here, the iteration parameter, λ need not be chosen very large, (i.e., $\lambda = \mathcal{O}(1)$) since we enforce the condition $\frac{\partial p}{\partial s} \to 0$ iteratively through ALM. It is worth mentioning that in practice, a stopping criteria has to be imposed, for instance $\|p^{s+1} - p^s\|_{L^2(\Omega)} < tol$, where $tol > 0$ is chosen to be arbitrary small. A detailed discussion on the optimal choice of the parameter λ can be found in [15].

1.3.2 Numerical Experiment using manufactured solution: Colliding Flow

Using a manufactured solution, we make numerical comparisons in terms of accuracy, rate of convergence and efficiency of the three methods presented above, namely the penalization, zero mean and augmented Lagrangian methods. For this purpose, we use the colliding flow [12]. The domain is $\Omega = (-1, 1) \times (-1, 1)$ and the exact solution of the colliding flow is given as follows:

$$(\mathbf{u}, p) = (u_1, u_2, p) = (20xy^3, 5x^4 - 5y^4, 60x^2y - 20y^3 + \text{constant}). \tag{1.3.4}$$

The Dirichlet boundary conditions used for the colliding flow are nothing but the value of u_1 and u_2 on the boundary which can be calculated using (1.3.4). For Stokes equations, one can verify that this is a solution for the viscosity $\nu = 1$ and the external force $\mathbf{f} = 0$. As mentioned earlier, we fix the penalization parameter $\epsilon = 10^{-8} h_{min}^{-1}$. For ALM, we fix the iteration parameter $\lambda = 1.4$, to produce a solution with least error on both velocity and pressure while having good convergence on the iterations. Further, the chosen tolerance in the stopping criteria is $tol = 10^{-12}$. Numerical results are shown in Figure 1.2.

The order of convergence p of the finite element approximation in L^2-norm using $\mathbb{P}_2$-$\mathbb{P}_1$ is computed numerically, using for instance,

$$\log\left(\frac{\|\mathbf{u} - \mathbf{u}_h\|_{L^2(\Omega)}}{\|\mathbf{u} - \mathbf{u}_{\frac{h}{m}}\|_{L^2(\Omega)}}\right) \approx p \log m, \tag{1.3.5}$$

where $\mathbf{u}$ and $\mathbf{u}_h$ are the exact and approximate solution for the velocity, respectively; $\mathbf{u}_{\frac{h}{m}}$ is the approximate velocity when the mesh size is refined by a factor of m. The order of convergence for the pressure in L^2-norm, and similarly for any variable in H^1-norm can be obtained analogously.

In terms of numerical accuracy on velocity and enforcing of divergence free condition, ALM performs slightly better than penalization and zero mean methods for all mesh size (Figure 1.2: top and bottom left). It is observed that the convergence of ALM enhances the incompressibility condition comparing to the two other methods. In a discrete setting, the **grad**-div term in the ALM momentum equation of (1.3.3) acts as a penalization term that naturally enforces the local incompressibility condition in $\mathbb{P}_2$-$\mathbb{P}_1$ elements. When using $\mathbb{P}_2$-$\mathbb{P}_1$ elements, satisfying the weak form of continuity equation $\int_\Omega q_h \nabla \cdot \mathbf{u}_h = 0,\ \forall q_h \in Q_h$, does not imply $\nabla \cdot \mathbf{u}_h = 0$ pointwise. Eleanor et al. [31], Olshanskii et al. [43] and Olshanskii & Reusken [44] had observed a significant error reduction on velocity in certain flow problems by adding the **grad**-div term to Stokes equations with Taylor–Hood elements. For pressure, the three methods are indistinguishable (Figure 1.2: bottom left), suggesting that

the perturbation introduced on pressure using the penalization method is negligible. The CPU cost for ALM is overwhelming as this method requires almost 30 minutes on finest mesh ($n = 200$) while penalization and zero mean methods require about 18 and 21 seconds, respectively (Figure 1.2: bottom right). Although ALM avoids solving a saddle point problem directly, the number of iterations required to ensure the convergence of velocity and pressure renders the method inefficient. For instance, ALM requires about 124 iterations for $n = 20$ and 106 iterations for $n = 200$ to fulfill the stopping criterion with $tol = 10^{-12}$. A small tolerance is usually required in ALM since the lack of incompressibility resulting from larger tolerances could easily spoil the solution. Consequently, with a direct solver and a small to moderate number of unknowns, Stokes equations could be more easily solved either by penalization or zero mean methods.

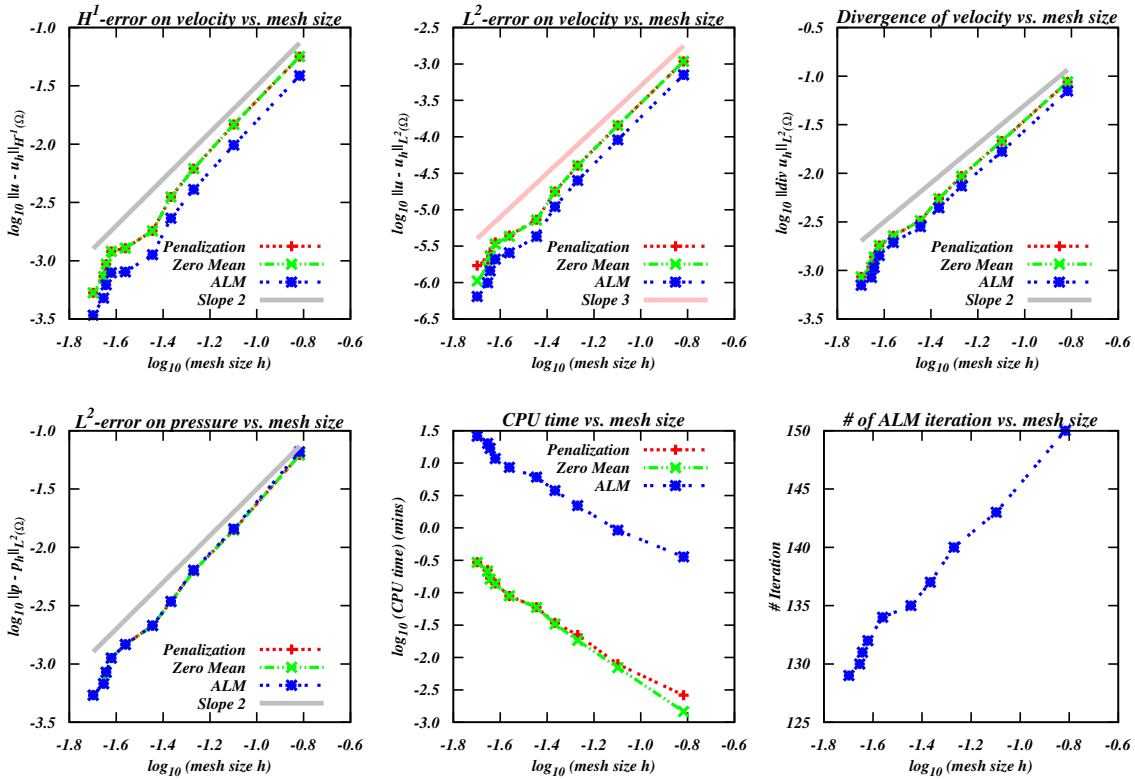

Figure 1.2: Numerical accuracy and efficiency using penalization, zero mean and ALM methods.

We observe that all the order of convergence presented in Figure 1.2 reproduces the theoretical values with $\mathbb{P}_2$-$\mathbb{P}_1$ elements (see Appendix A.0.1). For instance, we obtain 2^{nd}- and 3^{rd}-order convergence for velocity in H^1- and L^2-norm, respectively, and

2^{nd}-order for pressure in L^2-norm. The divergence term has 2^{nd}-order of convergence as h is refined to half of its size. From (1.3.4), we have the velocity $\mathbf{u} \in [\mathcal{C}^\infty(\Omega)]^2$, which leads to a 3^{rd}-order convergence in L^2-norm.

1.4 Steady Navier–Stokes Equations

The steady Navier–Stokes equations can be regarded as an extension of Stokes equation by the addition of nonlinear advection terms to account for inertia effects at greater flow velocities. The Navier–Stokes equations for steady incompressible flows are given as follows:

$$\begin{aligned} -\nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p &= \mathbf{f}, \quad && \text{in } \Omega, \\ \nabla\cdot\mathbf{u} &= 0, && \text{in } \Omega, \\ \mathbf{u} &= 0, && \text{on } \Gamma. \end{aligned} \tag{1.4.1}$$

By applying the same principle as for constructing a discrete weak formulation of Stokes equations, the discrete weak formulation for steady Navier–Stokes reads as: Find $\mathbf{u}_h \in V_h$ and $p_h \in Q_h$ such that

$$\begin{aligned} \int_\Omega \nu\nabla\mathbf{u}_h : \nabla\mathbf{v}_h \,\mathrm{d}x + \int_\Omega (\mathbf{u}_h\cdot\nabla\mathbf{u}_h)\mathbf{v}_h \,\mathrm{d}x - \int_\Omega p_h\nabla\cdot\mathbf{v}_h \,\mathrm{d}x, &= \int_\Omega \mathbf{f}\mathbf{v}_h \,\mathrm{d}x, \qquad && \forall \mathbf{v}_h \in V_h, \\ -\int_\Omega q_h\nabla\cdot\mathbf{u}_h \,\mathrm{d}x &= 0, && \forall q_h \in Q_h. \end{aligned} \tag{1.4.2}$$

The common issue when solving (1.4.2) is the presence of a nonlinear advection term that makes the resulting algebraic system nonlinear. A fixed point iterative method is then required to solve the nonlinear system. Two of the most popular fixed point methods are Newton's and Picard iterative methods [12]. In the following, we present the discrete weak formulations for steady Navier–Stokes equations where the nonlinear advection terms are linearized using both Newton's and Picard iterative methods, all using $\mathbb{P}_2$-$\mathbb{P}_1$ elements for space discretizations.

1.4.1 Newton's Method

The derivation of Newton's method for Navier–Stokes equations is based on Newton-Raphson's method (see Appendix B.2). This method reads as follows: Given an

initial data $\mathbf{u}_h^0 \in V_h$, we solve for $(\mathbf{u}_h^{m+1}, p_h^{m+1}) \in V_h \times Q_h$, $\forall m > 0$ the system

$$\begin{aligned}\int_\Omega \nu \nabla \mathbf{u}_h^{m+1} : \nabla \mathbf{v}_h \, \mathrm{d}x + \int_\Omega (\mathbf{u}_h^m \cdot \nabla \mathbf{u}_h^{m+1}) \mathbf{v}_h \, \mathrm{d}x + \int_\Omega (\mathbf{u}_h^{m+1} \cdot \nabla \mathbf{u}_h^m) \mathbf{v}_h \, \mathrm{d}x \\ - \int_\Omega p_h^{m+1} \nabla \cdot \mathbf{v}_h \, \mathrm{d}x = \int_\Omega \mathbf{f} \mathbf{v}_h \, \mathrm{d}x + \int_\Omega (\mathbf{u}_h^m \cdot \nabla \mathbf{u}_h^m) \mathbf{v}_h \, \mathrm{d}x, \text{ for all } \mathbf{v}_h \in V_h, \\ - \int_\Omega q_h \nabla \cdot \mathbf{u}_h^{m+1} \, \mathrm{d}x = 0, \text{ for all } q_h \in Q_h.\end{aligned} \tag{1.4.3}$$

A good initial guess consists in taking $\mathbf{u}_h^0$ the discrete solution of Stokes equations. A stopping criteria for Newton's method is given by $\|\mathbf{u}_h^{m+1} - \mathbf{u}_h^m\|_{L^2(\Omega)} < tol$, where $tol > 0$ is an arbitrary small real number. The expected rate of convergence of $\mathbf{u}^m$ to $\mathbf{u}^*$, the solution of steady Navier–Stokes equation, is quadratic. Few iterations are usually required.

1.4.2 Picard Iterative Method

One iteration of Picard iterative method is like solving the Oseen equations [58, 68], which is sometimes used to model flows associating with low Reynolds numbers. The solution of (1.4.2) using the Picard iterative method is formulated as follows: Given an initial data $\mathbf{u}_h^0 \in V_h$, we solve for $(\mathbf{u}_h^{m+1}, p_h^{m+1}) \in V_h \times Q_h, \forall m > 0$ the system

$$\begin{aligned}\int_\Omega \nu \nabla \mathbf{u}_h^{m+1} : \nabla \mathbf{v}_h \, \mathrm{d}x + \int_\Omega (\mathbf{u}_h^m \cdot \nabla \mathbf{u}_h^{m+1}) \mathbf{v}_h \, \mathrm{d}x - \int_\Omega p_h^{m+1} \nabla \cdot \mathbf{v}_h \, \mathrm{d}x \\ = \int_\Omega \mathbf{f} \cdot \mathbf{v}_h \, \mathrm{d}x, \text{ for all } \mathbf{v}_h \in V_h, \\ - \int_\Omega q_h \nabla \cdot \mathbf{u}_h^{m+1} \, \mathrm{d}x = 0, \text{ for all } q_h \in Q_h.\end{aligned} \tag{1.4.4}$$

The stopping criteria for the Newton's method is used for the Picard iterative method. Picard iterative method converges only linearly, hence many iterations are usually required. However this method is usually less sensitive to initial guesses than Newton's method. The implementation of the Picard iterative method is simpler than the implementation of the Newton's method. In practice, a few Picard iterations are used for initialization, followed by the Newton's method for speeding convergence, this is called the Picard–Newton's method.

1.5 Unsteady Navier–Stokes Equations

In this section section, we introduce another system of PDEs known as the unsteady Navier–Stokes equations which are written as follows:

$$\begin{aligned} \mathbf{u}_t - \nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p &= \mathbf{f}, && \text{in } \Omega\times(0,T),\\ \nabla\cdot\mathbf{u} &= 0, && \text{in } \Omega\times(0,T),\\ \mathbf{u} &= 0, && \text{on } \Gamma\times(0,T),\\ \mathbf{u}|_{t=0} &= \mathbf{u}_0, && \text{on } \Omega. \end{aligned} \tag{1.5.1}$$

To model nonstationary flows, the only additional term required in (1.4.1) is a time derivative for velocity which gives (1.5.1). The pressure remains a Lagrange multiplier to enforce incompressibility, hence no time derivative is present for pressure. This is one reason why solving time-dependent incompressible flows could be challenging. Before giving more details on this subject, we first review several numerical concepts related to time-discretization or time-stepping methods, and apply them to Navier–Stokes equations. Since we will be interested in high-order time-stepping methods, we will assume in this thesis that the solution of Navier–Stokes equations has all the required regularity in space and time.

1.5.1 Method of Lines

We consider a system of 1^{st}-order ordinary differential equations (ODE) which governs the evolution of a vector-field $\mathbf{y}$ with respect to time t, as follows:

$$\mathbf{y}_t = \mathbf{f}(\mathbf{y}, t), \quad \mathbf{y}(0) = \mathbf{y}_0. \tag{1.5.2}$$

A common numerical method to solve the above time-dependent system is based upon finite difference methods which reads as

$$\mathbf{y}^{n+k} = \mathbf{F}(\mathbf{y}^{n+k}, \mathbf{y}^{n+k-1}, \ldots, \mathbf{y}^{n}, t^{n+k-1}), \quad \mathbf{y}^0 = \mathbf{y}_0, \tag{1.5.3}$$

where $n = 0, 1, \ldots, M-k$ is the time-stepping index with integer $M = [\frac{T}{n}]$ and total time $T > 0$. Here, the index $k \in \mathbb{N}$ is fixed and controls the numbers of time steps required to approximate (1.5.2) by (1.5.3); e.g., $k = 1$ produces one-step method, $k = 2$ produces two-step method and so on. With a proper initialization for $\{\mathbf{y}^j\}_{j=0}^{k-1}$, discretization with (1.5.3) gives rise to a method with k^{th}-order of accuracy in time. The equation (1.5.3) gives rise to an implicit method if an algebraic system is to be solved in $\mathbf{y}^{n+k}$, otherwise it is an explicit method.

A detailed discussion on how to obtain a proper initialization will be provided in a later section. A problem similar to (1.5.2) is obtained by first discretizing the

nonstationary Navier–Stokes equations in space. This idea allows us to choose independently the space and time-discretization, such that at every time step, (1.5.3) yields a numerical solution of a steady problem of the form (1.4.1). This approach is called the method of lines. This method is appealing since a wide class of numerical methods for space and time can be easily produced with certain restriction on the mesh size and time step. For instance, one needs to fulfill Courant–Friedrich–Lewy (CFL) condition to ensure numerical stability.

1.5.2 Backward differentiation formula (BDF)

In the realm of linear multistep methods, one of the most efficient schemes (i.e., accurate and stable with relatively large time steps) to solve (1.5.1) is the backward differentiation formula (BDF) [60, 61, 64]. BDF, sometimes known as the Gear's backward difference methods, are fully-implicit time-stepping schemes since the diffusion term, nonlinear advection and pressure are evaluated at current step $(n+k)$ where a fixed $k \in \mathbb{N}$ is the order of the method. The BDF-k methods applied to Navier–Stokes equations are expressed as follows: For a given $\mathbf{u}^0 = \mathbf{u}(0)$, a fixed $k \in \mathbb{N}$, and proper initializations $\{\mathbf{u}^j\}_{j=1}^{k-1}$, we solve for $(\mathbf{u}^{n+k}, p^{n+k})$ the following algebraic system

$$\begin{aligned} \frac{1}{\tau}(\alpha \mathbf{u}^{n+k} + \sum_{j=1}^{k} \beta_j \mathbf{u}^{n+k-j}) - \nu \Delta \mathbf{u}^{n+k} + \mathbf{u}^{n+k} \cdot \nabla \mathbf{u}^{n+k} + \nabla p^{n+k} &= \mathbf{f}^{n+k}, && \text{in } \Omega, \\ \nabla \cdot \mathbf{u}^{n+k} &= 0, && \text{in } \Omega, \\ \mathbf{u}^{n+k} &= 0, && \text{on } \Gamma, \end{aligned} \tag{1.5.4}$$

for $n = 0, 1, \ldots, N-k$.

Here, we assume that the integer $N = [\frac{T}{\tau}]$ is the total number of time steps for a given time $T > 0$ and a constant time step τ. The coefficients α and β_j for all $j = 1, .., k$ must satisfy $\alpha = \sum_{j=1}^{k} \beta_j$ and are found in Table 1.1.

Table 1.1: List of coefficients in the approximation of the time derivative for the BDF schemes up to order 6, taken from [64].

k	α	$\{\beta_j\}_{j=0}^{k-1}$
1	1	$\{-1\}$
2	$\frac{3}{2}$	$\{-2, \frac{1}{2}\}$
3	$\frac{11}{6}$	$\{-3, \frac{3}{2}, -\frac{1}{3}\}$
4	$\frac{25}{12}$	$\{-4, 3, -\frac{4}{3}, \frac{1}{4}\}$
5	$\frac{137}{60}$	$\{-5, 5, -\frac{10}{3}, \frac{5}{4}, -\frac{1}{5}\}$
6	$\frac{147}{60}$	$\{-6, \frac{15}{2}, -\frac{20}{3}, \frac{15}{4}, -\frac{6}{5}, \frac{1}{6}\}$

The BDF-k methods require a special starting procedure for $k \geq 2$. The common strategy for proper initialization of any BDF-k method for all $k \geq 2$, is to generate $\mathbf{u}^j$ in incremental fashion using BDF-j methods, where $1 \leq j \leq k-1$. As an alternative, one can also use any one-step method (e.g., Runge-Kutta methods) with at least $(k-1)^{\text{th}}$-order of accuracy to generate all the required initialization data. With the suggested starting procedures, BDF-k methods produce at least k^{th}-order of accuracy for both velocity and pressure; i.e., a global error in time of order $\mathcal{O}(\tau^k)$.

Here, we limit our study to at most 3^{rd}-order BDF methods since the time step required for stability can be prohibitively small for $k > 3$. Furthermore, linear multistep methods with $k \geq 3$ are not unconditionally stable [64].

1.5.3 Oseen type method

This method can be derived by applying Picard iterative method but limiting it to one iteration at each time step. We fix the advection term $\mathbf{u}^n \cdot \nabla \mathbf{u}^{n+1}$ in the Navier–Stokes equations, the same principle is used to handle the linear advection term in the Oseen equations; i.e., $\mathbf{U} \cdot \nabla \mathbf{u}$ where $\mathbf{U}$ is a given steady velocity field. The Oseen type method of 1^{st}-order of accuracy can be summarized as follows: Given the initial data $\mathbf{u}^0 = \mathbf{u}(0)$, we solve for $(\mathbf{u}^{n+1}, p^{n+1})$ the system

$$\begin{aligned} \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\tau} - \nu \Delta \mathbf{u}^{n+1} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^{n+1} + \nabla p^{n+1} &= \mathbf{f}^{n+1}, \\ \nabla \cdot \mathbf{u}^{n+1} &= 0, \end{aligned} \tag{1.5.5}$$

for $n = 0, 1, \ldots, N-k$.

We noticed that by doing only one Picard iteration for each time step, a good saving on CPU time is achieved compared to the fully implicit BDF method of the same order of accuracy. The Oseen type method of 1^{st}-order is simple and quite stable to solve nonstationary Navier–Stokes equations, especially when a low Reynolds number is involved. However, the factorization of the global matrix still has to be done at each time step to solve the linear system.

1.5.4 Operator-Splitting schemes

Projection schemes, fractional-step approach and θ-methods can all be labelled as operator-splitting schemes. The origin of these methods can be traced back to the work of Chorin [9], Yanenko [71], Glowinski [20, 46] and Strang [62], for solving the time-dependent advection-diffusion or Navier–Stokes equations. An analysis of

splitting schemes for Navier–Stokes equations using Peaceman–Rachford approach can be found in [60] and references therein.

In parallel, a rich literature has documented projection methods [25, 35], which are based on similar principles as operator-splitting schemes. Operator-splitting schemes consist for instance in breaking the Navier–Stokes equations into simpler subproblems (e.g., Stokes, advection-diffusion). Usually two or at most three of these subproblems are solved at each time step. For Navier–Stokes equations, the configuration of these subproblems can be done in many ways. Typically, one of these subproblems updates the velocity with an advection-diffusion problem while treating pressure in explicit manner. For the other subproblem, the velocity is projected onto a divergence-free subspace (while obtaining the pressure) by solving Stokes equations and in turns, treating the nonlinear terms in explicit manner.

Although these schemes have more computational steps to deal with, they have some advantages. Besides being unconditionally stable, these schemes are one-step methods, hence self-starting and can be at most 2^{nd}-order accurate. Secondly, we observed that operator splitting schemes are more appealing in terms of the reduction of the unknowns required for the fixed point iterations. Fixed point iteration in operator splitting methods handles only the velocity compared to the BDF methods which involves both velocity and pressure. Of the many operator-splitting schemes, a few appealing ones are the Glowinski-splitting [48] and Dai-splitting [10]. They proceed as follows:

Glowinski Splitting: Let $\theta \in (0, \frac{1}{2})$, $\eta \in (0, 1)$ be fixed. Given the initial solution $\mathbf{u}^0 = \mathbf{u}(0)$, we solve the following subproblems for all $n = 0, 1, \ldots, N-1$.

(a) Linear Stokes equations: Using $\mathbf{u}^n$ from the previous time step, we solve for $(\mathbf{u}^{n+\theta}, p^{n+\theta})$ such that

$$\begin{aligned} \frac{\mathbf{u}^{n+\theta} - \mathbf{u}^n}{\tau\theta} - \eta\nu\Delta\mathbf{u}^{n+\theta} + \nabla p^{n+\theta} &= (1-\eta)\nu\Delta\mathbf{u}^n - \mathbf{u}^n \cdot \nabla\mathbf{u}^n + \mathbf{f}^n, \\ \nabla \cdot \mathbf{u}^{n+\theta} &= 0. \end{aligned} \tag{1.5.6}$$

(b) Nonlinear elliptic advection-diffusion equations: Using $(\mathbf{u}^{n+\theta}, p^{n+\theta})$ from (a) we solve for $\mathbf{u}^{n+1-\theta}$ such that

$$\begin{aligned} \frac{\mathbf{u}^{n+1-\theta} - \mathbf{u}^{n+\theta}}{\tau(1-2\theta)} + (1-\eta)\nu\Delta\mathbf{u}^{n+1-\theta} + (\mathbf{u}^{n+1-\theta} \cdot \nabla)\mathbf{u}^{n+1-\theta} &= \mathbf{f}^{n+\theta} - \eta\nu\Delta\mathbf{u}^{n+\theta} \\ &\quad - \nabla p^{n+\theta}. \end{aligned} \tag{1.5.7}$$

(c) Linear Stokes equations: Using $\mathbf{u}^{n+1-\theta}$ from (b), we solve for $(\mathbf{u}^{n+1}, p^{n+1})$ such that

$$\begin{aligned} \frac{\mathbf{u}^{n+1}-\mathbf{u}^{n+1-\theta}}{\tau\theta} - \eta\nu\Delta\mathbf{u}^{n+1} + \nabla p^{n+1} &= (1-\eta)\nu\Delta\mathbf{u}^{n+1-\theta} - \mathbf{u}^{n+1-\theta}\cdot\nabla\mathbf{u}^{n+1-\theta} \\ &\quad + \mathbf{f}^{n+1-\theta}, \\ \nabla\cdot\mathbf{u}^{n+1} &= 0. \end{aligned} \tag{1.5.8}$$

This splitting scheme is known to have 2^{nd}-order accuracy for both velocity and pressure if $\theta = 1 - \frac{\sqrt{2}}{2}$ and only 1^{st}-order accuracy otherwise.

Dai-splitting: Given the initial solution $\mathbf{u}^0 = \mathbf{u}(0)$, we set $p^{-1} = p^0 = p(0)$ by convention and we solve the following subproblems for $n = 0, 1, \ldots, N-1$

(a) Nonlinear elliptic advection-diffusion equations: Using (p^n, p^{n-1}) from previous time steps, we solve for $\widetilde{\mathbf{u}}^{n+1}$ such that

$$\begin{aligned} &\frac{\widetilde{\mathbf{u}}^{n+1}-\mathbf{u}^n}{\tau} - \frac{\nu}{2}\Delta(\widetilde{\mathbf{u}}^{n+1}+\mathbf{u}^n) \\ &+\left(\frac{\widetilde{\mathbf{u}}^{n+1}+\mathbf{u}^n}{2}\right)\nabla\cdot\left(\frac{\widetilde{\mathbf{u}}^{n+1}+\mathbf{u}^n}{2}\right) = \mathbf{f}^{n+\frac{1}{2}} - \frac{1}{2}\nabla(p^n + p^{n-1}). \end{aligned} \tag{1.5.9}$$

(b) Linear Stokes equations: Using $\widetilde{\mathbf{u}}^{n+1}$ from (a) and p^n from previous steps, we solve for $\mathbf{u}^{n+1}$ such that

$$\begin{aligned} \frac{\mathbf{u}^{n+1}-\widetilde{\mathbf{u}}^{n+1}}{\tau} - \frac{\nu}{2}\Delta(\mathbf{u}^{n+1}-\widetilde{\mathbf{u}}^{n+1}) + \frac{1}{2}\nabla(p^{n+1}+p^n) &= 0, \\ \nabla\cdot\mathbf{u}^{n+1} &= 0. \end{aligned} \tag{1.5.10}$$

As an alternative, the initialization for $(\mathbf{u}^{-1}, p^{-1})$ and $(\mathbf{u}^0, p^0)$ can be done by shifting the initial step as follows: For any given $(\mathbf{u}^{-1}, p^{-1})$, the value of $(\mathbf{u}^0, p^0)$ at $t = \tau$ can be obtained using for instance a 1^{st}-order method (e.g., BDF-1, Oseen, etc). Dai-splitting method is unconditionally stable with 2^{nd}-order accuracy for both velocity and pressure [10].

1.5.5 Crank–Nicolson Adam–Bashford (CNAB)

Another time-discretization scheme which received equally much attention to solve Navier–Stokes equations is the Crank–Nicolson Adam–Bashford (CNAB) method. This method uses trapezoidal rule (Crank–Nicolson) for time-discretizations of linear

terms such as Stokes operator $(-\nu\Delta\mathbf{u} + \nabla p)$ and linear external force. For the nonlinear operators such as the advection, 2^{nd}-order accurate explicit Adam–Bashford is used to get rid of the nonlinearity. The CNAB method is known to be conditionally stable with 2^{nd}-order accuracy for both velocity and pressure in time. Another advantage of CNAB method is that this scheme does not require fixed point strategy for nonlinear term. It falls under the category of semi-implicit schemes which will be discussed in more details in the following chapter. The stability and convergence analysis of CNAB for time-dependent Navier–Stokes equations can be found in [26, 27]. CNAB schemes for solving (1.5.1) is given as follow: Given the initial point $(\mathbf{u}^0, p^0) = (\mathbf{u}(0), p(0))$, the solution $(\mathbf{u}^{n+1}, p^{n+1})$ for $n = 0, 1, \ldots, N-1$ is computed by solving the system

$$\begin{aligned}\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\tau} + \nu\left(\frac{\Delta\mathbf{u}^{n+1} + \Delta\mathbf{u}^n}{2}\right) + \left(\frac{\nabla p^{n+1} + \nabla p^n}{2}\right) &= -\frac{3(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n}{2} \\ &\quad + \frac{(\mathbf{u}^{n-1} \cdot \nabla)\mathbf{u}^{n-1}}{2} + \frac{\mathbf{f}^n + \mathbf{f}^{n+1}}{2} \\ \nabla \cdot \mathbf{u}^{n+1} &= 0\end{aligned} \tag{1.5.11}$$

Note that an artificial initial condition on pressure is introduced for this scheme. If p^0 is not chosen properly, spurious pressure modes may appear for all even time steps.

1.5.6 Characteristic Galerkin method

The idea of solving unsteady Navier–Stokes equations using the so-called Characteristic Galerkin method was first introduced by Pironneau [47]. This method exploits the fact that both the time-derivative and advection term result from the material derivative

$$\begin{aligned}\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} &= \frac{\partial\mathbf{u}}{\partial t} + \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \cdot \frac{\partial\mathbf{u}}{\partial\mathbf{x}} \\ &= \frac{\partial\mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla\mathbf{u},\end{aligned} \tag{1.5.12}$$

where $\mathbf{u} = \mathbf{u}(t, \mathbf{X})$ is the velocity field written in Lagrangian coordinates $\mathbf{X} = \mathbf{X}(t; s, \mathbf{x})$. The characteristic curve or Lagrangian coordinates $\mathbf{X}$ at time t of a point with Eulerian coordinates $\mathbf{x}$ at time s satisfies the following initial value problem:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{X}(t; s, \mathbf{x})}{\mathrm{d}t} &= \mathbf{u}(t, \mathbf{X}(t; s, \mathbf{x})), \quad t \in (0, T), \\ \mathbf{X}(s; s, \mathbf{x}) &= \mathbf{x}.\end{aligned} \tag{1.5.13}$$

By applying the 1^{st}-order backward Euler scheme to discretize the material derivative in (1.5.12), the unsteady Navier–Stokes equations (1.5.1) can be solved using the

following scheme [48]: Given the initial data $\mathbf{u}^0 = \mathbf{u}(0)$, the solution $(\mathbf{u}^{n+1}, p^{n+1})$ for $n = 0, 1, \ldots, N-1$ is computed by solving the system

$$\begin{aligned} \frac{\mathbf{u}^{n+1} - \mathbf{u}^n \circ \mathbf{X}^n}{\tau} + \nu \Delta \mathbf{u}^{n+1} + \nabla p^{n+1} &= \mathbf{f}^{n+1} \\ \nabla \cdot \mathbf{u}^{n+1} &= 0 \end{aligned} \tag{1.5.14}$$

where $\mathbf{X}^n$ denotes a suitable approximation of $\mathbf{X}(t^n; t^{n+1}, \mathbf{x})$. The above Characteristic Galerkin method produces 1^{st}-order accuracy for both velocity and pressure in $\mathcal{O}(h + \tau + \frac{h^2}{\tau})$ as τ and $h \to 0$. This method is unconditionally stable, however both time step τ and mesh size h have to be chosen appropriately in order to exhibit an optimal 1^{st}-order convergence in time and space.

1.5.7 Numerical experiment using manufactured solution

The time-stepping schemes presented above are assessed using a manufactured solution. For this purpose, a square domain $\Omega = (0, 1)^2$ is used with a fixed viscosity parameter $\nu = 1$. The exact solution for this manufactured solution is given in [23]

$$\big(\mathbf{u}, p\big) = \big(u_1, u_2, p\big) = \big(\sin(x)\sin(y+t), \cos(x)\cos(y+t), \cos(x)\sin(y+t)\big). \tag{1.5.15}$$

For the top, bottom and right boundary, Dirichlet boundary conditions are imposed which can be computed using (1.5.15). Homogeneous Neumann boundary condition is used along the left boundary which gives $\nu \partial_x u - p = 0$ and $\nu \partial_x v = 0$. The Neumann boundary condition provides a unique solution for pressure p hence the penalization method can be avoided. In this setting, both the external force $\mathbf{f}$ and the divergence term $\nabla \cdot \mathbf{u}$ are zero. Computations are done for various time steps $\tau = 0.04, 0.02, 0.01$. A fixed mesh is used for all computation, which is generated by subdividing each edge of the square into 80 equal-length subintervals. The triangulation produces a uniform mesh size $h = 0.017\,677\,7$ with 51 842 and 6 561 unknowns for velocity and pressure, respectively.

Figure 1.3 shows the time evolution of L^2-errors on velocity and pressure for $t \in [0, 2]$ and various time steps. These curves show the global error between the numerical solution and exact solution given in (1.5.15); i.e., the error resulting from both space and time discretizations. The error on velocity is smaller than that on pressure for all methods since the $\mathbb{P}_2$-$\mathbb{P}_1$ elements are used. For any given time step, the 1^{st}-order methods produce the largest error, followed by 2^{nd}- and 3^{rd}-order methods. Among the 1^{st}-order methods, the largest error on velocity and pressure is produced by the Characteristic Galerkin, followed closely by the Oseen and BDF-1 methods. Of all 2^{nd}-order methods, the error produced by Dai and Glowinski splitting methods is the largest, then followed by the CNAB and BDF-2 methods which are as good for

velocity and pressure. However, both CNAB and Dai Splitting methods produce pressure oscillations at larger time step $\tau = 0.04$. If the time step is not sufficiently small, the trapezoidal rule behind these methods generates spurious pressure modes while iterating in time. In this manufactured solution, we observe that the order of convergence for both operator splitting schemes suffers from suboptimality; i.e., at most an order of 1.5 can be only achieved for both velocity and pressure. The error produced by BDF-3 method, which has the highest order of accuracy of all methods tested here, is the smallest among all methods. The solution converges drastically to the solution of the semi-discrete problem even with a larger time step. For $\tau = 0.01$, the remaining error with BDF-3 method results only from the space approximation.

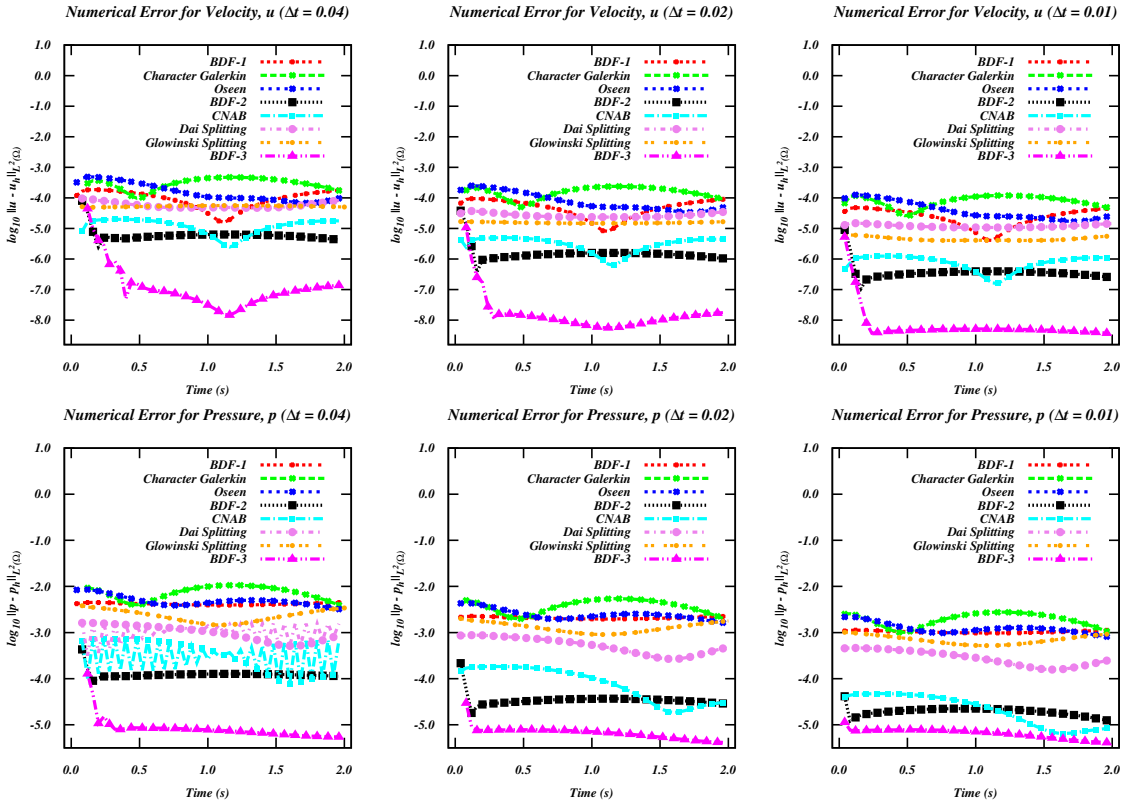

Figure 1.3: Manufactured Solution: Numerical error as a function of time for various time-stepping methods and time steps $\tau = 0.04, 0.02, 0.01$.

1.5.8 Numerical experiment using 2D flow around the cylinder

In this section, we provide another type of numerical assessment of the above time-stepping methods. We propose a test case known as the 2D flow around the cylinder, in which the lift and drag coefficients, noted respectively c_l and c_d are computed along the boundary of the cylinder. The methods are compared in terms of the quality of the solution and CPU-efficiency to compute these parameters. The computational setting of the test case, and the method to compute the lift and drag coefficients are detailed below.

Figure 1.4 (top) shows the computational domain $\Omega = (-10, 25) \times (-10, 10)$ which is relatively large compared to the circular cylinder located at the origin with diameter $D = 1$. This setting minimizes numerical error on c_l and c_d which could be possibly induced by the use of a bounded domain to compute this external flow.

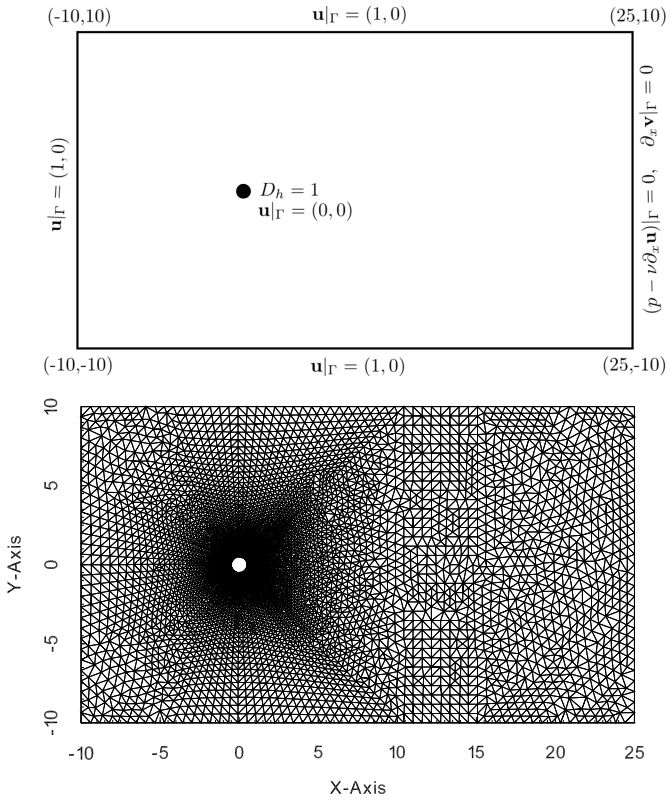

Figure 1.4: Domain for the 2D flow around a cylinder (top). The mesh consists in non-uniform triangular elements (generated automatically by subdividing the edge along x- and y-axis into 100 and 50 equal-length subintervals, respectively, while 60 equal-length subintervals is set along the cylinder (bottom).

In dimensionless setting, the flow is characterized by a constant known as the Reynolds number which is defined by

$$Re = \frac{\mathbf{U}_\infty D}{\nu}, \tag{1.5.16}$$

where the far-field velocity $\mathbf{U}_\infty = 1$ and the diameter of the cylinder $D = 1$ (also considered as the characteristic length). This simplifies (1.5.16) to $Re = \frac{1}{\nu}$. In this test case, the Reynolds number is set to 100 by fixing $\nu = 0.01$. Dirichlet boundary condition $\mathbf{u} = (1, 0)$ is imposed along the left, upper and bottom boundaries mimicking the constant flow towards the circular cylinder in x-direction. Non-slip condition $\mathbf{u} = (0, 0)$ is imposed along the boundary of the circular cylinder. To model the outflow which is occuring along the right boundary, homogeneous Neumann boundary condition works well for laminar flow. We shall see later in Chapter 3 that

similar boundary condition does not work with flows at higher Reynold numbers, (e.g., $Re > 300$) and the development of appropriate outflow boundary condition is not trivial. The implementation of another outflow boundary condition called the implicit damping schemes will be discussed later. Here, we assume the external force $\mathbf{f} = 0$. To compute c_l and c_d, the volume integration formulation is used [32]. The lift coefficient is given by

$$c_l(t) = -20 \int_\Omega \Big[\mathbf{u}_t \cdot \mathbf{v}_l + \nu \nabla \mathbf{u}(t) : \nabla \mathbf{v}_l + (\mathbf{u}(t) \cdot \nabla \mathbf{u}(t)) \cdot \mathbf{v}_l - p(t) \nabla \cdot \mathbf{v}_l \Big] \, dx, \quad (1.5.17)$$

where $\mathbf{v}_l \in [H^1(\Omega)]^2$ is the solution of an auxillary Stokes problem with boundary conditions $\mathbf{v}_{|S} = (0, 1)$ and $\mathbf{v}_{|\Gamma} = (0, 0)$ on all other boundaries. Similarly, one can compute the drag coefficient using

$$c_d(t) = -20 \int_\Omega \Big[\mathbf{u}_t \cdot \mathbf{v}_d + \nu \nabla \mathbf{u}(t) : \nabla \mathbf{v}_d + (\mathbf{u}(t) \cdot \nabla \mathbf{u}(t)) \cdot \mathbf{v}_d - p(t) \nabla \cdot \mathbf{v}_d \Big] \, dx, \quad (1.5.18)$$

where $\mathbf{v}_d \in [H^1(\Omega)]^2$ is the solution of an auxillary Stokes problem with boundary conditions $\mathbf{v}_{|S} = (1, 0)$ and $\mathbf{v}_{|\Gamma} = (0, 0)$ on all other boundaries. The test case is run for $t \in [0, 30]$ after restarting from a periodic solution (on both velocity and pressure). This periodic solution can be easily obtained, for instance with the Characteristic Galerkin method using a large time step $\tau = 0.05$.

At $Re = 100$, the time evolution of the lift and drag coefficients along the cylinder produce a periodic solution with a single frequency of about 0.44. The periodicity of the lift and drag results from the development of a von Kármán vortex street in the wake behind the cylinder. The time step used in the computation, the order of accuracy and the respective CPU time to execute 10 iterations for these methods, are summarized in Table 1.2. We fix two different time steps for all 1^{st}-order methods, with one five times smaller than the other. This is done to assess if this refinement is sufficient to ensure a comparable accuracy for the 1^{st}-order and the 2^{nd}-order methods. Time step for CNAB method has to be chosen slightly smaller than 0.01 to avoid pressure oscillation in time. The time step $\tau = 0.005$ is used in BDF-3 method to obtain very accurate reference solution since it is a 3^{rd}-order method. The lift and drag computed with BDF-3 method in this experiment are used as the reference solutions.

Table 1.2: Time steps chosen for the time-stepping methods with their order of accuracy and CPU time for 10 iterations.

Method	Order of accuracy	Time-step, τ	CPU(per 10 iter)
BDF-1	1^{st}-order	$0.01, 0.002$	162.5
Characteristic Galerkin	1^{st}-order	$0.01, 0.002$	10.6
Oseen	1^{st}-order	$0.01, 0.002$	54.8
BDF-2	2^{nd}-order	0.01	160.0
CNAB	2^{nd}-order	0.008	13.6
Glowinski Splitting	close to 2^{nd}-order	0.01	122.8
Dai Splitting	2^{nd}-order	0.01	120.7
BDF-3	3^{rd}-order	0.005	163.7

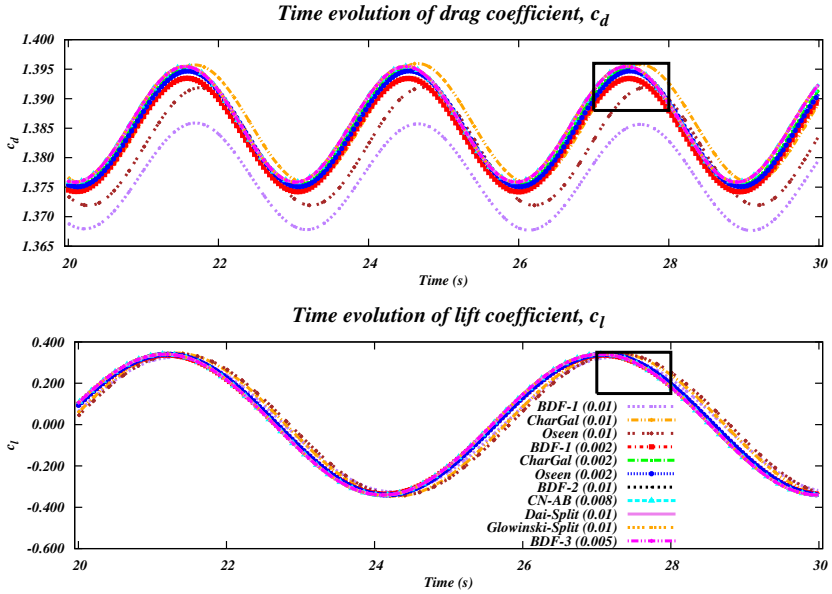

Figure 1.5: Time evolution of the lift and drag coefficients using various time-stepping methods for $t \in [20, 30]$ (Flow past a circular cylinder).

Figure 1.5 shows that the lift and drag coefficients are very close to one except for some which are computed with 1^{st}-order methods (BDF-1, Characteristic Galerkin and Oseen) using $\tau = 0.01$. Figure 1.6 shows a close up view of the lift and drag coefficients for $t \in [27, 28]$. The drag coefficient computed with 1^{st}-order methods with smaller time step $\tau = 0.002$, is still less accurate than the one computed with 2^{nd}-order methods with $\tau = 0.01$. We observed that all lift and drag coefficients obtained with 2^{nd}-order methods converged to the one computed with BDF-3 method, which is taken as the reference solution.

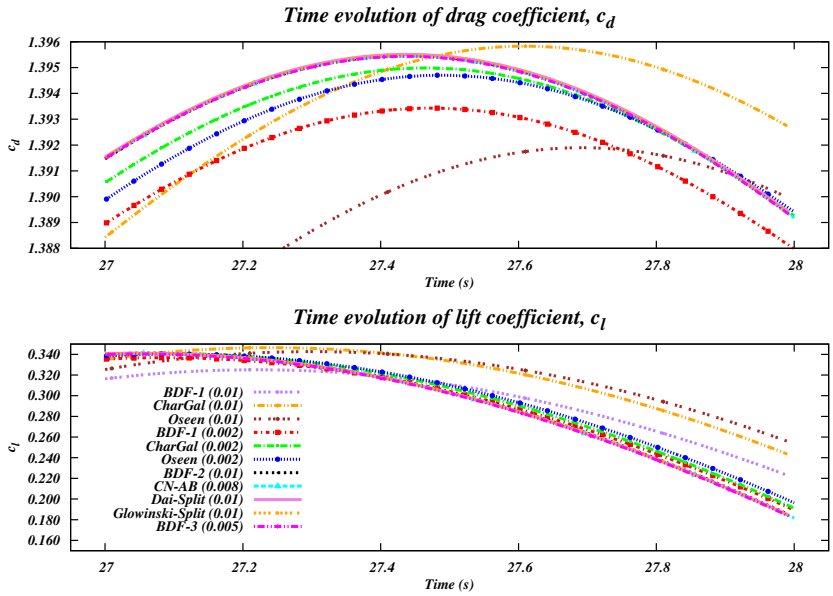

Figure 1.6: Flow past a circular cylinder: Time evolution of the lift and drag coefficients using various time-stepping methods (close-up) for $t \in [27, 28]$ (Flow past a circular cylinder).

This suggests that much finer time step has to be chosen (e.g., $\tau < 0.001$) with 1^{st}-order methods to produce results as accurate as those from 2^{nd}-order methods. Characteristic Galerkin is CPU-efficient since it solves an unsteady Stokes problem in semi-implicit fashion. The only additional step required for the Characteristic Galerkin method is an efficient search-interpolation algorithm to track the solution along the characteristics curves [19]. There is little difference in CPU time comparing 1^{st}-, 2^{nd}- and 3^{rd}-order BDF methods for a given number of time steps. This suggests that the numerical complexity of the resulting nonlinear system is independent of the order of the BDF method, except for the evaluation of the right hand side. It results that among the BDF methods, BDF-3 is the most efficient to reach a given accuracy in time when computing unsteady flows.

The BDF methods are not CPU-efficient since Newton's method is required for solving the resulting nonlinear system. Moreover, a factorization of the global matrix is needed at every fixed point iteration. For Dai- and Glowinski-splitting methods, the CPU time is almost identical (about 120 s). These operator-splitting methods solve the unsteady Navier–Stokes problem through simpler advection-diffusion and Stokes problems. Newton's method is still required for Dai- and Glowinski-splitting. The fixed point method now tackles smaller advection-diffusion problem while the linear Stokes problem results in a constant matrix. Globally this results in a lower CPU cost compared with that of the BDF methods.

For the Oseen method, the CPU time depends only on the factorization of the global matrix and a linear solve at each time step. Our numerical results show that BDF methods required about three Newton's iteration at each time step. The CPU time required for BDF methods (about 165 s) is about three times that with Oseen method (about 54 s), which demonstrate a perfect scalability with the number of Newton's iteration. CNAB is a semi-implicit method since the nonlinear terms are extrapolated using Adam–Bashford and the factorization of the global matrix is only done once. CNAB method is known for the oscillation produced on the pressure (in time) due to the artificial initial condition on this variable. If the oscillation in CNAB method is treated properly [45], the low CPU cost makes the method appealing.

1.6 Problem Statements and Outline of the Thesis

The method of lines allows us to construct time-stepping methods independently from the space discretization. Higher-order methods, (e.g., BDF-3 methods) produce very accurate results when solving nonstationary Navier–Stokes equations (1.5.1) without relying on a small time step. On the other hand, BDF methods may not be very efficient since the convergence of the Newton's method is essential at each time step and this further requires the re-assembly of the global matrix for the linear system. BDF methods becomes expensive to compute flows involving high Reynolds number when smaller mesh size and time step are involved.

From preliminary test cases (not shown here), we make the following observations. Of all methods, CNAB method is the most efficient to reach a given level of error. First of all, CNAB method has 2^{nd}-order accuracy and secondly, it has outstandingly low CPU cost at each time step. CNAB is still very efficient even though the time step is restricted by a CFL-like condition. This is due to the fact that CNAB falls under the group of "semi-implicit method" which will be described in details later on. A main drawback of the CNAB method is the pressure oscillations, a nontrivial issue to deal with. Dai- and Glowinski-splitting methods requires about 75% of the CPU cost of a BDF-2 method. However, the numerical error produced is larger than that for BDF-2 method for a given time step, which make them less appealing. Moreover, Dai-splitting method suffers from pressure oscillations just like CNAB since it is based on trapezoidal rule when solving the advection-diffusion equation. With the test case, we observed that the accuracy of these operator-splittings is limited by the order barrier [23] and recovering the theoretical order of convergence is difficult.

Of all 1^{st}-order methods, Characteristic Galerkin method is the cheapest but the quality of the solution requires very small time step. Constructing Characterictic Galerkin methods with a high-order accuracy can be done but these methods are

usually difficult to implement. Characteristic Galerkin methods with 2^{nd}-order accuracy were investigated in [70]. Nonetheless, all 1^{st}-order methods are very efficient to compute the solution of steady flows since very large time step can be chosen. In BDF-1 method, one can even fast-forward the transient solution to the steady regime by fixing a small number of Newton's iteration at each time step. Usually two to three iterations per time step are sufficient and this can be done since capturing the steady solution is our main interest.

Fully-explicit methods (e.g., forward differentiation formula) for time discretization can be very appealing since only a linear system with a mass matrix requires to be solved. Due to the stiffness of the Laplace operator, fully-explicit methods are very restrictive in terms of the time step required to fulfill the Θ-stability condition, where $\Theta := \frac{\tau}{h^2 Re} < \text{const}$. See [48] for a discussion on this topic. The restriction on the time step with $\tau = \mathcal{O}(h^2 Re)$ simply renders these methods impractical.

Semi-implicit methods use the implicit and explicit treatment of some of the terms to compute numerical solution of ODE and PDE. For instance, when discretizing (1.5.1) using semi-implicit methods, all terms are evaluated implicitly except the nonlinear advection. Therefore, the need for fixed point method as presented in Section 1.4.1 and 1.4.2 can be totally avoided. Semi-implicit methods are conditionally stable since they are required to fulfill certain CFL stability condition. Moreover, the time steps required for the stability of semi-implicit methods are far less restrictive than that of fully-explicit methods (see [1, 37]). Our goal is to investigate semi-implicit approaches that are both accurate and efficient to compute nonstationary flows.

Along this direction, we propose several variants of semi-implicit schemes for solving nonstationary Navier–Stokes equations in Chapter 2. We first introduce a class of methods which is based upon the multistep approach, the semi-implicit backward differentiation formulae (SBDF). The second type of semi-implicit methods is based upon defect correction strategy starting from 1^{st}-order SBDF up to DC methods which can be constructed easily to reach arbitrary high-order approximation in time [23]. Using both high-order artificial compressibility and bootstrapping strategies, it can be shown that velocity and pressure in DC methods can be decoupled to give rise to Guermond-Minev methods (GM) [23]. We propose a modification of GM methods—a combination of GM with the sequential regularization methods [40] that we call GM-SRM methods. All semi-implicit methods are benchmarked numerically using manufactured solution and test cases for 2D unsteady flows. This work is documented in terms of a scientific paper entitled, "On efficient high-order semi-implicit time-stepping schemes for unsteady incompressible Navier–Stokes equations" which was published in Computer and Fluids [41].

In Chapter 3, we propose two major improvements to SBDF methods: First, we add the **grad**-div stabilization term in the variational form of Navier–Stokes equations to

improve the mass conservation in $\mathbb{P}_2$-$\mathbb{P}_1$ elements. Second, numerical improvement of SBDF methods are associated with nonlinear discretization to produce solutions with smaller error at a given order of accuracy. Using some test problems, we show that the numerical improvements presented above have a significant impact on the numerical stability and accuracy of the methods while still being very efficient to study flows with high Reynolds number. We present these result in the next chapter as a scientific paper entitled, "Robust high-order semi-implicit backward difference formulae (SBDF) for Navier–Stokes equations", which will be submitted to the International Journal for Numerical Methods and Fluids.

We observe that the correction algorithm implemented in [23] to approximate nonlinear advection may result in certain numerical inaccuracy if the computation is extended for long period of time (t is large), which is necessary to capture full periodic solutions. In Chapter 4, we propose several variants of "nonlinear ansatz" to improve this numerical error. We conduct numerical comparisons for these new "nonlinear ansatz" in terms of numerical stability and convergence before the best "nonlinear ansatz" is chosen for all DC, GM and GM-SRM methods. The second part of Chapter 4 introduces the idea of **grad**-div splitting algorithm which can be implemented in GM and GM-SRM methods [24]. These splitting algorithms reduce the complexity of the linear system from the momentum equation to a few scalar parabolic problems, which can be solved separately for each of the velocity component u and v. The resulting **grad**-div splitting requires an auxillary numerical method to ensure the convergence of this splitting which is based on similar "nonlinear ansatz" and Gauss-Seidel iteration. All these auxillary methods will be assessed numerically in terms of the resulting numerical error on velocity and pressure, all using 2D manufactured solutions.

In Chapter 5, we showcase several interesting 3D computations of both steady and unsteady flows using some of the time-stepping methods discussed in the previous chapters. We also include several short discussions on the preparation of 3D mesh, and the required iterative linear solver and preconditioner to handle tremendous degrees of freedom arising from the 3D space discretizations. All numerical results of these 3D computations are compared both qualitatively and quantitively with the numerical results available in the literature.

Concluding remarks are given in Chapter 6, summarizing all the key findings, open problems and suggested future works to improve our current research.

Chapter 2

On Efficient High-order Semi-implicit Time-stepping Schemes for Unsteady Incompressible Navier–Stokes Equations

This chapter is presented in terms of a publication in Computers and Fluids, carrying the same title as mentioned above. Please see the attached paper for the content.

This paper is co-authored with my supervisor. We discussed the selection of methods together. I implemented the methods, carried the computations and analyzed the results. Finally, I drafted the paper and my supervisor provided the editorial components.

Computers and Fluids 148 (2017) 166–184

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

On efficient high-order semi-implicit time-stepping schemes for unsteady incompressible Navier–Stokes equations☆

K.C. Loy[a,b,1], Y. Bourgault[a,*]

[a] Department of Mathematics & Statistics, University of Ottawa, 585 King Edward Avenue, Ottawa, ON, K1N 6N5, Canada
[b] School of Informatics & Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Terengganu, Malaysia

ARTICLE INFO

Article history:
Received 6 January 2017
Accepted 17 February 2017
Available online 27 February 2017

Keywords:
High-order
Time-stepping
Semi-implicit
Defect correction
Navier–Stokes equations
Sequential regularization method

ABSTRACT

We focus on the development of higher-order semi-implicit time-stepping methods to solve the incompressible Navier–Stokes equations. The Taylor–Hood mixed finite elements method ($\mathbb{P}_2 - \mathbb{P}_1$) is used for space approximation of velocity and pressure. Semi-implicit time discretizations can provide very accurate approximations for nonstationary flow while treating the nonlinear terms explicitly and avoiding the need for a fancy nonlinear iterative solver. We first introduced three variants of higher-order semi-implicit time-stepping schemes: a multistep Semi-Implicit Backward Difference Formulae (SBDF) which is the easiest to implement; a one-step defect correction (DC) which produce better approximations than SBDF methods in terms of numerical errors on velocity and pressure (in time) for a given order; the one proposed by Guermond and Minev (GM), an uncoupled method which is based upon defect correction and artificial compressibility methods. SBDF and DC methods produce saddle point linear system while GM generates two linear systems with symmetric and positive definite matrix. We propose a modification on GM denoted as GM-SRM methods–which implement sequential regularization method of Navier–Stokes equations. GM-SRM methods produce more rapid convergence on pressure during the start-up while requiring smaller stabilization parameter than that for GM methods. We showcase the numerical accuracy, stability and efficiency of these methods through numerical test cases: two manufactured solutions; the flow past a circular cylinder; and the lid-driven cavity flow, all in 2D settings. All methods show an agreement with the theoretical convergence rate. SBDF methods are the most CPU-efficient, followed by DC, GM and GM-SRM methods. We observe that the presence of **grad** – div term in both GM and GM-SRM improves the numerical stability in terms of producing a higher CFL bound. Furthermore, relevant parameters, such as the Strouhal number, lift and drag coefficients are found to be very close to published values.

1. Introduction: incompressible Navier–Stokes equations

The time evolution of incompressible flows can be generally modeled by Navier–Stokes equations, which are expressed as follows

$$\begin{aligned} \mathbf{u}_t - \nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p &= \mathbf{f} && \text{on } \Omega\times(0,T),\\ \nabla\cdot\mathbf{u} &= 0 && \text{on } \Omega\times(0,T),\\ \mathbf{u}(\mathbf{x},0) &= \mathbf{u}_0 && \text{in } \Omega,\\ \mathbf{u}(\mathbf{x},t) &= \mathbf{g} && \text{on } \Gamma\times(0,T), \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the velocity field, p the pressure, ν the kinematic viscosity, $\mathbf{f}$ any external force, $\mathbf{u}_0$ the initial velocity and $\mathbf{g}$ the velocity field imposed on the boundary Γ of the domain, Ω. The space approximations of Eq. (1) can be done using various numerical techniques, e.g., finite difference, finite element, finite volume and pseudospectral methods. Independently from the space approximations, time-stepping schemes are used to solve the semi-discrete problem (e.g. backward-Euler, trapezoidal- or midpoint rules, Runge–Kutta, etc).

Fully-implicit schemes, such as backward-difference formulae (BDF), are popular time-stepping strategies to solve the semi-discrete Navier–Stokes equation. These fully implicit methods require a fixed point method (e.g. Newton's method, Picard iteration) for solving the nonlinear algebraic system resulting from the discretization. Splitting approaches, which include the θ-methods (operator-splitting) and projection schemes etc, has an equal success to solve nonstationary Navier–Stokes equations [10,21,25].

☆ This work is supported by a NSERC Discovery Grant.
* Corresponding author.
E-mail address: ybourg@uottawa.ca (Y. Bourgault).
[1] Author is a PhD scholarship recipient given by Ministry of Education Malaysia and Universiti Malaysia Terengganu.

http://dx.doi.org/10.1016/j.compfluid.2017.02.017
0045-7930/

These splitting schemes advocate predictor-corrector steps, i.e., advection-diffusion and Stokes subproblems for fractional time steps; and pressure correction algorithm. However, several of these methods are known to generate non-physical boundary layers that may further induce sub-optimality of the convergence rate known as the order barrier [21]. Moreover, a fixed point method is required to solve the nonlinear advection-diffusion sub-step impacting the efficiency of the method. Eq. (1) can also be resolved using semi-Lagrangian approach such as the Characteristic Galerkin method [17]. This can be done by discretizing the material derivative along the 'characteristic curves'. However, the numerical efficiency and accuracy of this method is limited by the efficiency of nodal search algorithm for the tip of characteristic curves and by the accuracy of numerical quadrature on mismatched elements. All of the above methods are unconditionally stable, making them attractive methods for solving stationary Navier–Stokes problems. At the expense of losing the unconditional stability, the nonlinear advection terms can be taken explicitly with so-called semi-implicit methods, hence nonlinear fixed point iterations can be totally avoided [12].

Semi-implicit approaches make it possible to factorize the 'constant' matrix only once which gives a significant reduction in CPU time. Although time steps may be restricted by stability, this is justifiable when smaller temporal change is crucial to study non-stationary flows with higher accuracy. On the other hand, semi-implicit methods are less suitable for computing steady flows where implicit schemes with large time steps can lead to significant reduction in computing cost. An earliest convergence analysis of higher-order semi-implicit schemes for Navier-Stokes equation was done by Baker et al. [7] but no numerical test cases were presented. Kress and Lötstedt [27] provided linear stability analysis for similar semi-implicit schemes solving Navier–Stokes equations using finite difference approach for space approximation. A similar idea was introduced with so-called, 'IMplicit-EXplicit' schemes, shortened as IMEX, which were used to solved advection-diffusion or reaction-diffusion problems [1,3,11,41]. The 2nd-order Crank–Nicolson Adam–Bashforth schemes (CNAB) is a relatively popular semi-implicit method. This method requires a special treatment to filter parasitic oscillations due to its sensitivity to the initial condition, which is a non-trivial procedure. Numerical experiment shows that if these oscillations are not properly handled, the theoretical rate of convergence cannot be reproduced. There is work dedicated to make CNAB a more robust scheme but it works only for 2nd-order accuracy in time [31]. In this paper, we propose semi-implicit methods which are based on backward-forward Euler. These methods can be easily extended to arbitrary orders and have a great potential in terms of numerical accuracy and efficiency, particularly for unsteady flows.

The main objective of this paper is to conduct a thorough numerical assessment of time-stepping methods for the nonstationary, incompressible Navier–Stokes equations. Here we focus on higher-order semi-implicit methods for time-stepping (i.e. 2nd- and 3rd-order). We found that the numerical assessment on these methods in the framework of incompressible Navier–Stokes equations are still lacking. This paper is organized as follows: In Section 2, we first present the Semi-Implicit Backward Differentiation Formulae (SBDF), semi-implicit methods using deferred correction (DC), and a variant of this method proposed by Guermond and Minev (GM). All these schemes are 2nd- or 3rd-order in time. We also proposed a further modification of GM using sequential regularization methods (SRM) of Navier–Stokes equations. The combination of GM and SRM will be called the GM-SRM scheme. A short discussion on **grad** – div terms which arise naturally in GM and GM-SRM and the choice of the stabilization parameter will be presented in Section 3. Section 4 presents numerical test cases to highlight issues regarding the performance of these time-stepping schemes. These test cases involve nonstationary 2D flows, two of which are manufactured solutions with analytical solutions. Further, more challenging test cases are presented featuring a 2D pulsating flow around the cylinder (von Kármán alley) and a periodic lid-driven cavity flow to investigate the numerical stability and efficiency of semi-implicit methods. For these two flows, the unsteadiness results from a bifurcation, not from the periodic forcing term or periodic boundary conditions. Our numerical results are compared to the results from the literature. In Section 5, we present a short discussion on the strengths and weaknesses of these higher-order methods.

2. Time-stepping schemes

As mentioned earlier, the time-discretization of any PDE can be treated independently from its space approximation. Using the finite element method, we first write down the semi-discrete problem for the 2D incompressible Navier–Stokes equations. This gives: Find $\mathbf{u}_h(t) \in V_h$ and $p(t) \in M_h$ such that $\forall t \in (0, T)$,

$$\begin{aligned}
&\int_\Omega \frac{d}{dt}\mathbf{u}_h(t)\cdot\mathbf{v}_h\,d\mathbf{x} + \int_\Omega \nu\nabla\mathbf{u}_h(t) : \nabla\mathbf{v}_h\,d\mathbf{x}\\
&+\int_\Omega (\mathbf{u}_h(t)\cdot\nabla\mathbf{u}_h(t))\cdot\mathbf{v}_h\,d\mathbf{x} - \int_\Omega p_h(t)\nabla\cdot\mathbf{v}_h\,d\mathbf{x}\\
&= \int_\Omega \mathbf{f}_h(t)\cdot\mathbf{v}_h\,d\mathbf{x}, \qquad \forall\mathbf{v}_h\in V_h,\\
&-\int_\Omega q_h\nabla\cdot\mathbf{u}_h(t)\,d\mathbf{x} = 0, \qquad \forall q_h\in M_h,\\
&\mathbf{u}_h(0) = \mathbf{u}_h^0,
\end{aligned} \tag{2}$$

where $\mathbf{u}_h^0$ is an approximation of $\mathbf{u}_0$. We use the Taylor–Hood finite element discretization in space (continuous piecewise quadratic polynomial for velocity and linear polynomial for pressure):

$$V_h = X_h \cap [H_0^1(\Omega)]^2, \quad X_h = \{\mathbf{v}_h \in [\mathbb{C}^0(\overline{\Omega})]^2 \mid \mathbf{v}_{h|K}\in\mathbb{P}_2,\ \forall K\in \mathrm{T}_h\},$$

$$M_h = Y_h \cap L_0^2(\Omega), \quad Y_h = \{q_h \in \mathbb{C}^0(\overline{\Omega}) \mid q_{h|K}\in\mathbb{P}_1,\ \forall K\in\mathrm{T}_h\}.$$

Here T_h denotes a regular triangulation of the domain Ω and $\mathbb{P}_k$ the space of Lagrange polynomials of degree k on the triangles K. The Taylor–Hood element satisfies the well-known inf-sup condition [5,16]. For finite element discretization of the Stokes and Navier–Stokes equations, the pressure p is taken in $L_0^2(\Omega)$, the space of $L^2(\Omega)$ function with zero-average on Ω, as pressure is unique only up to a constant when the velocity has Dirichlet boundary conditions on all $\partial\Omega$. In practice, the pressure is penalized to control the value of this constant and obtain a nonsingular algebraic system. The solution $(\mathbf{u}_\epsilon, p_\epsilon)$ of the penalized system converge to the solution $(\mathbf{u}, p)$ of the nonpenalized system in $\mathcal{O}(\epsilon)$, $\epsilon > 0$ being the penalization parameter [46].

2.1. High-order semi–implicit backward difference formulae (SBDF)

Semi-implicit time-stepping methods of kth-order accuracy (in time) for solving Navier–Stokes equations can be written as follows: Given values $\{\mathbf{u}^j\}_{j=0}^{k-1}$ at the first k-time steps, for all $n \geq 0$ find $(\mathbf{u}^{n+k}, p^{n+k})$ solution of

$$\overbrace{\frac{1}{\tau}\left(\alpha\mathbf{u}^{n+k} + \sum_{j=1}^{k}\beta_j\mathbf{u}^{n+k-j}\right)}^{\text{discretization of the time-derivative}} - \nu\Delta\mathbf{u}^{n+k} + \overbrace{\sum_{j=1}^{k}\gamma_j\mathbf{u}^{n+k-j}\cdot\nabla\mathbf{u}^{n+k-j}}^{\text{nonlinear advection extrapolated at } t=t^{n+k}} + \nabla p^{n+k} = \mathbf{f}^{n+k}, \tag{3}$$

$$\nabla \cdot \mathbf{u}^{n+k} = 0, \tag{4}$$

where a constant time step τ is considered. The set of coefficients $\alpha, \{\beta_j\}_{j=1}^k \in \mathbb{R}$ arises from the backward differentiation formula for the time-derivative with $\alpha + \sum_{j=1}^k \beta_j = 0$. The coefficients $\{\gamma_j\}_{j=1}^k \in \mathbb{R}$ produce the extrapolation formula for the nonlinear term satisfying $\sum_{j=1}^k \gamma_j = 1$. The diffusion term $-\Delta \mathbf{u}^{n+k}$ is taken implicitly to avoid stringent stability condition in $\mathcal{O}(\nu^{-1}h^2)$ on the time step. The terms ∇p^{n+k} and $\nabla \cdot \mathbf{u}^{n+k}$ for the velocity-pressure coupling are taken implicitly to strictly enforce the discrete incompressibility condition. SBDF schemes are multistep methods which are not self-starting, therefore requiring proper initialization for $\{\mathbf{u}^j\}_{j=0}^{k-1}$. Taking the 3rd-order methods as an illustration, the initialization can be conveniently done as follows: 1st-order method (SBDF-1) is used to get $\mathbf{u}^1$ from $\mathbf{u}^0$; then 2nd-order method (SBDF-2) is used to get $\mathbf{u}^2$ from $\mathbf{u}^0$ and $\mathbf{u}^1$. Then, the intended 3rd-order method (SBDF-3) can be used i.e. at $t^n = n\tau$ for all $n \geq 3$ until the final simulation time T is reached. We noticed that initial data generated by one-step method of an equal order of accuracy has potential to reduce the numerical error (in time) by one to two orders of magnitude. For instance, we propose initialization using a one-step 3rd-order method (e.g. DC-3) to generate $\mathbf{u}^1$ and $\mathbf{u}^2$ for SBDF-3; while a one-step 2nd-order method (e.g., Crank-Nicolson, DC-2, etc) is used to generate $\mathbf{u}^1$ for SBDF-2. DC methods is based on defect correction strategy which we will introduce shortly. Although SBDF methods are well known for their simplicity, the critical time step for numerical stability is reduced as the order of the method increases. We will illustrate this fact through numerical test cases. In this work, we focus only on the 1st-, 2nd- and 3rd-order semi-implicit backward difference schemes for Navier–Stokes equations, refered to as SBDF-1, SBDF-2 and SBDF-3, respectively. The time discretization of the momentum equations using these methods can be expressed as follows:

(a) **SBDF-1**. Given the initial solution $\mathbf{u}_h^0$, for all $n \in \mathbb{N}$, find $(\mathbf{u}_h^{n+1}, p_h^{n+1})$ solution of

$$\frac{\mathbf{u}_h^{n+1} - \mathbf{u}_h^n}{\tau} - \nu\Delta\mathbf{u}_h^{n+1} + \nabla p_h^{n+1} = \mathbf{f}_h^{n+1} - B(\mathbf{u}_h^n), \tag{5}$$

(b) **SBDF-2**. Given the initial solution $\mathbf{u}_h^0$ and a proper initialization for $\mathbf{u}_h^1$, for all $n \in \mathbb{N}$, find $(\mathbf{u}_h^{n+2}, p_h^{n+2})$ solution of

$$\begin{aligned}&\frac{3\mathbf{u}_h^{n+2} - 4\mathbf{u}_h^{n+1} + \mathbf{u}_h^n}{2\tau} - \nu\Delta\mathbf{u}_h^{n+2} + \nabla p_h^{n+2}\\ &= \mathbf{f}_h^{n+2} - 2B(\mathbf{u}_h^{n+1}) + B(\mathbf{u}_h^n),\end{aligned} \tag{6}$$

(c) **SBDF-3**. Given the initial solution $\mathbf{u}_h^0$ and a proper initialization for $\{\mathbf{u}_h^1, \mathbf{u}_h^2\}$, for all $n \in \mathbb{N}$, find $(\mathbf{u}_h^{n+3}, p_h^{n+3})$ solution of

$$\begin{aligned}&\frac{11\mathbf{u}_h^{n+3} - 18\mathbf{u}_h^{n+2} + 9\mathbf{u}_h^{n+1} - 2\mathbf{u}_h^n}{6\tau} - \nu\Delta\mathbf{u}_h^{n+3} + \nabla p_h^{n+3} = \mathbf{f}_h^{n+3}\\ &-3B(\mathbf{u}_h^{n+2}) + 3B(\mathbf{u}_h^{n+1}) - B(\mathbf{u}_h^n),\end{aligned} \tag{7}$$

with these three methods sharing the same discretization of the continuity equation $\nabla \cdot \mathbf{u}_h^{n+k} = 0$, $k = 1, 2$ and 3, respectively. For simplicity, we denote the nonlinear advection term by $B(\mathbf{u}_h) = \mathbf{u}_h \cdot \nabla\mathbf{u}_h$.

2.2. Defect correction method (DC)

Defect or 'deferred' correction methods were first proposed to solve ordinary differential equations (ODEs) [45]. More recent work can be found in [26,40]. The defect correction method was applied to the time-dependent Navier–Stokes equations to produce more accurate solutions [29] while a more practical construction of the defect correction method which motivates our work is given in [20]. To begin with, time-stepping schemes with arbitrary high-order of accuracy can be constructed using only SBDF-1 with the defect correction approach. Assuming that the discrete solution (in time) for velocity and pressure can be expressed in terms of asymptotic expansions in τ as

$$\begin{aligned}\mathbf{u}^{n+1} &:= \mathbf{u}_0^{n+1} + \tau\mathbf{u}_1^{n+1} + \tau^2\mathbf{u}_2^{n+1} + \cdots + \tau^k\mathbf{u}_k^{n+1} + \mathcal{O}(\tau^{k+1}),\\ p^{n+1} &:= p_0^{n+1} + \tau p_1^{n+1} + \tau^2 p_2^{n+1} + \cdots + \tau^k p_k^{n+1} + \mathcal{O}(\tau^{k+1}),\end{aligned} \tag{8}$$

where k also denotes the degree of the expansion. Similar expansions can be defined for $\mathbf{u}^n$ and p^n. We illustrate the construction of a DC-k method where $k = 3$ is fixed. Using Taylor's expansion, we first expand $u(t)$ around $t = t^{n+1}$ to produce the following

$$\begin{aligned}\mathbf{u}(t^n) = \mathbf{u}(t^{n+1}) - \tau\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}(t^{n+1}) + \frac{\tau^2}{2!}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{u}(t^{n+1}) - \frac{\tau^3}{3!}\frac{\mathrm{d}^3}{\mathrm{d}t^3}\mathbf{u}(t^{n+1})\\ + \frac{\tau^4}{4!}\frac{\mathrm{d}^4}{\mathrm{d}t^4}\mathbf{u}(\xi)\end{aligned} \tag{9}$$

for some $\xi = \xi(\mathbf{x}) \in [t^n, t^{n+1}]$, assuming that the velocity is sufficiently smooth in time. Let us define further $\mathbf{u}^{n+1} := \mathbf{u}(t^{n+1})$, $\mathbf{u}^n := \mathbf{u}(t^n)$, $\mathbf{d}^j\mathbf{u}^{n+1} := \frac{\mathrm{d}^j}{\mathrm{d}t^j}\mathbf{u}(t^{n+1})$. Combining Eq. (8) and (9) and keeping terms up to τ^3 give the following:

$$\begin{aligned}\mathbf{u}_0^n + \tau\mathbf{u}_1^n + \tau^2\mathbf{u}_2^n + \tau^3\mathbf{u}_3^n = \ &\mathbf{u}_0^{n+1} + \tau\mathbf{u}_1^{n+1} + \tau^2\mathbf{u}_2^{n+1} + \tau^3\mathbf{u}_3^{n+1}\\ &-\tau\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}^{n+1} + \frac{\tau^2}{2}(\mathbf{d^2}\mathbf{u}_0^{n+1} + \tau\mathbf{d^2}\mathbf{u}_1^{n+1})\\ &-\frac{\tau^3}{6}\mathbf{d^3}\mathbf{u}_0^{n+1} + \mathcal{O}(\tau^4).\end{aligned} \tag{10}$$

Here, we observe that the first derivative of velocity, $\frac{d}{dt}\mathbf{u}^{n+1}$, can be replaced by the momentum equation from Navier-Stokes equations (Eq. (1)) to give:

Asymptotic expansion for the momentum equation

$$\begin{aligned}&\mathbf{u}_0^{n+1} - \mathbf{u}_0^n + \tau\mathbf{u}_1^{n+1} - \tau\mathbf{u}_1^n + \tau^2\mathbf{u}_2^{n+1} - \tau^2\mathbf{u}_2^n + \tau^3\mathbf{u}_3^{n+1} - \tau^3\mathbf{u}_3^n\\ &= \tau\nu\Delta\mathbf{u}_0^{n+1} + \tau^2\nu\Delta\mathbf{u}_1^{n+1} + \tau^3\nu\Delta\mathbf{u}_2^{n+1}\\ &\quad -\tau\nabla p_0^{n+1} - \tau^2\nabla p_1^{n+1} - \tau^3\nabla p_2^{n+1}\\ &\quad -\tau B(\mathbf{u}^{n+1}) - \frac{\tau^2}{2}(\mathbf{d^2}\mathbf{u}_0^{n+1} + \tau\mathbf{d^2}\mathbf{u}_1^{n+1}) + \frac{\tau^3}{6}\mathbf{d^3}\mathbf{u}_0^{n+1} + \mathcal{O}(\tau^4).\end{aligned} \tag{11}$$

Further, we define approximations of high-order derivatives using finite difference schemes as follows:

$$\mathbf{d^2}\mathbf{u}_0^{n+1} = \frac{\mathbf{u}_0^{n+1} - 2\mathbf{u}_0^n + \mathbf{u}_0^{n-1}}{\tau^2} + \mathcal{O}(\tau^2), \quad \text{for } n \geq 0, \tag{12}$$

$$\mathbf{d^2}\mathbf{u}_1^{n+1} = \frac{\mathbf{u}_1^{n+1} - 2\mathbf{u}_1^n + \mathbf{u}_1^{n-1}}{\tau^2} + \mathcal{O}(\tau^2), \quad \text{for } n \geq 1, \tag{13}$$

$$\mathbf{d^3}\mathbf{u}_0^{n+1} = \frac{\mathbf{u}_0^{n+1} - 3\mathbf{u}_0^n + 3\mathbf{u}_0^{n-1} - \mathbf{u}_0^{n-2}}{\tau^3} + \mathcal{O}(\tau), \quad \text{for } n \geq 2. \tag{14}$$

By the linearity of both diffusion and pressure gradient terms and by using a simple ansatz for the nonlinear terms, these terms are expanded in increasing power of τ to produce the following:

$$\begin{aligned}&\left[\frac{\mathbf{u}_0^{n+1} - \mathbf{u}_0^n}{\tau} - \nu\Delta\mathbf{u}_0^{n+1} + B(\mathbf{u}_0^n) + \nabla p_0^{n+1} - \mathbf{f}^{n+1}\right]\\ &\tau\left[\frac{\mathbf{u}_1^{n+1} - \mathbf{u}_1^n}{\tau} - \nu\Delta\mathbf{u}_1^{n+1} + \frac{B(\mathbf{u}_0^{n+1} + \tau\mathbf{u}_1^n) - B(\mathbf{u}_0^n)}{\tau} + \nabla p_1^{n+1} + \frac{1}{2}\mathbf{d^2}\mathbf{u}_0^{n+1}\right]\\ &\tau^2\left[\frac{\mathbf{u}_2^{n+1} - \mathbf{u}_2^n}{\tau} - \nu\Delta\mathbf{u}_2^{n+1} + \frac{B(\mathbf{u}_0^{n+1} + \tau\mathbf{u}_1^{n+1} + \tau^2\mathbf{u}_2^n) - B(\mathbf{u}_0^{n+1} + \tau\mathbf{u}_1^n)}{\tau^2}\right.\\ &\left.+\nabla p_2^{n+1} + \frac{1}{2}\mathbf{d^2}\mathbf{u}_1^{n+1} - \frac{1}{6}\mathbf{d^3}\mathbf{u}_0^{n+1}\right] + \mathcal{O}(\tau^3) = 0,\end{aligned} \tag{15}$$

with the approximation $B(\mathbf{u}^{n+1}) \approx B(\mathbf{u}_0^{n+1} + \tau\mathbf{u}_1^{n+1} + \tau^2\mathbf{u}_2^n)$ since taking $\mathbf{u}_2^n$ instead of $\mathbf{u}_2^{n+1}$ in the nonlinear term gives rise to a semi-implicit method. It can be shown that the global approximation of the nonlinear term has a 3rd-order truncation error if and only if the condition

$$\|\mathbf{u}_2^{n+1} - \mathbf{u}_2^n\|_{L^2(\Omega)} \le c\tau \tag{16}$$

is satisfied (the approximation $\mathbf{u}_2^n$ converges at 1st-order in time) for some real constant $c > 0$ and for all $n \ge 0$. We omit the proof here. This results in Eq. (15) being an approximation to the momentum equation with 3rd-order accuracy in time. A similar approach can be applied to the continuity equation to ensure 3rd-order accuracy.

Asymptotic expansion for the continuity equation

$$\nabla\cdot\mathbf{u}_0^{n+1} + \tau\nabla\cdot\mathbf{u}_1^{n+1} + \tau^2\nabla\cdot\mathbf{u}_2^{n+1} + \mathcal{O}(\tau^3) \quad = 0. \tag{17}$$

Matching the coefficients of τ and using 'divide and conquer' strategy, Eqs. (15) and (17) are recasted into subproblems (18)–(20), which are solved in a multistage manner. For brevity, we provide only the 3rd-order DC schemes (DC-3). By initializing $\mathbf{u}_0^0 = \mathbf{u}(0)$, $\mathbf{u}_1^0 = 0$, $\mathbf{u}_2^0 = 0$, the velocity and pressure are obtained using the following algorithm: Find $(\mathbf{u}_0^{n+1}, p_0^{n+1})$ solution of Eq. (18), $(\mathbf{u}_1^n, p_1^n)$ solution of Eq. (19), $(\mathbf{u}_2^{n-1}, p_2^{n-1})$ solution of Eq. (20) with the final global update $(\mathbf{u}^{n-1}, p^{n-1})$:

$$\text{For } n \ge 0, \quad \begin{cases} \mathbf{nl}_0^{n+1} = \begin{cases} B(\mathbf{u}_0^n), & \text{for } n = 0, 1, \\ B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1}), & \text{for } n \ge 2, \end{cases} \\ \dfrac{\mathbf{u}_0^{n+1} - \mathbf{u}_0^n}{\tau} - \nu\Delta\mathbf{u}_0^{n+1} + \nabla p_0^{n+1} = \mathbf{f}^{n+1} - \mathbf{nl}_0^{n+1}, \\ \nabla\cdot\mathbf{u}_0^{n+1} = 0, \\ \mathbf{du}_0^{n+1} = (\mathbf{u}_0^{n+1} - \mathbf{u}_0^n)/\tau. \end{cases} \tag{18}$$

$$\text{For } n \ge 1, \quad \begin{cases} \mathbf{d^2u}_0^{n+1} = (\mathbf{du}_0^{n+1} - \mathbf{du}_0^n)/\tau, \\ \mathbf{nl}_1^n = \begin{cases} B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1}), & \text{for } n = 1, \\ B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-1}), & \text{for } n \ge 2, \end{cases} \\ \dfrac{\mathbf{u}_1^n - \mathbf{u}_1^{n-1}}{\tau} - \nu\Delta\mathbf{u}_1^n + \nabla p_1^n = -\dfrac{1}{2}\mathbf{d^2u}_0^{n+1} \\ \quad - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\tau}, \\ \nabla\cdot\mathbf{u}_1^n = 0, \\ \mathbf{du}_1^n = (\mathbf{u}_1^n - \mathbf{u}_1^{n-1})/\tau. \end{cases} \tag{19}$$

$$\text{For } n \ge 2, \quad \begin{cases} \mathbf{d^2u}_1^n = (\mathbf{du}_1^n - \mathbf{du}_1^{n-1})/\tau, \quad \mathbf{d^3u}_0^{n+1} \\ = (\mathbf{d^2u}_0^{n+1} - \mathbf{d^2u}_0^n)/\tau, \\ \mathbf{nl}_2^{n-1} = B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2}), \\ \dfrac{\mathbf{u}_2^{n-1} - \mathbf{u}_2^{n-2}}{\tau} - \nu\Delta\mathbf{u}_2^{n-1} + \nabla p_2^{n-1} \\ = -\dfrac{1}{2}\mathbf{d^2u}_1^n + \dfrac{1}{6}\mathbf{d^3u}_0^{n+1} \\ \quad - \dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\tau^2}, \\ \nabla\cdot\mathbf{u}_2^{n-1} = 0, \\ \mathbf{u}^{n-1} = \mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-1}, \\ p^{n-1} = p_0^{n-1} + \tau p_1^{n-1} + \tau^2 p_2^{n-1}. \end{cases} \tag{20}$$

We propose new approximations to handle the nonlinear terms, i.e., $\mathbf{nl}_0^{n+1}$ and $\mathbf{nl}_1^n$ for DC-3 method which are different from the ones proposed in [20]. We found that these new approximations result in a significant reduction of the numerical error while still being 3rd-order accurate on both velocity and pressure. Unlike higher-order SBDF methods which are multistep schemes, DC schemes are self-starting methods. In a hierarchical manner, the high-order derivatives, $\mathbf{d^2u}_0^{n+1}$, $\mathbf{d^2u}_1^n$ and $\mathbf{d^3u}_0^{n-1}$, for $n \ge 0$, $n \ge 1$ and $n \ge 2$, respectively, are evaluated explicitly using Eqs. (12)–(14), to produce a method with kth-order accuracy in time for both velocity and pressure. For any kth-order methods, the structure of the resulting linear system for all subproblems is similar to that of SBDF-1 with a different right hand side. The downside of DC methods arises from the need to solve k saddle point problems at each time step compared to only one for SBDF methods. For a small scale problem, DC may still be efficient since the same matrix is used for all subproblems and can thus be factorized once during the first time step.

2.3. Guermond–Minev method (GM)

The artificial compressibility technique was first introduced by Chorin [9] to eliminate both the undetermined constant in the pressure and complications that arise when solving saddle point systems. This can be done by modifying the continuity equation in Eq. (1) such that we have $\epsilon p_t + \nabla\cdot\mathbf{u} = 0$ where $\epsilon > 0$ is known as the penalization parameter. For illustration, we discretized the time-derivatives for velocity $\mathbf{u}_t$ and pressure p_t using 1st-order backward Euler formula and by fixing $r = \frac{\tau}{\epsilon}$, the iteration parameter. Meanwhile, the nonlinear term is treated explicitly. Using the discretized continuity equation, we eliminate the pressure gradient ∇p_{s+1}^{n+1}, in the momentum equation. It results an augmented Lagrangian method (ALM) for solving the unsteady Navier–Stokes equations, which reads: Given the initial pressure $p_0^0 = p(0)$ and suitable $r > 0$, we iterate the problem for $s = 0, 1, 2, \cdots$

$$\begin{aligned} &\frac{\mathbf{u}_{s+1}^{n+1} - \mathbf{u}_{s+1}^n}{\tau} - \nu\Delta\mathbf{u}_{s+1}^{n+1} - r\nabla\nabla\cdot\mathbf{u}_{s+1}^{n+1} + \mathbf{u}_{s+1}^n\cdot\nabla\mathbf{u}_{s+1}^n \\ &\qquad + \nabla p_s^n = \mathbf{f}, \quad \text{in } \Omega\times(0,T), \\ &p_{s+1}^{n+1} = p_s^{n+1} - r\nabla\cdot\mathbf{u}_{s+1}^{n+1}, \ \text{in } \Omega\times(0,T). \end{aligned} \tag{21}$$

while the stopping criterion $\|p_{s+1}^{n+1} - p_s^{n+1}\|_{L^2(\Omega)} < tol$ is met at each time step t_{n+1}, where $tol > 0$ is chosen small. The convergence of this method is only proven for Stokes equations in [14]. For semi-implicit schemes, this idea is remarkable since only two linear systems with symmetric and positive definite matrices are required to be solved and iterative methods (e.g. conjugate gradient method) work well for such linear systems. For unsteady problem, however, one may predict that the large number of iteration required for the convergence of the pressure at every time step makes the method impractical. Recently, Guermond and Minev [20] proposed a high-order time-stepping schemes based upon artificial compressibility for Navier–Stokes equations. Using defect correction methods and the bootstrapping technique, the resulting uncoupled scheme requires only one update of the pressure at every time step, which produces a significant reduction in CPU time compared to the native implementation of ALM. Without providing technical details on the derivation of this method, the Guermond–Minev scheme with 3rd-order accuracy (GM-3) is given as follows [20]: For a given initial data $(\mathbf{u}_0^0, p_0^0) = (\mathbf{u}(\mathbf{0}), p(0))$ and setting $(\mathbf{u}_1^0, p_1^0) = (0,0)$, $(\mathbf{u}_2^0, p_2^0) = (0,0)$, we solve for $(\mathbf{u}^{n+1}, p^{n+1})$ through the computation of $(\mathbf{u}_0^{n+1}, p_0^{n+1})$, $(\mathbf{u}_1^n, p_1^n)$ and $(\mathbf{u}_2^{n-1}, p_2^{n-1})$

using Eqs. (22), (23) and (24), respectively:

$$\text{For } n \geq 0, \quad \begin{cases} \mathbf{nl}_0^{n+1} = \begin{cases} B(\mathbf{u}_0^n), & \text{for } 0 \leq n \leq 1, \\ B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1}), & \text{for } n \geq 2, \end{cases} \\ \dfrac{\mathbf{u}_0^{n+1} - \mathbf{u}_0^n}{\tau} - \nu \Delta \mathbf{u}_0^{n+1} - \lambda \nabla \nabla \cdot \mathbf{u}_0^{n+1} + \nabla p_0^n \\ = \mathbf{f}^{n+1} - \mathbf{nl}_0^{n+1}, \\ p_0^{n+1} = p_0^n - \lambda \nabla \cdot \mathbf{u}_0^{n+1}, \\ \mathbf{du}_0^{n+1} = (\mathbf{u}_0^{n+1} - \mathbf{u}_0^n)/\tau, \quad dp_0^{n+1} = (p_0^{n+1} - p_0^n)/\tau. \end{cases} \tag{22}$$

$$\text{For } n \geq 1, \quad \begin{cases} \mathbf{d^2 u}_0^{n+1} = (\mathbf{du}_0^{n+1} - \mathbf{du}_0^n)/\tau, \\ \mathbf{nl}_1^n = \begin{cases} B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1}), & \text{for } n = 1, \\ B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}), & \text{for } n \geq 2, \end{cases} \\ \dfrac{\mathbf{u}_1^n - \mathbf{u}_1^{n-1}}{\tau} - \nu \Delta \mathbf{u}_1^n - \lambda \nabla \nabla \cdot \mathbf{u}_1^n + \nabla (p_1^{n-1} + dp_0^n) \\ = -\dfrac{1}{2} \mathbf{d^2 u}_0^{n+1} - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\tau}, \\ p_1^n = p_1^{n-1} + dp_0^n - \lambda \nabla \cdot \mathbf{u}_1^n, \\ \mathbf{du}_1^n = (\mathbf{u}_1^n - \mathbf{u}_1^{n-1})/\tau, \quad dp_1^n = (p_1^n - p_1^{n-1})/\tau. \end{cases} \tag{23}$$

$$\text{for } n \geq 2, \quad \begin{cases} \mathbf{d^2 u}_1^n = (\mathbf{du}_1^n - \mathbf{du}_1^{n-1})/\tau, \\ \mathbf{d^3 u}_0^{n+1} = (\mathbf{d^2 u}_0^{n+1} - \mathbf{d^2 u}_0^n)/\tau, \\ \mathbf{nl}_2^{n-1} = B(\mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-2}), \\ \dfrac{\mathbf{u}_2^{n-1} - \mathbf{u}_2^{n-2}}{\tau} - \nu \Delta \mathbf{u}_2^{n-1} - \lambda \nabla \nabla \cdot \mathbf{u}_2^{n-1} \\ + \nabla (p_2^{n-2} + dp_1^{n-1}) \\ \quad = -\dfrac{1}{2} \mathbf{d^2 u}_1^n + \dfrac{1}{6} \mathbf{d^3 u}_0^{n+1} - \dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\tau^2}, \\ p_2^{n-1} = p_2^{n-2} + dp_1^{n-1} - \lambda \nabla \cdot \mathbf{u}_2^{n-1}, \\ \mathbf{u}^{n-1} = \mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}, \\ p^{n-1} = p_0^{n-1} + \tau p_1^{n-1} + \tau^2 p_2^{n-1}. \end{cases} \tag{24}$$

The stability and convergence analysis of the method is done only for the linear problem (i.e $B(\mathbf{u}) = 0$) in [20]. In later section, we will illustrate through our numerical results that the method reaches the expected rate of convergence and is stable with a suitable choice of time step on a given mesh.

2.4. Guermond-Minev with sequential regularization method (GM-SRM)

We propose a modification of the GM method using the so-called sequential regularization method (SRM) which was first investigated by [30,32] considering that Navier–Stokes equations is a system of differential-algebraic equations (DAE) of index 2. This gives rise to the new formulation of Navier-Stokes equations which reads

$$\begin{aligned} &\mathbf{u}_t^{s+1} - \nu \Delta \mathbf{u}^{s+1} - \alpha_1 \nabla \nabla \cdot \mathbf{u}_t^{s+1} - \alpha_2 \nabla \nabla \cdot \mathbf{u}^{s+1} + \mathbf{u}^{s+1} \cdot \nabla \mathbf{u}^{s+1} \\ &\quad + \nabla p^s = \mathbf{f}, \quad \text{in } \Omega \times (0, T), \\ &\quad p^{s+1} = p^s - \alpha_1 \nabla \cdot \mathbf{u}_t^{s+1} - \alpha_2 \nabla \cdot \mathbf{u}^{s+1}, \ \text{in } \Omega \times (0, T). \end{aligned} \tag{25}$$

This problem can be solved using any suitable time-stepping scheme and stopping criteria as for ALM. Both $\alpha_1, \alpha_2 \geq 0$ are user-defined stabilization parameters for SRM. By choosing $\alpha_1 = 0$ and $\alpha_2 = \lambda$, we fall back on the GM methods. The time-derivative of divergence of $\mathbf{u}$, i.e., $\nabla \cdot \mathbf{u}_t$, is discretized using similar Taylor series expansion as in Eq. (9). We retain the high-order derivative terms for defect correction purposes, as was done for the other terms in the PDE. The method GM-SRM with 3rd-order accuracy is given as follows: For a given initial data $(\mathbf{u}_0^0, p_0^0) = (\mathbf{u}(\mathbf{0}), p(0))$ and setting $(\mathbf{u}_1^0, p_1^0) = (0, 0)$, $(\mathbf{u}_2^0, p_2^0) = (0, 0)$, we solve for $(\mathbf{u}^{n+1}, p^{n+1})$ through the computation of $(\mathbf{u}_0^{n+1}, p_0^{n+1})$, $(\mathbf{u}_1^n, p_1^n)$ and $(\mathbf{u}_2^{n-1}, p_2^{n-1})$ using Eqs. (26), (27) and (28), respectively:

$$\text{For } n \geq 0, \quad \begin{cases} \mathbf{nl}_0^{n+1} = \begin{cases} B(\mathbf{u}_0^n), & \text{for } 0 \leq n \leq 1, \\ B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1}), & \text{for } n \geq 2, \end{cases} \\ \dfrac{\mathbf{u}_0^{n+1} - \mathbf{u}_0^n}{\tau} - \nu \Delta \mathbf{u}_0^{n+1} - \mu_1 \nabla \nabla \cdot \mathbf{u}_0^{n+1} \\ + \mu_2 \nabla \nabla \cdot \mathbf{u}_0^n + \nabla p_0^n \\ \quad = \mathbf{f}^{n+1} - \mathbf{nl}_0^{n+1}, \\ p_0^{n+1} = p_0^n - \mu_1 \nabla \cdot \mathbf{u}_0^{n+1} + \mu_2 \nabla \cdot \mathbf{u}_0^n, \\ \mathbf{du}_0^{n+1} = (\mathbf{u}_0^{n+1} - \mathbf{u}_0^n)/\tau, \quad dp_0^{n+1} = (p_0^{n+1} - p_0^n)/\tau. \end{cases} \tag{26}$$

$$\text{For } n \geq 1, \quad \begin{cases} \mathbf{d^2 u}_0^{n+1} = (\mathbf{du}_0^{n+1} - \mathbf{du}_0^n)/\tau, \\ \mathbf{nl}_1^n = \begin{cases} B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1}), & \text{for } n = 1, \\ B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}), & \text{for } n \geq 2, \end{cases} \\ \dfrac{\mathbf{u}_1^n - \mathbf{u}_1^{n-1}}{\tau} - \nu \Delta \mathbf{u}_1^n - \mu_1 \nabla \nabla \cdot \mathbf{u}_1^n + \mu_2 \nabla \nabla \cdot \mathbf{u}_1^{n-1} \\ \quad + \nabla (p_1^{n-1} + dp_0^n) = -\dfrac{1}{2} \\ (\mathbf{d^2 u}_0^{n+1} - \mu_2 \tau \nabla \nabla \cdot \mathbf{d^2 u}_0^{n+1}) \\ \quad - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\tau}, \\ p_1^n = p_1^{n-1} + dp_0^n - \mu_1 \nabla \cdot \mathbf{u}_1^n + \mu_2 \nabla \cdot \mathbf{u}_1^{n-1} \\ - \dfrac{\mu_2 \tau}{2} \nabla \cdot \mathbf{d^2 u}_0^{n+1}, \\ \mathbf{du}_1^n = (\mathbf{u}_1^n - \mathbf{u}_1^{n-1})/\tau, \quad dp_1^n = (p_1^n - p_1^{n-1})/\tau. \end{cases} \tag{27}$$

$$\text{for } n \geq 2, \quad \begin{cases} \mathbf{d^2 u}_1^n = (\mathbf{du}_1^n - \mathbf{du}_1^{n-1})/\tau, \\ \mathbf{d^3 u}_0^{n+1} = (\mathbf{d^2 u}_0^{n+1} - \mathbf{d^2 u}_0^n)/\tau, \\ \mathbf{nl}_2^{n-1} = B(\mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-2}), \\ \dfrac{\mathbf{u}_2^{n-1} - \mathbf{u}_2^{n-2}}{\tau} - \nu \Delta \mathbf{u}_2^{n-1} - \mu_1 \nabla \nabla \cdot \mathbf{u}_2^{n-1} \\ + \mu_2 \nabla \nabla \cdot \mathbf{u}_2^{n-2} \\ \quad + \nabla (p_2^{n-2} + dp_1^{n-1}) = -\dfrac{1}{2} (\mathbf{d^2 u}_1^n \\ - \mu_2 \tau \nabla \nabla \cdot \mathbf{d^2 u}_1^n) \\ \quad + \dfrac{1}{6} (\mathbf{d^3 u}_0^{n+1} - \mu_2 \tau \nabla \nabla \cdot \mathbf{d^3 u}_0^{n+1}) \\ - \dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\tau^2}, \\ p_2^{n-1} = p_2^{n-2} + dp_1^{n-1} - \mu_1 \nabla \cdot \mathbf{u}_2^{n-1} + \mu_2 \nabla \cdot \mathbf{u}_2^{n-2} \\ - \dfrac{\mu_2 \tau}{2} \nabla \cdot \mathbf{d^2 u}_1^n + \dfrac{\mu_2 \tau}{6} \nabla \cdot \mathbf{d^3 u}_0^{n+1}, \\ \mathbf{u}^{n-1} = \mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}, \\ p^{n-1} = p_0^{n-1} + \tau p_1^{n-1} + \tau^2 p_2^{n-1}. \end{cases} \tag{28}$$

where $\mu_1 = \dfrac{\alpha_1}{\tau} + \alpha_2$ and $\mu_2 = \dfrac{\alpha_1}{\tau}$. The existence and uniqueness of the solution of SRM formulation for time-dependent Navier–Stokes equations was studied in [32] for 1st-order approximations in time. There are very few numerical test cases available in the literature for SRM applied to unsteady flows, not to mention when this involves high-order methods in time. In this paper, we intend to fill this gap.

3. Stabilization parameter for GM and GM-SRM

The **grad** – div stabilization term arises naturally in GM and GM-SRM methods since the Navier–Stokes equations are solved by uncoupling the velocity and pressure. In the continuous setting of the Navier–Stokes equations, the presence of **grad** – div term in the momentum equation does not change the solution. In the discrete setting, however, the **grad** – div term does not change the solution only under certain conditions, for instance when finite elements lie within the divergence free subspace or when $\nabla \cdot$

$V_h \subset M_h$. In particular, when $\mathbb{P}_2 - \mathbb{P}_1$ finite elements are used, we have $\nabla \cdot V_h \subseteq Z_h \not\subset M_h$, where

$$Z_h = \{\mathbf{v}_h \in L^2(\Omega) \mid \mathbf{v}_{h|K} \in \mathbb{P}_1, \ \forall K \in \mathrm{T}_h\}. \tag{29}$$

For the discrete weak formulation of GM and GM-SRM, the **grad** – div term in the momentum equation cannot be eliminated as is done from the continuity equation at the continuous level. This explains that for both GM and GM-SRM methods, the computed solution may differ from that with SBDF and DC methods even when we let $\tau \to 0$, with h fixed. When solving flow problem with $\mathbb{P}_2 - \mathbb{P}_1$ elements on a coarser grid, it is known that the **grad** – div term helps to penalize the incompressibility condition at the element level hence improving the overall quality of the solutions. Several other advantages of adding the **grad** – div stabilization term for solving Stokes equations have been addressed by Olshanskii et al. [35,36] and references therein. Our work will only provide a comparison in terms of the solution and numerical behaviour between our methods with **grad** – div and without **grad** – div on a given mesh. The numerical improvement of $\mathbb{P}_2 - \mathbb{P}_1$ elements by the addition of **grad** – div term will be addressed in a separate work.

A challenging issue when using GM and GM-SRM is the choice of stabilization parameter, i.e., λ for GM, α_1 and α_2 for GM-SRM, which remains a controversial issue. For instance, by picking a large λ, it helps to enforce the incompressibility condition rapidly (in very few time steps). However, the choice of such large stabilization parameter may result in the poor conditioning of the linear system due to the "locking" or very stiff behaviour imposed by the **grad** – div term in the momentum equation [20]. A similar problem was studied by Glowinski and Fortin [14] but involving only Stokes equation with ALM. Our linear system for the momentum equation (in GM and GM-SRM) is almost identical with theirs except for the presence of an additional diagonal block with the mass matrix from the time discretization in our case. They had proven that a large "iterative parameter" akin to our stabilization parameter results in poor convergence behaviour (ill-conditioned matrix) when iterative solvers are used to solve this linear system. Since we use a direct solver for the linear system, this is not a major issue. Large stabilization parameter is also related to over-stabilization effects which induces numerical dissipation and lost of accuracy [35]. Small λ on the other hand, may complicate the convergence of pressure between momentum and continuity equation. This results in parasitic oscillations on both pressure and velocity due to poor mass conservation. GM and GM-SRM methods do not have additional iterations between subproblems to ensure such convergence is satisfied, as is done with ALM. The incompressibility condition is enforced only with bootstrapping technique which critically depends on a well-chosen stabilization parameter. As similar problems arise for GM-SRM methods, the choice of α_1 and α_2 remains an open problem.

At this moment, we are not able to establish a mathematical theory to determine an optimal value of these parameters in terms of producing the least numerical error at a given time step. In our work, best values of the parameters, λ, α_1 and α_2 can be estimated numerically (trial and error). For instance in GM, we start by picking very large λ and observing the velocity and pressure during first few time steps. If no oscillations are found, computations are rerun with smaller λ. The 'optimal' λ is chosen by picking the one that starts to induce oscillations but that progressively dampens the oscillations from one time step to the next. For the GM-SRM method, a good combination of α_1 and α_2 is harder to identify. For simplicity, we propose $\alpha_1 = \tau\alpha_2$ for test cases which require larger α_2 (e.g., $\alpha_2 \gg 1$) while $\alpha_1 = 1$ is typically chosen when only smaller α_2 is required (e.g., $\alpha_2 = 1$). We only need to determine the value of α_2 using an approach that is similar to what is done to obtain λ in the GM method. Nonetheless, this approach delivers good results in our computations. The nearly optimal values of λ in GM and α_1 and α_2 in GM-SRM are provided for each of the test cases.

4. Numerical results

Numerical validation and benchmarking are done for the time-stepping schemes SBDF, DC, GM and GM-SRM, all with 2nd- and 3rd-order accuracy. All our tests are done with 2D flows. We start with two manufactured solutions, one involving Dirichlet–Neumann boundary conditions adapted from [20] and the Taylor's Vortex flow with Dirichlet boundary conditions [38]. For test cases which are deemed to be more challenging, the flow around a cylinder (von Kármán alley) (see for instance [19,38,48]) and the lid-driven cavity flow (see [18,37,43]) are chosen. Since exact solutions do not exist for these last two test cases, we made comparisons with reference solutions and the results found in the literature.

Let us define $t^n = n\tau^*$ and $T = M\tau^*$ where T is total simulation time, τ^* the time step for output purposes and $M \in \mathbb{N}$ given by the user. For convenience, we choose τ^* as a multiple of τ. The error in time can be computed using the reference solutions $(\mathbf{u}_{\mathrm{ref}}, p_{\mathrm{ref}})$ where $\mathbf{u}_{\mathrm{ref}} = \mathbf{u}_{h,\tau_{\mathrm{ref}}}$ and $p_{\mathrm{ref}} = p_{h,\tau_{\mathrm{ref}}}$ are the best approximants to the solution of the semi-discrete problem (in space). Very small time step, given as τ_{ref} is used to compute the reference solutions. The error from the time discretization, by fixing the discretization in space, can be computed as follows:

$$\begin{aligned} \|\mathbf{u}_{\mathrm{ref}} - \mathbf{u}_{h,\tau}\|^*_{L^2(0,T;L^2(\Omega))} &= \left(\tau^* \sum_{n=1}^{M} \|\mathbf{u}^n_{\mathrm{ref}} - \mathbf{u}^n_{h,\tau}\|_{[L^2(\Omega)]^2}\right)^{\frac{1}{2}}, \\ \|p_{\mathrm{ref}} - p_{h,\tau}\|^*_{L^2(0,T;L^2(\Omega))} &= \left(\tau^* \sum_{n=1}^{M} \|p^n_{h,\tau} - p^n_{\mathrm{ref}}\|_{L^2(\Omega)}\right)^{\frac{1}{2}}. \end{aligned} \tag{30}$$

This type of error is useful when exact solutions are not available and to study the order of convergence in time only. Due to the numerical discrepancies found between methods with **grad** – div and without **grad** – div term, we compare the errors produced by DC and SBDF methods with the reference solution that is generated by DC-3. On the other hand, we consider a reference solution that is generated by GM-3 in the assessment of the GM and GM-SRM methods. The formula to obtain the order of convergence in time, k, for velocity and pressure (respectively $w = \mathbf{u}$ and $w = p$) reads:

$$k = \ln\left(\frac{\|w_{\mathrm{ref}} - w_{h,\tau}\|^*_{L^2(0,T;L^2(\Omega))}}{\|w_{\mathrm{ref}} - w_{h,\tau/r}\|^*_{L^2(0,T;L^2(\Omega))}}\right) \Big/ \ln r, \tag{31}$$

where $r \in \mathbb{R}^+$ is the refinement factor between consecutive time steps. The value $r = 2$ was chosen. For the finite element approximation, the mesh is assumed to be quasi-uniform, i.e., $h_{\min} \le h \le h_{\max}$. Suppose that the space approximation is done using $\mathbb{P}_2$–$\mathbb{P}_1$ mixed finite elements with sufficient regularity on $(\mathbf{u}, p)$, we expect that the L^2-error between numerical and exact solutions using kth-order time-stepping schemes to behave as

$$\begin{aligned} \|(\mathbf{u} - \mathbf{u}_{h,\tau})(t_i)\|_{L^2(\Omega)} &= \mathcal{O}(\tau^k + h^3), \quad 1 \le i \le N, \\ \|(p - p_{h,\tau})(t_i)\|_{L^2(\Omega)} &= \mathcal{O}(\tau^k + h^2), \quad 1 \le i \le N, \end{aligned} \tag{32}$$

where $N = T/\tau \in \mathbb{N}$ is the total number of time step. Similarly, the L^2-error between numerical and reference semi-discrete solutions obtained on the same mesh is expected to have the following order of accuracy

$$\begin{aligned} \|(\mathbf{u}_{\mathrm{ref}} - \mathbf{u}_{h,\tau})(t_i)\|_{L^2(\Omega)} &= \mathcal{O}(\tau^k), \quad 1 \le i \le N, \\ \|(p_{\mathrm{ref}} - p_{h,\tau})(t_i)\|_{L^2(\Omega)} &= \mathcal{O}(\tau^k), \quad 1 \le i \le N. \end{aligned} \tag{33}$$

The critical time step, τ_{crit} for numerical stability of each time-stepping scheme is computed using a bisection method. For instance, the initial time step, τ_0 is chosen large enough to result

in a numerical blow-up after a few time steps. Next, same computation is carried out using the time step, $\tau_1 = 0.5\tau_0$. If the numerical stability is observed for a chosen total time T, computation is restarted again with the larger time-step $\tau_2 = 1.5\tau_1$, otherwise we set $\tau_2 = 0.5\tau_1$ and the process is repeated. Typically, $T = 10$ is chosen to account for nonlinear instability effects that may show-up in the computation. A fully developed periodic flow is used as initial condition. To obtain a converged critical time step τ_{crit}, the stopping criteria, i.e., $|\tau_{n+1} - \tau_n| < tol_\tau$, is used for some small $tol_\tau > 0$. All computations are done using FreeFEM++ [22]. We intend to focus only on the performance of the time-stepping schemes, hence the impact of linear solvers and preconditioners are not taken into consideration. We used MUMPS (Multifrontal Massively Parallel Sparse Direct Solver) which is one of the best direct solver available in FreeFEM++. MUMPS is chosen as it turns out to be twice as fast as the Unsymmetric MultiFrontal method (UMFPACK). All simulations are run under a Linux-platform PC with a Intel®Core™ i7-3770 CPU 3.40 GHz and 32Gb of RAM. We compare the numerical efficiency of the time-stepping methods by looking at the CPU time (total runtime) required to reach a desired level of error in time. Some of the methods may be very efficient when the error is large but surpassed by others when smaller error is required. Relative efficiency depends on several factors, namely the numerical complexity (i.e., number of subproblems and linear systems to be solved per time step), accuracy, stability condition (which determines the size of the maximal time step) and convergence behaviour of the methods. For 2D problems, the memory requirement of all these methods can be easily fulfilled by modern desktop computers. However, this may not be true for 3D problems.

4.1. Manufactured solution I

This manufactured solution is an example where the discretization error in time is larger than that in space which was presented in [20]. For this test case, we use the square domain $\Omega = (0, 1) \times (0, 1)$. The exact solutions for velocity and pressure are given by

$$\mathbf{u} = (\sin(x)\ \sin(y+t), \cos(x)\ \cos(y+t)),$$
$$p = \cos(x)\ \sin(y+t), \tag{34}$$

for $(x, y) \in \Omega, t \geq 0$. Homogeneous Neumann boundary conditions are prescribed at the left boundary while homogeneous Dirichlet boundary conditions (no-slip) are prescribed at bottom, right and upper boundaries. By fixing the viscosity constant, $\nu = 1$, initial and boundary conditions, and $\mathbf{f}$ can be computed using Eq. (34). The domain is discretized using an unstructured mesh with uniform mesh size $h = 0.01768$ (subdivision $80\times$ 80 along the boundary). This gives a total of 6561 vertices, 12800 triangles and 58403 degrees of freedom for velocity and pressure. This test case is run for a total time $T = 2$ with varying time steps, $\tau = 0.02, 0.01, 0.005$ and 6.25×10^{-5}. The smallest time step is used to generate the reference solution ($\mathbf{u}_{\text{ref}}$, p_{ref}) for computing the error (in time only) in the convergence analysis. For the stabilization parameters, we fix $\lambda = 83.5$ in GM methods while $\alpha_1 = 40\tau$ and $\alpha_2 = 40.0$ are used in GM-SRM methods.

Fig. 1 shows the time evolution of log-scaled L^2-error computed with Eq. (30) for all 2nd- and 3rd-order methods, both for velocity and pressure with $\tau = 0.02, 0.01$ and 0.005. These plots confirm as expected, the following facts: the accuracy of the velocity obtained with all methods is better than that of pressure since $\mathbb{P}_2 - \mathbb{P}_1$ finite element is used; the 3rd-order methods are more accurate than their 2nd-order counterparts at any fixed time $t \in [0, 2]$ and the numerical errors for velocity and pressure decrease as the time step is refined. For $\tau = 0.02$, all graphs for 3rd-order methods (except SBDF-3 for velocity) are superposed with the lowest curve, where only the error from the space approximation remains. The error of velocity for SBDF-3 is still large with the time step $\tau = 0.02$. We suspect that this error is induced from a time step chosen too close to the critical value for SBDF-3 method. The critical time step for SBDF-3 is known to be smaller than that for other 3rd-order methods to fulfill the CFL stability condition. With the smallest time step, $\tau = 0.005$, both DC-3 and SBDF-3 produce the least error for both velocity and pressure, followed by GM-3 and GM-SRM-3. Here, GM-SRM produces very similar errors to that computed with GM for both 2nd- and 3rd-order accuracy as computations proceed further from the start-up. However, it can be observed that numerical oscillations on the solutions of GM-3 (and consequently on the error) at start-up require more time steps to attenuate completely compared to GM-SRM-3, which suggests that GM-SRM-3 may have better stability property than GM-3 in terms of enforcing divergence free condition at element level. For $\tau = 0.005$, we observed that the graph produced with all methods except GM-2, GM-SRM-2 and SBDF-2 methods collapse to the bottom graph with space error only except for minor discrepancies due to the presence of the **grad** − div term in GM and GM-SRM. We note that the error produced by GM and GM-SRM can be reduced further and brought closer to minimal level by choosing a slightly smaller stabilization parameter, e.g., $\lambda < 83.5$ and $\alpha_2 < 40.0$. A disadvantage for this, however, is by having more severe oscillations generated at the beginning of the computations which require more time steps to be damped.

We analyze the numerical efficiency and convergence in time of the scheme, i.e., by comparing the numerical and reference solutions for all methods (see Fig. 2). The numerical efficiency can be studied by considering first, the L^2-error versus CPU time and second, the L^2-error versus the time step, τ for both velocity and pressure. These errors are computed using Eq. (30). To disregard possible pressure oscillations generated by GM and GM-SRM methods at start-up, we evaluate the L^2-error using $\tau^* = 0.2$ only for $t \in [1, 2]$. Fig. 2 shows that all 3rd-order methods are more efficient than 2nd-order methods for both velocity and pressure as the accuracy reached for a given CPU time is always better with 3rd-order methods. Among all methods, the accuracy of SBDF is found to be the lowest among all methods of the same order. However, we noted that the accuracy of SBDF methods can be improved by implementing different discretizations of the nonlinear term. Further discussion on this will be provided in a separate paper. At any given level of error, DC-3 method is found to be the most efficient, for instance it can reach an error of about 10^{-8} for velocity and about 10^{-6} for pressure with shortest CPU time, closely followed by GM-3, GM-SRM-3 and SBDF-3 methods. It is noteworthy that DC-3 method requires to solve three saddle point problems at every time step, and still is the most efficient for this test case. With a small number of unknowns, the use of a direct solver (e.g., MUMPS) makes these computations efficient. For a fixed time step and order of the method, DC, GM and GM-SRM methods produce smaller error in time than that with SBDF methods since they are all based upon defect correction strategy. Fig. 2 (at the bottom) shows that all time-stepping schemes converge in time with the theoretical rate for both velocity and pressure.

4.2. Manufactured solution II (Taylor Vortex)

The Taylor vortex is a manufactured solution modeling flow problems with Dirichlet boundary conditions [28] which it is often used for numerical validation. It models a decaying flow in time and exhibits a dominating error in space over the error in time. This test case mimics industrial fluid stirring mechanism using rotating and counter-rotating cylinder rods in a mixing tank. In our test case, the square domain $\Omega = (0.25, 1.25) \times (0.5, 1.5)$ is used

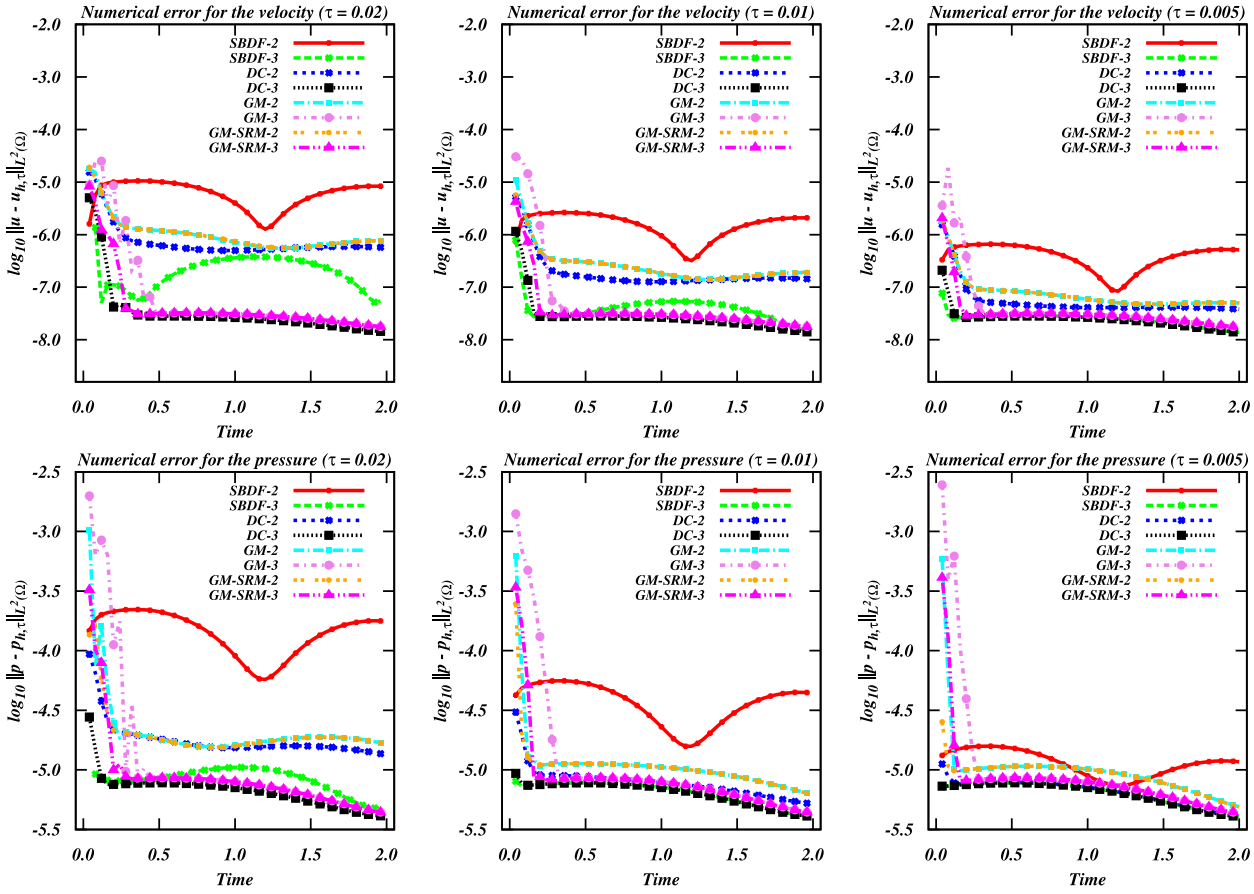

Fig. 1. Manufactured solution I: Time history of L^2-error for velocity and pressure between numerical and exact solutions using 2nd- and 3rd-order SBDF, DC, GM, GM-SRM schemes (From left to right, the time steps $\tau = 0.02$, 0.01 and 0.005 are used).

with the exact solutions for velocity and pressure given by

$$\mathbf{u} = (-\cos(2\pi x)\sin(2\pi y)e^{-8\pi^2\nu t}, \sin(2\pi x)\cos(2\pi y)e^{-8\pi^2\nu t}),$$
$$p = -\frac{1}{4}(\cos(4\pi x) + \cos(4\pi y))e^{-16\pi^2\nu t}, \tag{35}$$

in which the external force, $\mathbf{f} \equiv 0$. We set the kinematic viscosity, ν to 0.01. Dirichlet boundary conditions which are prescribed on the four edges of the square and initial conditions can be obtained through the exact solution (Eq. (35)). The domain is discretized with an unstructured triangular mesh by subdividing each side of the square in 120 equal length edges. With $\mathbb{P}_2 - \mathbb{P}_1$ finite elements, a total of 34210 elements, 17346 vertices and 155148 degrees of freedom for both velocity and pressure are produced. The element size, h_K lies in the range $0.00645 \leq h_K \leq 0.01518$. Since this is an exponentially decaying solutions, we limit the simulation only to $t \in [0, 1]$. We fix $\lambda = 3.3$ in GM methods and $\alpha_1 = \alpha_2 = 1$ in GM-SRM methods. The numerical comparisons are done in a similar fashion as in Section 4.1. For the error (in time) and convergence analysis, we only consider the solution generated at every step $\tau^* = 0.2$ for $t \in [0.6, 1.0]$ (using Eq. (30)).

Fig. 3 shows the time evolution of the L^2-error on velocity and pressure comparing with the exact solutions, computed by various methods with time steps $\tau = 0.02, 0.01, 0.005$. Remarks are similar as for the previous test case, except for the observation that GM-SRM methods slightly outperform other methods of the same order since they have the smallest error on velocity. This suggests that the presence of a **grad** − div term with nearly optimal choice of the stabilization parameters helps to enforce the incompressibility condition while improving the accuracy on velocity (in space) by about one order of magnitude compared to other methods. We observe again that the accuracy of SBDF-3 is compromised when the time step is chosen too close to the critical time step for stability (see top left graph in Fig. 3). Fig. 4 illustrates the numerical efficiency of the methods by showing graphs of the L^2-error (in time only) versus the required CPU time and chosen time step, respectively, for all methods. Here, we set $T = 1$ and a very small time step $\tau = 6.25 \times 10^{-5}$ to compute the reference solution. The numerical rate of convergence of all time-stepping schemes shows a satisfactory agreement with the theoretical rate of convergence in time for both velocity and pressure. At larger time step (e.g., $\tau > 0.01$), SBDF-3 does not exhibit the expected order of convergence when the chosen time step is too close to the critical time step. However, the SBDF-3 method is found to quickly surpass DC-3, GM-3 and GM-SRM-3 methods for errors below 1×10^{-6} and 3×10^{-7} on velocity and pressure, respectively. At any given order of accuracy, SBDF methods are found to be the most efficient for both velocity and pressure at least when sufficiently small time steps are used. Interestingly, SBDF methods produce the smallest error on velocity in time at smaller time step, e.g., $\tau < 0.01$ for this test case. This is observed since the initialization procedure is done using high-order method, i.e., DC methods. For pressure, GM, GM-SRM and DC methods produced smaller error than SBDF methods for a given time step and order of accuracy.

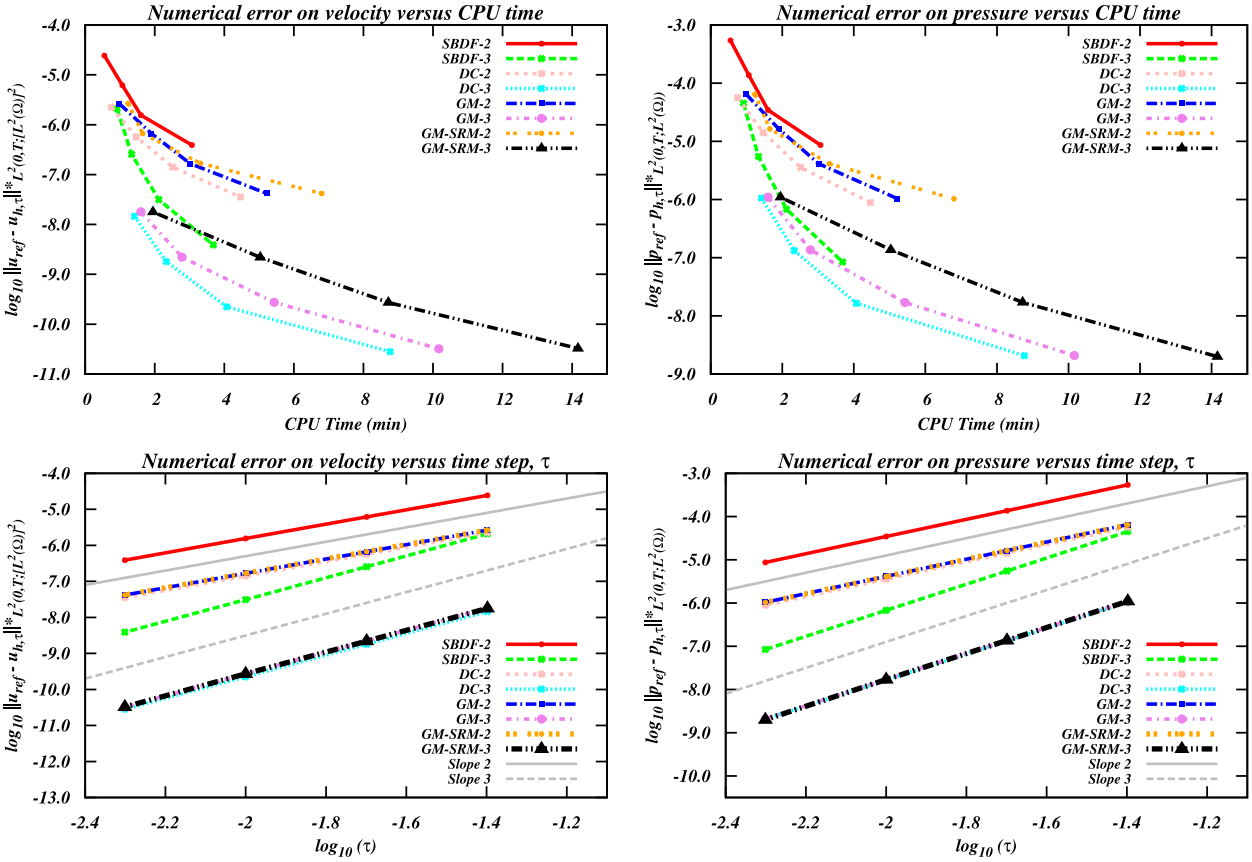

Fig. 2. Manufactured Solution I: Plot of the error on velocity (left) and pressure (right) as a function of CPU time (top) and time-step (bottom) for SBDF, DC, GM, GM-SRM methods.

4.3. Flow past a circular object: von Kármán Alley

The flow past a circular cylinder is one of the most famous flow experiments in fluid dynamics (see e.g. [23,38,42,48]). In many numerical studies for Navier–Stokes equations, this test case has been used for benchmaking purposes, since only accurate numerical methods are able to reproduce the right flow properties. The categorization of vortex shedding behind the circular cylinder such as steady or unsteady laminar, subcritical and turbulent flows are strictly determined by the Reynolds numbers

$$Re = \frac{\mathbf{U}_\infty D}{\nu}, \tag{36}$$

where D is the diameter of the cylinder, $\mathbf{U}_\infty$ the far-field speed and ν the kinematic viscosity. The wake behind the cylinder gives rise to a periodic flow with symmetry breaking whenever $40 < Re < 150$. The so-called laminar von Kármán vortex sheet is then fully developed. To quantify the periodicity of the flow, the Strouhal number is used, i.e., $Str = \frac{fD}{\mathbf{U}_\infty}$ where f is the frequency of the vortex shedding.

There are several ways to set the computational domain for this flow. In this paper, we use a conventional rectangular domain, $\Omega = (-10, 25) \times (-10, 10)$ which is discretized by a total of 26936 non-uniform triangles (unstructured mesh) and 13648 vertices. The circular cylinder is located at the origin with $D = 1$. As $\mathbb{P}_2 - \mathbb{P}_1$ finite elements are used for space discretization, there are 108474 unknowns for the velocity $\mathbf{u} = (u, v)$ and 13648 unknowns for the pressure p. The varying element size, h_K, lies in the interval [0.05129, 0.61148], with a sufficiently fine mesh generated near the cylinder to capture the boundary layer and coarsest element along the outer boundary (see Fig. 5). Setting $\mathbf{U}_\infty = 1$ and $\nu = 0.01$, a Reynolds number, $Re = 100$ is obtained. For GM methods, the stabilization parameter $\lambda = 10^4$ is considered. For GM-SRM methods, typically smaller stabilization parameters $\alpha_1 = 200\tau$ and $\alpha_2 = 200$ are chosen. The boundary conditions are prescribed as follows: along the left, upper and lower boundaries, Dirichlet boundary conditions $\mathbf{u} = (u, v) = (1, 0)$ are set. Upper and lower boundaries in this paper are dealt differently from what is done in [23,33,38,48]. In these papers, non-slip boundary $\mathbf{u} = (0, 0)$ or $\partial_x u = 0$ with $v = 0$ were used along the upper and lower boundaries on much smaller computational domain. Neumann boundary conditions (free exit) are employed along the right outflow boundary, i.e., $p - \nu \partial_x u|_{\Gamma_N} = 0$ and $-\nu \partial_x v|_{\Gamma_N} = 0$ where $\Gamma_N = \{(x, y) \in \Gamma \mid x = 25\}$. Non-slip boundary condition $\mathbf{u} = (u, v) = (0, 0)$ is prescribed on the circular cylinder. Starting from an initial state at rest, a periodic motion is reached for about $T = 80$. This periodic flow then shows vortex shedding behind the cylinder propagating all the way to the outflow boundary. To reduce the computational time, we use the periodic solution that is computed with SBDF-2 as the initial state for all subsequent numerical tests. Since there is no analytical solution for this test case, we use a reference solution $(\mathbf{u}_{\text{ref}}, p_{\text{ref}})$ to assess all our methods.

In Fig. 6, the numerical efficiency is assessed by plotting the L^2-error (in time) on velocity and pressure versus the CPU time for completing the computation for $t \in [0, 8]$ starting from the peri-

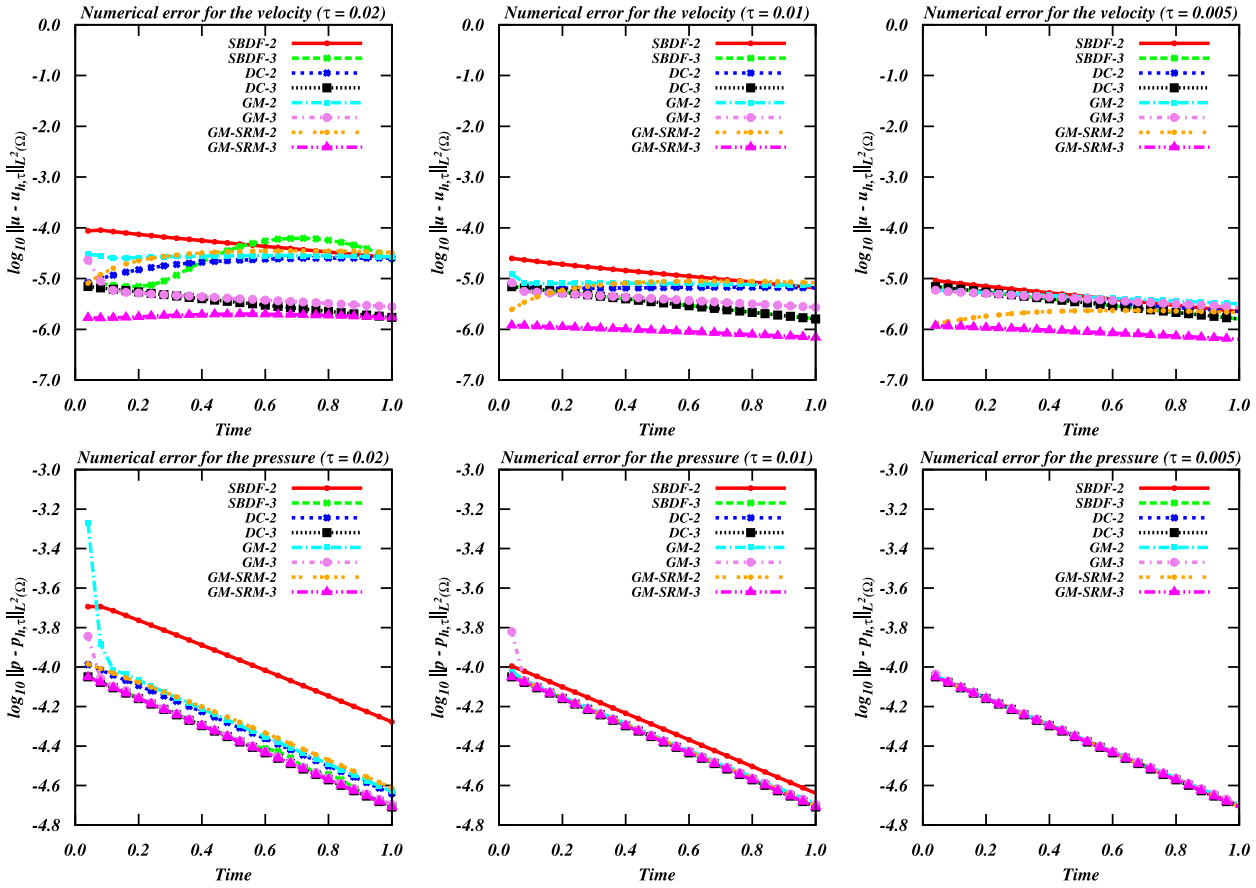

Fig. 3. Manufactured solution II: Time history of L^2-error for velocity and pressure between numerical and exact solutions using 2nd- and 3rd-order SBDF, DC, GM, GM-SRM schemes (From left to right, the time steps $\tau = 0.02$, 0.01 and 0.005 are used).

Table 1
Computed critical time step and maximum CFL bound for stability.

Methods	Critical time step, τ_{crit}	CFL_{max}
SBDF-1	1.7744×10^{-2}	4.6319×10^{-1}
SBDF-2	1.1951×10^{-2}	4.1453×10^{-1}
SBDF-3	7.0278×10^{-3}	1.8320×10^{-1}
DC-2	1.1551×10^{-2}	3.0124×10^{-1}
DC-3	8.1168×10^{-3}	2.1156×10^{-1}
GM-2	1.4529×10^{-2}	3.7814×10^{-1}
GM-3	1.0537×10^{-2}	2.7438×10^{-1}
GM-SRM-2	1.4522×10^{-2}	3.7697×10^{-1}
GM-SRM-3	1.0525×10^{-2}	2.7407×10^{-1}

odic solution mentioned above. We also illustrate the rate of convergence for each of the time-stepping schemes in the same figure. For a given time step, DC, GM and GM-SRM methods produce the least numerical errors for both velocity and pressure in time, then followed by SBDF methods. However, SBDF methods are the most CPU-efficient schemes, followed by DC, GM and GM-SRM methods at a given order of accuracy. All methods reproduce the theoretical rate of convergence. Table 1 summarizes the critical time steps τ_{crit} for each of the time-stepping schemes that are obtained using the bisection method. In addition, we provide the maximal CFL number or the CFL_{max} attained in a local element which can be computed using the following:

$$\text{CFL}_{\text{max}} = \max_{K \in \mathcal{T}_h} \left\{ \frac{\tau_{\text{cri}}}{h_K} \|\mathbf{u}_h\|_{L^2(K)} \right\}, \tag{37}$$

where each K is an element of the mesh $\mathcal{T}_h$. The critical time step (or equivalently the CFL_{max}) of GM-2/GM-SRM-2 is about 20% larger than that of both DC-2 and SBDF-2 methods, while the critical time step of GM-3/GM-SRM-3 is about 30% and 50% larger than DC-3 and SBDF-3 methods, respectively. These percentage are computed using the critical time step of GM/GM-SRM as the reference. We conjecture that the difference gained in terms of τ_{crit} in GM/GM-SRM methods compared to others will be larger as the order of the methods increases but this requires further validation. The stability of GM and GM-SRM methods is exceptional for such high-order semi-implicit schemes thanks to the presence of **grad** – div terms. The so-called **grad** – div stabilization term improves certain stability property akin to the streamline-upwinding Petrov Galerkin or the SUPG methods to solve highly nonlinear flow which had been addressed in [35,36]. Interestingly, the critical time step for SBDF and DC methods are comparable. For instance, the critical time step of SBDF-2 method is slightly larger than that for DC-2 method while the critical time step of SBDF-3 method is slightly smaller than that for DC-3 method.

Our next task is to compare the lift and drag coefficients which are computed by all our methods to the values in the literature. To achieve this goal, we repeat similar computations of the flow past

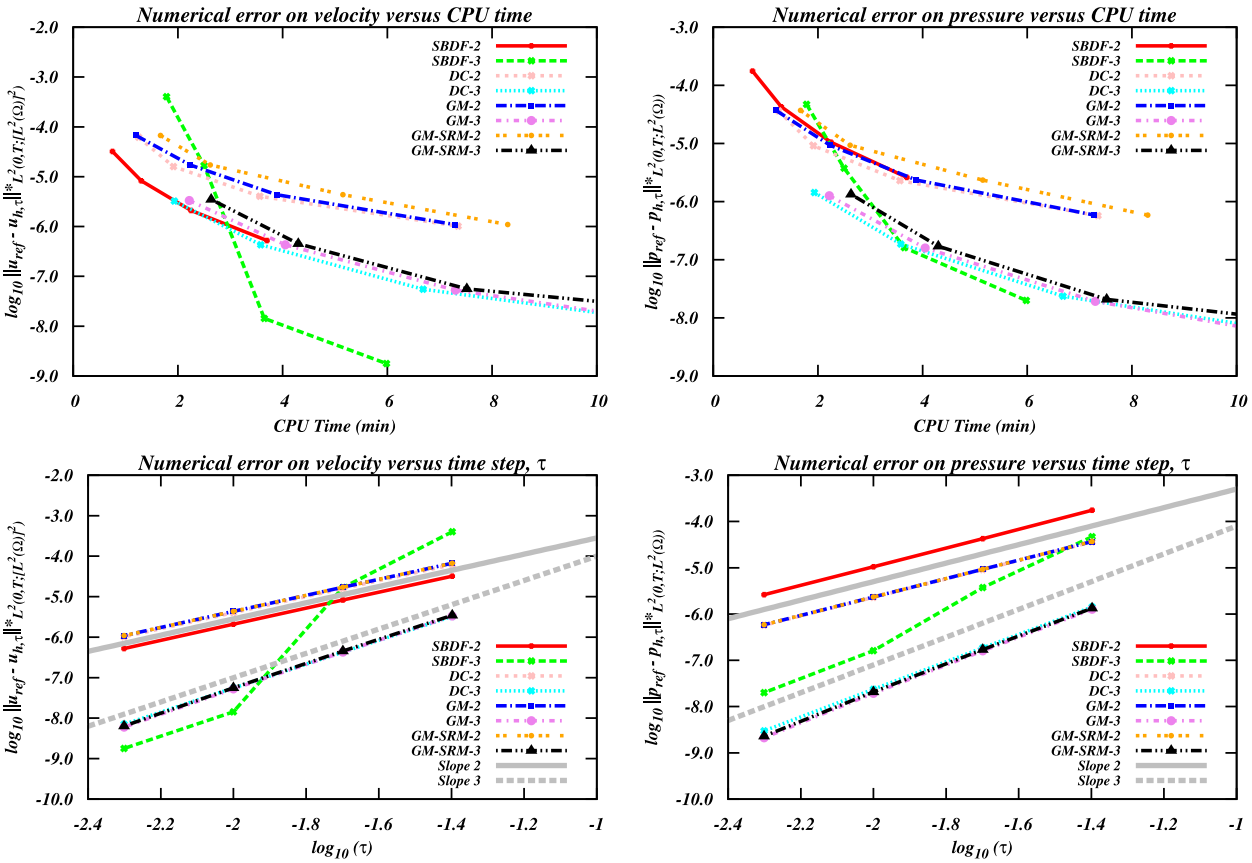

Fig. 4. Manufactured Solution II: Plot of the error on velocity (left) and pressure (right) as a function of CPU time (top) and time-step (bottom) for SBDF, DC, GM, GM-SRM time-stepping schemes.

the circular cylinder at $Re = 100$ but using a larger time interval, i.e., $t \in [0, 2000]$, to be sure to reach a fully periodic state. The time step, $\tau = 0.005$ is chosen for all methods. The lift and drag coefficients, c_l and c_d, respectively, are defined as follows:

$$c_d(t) = \frac{2}{\rho \mathbf{U}_\infty D}\int_S \left(\rho\nu \frac{\partial \mathbf{u}_{t_s}(t)}{\partial n} n_y - p(t) n_x\right) \mathrm{d}S,$$
$$c_l(t) = -\frac{2}{\rho \mathbf{U}_\infty D}\int_S \left(\rho\nu \frac{\partial \mathbf{u}_{t_s}(t)}{\partial n} n_x - p(t) n_y\right) \mathrm{d}S. \quad (38)$$

The parameter $\rho = 1$ is the density of the fluid, S the boundary of the cylinder, $n = (n_x, n_y)^T$ is the unit normal vector on S pointing inward Ω, $t_s = (n_y, -n_x)^T$ is the tangential vector and $\mathbf{u}_{t_s}$ is the tangential velocity. According to John [23], the computation of lift and drag using a volume integral formulation is more accurate and less sensitive to the mesh size around the cylinder than using the conventional line integral (Eq. 38). This method gives

$$c_l(t) = -20\int_\Omega [\mathbf{u}_t \cdot \mathbf{v}_l + \nu \nabla \mathbf{u}(t) : \nabla \mathbf{v}_l + (\mathbf{u}(t)\cdot\nabla)\mathbf{u}(t) \cdot \mathbf{v}_l - p(t)(\nabla \mathbf{v}_l)]\mathrm{d}x, \quad (39)$$

for $\mathbf{v}_l \in [H^1(\Omega)]^2$ solution of an auxillary Stokes equations with boundary conditions $\mathbf{v}_{1|S} = (0, 1)$ and $\mathbf{v}_{1|\Gamma} = (0, 0)$ on all other boundaries. Similarly,

$$c_d(t) = -20\int_\Omega [\mathbf{u}_t \cdot \mathbf{v}_d + \nu \nabla \mathbf{u}(t) : \nabla \mathbf{v}_d + (\mathbf{u}(t)\cdot\nabla)\mathbf{u}(t) \cdot \mathbf{v}_d - p(t)(\nabla \mathbf{v}_d)]\mathrm{d}x, \quad (40)$$

for $\mathbf{v}_d \in [H^1(\Omega)]^2$ solution of an auxillary Stokes equations with boundary conditions $\mathbf{v}_{d|S} = (1, 0)$ and $\mathbf{v}_{d|\Gamma} = (0, 0)$ on all other boundaries. Knowing a priori that the solution produces a single periodic mode, the mean, amplitude and frequency for both lift and drag coefficients are computed with a simple approximation using the time history of these parameters in the last 50 time units. For instance, the period of the lift and drag can be approximated using the successive difference of the times taken when the maximal slope occurs. These slopes are computed using linear interpolation of two consecutive points generated at each time step. For flow with a single periodic mode, we found that this method delivers more accurate results than the one with the conventional Fast Fourier Transform (FFT).

Table 2 shows the mean, amplitude and frequency of the lift and drag coefficients produced by our time-stepping methods. All methods produce a frequency for drag which is about twice the frequency for lift. This is in agreement with the literature on 2D laminar flows around the cylinder at $Re = 100$. The Strouhal number which is also the frequency of the lift coefficient in our present work is close to 0.17. Since the flow is still in laminar regime, the Strouhal number is less than 0.20—which is a value known for subcritical flows [4]. All methods produce a mean for lift coefficient close to zero, again as expected for such laminar flows (see, e.g., [23,44]).

We observe that the mean, amplitude and frequency for the drag and lift coefficients computed using our methods are very close to each other. These values differ only in the second deci-

Table 2
The mean, amplitude and frequency of the lift and drag coefficients.

Method	c_l mean	c_l amplitude	c_l frequency	c_d mean	c_d amplitude	c_d frequency
SBDF-2	-1.78129×10^{-5}	3.40513×10^{-1}	1.70068×10^{-1}	1.38565	9.79036×10^{-3}	3.40136×10^{-1}
SBDF-3	-1.76713×10^{-5}	3.40500×10^{-1}	1.70047×10^{-1}	1.38565	9.78998×10^{-3}	3.40099×10^{-1}
DC-2	-1.77619×10^{-5}	3.40506×10^{-1}	1.70049×10^{-1}	1.38564	9.79031×10^{-3}	3.40092×10^{-1}
DC-3	-9.92424×10^{-6}	3.45476×10^{-1}	1.69937×10^{-1}	1.39060	1.00610×10^{-2}	3.39936×10^{-1}
GM-2	-8.87375×10^{-5}	3.29150×10^{-1}	1.70182×10^{-1}	1.37285	9.06383×10^{-3}	3.39443×10^{-1}
GM-3	-7.41786×10^{-5}	3.29204×10^{-1}	1.70180×10^{-1}	1.37299	9.20195×10^{-3}	3.40362×10^{-1}
GM-SRM-2	-6.50141×10^{-5}	3.29261×10^{-1}	1.70182×10^{-1}	1.37304	9.20363×10^{-3}	3.40367×10^{-1}
GM-SRM-3	-6.50348×10^{-5}	3.29256×10^{-1}	1.70180×10^{-1}	1.37295	9.11191×10^{-3}	3.40362×10^{-1}

Table 3
Comparison between the present work with other numerical results and measurement from various references at $Re = 100$ (Laminar flow).

Source of results	Time-stepping method	*Str*	c_d mean	c_l amplitude
Present	SBDF, DC, GM, GM-SRM	0.1699 – 0.1701	1.3729 – 1.3906	0.3292 – 0.3455
Ranjani et al. (2009) [39]	2nd-order implicit	0.1569	1.3353	
Mittal & Raghuvanshi (2001) [33]	1st-order stabilized space-time	0.1680	1.4020	0.3550
Mittal & Tezduyar (1992) [34]	1st-order stabilized space-time	0.1670	≈ 1.3825	≈ 0.3500
Quarteroni et al. (1998) [38]	1st-order projection	0.1667	1.4486	0.3714
Simo & Armero (1994) [44]	1st-order implicit/semi-implicit	0.1670/0.1690	1.4500/1.4900	0.3500/0.3900
Simo & Armero (1994) [44]	2nd-order implicit/semi-implicit	0.1780	≈ 1.5375	≈ 0.3900
Baek & Karniadakis (2011) [6]	1st-/2nd-order implicit		1.4914/1.4929	0.2568/0.2588
Goldstein (1938) [15]	Experimentation	0.1680		

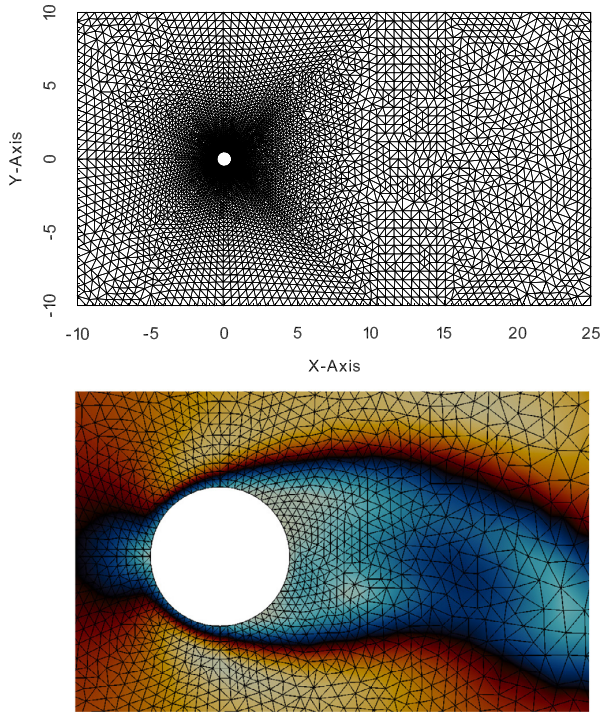

Fig. 5. The computational domain and mesh for flow past a circular cylinder (top). A blow-up of the mesh with boundary layer near the cylinder (yellow represents higher flow speed and blue represents lower flow speed) (bottom).

mal digit, e.g., the lift amplitude shows a maximal difference of 4.7% (DC-3 versus GM-3 methods) and the mean drag, a maximal difference of 1.2% (DC-3 versus GM-SRM-3 methods). We observe that the most significant discrepancies occur for methods with **grad** – div term (GM/GM-SRM) versus methods without **grad** – div term (SBDF/DC). Still, we cannot assert which solver produces the most accurate result.

Table 4
The critical time step, CFL bound, CPU time for computing one time step for each method when computing the lid-driven cavity flow at $Re = 8500$.

Method	Critical time step, τ_{crit}	CFL_{max}	CPU time/time step
SBDF-2	1.45508×10^{-3}	2.28314×10^{-1}	0.9802
SBDF-3	1.57227×10^{-3}	2.23712×10^{-1}	1.3086
DC-2	1.75781×10^{-3}	2.23343×10^{-1}	1.9948
DC-3	1.59180×10^{-3}	1.95099×10^{-1}	3.6222
GM-2	2.58789×10^{-3}	3.17185×10^{-1}	2.4570
GM-3	2.37305×10^{-3}	2.90853×10^{-1}	4.1908
GM-SRM-2	2.59766×10^{-3}	3.18382×10^{-1}	2.8796
GM-SRM-3	2.37305×10^{-3}	2.90853×10^{-1}	4.8534

We also make comparison with lift and drag coefficients obtained from the literature, both for numerical and experimental values. From Table 3, one can see that our computed Strouhal numbers = 0.17 is in a good agreement with published values except the one from Ranjani et. al [39] ($St = 0.1569$). For mean drag, our result differs by 0.8% – 10.7% compared to values in the literature. Meanwhile, the differences on lift amplitude with published values are within 1.3%–15.6%. We cannot find sufficient data on the drag amplitude to provide comparison. Notice that none of the published values are computed by time-stepping schemes with 3rd- or higher-order accuracy. There are also differences in space approximations (e.g., finite volume, difference etc). These discrepancies may result also from the different methods used to compute the lift and drag (e.g., integration along the circular boundary [49] versus integration on volume [23]). The FFT analysis which is used by many may not be able to provide very accurate spectrum for the lift and drag coefficients.

4.4. Lid-driven cavity flow

The lid-driven cavity is one of the most documented test cases in computational fluid dynamics. Cavity flows occur in many applications in modern industry. To name a few, lid-driven cavity can be used to study the efficiency of fluid mixing, stirring processes and fluid cooling mechanisms, where all of these take place either in an enclosed or semi-enclosed region. Although the experiment setting of lid-driven cavity is simple, the study of the resulting flow patterns and generated vortices, especially near corners,

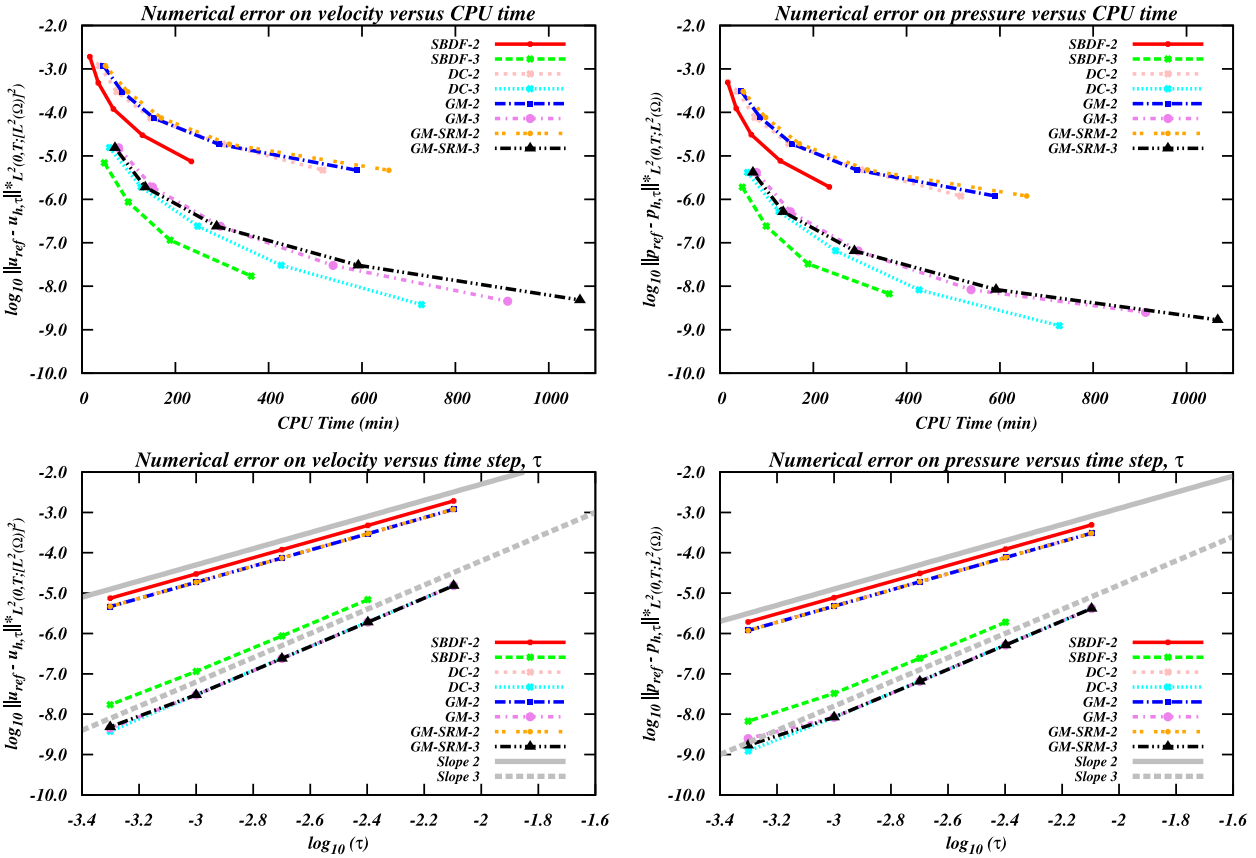

Fig. 6. Flow around the circular cylinder: Plot of the error on velocity (left) and pressure (right) as a function of CPU time (top) and time-step (bottom) for SBDF, DC, GM, GM-SRM time-stepping schemes.

is very challenging. In dimensionless settings, the characterisation of flow patterns, the number of vortices and the stability of the flow in a square cavity is determined only by the Reynolds number. For Reynolds numbers, $Re < 8000$, the lid-driven cavity flow is known to be steady. However, for $Re \geq 8000$, the flow becomes unsteady, following a Hopf bifurcation leading to a periodic solution in both velocity and pressure. The critical Reynolds number, $Re_{1,\,\mathrm{cri}}$, where the first Hopf bifurcation takes place is given under 7500 by [18] and is reported very close to 8000 in more recent publications [2,8,13,47].

In this paper, we investigate the performance of our methods using the lid-driven cavity flow at $Re = 8500$ [2,37]. This Reynolds number is chosen for two reasons: to make sure that the flow is unsteady since $Re > Re_{1,\,\mathrm{cri}}$ with a sufficiently large gap, and simulations with $Re = 8500$ are known to be challenging. For instance, excessively long computational time is required beginning from a 'cold' start to go over a long transient and finally reach a fully periodic flow with a frequency around 0.44. Meanwhile, $Re = 8500$ is in the vicinity of a second Hopf bifurcation, which occurs around $8700 < Re_{2,\,\mathrm{cri}} < 10000$. This second Hopf bifurcation introduces a second periodic mode with a frequency around 0.61 [8,47].

The numerical comparison is done both qualitatively and quantitatively. For qualitative analysis, we compare the formation of vortices using stream function plots with published results. While for quantitative analysis, we first establish a set of numerical results in terms of the frequency, amplitude and mean value of the x-velocity, u, at several monitoring points. Then the results computed by our methods are compared among themselves. Finally, we compare the frequency of the pulsating flow or the Strouhal number with similar values found in the literature.

We consider a unit square domain $\Omega = (0,1) \times (0,1)$ discretized using a non-uniform triangular mesh with 120 triangles along each boundary edge. This produces a mesh with element size h_K within the range [0.00628, 0.01660], with 34164 vertices, 17323 triangles and 154941 degrees of freedom with $\mathbb{P}_2 - \mathbb{P}_1$ mixed finite elements. A constant velocity $\mathbf{u} = (1,0)$ is imposed on the top boundary (the lid), while for the remaining boundaries, we set $\mathbf{u} = (0,0)$. We prescribe the state of rest for velocity and pressure at $t = 0$. For this test case, we do not pursue an efficiency analysis similar to those carried out in above sections. Equally useful, the CPU time per time step (found as an average over 100 time steps) for each of the methods is given in Table 4. We choose a smaller time step, $\tau = 0.001$ for all 3rd-order methods to produce very accurate solution in time, which will serve as 'reference' solutions for each method. In order to assess the quality of the 2nd-order methods, the following time step is set for each method: $\tau = 0.001$ in both SBDF-2 and DC-2 method and $\tau = 0.025$ in both GM-2 and GM-SRM-2 methods. We also include SBDF-2 method with a larger time step $\tau = 0.014$ which we call as SBDF-2*. For stabilization parameter, we fix $\lambda = 1$ in both GM-2 and GM-3 while prescribing $\alpha_1, \alpha_2 = 1$ in both GM-SRM-2 and GM-SRM-3.

Table 4 summarizes the critical time step of each method obtained using the bisection method and maximum CFL bound computed using Eq. (37). In terms of numerical stability, the maxi-

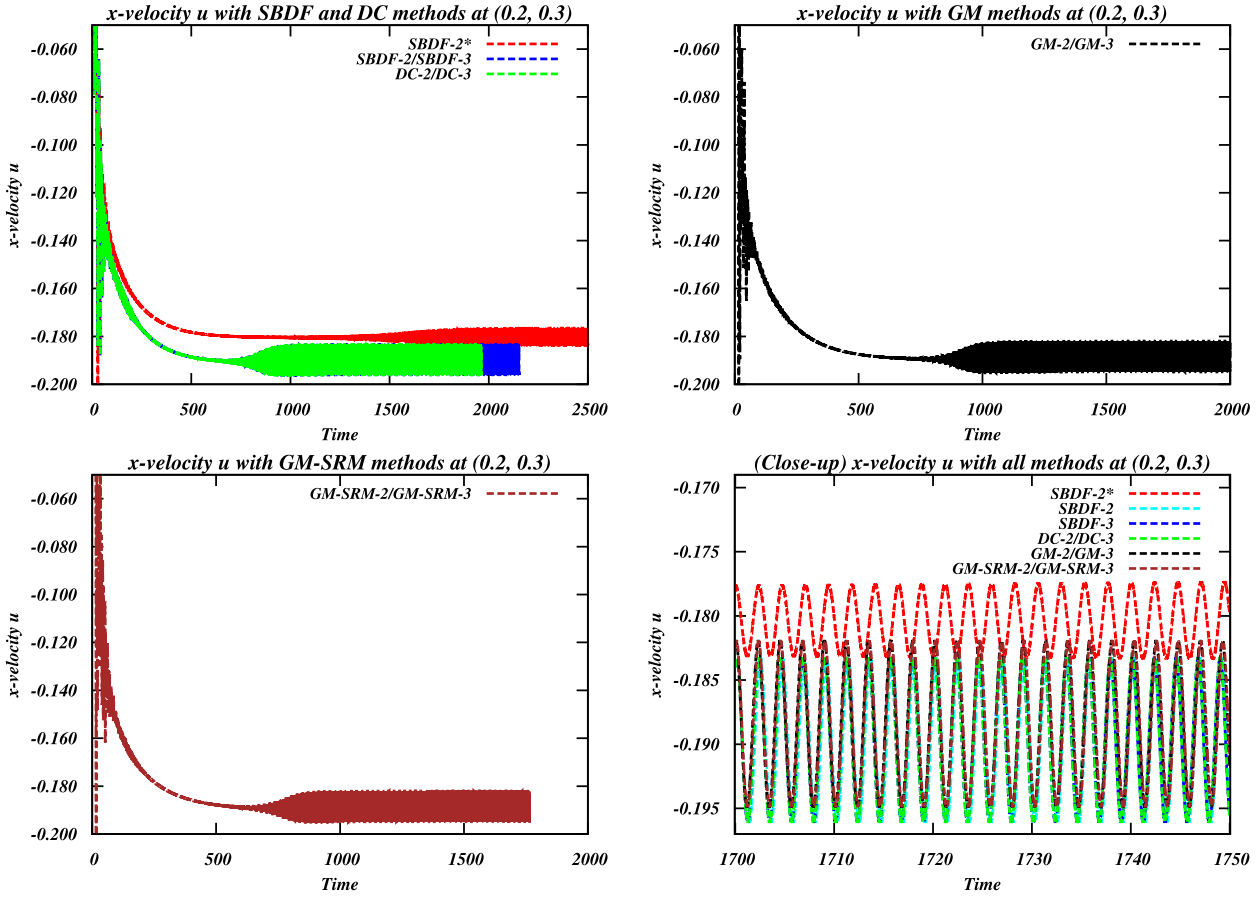

Fig. 7. The x-velocity, u near bottom left corner (0.2, 0.3) for SBDF, SBDF-2* and DC methods (top left graph), GM methods (top right graph), and GM-SRM methods (bottom left graph). A close-up view of x-velocity for all time-stepping methods for $t \in [1700, 1750]$ (bottom right graph).

mum CFL number of each method decreases slightly from the 2nd- to the 3rd-order accuracy. Interestingly, the critical time-step for SBDF-3 is larger than that of SBDF-2 while the opposite is observed when comparing their CFL bounds. We expect that the critical time step has a similar trend to the CFL bound. This anomaly is explained in the following manner. The CFL bound of SBDF-2 methods must be larger than that of SBDF-3 methods as the stability region of a higher order methods becomes smaller. The maximum CFL should be attained within a same local element regardless of the methods used, provided the flow computed is sufficiently accurate. By fixing the time steps near their critical values, the SBDF-2 method produces a larger error than SBDF-3 methods, causing the SBDF-2 method to attain its CFL bound in a different element than with the SBDF-3 method. We observe that both GM and GM-SRM methods have the largest maximum CFL bound (about 0.30) which are followed by SBDF and DC methods (about 0.22). Hence, GM/GM-SRM methods are the most stable for this lid-driven cavity flow.

The x-velocity u is sampled at three different points near the corners within the domain, i.e., bottom left at (0.2, 0.3), bottom right at (0.8, 0.3) and top right at (0.8, 0.7).

Fig. 7 shows the x-velocity u sampled over time at (0.2, 0.3) near the bottom left corner for all proposed time-stepping methods. To narrate the behaviour of the flow, we use the time-history of x-velocity u computed by DC-3 (in green, top left in Fig. 7). The x-velocity u shows three distinct transient phases. The first phase is the fast transient solution starting immediately from the state of rest that occurs for $t \in [0, 250]$. During this phase, the constant flow along the lid induces a large circulation which quickly transforms into a major vortex at the center of the cavity. Since the advective term is more predominant than diffusion for $Re = 8500$, the interaction of the center vortex and the walls generates several counter-rotating vortices at the four corners of the domain. The flow is gradually stabilized with well-developed secondary vortices appearing at three corners, i.e., two at bottom right, one at bottom left and another one at top left. Near the end of the first phase, there are several smaller blinking vortices that can be observed, two are formed just below the top left corner and one at the bottom left corner. The total number of vortices generated at this stage varies between 5 and 8. The second phase of the flow takes place for $t \in [250, 500]$ where the x-velocity further decreases. During this phase, the blinking vortices at the bottom left corner can still be observed while the two at the top left corner diminish. The number of vortices generated varies between 5 and 6. During the third phase of flow, the solution experiences a gradual growth of a periodic mode that can be easily observed beginning at $t = 600$. This is because the flow has lost its stability and the mode corresponding to the first Hopf bifurcation is taking over. For t around 1100, the x-velocity becomes fully periodic (single mode).

From Fig. 7, we also observe that the total time of the transient solution depends on the time-stepping methods, the order of accuracy and time step used. SBDF-2, SBDF-3, DC-2 and DC-3

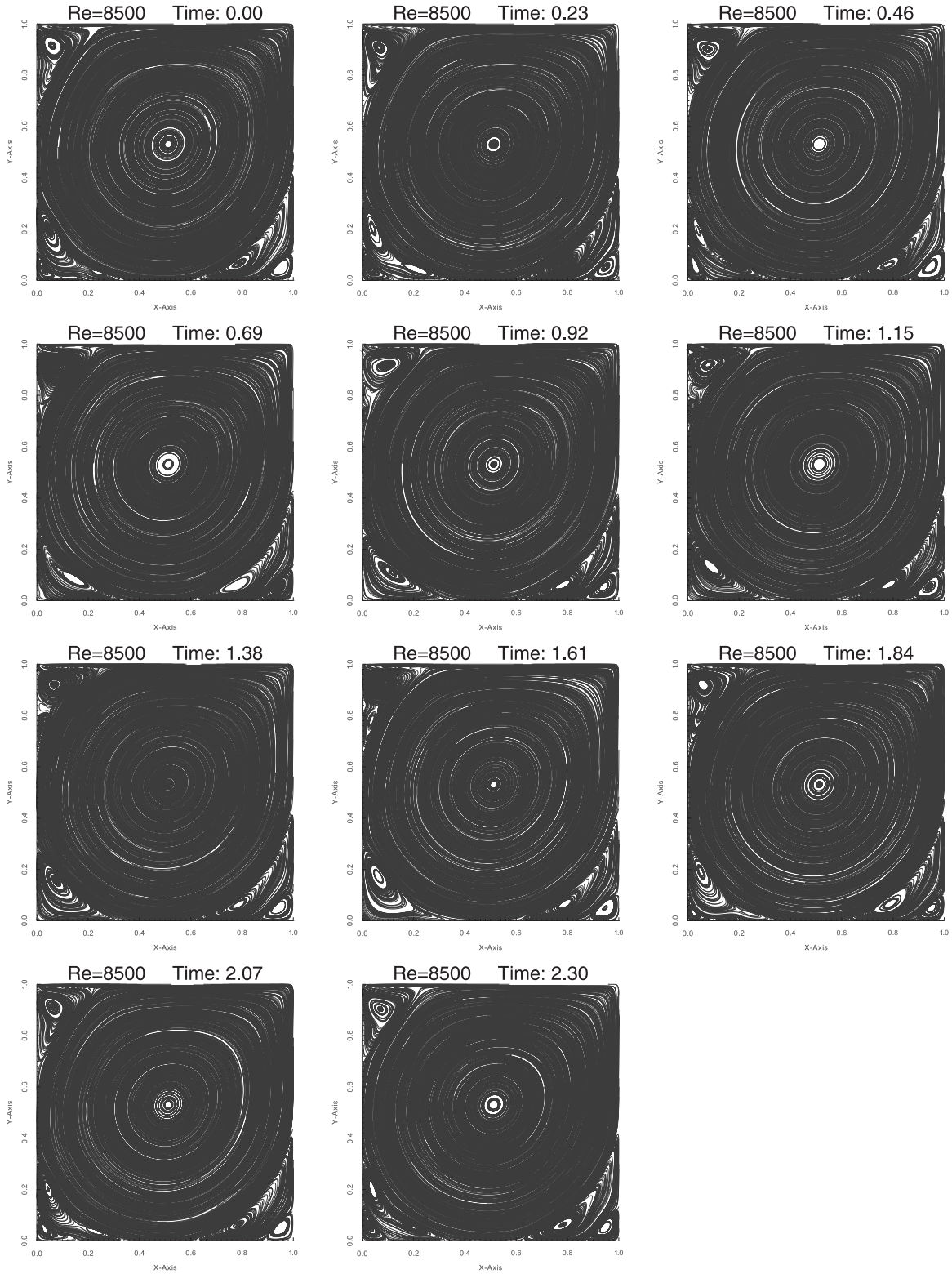

Fig. 8. Snapshot of streamlines for 2D lid-driven cavity flow ($Re = 8500$) by increment of 0.23 time units over 2.30 time unit, a time interval slightly larger than one period of the flow, $T_f \approx 2.22$.

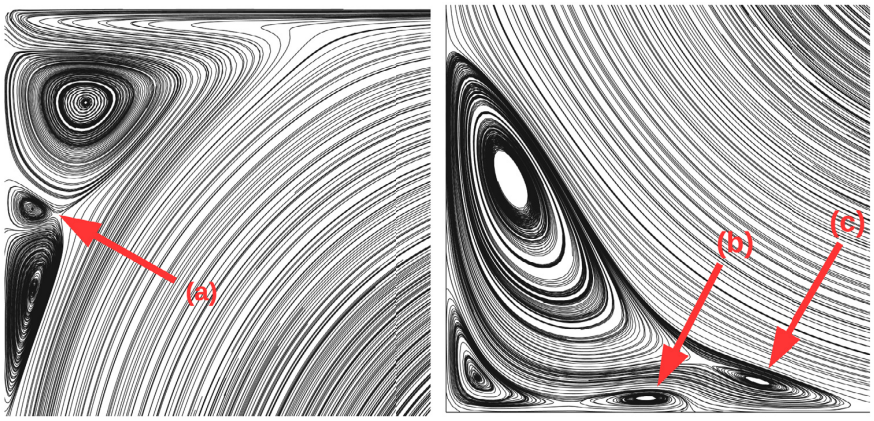

Fig. 9. Lid-Driven Cavity: Close-up view of vortices near the top left corner at $t = 1.38$ (left) and bottom left corner at $t = 1.84$ (right). The arrows labelled as (a), (b) and (c) indicate the vortices that were not captured by previous published results.

methods reach a full periodic solution beginning at $t = 1100$; GM methods around $t = 1150$; and GM-SRM methods around $t = 1000$ which have the shortest transient duration. SBDF-2* attains its full periodic solution beginning at $t = 2200$. In addition to the accuracy gained from the defect correction strategy, the sequential regularization method improves stability in connection with the incompressibility condition, hence reduces the time for transient solution of GM-SRM compared to other methods. In general, methods that produce shorter transient solutions help to reduce the total CPU time to reach time-periodic flows. We also plotted the x-velocity u, computed with all methods at point (0.2, 0.3) for $t \in [1700, 1750]$. The phase for each u is shifted for better illustration. The graph indicates that SBDF-2 methods may not produce a desirable results if the time step is chosen close to its critical value.

Fig. 8 illustrates the evolution of the time-periodic flow over one cycle with snapshots of the streamlines every 0.23 time unit (for a period $T_f = 2.22$). During a full cycle, we observe one to two smaller pulsating vortices appearing below the secondary vortex near top left corner. Near the bottom left corner a more complex vortex formation—with as many as 4 pulsating vortices are observed, e.g., at $t = 1.84$. Compared to previous published results, our numerical solution captures two extra smaller vortices as shown in Fig. 9 (at right). The maximal number vortices for a complete cycle is 10. Using our current computational settings, similar streamlines can be reproduced with all methods except SBDF-2 with $\tau = 0.0014$ (SBDF-2*). To some extend, our results contradict with few existing results. For instance, Pan and Glowinski [37] produced only 2 vortices near the bottom left corner, while other appearances of vortex are identical to our results at $Re = 8500$. They used $\mathbb{P}_1$-iso-$\mathbb{P}_2$ finite elements and Chorin's projection method for space and time discretizations, respectively on a mesh of size 256 $\times$ 256 mesh. Kufferman [24] produced results similar to Pan and Glowinski but using 2nd-order center-difference schemes for space and Crank-Nicolson for time stepping on 128 $\times$ 128 mesh. Our results were produced with higher-order methods, both in space and time, which explain the extra flow features observed. The quantitative results on x-velocity u at various locations in terms of mean, amplitude, period and frequency are summarized in Table 5. It is observed that the accuracy of SBDF-2 is compromised when using a time step near its critical value. On the other hand, SBDF-2 method produces satisfactory result with $\tau = 0.001$, which can be accurate up to 4th decimal point compared to the results computed with 3rd-order SBDF and DC methods. The numerical results computed by DC-2 method are in agreement with the one computed with DC-3 and SBDF-3 using the same time step, $\tau = 0.001$. Both 2nd-order GM and GM-SRM methods with time steps near the critical value produce very close result to that produced by their 3rd-order counterparts with time step, $\tau = 0.001$ (e.g., maximum difference starts at the 3rd decimal point). This suggests that the defect correction algorithm produces an unmatched accuracy even with 2nd-order method at larger time step.

Fig. 10 shows the phase portraits of x- and y-velocity at bottom left (0.2, 0.3), which is a plot of $u(t)$ versus $u(t + \frac{T_f}{8})$ for $t \in [1700, 1750]$, where T_f is the respective period of the x-velocity produced by each of the time-stepping methods. There are three distinct phase plots that can be observed in these diagrams. The first one is obtained with SBDF-2* (Phase Portrait I, PPI) shown as the red ellipse, second one with DC and SBDF methods (PPII) shown as the overlapping pink ellipse and the third one with GM and GM-SRM methods (PPIII) shown as the black ellipse. In this numerical experiment, the phase plot with 2nd-order methods are found to be indistinguishable from their respective 3rd-order methods. Again, SBDF-2* produce undesirable results since the time step is taken too large. Methods with defect correction strategy allows one to choose larger time step while still obtaining satisfactory results.

The 'offset' between PPII and PPIII at three monitoring points can only be explained by the presence of **grad** − div term in both GM and GM-SRM methods. GM-SRM methods can provide very accurate solutions for flow problems involving Dirichlet boundary condition as for the Taylor's vortex flow presented above. Boundary conditions alone may not suffice to explain the difference between PPII and PPIII. Therefore, the question concerning which of PPII and PPIII is more accurate requires further investigations.

Table 6 compares the frequency of the periodic solution that we obtain with results found in the literature. The frequency of all our methods stand between 0.4472 and 0.4501, which is in the range published by Pan and Glowinski [37], i.e., 0.4405 − 0.4505. Meanwhile, the frequency computed by GM and GM-SRM (i.e., 0.4472 and 0.4473, respectively) are very close to 0.4470, the value found by Auteri et al. [2]. The frequency value by Kufferman [24], might be a little underestimated.

5. Conclusions

We have conducted a thorough numerical assessment and benchmark of several 2nd- and 3rd-order semi-implicit methods, e.g., SBDF, DC, GM and GM-SRM methods using two manufactured solutions and two well-known test cases. So far, these methods are proven to be very robust to compute unsteady laminar flows. With a direct solver, SBDF methods are the most efficient method to compute 2D unsteady flows. However, SBDF methods may depend on proper initialization to produce such remarkable results.

Table 5
The computed average, amplitude, frequency of the x-velocity u of the periodic flow at points near the three corners: bottom left (0.2, 0.3), bottom right (0.8, 0.3) and top right (0.8, 0.7).

Bottom left (0.2, 0.3)					
Method	Time step, τ	Mean value	Amplitude	Period (s)	Frequency(Hz)
SBDF-2*	1.4×10^{-3}	-1.80169×10^{-1}	3.63427×10^{-3}	2.35392	4.24823×10^{-1}
SBDF-2	1.0×10^{-3}	-1.89676×10^{-1}	6.36170×10^{-3}	2.22166	4.50113×10^{-1}
SBDF-3	1.0×10^{-3}	-1.89676×10^{-1}	6.36155×10^{-3}	2.22167	4.50111×10^{-1}
DC-2	1.0×10^{-3}	-1.89677×10^{-1}	6.36165×10^{-3}	2.22167	4.50113×10^{-1}
DC-3	1.0×10^{-3}	-1.89676×10^{-1}	6.36157×10^{-3}	2.22168	4.50111×10^{-1}
GM-2	2.5×10^{-3}	-1.88397×10^{-1}	6.44025×10^{-3}	2.23585	4.47258×10^{-1}
GM-3	1.0×10^{-3}	-1.88397×10^{-1}	6.43753×10^{-3}	2.23607	4.47212×10^{-1}
GM-SRM-2	2.5×10^{-3}	-1.88406×10^{-1}	6.41414×10^{-3}	2.23569	4.47289×10^{-1}
GM-SRM-3	1.0×10^{-3}	-1.88404×10^{-1}	6.41487×10^{-3}	2.23615	4.47198×10^{-1}
Bottom right (0.8, 0.3)					
Method	Time step, τ	Mean value	Amplitude	Period (s)	Frequency(Hz)
SBDF-2*	1.4×10^{-3}	-2.20319×10^{-1}	3.70781×10^{-4}	2.35390	4.24826×10^{-1}
SBDF-2	1.0×10^{-3}	-2.32445×10^{-1}	5.45568×10^{-4}	2.22166	4.50113×10^{-1}
SBDF-3	1.0×10^{-3}	-2.32445×10^{-1}	5.45596×10^{-4}	2.22167	4.50111×10^{-1}
DC-2	1.0×10^{-3}	-2.32445×10^{-1}	5.45599×10^{-4}	2.22165	4.50114×10^{-1}
DC-3	1.0×10^{-3}	-2.32445×10^{-1}	5.45590×10^{-4}	2.22168	4.50111×10^{-1}
GM-2	2.5×10^{-3}	-2.30863×10^{-1}	4.97679×10^{-4}	2.23585	4.47258×10^{-1}
GM-3	1.0×10^{-3}	-2.30866×10^{-1}	4.98207×10^{-4}	2.23607	4.47214×10^{-1}
GM-SRM-2	2.5×10^{-3}	-2.30875×10^{-1}	4.97103×10^{-4}	2.23568	4.47292×10^{-1}
GM-SRM-3	1.0×10^{-3}	-2.30878×10^{-1}	4.99226×10^{-4}	2.23612	4.47204×10^{-1}
Top right (0.8, 0.7)					
Method	Time step, τ	Mean value	Amplitude	Period (s)	Frequency(Hz)
SBDF-2*	1.4×10^{-3}	8.23003×10^{-2}	1.09002×10^{-3}	2.35394	4.24820×10^{-1}
SBDF-2	1.0×10^{-3}	8.72107×10^{-2}	2.02720×10^{-3}	2.22166	4.50113×10^{-1}
SBDF-3	1.0×10^{-3}	8.72107×10^{-2}	2.02726×10^{-3}	2.22168	4.50111×10^{-1}
DC-2	1.0×10^{-3}	8.72110×10^{-2}	2.02722×10^{-3}	2.22167	4.50112×10^{-1}
DC-3	1.0×10^{-3}	8.72105×10^{-2}	2.02727×10^{-3}	2.22168	4.50111×10^{-1}
GM-2	2.5×10^{-3}	8.64370×10^{-2}	2.03537×10^{-3}	2.23585	4.47258×10^{-1}
GM-3	1.0×10^{-3}	8.64340×10^{-2}	2.03690×10^{-3}	2.23607	4.47213×10^{-1}
GM-SRM-2	2.5×10^{-3}	8.64412×10^{-2}	2.03213×10^{-3}	2.23569	4.47289×10^{-1}
GM-SRM-3	1.0×10^{-3}	8.64336×10^{-2}	2.03730×10^{-3}	2.23614	4.47200×10^{-1}

Table 6
Comparison between our work and numerical values from the literature for frequencies of the lid-driven cavity flow at $Re = 8500$.

Source of results	Time-stepping schemes	Frequency
Present	SBDF, DC, GM, GM-SRM	0.4501, 0.4501, 0.4472, 0.4473
Auteri et. al (2002) [2]	2nd-order projection	0.4470
Kupferman (2001) [24]	2nd-order CN-midpoint rule	0.4000
Pan & Glowinski (2000) [38]	2nd-order projection	0.4405 – 0.4505

On the other hand, DC, GM and GM-SRM methods are self-starting and produce the least error in time for a fixed time step since they are based on the same defect correction method. DC methods require the solution of k similar saddle point problem to obtain kth-order of accuracy, which turns out to be less efficient at each time step. GM and GM-SRM methods require smaller memory and CPU time for each linear solve. However, more of these linear systems have to be solved at each time step, i.e., one from the momentum and one from the continuity equation, and this k times to reach kth-order of accuracy. GM and GM-SRM methods have good potential to compute unsteady flows involving millions of unknowns (e.g., 3D flows) since the linear systems with symmetric positive definite matrices can be easily handled by any efficient iterative solvers such as the conjugate gradient method.

In terms of stability, GM and GM-SRM methods are more stable than SBDF and DC methods due to the presence of a **grad** – div stabilization term. The combination of **grad** – div term and defect correction strategy allows us to choose larger time step with GM and GM-SRM methods while still producing good results. The **grad** – div stabilization term in GM and GM-SRM helps to enforce incompressibility condition with $\mathbb{P}_2 - \mathbb{P}_1$ elements, thus reducing even more the error in space. This property has been demonstrated by GM-SRM methods with a nearly 'optimal' choice of the stabilization parameters when computing the manufactured solution II. This was attempted with GM methods with much smaller λ (e.g. $\lambda < 3.3$) but it resulted in more oscillations at start-up before they finally damp at larger t to produce similar accuracy to that with GM-SRM methods (result not shown here). We mention that the optimal choice of the stabilization parameters, λ in GM methods and α_1, α_2 in GM-SRM methods is difficult to obtain to produce the least error in space on a fixed mesh. The analysis of these stabilization terms in terms of the errors induced on velocity and pressure has not been done yet for unsteady flows. The optimal choice of these stabilization parameters λ and α for Navier–Stokes equations is still an open problem.

In the last two test cases, we observed that the use of higher-order semi-implicit methods with reasonably small time step is crucial for highly nonlinear flows (large Reynolds number). Besides reproducing results from the literature, e.g., the lift and drag coefficients along the cylinder and pulsating frequency, Strouhal number in both flows, we managed to capture additional vortices in the lid-driven cavity at $Re = 8500$ with all our methods. To the best of

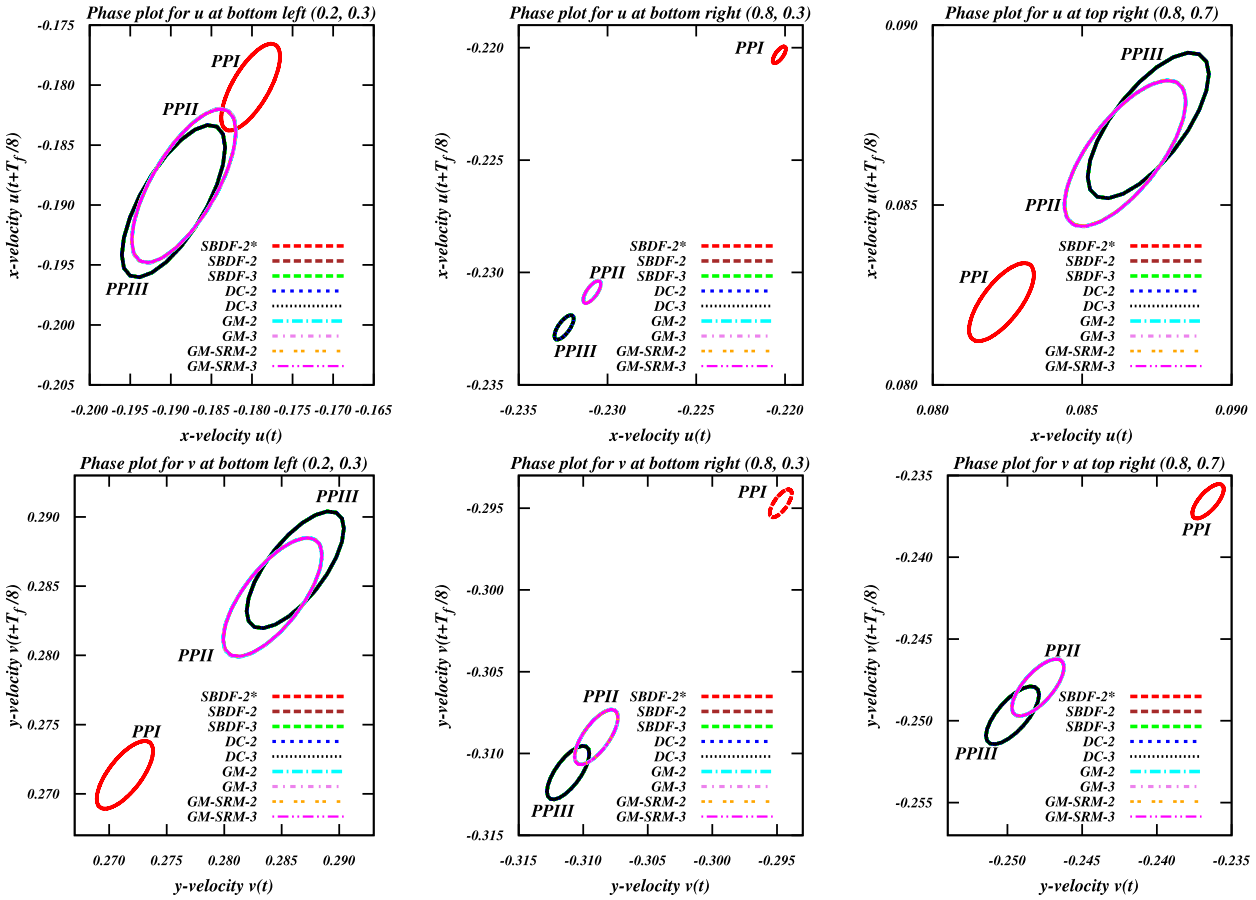

Fig. 10. Phase plots for x- and y-velocity, u and v respectively, monitored at three different locations with SBDF, DC, GM and GM-SRM methods (2nd- and 3rd-order).

our knowledge, this finding has not been reported in the recent literature. This indicates that higher-order time-stepping methods with smaller time step are indispendable to obtain very accurate solutions for inertia-dominant flows.

Acknowledgement

We would like to express our gratitute for funding given by the NSERC Discovery Grant program to conduct this research. The first author, Kak Choon, Loy is a recipient of a Ph.D scholarship granted by the Universiti Malaysia Terengganu and the Ministry of Education of Malaysia, for this we deeply acknowledge. Thanks to the FreeFEM++ development team headed by Professor Frédéric Hecht at the Laboratoire Jacques–Louis Lions, Université Pierre et Marie Curie. Last but not least, we are indebted to Peter Minev and Jean-Luc Guermond for their willingness to share their submitted manuscript, long before their work is finally published online in November 12, 2015 [20].

References

[1] Akrivis G, Crouzeix M, Makridakis C. Implicit-explicit multistep finite element methods for nonlinear parabolic problems. Math Comput 1998;67(222):457–77.
[2] Auteri F, Parolini N, Quartapelle L. Numerical investigation on the stability of singular driven cavity flow. J Comput Phys 2002;183:1–25.
[3] Ascher UM, Ruuth SJ, Wetton BTR. Implicit-explicit methods for time-dependent partial differential equations. SIAM J Numer Anal 1995;32(3):797–823.
[4] Blevins RD. Flow-induced vibration. NY, USA: Van Nostrand Reinhold Co., Inc. New York; 1990. p. 377.
[5] Boffi D, Brezzi F, Fortin M. Mixed finite element methods and applications. Berlin, Heidelberg: Springer Verlag; 2013. p. 691.
[6] Baek H, Karniadakis GE. Sub-iteration leads to accuracy and stability enhancements of semi-implicit schemes for the Navier–Stokes equations. J Comput Phys 2011;230:4384–402.
[7] Baker GA, Dougalis VA, Karakashian OA. On a higher-order accurate fully discrete Galerkin approximation to the Navier–Stokes equations. Math Comput 1982;39(160):339–75.
[8] Bruneau CH, Saad M. The 2d lid-driven cavity problem revisited. Comput Fluids 2005;35:326–48.
[9] Chorin AJ. A numerical method for solving incompressible viscous flow problems. J Comput Phys 1967;2(1):12–26.
[10] Dai X, Sun J, Cheng XL. Error estimates for an operator-splitting method for Navier–Stokes equations: Second-order schemes. J Comput Appl Math 2009;231:696–704.
[11] Ethier M, Bourgault Y. Semi-implicit time-discretization schemes for the bidomain model. SIAM J Numer Anal 2008;46(5):2443–68.
[12] Elman H, Silvester D, Wathen A. Finite elements and fast iterative solvers with applications in incompressible fluid dynamics. New York: Oxford University Press; 2008. p. 400.
[13] Fortin A., Jardak M., Gervais J.J., Pierre R.. 1997. Localization of Hopf bifurcations in fluid flow problems. 24. pp. 1185–1210.
[14] Fortin M, Glowinski R. Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems. Elsevier; 2000. p. 340.
[15] Goldstein S.. 1938. Modern development in fluid dynamics. 35. pp. 421–447.
[16] Girault V, Raviart PA. Finite element methods for Navier–Stokes equations: theory and algorithms. Heidelberg: Springer-Verlag; 2011. p. 376.
[17] Glowinski R, Pironneau O. Finite element methods for Navier–Stokes equations. Ann Rev Fluid Mech 1992;24:167–204.
[18] Ghia U, Ghia KN, Shin CT. High-re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. J Comput Phys 1982;48:387–411.
[19] Gresho PM, Griffiths DF, Silvester DJ. Adaptive time-stepping for incompressible flow part i: Scalar advection-diffusion. SIAM J Sci Comput 2008;30(4):2018–54.

[20] Guermond JL, Minev PD. High-order time stepping for the incompressible Navier–Stokes equations. SIAM J Sci Comput 2015;37(6):A2656–81.
[21] Guermond JL, Minev P, Jie S. An overview of projection methods for incompressible flow. Comput Methods Appl Mech Eng 2006;195:6011–45.
[22] Hecht F.. FreeFEM++: Third edition, version 3.32. 2014. http://www.freefem.org/ff++/ftp/freefem++doc.pdf. pp. 398.
[23] John V. Reference values for drag and lift of a two-dimensional time-dependent flow around a cylinder. Int J Numer Methods Fluids 2004;44:777–88.
[24] Kupferman R. A central-difference scheme for a pure stream function formulation of incompressible viscous flow. SIAM J Sci Comput 2001;23(1):1–18.
[25] Kloucek P, Rys FS. Stability of the fractional step θ-scheme for the nonstationary Navier–Stokes equations. SIAM J Numer Anal 1994;31(5):1312–35.
[26] Kress W, Gustafsson B. Deferred correction methods for initial boundary value problems. J Sci Comput 2001;17(1-4):241–51.
[27] Kress W, Lötstedt P. Time-step restrictions using semi-implicit methods for the incompressible Navier–Stokes equations. Comput Methods Appl Mech Eng 2006;195:4433–47.
[28] Kim J, Moin P. Application of a fractional step method to incompressible Navier–Stokes equations. J Comput Phys 1985;59:308–23.
[29] Labovschii A. A defect correction method for the time-dependent Navier–Stokes equations. Numer Methods Partial Differ Equ 2009;25(1):1–25.
[30] Lin P. A sequential regularization method for time-dependent incompressible Navier–Stokes equations. SIAM J Numer Anal 1997;34(3):1051–71.
[31] Labovsky A, Layton WJ, Manica CC, Neda M, Rebholz LG. The stabilized extrapolated trapezoidal finite-element methods for Navier–Stokes equations. Comput Methods Appl Mech Eng 2009;198:958–74.
[32] Lu XL, Lin P, Liu JG. Analysis of a sequential regularization method for the unsteady Navier–Stokes equations. Math Comput 2008;77(263):1467–94.
[33] Mittal S, Raghuvanshi A. Control of vortex shedding behind circular cylinder for flows at low Reynold numbers. Internal J Numer Methods Fluids 2001;35:421–47.
[34] Mittal S., Tezduyar T.E.. 1992. Notes on the Stabilized Space–Time Finite–Element Formulation of Unsteady Incompressible Flows. 73. pp. 93–112.
[35] Olshanskii MA, Lube G, Heister T, Löwe J. Grad-div stabilization and subgrid pressure models for the incompressible Navier–Stokes equations. Comput Methods Appl Mech Eng 2009;198:3975–88.
[36] Olshanskii MA, Reusken A. Grad-div stabilization for Stokes equations. Math Comput 2003;73(248):1699–718.
[37] Pan TW, Glowinski R. A projection/wave-like equation method for the numerical simulation of incompressible viscous fluid flow modeled by the Navier–Stokes equations. Comput Fluid Dyn J 2000;9:28–42.
[38] Quarteroni A, Saleri F, Veneziani A. Factorization methods for the numerical approximation of Navier–Stokes equations. Comput Methods Appl Mech Eng 1998;188:505–26.
[39] Rajani BN, Kandasamy A, Majumdar S. Numerical simulation of laminar flow past a circular cylinder. Appl Math Modell 2009;33:1228–47.
[40] Rangan A. Deferred correction methods for low index differential algebraic equations. In: BIT numerical mathematics, 43. Kluwer Academic Publishers; 2003. p. 1–18.
[41] Ruuth SJ. Implicit-explicit methods for reaction-diffusion problems in pattern formation. J Math Biol 1995;34:148–76.
[42] Schäfer M, Turek S, Durst F, Krause E, Rannacher R. Benchmark computations of laminar flow around a cylinder. Vieweg+ Teubner Verlag; 1996. p. 547–66.
[43] Shankar PN, Deshpande MD. Fluid mechanics in the driven cavity. Ann Rev Fluid Mech 2000;32:93–136.
[44] Simo JC, Armero F. Unconditional stability and long-term behaviour of transient algorithms for the incompressible Navier–Stokes and Euler equations. Comput Methods Appl Mech Eng 1993;111:111–54.
[45] Stetter HJ. The defect corection principle and discretization methods. In: Numerische Mathematik, 29. Springer; 1978. p. 425–43.
[46] Temam R. Navier–Stokes equations: theory and numerical analysis. Providence, Rhode Island: AMS Chelsea Publishing; 2001. p. 407.
[47] Tiesinga G, Wubs FB, Veldman AEP. Bifurcation analysis of incompressible flow in a driven cavity by the Newton-Picard method. J Comput Appl Math 2001;140:751–72.
[48] Turek S. A comparative study of time-stepping techniques for the incompressible Navier–Stokes equations: From fully implicit non-linear schemes to semi-implicit projection methods. Int J Numer Methods Fluids 1996;22:987–1011.
[49] Turek S. Efficient solvers for incompressible flow problems: an algorithmic and computational approach. In: Lecture Notes in Computational Science and Engineering. Springer-Verlag Berlin; 1999. p. 358.

Chapter 3

Robust High-order Semi-implicit Backward Difference Formulae (SBDF) for Navier–Stokes Equations

This chapter is presented in terms of a publication and will be submitted to International Journal for Numerical Methods and Fluids (IJNMF), carrying the same title as mentioned above. Please see the attached paper for the content.

This paper is co-authored with my supervisor. We discussed the selection of methods together. I implemented the methods, carried the computations and analyzed the results. Finally, I drafted the paper and my supervisor provided the editorial components.

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2017; **00**:1–24
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld

Robust high-order semi-implicit backward difference formulae (SBDF) for Navier–Stokes equations

K. C. Loy[1,2], Y. Bourgault*

[1]*Department of Mathematics & Statistics, 585 King Edward Avenue, Ottawa, ON, K1N 6N5, CANADA*
[2]*School of Informatics & Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, MALAYSIA*

SUMMARY

Taylor–Hood elements provide a robust numerical discretization of Navier–Stokes equations (NSE) in space. To handle unsteady flows, we propose a very efficient time-stepping method which is based on high-order semi-implicit backward difference formulae (SBDF). We construct SBDF methods by the inclusion of $\mathbf{grad}$-div term in the NSE. As a result, we observed a remarkable accuracy, stability and efficiency of the methods in terms of CPU time when computing unsteady flows. The presence of $\mathbf{grad}$-div term is also known to improve local mass conservation in Taylor–Hood elements. To reduce the numerical errors (in time), several variants of nonlinear extrapolation formulae for SBDF methods are investigated. The first approach is based on an extrapolation of the nonlinear advection term itself. The second formula uses the extrapolation of the velocity prior to the evaluation of the nonlinear advection term as a whole. The third variant is constructed such that it produces similar error on both velocity and pressure to that with fully implicit BDF methods at a given order of accuracy. This can be done by fixing one order higher in the extrapolation formula for the nonlinear advection term than is usually done, while keeping the same extrapolation formula for the time derivative. The resulting truncation errors (in time) between these formulae are identified using Taylor expansions. These truncation error formula are shown to properly represent the error seen in numerical tests using a 2D manufactured solution. Lastly, we show the robustness of the proposed semi-implicit methods by computing several test problems with high Reynold numbers using one of the nonlinear extrapolation formulae, namely the 2D flow past circular cylinder at $Re = 300$ and $Re = 1000$; and the 2D lid-driven cavity at $Re = 50\,000$ and $Re = 100\,000$. Our numerical solutions are found to be in a good agreement with those obtained in the literature, both qualitatively and quantitatively.

Received . . .

KEY WORDS: $\mathbb{P}_2$-$\mathbb{P}_1$ element, Navier–Stokes equations, semi-implicit backward difference formula, time-stepping, $\mathbf{grad}$-div stabilization, high Reynolds flows

1. INTRODUCTION

Method of lines (MOL) are versatile to solve time-dependent partial differential equations (PDEs) due to their capability to deal with the time-discretization independently from the space-discretization. Time-discretizations with MOL mimic the approach used to solve ordinary differential equations (ODEs), which offers many advantages in terms simplicity and computability. Backward difference formulae (BDF) are robust to solve many PDEs in particular Navier-Stokes equations (NSE) since they are unconditionally stable and accurate. However, BDF are fully implicit

*Correspondence to: Yves Bourgault, Department of Mathematics & Statistics, 585 King Edward Avenue, Ottawa, ON, CANADA, K1N 6N5, ybourg@uottawa.ca
†Author is a Ph.D scholarship recipient given by Ministry of Education Malaysia and Universiti Malaysia Terengganu. This work is also supported by the NSERC Discovery Grant; contract/grant number

methods, consequently fixed point iterations are needed to handle the nonlinear advection term. For unsteady flows with medium to high Reynolds numbers, very small time steps are required to resolve finer flow features, which renders these methods less competitive with explicit or uncoupled methods.

Among several possible schemes, semi-implicit methods based upon backward differentiation formulae (SBDF) are very competitive when computing unsteady flows [20]. With SBDF methods, the linear terms such as the diffusion and pressure terms are solved implicitly while the nonlinear advection term is evaluated explicitly using extrapolation formula. We emphasize two main benefits resulting from such semi-implicit schemes. First of all, no fixed point iteration for solving a nonlinear system is required. Only one linear system needs to be solved at every time step. Even better, SBDF methods produce a matrix that remains constant over time, which has proven to be CPU-efficient when computing unsteady flows. Though the time step can be much smaller than for BDF methods, the required CPU time is still significantly smaller than that with BDF methods. Similar advantage is observed when one needs to construct preconditioner when iterative solvers are involved.

Semi-implicit methods based on 2^{nd}-order Crank–Nicolson and Adam–Bashford (CNAB) for NSE have been studied by many authors in terms of numerical stability and convergence [13]. Being based on the trapezoidal rule, two constant matrices are constructed in CNAB method. When used to solve NSE, it is known that the trapezoidal rule produces oscillations from one time step to the next if the initial conditions are not properly imposed. Several tricks have been proposed to mitigate this problem which is non-trivial [25]. Methods of characteristics [26, 27], various projection steps [10, 36] and operator-splitting methods [28, 34] are popular alternatives but are limited to 2^{nd}-order of accuracy. Some of these methods still require fixed point methods to resolve nonlinear approximations (e.g., substeps for operator-splitting methods). High-order methods based on implicit Runge–Kutta formulae are very accurate and stable but come with with a greater price in terms their complexity [2]. However, the efficiency of RK methods remains unclear. For these reasons, we choose SBDF methods since they can be easily constructed for higher-order accuracy while maintaining simplicity.

SBDF methods are successful to solve nonlinear parabolic PDEs in various modeling applications (see e.g., [1, 3, 9, 18, 33]). For NSE, the earliest work found pertaining to the convergence analysis of high-order SBDF methods was done by Baker [7]. Few more extrapolation formulae for the nonlinear advection term in SBDF methods have been proposed, e.g. using high-order Adam–Bashford schemes [37] and ensemble method [14]). Unfortunately, these methods have not carefully studied in terms of the resulting truncation errors on the solution and their practical application for unsteady flows. The numerical effort in comparing both BDF and SBDF methods is found to be lacking as well.

In this paper, we focus on two major types of extrapolation formula. We compare these extrapolation formulae in terms of truncation error (in time) both using a simple mathematical analysis and numerical tests for 2^{nd}- and 3^{rd}-order methods. We realized that these two variants have not been properly compared in particular, when solving NSE numerically. BDF methods are typically more accurate compared to the regular SBDF methods for a given order of accuracy. In our work, we found a "new" formulation of SBDF which produced an error on velocity and pressure as small as with BDF methods. In the following section, we first introduce NSE and related functional spaces. We also provide the weak formulation of NSE, since the solution is sought using mixed finite element methods.

1.1. Nonstationary Incompressible Navier–Stokes equations

Consider an incompressible flow that occurs in a domain $\Omega \subset \mathbb{R}^d$ (d = 2 or 3) with a regular boundary Γ over a finite time interval $(0, T)$. The Navier–Stokes equations governing this flow is

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{f}, \quad \text{in } \Omega,\ t \in (0,T), \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \quad \text{in } \Omega,\ t \in (0,T), \tag{2}$$

$$\mathbf{u} = \mathbf{u}_0, \quad \text{in } \Omega,\ t = 0, \tag{3}$$

$$\mathbf{u} = 0, \quad \text{on } \Gamma, \tag{4}$$

where $\mathbf{u}$ is the velocity field, p is the pressure, $\mathbf{f}$ is the external force (such as gravity), ν is the viscosity and T is the total time. The initial value $\mathbf{u}_0$ is provided, for instance we can choose a divergence free velocity field or the state of rest. Other boundary conditions can also be considered here, e.g., Neumann type $\nabla\mathbf{u}\cdot\mathbf{n} = 0$, which implies uniqueness of the solution for the pressure as opposed to non-uniqueness for the Dirichlet boundary conditions (4).

1.2. Functional spaces and weak formulation

We introduce the mathematical notations and relevant functional spaces that are required in our analysis. Given an integer $s \geq 0$ and real $p \in [1,\infty]$, $W_p^s = W_p^s(\Omega)$ is the Sobolev spaces defined in the usual way for scalar real valued functions on the simply connected domain Ω and $\|\ .\ \|_{s,p}$ is the associated norms. We let $H^s = W_2^s$ and $\|\ .\ \|_s$, $|\ .\ |_s$ and $(.,.)_s$ be the associated norm, seminorm and inner product, respectively. For $s = 0$, we denote the norm on $W_0^p = L^p$ by $\|\ .\ \|_{L^p}$; in particular, on L^2 by $\|\ .\ \|_0$ and the associated L^2-inner product by $(.,.)$. We let $H_0^1(\Omega)$ be the space of those functions in H^1 which vanish on the boundary Γ and $H^{-1}(\Omega)$ be the dual space of $H_0^1(\Omega)$.

Further, $\mathbf{H}^s = (H^s)^d$ is the space of $\mathbb{R}^d$-valued functions $\mathbf{u} = (u_1,\ldots,u_d)$ defined on Ω such that $u_i \in H^s$, where $1 \leq i \leq d$. Similarly, we have $\mathbf{H}_0^1 = (H_0^1)^d$ and equip $\mathbf{H}^s$ with the inner product $(\mathbf{u},\mathbf{v})_s = \sum_{i=1}^d (u_i,v_i)_s$, resulting in the norm $\|\ .\ \|_s = (.,.)_s^{1/2}$. On $\mathbf{L}^2 = (L^2)^d$, the inner product and norm are defined as $(.,.)$ and $\|\ .\ \|_0$, respectively. The quotient space $L^2/\mathbb{R}$ or simply L_0^2 (also known as L^2 function with zero average over the domain Ω) is equipped with the norm $\|\mathbf{v}\|_{L^2/\mathbb{R}} = \inf_{c\in\mathbb{R}} \|\mathbf{v}+c\|_0$.

We reformulate the Navier–Stokes equations (1)–(4) as a mixed variational problem which can be conveniently solved by mixed finite element methods. In simple terms, this problem is expressed as follows: For a given $\mathbf{f} \in \mathbf{H}^{-1}$, find the solution $(\mathbf{u}(t), p(t)) \in \mathbf{H}_0^1 \times L_0^2$, for a.e $t \in (0,T)$, such that

$$\frac{d}{dt}(\mathbf{u},\mathbf{v}) + \nu(\nabla\mathbf{u},\nabla\mathbf{v}) + (\mathbf{u}\cdot\nabla\mathbf{u},\mathbf{v})$$

$$-(p,\nabla\cdot\mathbf{v}) = \langle f,v\rangle, \quad \forall\mathbf{v}\in\mathbf{H}_0^1, \tag{5}$$

$$-(q,\nabla\cdot\mathbf{u}) = 0, \quad \forall q \in L_0^2, \tag{6}$$

$$\mathbf{u}(0) = \mathbf{u}_0. \tag{7}$$

1.3. Space Discretization–$\mathbb{P}_2$-$\mathbb{P}_1$ finite elements with $\mathbf{grad}$-div *stabilization term*

Mixed finite element methods provide a well-documented robust numerical approach to compute solutions of (1)–(4). These methods are combined with time-stepping schemes to handle the time-derivative in the momentum equations, with the order of the time-stepping schemes being critical for dealing with unsteady flows. The Taylor–Hood element is one of many popular stable mixed finite elements (abbreviated as $\mathbb{P}_k$-$\mathbb{P}_{k-1}$, $k = 2,3,\ldots$) used to solve the Navier–Stokes equations (1)–(4) [4]. Here, we choose $k = 2$, giving rise to $\mathbb{P}_2$-$\mathbb{P}_1$ elements with 2^{nd}- and 1^{st}-order approximation in space for velocity and pressure, respectively. The $\mathbb{P}_2$-$\mathbb{P}_1$ finite element space approximates the velocity and pressure using continuous piecewise polynomials, respectively defined as

$$V_h = \{\mathbf{v}_h \in [\mathcal{C}^0(\overline{\Omega})]^d \mid \mathbf{v}_{h|K} \in [\mathbf{P}_2]^d, \forall K \in \mathcal{T}_h\}, \tag{8}$$

$$M_h = \{q_h \in \mathcal{C}^0(\overline{\Omega}) \mid q_{h|K} \in \mathbf{P}_1, \forall K \in \mathcal{T}_h\}. \tag{9}$$

The discrete spaces for velocity and pressure are conformal in the sense that $V_h \subset \mathbf{H}_0^1$ and $M_h \subset L_0^2$, respectively. The $\mathbb{P}_2$-$\mathbb{P}_1$ element is often chosen to compute 2D flows for its simplicity. It is

Int. J. Numer. Meth. Fluids (2017)
Prepared using fldauth.cls
DOI: 10.1002/fld

reasonably cheap in terms of CPU time and memory to allow the use of a direct solver. We implement this finite element discretization for Navier–Stokes equations by including a **grad**-div term to impose stronger mass conservation with $\mathbb{P}_2$-$\mathbb{P}_1$ elements. In general, Taylor–Hood elements are not pointwise mass conservative (see e.g. [17, 21, 23, 24]). This term acts as a penalization term bringing $\nabla \cdot \mathbf{u}_h$ closer to be pointwise null on each element K of the mesh.

The presence of the **grad**-div term also enhances the numerical stability when computing flows with high Reynolds numbers—the reason is often called a "**grad**-div stabilization term". As ν becomes small (large Reynolds number), the Navier–Stokes equations is still solvable as long as $\lambda > 0$, which is the key to stabilizing the discrete solution. However, if λ is taken too large, this results in so-called over-stabilization (numerical diffusion) and the possibility of ill-conditioning of the linear system, leading to issues when an iterative solver is used.

The resulting semi-discrete (in space) weak formulation of problem (1)–(4) is given as follows: For a given $\mathbf{f} \in \mathbf{L}^2 \subset \mathbf{H}^{-1}$ and a suitable $\lambda > 0$, find the solution of $(\mathbf{u}_h(t), p_h(t)) \in V_h \times M_h$, for all $t \in [0, T]$, such that

$$\left(\frac{\partial \mathbf{u}_h}{\partial t}, \mathbf{v}_h\right) + \nu(\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) + (\mathbf{u}_h \cdot \nabla \mathbf{u}_h, \mathbf{v}_h) \tag{10}$$

$$+\lambda(\nabla \cdot \mathbf{u}_h, \nabla \cdot \mathbf{v}_h) - (p_h, \nabla \cdot \mathbf{v}_h) = \langle f, v_h \rangle, \qquad \forall \mathbf{v}_h \in V_h,$$

$$-(q_h, \nabla \cdot \mathbf{u}_h) = 0, \qquad \forall q_h \in M_h, \tag{11}$$

$$\mathbf{u}_h(0) = \Pi_h \mathbf{u}(0), \tag{12}$$

where $\Pi_h : V \to V_h$ is an interpolation or projection operator.

1.4. The choice of the stabilization parameter $\boldsymbol{\lambda}$

The optimal choice of the stabilization parameter λ with the Taylor–Hood elements still remains an open problem. We observe that the optimal choice of λ is problem dependent, combining the following properties: characteristics of the flow, boundary conditions, external forcing, Reynolds number, 2D and 3D problem, computational mesh and the type of finite element pairs. It also depends on the norm used to assess the error from poor mass conservation and one's personal choice of either to minimizing the error on velocity or the error on pressure.

Several mathematical analyses have been done on the impact of **grad**-div stabilization with certain finite elements, mainly for Stokes equations (see [8, 17, 21, 23, 24] and reference therein). Several methodologies are available for choosing λ. Based on error estimates established for Stokes equations with **grad**-div term, Jenkins at al. [17] proposed an optimal λ, noted λ_{opt}, which is independent from ν.

$$\lambda_{\text{opt}} = \frac{C_{\mathcal{M}_h}}{C_{\mathcal{X}_h}} \frac{|p|_k}{|\mathbf{u}|_{k+1}}, \tag{13}$$

where $C_{\mathcal{M}_h}$ and $C_{\mathcal{X}_h}$ are constants from the interpolation estimates, with some dependence of $C_{\mathcal{X}_h}$ on β_h^{-1}, where $\beta_h > 0$ is the constant in the discrete inf-sup condition for a LLB-stable pair of elements. We remind the reader that k denotes the degree of the Taylor–Hood element, $\mathbb{P}_k$-$\mathbb{P}_{k-1}$. A dynamic and element-based stabilization parameter λ_K has been proposed for Stokes equations using similar error estimates [24]:

$$\lambda_K = \max_{K \in \mathcal{T}_h} \left\{ \frac{|p|_{k,K}}{|\mathbf{u}|_{k+1,K}} - \nu,\ 0 \right\}, \tag{14}$$

where K is any element of the mesh $\mathcal{T}_h$.

Keeping the advantage of a semi-implicit approach, we avoid computing λ based on the solution since this involves matrix rebuild at every time step, which reduces the efficiency in our methods. Similar inefficiency may occur if a preconditioner is involved in conjunction with an iterative method. In our case, we are more interested in finding the "best fixed" λ for the entire computation. The estimate $\lambda \sim \beta_h^2 = \mathcal{O}(1)$ is given in [24]. Combining (13) and (14), the choice of $\lambda = \mathcal{O}(1)$,

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

(e.g., $0 < \lambda < 10$) makes sense if $|\mathbf{u}|_{k+1} \approx |p|_k$; which can be observed when ν is small in many unsteady flows. This explains why the choice of $\lambda \in (0, 10)$ is often done. At this moment, the approach available to investigate the best fixed λ (and only for the sake of such investigation) is through a series of direct computations with many trials and errors based on minimizing the L^2-error between the solution and the exact solution (from the manufactured solution).

However, for test problems without manufactured solution, the best fixed λ can be approximated using similar trial and error by minimizing the L^2-error between the computed solution (e.g., velocity, pressure and divergence of velocity) and a reference solution. The reference solution can be generated using a reasonably fine time τ and mesh size h, and by fixing $\lambda = 0$.

Either way, the choice $\lambda = 1$ is often proposed as an initial guess and is gradually increased while the error on velocity, pressure and mass conservation are monitored within a time interval $t \in (0, T)$. We wish to have λ sufficiently large to produce the minimum time-averaged error for the above quantity within a time interval, but small enough to avoid over-stabilization that pollutes the solution. We observed that $\lambda \in (1, 5)$ works perfectly well for test problems involving lid-driven cavity at $Re = 8\,500$ [20], and Taylor–Green vortex at $Re = 100$ [20]. On the other hand, $\lambda \in (10^3, 10^4)$ is required to compute the flow around cylinder at $Re = 100$ [20]. This shows that the choice of λ is problem dependent. The choice of the stabilization parameter λ will be provided for each test case in later sections.

2. HIGHER-ORDER SEMI-IMPLICIT BACKWARD DIFFERENCE METHODS

SBDF methods are categorized as multistep schemes and conditionally stable. To maintain the numerical stability, the choice of time step and diameter of the mesh should not violate certain condition known as the CFL criterion which we will discuss next.

2.1. CFL stability criterion

We denote by τ the time step, $N = \frac{T}{\tau}$ the total number of time steps to reach time T and $t_n = n\tau$ for $n = 0, 1, \ldots, N$. The operator $B(\mathbf{u}_h)$ is the nonlinear advection term in the Navier–Stokes equations; i.e., $B(\mathbf{u}) = \mathbf{u} \cdot \nabla \mathbf{u}$. To ensure numerical stability, the choice of the time step τ is restricted by a condition on the CFL number

$$\text{CFL} := \max_{K \in \mathcal{T}_h} \left\{ \frac{\tau}{h_K} \|\mathbf{u}_h\|_{L^\infty(K)} \right\} \leq \text{CFL}_{\max}, \tag{15}$$

where h_K is the diameter of the element K and $\text{CFL}_{\max} > 0$ is a positive constant known as the critical or maximal CFL bound to maintain numerical stability. We define the critical time step τ_{crit} as the largest time step τ such that the condition (15) is satisfied with equality. A numerical solution is stable if for any sufficiently large time $T > 0$, the condition $\max\limits_{t_n \in (0,T)} \|\mathbf{u}_h^n\|_{L^\infty(\Omega)} = M < \infty$ is satisfied for some $M > 0$.

2.2. 2^{nd}-order SBDF method

Given a suitable approximation of the initial solution $\mathbf{u}_h^0 = \Pi_h \mathbf{u}(0) \in V_h$ and proper initializations for $\mathbf{u}_h^1 \in V_h$, $(\mathbf{u}_h^{n+1}, p_h^{n+1}) \in V_h \times M_h$ is given by

$$\begin{aligned} &\frac{1}{2\tau}\Big(3\mathbf{u}_h^{n+1} - 4\mathbf{u}_h^n + 2\mathbf{u}_h^{n-1}, \mathbf{v}_h\Big) + \nu(\nabla \mathbf{u}_h^{n+1}, \nabla \mathbf{v}_h) + \lambda(\nabla \cdot \mathbf{u}_h^{n+1}, \nabla \cdot \mathbf{v}_h) \\ &-(p_h^{n+1}, \nabla \cdot \mathbf{v}_h) = (\mathbf{f}^{n+1}, \mathbf{v}_h) - \Big(2B(\mathbf{u}_h^n) - B(\mathbf{u}_h^{n-1}), \mathbf{v}_h\Big), \qquad \forall \mathbf{v}_h \in V_h, \end{aligned} \tag{16}$$

$$-(q_h, \nabla \cdot \mathbf{u}_h^{n+1}) = 0, \qquad \forall q_h \in M_h, \tag{17}$$

for all $1 \leq n \leq N - 1$.

Int. J. Numer. Meth. Fluids (2017)
Prepared using fldauth.cls
DOI: 10.1002/fld

2.3. 3^{rd}-order SBDF method

Given a suitable approximation $\mathbf{u}_h^0 = \Pi_h \mathbf{u}(0) \in V_h$ and proper initializations for $\mathbf{u}_h^1, \mathbf{u}_h^2 \in V_h$, $(\mathbf{u}_h^{n+2}, p_h^{n+2}) \in V_h \times M_h$ is given by

$$\begin{aligned}\frac{1}{6\tau}\Big(11\mathbf{u}_h^{n+2} - 18\mathbf{u}_h^{n+1} + 9\mathbf{u}_h^n - 2\mathbf{u}_h^{n-1}, \mathbf{v}_h\Big) + \nu(\nabla \mathbf{u}_h^{n+2}, \nabla \mathbf{v}_h) + \\ \lambda(\nabla \cdot \mathbf{u}_h^{n+2}, \nabla \cdot \mathbf{v}_h) - (p_h^{n+2}, \nabla \cdot \mathbf{v}_h) = (\mathbf{f}_h^{n+2}, \mathbf{v}_h) \\ -\Big(3B(\mathbf{u}_h^{n+1}) - 3B(\mathbf{u}_h^n) + B(\mathbf{u}_h^{n-1}), \mathbf{v}_h\Big), \qquad \forall \mathbf{v}_h \in V_h, \qquad (18) \\ -(q_h, \nabla \cdot \mathbf{u}_h^{n+2}) = 0, \qquad \forall q_h \in M_h, \qquad (19)\end{aligned}$$

for all $1 \leq n \leq N-2$.

According to Baker et al. [7], under a suitable space approximations and whenever the methods satisfy the CFL condition, the k^{th}-order SBDF method converges as follows:

$$\|\mathbf{u}(t_n) - \mathbf{u}_h^n\|_1 = \mathcal{O}(\tau^k + h^r), \qquad \|p(t_n) - p_h^n\|_0 = \mathcal{O}(\tau^k + \tau^{-1}h^r + h^{r-1}), \qquad (20)$$

where $(\mathbf{u}(t_n), p(t_n))$ and $(\mathbf{u}_h^n, p_h^n)$ are the exact and fully discrete solutions, respectively. The initialization is done in a consecutive manner. First, we use the 1^{st}-order SBDF method to produce the required initial value $\mathbf{u}_h^1$ for the 2^{nd}-order SBDF method, which in turn will produce $\mathbf{u}_h^2$ such that the intended 3^{rd}-order SBDF method can be employed using $\mathbf{u}_h^1$ and $\mathbf{u}_h^2$. The parameter $k \in \mathbb{N}$ stands for the order of accuracy in time and $r \in \mathbb{N}$ the order of accuracy in space. By fixing the error in space (i.e., computing on the same mesh), the estimate (20) will be verified numerically for SBDF methods of 2^{nd}- and 3^{rd}-order of accuracy using a manufactured solution.

Since we are mainly interested in the impact of the time discretization on the accuracy in time of the numerical solution, we proceed with a consistency analysis in time. In the two SBDF methods given above, all linear terms are evaluated implicitly at the current time t_{n+1} for $k = 2$ (or t_{n+2} for $k = 3$), therefore the resulting rate of convergence (in time) depends only on the order of the finite difference scheme for the time-derivative $\frac{\partial}{\partial t}\mathbf{u}_h(t)$ and the extrapolation formula for the nonlinear advection term $B(\mathbf{u}_h(t))$. For instance, if a 2^{nd}-order scheme is used for the time-derivative while a 3^{rd}-order extrapolation formula is used for the nonlinear term, the resulting method is still limited to 2^{nd}-order of accuracy in time. If a k^{th}-order approximation is used for both terms, the resulting method has k^{th}-order of accuracy.

As an illustration, we show that the method (16)–(17) converges to (10)–(11) with 2^{nd}-order truncation error (in time). We replace $\mathbf{u}_h^n, \mathbf{u}_h^{n+1}, \mathbf{u}_h^{n+2}, \ldots$ with their semi-discrete counterparts $\mathbf{u}_h(t_n), \mathbf{u}_h(t_{n+1}), \mathbf{u}_h(t_{n+2}), \ldots$, for all $n \geq 0$, and we proceed similarly for pressure. Assuming the regularity of the semi-discrete solution $\mathbf{u}_h = \mathbf{u}_h(t) \in \mathcal{C}^3(0, T; V_h)$, we obtain

$$\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+1}) = \frac{3\mathbf{u}_h(t_{n+1}) - 4\mathbf{u}_h(t_n) + \mathbf{u}_h(t_{n-1})}{2\tau} + \frac{\tau^2}{3}\frac{\partial^3}{\partial t^3}\mathbf{u}_h(\hat{\xi}), \qquad (21)$$

where $\hat{\xi} = \hat{\xi}(\mathbf{x}) \in [t_{n-1}, t_{n+1}]$ for all $\mathbf{x} \in \Omega$. We expand the nonlinear terms $B(\mathbf{u}(t_{n+1}))$ and $B(\mathbf{u}(t_{n-1}))$ about the point t_n using a truncated Taylor series to get

$$B(\mathbf{u}_h(t_{n\pm1})) \quad = \quad B(\mathbf{u}_h(t_n)) \pm \tau\frac{\partial}{\partial t}B(\mathbf{u}_h(t_n))) + \frac{\tau^2}{2}\frac{\partial^2}{\partial t^2}(B(\mathbf{u}_h(\xi_\pm))),$$

where $\xi_+ = \xi_+(\mathbf{x}) \in [t_n, t_{n+1}]$ and $\xi_- = \xi_-(\mathbf{x}) \in [t_{n-1}, t_n]$ for all $\mathbf{x} \in \Omega$. Summing these two expressions, we eliminate the derivative $\dfrac{\partial}{\partial t}B(\mathbf{u}_h(t_n))$ and obtain

$$B(\mathbf{u}_h(t_{n+1})) = 2B(\mathbf{u}_h(t_n)) - B(\mathbf{u}_h(t_{n-1})) + \tau^2\frac{\partial^2}{\partial t^2}(B(\mathbf{u}_h(\hat{\xi}))), \qquad (22)$$

where $\hat{\xi} \in [\xi_-, \xi_+]$. Substituting (21) and (22) back into (16), we recover the variational formulation as in (10)–(11) at time $t = t_{n+1}$ with a truncation error in $\mathcal{O}(\tau^2)$. A similar approach can be used to obtain the truncation error for the 3^{nd}-order SBDF method.

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

3. ON SEVERAL NONLINEAR EXTRAPOLATIONS FORMULAE

There are several ways to extrapolate the nonlinear advection term at any specified order of accuracy. Suppose that we have the coefficients $\{c_j\}_{j=1}^{k-1}$ in an extrapolation formula, for $B(\mathbf{u}_h^k)$ of the form $\sum_{j=1}^{k-1} c_j B(\mathbf{u}_h^j)$. This extrapolation strategy in SBDF methods was used by many authors; e.g., to numerically solve convection-diffusion and reaction-diffusion equations [3, 9, 33], and Navier–Stokes equations [6] numerically. Another approach was proposed by Baker et al. [7] and uses the extrapolation formula $B\Big(\sum_{j=1}^{k-1} c_j \mathbf{u}_h^j\Big)$, the resulting methods are called SBDFB methods. Albeit the small difference between the two implementations, it remains unclear why SBDF methods are often preferred over SBDFB methods in many applications. We suspect that the two methods exhibit distinct numerical behaviour in terms of accuracy and stability. To our knowledge, no comparisons of these extrapolation schemes have been attempted in the literature. We also propose a new method an extrapolation type of the form $B\Big(\sum_{j=0}^{k-1} c_j \mathbf{u}_h^j\Big)$. This scheme results in a $(k+1)^{\text{th}}$-order approximation of the nonlinear term while only a k^{th}-order approximation is used for the time-derivative term. These methods will be called SBDFBEx (the suffix "Ex" is added to indicate an increase by one of the order of extrapolation of the nonlinear term). Although this combination of approximation may look suboptimal, we will see that these are the sole SBDF methods that can match the accuracy of BDF methods in practice.

We now state several variants of the SBDF and the BDF methods before making a numerical comparison between them.

3.1. 2^{nd}-order SBDFB method

Given $\mathbf{u}_h^0 = \Pi_h \mathbf{u}(0) \in V_h$ and proper initializations for $\mathbf{u}_h^1 \in V_h$, $(\mathbf{u}_h^{n+1}, p_h^{n+1}) \in V_h \times M_h$ is given by

$$\frac{1}{2\tau}\Big(3\mathbf{u}_h^{n+1} - 4\mathbf{u}_h^n + 2\mathbf{u}_h^{n-1}, \mathbf{v}_h\Big) + \nu(\nabla \mathbf{u}_h^{n+1}, \nabla \mathbf{v}_h) + \lambda(\nabla \cdot \mathbf{u}_h^{n+1}, \nabla \cdot \mathbf{v}_h)$$
$$-(p_h^{n+1}, \nabla \cdot \mathbf{v}_h) = (\mathbf{f}^{n+1}, \mathbf{v}_h) - \Big(B(2\mathbf{u}_h^n - \mathbf{u}_h^{n-1}), \mathbf{v}_h\Big), \quad \forall \mathbf{v}_h \in V_h, \tag{23}$$
$$-(q_h, \nabla \cdot \mathbf{u}_h^{n+1}) = 0, \quad \forall q_h \in M_h, \tag{24}$$

for all $1 \le n \le N-1$.

3.2. 3^{rd}-order SBDFB method

Given a suitable approximation $\mathbf{u}_h^0 = \Pi_h \mathbf{u}(0) \in V_h$ and proper initializations for $\mathbf{u}_h^1, \mathbf{u}_h^2 \in V_h$, $(\mathbf{u}_h^{n+2}, p_h^{n+2}) \in V_h \times M_h$ is given by

$$\frac{1}{6\tau}\Big(11\mathbf{u}_h^{n+2} - 18\mathbf{u}_h^{n+1} + 9\mathbf{u}_h^n - 2\mathbf{u}_h^{n-1}, \mathbf{v}_h\Big) + \nu(\nabla \mathbf{u}_h^{n+2}, \nabla \mathbf{v}_h) +$$
$$\lambda(\nabla \cdot \mathbf{u}_h^{n+2}, \nabla \cdot \mathbf{v}_h) - (p_h^{n+2}, \nabla \cdot \mathbf{v}_h) = (\mathbf{f}^{n+2}, \mathbf{v}_h)$$
$$-\Big(B(3\mathbf{u}_h^{n+1} - 3\mathbf{u}_h^n + \mathbf{u}_h^{n-1}), \mathbf{v}_h\Big), \quad \forall \mathbf{v}_h \in V_h, \tag{25}$$
$$-(q_h, \nabla \cdot \mathbf{u}_h^{n+2}) = 0, \quad \forall q_h \in M_h, \tag{26}$$

for all $1 \le n \le N-2$.

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

3.3. 2^{nd}-order SBDFBEx method

Given a suitable approximation $\mathbf{u}_h^0 = \Pi_h \mathbf{u}(0) \in V_h$ and proper initializations for $\mathbf{u}_h^1, \mathbf{u}_h^2 \in V_h$, $(\mathbf{u}_h^{n+2}, p_h^{n+2}) \in V_h \times M_h$ is given by

$$\frac{1}{2\tau}\Big(3\mathbf{u}_h^{n+2} - 4\mathbf{u}_h^{n+1} + 2\mathbf{u}_h^n, \mathbf{v}_h\Big) + \nu(\nabla \mathbf{u}_h^{n+2}, \nabla \mathbf{v}_h) + \lambda(\nabla \cdot \mathbf{u}_h^{n+2}, \nabla \cdot \mathbf{v}_h)$$
$$-(p_h^{n+2}, \nabla \cdot \mathbf{v}_h) = (\mathbf{f}^{n+2}, \mathbf{v}_h) - \Big(B(3\mathbf{u}_h^{n+1} - 3\mathbf{u}_h^n + \mathbf{u}_h^{n-1}), \mathbf{v}_h\Big), \quad \forall \mathbf{v}_h \in V_h, \tag{27}$$
$$-(q_h, \nabla \cdot \mathbf{u}_h^{n+2}) = 0, \qquad \forall q_h \in M_h, \tag{28}$$

for all $1 \leq n \leq N-1$.

3.4. 3^{rd}-order SBDFBEx method

Given a suitable approximation $\mathbf{u}_h^0 = \Pi_h \mathbf{u}(0) \in V_h$ and proper initializations for $\mathbf{u}_h^1, \mathbf{u}_h^2, \mathbf{u}_h^3 \in V_h$, $(\mathbf{u}_h^{n+3}, p_h^{n+3}) \in V_h \times M_h$ is given by

$$\frac{1}{6\tau}\Big(11\mathbf{u}_h^{n+3} - 18\mathbf{u}_h^{n+2} + 9\mathbf{u}_h^{n+1} - 2\mathbf{u}_h^n, \mathbf{v}_h\Big) + \nu(\nabla \mathbf{u}_h^{n+3}, \nabla \mathbf{v}_h) +$$
$$\lambda(\nabla \cdot \mathbf{u}_h^{n+3}, \nabla \cdot \mathbf{v}_h) - (p_h^{n+3}, \nabla \cdot \mathbf{v}_h) = (\mathbf{f}^{n+3}, \mathbf{v}_h)$$
$$-\Big(B(4\mathbf{u}_h^{n+2} - 6\mathbf{u}_h^{n+1} + 4\mathbf{u}_h^n - \mathbf{u}_h^{n-1}), \mathbf{v}_h\Big), \qquad \forall \mathbf{v}_h \in V_h, \tag{29}$$
$$-(q_h, \nabla \cdot \mathbf{u}_h^{n+3}) = 0, \qquad \forall q_h \in M_h, \tag{30}$$

for all $1 \leq n \leq N-2$,

3.5. 2^{nd}-order BDF method

Given a suitable approximation $\mathbf{u}_h^0 = \Pi_h \mathbf{u}(0) \in V_h$ and proper initializations for $\mathbf{u}_h^1 \in V_h$, $(\mathbf{u}_h^{n+1}, p_h^{n+1}) \in V_h \times M_h$ is given by

$$\frac{1}{2\tau}\Big(3\mathbf{u}_h^{n+1} - 4\mathbf{u}_h^n + 2\mathbf{u}_h^{n-1}, \mathbf{v}_h\Big) + \nu(\nabla \mathbf{u}_h^{n+1}, \nabla \mathbf{v}_h) + \lambda(\nabla \cdot \mathbf{u}_h^{n+1}, \nabla \cdot \mathbf{v}_h) +$$
$$(B(\mathbf{u}_h^{n+1}), \mathbf{v}_h) - (p_h^{n+1}, \operatorname{div} \mathbf{v}_h) = (\mathbf{f}^{n+1}, \mathbf{v}_h), \quad \forall \mathbf{v}_h \in V_h, \tag{31}$$
$$-(q_h, \nabla \cdot \mathbf{u}_h^{n+1}) = 0, \qquad \forall q_h \in M_h, \tag{32}$$

for all $1 \leq n \leq N-1$.

3.6. 3^{rd}-order BDF method

Given a suitable approximation $\mathbf{u}_h^0 = \Pi_h \mathbf{u}(0) \in V_h$ and proper initializations for $\mathbf{u}_h^1, \mathbf{u}_h^2 \in V_h$, $(\mathbf{u}_h^{n+2}, p_h^{n+2}) \in V_h \times M_h$ is given by

$$\frac{1}{6\tau}\Big(11\mathbf{u}_h^{n+2} - 18\mathbf{u}_h^{n+1} + 9\mathbf{u}_h^n - 2\mathbf{u}_h^{n-1}, \mathbf{v}_h\Big) + \nu(\nabla \mathbf{u}_h^{n+2}, \nabla \mathbf{v}_h) +$$
$$\lambda(\nabla \cdot \mathbf{u}_h^{n+2}, \nabla \cdot \mathbf{v}_h) + (B(\mathbf{u}_h^{n+2}), \mathbf{v}_h)$$
$$-(p_h^{n+2}, \nabla \cdot \mathbf{v}_h) = (\mathbf{f}^{n+2}, \mathbf{v}_h), \qquad \forall \mathbf{v}_h \in V_h, \tag{33}$$
$$-(q_h, \nabla \cdot \mathbf{u}_h^{n+2}) = 0, \qquad \forall q_h \in M_h, \tag{34}$$

for all $1 \leq n \leq N-2$.

We refer the reader for several initialization techniques of these methods to Baker et al. [7] and Loy and Bourgault [20]. Fixed point iterations, (e.g., Newton method) are required to solve the nonlinear systems resulting from BDF methods. BDF methods are unconditionally stable since the time step is not restricted by a CFL condition. Larger time steps are allowed in BDF methods, hence rendering these methods very efficient to compute steady flows. On the other hand, a smaller time

DOI: 10.1002/fld

step has to be chosen to produce good accuracy for unsteady flows, often resulting in very poor efficiency.

For a chosen time step τ and a fixed mesh which satisfy the CFL condition (15), the resulting truncation errors (due to the nonlinear extrapolation formula) for SBDF and SBDFB methods can be studied using a simple mathematical analysis. For this purpose, we will evaluate only the truncation error resulting from the various extrapolation formula for the nonlinear term. We first assume that the regularity of the solution in time $\mathbf{u}_h \in \mathcal{C}^4(0,T;V_h)$ holds. The Taylor expansions of $\mathbf{u}_h(t_n)$ and $\mathbf{u}_h(t_{n+1})$ about the point $t = t_{n+2}$ are given by

$$\begin{aligned}
\mathbf{u}_h(t_n) &= \mathbf{u}_h(t_{n+2}) - 2\tau\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+2}) + 2\tau^2\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2}) - \frac{4\tau^3}{3}\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+2}) \qquad (35)\\
&\quad + \frac{2\tau^4}{3}\frac{\partial^4}{\partial t^4}\mathbf{u}_h(\xi_1), \quad \xi_1 = \xi_1(\mathbf{x}) \in (t_n, t_{n+1}),\\
\mathbf{u}_h(t_{n+1}) &= \mathbf{u}_h(t_{n+2}) - \tau\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+2}) + \frac{\tau^2}{2}\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2}) - \frac{\tau^3}{6}\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+2}) \qquad (36)\\
&\quad + \frac{\tau^4}{24}\frac{\partial^4}{\partial t^4}\mathbf{u}_h(\xi_2), \quad \xi_2 = \xi_2(\mathbf{x}) \in (t_{n+1}, t_{n+2}).
\end{aligned}$$

Combining (35) and (36), we obtain the following result.

$$\begin{aligned}
2\mathbf{u}_h(t_{n+1}) - \mathbf{u}_h(t_n) &= \mathbf{u}_h(t_{n+2}) - \tau^2\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2}) + \tau^3\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+2}) \qquad (37)\\
&\quad - \frac{7\tau^4}{12}\frac{\partial^4}{\partial t^4}\mathbf{u}_h(\xi_3), \quad \xi_3 = \xi_3(\mathbf{x}) \in (t_n, t_{n+2}).
\end{aligned}$$

For brevity, we introduce the notation $B(\mathbf{u},\mathbf{v}) := \mathbf{u}\cdot\nabla\mathbf{v}$. By applying (37) to the nonlinear operator defined in the momentum equation as in (16) and (23) for SBDF-2 and SBDFB-2 methods, respectively, we obtain the following results.

Extrapolation of the nonlinear terms in SBDF-2 method

$$\begin{aligned}
2B(\mathbf{u}_h(t_{n+1})) - B(\mathbf{u}_h(t_n)) &= 2B\Big(\mathbf{u}_h(t_{n+2}) - \tau\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+2}) + \frac{\tau^2}{2}\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2})\\
&\quad - \frac{\tau^3}{6}\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+2}) + \frac{\tau^4}{24}\frac{\partial^4}{\partial t^4}\mathbf{u}_h(\eta_2)\Big)\\
&\quad - B\Big(\mathbf{u}_h(t_{n+2}) - 2\tau\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+2}) + 2\tau^2\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2})\\
&\quad - \frac{4\tau^3}{3}\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+2}) + \frac{2\tau^4}{3}\frac{\partial^4}{\partial t^4}\mathbf{u}_h(\eta_1)\Big)\\
&= B(\mathbf{u}_h(t_{n+2}))\\
&\quad - 2\tau^2 B\Big(\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+2})\Big) - \tau^2 B\Big(\mathbf{u}_h(t_{n+2}), \frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2})\Big)\\
&\quad - \tau^2 B\Big(\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2}), \mathbf{u}_h(t_{n+2})\Big) + \mathcal{O}(\tau^3),
\end{aligned}$$

which implies that the truncation error on $B(\mathbf{u}_h(t_{n+2}))$ is

$$\begin{aligned}
&\Big|\Theta_{\mathrm{SBDF2}}(t_{n+2})\Big| := \Big|B(\mathbf{u}_h(t_{n+2})) - \big(2B(\mathbf{u}_h(t_{n+1})) - B(\mathbf{u}_h(t_n))\big)\Big|\\
\le\ & 4\tau^2 \max\Big\{\Big|B\Big(\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+2})\Big)\Big|, \Big|B\Big(\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2}), \mathbf{u}_h(t_{n+2})\Big)\Big|,\\
&\Big|B\Big(\mathbf{u}_h(t_{n+2}), \frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2})\Big)\Big|\Big\} + \mathcal{O}(\tau^3).
\end{aligned}$$

Int. J. Numer. Meth. Fluids (2017)
Prepared using fldauth.cls
DOI: 10.1002/fld

Extrapolation of the nonlinear terms in SBDFB-2 method

$$\begin{aligned} B(2\mathbf{u}_h(t_{n+1}) - \mathbf{u}_h(t_n)) &= B\Big(\mathbf{u}_h(t_{n+2}) - \tau^2 \frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2}) + \tau^3 \frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+2}) \\ &\quad -\frac{7\tau^4}{12}\frac{\partial^4}{\partial t^4}\mathbf{u}_h(\eta_1)\Big) \\ &= B(\mathbf{u}_h(t_{n+2})) - \tau^2 B\Big(\mathbf{u}_h(t_{n+2}), \frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2})\Big) \\ &\quad -\tau^2 B\Big(\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2}), \mathbf{u}_h(t^{n+2})\Big) + \mathcal{O}(\tau^3). \end{aligned}$$

which implies that the truncation error on $B(\mathbf{u}_h(t_{n+2}))$ is

$$\begin{aligned} &\Big|\Theta_{\mathrm{SBDFB2}}(t_{n+2})\Big| := \Big|B(\mathbf{u}_h(t_{n+2})) - \big(B(2\mathbf{u}_h(t_{n+1}) - \mathbf{u}_h(t_n))\big)\Big| \\ \leq\ & 2\tau^2 \max\Big\{\Big|B\Big(\mathbf{u}_h(t_{n+2}), \frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2})\Big)\Big|, \Big|B\Big(\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+2}), \mathbf{u}_h(t^{n+2})\Big)\Big|\Big\} \\ & +\mathcal{O}(\tau^3). \end{aligned}$$

It is worth-mentioning that the truncation error on $\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+2})$ in all of SBDF-2, SBDFB-2 and SBDFBEx-2 methods is asymptotically the same; i.e., $\mathcal{O}(\tau^2)$ by virtue of (21). Therefore, any numerical difference between these methods results from the different extrapolation strategy used to approximate the nonlinear term. For instance, we observe that the truncation error for SBDFB-2 method is smaller than for SBDF-2 method at least by a factor 2.

For the 3^{rd}-order extrapolation schemes, we use a similar approach. We first write using Taylor expansion for $\mathbf{u}_h(t_n)$, $\mathbf{u}_h(t_{n+1})$ and $\mathbf{u}_h(t_{n+2})$ at $t = t_{n+3}$ to obtain

$$\begin{aligned} 3\mathbf{u}_h(t_{n+2}) - 3\mathbf{u}_h(t_{n+1}) + \mathbf{u}_h(t_n) &= \mathbf{u}_h(t_{n+3}) - \tau^3 \frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3}) + \frac{3\tau^4}{2}\frac{\partial^4}{\partial t^4}\mathbf{u}_h(\xi), \\ &\quad \xi = \xi(\mathbf{x}) \in (t_n, t_{n+3}). \end{aligned} \tag{38}$$

By applying (38) to the nonlinear operator in the momentum equation of (18) and (25) for SBDF-3 and SBDFB-3 methods, respectively, we obtain the following results.

***Extrapolation of nonlinear term in SBDF-3* method**

$$\begin{aligned} 3B(\mathbf{u}_h(t_{n+2})) - 3B(\mathbf{u}_h(t_{n+1})) + B(\mathbf{u}_h(t_n))) &= B(\mathbf{u}_h(t_{n+3})) \\ &\quad -\tau^3 B\Big(\mathbf{u}_h(t_{n+3}), \frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3})\Big) \\ &\quad -\tau^3 B\Big(\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3}), \mathbf{u}_h(t_{n+3})\Big) \\ &\quad -3\tau^3 B\Big(\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+3}), \frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+3})\Big) \\ &\quad -3\tau^3 B\Big(\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+3}), \frac{\partial}{\partial t}\mathbf{u}_h(t_{n+3})\Big) \\ &\quad +\mathcal{O}(\tau^4), \end{aligned}$$

which implies that the truncation error on $B(\mathbf{u}_h(t_{n+3}))$ is

$$\begin{aligned} &\Big|\Theta_{\mathrm{SBDF3}}(t_{n+3})\Big| := \\ &\Big|B(\mathbf{u}_h(t_{n+3})) - \big(3B(\mathbf{u}_h(t_{n+2})) - 3B(\mathbf{u}_h(t_{n+1})) + B(\mathbf{u}_h(t_n))\big)\Big| \\ \leq\ & 8\tau^3 \max\Big\{\Big|B\Big(\mathbf{u}_h(t_{n+3}), \frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3})\Big)\Big|, \Big|B\Big(\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3}), \mathbf{u}_h(t_{n+3})\Big)\Big|, \\ & \Big|B\Big(\frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+3}), \frac{\partial}{\partial t}\mathbf{u}_h(t_{n+3})\Big)\Big|, \Big|B\Big(\frac{\partial}{\partial t}\mathbf{u}_h(t_{n+3}), \frac{\partial^2}{\partial t^2}\mathbf{u}_h(t_{n+3})\Big)\Big|\Big\} + \mathcal{O}(\tau^4). \end{aligned}$$

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

Extrapolation of nonlinear term in SBDFB-3 method

$$\begin{aligned} B(3\mathbf{u}_h(t_{n+2}) - 3\mathbf{u}_h(t_{n+1}) + \mathbf{u}_h(t_n)) &= B\Big(\mathbf{u}_h(t_{n+3}) - \tau^3 \frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3}) \\ &\quad + \frac{3\tau^4}{2}\frac{\partial^4}{\partial t^4}\mathbf{u}_h(\xi_4)\Big) \\ &= B(\mathbf{u}_h(t_{n+3})) - \tau^3 B\Big(\mathbf{u}_h(t_{n+3}), \frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3})\Big) \\ &\quad -\tau^3 B\Big(\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3}), \mathbf{u}(t_{n+3})\Big), \end{aligned}$$

which implies that the truncation error on $B(\mathbf{u}_h(t_{n+3}))$ is

$$\begin{aligned} \Big|\Theta_{\mathrm{SBDFB3}}(t_{n+3})\Big| &:= \Big|B(\mathbf{u}_h(t_{n+3})) - B(3\mathbf{u}_h(t_{n+2}) - 3\mathbf{u}_h(t_{n+1}) + \mathbf{u}_h(t_n))\Big| \\ \leq \quad 2\tau^3 \max\Big\{\Big|B\Big(\mathbf{u}_h(t_{n+3}), \frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3})\Big)\Big|, \Big|B\Big(\frac{\partial^3}{\partial t^3}\mathbf{u}_h(t_{n+3}), \mathbf{u}_h(t_{n+3})\Big)\Big|\Big\} + \mathcal{O}(\tau^4). \end{aligned}$$

By comparing the truncation error in time, we see that SBDFB-3 method produces an error smaller than SBDF-3 method at least by a factor of 4. The nonlinear extrapolation formula for $B(\mathbf{u})$ in SBDFBEx-2 (see (27)) has 3^{rd}-order of accuracy in time. As we mentioned earlier, the resulting method is still limited to 2^{nd}-order accuracy from the discretizations of the time-derivative. However, it occurs that the resulting error with SBDFBEx-2 method will be less than that with SBDFB-2 method. It is noteworthy that the overall error produced by SBDFBEx-2 method (whenever it is stable) is similar to that of BDF-2 method, for which there is no truncation error in time resulting from the discretization of $B(\mathbf{u}_h^{n+2})$. However, the numerical convergence and error in BDF methods are only influenced by the discretization of the time-derivative as long as the stopping criteria for Newton's method is set close to machine accuracy (see (41) below). At a given order, BDF and SBDFBEx methods are the most accurate methods among all the discussed variants. This observation can be generalized to any k^{th}-order BDF and SBDFBEx methods.

3.7. Numerical results

A manufactured solution is used to verify that the error estimates based on truncation error that we derived above correspond to the practical behaviour in numerical test cases. This is done by comparing the numerical error (in time) computed with SBDF, SBDFB, SBDFBEx and BDF methods, all with 2^{nd}- and 3^{rd}-order of accuracy. Further, we attempt to recover the error factors among these methods since they are influenced by the extrapolation strategy implemented to approximate the nonlinear terms. For this purpose, we consider the domain $\Omega = (0,1)^2$ and the solution of (1)–(2) given by

$$\begin{aligned} (u_1, u_2) &= (\sin(x)\sin(y+t), \cos(x)\cos(y+t)), \\ p &= \cos(x)\sin(y+t). \end{aligned} \tag{39}$$

The domain is discretized using a uniform mesh which results in a fixed mesh size $h = 0.01768$ (subdivision 80×80 along both edges). This produces 6561 vertices, 12800 triangles and 58403 unknowns for velocity and pressure. The external force $\mathbf{f}$, the initial condition $(\mathbf{u}_0, p_0)$, the non-homogeneous Dirichlet boundary conditions at bottom, left and upper boundaries, and homogeneous Neumann boundary condition for left boundary are computed from the analytical solution in (39). The final time for computation is defined as $T = 2$. The computation will be done with different time steps $\tau = 0.04, 0.02, 0.01$ and 0.005. For the error and convergence analysis in time only (as opposed to space and time), the reference solution $(\mathbf{u}_{h,\mathrm{ref}}, p_{h,\mathrm{ref}})$ is computed using SBDFBEx-3 method with a very small time step $\tau = 6.25 \times 10^{-5}$. SBDFBEx-3 method is chosen since it is CPU-efficient and it has the potential to produce the least numerical error at a given order.

We define $T = M\tau^*$ where T is the total computation time τ^* the time step for the output purposes and $M \in \mathbb{N}$ a user defined parameter. For convenience, we pick τ^* as a multiple of τ.

DOI: 10.1002/fld

We analyze numerical convergence in time by comparing the numerical and reference solutions for all methods. The error on velocity (in time) is computed as follows:

$$\|\mathbf{u}_{h,\mathrm{ref}} - \mathbf{u}_{h,\tau}\|_{L^2(0,T;L^2(\Omega))} = \left(\tau^* \sum_{j=1}^{M} \|\mathbf{u}^j_{h,\mathrm{ref}} - \mathbf{u}^j_{h,\tau}\|^2\right)^{\frac{1}{2}}. \tag{40}$$

To disregard possible numerical artifact generated at start-up, we evaluate the L^2-error using $\tau^* = 0.2$ only for $t \in [1, 2]$. For BDF methods, we set the stopping criteria for the Newton's method as follows:

$$\|\mathbf{u}^{n+1,k+1}_{h,\tau} - \mathbf{u}^{n+1,k}_{h,\tau}\|_{L^2(\Omega)} \leq \mathrm{tol}, \quad \text{where } \mathrm{tol} = 10^{-12}, \tag{41}$$

where $\mathbf{u}^{n+1,k+1}_{h,\tau}$ is the solution at time t^{n+1} for the k^{th}-Newton iteration.

From Table I and II, we observe that all methods converge to the reference solution computed by the SBDFBEx-3 method. For each time step, the velocity error produced with SBDFB-2 method is about twice as small as the error produced with SBDF-2 method while there is little difference between the error on pressure for both methods. On the other hand, the error on both velocity and pressure with SBDFB-3 method is about four times smaller than that for SBDF-3 method. Again, we note that the resulting error with SBDFBEx-2 is smaller than that with SBDFB-2 method, by a factor of 2 and 4 comparing the error on velocity and pressure, respectively. A similar comparison of the error on velocity between SBDFBEx-3 and SBDFB-3 methods produce a factor of about 5.5–7.4 smaller while the error on pressure is about 3.0–3.4 smaller. The error ratio that we obtained numerically for the velocity agrees well with the theoretical ratio of truncation error estimates obtained earlier.

At a given order, the error on velocity and pressure produced by SBDFBEx methods is very closed to the error produced by BDF methods. By using a $(k+1)^{\mathrm{th}}$-order extrapolation scheme for $B(\mathbf{u}_h)$ in SBDFBEx methods, it can be seen in Table I and II that the accuracy of this semi-implicit scheme is similar to fully-implicit BDF methods. This suggests that the error from the nonlinear extrapolation is minimized and any time discretization error comes mostly from the approximation of the time-derivative.

For all methods that we have studied, the theoretical order of convergence are reproduced satisfactorily. Whenever smaller error is required, semi-implicit methods demonstrate a significant saving on CPU time compared to the implicit counterpart (by a factor of 50) when solving unsteady NSE. The CPU times for 2^{nd} and 3^{rd}-order semi-implicit methods are small and almost similar. Larger differences are observed between the implicit methods of 2^{nd}- and 3^{rd}-order of accuracy. This suggests that more Newton's iterations are required for solving the nonlinear system in BDF-3 method than that in BDF-2 method, even with the same tolerance in the stopping criteria.

4. 2D FLOW AROUND THE CYLINDER AT $Re = 300$ AND 1 000

In this section, we showcase that our methods are robust, accurate and stable when computing flows with high Reynolds numbers. For this purpose, we first attempt the computation of 2D flows around the cylinder at $Re = 300$ and 1 000. For the domain, we use $\Omega = (-5, 15) \times (-5, 5)$ with a 2D cylinder of diameter $D = 1$ centered at the origin $(0, 0)$. The domain is discretized using a triangulation (unstructured) by dividing the long and short edges of the domain into 320 and 160 elements, respectively, and by using 160 nodes on the boundary of the circular cylinder. The mesh has elements of varying mesh size h_K where $0.016\,774 \leq h_K \leq 0.127\,691$ for a total of 181 000 triangles. This gives 721 760 and 90 500 d.o.f. for velocity and pressure, respectively. This space discretization is chosen to be reasonably fine to ensure that there are sufficiently many points to resolve the boundary layer.

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

Table I. L^2-error and order of convergence for velocity and pressure (in time), and CPU time of all 2^{nd}-order methods.

Error (SBDF-2)			**Order (SBDF-2)**		**CPU(s)**
Time step, τ	**Velocity**	**Pressure**	**Velocity**	**Pressure**	
0.04	2.4309×10^{-5}	5.4118×10^{-4}	1.9779	1.9810	26
0.02	6.1711×10^{-6}	1.3709×10^{-4}	1.9894	1.9909	42
0.01	1.5542×10^{-6}	3.4490×10^{-5}	1.9948	1.9956	75
0.005	3.8993×10^{-7}	8.6491×10^{-6}	–	–	134
Error (SBDFB-2)			**Order (SBDFB-2)**		**CPU(s)**
Time step, τ	**Velocity**	**Pressure**	**Velocity**	**Pressure**	
0.04	1.0473×10^{-5}	5.7804×10^{-4}	1.9893	2.0027	26
0.02	2.6377×10^{-6}	1.4424×10^{-4}	1.9944	2.0013	44
0.01	6.6199×10^{-7}	3.6028×10^{-5}	1.9971	2.0007	72
0.005	1.6583×10^{-7}	9.0028×10^{-6}	–	–	132
Error (SBDFBEx-2)			**Order (SBDFBEx-2)**		**CPU(s)**
Time step, τ	**Velocity**	**Pressure**	**Velocity**	**Pressure**	
0.04	5.0956×10^{-6}	1.1369×10^{-4}	1.9534	1.9526	26
0.02	1.3157×10^{-6}	2.9372×10^{-5}	1.9783	1.9789	59
0.01	3.3392×10^{-7}	7.4513×10^{-6}	1.9895	1.9901	94
0.005	8.4090×10^{-8}	1.8757×10^{-6}	–	–	164
Error (BDF-2)			**Order (BDF-2)**		**CPU(s)**
Time step, τ	**Velocity**	**Pressure**	**Velocity**	**Pressure**	
0.04	5.4965×10^{-6}	1.2127×10^{-4}	2.0099	2.0025	1 421
0.02	1.3647×10^{-6}	3.0264×10^{-5}	2.0051	2.0013	2 361
0.01	3.3998×10^{-7}	7.5594×10^{-6}	2.0025	2.0006	4 298
0.005	8.4846×10^{-8}	1.8890×10^{-6}	–	–	7 683

4.1. Nonreflecting Boundary–Directional Implicit Damping

At a lower Reynolds number; e.g., $Re = 100$, we observe that Neumann boundary condition works perfectly well to allow the generated vortices to slip through the free exit. As the Reynolds number increases, the flow becomes more sensitive to perturbation including those induced by boundary condition. Neumann boundary conditions on the outflow boundary do not properly damp the wave generated by the vortices exiting the domain. The instability shows up as highly oscillating velocity patterns along this boundary, leading to an unbounded solution in few time steps. The flow around the cylinder at high Reynolds number thus requires a special treatment along the outflow boundary. We apply a simple approach called the directional implicit damping [29], which shares similar principles with the perfectly-matched-layer (PML). The idea of directional implicit damping is to absorb fast moving waves but only in a layer near the boundary (see Figure 1). The absorbing layer lies in the zone between the dashed-line and the actual boundary Γ of the domain Ω (i.e., top, bottom and right boundaries). The thickness of this layer is defined by $\xi > 0$. In this absorbing layer, NSE with damping term $\sigma\mathbf{u}$ are solved, while the regular NSE are solved in the middle (as indicated).

A modified NSE with $\mathbf{grad}$-div stabilization term incorporating the damping term $\sigma\mathbf{u}$ are given as follows:

$$\mathbf{u}_t + \sigma\mathbf{u} - \nu\Delta\mathbf{u} + \lambda\nabla\nabla\cdot\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{f}, \quad \text{in } \Omega\times(0,T), \tag{42}$$

$$\nabla\cdot\mathbf{u} = 0, \quad \text{in } \Omega\times(0,T), \tag{43}$$

$$\nabla\mathbf{u}\cdot\mathbf{n} = 0, \quad \text{on } \Gamma_{\text{out}}\times(0,T), \tag{44}$$

$$\mathbf{u} = 0, \quad \text{on } \Gamma\backslash\Gamma_{\text{out}}, \tag{45}$$

$$\mathbf{u}|_{t=0} = \mathbf{u}_0, \quad \text{on } \Omega, \tag{46}$$

Int. J. Numer. Meth. Fluids (2017)
Prepared using fldauth.cls
DOI: 10.1002/fld

where the damping function, $\sigma_{x,y} = \sigma(x,y)$ is called a "half-step" function. In a rectangular domain $\Omega = (x_{\min}, x_{\max}) \times (y_{\min}, y_{\max})$, the function $\sigma = \sigma(x,y)$ is defined as follows [29]:

$$\sigma(x,y) = \begin{cases} 0, & \text{if } x = [0, x_{\max} - \xi] \text{ and } \ y \in [\xi, y_{\max} - \xi], \\ \frac{x - x_{\max} + \xi}{\xi} - \frac{1}{\pi}\sin\left(\frac{(x - x_{\max} + \xi)\pi}{\xi}\right), & \text{if } x \in [x_{\max} - \xi, x_{\max}] \text{ and } y \in [y_{\min} + \xi, y_{\max} - \xi], \\ \frac{y - y_{\max} + \xi}{\xi} - \frac{1}{\pi}\sin\left(\frac{(y - y_{\max} + \xi)\pi}{\xi}\right), & \text{if } y \in [y_{\max} - \xi, y_{\max}], \\ \frac{y_{\min} + \xi - y}{\xi} - \frac{1}{\pi}\sin\left(\frac{(y_{\min} + \xi - y)\pi}{\xi}\right), & \text{if } y \in [y_{\min}, y_{\min} + \xi], \end{cases} \tag{47}$$

where we denote the left bound by $x_{\min}$, the right bound by $x_{\max}$, the lower bound by $y_{\min}$ and the upper bound by $y_{\max}$. The method works well with a small time step, which is perfectly suited for the semi-implicit methods. The parameter ξ is a user-defined parameter denoting the "thickness" of the absorbing layer. A larger parameter ξ (typically $\xi \in (3,5)$) is prescribed to allow wave damping on fast moving vortices while a smaller ξ (typically $\xi \in (1,3)$) is adequate to damp the slow moving vortices. In this particular test case, we fix $\xi = 3$ which produces a reasonable damping on the outgoing vortices.

Table II. L^2-error and order of convergence for velocity and pressure (in time), and CPU time of all 3^{rd}-order methods.

Error (SBDF-3)			**Order (SBDF-3)**		**CPU(s)**
Time step, τ	**Velocity**	**Pressure**	**Velocity**	**Pressure**	
0.04	2.0906×10^{-6}	4.5220×10^{-5}	3.0419	3.0368	33
0.02	2.5385×10^{-7}	5.5102×10^{-6}	3.0217	3.0189	56
0.01	3.1257×10^{-8}	6.7983×10^{-7}	3.0113	3.0099	92
0.005	3.8768×10^{-9}	8.4399×10^{-8}	–	–	169
Error (SBDFB-3)			**Order (SBDFB-3)**		**CPU(s)**
Time step, τ	**Velocity**	**Pressure**	**Velocity**	**Pressure**	
0.04	5.3496×10^{-7}	1.1609×10^{-5}	3.0169	2.9954	25
0.02	6.6092×10^{-8}	1.4557×10^{-6}	3.0086	2.9960	54
0.01	8.2121×10^{-9}	1.8247×10^{-7}	3.0056	2.9994	93
0.005	1.0225×10^{-9}	2.2817×10^{-8}	–	–	156
Error (SBDFBEx-3)			**Order (SBDFBEx-3)**		**CPU(s)**
Time step, τ	**Velocity**	**Pressure**	**Velocity**	**Pressure**	
0.04	7.2254×10^{-8}	3.9216×10^{-6}	2.7512	3.1185	28
0.02	1.0732×10^{-8}	4.5155×10^{-7}	2.8861	3.0515	50
0.01	1.4516×10^{-9}	5.4464×10^{-8}	2.9606	3.0350	89
0.005	1.8648×10^{-10}	6.6450×10^{-9}	–	–	164
Error (BDF-3)			**Order (BDF-3)**		**CPU(s)**
Time step, τ	**Velocity**	**Pressure**	**Velocity**	**Pressure**	
0.04	9.3772×10^{-8}	3.3535×10^{-6}	2.9537	2.9955	1 636
0.02	1.2104×10^{-8}	4.2049×10^{-7}	2.9814	3.0012	3 014
0.01	1.5326×10^{-9}	5.2518×10^{-8}	3.0230	3.0283	5 101
0.005	1.8855×10^{-10}	6.4375×10^{-9}	–	–	8 568

Boundary conditions are still required and set as follows: $\mathbf{u} = (u,v) = (1,0)$ along left, upper and lower boundaries. Setting zero tangential flow (non-flux condition) along the upper and lower boundaries can be done, i.e., $\partial_x u = 0$ and $v = 0$, but we note that the difference between this and the earlier prescribed velocity is negligible. Along the cylinder, no-slip condition is used, i.e., $\mathbf{u} = (u,v)|_\Gamma = (0,0)$.

From the manufactured solution, we note that SBDFBEx methods produce the least error on velocity and pressure. However, SBDFBEx methods are very restrictive in terms of critical time step to maintain CFL stability condition (about twice as smaller as that for SBDF/SBDFB methods

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

at a given order of accuracy). Through numerical experiment, we observed that SBDFB methods are more accurate than SBDF methods and having similar critical time step to that of SBDF to maintain CFL stability condition. Hence SBDFB-3 method is chosen to solve the two test cases, the first with a time step $\tau = 0.002$ at $Re = 300$, and the second with a time step $\tau = 0.001$ at $Re = 1\,000$. These time steps are chosen small enough to maintain numerical stability and to yield an accurate solution. The parameter $\lambda = 10^2$ is chosen for the $\mathbf{grad}$-div stabilization term. The resulting linear system can still be handled with a direct solver, MUMPS, which makes the SBDFB-3 method very efficient on these test problems. Both computations are run without prescribing the value T. The time history of lift and drag is examined from time to time to check if a fully periodic solution (if any) is reached before computations are terminated. The computation for $Re = 300$ requires approximately $25\,000$ time steps to obtain a fully periodic solution from the state of rest. This takes about 2 days and 13 hours of CPU time to run on a personal workstation. To achieve a similar periodic flow for $Re = 1\,000$, a total of about $80\,000$ time steps are required which takes about 8 days of CPU time from the state of rest. This is expected since the flow at $Re = 1\,000$ is less stable and larger CPU time is required to capture the fully periodic flow.

Figure 2 illustrates the streamlines for 2D flow around the cylinder at $Re = 300$ and $1\,000$ over one period of the lift coefficient (or $1/Str$, where Str is the Strouhal number of the flow separation behind the cylinder). By comparing the plots of streamlines between the first (first top) and the fifth (last bottom) figures—for the respective Reynolds number, it can be verified that both flows achieve fully periodic solution in time.

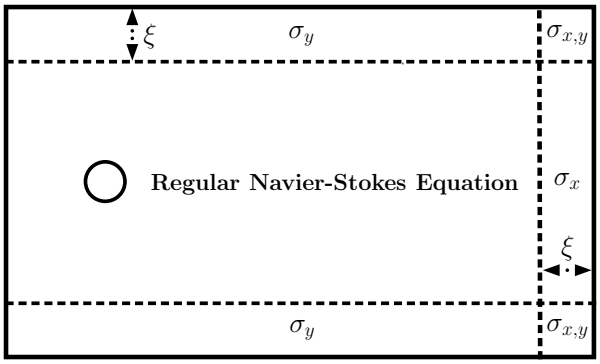

Figure 1. Computational domain with the layer for the directional implicit damping.

The dynamics of the shear layer at the back of the 2D cylinder is more complex at $Re = 1\,000$ than at $Re = 300$. As expected, increasing the Reynolds number induces more and finer vortices, requiring finer meshes to maintain up 6–8 nodes with $\mathbb{P}_2$-$\mathbb{P}_1$ elements across the boundary layer (see Figure 3).

The information on the lift and drag coefficients is summarized in Table III. The Strouhal number for the flows at $Re = 300$ and $1\,000$ are 0.2226 and 0.2491, respectively. For both Reynolds numbers, the resulting flows around the circular cylinder produce a space-time symmetric vortex shedding behind the cylinder (not shown). Therefore, the mean lift coefficient is expected to be zero for both $Re = 300$ and $1\,000$, which was reproduced at order 10^{-4} in our computations. For both cases, the fully time-periodic vortex shedding behind the cylinder; i.e., $\mathbf{u}(5, y, t) = \mathbf{u}\big(5, y, t + \frac{\varphi}{2}\big)$ is demonstrated where φ is the period of the flow. Further, the oscillations on lift have a frequency that is twice as small as that for drag. This is also observed for the flow past a circular cylinder at $Re = 100$ (see [20]) which suggests that the unsteady periodic flows at $Re = 300$ and $1\,000$ stay within the laminar regime.

As a matter of fact, computed 2D flows around the cylinder at larger Reynolds number (typically $Re > 200$) falls short to produce reasonable drag and lift coefficients since flows around cylinder exhibit significant 3D features for such Reynolds number [30]. The computed lift and drag coefficients for $Re = 1\,000$ cannot be compared due to this shortcoming. However, we showcase

DOI: 10.1002/fld

that our method is very stable when computing highly nonlinear flows using time step $\tau = 0.001$, where very small time steps are typically required; e.g., $\tau < 0.001$ for direct numerical simulation (DNS) computations to capture all possible turbulent scales present in the flow. We tested that the same time step $\tau = 0.001$ allows stable computations for a 2D flow around the circular cylinder at $Re = 3\,900$ (not shown here). In addition, we demonstrate that the directional implicit damping method works reasonably well to allow vortices to travel away from the domain without causing any numerical reflection. Directional implicit damping schemes can be implemented for various flow problems involving high Reynolds numbers; e.g., external flows, open channel and pipe flows, etc.

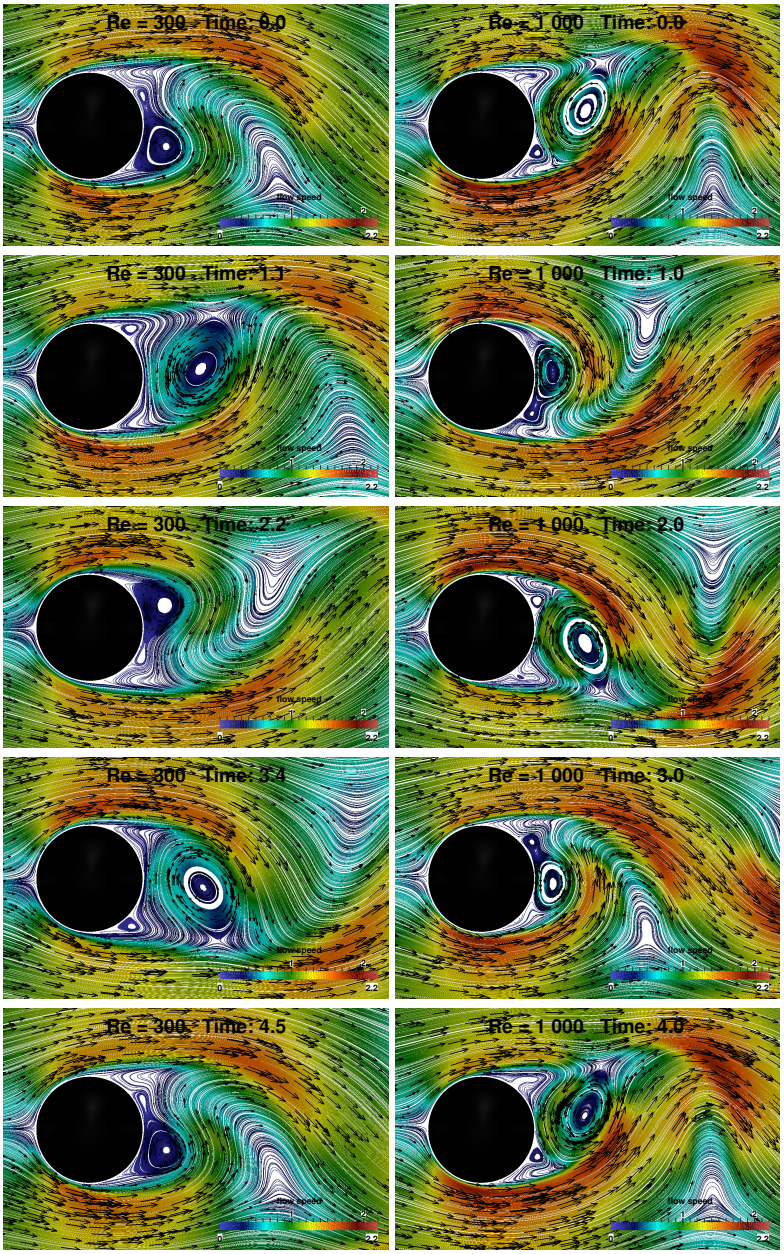

Figure 2. 2D flow around the cylinder at $Re = 300$ (left column) and $Re = 1\,000$ (right column): Snapshots of streamlines with flow speed (color-coded) plotted at every quarter period over one period of the flow; i.e., 4.4917 time units for $Re = 300$ and 4.0135 time units for $Re = 1\,000$.

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

Table III. The mean, amplitude and frequency of the lift and drag coefficients for 2D flow around the cylinder at $Re = 300$ and $1\,000$.

Re = 300						
Source of results	c_l **mean**	c_l **amp.**	c_l **freq.**	c_d **mean**	c_d **amp.**	c_d **freq.**
Present	-5.1086×10^{-4}	0.9987	0.2226	1.4099	8.7317×10^{-2}	0.4453
Mittal & Balachandar [22]	–	–	0.2150	1.3667	–	–
Ranjani et al. [31]	–	–	0.2130	1.3800	–	–

Re = 1 000						
Source of results	c_l **mean**	c_l **amp.**	c_l **freq.**	c_d **mean**	c_d **amp.**	c_d **freq.**
Present	6.6455×10^{-4}	1.5709	0.2492	1.5548	0.2501	0.4983

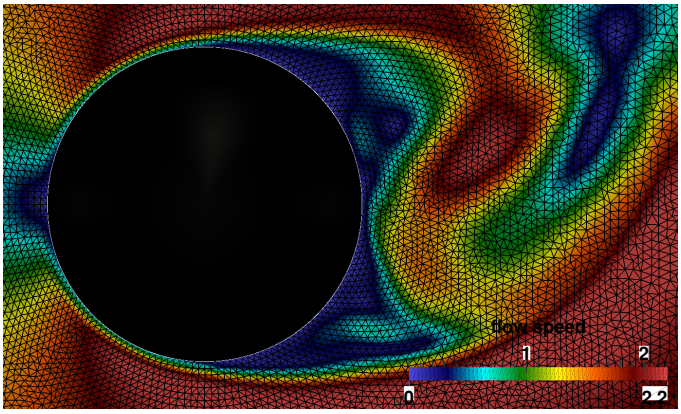

Figure 3. Zoom of the mesh for the flow past a circular cylinder at $Re = 1\,000$.

5. 2D LID-DRIVEN CAVITY FLOW AT $Re = 50\,000$ AND $100\,000$

At $Re > 10\,000$, 2D lid-driven cavity exhibits a transition from laminar to turbulent flow. We demonstrate that $\mathbb{P}_2$-$\mathbb{P}_1$ elements combining high-order time-stepping semi-implicit methods with **grad**-div stabilization term are both accurate and stable when computing flows at such high Reynolds numbers. We use a 2D setting which is ideal for carrying tests in terms of efficiency and simplicity. It is undeniable that 2D lid-driven cavity flow at such high Reynolds number is fictitious. From an application point of view, however, 2D settings may still be relevant when studying driven cavity flows possibly occurring in a very fine lateral gap.

The numerical computation of the lid-driven cavity with $Re = 100\,000$ was attempted by Bruneau and Saad [6] using 3^{rd}-order finite difference scheme (with staggered grid) for space approximation while both 1^{st}- and 2^{nd}-order semi-implicit BDF methods were used for time-stepping. The finest grid resolution used in their computation is 2048×2048 cells. The time step that they have used for the numerical experiment are not reported. We conjectured that the time step fixed in their computation was very small since the mesh used was finer than ours. With similar semi-implicit approach, maintaining CFL stability condition is more difficult without any stabilization mechanism.

In this section, we present numerical results pertaining to lid-driven flow using two different Reynolds number, $Re = 50\,000$ and $100\,000$. We use the standard square domain $\Omega = (0, 1)^2$. A uniform triangular-shaped elements is generated by first dividing each edge of Ω into 200×200 equal-length elements for $Re = 50\,000$ and 300×300 equal-length mesh for $Re = 100\,000$. For the 200×200 mesh and with $\mathbb{P}_2$-$\mathbb{P}_1$ elements, the triangulation produces a varying mesh size h_K where $0.003\,792 \leq h_K \leq 0.009\,868$ and a total of $94\,910$ nodes, which gives $381\,242$ and $47\,856$ d.o.f. for velocity and pressure, respectively. For the 300×300 mesh, the unstructured triangulation contains $147\,650$ elements with varying mesh size h_K where $0.003\,002 \leq h_K \leq 0.008\,083$, and $592\,602$ and $74\,326$ d.o.f. for velocity and pressure, respectively. With the two different meshes, we ensure that

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

any boundary layer can be discretized in the crosswise direction by at least 5–10 nodes, especially for the velocity at such high Reynolds number.

Dirichlet boundary conditions only are defined with a constant velocity prescribed as follows: $\mathbf{u}|_{\Gamma_{\text{lid}}} = (1, 0)$ along the lid $\{(x, 1) \mid 0 \leq x \leq 1\}$ and $\mathbf{u}|_{\Gamma_{\text{wall}}} = (0, 0)$ along the remaining walls. In some articles, these boundary conditions are known as the "leaky cavity", which produce less oscillations at the top left corner and are suitable to handle driven flow with very high Reynolds number. However, for smaller Reynold numbers, (e.g., $Re < 50000$) we use $\mathbf{u}|_{\Gamma_{\text{lid}}} = (1, 0)$ along the lid $\{(x, 1) \mid 0 < x < 1\}$ and $\mathbf{u}|_{\Gamma_{\text{wall}}} = (0, 0)$ along the remaining walls, called as the "watertight cavity". We observed that the latter setting can reproduce very accurately the Strouhal number of the flow for lid-driven flow at $Re = 8500$ [20]. For initial condition, we prescribed the state of rest, i.e., $\mathbf{u} = (0, 0)$ in the domain Ω. SBDFB-3 method is used to ensure a high-order accuracy in time without the need for a very small time step.

Due to the (nearly) turbulent state of the flows at such high Reynolds number, the numerical stability related to the CFL condition is harder to achieve. We observed that instability could occur unpredictably at some large t if the time step τ is not chosen fine enough. By picking a very fine time step, stability is always guaranteed but at the expense of making the computation inefficient. To avoid restrictions on the time step (e.g., $\tau < 0.0008$) due to CFL criteria, the numerical stability can be improved by choosing a larger stabilization parameter (e.g., $\lambda > 10^3$). However, the larger λ, the more numerical diffusion occurs. This is what we would like to avoid altogether. At this moment, we are not able to quantify the mathematical relationship between λ and the resulting numerical diffusion in SBDF methods (with $\mathbf{grad}$-div stabilization term).

The mesh used in the current computation is nearly three times finer than the mesh used for $Re = 8\,500$ in [20] while the Reynolds number $100\,000$ is about 12 times larger. The approximated critical time step for stability of the SBDFB-3 method with $\lambda = 1$ for the lid-driven cavity flow at $Re = 8\,500$ on a 120×120 mesh is about $0.002\,343$. For this flow experiment, the 300×300 mesh used is 2.5 times finer than the 120×120 mesh. Both Reynolds numbers $Re = 50\,000$ and $100\,000$ are about 6 and 12 times larger than $Re = 8500$, respectively. The combined effect of finer mesh and higher Re would definitely impose more serious restriction on the time step needed to achieve stability. However, by fixing the stabilization parameter at $\lambda = 10^3$, we observe that the SBDFB-3 method remains stable with time step $\tau = 0.001$ at $Re = 50\,000$. For $Re = 100\,000$, the time step $\tau = 0.000\,8$ produces a stable solution as well. If both time steps are slightly increased, numerical instabilities occurs due to the CFL condition.

Figure 4 shows snapshots of vorticity magnitude, plotted at $t = 1$, 2, 4, 8, 16, 32, 64, 128 and 256 time units. The flow is composed of several rotating and counter-rotating vortices. The largest vortex that formed at the center of the cavity for $Re = 50\,000$ does not exhibit a fixed circular structure at the center of the cavity as it is observed for the lid-driven cavity with a Reynolds number up to $30\,000$ [11].

Shear layer instabilities are found to be more predominant along the wall induced by the sharp boundary layers. For instance, along the lower right wall, the boundary layer produces many interesting flow instabilities in terms of vortex sheddings. This further generates several intermingling smaller vortices travelling in a spiraling motion before "collapsing" into the main vortex at the center. The main circulation also induces the generation of several other secondary vortices near the bottom and upper left corner of the domain.

Figure 5 shows the time history of the x- and y-velocity and pressure monitored at four designated points located near the corners. The time-evolution of all variables is chaotic with time intervals showing mild variation separated by smaller intervals with sharper gradients at about $t = 407$, 425, 435 and 440. For these values of t, sharper gradient or larger amplitude variation are syncronized among variables. These correspond to the passage of vortices through the sampling point. A similar chaotic behaviour is observed for the velocity and pressure at $Re = 100\,000$ (not shown).

Figure 6 illustrates the evolution of the vorticity contours for the 2D lid-driven cavity flow at $Re = 100\,000$ plotted for $t = 4$, 6, 10, 14, 16 and 20. We compare our results with the one computed by Bruneau and Saad [6] in a qualitative manner. Our vorticity contours show the formation of complex vortical patterns. We also notice that boundary and shear layers can be

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

resolved satisfactorily using our a 300×300 mesh (with $\mathbb{P}_2$-$\mathbb{P}_1$ elements). During start up, the boundary layer and vortices growing along the right wall is somewhat similar to those seen on Figure 3 of [6]. The evolution of vorticity demonstrates that the flow in our computations progresses faster by about 4–5 time units than the results from [6]. From the state of rest, our computed flow fills up the space of cavity (spanning the x-direction) at about $16 < t < 20$. However, in [6], similar flow only fills up about 70%–80% of the cavity (spanning the x-direction) at $t = 20$. This may result from the slightly coarser mesh that we used. Our space approximation using a 300×300 mesh with $\mathbb{P}_2$-$\mathbb{P}_1$ elements corresponds to having a 600×600 mesh using finite difference schemes, compared to their $2\,048 \times 2\,048$ mesh.

A sharp boundary layer is generated consistently along the lid and right cavity wall, while intermittently along the bottom and left walls. These features can only be resolved with a reasonably fine mesh (see Figure 7).

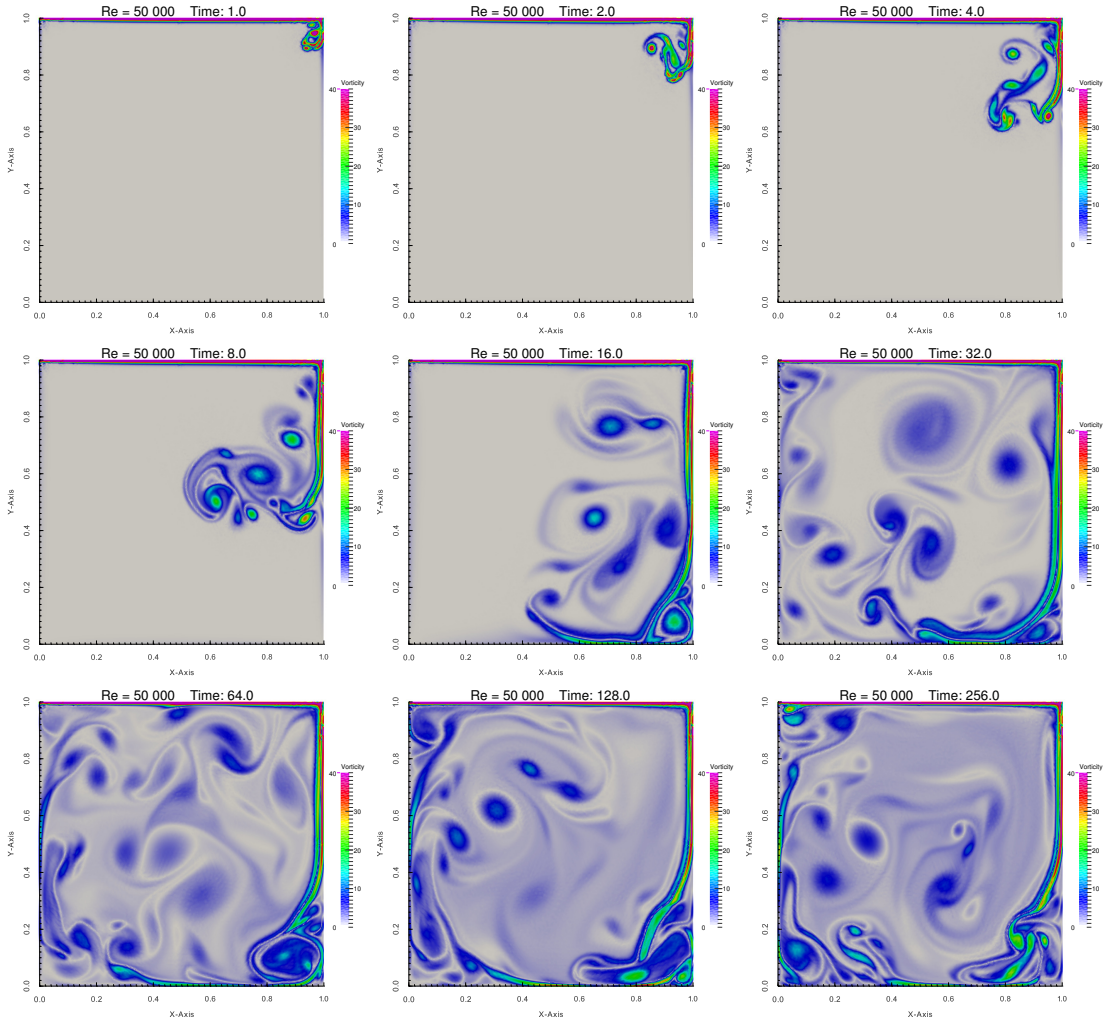

Figure 4. Vorticity for 2D lid-driven cavity flow with $Re = 50\,000$ at $t = 1$, 2, 4, 8, 16, 32, 64, 128 and 256 showcasing the time evolution of the flow from start-up to partially developed flow.

On Figure 8, we present the evolution of streamlines, vorticity and pressure contours for 2D lid driven cavity flow with $Re = 100\,000$ plotted at $t =$ 10, 100, 200 and 300. As seen from the streamlines, once the flow is established, there is a main vortex with multiple neighboring smaller vortices intermingling with each other. The eye of the primary vortex is rarely fixed at one location but rather in a rotating motion within the domain with several other vortices intermittently

DOI: 10.1002/fld

"absorbed" into a major driven circulation. Meanwhile, secondary vortices are formed near the upper left, bottom left and bottom right corners with irregular shapes, giving rise to flow separations and creating smaller vortices. Once the flow is well-developed; e.g., for $t > 50$, no vortex can be found near the upper right corner. This observation agrees with the qualitative results in [6]. The vorticity contours show that vortex shedding is occurring at the end tip of the boundary layer formed along the right wall. The isolines of pressure indicate mostly the troughs in syncronisation with the center of the generated vortices. We also observed two points of near singularity (one positive and one negative pressure) occuring at both upper corners, as expected for 2D lid-driven cavity flows. Again, these observations agree qualitatively well with the numerical results in [6].

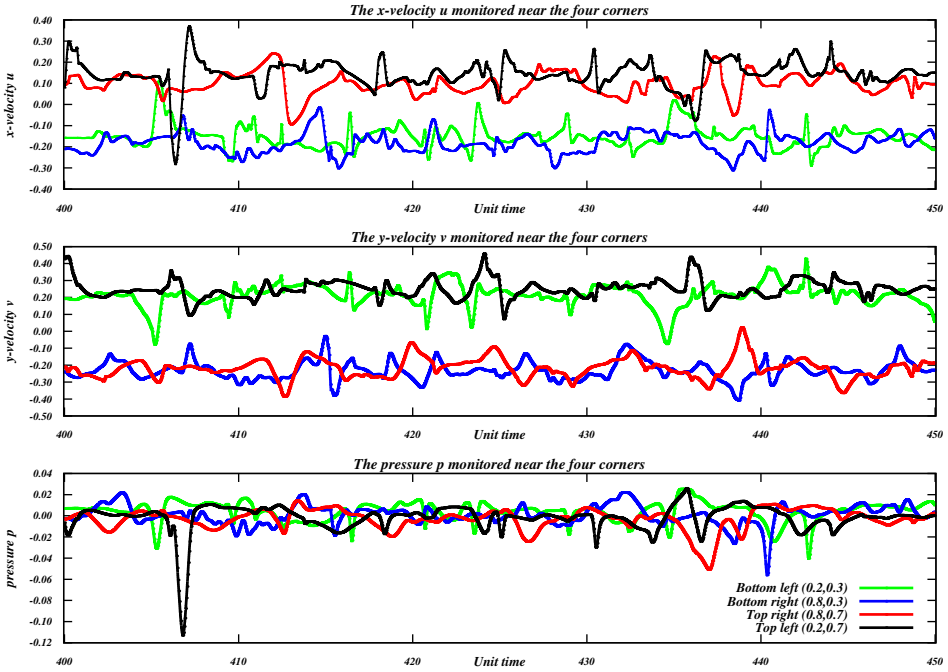

Figure 5. 2D lid-driven cavity at $Re = 50\,000$: Time evolution of x-,y-velocity and pressure monitored near the four corners, bottom left $(0.2, 0.3)$, bottom right $(0.8, 0.3)$, top right $(0.8, 0.7)$ and top left $(0.2, 0.7)$, for $t \in [400, 450]$.

6. CONCLUSIONS AND DISCUSSIONS

In this paper, the ratio of the truncation errors on velocity between SBDF and SBDFB methods for 2^{nd}- and 3^{rd}-order of accuracy were studied using a simple mathematical analysis (Taylor-series expansion). Similar analysis of truncation error could not be established for pressure at this moment. We found that the error on velocity and pressure produced by SBDFBEx methods and BDF methods are almost identical for a given order of accuracy.

We observed that the $\mathbf{grad}$-div term produces better numerical stability for high-order semi-implicit methods (i.e., SBDF methods). The $\mathbf{grad}$-div stabilization term is the simplest to implement as compared to other stabilization strategies used in finite element methods, e.g. streamline upwind Petrov-Galerkin (SUPG), pressure stabilized Petrov-Galerkin (PSPG) and Galerkin least square (GALS) methods.

When computing high Reynolds flows, the time step for our methods can be larger than that for the same method without $\mathbf{grad}$-div stabilization term. However, the choice of the stabilization parameter $\lambda > 0$ is still an open problem. We approximate this value only by resorting to direct computations involving many trials and errors.

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

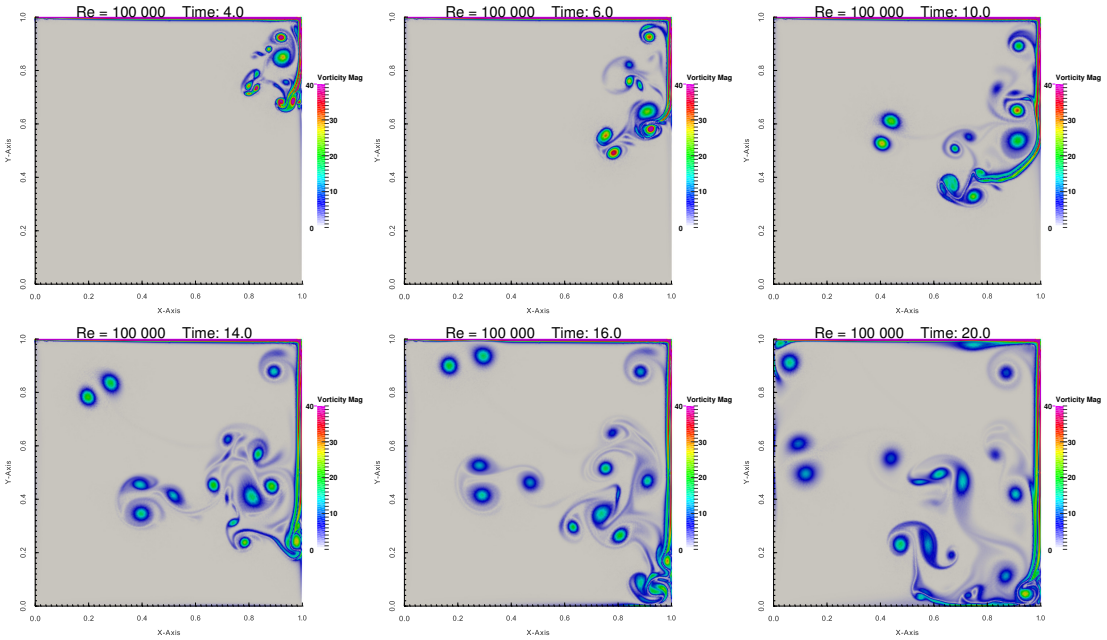

Figure 6. Vorticity contours for the 2D lid-driven cavity flow at $Re = 100\,000$ and times $t = 4$, 6, 10, 14, 16 and 20 from state of rest to partially developed flow.

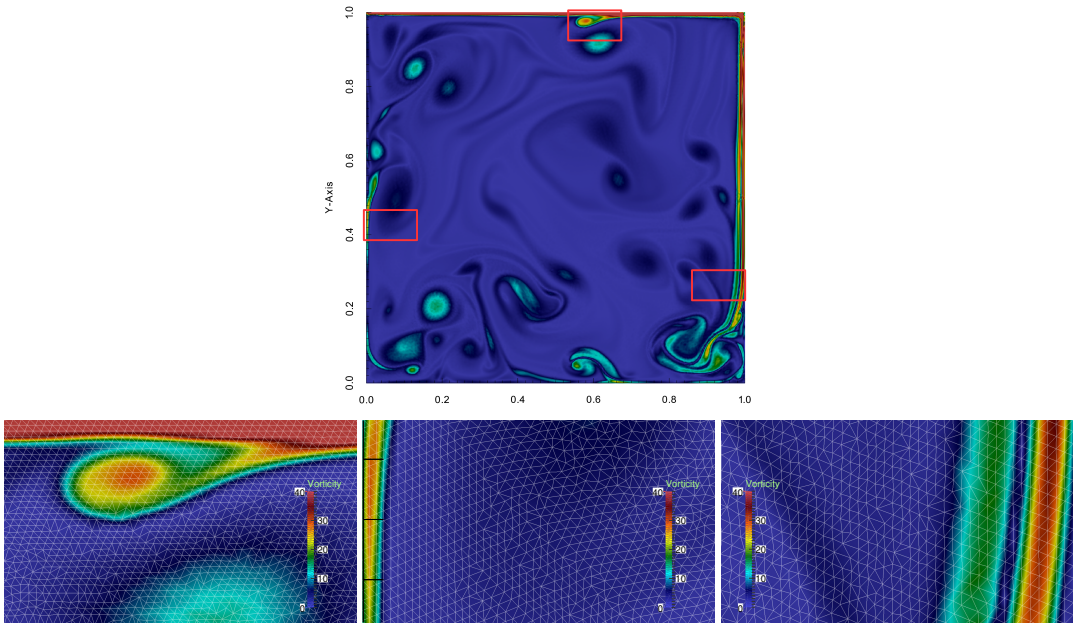

Figure 7. A blow-up of the mesh used to compute the 2D lid-driven cavity flow at $Re = 100\,000$ with the magnitude of vorticity at $t = 78.5$ time units: Full view (top), The top lid (bottom left), the left boundary (bottom center) and the right boundary (bottom right).

Our numerical results for the flow past the cylinder at $Re = 300$ is very close to the value in the literature. The numerical results for $Re = 1\,000$ cannot be compared due to the lack of similar work in 2D around the cylinder. For the lid-driven flow, our results reproduce those from [6] qualitatively with a time lag since the mesh used in our computations is not as fine as theirs. Nonetheless, our numerical results show that the SBDFB-3 method used is stable and very accurate without

DOI: 10.1002/fld

depending on very small time step. This indicates a great potential for SBDFB methods to resolve turbulent flows by direct numerical simulations (without closure models).

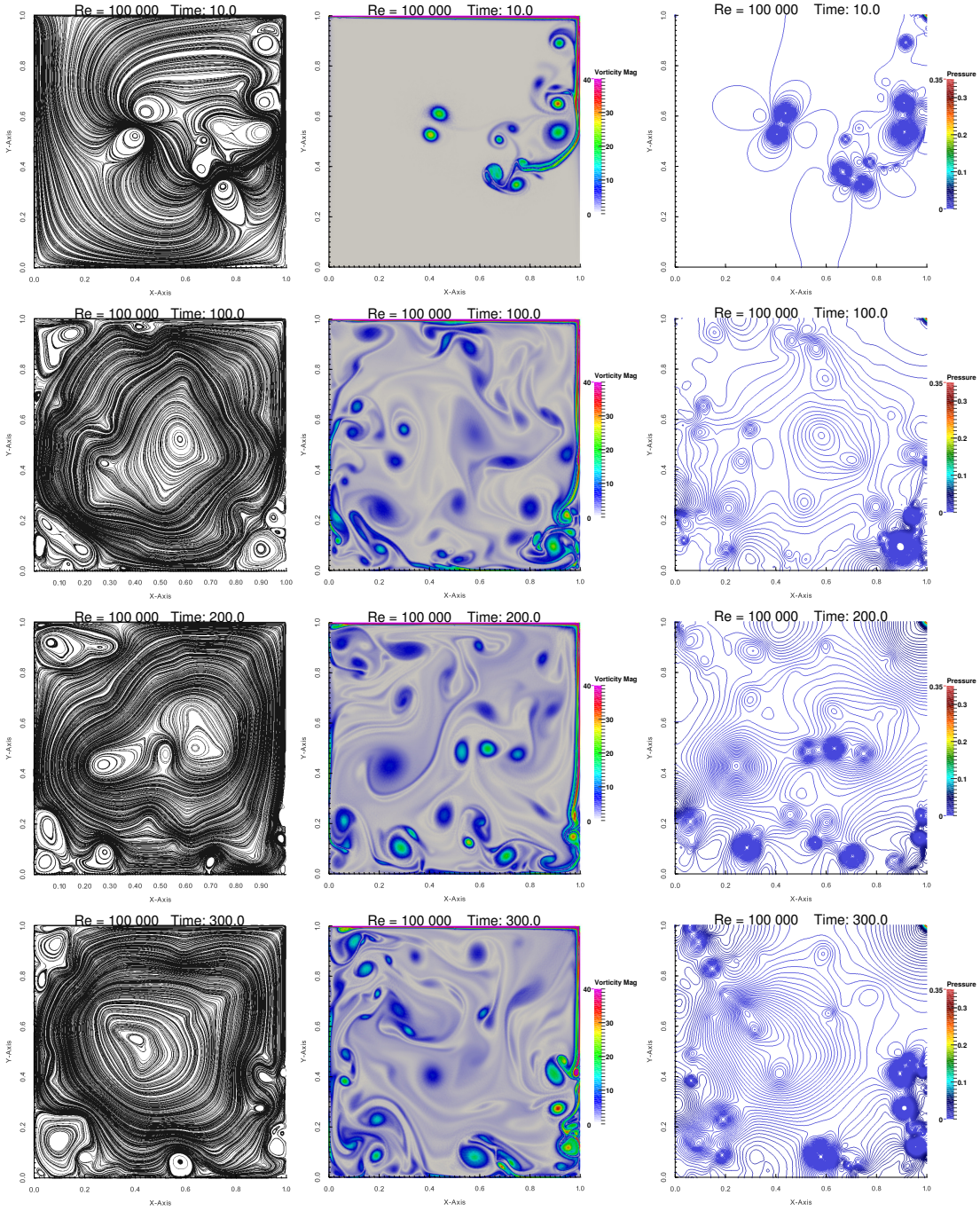

Figure 8. Snapshots of streamlines (left column), vorticity (middle column) and pressure (right column) for 2D lid-driven cavity flow at $Re = 100\,000$ at $t = 10$, 100, 200, 300 (from top to bottom).

7. ACKNOWLEDGEMENTS

We would like to acknowledge the NSERC for its financial support to conduct this research. The first author, Kak Choon, Loy would like to express his deepest gratitute to the Universiti Malaysia

Int. J. Numer. Meth. Fluids (2017)
DOI: 10.1002/fld

Terengganu and the Ministry of Education of Malaysia for a scholarship awarded during his studies for the PhD at the University of Ottawa. We also thank the FreeFEM++ development team leaded by Professor Frédéric Hecht at the Laboratoire Jacques–Louis Lions, Université Pierre et Marie Curie, for making their software available as an open source. We are indebted to our colleague Dr. Benoit Dionne for his great contribution to provide valuable comments and to proof-read this manuscript.

REFERENCES

1. Akrivis G, Crouzeix M, Makridakis C. Implicit-explicit multistep finite element methods for nonlinear parabolic problems. *Mathematics of Computation*. 1998;67(222):457–477.
2. Ascher UM, Ruuth SJ, Spiteri RJ. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*. 1997;25(2-3):151–167.
3. Ascher UM, Ruuth SJ, Wetton BTR. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal of Numerical Analysis*. 1995;32(3):797–823.
4. Brezzi F, Fortin M. *Mixed and Hybrid Finite Element Methods (Vol. 15)*. Springer Science & Business Media; 2012.
5. Baek H, Karniadakis GE. Sub-iteration leads to accuracy and stability enhancements of semi-implicit schemes for the Navier-Stokes equations. *Journal of Computational Physics*. 2011;230(12):4384–4402.
6. Bruneau C, Saad M. The behaviour of high Reynolds flows in a driven cavity. *Computational Fluid Dynamics Journal*. 2006; 15(3):303(1–17).
7. Baker GA, Dougalis VA, Karakashian OA. On a higher order accurate fully discrete Galerkin approximation to the Navier–Stokes equations. *Mathematics of Computation*. 1982;39(160):339–375.
8. Case MA, Ervin VJ, Linke A, Rebholz LG. A connection between Scott–Vogelius and grad-div stabilized Taylor–Hood FE approximations of the Navier-Stokes equations. *SIAM Journal of Numerical Analysis*. 2011; 49(4):1461–1481.
9. Ethier M, Bourgault Y. Semi-implicit time-discretization schemes for the bidomain model. *SIAM Journal of Numerical Analysis*. 2008;46(5):2443–2468.
10. Guermond J-L, Minev P, Shen J. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*. 2006;195(44):6011-6045.
11. Hachem E, Rivaux B, Kloczko T, Digonnet H, Coupez T. Stabilized finite element method for incompressible flows with high Reynolds number. *Journal of Computational Physics*. 2010;229(23):8643–8665.
12. Hou Y, Liu Q. A stabilized semi-implicit Galerkin scheme for Navier-Stokes equations. *Journal of Computational and Applied Mathematics*. 2009;231(2):552–560.
13. He Y, Sun W. Stability and convergence of the Crank-Nicolson/Adams-Bashforth scheme for the time-dependent Navier-Stokes equations. *SIAM Journal on Numerical Analysis*. 2007;45(2):837–869.
14. Jiang N. A second-order ensemble method based on a blended backward differentiation formula timestepping scheme for time-dependent Navier–Stokes equations. *Numerical Methods for Partial Differential Equations*. 2016; 33(1):34–61.
15. John V. Reference values for drag and lift of a two-dimensional time-dependent flow around a cylinder. *International Journal for Numerical Methods in Fluids*. 2004;44:777–788.
16. John V, Matthies G, Rang J. A comparison of time-discretization/linearization approaches for the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*. 2006;195(44):5995–6010.
17. Jenkins EW, John V, Linke A, Rebholz LG. On the parameter choice in grad-div stabilization for the Stokes equations. *Advanced Computer and Mathematical Sciences*. 2014;40:491–516.
18. Kress W, Lötstedt P. Time step restrictions using semi-implicit methods for the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*. 2006;195:4433–4447.
19. Löwe J. A locally adapting parameter design for the divergence stabilization of FEM discretizations of the Navier–Stokes equations. BAIL 2008–Boundary and Interior Layers. *Springer Berlin Heidelberg*. 2009; pp. 195–204.
20. Loy KC, Bourgault Y. On efficient high-order semi-implicit schemes for unsteady incompressible Navier–Stokes equations. *Computer and Fluids*. 2017;148:166–184.
21. Linke A, Rebholz LG, Wilson NE. On the convergence rate of grad-div stabilized Taylor–Hood to Scott–Vogelius solutions for incompressible flow problems. *Journal of Mathematical Analysis and Applications*. 2011;381(2):612–626.
22. Mittal R, Balachandar S. Generation of streamwise vortical structures in bluff body wakes. *Physical Review Letters*. 1995;75(7):1300–1330.
23. Olshanskii MA, Lube G, Heister T, Löwe J. Grad-div stabilization and subgrid pressure models for the incompressible Navier–Stokes equations. *Computer Methods and Applied Mechanical Engineering*. 2009;198:3975–3988.
24. Olshanskii MA, Reusken A. Grad-div stabilization for Stokes equations. *Mathematics of Computation*. 2003;73(248):1699–1718.
25. Østerby O. Five ways of reducing the Crank–Nicolson oscillations. *BIT Numerical Mathematics*. 2003;43(4):811–822.
26. Pironneau O. On the transport-diffusion algorithm and its applications to the Navier-Stokes equations. *Numerische Mathematik*. 1982;38(3):309–332.
27. Pironneau O, Liou J, Tezduyar T. Characteristic-Galerkin and Galerkin/least-squares space-time formulations for the advection-diffusion equation with time-dependent domains. *Computer Methods in Applied Mechanics and Engineering*. 1992;100(1):117–141.
28. Quarteroni A, Saleri F, Veneziani A. Factorization methods for the numerical approximation of Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*. 2000;188(1):505–526.

DOI: 10.1002/fld

29. Söderström A, Karlsson M, Museth K. A PML-based nonreflective boundary for free surface fluid animation. *ACM Transactions on Graphics (TOG)*. 2010;29(5):136(1–17).
30. Singh SP, Mittal S. Flow past a cylinder: Shear layer instability and drag crisis. *International Journal for Numerical Methods in Fluids*. 2005;47(1):75–98.
31. Rajani BN, Kandasamy A, Majumdar S. Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modeling*. 2009;33:1228–1247.
32. Ravindran SS. An extrapolated second order backward difference time-stepping scheme for the magnetohydrodynamics system. *Numerical Functional Analysis and Optimization*. 2016;37(8):990–1020.
33. Ruuth SJ. Implicit-explicit methods for reaction-diffusion problems in pattern formation. *Journal of Mathematical Biology*. 1995;34:148–176.
34. Smith A, Silvester D. Implicit algorithms and their linearization for the transient incompressible Navier-Stokes equations. *IMA Journal of Numerical Analysis*. 1997;17(4):527–545.
35. Temam R. *Navier–Stokes equations: Theory and numerical analysis*. Providence, Rhode Island. AMS Chelsea Publishing; 2001.
36. Turek S. A comparative study of time-stepping techniques for the incompressible Navier–Stokes equations: From fully implicit non-linear schemes to semi-implicit projection methods. *International Journal for Numerical Methods in Fluids*. 1996;22:987–1011.
37. Wang K, Chao L. Third-order temporal discrete scheme for the non-stationary Navier–Stokes equations. *International Journal of Computer Mathematics*. 2012;89(15):1996-2018.
38. Wang K, He Y. Convergence analysis for a higher order scheme for the time-dependent Navier–Stokes equations. *Applied Mathematics and Computation*. 2012;128:8269–8278.
39. Zang TA. On the rotation and skew-symmetric forms for incompressible flow simulations. *Applied Numerical Mathematics*. 1991;7(1):27–40.
40. Zhang T, Jin J, Xu S. The Euler implicit/explicit scheme for the Boussinesq equations. *Boundary Value Problems*. 2016;1:181.

DOI: 10.1002/fld

Chapter 4

Several Numerical Improvements to High-Order Artificial Compressibility Methods

To avoid solving a saddle point problem, ALM (see Section 1.3.1) can be implemented with any semi-implicit time-stepping methods (e.g., SBDF, CNAB, DC methods, etc). An ad-hoc implementation of ALM in the SBDF-2 method produces the following scheme: Given an initial value $(\mathbf{u}^0, p^0) = (\mathbf{u}(0), 0)$ and a proper initialization for $\mathbf{u}^1$, we compute $(\mathbf{u}^{n+2}, p^{n+2})$ for $n = 0, 1, \ldots, N-2$ by solving for $(\mathbf{u}^{n+2,s+1}, p^{n+2,s+1})$ the following system

$$\frac{3\mathbf{u}^{n+2,s+1} - 4\mathbf{u}^{n+1} + \mathbf{u}^n}{2\tau} - \nu\Delta\mathbf{u}^{n+2,s+1} \tag{4.0.1}$$

$$\begin{aligned} -\lambda\nabla\nabla\cdot\mathbf{u}^{n+2,s+1} + \nabla p^{n+2,s} &= \mathbf{f}^{n+2} - 2B(\mathbf{u}^{n+1}) + B(\mathbf{u}^n), \quad \text{on } \Omega, \\ p^{n+2,s+1} &= p^{n+2,s} - \lambda\nabla\cdot\mathbf{u}^{n+2,s+1}, \quad \text{on } \Omega. \end{aligned} \tag{4.0.2}$$

At each time step, we set $p^{n+2,0}$ to a properly chosen value and iterate on $(\mathbf{u}^{n+2,s+1}, p^{n+2,s+1})$ for $s \in \mathbb{N}$. Reduced linear systems taking the form of a parabolic problem (4.0.2) and a pressure update (4.0.2) have to be solved until $\|p^{n+2,s+1} - p^{n+2,s}\|_{L^2(\Omega)} \leq tol$, where tol is the chosen small tolerance. Once the stopping criterion is met for some s, we set $\mathbf{u}^{n+2} := \mathbf{u}^{n+2,s+1}$ and $p^{n+2} := p^{n+2,s+1}$.

Remark 4.0.1. A proper initialization $p^{n+1,0} := p^n$ for all $1 \leq n \leq N-1$ can be implemented to reduce the number of ALM iterations required to reach the stopping criterion.

Any higher-order SBDF methods with ALM can be constructed in an analogous

manner. The solution of the above scheme is equivalent to SBDF-2 method with an additional **grad**-div term in the momentum equation, as shown in Chapter 3. Due to the need to do many ALM iterations, the efficiency of this method is not appealing to compute unsteady flows.

In Chapter 2, we proposed and implemented two methods, namely GM and GM-SRM, which are constructed based on higher-order artificial compressibility techniques and bootstrapping strategy. In fact, these two methods have undergone numerical improvements in terms of nonlinear interpolation formulae without providing a detailed justification. In this chapter, we shall look into these numerical improvements more closely. We analyze the convergence and stability of GM and GM-SRM methods, and propose ways to improve the accuracy and efficiency of these methods at any given order.

4.0.1 The Guermond–Minev methods

Before analyzing the numerical improvement in details, we recall the 3$^{\text{rd}}$-order GM method proposed by Guermond and Minev [23]: Given the initial condition $(\mathbf{u}_0^0, p_0^0) = (\mathbf{u}(\mathbf{0}), p(0))$ and setting $(\mathbf{u}_1^0, p_1^0) = (0,0)$, $(\mathbf{u}_2^0, p_2^0) = (0,0)$, we solve for $(\mathbf{u}_0^{n+1}, p_0^{n+1})$, $(\mathbf{u}_1^n, p_1^n)$ and $(\mathbf{u}_2^{n-1}, p_2^{n-1})$ the subproblems (4.0.3), (4.0.4) and (4.0.5), respectively.

$$\text{For } n \geq 0, \begin{cases} \mathbf{nl}_0^{n+1} = B(\mathbf{u}_0^n), \\ \dfrac{\mathbf{u}_0^{n+1} - \mathbf{u}_0^n}{\tau} - \nu\Delta\mathbf{u}_0^{n+1} - \lambda\nabla\nabla\cdot\mathbf{u}_0^{n+1} + \nabla p_0^n = \mathbf{f}^{n+1} - \mathbf{nl}_0^{n+1}, \\ p_0^{n+1} = p_0^n - \lambda\nabla\cdot\mathbf{u}_0^{n+1}, \\ \mathbf{du}_0^{n+1} = (\mathbf{u}_0^{n+1} - \mathbf{u}_0^n)/\tau, \quad dp_0^{n+1} = (p_0^{n+1} - p_0^n)/\tau. \end{cases} \tag{4.0.3}$$

$$\text{For } n \geq 1, \begin{cases} \mathbf{d^2u}_0^{n+1} = (\mathbf{du}_0^{n+1} - \mathbf{du}_0^n)/\tau, \\ \mathbf{nl}_1^n = B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1}), \\ \dfrac{\mathbf{u}_1^n - \mathbf{u}_1^{n-1}}{\tau} - \nu\Delta\mathbf{u}_1^n - \lambda\nabla\nabla\cdot\mathbf{u}_1^n + \nabla(p_1^{n-1} + dp_0^n) \\ \qquad = -\dfrac{1}{2}\mathbf{d^2u}_0^{n+1} - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\tau}, \\ p_1^n = p_1^{n-1} + dp_0^n - \lambda\nabla\cdot\mathbf{u}_1^n, \\ \mathbf{du}_1^n = (\mathbf{u}_1^n - \mathbf{u}_1^{n-1})/\tau, \quad dp_1^n = (p_1^n - p_1^{n-1})/\tau. \end{cases} \tag{4.0.4}$$

$$\text{for } n \geq 2, \begin{cases} \mathbf{d^2 u}_1^n = (\mathbf{d u}_1^n - \mathbf{d u}_1^{n-1})/\tau, \quad \mathbf{d^3 u}_0^{n+1} = (\mathbf{d^2 u}_0^{n+1} - \mathbf{d^2 u}_0^n)/\tau, \\ \mathbf{nl}_2^{n-1} = B(\mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-2}), \\ \dfrac{\mathbf{u}_2^{n-1} - \mathbf{u}_2^{n-2}}{\tau} - \nu \Delta \mathbf{u}_2^{n-1} - \lambda \nabla \nabla \cdot \mathbf{u}_2^{n-1} + \nabla (p_2^{n-2} + dp_1^{n-1}) \\ \qquad = -\dfrac{1}{2} \mathbf{d^2 u}_1^n + \dfrac{1}{6} \mathbf{d^3 u}_0^{n+1} - \dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\tau^2}, \\ p_2^{n-1} = p_2^{n-2} + dp_1^{n-1} - \lambda \nabla \cdot \mathbf{u}_2^{n-1}, \\ \mathbf{u}^{n-1} = \mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}, \quad p^{n-1} = p_0^{n-1} + \tau p_1^{n-1} + \tau^2 p_2^{n-1}. \end{cases} \tag{4.0.5}$$

For the 2nd-order GM method, we refer only to (4.0.3)–(4.0.4) with the asymptotic expansion $\mathbf{u}^{n-1} = \mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1}$ and $p^{n-1} = p_0^{n-1} + \tau p_1^{n-1}$ to respectively update the velocity and pressure, globally at each time step. Higher-order GM methods ($k > 3$) can be constructed by adding subproblems and going further in the asymptotic expansions. As one can see, the only difference found in the GM-3 method above and the GM-3 algorithm mentioned in Chapter 2 is the nonlinear extrapolation formulae $\mathbf{nl}_0^{n+1}$, $\mathbf{nl}_1^n$ and $\mathbf{nl}_2^{n-1}$ used in all subproblems.

GM methods are self-starting. The initialization in GM methods rely on a dependence tree (see Figure 4.1). The dependence tree illustrates the order of evaluation for each of the required velocity and pressure of the subproblems. This also explains how numerical errors can be corrected in a cascade manner, starting from the upper to the lower subproblems using divided differences of high-order derivative terms.

The dependence tree for GM-3 method is shown in Figure 4.1. This tree is identical for DC-3 and GM-SRM-3 method. GM-3 method has a built-in initialization which works in the following manner: At $n = 0$, $(\mathbf{u}_0^1, p_0^1)$ is evaluated using (4.0.3) (red arrow). Then for $n = 1$, $(\mathbf{u}_0^2, p_0^2)$ and $(\mathbf{u}_1^1, p_1^1)$ are evaluated using (4.0.3) and (4.0.4), respectively (green arrow). Finally for $n = 2$, $(\mathbf{u}_0^3, p_0^3)$, $(\mathbf{u}_1^2, p_1^2)$ and $(\mathbf{u}_2^1, p_2^1)$ are computed using (4.0.3), (4.0.4) and (4.0.5), respectively (blue arow). This gives $(\mathbf{u}^1, p^1)$ using the asymptotic expansions found in the last two equations of (4.0.5). The black dashed arrow in Figure 4.1 indicates that the evaluation of a particular variable depends on the variables from upper subproblem as well. For instance, the solution $\mathbf{u}_1^1$ of the 2nd-subproblem depends on the solution $\mathbf{u}_0^2$ of the 1st-subproblem, the solution $\mathbf{u}_2^1$ of the 3rd-subproblem depends on the solution $\mathbf{u}_1^2$ of the 2nd-subproblem, etc.

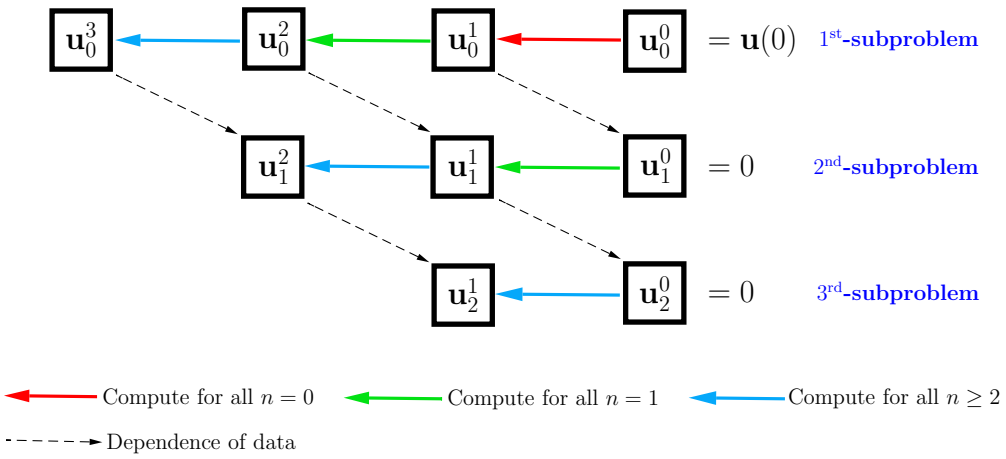

Figure 4.1: Dependence tree for the GM-3 method to compute $\mathbf{u}^1 = \mathbf{u}_0^1 + \tau \mathbf{u}_1^1 + \tau^2 \mathbf{u}_2^1$.

With the defect correction strategy, GM-3 method produces smaller error and is more stable compared to SBDF methods for a given order of accuracy. In GM methods, the same constant matrices (one from the momentum equation and one from the pressure update) are used over and over again to solve all subproblems. This simplifies the algorithm in GM methods, though these are still not as efficient as SBDF methods when computing 2D flows, even when using a direct solver such as MUMPS.

In principle, any method of lines presented in Section 1.5.1 can be combined with defect correction strategy to obtain a scheme with an order of accuracy (in time) that is a multiple of the order of the initial scheme. For instance, we can use any method of lines of m^{th}-order of accuracy (SBDF-m with $m > 0$, CNAB with $m = 2$, etc), and formulate the velocity and pressure using asymptotic expansions of fixed index k (see (8) in Chapter 2). Then, the resulting method has $(mk)^{\text{th}}$-order of accuracy in time. The implementation of such high-order methods could be technically challenging.

4.1 Theoretical Results on GM methods

In this section, we proof the consistency of GM methods, which leads to semi-discrete variational formulation of Navier–Stokes equations with an inclusion of **grad**-div stabilization term. This analysis is achieved assuming the stability of the methods. For simplicity, we give the proof for the 2nd-order GM method, but the consistency of an arbitrary order GM methods can be demonstrated in a similar fashion.

Proposition 4.1.1. *The solution of the* 2^{nd}*-order GM method (4.0.3)-(4.0.4) converges to the solution of*

(a) the semi-discrete (in space) problem for Navier–Stokes equations if the mixed finite element spaces satisfy $\nabla \cdot V_h \subseteq M_h$*;*

(b) a semi-discrete (in space) problem of Navier–Stokes equations with **grad**-div *stabilization term if the mixed element spaces satisfy* $\nabla \cdot V_h \not\subseteq M_h$*;*

with a rate of convergence of $\mathcal{O}(\tau^2)$ *whenever* τ *goes to zero, assuming the mesh is fixed (hence* V_h *and* Q_h *are held fixed).*

Proof: We first denote by $(\mathbf{U}^j, P^j)$ the fully-discrete solution obtained by GM-2 method and $(\mathbf{u}^j, p^j) = (\mathbf{u}_h(t_j), p_h(t_j))$ the semi-discrete solution (in space) at time t_j. Following the convention from the dependence tree in GM-2, we fix $n = j$ and $n = j + 1$ in (4.0.3) and (4.0.4), respectively. Taking the sum of (4.0.3) and $\tau \times$(4.0.4) for any $0 \le j \le N$ gives the following

$$\begin{aligned}
&\frac{1}{\tau}\Big(\mathbf{U}_0^{j+1} + \tau\mathbf{U}_1^{j+1}, \mathbf{v}\Big) + \nu\Big(\nabla(\mathbf{U}_0^{j+1} + \tau\mathbf{U}_1^{j+1}), \nabla\mathbf{v}\Big) \\
&+\lambda\Big(\nabla\cdot(\mathbf{U}_0^{j+1} + \tau\mathbf{U}_1^{j+1}), \nabla\cdot\mathbf{v}\Big) - (P_0^j + \tau P_1^j, \nabla\cdot\mathbf{v}) - (\tau d P_0^{j+1}, \nabla\cdot\mathbf{v}) \\
&= (\mathbf{f}^{j+1}, \mathbf{v}) + \frac{1}{\tau}\Big(\mathbf{U}_0^j + \tau\mathbf{U}_1^j, \mathbf{v}\Big) - \Big(B(\mathbf{U}_0^{j+1} + \tau\mathbf{U}_1^j), \mathbf{v}\Big) - \frac{\tau}{2}(\mathbf{d^2}\mathbf{U}_0^{j+2}, \mathbf{v})
\end{aligned}$$

Here the term $\mathbf{d^2}\mathbf{U}_0^{j+2} = \frac{\mathbf{U}_0^{j+2} - 2\mathbf{U}_0^{j+1} + \mathbf{U}_0^j}{\tau^2}$ is in fact the difference scheme for the second order derivative $\mathbf{u}_0''(t^{j+1})$. Note, that this one is taken at time t^{j+2} instead of times t^{j+1} or t^j as for the other terms in the equation. We group the terms as follows:

$$\begin{aligned}
&\underbrace{\left(\frac{\mathbf{U}^{j+1} - \mathbf{U}^j}{\tau} + \frac{\tau}{2}\mathbf{d^2}\mathbf{U}_0^{j+2}, \mathbf{v}\right)}_{*} + \nu(\nabla\mathbf{U}^{j+1}, \nabla\mathbf{v}) + \underbrace{\Big(B(\mathbf{U}_0^{j+1} + \tau\mathbf{U}_1^j), \mathbf{v}\Big)}_{**} \\
&\qquad + \underbrace{\lambda(\nabla\cdot\mathbf{U}^{j+1}, \nabla\cdot\mathbf{v}) - (P^j + \tau d P_0^{j+1}, \nabla\cdot\mathbf{v})}_{***} = (\mathbf{f}^{j+1}, \mathbf{v}). \quad (4.1.1)
\end{aligned}$$

The simplification is made using the asymptotic expansions, $\mathbf{U}^{j+1} = \mathbf{U}_0^{j+1} + \tau\mathbf{U}_1^{j+1}$ and $P^{j+1} = P_0^{j+1} + \tau P_1^{j+1}$. To determine the truncation error of the term (*), we apply the asymptotic expansion:

$$\begin{aligned}
\left(\frac{\mathbf{U}^{j+1} - \mathbf{U}^j}{\tau} + \frac{\tau}{2}\mathbf{d^2}\mathbf{U}_0^{j+2}, \mathbf{v}\right) &= \left(\frac{\mathbf{U}^{j+1} - \mathbf{U}^j}{\tau} + \frac{\tau}{2}\mathbf{d^2}\big(\mathbf{U}^{j+2} - \tau\mathbf{U}_1^{j+2}\big), \mathbf{v}\right) \\
&= \left(\frac{\mathbf{U}^{j+1} - \mathbf{U}^j}{\tau} + \frac{\tau}{2}\mathbf{d^2}\mathbf{U}^{j+2} - \frac{\tau^2}{2}\mathbf{d^2}\mathbf{U}_1^{j+2}, \mathbf{v}\right).
\end{aligned}$$

Define the smallest upper integer $N = [T/\tau]$ where $T > 0$ is fixed. Assume that we have sufficient regularity of the solution, i.e., $\mathbf{u} \in \mathcal{C}^4([0,T], V_h)$, and that there exists a constant $c = c(T,\Omega)$ such that $\sup_{0<j<N} \|\mathbf{d^2 u}_1^{j+2}\| \leq c\|\mathbf{u}\|_{\mathcal{C}^4([0,T],V_h)}$. Since, $\mathbf{d^2 U}_1^{j+2} = \frac{\mathbf{U}_1^{j+2} - 2\mathbf{U}_1^{j+1} + \mathbf{U}_1^j}{\tau^2}$ and by the Taylor series expansions, we have for the 2^{nd}-order centered finite difference scheme

$$\frac{\mathbf{u}(t_{j+2}) - 2\mathbf{u}(t_{j+1}) + \mathbf{u}(t_j)}{\tau^2} = \frac{\partial^2}{\partial t^2}\mathbf{u}(t_{j+1}) + \frac{\tau^2}{12}\frac{\partial^4}{\partial t^4}\mathbf{u}(t_{j+1}) + \mathcal{O}(\tau^4) \quad (4.1.2)$$

and the 1^{st}-order backward-difference scheme

$$\frac{\mathbf{u}(t_{j+1}) - \mathbf{u}(t_j)}{\tau} = \frac{\partial}{\partial t}\mathbf{u}(t_{j+1}) - \frac{\tau}{2}\frac{\partial^2}{\partial t^2}\mathbf{u}(t_{j+1}) + \frac{\tau^2}{6}\frac{\partial^3}{\partial t^3}\mathbf{u}(t_{j+1}) + \mathcal{O}(\tau^3), \quad (4.1.3)$$

whenever these are applied to the semi-discrete solution $\mathbf{u}$ (in space). To get the truncation error on (*), we substitute $\mathbf{u}^k$ for $\mathbf{U}^k$ and $\mathbf{u}_1^k$ for $\mathbf{U}_1^k$ in (*) to get

$$\begin{aligned}(*) &= \left(\frac{\mathbf{u}^{j+1} - \mathbf{u}^j}{\tau} + \frac{\tau}{2}\left(\frac{\mathbf{u}^{j+2} - 2\mathbf{u}^{j+1} + \mathbf{u}^j}{\tau^2}\right) - \frac{\tau^2}{2}\left(\frac{\mathbf{u}_1^{j+2} - 2\mathbf{u}_1^{j+1} + \mathbf{u}_1^j}{\tau^2}\right), \mathbf{v}\right) \\ &= \left(\frac{\partial}{\partial t}\mathbf{u}^{j+1} + \frac{\tau^2}{6}\frac{\partial^3}{\partial t^3}\mathbf{u}^{j+1} + \frac{\tau^2}{2}\frac{\partial^2}{\partial t^2}\mathbf{u}_1^{j+1}, \mathbf{v}\right) + \mathcal{O}(\tau^3)\end{aligned}$$

Therefore, (*) admits a 2^{nd}-order truncation error in time.

We next get the truncation error of the nonlinear term (**). Recalling that $B(\mathbf{u}) = \mathbf{u} \cdot \nabla\mathbf{u}$, we substitute $\mathbf{u}_0^k$ for $\mathbf{U}_0^k$ and $\mathbf{u}_1^k$ for $\mathbf{U}_1^k$ to get

$$\begin{aligned}(**) &= (B(\mathbf{u}_0^{j+1} + \tau\mathbf{u}_1^j), \mathbf{v}) \\ &= (B(\mathbf{u}^{j+1} - \tau(\mathbf{u}_1^{j+1} - \mathbf{u}_1^j)), \mathbf{v}) \\ &= (B(\mathbf{u}^{j+1}), \mathbf{v}) - \tau((\mathbf{u}_1^{j+1} - \mathbf{u}_1^j) \cdot \nabla\mathbf{u}^{j+1}, \mathbf{v}) - \\ &\quad \tau(\mathbf{u}^{j+1} \cdot \nabla(\mathbf{u}_1^{j+1} - \mathbf{u}_1^j), \mathbf{v}) + \tau^2(B(\mathbf{u}_1^{j+1} - \mathbf{u}_1^j), \mathbf{v}) \\ &= (B(\mathbf{u}^{j+1}), \mathbf{v}) - \tau^2\Big(\Big(\frac{\partial}{\partial t}\mathbf{u}_1^{j+1} - \frac{\tau}{2}\frac{\partial^2}{\partial t^2}\mathbf{u}_1^{j+1}\Big) \cdot \nabla\mathbf{u}^{j+1}, \mathbf{v}\Big) \\ &\quad -\tau^2\Big(\mathbf{u}^{j+1} \cdot \nabla\Big(\frac{\partial}{\partial t}\mathbf{u}_1^{j+1} - \frac{\tau}{2}\frac{\partial^2}{\partial t^2}\mathbf{u}_1^{j+1}\Big), \mathbf{v}\Big) + \mathcal{O}(\tau^4),\end{aligned}$$

where the last line is deduced from (4.1.3). We deduced that (**) admits a 2^{nd}-order truncation error in time.

Before studying the truncation error of the term (***), we first obtain the truncation error resulting from the pressure term p in the continuity equation. Similar to what was done for the momentum equation, the sum of the continuity equations $(4.0.3)|_{n=j}$ $+ \tau(4.0.4)|_{n=j+1}$ produces the following:

$$\Big(P_0^{j+1} + \tau P_1^{j+1}, q\Big) = \Big(P_0^j + \tau P_1^j + \tau dP_0^{j+1} + \lambda\nabla \cdot \big(\mathbf{U}_0^{j+1} + \tau\mathbf{U}_1^{j+1}\big), q\Big) \quad (4.1.4)$$

$$\begin{aligned} \iff \Big(P^{j+1}, q\Big) &= \Big(P^j + \tau dP_0^{j+1}, q\Big) + \lambda\Big(\nabla\cdot\mathbf{U}^{j+1}, q\Big) \\ \iff \Big(\tau^2 dP_1^{j+1}, q\Big) &= \lambda\Big(\nabla\cdot\mathbf{U}^{j+1}, q\Big) \end{aligned} \tag{4.1.5}$$

We have for the semi-discrete pressure p (in space)

$$\frac{p(t_{j+1})-p(t_j)}{\tau} = \frac{\partial}{\partial t}p(t_{j+1}) - \frac{\tau}{2}\frac{\partial^2}{\partial t^2}p(t_{j+1}) + \frac{\tau^2}{6}\frac{\partial^3}{\partial t^3}p(t_{j+1}) + \mathcal{O}(\tau^3). \tag{4.1.6}$$

By assuming $p\in\mathcal{C}^3([0,T];M_h)$, it follows that $\sup_{0<j<N}\|dp_1^{j+1}\| < c(T,\Omega)\|p\|_{\mathcal{C}^3([0,T];M_h)}$ where $c(T,\Omega)$ is a constant which depends on T and Ω but is independent from τ. To get the truncation error on pressure, we substitute p^k for P^k, p_1^k for P_1^k and $\mathbf{u}^k$ for $\mathbf{U}^k$ in the pressure update relation. This gives

$$\begin{aligned} \Big(\tau^2 dp_1^{j+1}, q\Big) - \lambda\Big(\nabla\cdot\mathbf{u}^{j+1}, q\Big) &= \tau^2\left(\frac{p_1^{j+1}-p_1^j}{\tau}, q\right) - \lambda\Big(\nabla\cdot\mathbf{u}^{j+1}, q\Big) \\ \implies \left(\tau^2\frac{\partial}{\partial t}p_1^{j+1} - \frac{\tau^3}{2}\frac{\partial^2}{\partial t^2}p_1^{j+1}, q\right) + \mathcal{O}(\tau^4) &= -\lambda\Big(\nabla\cdot\mathbf{u}^{j+1}, q\Big) \end{aligned}$$

Therefore, $(\nabla\cdot\mathbf{u}^{j+1}, q) = \mathcal{O}(\tau^2)$; i.e., the truncation error on mass conservation is order 2.

Now, we are ready to check the consistency behaviour for (***). To do so, we apply the asymptotic expansion formula for pressure $P^{j+1} = P_0^{j+1} + \tau P_1^{j+1}$ and substitute $\mathbf{u}^{j+1}$ for $\mathbf{U}^{j+1}$, p^j for P^j, etc. We get

$$\begin{aligned} (***) &= \Big(\lambda\nabla\cdot\mathbf{u}^{j+1}, \nabla\cdot\mathbf{v}\Big) - \Big(p^j + \tau dp^{j+1} - \tau^2 dp_1^{j+1}, \nabla\cdot\mathbf{v}\Big) \\ &= \Big(\lambda\nabla\cdot\mathbf{u}^{j+1} - p^{j+1}, \nabla\cdot\mathbf{v}\Big) + \tau\Big(p_1^{j+1} - p_1^j, \nabla\cdot\mathbf{v}\Big) \\ &= \Big(\lambda\nabla\cdot\mathbf{u}^{j+1} - p^{j+1}, \nabla\cdot\mathbf{v}\Big) + \Big(\tau^2\frac{\partial}{\partial t}p_1^{j+1}, \nabla\cdot\mathbf{v}\Big) + \mathcal{O}(\tau^3) \end{aligned} \tag{4.1.7}$$

From here on, we consider two cases:

Case I

For any $\mathbf{v}\in V_h$, we deduced from $\nabla\cdot V_h \subseteq M_h$ that $\exists q\in M_h$ such that $\nabla\cdot\mathbf{v} = q$,

$$\begin{aligned} \implies \Big(\nabla\cdot\mathbf{u}^{j+1}, \nabla\cdot\mathbf{v}\Big) &= \Big(\nabla\cdot\mathbf{u}^{j+1}, q\Big) \\ &= \mathcal{O}(\tau^2) \end{aligned}$$

and from (4.1.7), we conclude that $(***) = (-p^{j+1}, \nabla\cdot\mathbf{v}) + \mathcal{O}(\tau^2)$. Combining all results on truncation error that we obtained earlier, part (a) of the proposition is proven.

Case II

The mixed finite element spaces are such that $\nabla \cdot V_h \not\subseteq M_h$. Then (4.1.7) produces

$$(***) \quad = \quad (\lambda \nabla \cdot \mathbf{u}^{j+1} - p^{j+1}, \nabla \cdot \mathbf{v}) + \mathcal{O}(\tau^2),$$

and we cannot get rid of the term $\lambda(\nabla \cdot \mathbf{u}^{j+1}, \nabla \cdot \mathbf{v})$. Combining all results on truncation error that we obtained earlier, part (b) of the proposition is proven. ∎

Remark 4.1.2. The term "τdP_0^{j+1}" is present in (4.1.5) due to the bootstrapping strategy used for pressure update. This is to ensure that the order of accuracy for pressure is similar to that obtained for velocity since only one ALM iteration is done at each time step. We refer the reader to Guermond and Minev [23] for further details about bootstrapping strategy.

Remark 4.1.3. Truncation error on the momentum equation from the Proposition 4.1.1 can be expressed as

$$\tau^2 \left| \frac{1}{6}\frac{\partial^3}{\partial t^3}\mathbf{u}^{j+1} + \frac{1}{2}\frac{\partial^2}{\partial t^2}\mathbf{u}_1^{j+1} - \Big(\frac{\partial}{\partial t}\mathbf{u}_1^{j+1}\Big) \cdot \nabla \mathbf{u}^{j+1} - \mathbf{u}^{j+1} \cdot \nabla\Big(\frac{\partial}{\partial t}\mathbf{u}_1^{j+1}\Big) + \frac{\partial}{\partial t}p_1^{j+1} \right|$$

We observed numerically in Chapter 2 that the error in time is of order 2 for GM-2, which is consistent with the truncation error $\mathcal{O}(\tau^2)$.

We now present a result that addresses the impact of the **grad**-div term on local mass conservation. This property can be investigated using the following proposition.

Proposition 4.1.4. *By adding the $-\lambda \nabla\nabla \cdot \mathbf{u}$ term ($\lambda > 0$ is the stabilization parameter) in the momentum equation of Stokes problem with a fixed viscosity constant $\nu > 0$ and a fixed mesh, the divergence of $\mathbf{u}$ converges to zero a.e. as $\lambda \to \infty$, according to the following estimate.*

$$\|\nabla \cdot \mathbf{u}_h\|_{0,\Omega} \quad \leq \quad C_\Omega \sqrt{\frac{1}{\nu^2 + 2\lambda\nu}} \|\mathbf{f}\|_{-1,\Omega},$$

Proof: Consider the Stokes problem with the stabilization term $-\lambda \nabla\nabla \cdot \mathbf{u}$ in the momentum equation. This leads to the discrete weak formulation: Find $(\mathbf{u}_h, p_h) \in (V_h \times Q_h)$ such that

$$\begin{aligned} \nu(\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) + \lambda(\nabla \cdot \mathbf{u}_h, \nabla \cdot \mathbf{v}_h) - (p_h, \nabla \cdot \mathbf{v}_h) &= \langle \mathbf{f}, \mathbf{v}_h \rangle, \quad \forall \mathbf{v}_h \in V_h, \\ -(q_h, \nabla \cdot \mathbf{u}_h) &= 0, \quad \forall q_h \in Q_h. \end{aligned}$$

By choosing $\mathbf{v}_h = \mathbf{u}_h$, $q_h = p_h$, we obtain

$$\nu \|\mathbf{u}_h\|_{0,\Omega}^2 + \lambda \|\nabla \cdot \mathbf{u}_h\|_{0,\Omega}^2 \quad \leq \quad \|\mathbf{f}\|_{-1,\Omega} \|\mathbf{u}_h\|_{1,\Omega}$$

Using Poincaré inequality, with a constant $C_\Omega > 0$ that depends on the domain Ω, and Young's inequality, we get

$$\begin{aligned}\frac{\nu}{C_\Omega}\|\mathbf{u}_h\|_{1,\Omega}^2 + \lambda\|\nabla\cdot\mathbf{u}_h\|_{0,\Omega}^2 &\leq \frac{C_\Omega}{2\nu}\|\mathbf{f}\|_{-1,\Omega}^2 + \frac{\nu}{2C_\Omega}\|\mathbf{u}_h\|_{1,\Omega}^2\\ \frac{\nu}{C_\Omega}\|\mathbf{u}_h\|_{1,\Omega}^2 + 2\lambda\|\nabla\cdot\mathbf{u}_h\|_{0,\Omega}^2 &\leq \frac{C_\Omega}{\nu}\|\mathbf{f}\|_{-1,\Omega}^2\end{aligned}$$

Since $\|\nabla\cdot\mathbf{u}_h\|_{0,\Omega} \leq \|\nabla\mathbf{u}_h\|_{0,\Omega} \leq \|\mathbf{u}_h\|_{1,\Omega}$, we get

$$\begin{aligned}\left(\frac{\nu}{C_\Omega} + 2\lambda\right)\|\nabla\cdot\mathbf{u}_h\|_{0,\Omega}^2 &\leq \frac{C_\Omega}{\nu}\|\mathbf{f}\|_{-1,\Omega}^2\\ \Longrightarrow \|\nabla\cdot\mathbf{u}_h\|_{0,\Omega} &\leq C_\Omega\sqrt{\frac{1}{\nu^2 + 2C_\Omega\lambda\nu}}\|\mathbf{f}\|_{-1,\Omega}\end{aligned}$$

∎

Remark 4.1.5. For $\mathbf{f} \neq 0$ and very small viscosity constant ν (or high Reynolds number), if $\lambda = 0$, the incompressibility condition may not be locally enforced since the inequality reduces to $\|\nabla\cdot\mathbf{u}_h\|_{0,\Omega} \leq \frac{C_\Omega}{\nu}\|\mathbf{f}\|_{-1,\Omega}$ in this case. On the other hand, the choice $\lambda \gg \nu^{-1}$ guarantees local incompressibility even for ν small.

Remark 4.1.6. In GM methods for $\lambda \sim 1$, the condition number of the linear system associated to the momentum equation (for each subproblem) behaves as $\mathcal{O}(\tau h^{-2})$ [23].

4.2 The choice of the stabilization parameter λ in GM methods

The stabilization parameter λ in the momentum equation (denoted as λ_m) and continuity equation (denoted as λ_c) can be chosen differently under certain constraints. Our computations show that different λ can be used without compromising the convergence and stability in GM methods (for each subproblem) as long as $0 < \lambda_c \leq 2\lambda_m$. Interestingly, our results are in agreement with those for the ALM formulation of Stokes equations. For instance, Fortin and Glowinski [15] (Theorem 2.1) proved that $0 < \alpha_0 \leq \lambda_c \leq \lambda_m$ is sufficient for convergence, where α_0 is the smallest eigenvalue for the Laplacian operator over $H_0^1(\Omega)$.

We observed a good convergence with GM methods for $\lambda_m = \lambda_c = \lambda$ when λ is taken large. This also holds for the ALM formulation of Stokes equations. Our computatations support Remark 2.4 in [15] to the effect that the stabilization parameter

should not be taken too large to avoid excessive computational requirements resulting from the linear system becoming progressively ill-conditioned as λ increases. Of course, this is less of a problem if direct solvers are used. To ease the set-up of our computations, we consider only $\lambda = \lambda_m = \lambda_c$.

In our case, a suitable value of λ is obtained by trial and error. In essence, the choice of λ impacts GM methods in two ways. The first is how fast the initial oscillations are damped (which results from the velocity and pressure uncoupling). For larger λ, the oscillations are damped within few time steps while smaller λ requires far more time steps. The second impact of the choice of λ is on the quality of the solution. This issue arises with the choice of λ in SBDF methods with **grad**-div stabilization term. The value of λ is linked to the local (or pointwise) mass conservation, which has an impact on the accuracy of the pressure and velocity. Unlike SBDF methods, GM and GM-SRM methods enforce mass conservation in many time steps depending on the choice of λ, as mentioned above. Hence, the choice of λ in GM methods is more delicate.

4.3 New extrapolation formulae of nonlinear terms in DC, GM and GM-SRM methods

In Chapter 2, new nonlinear extrapolation formulae are proposed for the 2^{nd}- and 3^{rd}-order DC, GM and GM-SRM methods without providing an in-depth explanation behind these formula. Nonetheless, the new extrapolation formulae produce numerical improvements in terms of reduction of error (in time) on velocity and pressure.

When combining the formulae proposed in [23] with $\mathbb{P}_2$-$\mathbb{P}_1$ finite elements, we observed a partial loss of numerical accuracy. The loss of numerical accuracy is somewhat negligible for test cases involving only

(a) manufactured solutions,
(b) flow with low Reynolds number (high viscosity), and
(c) short time interval; i.e., T small.

To be more specific, we observed a lost of numerical accuracy (in time) for unsteady flows over a large time interval involving high Reynolds numbers. For instance, this occurs when a long transient is required to capture periodic solutions. We realized that the resulting error on velocity and pressure with DC, GM and GM-SRM methods can be reduced by using improved extrapolation formulae for the nonlinear advection term. This improvement will be illustrated numerically.

To begin with, we recall that the nonlinear term for the 3^{rd}-order methods is approximated by $B(\mathbf{u}_0^{j+1} + \tau\mathbf{u}_1^{j+1} + \tau^2\mathbf{u}_2^{j})$ to give rise to a semi-implicit method. This

expression can be recovered by summing up the nonlinear terms in each subproblem using the asymptotic expansion $(4.0.3)|_{n=m} + \tau(4.0.4)|_{n=m+1} + \tau^2(4.0.5)|_{n=m+2}$. Moreover, for any k^{th}-order methods, the sum of the nonlinear terms can be generalized as follows

$$B(\mathbf{u}^{j+1}) \approx B(\mathbf{u}_0^{j+1} + \tau\mathbf{u}_1^{j+1} + \tau^2\mathbf{u}_2^{j+1} + \cdots + \tau^{k-1}\mathbf{u}_{k-1}^{j}). \tag{4.3.1}$$

The approximation $\mathbf{u}_{k-1}^{j+1} \approx \mathbf{u}_{k-1}^{j}$ is made to maintain the semi-implicit nature of the time-stepping scheme.

Proposition 4.3.1. *The consistency of the nonlinear term in (4.3.1) can be established; i.e.,*

$$\left\|B(\mathbf{u}^{n+1}) - B(\mathbf{u}_0^{n+1} + \tau\mathbf{u}_1^{n+1} + \tau^2\mathbf{u}_2^{n+1} + \cdots + \tau^{k-1}\mathbf{u}_{k-1}^{n})\right\| = \mathcal{O}(\tau^k), \tag{4.3.2}$$

for $k = 1, 2, \ldots$, and any τ such that the resulting method is stable, provided $\|\mathbf{u}_{k-1}^{n+1} - \mathbf{u}_{k-1}^{n}\| \leq c\tau$ is satisfied for some constant $c > 0$.

Proof: Suppose that we have $\mathbf{a} = \sum_{j=0}^{k-2} \tau^j \mathbf{u}_j^{n+1}$, $\mathbf{b} = \tau^{k-1}\mathbf{u}_{k-1}^{n+1}$ and $\mathbf{c} = \tau^{k-1}\mathbf{u}_{k-1}^{n}$. Using the bound established in Appendix B.3, we have the following

$$\begin{aligned}
&\left\|B(\mathbf{u}^{n+1}) - B(\mathbf{u}_0^{n+1} + \tau\mathbf{u}_1^{n+1} + \tau^2\mathbf{u}_2^{n+1} + \cdots + \tau^{k-1}\mathbf{u}_{k-1}^{n})\right\| \\
= \;& \left\|B\Big(\sum_{j=0}^{k-2} \tau^j \mathbf{u}_j^{n+1} + \tau^{k-1}\mathbf{u}_{k-1}^{n+1}\Big) - B\Big(\sum_{j=0}^{k-2} \tau^j \mathbf{u}_j^{n+1} + \tau^{k-1}\mathbf{u}_{k-1}^{n}\Big)\right\| \\
\leq \;& \tau^{k-1}\left\|\Big(\sum_{j=0}^{k-1} \tau^j \mathbf{u}_j^{n+1}\Big) \cdot \nabla(\mathbf{u}_{k-1}^{n+1} - \mathbf{u}_{k-1}^{n})\right\| \\
&+\tau^{k-1}\left\|(\mathbf{u}_{k-1}^{n+1} - \mathbf{u}_{k-1}^{n}) \cdot \nabla\Big(\sum_{j=0}^{k-2} \tau^j \mathbf{u}_j^{n+1} + \tau^{k-1}\mathbf{u}_j^{n}\Big)\right\|.
\end{aligned} \tag{4.3.3}$$

∎

However, the consistency of the nonlinear term does not tell the whole story about the potential error on the velocity and pressure. One should note that the defect correction steps can be constructed in arbitrary manner for each subproblem, especially regarding the nonlinear terms, as long as the sum of these terms gives (4.3.1). We propose new extrapolation formulae for the nonlinear term when $n \geq 2$. The original extrapolation formulae from [23] are maintained for $n = 0$ and $n = 1$. This is to ensure that the initialization of these methods (DC, GM and GM-SRM) is properly done. The new formulae are given in Table 4.1. We denote the approach taken in [23]

as the NL_{123}-formula. We also propose several new formulae; e.g., NL_{223}-, NL_{233}-, NL_{333}, etc. The triple index is used to indicate the number of the past variables used to evaluate the approximations of the nonlinear terms in the 1st-, 2nd- and 3rd-subproblem. As an illustration, the NL_{123}-formula means that a one-step value is used in the 1st-subproblem, while two- and three-step values are used in the 2nd- and 3rd subproblem, respectively.

We set the formula for MS-EX2 and MS-EX3 in a similar way to high-order extrapolation scheme used for multistep methods (see Table 4.1). For instance for MS-EX2, we replace "$\mathbf{u}_0^n$" by "$\mathbf{u}_0^n - \mathbf{u}_0^{n-1}$" and "$\mathbf{u}_1^{n-1}$" by "$\mathbf{u}_1^{n-1} - \mathbf{u}_1^{n-2}$", while for MS-EX3, we replace "$\mathbf{u}_0^n$" by "$3\mathbf{u}_0^n - 3\mathbf{u}_0^{n-1} + \mathbf{u}_0^{n-2}$" and "$\mathbf{u}_1^{n-1}$" by "$\mathbf{u}_1^{n-1} - \mathbf{u}_1^{n-2}$".

We encounter a very restrictive stability condition on the time step with MS-EX2 and MS-EX3 formulae similar to the SBDF methods. The resulting error on velocity and pressure with MS-EX2 and MS-EX3 is larger than that with NL_{123} formula. NL_{023} and NL_{003} formulae produce less accurate approximations than those with the NL_{123} formula. The only advantage is that they allow slightly larger critical time step than NL_{123} formula and still be stable. For these reasons, we do not include MS-EX2, MS-EX3, NL_{023} and NL_{003} formula in our numerical results below.

Table 4.1: Several extrapolation formulae of the nonlinear term for DC-3, GM-3 and GM-SRM-3 methods implemented for $n \geq 2$ (e.g., see (4.0.3)–(4.0.5)).

Subproblem / Formula	**1** ($\mathbf{nl_0^{n+1}}$)	**2** ($\mathbf{nl_1^n}$)	**3** ($\mathbf{nl_2^{n-1}}$)
NL_{123} [23]	$B(\mathbf{u}_0^n)$	$B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1})$	$B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2})$
NL_{223}	$B(\mathbf{u}_0^n + \tau\mathbf{u}_1^n)$	$B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1})$	$B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2})$
NL_{233}	$B(\mathbf{u}_0^n + \tau\mathbf{u}_1^n)$	$B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-1})$	$B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2})$
NL_{333}	$B(\mathbf{u}_0^n + \tau\mathbf{u}_1^n + \tau^2\mathbf{u}_2^{n-1})$	$B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-1})$	$B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2})$
MS-EX2	$B(2\mathbf{u}_0^n - \mathbf{u}_0^{n-1})$	$B(\mathbf{u}_0^n + \tau(2\mathbf{u}_1^{n-1} - \mathbf{u}_1^{n-2}))$	$B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2})$
MS-EX3	$B(3\mathbf{u}_0^n - 3\mathbf{u}_0^{n-1} + \mathbf{u}_0^{n-2})$	$B(\mathbf{u}_0^n + \tau(2\mathbf{u}_1^{n-1} - \mathbf{u}_1^{n-2})$	$B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2})$
NL_{023}	0	$B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1})$	$B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2})$
NL_{003}	0	0	$B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2})$

For the sake of completeness, we also include two types of extrapolation formulae for DC-2, GM-2 and GM-SRM-2 methods; i.e., NL_{12} and NL_{22} formulae as shown in Table 4.2. The interpolation formula NL_{12} is backtracked from the 3rd-order GM methods [23] since the 2nd-order method was not provided explicitly therein.

Table 4.2: Two types of extrapolation formula for DC-2, GM-2 and GM-SRM-2 methods to be implemented only for $n \geq 1$.

Subproblem / Formula	**1** ($\mathbf{nl_0^{n+1}}$)	**2** ($\mathbf{nl_1^n}$)
NL_{12}	$B(\mathbf{u}_0^n)$	$B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1})$
NL_{22}	$B(\mathbf{u}_0^n + \tau \mathbf{u}_1^n)$	$B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1})$

In the following section, we present numerical comparisons implementing several of the proposed nonlinear extrapolation formulae. The numerical comparisons are done in terms of the numerical error (in time), both on velocity and pressure, the resulting numerical stability and the ability to deliver the robustness when computing unsteady flows for large T.

4.3.1 Testing the nonlinear extrapolation formulae : Numerical error in time

In this section, we compare the extrapolation formulae for the nonlinear term in terms of the numerical error (in time) on velocity only. We will compare the NL_{12} and NL_{22} formulae for the 2^{nd}-order methods; and the NL_{123}, NL_{223} and NL_{333} formulae for the 3^{rd}-order methods.

For this purpose, we consider a test case which was used in Chapter 1 and 2; i.e., the 2D flow around the cylinder. The geometry, mesh and computational settings, including boundary and initial conditions, used are taken from Section 4.3 of Chapter 2. Hence, all these informations will not be repeated here. The L^2-error on the velocity between the reference and numerical solutions (with varying time steps) is computed at $t = 8$ using (30) while the order of the convergence (in time) is computed using (31), both given in Chapter 2. We don't compute an exhaustive set of test cases, but instead consider only the 2^{nd}- and 3^{rd}-order DC and GM methods. We conjecture that similar results can be observed with GM-SRM methods using identical nonlinear approximations. Lastly, the stabilization parameter λ is set to 10^4 in both GM-2 and GM-3 methods.

Figure 4.2 shows the log-log plot of the L^2-error on velocity versus the time step τ using DC and GM methods for both 2^{nd}- and 3^{rd}-order accuracy.

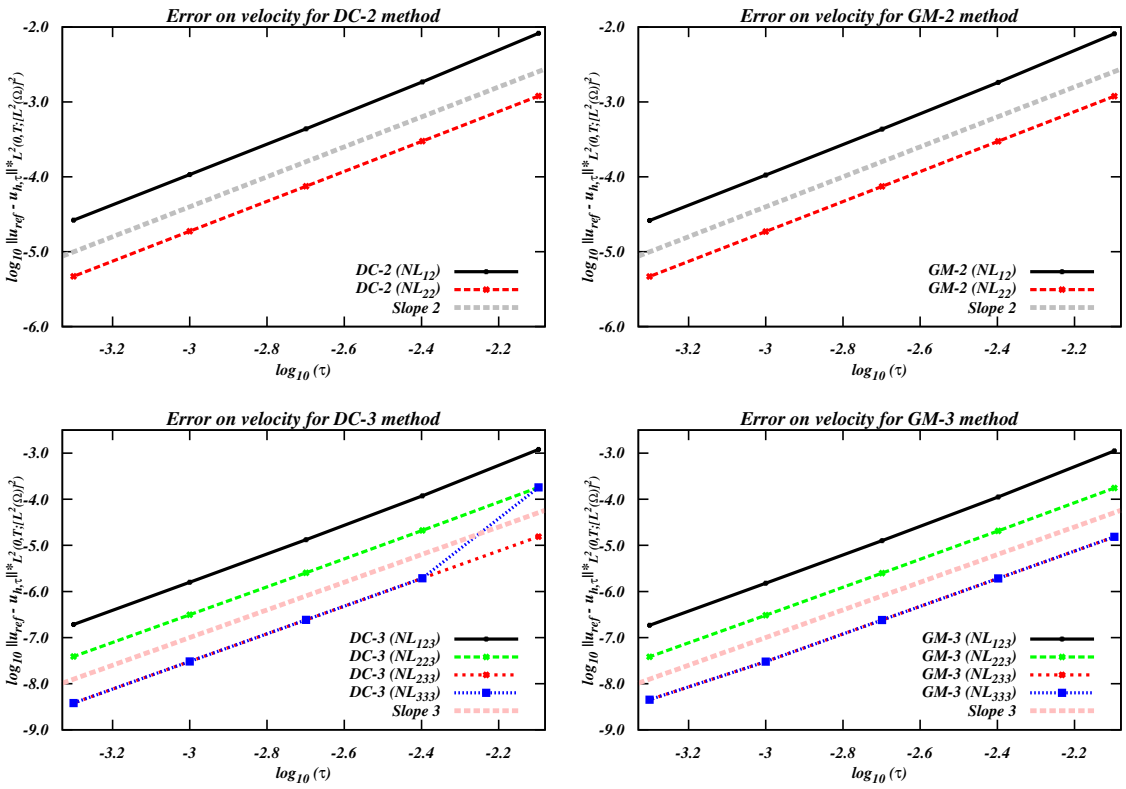

Figure 4.2: L^2 error on velocity as a function of time step τ for DC and GM methods using different extrapolation formulae.

DC-2, DC-3, GM-2 and GM-3 methods reproduced the theoretical order of convergence with all the nonlinear extrapolation formulae. This is not surprising by virtue of Proposition 4.3.1. For DC-3 with the NL_{333} formula, the velocity error is larger at $\tau = 0.008$ but gradually overlaps with the error for the NL_{233} formula for $\tau \leq 0.004$. We suspect that the compromised accuracy with NL_{333} formula at $\tau = 0.008$ could be induced by the onset of the numerical instability.

For GM-2 and DC-2 methods, the L^2-error on velocity at any fixed time step for the NL_{22} formula is about 6 times smaller than the one for the NL_{12} formula. The smallest L^2-error is produced by NL_{233} and NL_{333} formulae. The error reduction with NL_{233} and NL_{333} formulae reaches an order of magnitude (10 times) smaller than the error with the NL_{223} formula. Interestingly, an error reduction of about two orders of magnitude (30-50 times) is observed when $\mathrm{NL}_{233}/\mathrm{NL}_{333}$ formulae are compared to the NL_{123} formula [23]. Being the most accurate (in time) alone is not sufficient to make a method efficient. In the following section, the stability of these nonlinear extrapolation formulae will be investigated. We suspect that the numerical error does not originate from the **grad**-div term since GM and DC methods produce error of

the same magnitude while the GM methods have a **grad**-div term but DC methods do not.

4.3.2 Testing the nonlinear interpolation formulae: Numerical stability

We observed that the extrapolation formulae for the nonlinear term impact the numerical stability of the time-stepping methods. The stability analysis for methods with defect correction (e.g., DC, GM, GM-SRM methods) is particularly challenging as it involves several subproblems and asymptotic expansions. We proceed as in Chapter 2 to obtain the critical time step τ_{crit} and CFL bound using the bisection method.

In addition, we also compute a parameter Θ which controls the Θ-stability condition [37] for the stiffness term using the following formula

$$\Theta := \max_{K\in\mathcal{T}_h} \left\{ \frac{\tau_{\text{crit}}}{Re\ h_K^2} \|\mathbf{u}_h\|_{L^\infty(K)} \right\}. \tag{4.3.4}$$

While τ_{crit} gives us information for picking an ideal time step for the computation of unsteady flows, both parameters CFL and Θ are more relevant to study the stability of our methods. In this section, we also check if the CFL and Θ numbers are mutually related to control the linear stability of our methods, as was proposed by Kress and Lötstedt for semi-implicit schemes [37]. These authors noticed a mild dependence of the CFL bound on Θ in their linear stability analysis. The stability comparisons are done for all methods, i.e., 2^{nd}- and 3^{rd}-order DC, GM and GM-SRM methods (all with the various nonlinear extrapolation formulae that we proposed earlier). In addition, we check whether or not the stability in our methods exhibit a predictable behaviour based on the linear stability analysis established in [37].

For the 2D flow around the cylinder at $Re = 100$, all computations were done as in Section 4.3.1 of Chapter 2. In this particular test however, $T = 12$ is used, ensuring that about two periods of the solution are computed since a period lasts about 5.88 time units. Table 4.3 shows τ_{crit}, CFL and Θ for our methods with different nonlinear extrapolation formulae. We make the following observation

(a) Generally, the CFL bound of SBDF, DC, GM and GM-SRM methods decreases from 2^{nd}-order to 3^{rd}-order accuracy. All 2^{nd}-order methods are found to be more stable than 3^{rd}-order methods in this test case.
(b) For anyone of DC, GM and GM-SRM methods, the CFL bound decreases according to the following ordering of the extrapolation formulae used: NL_{123}, NL_{223},

NL_{233}, NL_{333} (3^{rd}-order accuracy); and NL_{12}, NL_{22} (2^{nd}-order accuracy). The nonlinear extrapolation formulae proposed by [23] produces the most stable methods.

(c) For all methods at a given order of accuracy, the CFL value decreases according to the following order: GM, GM-SRM, DC, SBDF. DC methods are more stable than multistep SBDF methods at a given order of accuracy since the linear systems of the momentum equation retain the structure of 1^{st}-order backward-forward Euler. Actually, GM methods are as stable as GM-SRM methods (their CFL values do not differ significantly). Methods with a **grad-div** term produce better stability behaviour (This has been explained in Remark 4.1.5).

(d) Trends similar to those observed for the CFL bound are observed for τ_{crit} for all methods. The τ_{crit} and CFL of the 3^{rd}-order methods with NL_{123} formula is about twice as large as those for the 3^{rd}-order methods with NL_{333} formula.

Table 4.3: Critical time step τ_{crit}, CFL bound and Θ-stability bound of various methods for the 2D flow around the cylinder at $Re = 100$. The range of Θ-stability bound denoted by "Range Θ" taken from [37] is also included.

Method (formula)	τ_{crit}	**CFL**	Θ	**Range Θ** [37]
SBDF-2	1.19511×10^{-2}	4.14535×10^{-1}	6.74920×10^{-2}	$0.05 - 0.10$
SBDF-3	7.02782×10^{-3}	1.83198×10^{-1}	3.57147×10^{-2}	< 0.001
DC-2 (NL_{12})	1.63044×10^{-2}	4.26562×10^{-1}	8.31591×10^{-2}	$0.05 - 0.10$
DC-2 (NL_{22})	1.15514×10^{-2}	3.01250×10^{-1}	5.87291×10^{-2}	$0.02 - 0.05$
DC-3 (NL_{123})	1.50245×10^{-2}	3.96548×10^{-1}	7.73077×10^{-2}	$0.05 - 0.10$
DC-3 (NL_{223})	1.15514×10^{-2}	3.01178×10^{-1}	5.87152×10^{-2}	$0.02 - 0.05$
DC-3 (NL_{233})	8.11682×10^{-3}	2.11563×10^{-1}	4.12445×10^{-2}	$0.02 - 0.05$
DC-3 (NL_{333})	7.19604×10^{-3}	1.87557×10^{-1}	3.65645×10^{-2}	$0.01 - 0.02$
GM-2 (NL_{12})	2.21248×10^{-2}	5.76103×10^{-1}	1.12312×10^{-1}	$0.10 - 0.15$
GM-2 (NL_{22})	1.45294×10^{-2}	3.78189×10^{-1}	7.37280×10^{-2}	$0.05 - 0.10$
GM-3 (NL_{123})	2.07989×10^{-2}	5.41585×10^{-1}	1.05583×10^{-1}	$0.10 - 0.15$
GM-3 (NL_{223})	1.45193×10^{-2}	3.78065×10^{-1}	7.37044×10^{-2}	$0.05 - 0.10$
GM-3 (NL_{233})	1.05377×10^{-2}	2.74389×10^{-1}	5.34925×10^{-2}	$0.02 - 0.05$
GM-3 (NL_{333})	9.57642×10^{-3}	2.49359×10^{-1}	4.86129×10^{-2}	$0.02 - 0.05$
GM-SRM-2 (NL_{12})	2.20814×10^{-2}	5.74972×10^{-1}	1.12092×10^{-1}	$0.10 - 0.15$
GM-SRM-2 (NL_{22})	1.45223×10^{-2}	3.76852×10^{-1}	7.34680×10^{-2}	$0.05 - 0.10$
GM-SRM-3 (NL_{123})	2.09778×10^{-2}	5.46247×10^{-1}	1.06492×10^{-1}	$0.10 - 0.15$
GM-SRM-3 (NL_{223})	1.44806×10^{-2}	3.77057×10^{-1}	7.35079×10^{-2}	$0.05 - 0.10$
GM-SRM-3 (NL_{233})	1.05255×10^{-2}	2.74120×10^{-1}	5.34401×10^{-2}	$0.02 - 0.05$
GM-SRM-3 (NL_{333})	9.56957×10^{-3}	2.49180×10^{-1}	4.85780×10^{-2}	$0.02 - 0.05$

For the 2D lid-driven cavity flow at $Re = 8\,500$, all computations are set as in Section 4.2.2 of Chapter 2. However, $T = 10$ is used to compute several periods of the solution (the period is 2.23 time units). Table 4.4 shows τ_{crit}, CFL and Θ for our methods with different nonlinear extrapolation formulae. We summarize our conclusion as follows:

(a) The CFL bound for SBDF and DC methods decreases from 2^{nd}-order to 3^{rd}-order methods, while the CFL bound for GM and GM-SRM methods increases from

2^{nd}-order to 3^{rd}-order formulae when comparing NL_{12} with all 3^{rd}-order nonlinear extrapolation formulae.

(b) For DC and GM-SRM methods, the CFL bound decreases according to the following order of the extrapolation formulae used: NL_{333}, NL_{233}, NL_{223}, NL_{123} (3^{rd}-order accuracy); and NL_{22}, NL_{12} (2^{nd}-order accuracy). The nonlinear extrapolation formula proposed by [23] produces the least stable method.

(c) We observed the same behaviour for τ_{crit} than what was observed for the CFL bound for all methods except that τ_{crit} of SBDF-3 method is larger than that of SBDF-2 method (while the opposite was observed for the CFL bound). The τ_{crit} and CFL bound of the 2^{nd}-order method with NL_{22} formula are nearly twice as large as those for the 2^{nd}-order methods with NL_{12} formula. The τ_{crit} and CFL of the 3^{rd}-order method with NL_{333} formula is nearly twice as large as those for the 3^{rd} methods with the NL_{123} formula.

(d) The CFL bound (for all nonlinear extrapolation formulae) is greater for SBDF methods than for DC methods at any given order of accuracy. SBDF methods are more stable than DC methods in 2D lid-driven cavity at $Re = 8\,500$.

Table 4.4: Critical time step τ_{crit}, CFL bound and Θ-stability bound of various methods for the 2D lid-driven cavity flow at $Re = 8500$. The range of Θ-stability bound denoted by "Range Θ" taken from [37] is also included.

Method (formula)	τ_{crit}	**CFL**	Θ	**Range Θ** [37]
SBDF-2	1.45508×10^{-3}	2.28314×10^{-1}	3.22230×10^{-3}	$0.001 - 0.005$
SBDF-3	1.57227×10^{-3}	2.23712×10^{-1}	3.14192×10^{-3}	< 0.001
DC-2 (NL_{12})	7.12891×10^{-4}	8.73756×10^{-2}	1.25991×10^{-3}	$0.001 - 0.005$
DC-2 (NL_{22})	1.75781×10^{-3}	2.23343×10^{-1}	3.25735×10^{-3}	$0.02 - 0.05$
DC-3 (NL_{123})	6.64062×10^{-4}	8.13909×10^{-2}	1.17361×10^{-3}	$0.001 - 0.005$
DC-3 (NL_{223})	7.81250×10^{-4}	9.57540×10^{-2}	1.38072×10^{-3}	$0.001 - 0.005$
DC-3 (NL_{233})	1.59180×10^{-3}	1.95099×10^{-1}	2.81322×10^{-3}	$0.01 - 0.02$
DC-3 (NL_{333})	1.59180×10^{-3}	1.95099×10^{-1}	2.81322×10^{-3}	$0.01 - 0.02$
GM-2 (NL_{12})	1.16211×10^{-3}	1.42434×10^{-1}	2.05382×10^{-3}	$0.005 - 0.010$
GM-2 (NL_{22})	2.58789×10^{-3}	3.17185×10^{-1}	4.57363×10^{-3}	$0.02 - 0.05$
GM-3 (NL_{123})	1.34766×10^{-3}	1.65176×10^{-1}	2.38174×10^{-3}	$0.01 - 0.02$
GM-3 (NL_{223})	1.38672×10^{-3}	1.69963×10^{-1}	2.45078×10^{-3}	$0.01 - 0.02$
GM-3 (NL_{233})	2.37305×10^{-3}	2.90853×10^{-1}	4.19393×10^{-3}	$0.02 - 0.05$
GM-3 (NL_{333})	2.39258×10^{-3}	2.93247×10^{-1}	4.22845×10^{-3}	$0.02 - 0.05$
GM-SRM-2 (NL_{12})	1.16211×10^{-3}	1.42434×10^{-1}	2.05382×10^{-3}	$0.005 - 0.010$
GM-SRM-2 (NL_{22})	2.59766×10^{-3}	3.18382×10^{-1}	4.59089×10^{-3}	$0.02 - 0.05$
GM-SRM-3 (NL_{123})	1.30859×10^{-3}	1.60388×10^{-1}	2.31270×10^{-3}	$0.01 - 0.02$
GM-SRM-3 (NL_{223})	1.35742×10^{-3}	1.66373×10^{-1}	2.39900×10^{-3}	$0.01 - 0.02$
GM-SRM-3 (NL_{233})	2.37305×10^{-3}	2.90853×10^{-1}	4.19393×10^{-3}	$0.02 - 0.05$
GM-SRM-3 (NL_{333})	2.39258×10^{-3}	2.93247×10^{-1}	4.22845×10^{-3}	$0.02 - 0.05$

By comparing the two test cases we observe that the stability of the methods differs on several aspects. First, we observe an opposite stability behaviour in DC, GM and GM-SRM methods in terms of the nonlinear extrapolation formulae in the two

test cases. NL_{12} and NL_{123} formulae produce the largest CFL bounds in the 2D flow around the cylinder for 2nd- and 3rd-order methods, respectively. However, the same formulae produce the smallest CFL bounds in the 2D lid-driven cavity. Second, the CFL bound and τ_{crit} have opposite behaviour in 2nd- and 3rd-order SBDF methods that had not been explained in Section 4.4 of Chapter 2. Third, the stability is weaker for DC methods than for SBDF methods though DC methods have the structure of 1st-order backward-forward Euler usually associated with a better stability. We don't have a clear explanation for these contradictions except the fact that the stability behaviour is sometimes case dependent.

Next, we compare our stability results in terms of CFL and Θ with the one obtained using a linear stability analysis in [37]. Figure 4.3 shows the stability region for 1st-, 2nd- and 3rd-order SBDF methods; i.e., the bounding curve for the CFL number for various Θ values.

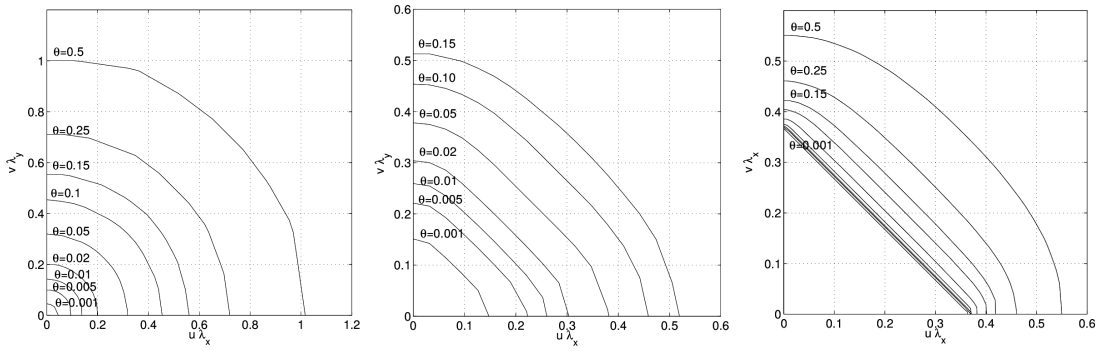

Figure 4.3: Stability domain for 1st-(left), 2nd- (center) and 3rd-order (right) SBDF methods for various values of the parameter Θ. These plots are taken from Kress and Lötstedt [37].

The space discretization employed in [37] is a 4th-order compact finite difference scheme with equidistant staggered Cartesian grid. The CFL number is thus dependent on the direction of propagation of the wave. Our CFL number is computed using the diameter h_K of an element K, hence it is not directly linked with their directional CFL number; e.g., $u\frac{\tau}{h_x}$ and $v\frac{\tau}{h_y}$ where h_x and h_y are the mesh size in x- and y-direction, as shown on Figure 4.3. We compare our CFL number with either $u\frac{\tau}{h_x}$ or $v\frac{\tau}{h_y}$ to have a rough estimate of the CFL bound as a function of Θ. The "Range Θ" that are given in Tables 4.3 and 4.4 correspond to the (approximate) value of Θ such that $u\frac{\tau}{h_x}$ (or $v\frac{\tau}{h_y}$) equals our CFL bound. For DC, GM and GM-SRM methods, we use the graph for the semi-implicit BDF-1 method [37] (similar to the SBDF methods in our work) since the defect correction method has the structure of backward-forward Euler method.

In Tables 4.3 and 4.4, the blue font is used to indicate that the Θ values obtained with our methods fall within the "Range Θ" found in the [37] for a given CFL bound. The red font denotes that the Θ values lie outside the published "Range Θ".

For 2D flow around the cylinder (see Table 4.3), the Θ values of all methods agree well with "Range Θ" except for DC-2 (NL_{22}-formula), DC-3 (NL_{223} formula), DC-2 (NL_{333} formula), GM-3 (NL_{233} formula) and GM-3 (NL_{223} formula) methods which are about 2% to 8% off their bounding curve (see Figure 4.3).

For the 2D lid-driven cavity flow at $Re = 8\,500$, we do not have a favourable comparison. Only the Θ values of SBDF-2, SBDF-3, DC-2 (NL_{12} formula), DC-3 (NL_{123} formula) and DC-3 (NL_{233} formula) methods agree with the "Range Θ" from [37]. Linear stability analysis conducted in [37] may not be applicable to study flows with high Reynolds numbers and time-stepping methods where a **grad**-div stabilization term is present (such as GM and GM-SRM methods).

4.3.3 Testing the nonlinear interpolation formulae: The propagation of numerical error in time

We found that some error can potentially grow when simulation are carried over long period of time. This error is otherwise almost undetectable when test case is computed over a very short period of time or when time step τ is chosen very small. This numerical issue however, does not affect the 2^{nd}-order DC, GM, GM-SRM and all SBDF methods.

To illustrate this, we computed the 2D flow around the cylinder at $Re = 100$ which was used in Section 4.3.1 except that we set $T = 2\,000$ and take a reasonably large time step for the semi-implicit methods; i.e., $\tau = 0.005$. We solve the test problem with DC-3 and GM-3 methods using the formulae NL_{123}, NL_{223}, NL_{233} and NL_{333}. Figures 4.4 and 4.5 show the time-evolution of the x-velocity u monitored at two locations; i.e., near the upper boundary $(13, 3)$ and near the free exit $(22, 0)$.

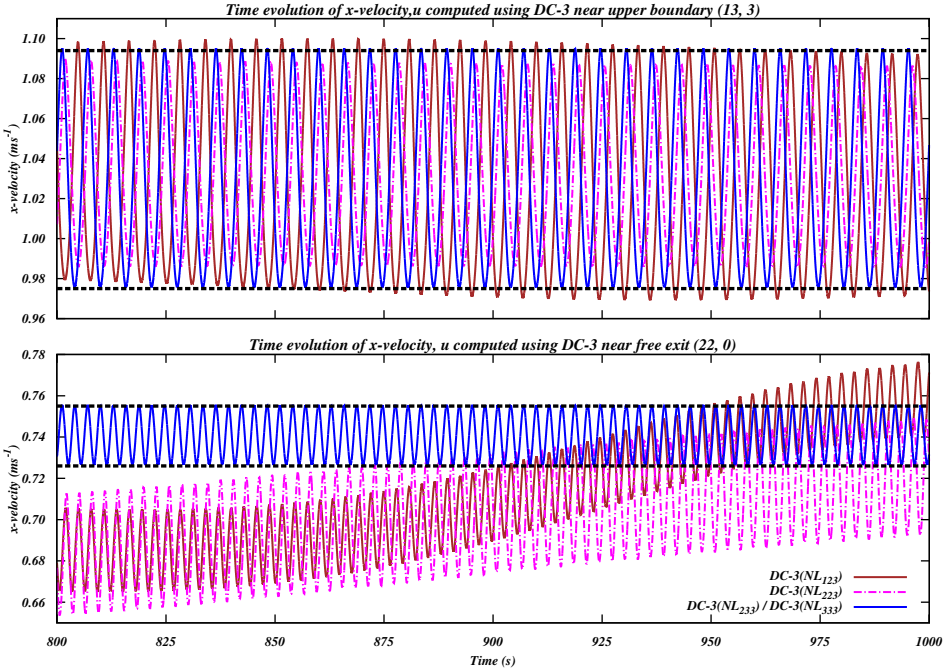

Figure 4.4: Time evolution of the x-velocity u monitored at the point (13, 3) near the upper boundary and at the point (22, 0) near the free exit with DC-3 method and the extrapolation formulae NL_{123}, NL_{223}, NL_{233} and NL_{333}.

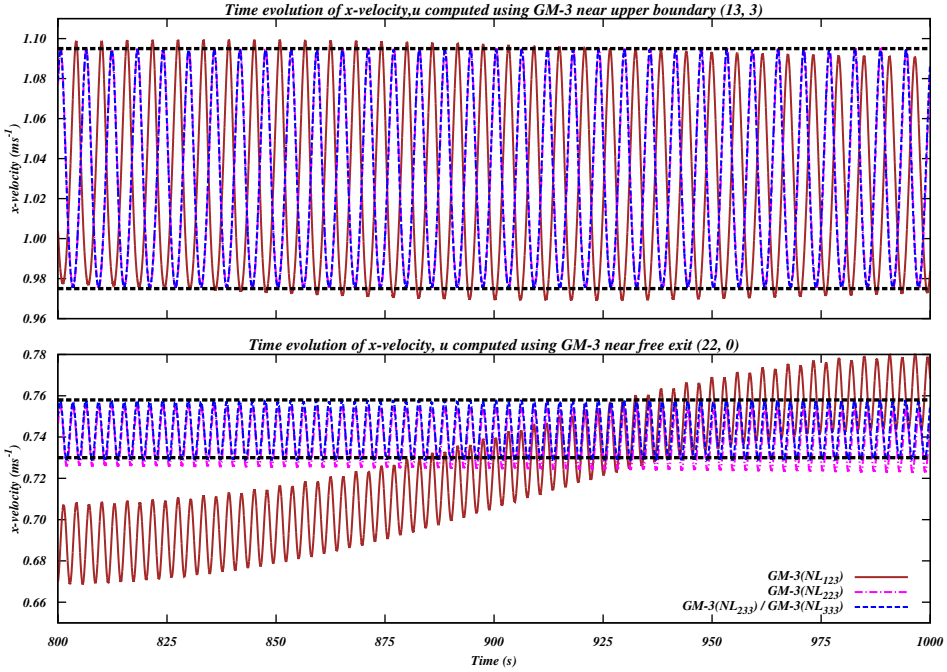

Figure 4.5: Time evolution of the x-velocity u monitored at the point (13, 3) near the upper boundary and at the point (22, 0) near the free exit with GM-3 method and the extrapolation formulae NL_{123}, NL_{223}, NL_{233} and NL_{333}.

The velocity of the 2D flow around the cylinder at $Re = 100$ is periodic with one frequency mode having a period of about 5.9 time units. The velocity $\mathbf{u}$ obtained with NL_{123} and NL_{223} formulae is very far from achieving the above expectations. Error propagates in time and becomes more severe at the point near the free exit than near the upper boundary. On the other hand, the full periodic solution is maintained with both NL_{233} and NL_{333} formulae with a period of about 5.9 time units. We plot the same curve for both NL_{233} and NL_{333} formulae since they overlap. Using NL_{233} formula, the DC, GM and GM-SRM methods produce nearly identical periods for this flow (see Chapter 2).

We observe that DC-3 and GM-3 methods with the NL_{123} and NL_{223} formulae produce solutions with similar numerical artifact. Hence, we rule out that the **grad**-div stabilization term or the decoupling of velocity and pressure produce this type of error. Clearly, the error propagating in time can be improved by implementing NL_{233} and NL_{333} formulae in DC-3 and GM-3 methods.

4.3.4 Testing the nonlinear schemes: Short conclusions

In Section 4.3.1, we have illustrated numerically that the NL_{22} formula (2^{nd}-order methods) as well as NL_{233} and NL_{333} formulae (3^{rd}-order methods) produce the least numerical error (in time) while still maintaining the theoretical order of convergence. A stability comparison was made with test cases for all methods in terms of the critical time step τ_{crit}, the maximum CFL number and Θ-stability condition. We noted that the stability behaviour induced by the nonlinear interpolation formulae depends on the test case and is difficult to predict, at least in term of which method is the most stable.

For the 2^{nd}-order methods, NL_{12} formula produces a better numerical stability than NL_{22} formula when computing 2D flow around the cylinder. However, for 2D lid-driven cavity flow, NL_{22} formula produces a better numerical stability than NL_{12} formula. For the 3^{rd}-order variants, NL_{123} formula is the most appealing in terms of stability when computing 2D flow around the cylinder, while NL_{333} formula is the best for 2D lid-driven cavity flow. A good balance in term of stability for both test cases is provided by the NL_{223} and NL_{233} formulae. Moreover, NL_{233} formula is the best candidate for 3^{rd}-order methods since it does not produce numerical error in computations over a reasonably long time interval.

To conclude, NL_{22} and NL_{233} formulae seem to be the ideal choices for all 2^{nd}- and 3^{rd}-order methods, respectively, justifying why these interpolation formulae had been implemented in Chapter 2.

4.4 The grad-div splitting for GM and GM-SRM

We have seen in Chapter 2 that both GM and GM-SRM methods are very promising in terms of accuracy since they can reach arbitrary order from the defect correction strategy. Moreover, computations with GM and GM-SRM do not require the solution of a large saddle point problem. By using velocity and pressure splitting akin to a formulation that mimics ALM, the saddle point problems in GM and GM-SRM methods are broken into smaller linear system with symmetric and positive definite matrices. With $\mathbb{P}_2$-$\mathbb{P}_1$ elements, the ratio of the degrees of freedom between velocity and pressure is about 9:1 in 2D while being 19:1 in 3D. The linear system for the momentum equation in GM and GM-SRM is still significantly larger than the linear system for the pressure update (continuity equation). Below we will present the GM method only but the proposed strategies apply to GM-SRM as well.

In this section, we further reduce the complexity of the linear system for the momentum equation in GM methods using a **grad**-div splitting. **Grad**-div splitting uncou-

ples the velocity components u, v and w (in 3D problem). As a result, much simpler linear systems are solved which involve only scalar parabolic problems for each velocity component. We got our inspiration from [24]. This results in a smaller symmetric and positive definite matrice, for each velocity component separately. However, the trade-of is that more linear systems (e.g., two linear systems for 2D flows and three linear systems for 3D flows) are solved in each subproblem. Also, we recall that the number of subproblem follows the order k of the method. Nonetheless, if a finer grid is required in 3D computations, this approach is viable, especially if combined with an iterative linear solver.

To give a general idea how this can be realized, we first recall from Proposition 4.1.1 that the resulting **grad**-div term in GM methods does not vanish with $\mathbb{P}_2$-$\mathbb{P}_1$ elements. Since GM methods treat the pressure in an explicit way in the momentum equation, the only coupling mechanism between the velocity components in the linear solve comes from the mixed partial differential operators from the **grad**-div term; i.e., ∂_{xy} and ∂_{yx} for 2D problem, and ∂_{xy}, ∂_{yx}, ∂_{xz}, ∂_{zx}, ∂_{yz} and ∂_{zy} for 3D problem. To uncouple the velocity components, these mixed differential terms can be computed explicitly.

In the two-dimensional case, we consider the mass matrix M, the stiffness matrix A is built from $-\nu\Delta\mathbf{u}$, and the matrices obtained from discretizing 2^{nd}-order differential terms; i.e., $C_{11} = -\lambda\partial_{xx}$, $C_{12} = -\lambda\partial_{xy}$, $C_{21} = -\lambda\partial_{yx}$ and $C_{22} = -\lambda\partial_{yy}$. For brevity, the right hand side of the linear system for the x- and y-components (which contains the past steps, nonlinear advection term, pressure gradient, higher-order defect correction term and external forces) are noted with $F_1 = F(u_i^n, v_i^n, p_i^n, f_x^{n+1})$ and $F_2 = F(u_i^n, v_i^n, p_i^n, f_y^{n+1})$, respectively. Then, the momentum equation of all subproblems in GM-3 method (4.0.3)–(4.0.5) can be expressed algebraically as follows:

$$\begin{bmatrix} \dfrac{M}{\tau} + A + C_{11} & C_{12} \\ C_{21} & \dfrac{M}{\tau} + A + C_{22} \end{bmatrix} \begin{bmatrix} u_i^{n+1} \\ v_i^{n+1} \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}, \tag{4.4.1}$$

where $i = 0, 1, 2$ is the index representing the variable for subproblem 1, 2 and 3, respectively. The superscript n is the time index. The data u_i^0, v_i^0 and p_i^0 for $i = 0, 1, 2$ are required to initiate the computations. The above linear system (4.4.1) can be solved independently for the x-velocity u and y-velocity v, using the block Gauss-Seidel iterative method (see [48]): For any given $(u^{0,0}, v^{0,0}, p^{0,0}) := (u^0, v^0, p^0)$ and n fixed, we compute

$$\left(\frac{M}{\tau} + A + C_{11}\right) u_i^{n+1,m+1} = F_1 - C_{12} u_i^{n+1,m},$$

$$\left(\frac{M}{\tau}+A+C_{22}\right)v_i^{n+1,m+1} \;=\; F_2-C_{21}u_i^{n+1,m+1},\quad i=0,1,2,$$

for $m=0,1,2,\ldots$ until $\|\mathbf{u}^{n+1,m+1}-\mathbf{u}^{n+1,m}\|<\epsilon$ for a given tolerance $\epsilon>0$. Then, we set $u_i^{n+1}:=u_i^{n+1,m+1}$, $v_i^{n+1}:=v_i^{n+1,m+1}$ and $p_i^{n+1}:=p_i^{n+1,m+1}$ and the iteration goes on with the next value of n. The convergence of this algorithm is important to ensure numerical accuracy. We require that these solutions be as close as possible to the solution produced by GM methods without the **grad**-div splitting.

The convergence of **grad**-div uncoupling can be enforced in several ways. In the following section, we present three different approaches: first, a variant that employs Jacobi/Gauss–Seidel iterations with stopping criterion; second, the approach proposed by Guermond and Minev [24] which is known as the "bootstrapping" technique; and third, a correction method similar to the one used for the nonlinear term. For the second and third methods, no iteration is needed and no stopping criterion is required for the convergence of the **grad**-div uncoupling. This is very appealing for numerical efficiency. For the sake of illustration, all **grad**-div splitting schemes are constructed based upon GM-3 method. Similar techniques are also applicable to the GM-SRM method.

4.4.1 Grad-div splitting: Iterating using the Jacobi/Gauss–Seidel method

Let us define the following matrices

$$C_\Delta=\begin{bmatrix}0 & C_{12}\\ 0 & 0\end{bmatrix}\quad\text{or}\quad C_\Delta=\begin{bmatrix}0 & C_{12} & C_{13}\\ 0 & 0 & C_{23}\\ 0 & 0 & 0\end{bmatrix},$$

for 2D and 3D Navier–Stokes equations, respectively. Here, $C_{ij}=-\lambda\partial_{ij}$ where $i,j=1,2$ or 3 denotes the x-, y- and z-component of the velocity, respectively. The Jacobi method applied to the 3rd-order GM method with NL_{233} formula and **grad**-div splitting reads: Given the initial condition $(\mathbf{u}_0^0,p_0^0)=(\mathbf{u}(\mathbf{0}),p(0))$ and setting $(\mathbf{u}_1^0,p_1^0)=(\mathbf{u}_2^0,p_2^0)=(0,0)$, we compute $(\mathbf{u}^{n+1},p^{n+1})$, $n\geq 0$, by solving the equations (4.4.2), (4.4.3) and (4.4.4) for $(\mathbf{u}_0^{n+1},p_0^{n+1})$, $(\mathbf{u}_1^n,p_1^n)$ and $(\mathbf{u}_2^{n-1},p_2^{n-1})$, re-

spectively.

$$\text{For } n \geq 0, \begin{cases} \mathbf{nl}_0^{n+1} = \begin{cases} B(\mathbf{u}_0^n), & \text{for } 0 \leq n \leq 1, \\ B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1}), & \text{for } n \geq 2, \end{cases} \\ \\ \text{Initialize } \mathbf{u}_0^{n+1} := \mathbf{u}_0^n. \text{ Set } \overline{\mathbf{u}} = \mathbf{u}_0^{n+1} \text{ and solve for } \mathbf{u}_0^{n+1} \text{ the equation} \\ \dfrac{\mathbf{u}_0^{n+1} - \mathbf{u}_0^n}{\tau} - \nu \Delta \mathbf{u}_0^{n+1} - \lambda \nabla \nabla \cdot \mathbf{u}_0^{n+1} - C_\Delta(\mathbf{u}_0^{n+1} - \overline{\mathbf{u}}) + \nabla p_0^n \\ \qquad = \mathbf{f}^{n+1} - \mathbf{nl}_0^{n+1}, \\ \text{until } \|\mathbf{u}_0^{n+1} - \overline{\mathbf{u}}\| < e \text{ for a small } e > 0. \\ \\ p_0^{n+1} = p_0^n - \lambda \nabla \cdot \mathbf{u}_0^{n+1}, \\ \mathbf{du}_0^{n+1} = (\mathbf{u}_0^{n+1} - \mathbf{u}_0^n)/\tau, \quad dp_0^{n+1} = (p_0^{n+1} - p_0^n)/\tau. \end{cases} \tag{4.4.2}$$

$$\text{For } n \geq 1, \begin{cases} \mathbf{d^2 u}_0^{n+1} = (\mathbf{du}_0^{n+1} - \mathbf{du}_0^n)/\tau, \\ \mathbf{nl}_1^n = \begin{cases} B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1}), & \text{for } n = 1, \\ B(\mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}), & \text{for } n \geq 2, \end{cases} \\ \\ \text{Initialize } \mathbf{u}_1^n := \mathbf{u}_1^{n-1}. \text{ Set } \overline{\mathbf{u}} = \mathbf{u}_1^n \text{ and solve for } \mathbf{u}_1^n \text{ the equation} \\ \dfrac{\mathbf{u}_1^n - \mathbf{u}_1^{n-1}}{\tau} - \nu \Delta \mathbf{u}_1^n - \lambda \nabla \nabla \cdot \mathbf{u}_1^n - C_\Delta(\mathbf{u}_1^n - \overline{\mathbf{u}}) + \nabla(p_1^{n-1} + dp_0^n) \\ \qquad = -\dfrac{1}{2}\mathbf{d^2 u}_0^{n+1} - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\tau}, \\ \text{until } \|\mathbf{u}_1^n - \overline{\mathbf{u}}\| < e \text{ for a small } e > 0. \\ \\ p_1^n = p_1^{n-1} + dp_0^n - \lambda \nabla \cdot \mathbf{u}_1^n, \\ \mathbf{du}_1^n = (\mathbf{u}_1^n - \mathbf{u}_1^{n-1})/\tau, \quad dp_1^n = (p_1^n - p_1^{n-1})/\tau. \end{cases} \tag{4.4.3}$$

$$
\text{for } n \geq 2, \begin{cases}
\mathbf{d^2 u}_1^n = (\mathbf{du}_1^n - \mathbf{du}_1^{n-1})/\tau, \quad \mathbf{d^3 u}_0^{n+1} = (\mathbf{d^2 u}_0^{n+1} - \mathbf{d^2 u}_0^n)/\tau, \\
\mathbf{nl}_2^{n-1} = B(\mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-2}), \\
\\
\text{Initialize } \mathbf{u}_2^{n-1} := \mathbf{u}_2^{n-2}. \text{ Set } \overline{\mathbf{u}} = \mathbf{u}_2^{n-1} \text{ and solve for } \mathbf{u}_2^{n-1} \text{ the equation} \\
\dfrac{\mathbf{u}_2^{n-1} - \mathbf{u}_2^{n-2}}{\tau} - \nu \Delta \mathbf{u}_2^{n-1} - \lambda \nabla \nabla \cdot \mathbf{u}_2^{n-1} - C_\Delta(\mathbf{u}_2^{n-1} - \overline{\mathbf{u}}) \\
\qquad + \nabla(p_2^{n-2} + dp_1^{n-1}) = -\dfrac{1}{2}\mathbf{d^2 u}_1^n + \dfrac{1}{6}\mathbf{d^3 u}_0^{n+1} - \dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\tau^2}, \\
\text{until } \|\mathbf{u}_2^{n-1} - \overline{\mathbf{u}}\| < e \text{ for a small } e > 0. \\
\\
p_2^{n-1} = p_2^{n-2} + dp_1^{n-1} - \lambda \nabla \cdot \mathbf{u}_2^{n-1}, \\
\mathbf{u}^{n-1} = \mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}, \quad p^{n-1} = p_0^{n-1} + \tau p_1^{n-1} + \tau^2 p_2^{n-1}.
\end{cases}
\tag{4.4.4}
$$

Remark 4.4.1. For the Gauss–Seidel method, the assignment of $\overline{\mathbf{u}}$ is modified. When solving the x-component of the momentum equation in each subproblem, we set

$$
\overline{\mathbf{u}} = \begin{bmatrix} u_i^n \\ v_i^n \end{bmatrix} \quad \text{and} \quad \overline{\mathbf{u}} = \begin{bmatrix} u_i^n \\ v_i^n \\ w_i^n \end{bmatrix}
$$

for 2D and 3D problems, respectively. When solving the y-component of the momentum equation in each subproblem, we set

$$
\overline{\mathbf{u}} = \begin{bmatrix} u_i^{n+1} \\ v_i^n \end{bmatrix} \quad \text{and} \quad \overline{\mathbf{u}} = \begin{bmatrix} u_i^{n+1} \\ v_i^n \\ w_i^n \end{bmatrix},
$$

for 2D and 3D problems, respectively. When solving the z-component of the momentum equation in each subproblem, we set

$$
\overline{\mathbf{u}} = \begin{bmatrix} u_i^{n+1} \\ v_i^{n+1} \\ w_i^n \end{bmatrix}
$$

for 3D problem, where $i = 1, 2, 3$ is the index of the subproblems involved. We observe that the Gauss–Seidel algorithm converges faster than Jacobi algorithm. Hence, Gauss–Seidel method is chosen in our work.

The only drawback of the method above is the need to solve several scalar parabolic problems to satisfy the stopping criterion. The CPU time becomes prohibitive for 3D problems with high-order methods; e.g., GM-3 method. However, in certain test problems when steady flows are involved, the Gauss–Seidel iterations can be limited to two or three iterations since time-stepping eventually resolves the fully coupled system.

4.4.2 Grad-div splitting: Guermond and Minev

The computation of **grad**-div splitting in GM methods can be accelerated using a scheme proposed by Guermond and Minev [24]. However, the setback is a small lost of accuracy in both time and space. This method for 3rd-order GM method proceeds as follows: Set $(\mathbf{u}_0^0, p_0^0) = (\mathbf{u}(\mathbf{0}), p(0))$ and $(\mathbf{u}_1^0, p_1^0) = (\mathbf{u}_2^0, p_2^0) = (0, 0)$. We compute $(\mathbf{u}^{n+1}, p^{n+1})$, $n \geq 0$, by solving the equations (4.4.5), (4.4.6) and (4.4.7) for $(\mathbf{u}_0^{n+1}, p_0^{n+1})$, $(\mathbf{u}_1^n, p_1^n)$ and $(\mathbf{u}_2^{n-1}, p_2^{n-1})$, respectively.

$$\text{For } n \geq 0, \begin{cases} \mathbf{nl}_0^{n+1} = B(\mathbf{u}_0^n), \\ \dfrac{\mathbf{u}_0^{n+1} - \mathbf{u}_0^n}{\tau} - \nu\Delta\mathbf{u}_0^{n+1} - \lambda\nabla\nabla\cdot\mathbf{u}_0^{n+1} - C_\Delta(\mathbf{u}_0^{n+1} - \mathbf{u}_0^n) + \nabla p_0^n \\ \qquad = \mathbf{f}^{n+1} - \mathbf{nl}_0^{n+1}, \\ p_0^{n+1} = p_0^n - \lambda\nabla\cdot\mathbf{u}_0^{n+1}, \\ \mathbf{du}_0^{n+1} = (\mathbf{u}_0^{n+1} - \mathbf{u}_0^n)/\tau, \quad dp_0^{n+1} = (p_0^{n+1} - p_0^n)/\tau. \end{cases} \tag{4.4.5}$$

$$\text{For } n \geq 1, \begin{cases} \mathbf{d^2u}_0^{n+1} = (\mathbf{du}_0^{n+1} - \mathbf{du}_0^n)/\tau, \\ \mathbf{nl}_1^n = B(\mathbf{u}_0^n + \tau\mathbf{u}_1^{n-1}), \\ \dfrac{\mathbf{u}_1^n - \mathbf{u}_1^{n-1}}{\tau} - \nu\Delta\mathbf{u}_1^n - \lambda\nabla\nabla\cdot\mathbf{u}_1^n - C_\Delta(\mathbf{u}_1^n - \mathbf{u}_1^{n-1}) + \dfrac{C_\Delta(\mathbf{u}_0^n - \mathbf{u}_0^{n-1})}{\tau} \\ \qquad +\nabla(p_1^{n-1} + dp_0^n) = -\dfrac{1}{2}\mathbf{d^2u}_0^{n+1} - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\tau} + \lambda\nabla\nabla\cdot\mathbf{du}_0^n, \\ p_1^n = p_1^{n-1} + dp_0^n - \lambda\nabla\cdot\mathbf{u}_1^n, \\ \mathbf{du}_1^n = (\mathbf{u}_1^n - \mathbf{u}_1^{n-1})/\tau, \quad dp_1^n = (p_1^n - p_1^{n-1})/\tau. \end{cases} \tag{4.4.6}$$

$$\text{for } n \geq 2, \begin{cases} \mathbf{d^2u}_1^n = (\mathbf{du}_1^n - \mathbf{du}_1^{n-1})/\tau, \quad \mathbf{d^3u}_0^{n+1} = (\mathbf{d^2u}_0^{n+1} - \mathbf{d^2u}_0^n)/\tau, \\ \mathbf{nl}_2^{n-1} = B(\mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-2}), \\ \dfrac{\mathbf{u}_2^{n-1} - \mathbf{u}_2^{n-2}}{\tau} - \nu\Delta\mathbf{u}_2^{n-1} - \lambda\nabla\nabla\cdot\mathbf{u}_2^{n-1} - C_\Delta(\mathbf{u}_2^{n-1} - \mathbf{u}_2^{n-2}) \\ \qquad +\dfrac{C_\Delta(\mathbf{u}_1^{n-1} - \mathbf{u}_1^{n-2})}{\tau} + \nabla(p_2^{n-2} + dp_1^{n-1}) = -\dfrac{1}{2}\mathbf{d^2u}_1^n + \dfrac{1}{6}\mathbf{d^3u}_0^{n+1} \\ \qquad -\dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\tau} + \lambda\nabla\nabla\cdot\mathbf{du}_1^{n-1}, \\ p_2^{n-1} = p_2^{n-2} + dp_1^{n-1} - \lambda\nabla\cdot\mathbf{u}_2^{n-1}, \\ \mathbf{u}^{n-1} = \mathbf{u}_0^{n-1} + \tau\mathbf{u}_1^{n-1} + \tau^2\mathbf{u}_2^{n-1}, \quad p^{n-1} = p_0^{n-1} + \tau p_1^{n-1} + \tau^2 p_2^{n-1}. \end{cases} \tag{4.4.7}$$

One advantage of such scheme is that there is no subiterations done to address the **grad**-div splitting. A careful investigation of the algorithm shows that this can be achieved for two reasons. The first is the defect correction strategy to improve the error from uncoupling from one subproblem to the next in a cascade manner. Second, a bootstrapping technique is used to improve the divergence free condition; i.e., the extra terms $\lambda\nabla\nabla\cdot\mathbf{du}_0^n$ and $\lambda\nabla\nabla\cdot\mathbf{du}_1^{n-1}$ that can be seen in the right hand side of the momentum equation (4.4.6) and (4.4.7), respectively. This is similar to the bootstrapping technique that is used for the pressure; i.e., the two extra terms dp_0^n and dp_1^{n-1} in the momentum equation (4.4.6) and (4.4.7), respectively. The stability and convergence of this approach are investigated in [24].

4.4.3 Grad-div splitting: An analogue to the nonlinear ansatz formulation

In this section, we propose a third approach to handle the **grad**-div splitting in GM and GM-SRM methods. Let us define the following matrices

$$\overline{C}_\Delta = \begin{bmatrix} 0 & 0 \\ C_{21} & 0 \end{bmatrix} \quad \text{and} \quad \overline{A} = \begin{bmatrix} -(\nu+\lambda)\partial_{xx} - \nu\partial_{yy} & 0 \\ 0 & -\nu\partial_{xx} - (\nu+\lambda)\partial_{yy} \end{bmatrix};$$

$$\overline{C}_\Delta = \begin{bmatrix} 0 & 0 & 0 \\ C_{21} & 0 & 0 \\ C_{31} & C_{32} & 0 \end{bmatrix}$$

and

$$\overline{A} = \begin{bmatrix} -(\nu+\lambda)\partial_{xx} - \nu(\partial_{yy}+\partial_{zz}) & 0 & 0 \\ 0 & -\nu(\partial_{xx}+\partial_{zz}) - (\nu+\lambda)\partial_{yy} & 0 \\ 0 & 0 & -\nu(\partial_{xx}+\partial_{yy}) - (\nu+\lambda)\partial_{zz} \end{bmatrix};$$

for 2D and 3D Navier–Stokes equations, respectively. Using GM-3 method as an illustration, we provide the algorithm of the method as follows: Set $(\mathbf{u}_0^0, p_0^0) = (\mathbf{u}(\mathbf{0}), p(0))$ and $(\mathbf{u}_1^0, p_1^0) = (\mathbf{u}_2^0, p_2^0) = (0,0)$. We compute $(\mathbf{u}^{n+1}, p^{n+1})$, $n \geq 0$, by solving the equations (4.4.8), (4.4.9) and (4.4.10) for $(\mathbf{u}_0^{n+1}, p_0^{n+1})$, $(\mathbf{u}_1^n, p_1^n)$ and $(\mathbf{u}_2^{n-1}, p_2^{n-1})$, respectively.

$$\text{For } n \geq 0, \begin{cases} \mathbf{s}_0^{n+1} = \begin{cases} \mathbf{u}_0^n, & \text{for } n = 0 \text{ or } n = 1, \\ \mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1}, & \text{for } n \geq 2, \end{cases} \\ \mathbf{nl}_0^{n+1} = B(\mathbf{s}_0^{n+1}), \qquad \mathbf{gd}_0^{n+1} = C_\Delta \mathbf{s}_0^{n+1}, \\ \dfrac{\mathbf{u}_0^{n+1} - \mathbf{u}_0^n}{\tau} + \overline{A}\mathbf{u}_0^{n+1} + \nabla p_0^n = \mathbf{f}^{n+1} - \mathbf{nl}_0^{n+1} - \mathbf{gd}_0^{n+1} - \overline{C}_\Delta \mathbf{u}_0^{n+1}, \\ p_0^{n+1} = p_0^n - \lambda \nabla \cdot \mathbf{u}_0^{n+1}, \\ \mathbf{du}_0^{n+1} = (\mathbf{u}_0^{n+1} - \mathbf{u}_0^n)/\tau, \quad dp_0^{n+1} = (p_0^{n+1} - p_0^n)/\tau. \end{cases} \tag{4.4.8}$$

$$\text{For } n \geq 1, \begin{cases} \mathbf{d^2 u}_0^{n+1} = (\mathbf{du}_0^{n+1} - \mathbf{du}_0^n)/\tau, \\ \mathbf{s}_1^n = \begin{cases} \mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1}, & \text{for } n = 1, \\ \mathbf{u}_0^n + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}, & \text{for } n \geq 2, \end{cases} \\ \mathbf{nl}_1^n = B(\mathbf{s}_1^n), \qquad \mathbf{gd}_1^n = C_\Delta \mathbf{s}_1^n, \\ \dfrac{\mathbf{u}_1^n - \mathbf{u}_1^{n-1}}{\tau} + \overline{A}\mathbf{u}_1^n + \nabla(p_1^{n-1} + dp_0^n) = -\dfrac{1}{2}\mathbf{d^2 u}_0^{n+1} - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\tau} \\ \qquad\qquad - \dfrac{\mathbf{gd}_1^n - \mathbf{gd}_0^n}{\tau} - \overline{C}_\Delta \mathbf{u}_1^n, \\ p_1^n = p_1^{n-1} + dp_0^n - \lambda \nabla \cdot \mathbf{u}_1^n, \\ \mathbf{du}_1^n = (\mathbf{u}_1^n - \mathbf{u}_1^{n-1})/\tau, \quad dp_1^n = (p_1^n - p_1^{n-1})/\tau. \end{cases} \tag{4.4.9}$$

$$\text{for } n \geq 2, \begin{cases} \mathbf{d^2 u}_1^n = (\mathbf{du}_1^n - \mathbf{du}_1^{n-1})/\tau, \quad \mathbf{d^3 u}_0^{n+1} = (\mathbf{d^2 u}_0^{n+1} - \mathbf{d^2 u}_0^n)/\tau, \\ \mathbf{s}_2^{n-1} = \mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-2}, \\ \mathbf{nl}_2^{n-1} = B(\mathbf{s}_2^{n-1}), \qquad \mathbf{gd}_2^{n-1} = C_\Delta \mathbf{s}_2^{n-1}, \\ \dfrac{\mathbf{u}_2^{n-1} - \mathbf{u}_2^{n-2}}{\tau} + \overline{A}\mathbf{u}_2^{n-1} + \nabla(p_2^{n-2} + dp_1^{n-1}) = -\dfrac{1}{2}\mathbf{d^2 u}_1^n + \dfrac{1}{6}\mathbf{d^3 u}_0^{n+1} \\ \qquad\qquad - \dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\tau^2} - \dfrac{\mathbf{gd}_2^{n-1} - \mathbf{gd}_1^{n-1}}{\tau^2} - \overline{C}_\Delta \mathbf{s}_2^{n-1}, \\ p_2^{n-1} = p_2^{n-2} + dp_1^{n-1} - \lambda \nabla \cdot \mathbf{u}_2^{n-1}, \\ \mathbf{u}^{n-1} = \mathbf{u}_0^{n-1} + \tau \mathbf{u}_1^{n-1} + \tau^2 \mathbf{u}_2^{n-1}, \quad p^{n-1} = p_0^{n-1} + \tau p_1^{n-1} + \tau^2 p_2^{n-1}. \end{cases} \tag{4.4.10}$$

The convergence of the **grad**-div splitting is done using a correction algorithm similar to the one implemented to approximate the nonlinear advection term, the "nonlinear ansatz". For instance, we recover the global **grad**-div term (right hand side of the

momentum equation) by summing all the momentum equations from each subproblem: $(4.4.8)|_{n=j}+\tau(4.4.9)|_{n=j+1}+\tau^2(4.4.10)|_{n=j+2}$ for any fixed positive integer $j>0$. Ignoring all terms except the **grad**-div term, we have the following:

$$\begin{aligned} & -C_\Delta \mathbf{s}_0^{j+1} - \tau\left(\frac{C_\Delta \mathbf{s}_1^{j+1} - C_\Delta \mathbf{s}_0^{j+1}}{\tau}\right) - \tau^2\left(\frac{C_\Delta \mathbf{s}_2^{j+1} - C_\Delta \mathbf{s}_1^{j+1}}{\tau^2}\right) \\ = \;& -C_\Delta \mathbf{s}_2^{j+1} \\ = \;& \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} u_0^{j+1} + \tau u_1^{j+1} + \tau^2 u_2^j \\ v_0^{j+1} + \tau v_1^{j+1} + \tau^2 v_2^j \end{bmatrix} \\ = \;& -\lambda \nabla\nabla\cdot(\mathbf{u}_0^{j+1} + \tau \mathbf{u}_1^{j+1} + \tau^2 \mathbf{u}_2^j) \\ = \;& -\lambda \nabla\nabla\cdot\mathbf{u}^{j+1} - \tau^2 \nabla\nabla\cdot(\mathbf{u}_2^{j+1} - \mathbf{u}_2^j). \end{aligned}$$

If the condition $\|\mathbf{u}_2^{j+1} - \mathbf{u}_2^j\| \leq c\tau$ is met, then the **grad**-div splitting produces a truncation error of order 3 (in time). This can be achieved with a suitable choice of the time step τ and stabilization parameter λ. The simplicity of this algorithm is similar to the one proposed by Guermond and Minev since no Jacobi or Gauss–Seidel methods is involved.

4.4.4 Numerical results

In this section, we conduct simple numerical test cases to compare various **grad**-div splitting approaches in terms of L^2-error (in time only) on velocity and pressure. This L^2-error is computed by taking the difference between the reference and numerical solutions on the same mesh at each time step for a span of time T. The reference solution is generated with GM-2 method with a very fine time step $\tau = 6.25 \times 10^{-5}$. All these are done using the two manufactured solutions used in Section 4.1 (Manufactured Solution I) and Section 4.2 (Manufactured Solution II) of Chapter 2.

For both manufactured solutions, we use the same domain, mesh, boundary and initial conditions as in Chapter 2. In this numerical test, we use only the 2^{nd}-order GM method. The extension to the 3^{rd}-order GM can be done in a straightforward manner but is not attempted here. We denote by GMS2-GS$_e$ the first variant of the **grad**-div splitting technique presented in Section 4.4.1 which combines GM-2 with a stopping criterion based on a fixed tolerance e and by GMS2-GS$_n$ the variant with a fixed number n of Gauss-Seidel iteration. We denote by GMS2-GM the variant of the **grad**-div splitting proposed in [24] (Section 4.4.2) and by GMS2-NLGD the one proposed in Section 4.4.3. Various stabilization parameters used for each of these **grad**-div splitting for Manufactured Solution I and II are summarized in Tables 4.5 and 4.6, respectively.

Table 4.5: Parameters used with different **grad**-div splitting techniques for the Manufactured Solution I.

Manufactured Solution I				
Method	τ	n	e	λ
GM-2	0.02	–	–	25.0
GMS2-NLGD	0.02	–	–	25.0
GMS2-GM	0.02	–	–	7.0
GMS2-GS$_e$	0.02	–	10^{-15}	25.0
GMS2-GS$_n$	0.02	3	–	25.0
GM-2	0.002	–	–	6.5
GMS2-NLGD	0.002	–	–	6.5
GMS2-GM	0.002	–	–	6.5
GMS2-GS$_e$	0.002	–	10^{-15}	6.5
GMS2-GS$_n$	0.002	3	–	6.5

Table 4.6: Parameter used with different **grad**-div splitting techniques for the Manufactured Solution II.

Manufactured Solution II				
Method	τ	n	e	λ
GM-2	0.02	–	–	3.13
GMS2-NLGD	0.02	–	–	1.0
GMS2-GM	0.02	–	–	1.0
GMS2-GS$_e$	0.02	–	10^{-15}	1.0
GMS2-GS$_n$	0.02	2	–	3.13
GM-2	0.002	–	–	0.75
GMS2-NLGD	0.002	–	–	0.75
GMS2-GM	0.002	–	–	1.0
GMS2-GS$_e$	0.002	–	10^{-15}	0.75
GMS2-GS$_n$	0.002	2	–	0.75

For Manufactured Solution I, the time evolution of the numerical error (in time only) on velocity and pressure is shown in Figure 4.6. Here, we also include GM-2 method for benchmarking purpose. For $\tau = 0.02$, we observe that the GMS2-GS$_n$ method produces the largest error on velocity and pressure followed by GMS2-GM, GMS2-NLGD and GMS2-GS$_e$ methods. The GMS2-GS$_e$ method converges to the solution of GM-2 method by setting the tolerance $e = 10^{-15}$ at the expense of a very large CPU time.

For $\tau = 0.002$, error curves on velocity computed with all **grad**-div splitting are in good agreement with the one computed by GM-2 method except for GMS2-GM and GMS2-GS$_n$ methods. This suggests that the GMS2-GS$_n$ method requires more Gauss–Seidel iterations, (e.g., $n > 3$) to produce better results on velocity. In fact, the number of Gauss–Seidel iterations required with $e = 10^{-15}$ are 29 and 22 iterations for subproblems 1 and 2, respectively, if $\tau = 0.02$ while 8 and 7 iterations are required in subproblems 1 and 2, respectively, if $\tau = 0.002$.

The error on pressure computed by all **grad**-div splitting techniques are in good agreement with the one computed by GM-2 method. We notice that the GMS2-NLGD method is the most efficient since it is both accurate and requires the least CPU time among all **grad**-div splitting techniques.

Numerical oscillations (in time) were produced at startup but were quickly damped in 20–30 time steps. This shows that the choice of λ for each of the **grad**-div formula is almost optimal in the sense that the numerical error from over-stabilization is minimized and the solution is very accurate for $t > 0.5$. Larger λ can be used to damp these oscillations more quickly but the risk of having numerical errors due to over-stabilization increases.

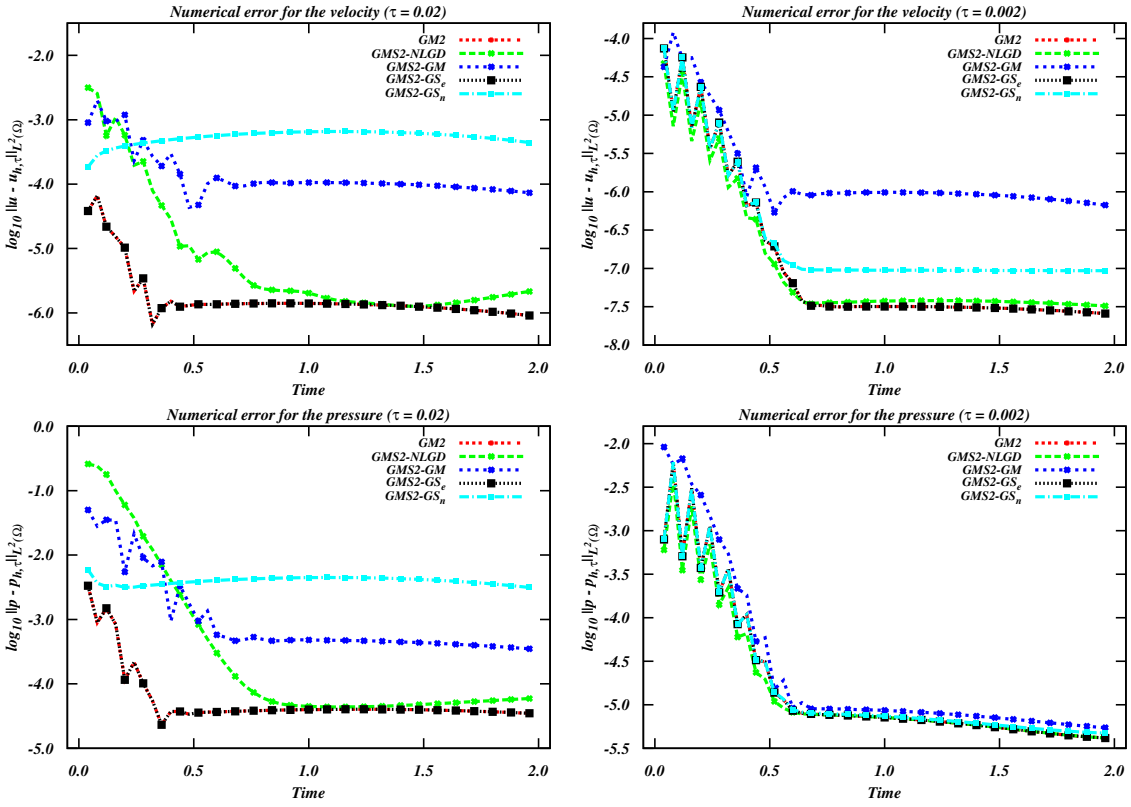

Figure 4.6: Time evolution of the L^2-error on velocity and pressure for the Manufactured Solution I (see Chapter 2) with various **grad**-div splitting strategies.

Different numerical results are observed for Manufactured Solution II. For instance, GMS2-GM produces the largest numerical error followed by GMS2-NLGD, GMS2-GS$_n$ and GMS2-GS$_e$ (see Figure 4.7). In particular, we observed that GMS2-NLGD method is not as accurate as GMS2-GS$_n$. The GMS2-GS$_n$ method produces very small

error for both time steps $\tau = 0.02$ and 0.002 with only two Gauss–Seidel iterations. However, the efficiency and the resulting error in GMS2-NLGD method could be as competitive as GMS2-GS$_n$ method if a smaller time step, (e.g., $\tau = 0.001$) is chosen.

In Manufactured Solution II, the number of Gauss–Seidel iterations required with $e = 10^{-15}$ are 25 and 42 iterations in subproblems 1 and 2, respectively, if $\tau = 0.02$, while 3 iterations are required in subproblems 1 and 2 if $\tau = 0.002$. We did not carry out any efficiency analysis for these methods applied to our manufactured solutions as we did in Chapter 2, where the CPU time to reach a certain error is investigated.

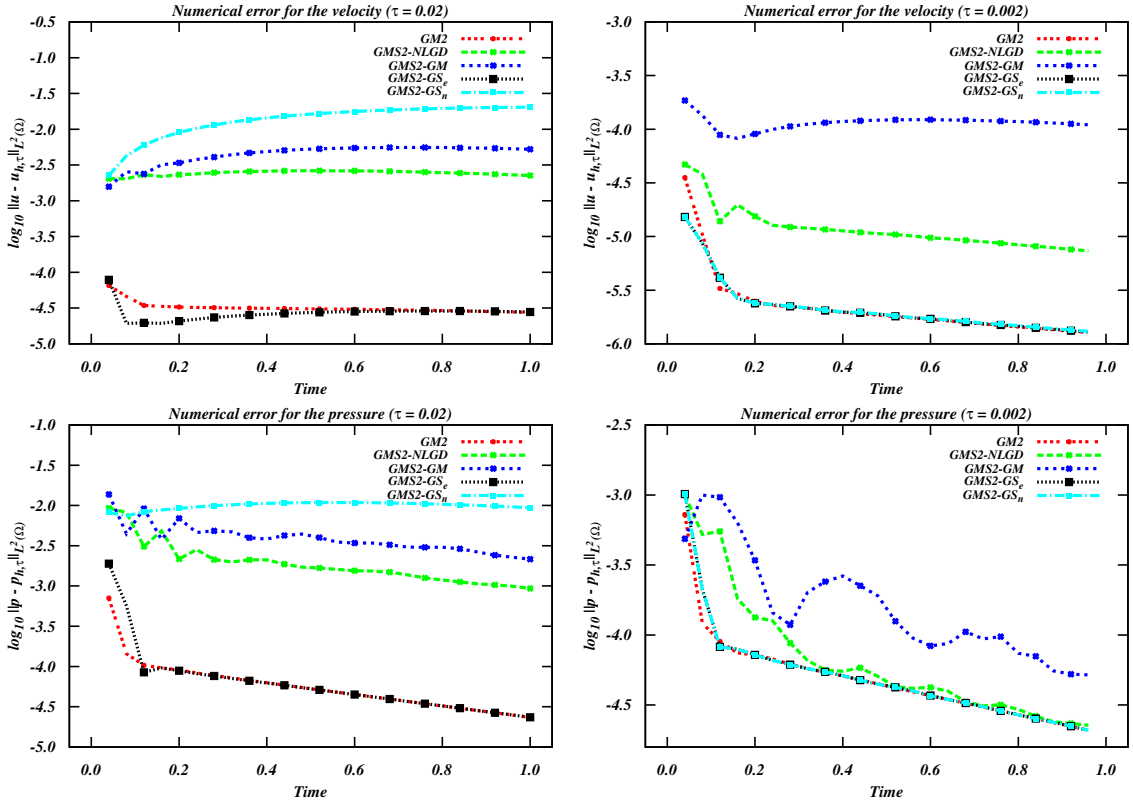

Figure 4.7: Time evolution of the L^2-error on velocity and pressure for the Manufactured Solution II (see Chapter 2) with various **grad**-div splitting strategies.

4.5 Concluding Remarks

The first part of this chapter has been devoted to complete the justification of the GM and GM-SRM methods introduced in Chapter 2. For a fixed λ in GM method and

fixed α_1, α_2 in GM-SRM method, we showed that these methods converge to SBDF and DC methods with an additional **grad**-div stabilization term under $\mathbb{P}_2$-$\mathbb{P}_1$ finite element approximation. However, the optimal choice of stabilization parameters in GM and GM-SRM methods is still an open problem. We only resort to a method of trial and error to obtain a nearly optimal value for the stabilization parameter (by comparing to the exact solution or a reference solution obtained with reasonably fine mesh and time step using the most efficient SBDF-3 method). This provides insight on the choice of the stabilization parameters in many applications of 2D flows, but cannot be used in practice since no exact or reference solution is available.

The second part of this chapter proposed several numerical improvements to reduce the numerical error in DC, GM and GM-SRM methods, for instance new extrapolation formulae for the nonlinear term. The NL_{22} and NL_{233} formulae are a good trade-off in terms of stability and accuracy for 2^{nd}- and 3^{rd}-order methods, respectively. The final part of this chapter introduced **grad**-div splitting techniques which reduce the complexity and size of the resulting linear systems in GM and GM-SRM methods. We proposed a new **grad**-div splitting which is based on a nonlinear ansatz and works satisfactorily well with no Gauss–Seidel iteration. It is preferable over the one proposed in [24] (which was less accurate but CPU efficient) and Gauss–Seidel iterations with small tolerance (which was very accurate but CPU intensive).

Chapter 5

3D Computations with Semi-Implicit Methods

In this chapter, we illustrate how semi-implicit methods perform to compute 3D flows. Three-dimensional flows arise naturally in many practical and industrial applications. As 3D computations typically involve many unknowns, the application of our semi-implicit methods becomes more relevant. First of all, we require the global matrix to be assembled only once. This also allows the preconditioner to be built only once when an iterative method is used at each time step. Fully implicit methods, such as BDF and operator splitting, may fall short in terms of efficiency, especially when computing 3D unsteady flows, since reassembling the matrix and recomputing a preconditioner becomes expensive.

Undeniably, the required CPU time and computer memory can be exhaustive when 3D computations are carried with fine meshes and small time steps are required for the numerical solution of Navier–Stokes equations with high Reynolds numbers. In the realm of computational fluid dynamics (CFD), this is known as direct numerical simulations (DNS). For certain fluid simulations, DNS is deemed to be less practical even with the best available supercomputers. This limitation can be mitigated by resorting to Large Eddy Simulation (LES), where only the mean flow is computed on coarser meshes while turbulent scales that occur on a finer grid are resolved using subgrid or turbulence closure models (see for instance [36, 38, 52]). On the other hand, DNS does not require such subgrid modeling. Using higher-order methods both in time and space, DNS may not have to rely on overly fine meshes and small time steps to produce accurate results. This potentially makes DNS more appealing and promising for certain applications.

In our work, the computations are done with FreeFEM++ in sequential mode. Parallel implementations based on domain decomposition method (DDM) and similar

techniques can be applied with our time-stepping methods, but none were attempted here. Computations in a full parallel mode with high performance computer (HPC) can handle up to hundreds of millions of unknowns depending on the computing resources. Moreover, the computer memory offered in HPC goes up to a thousand of Gigabytes (Gb) which is nowhere to be compared with the memory available on personal computers. Our 3D computations in sequential mode done on personal computers can handle up to 5 millions of unknowns. This will be enough to showcase the performance of our time-stepping methods.

The 3D computations done in this chapter split into two parts. For the first part, we showcase 3D computations with GM method and a **grad**-div splitting algorithm to handle the larger number of unknowns required in 3D lid-driven cavity flows. Two experiments for lid-driven cavity are proposed. The first one involves steady solutions with the Reynolds number at 100, 400 and 1 000, and the other involves an unsteady 3D flow by setting $Re = 1\,970$. For the second part, we compute two closely related unsteady flows with GM-SRM method, one being the 3D flow around the circular cylinder at $Re = 100$ and the other, the 3D flow around the sphere at $Re = 300$. We noted that GM and GM-SRM methods are preferable over SBDF and DC methods since the resulting linear systems with the first two methods are smaller. We make qualitative and quantitative comparisons of our results with the values reported in the literature. From a qualitative standpoint, we compare the overall flow patterns such as vortex shedding patterns and streamlines. From a quantitative standpoint, we compare lift and drag coefficients, and the Strouhal numbers of the periodic flows. We devote the last section of this chapter to an efficiency comparison of all the methods studied, gathered from all 2D and 3D test cases presented in this thesis, in terms of CPU time per degree of freedom per time step.

5.1 Special considerations for large scale computations

Unique characteristics in the methods presented can be exploited to tackle different types of test problems in the most efficient manner with available computing resources. For instance, if a saddle point system is not too large (typically less than 700 000 unknowns), computation can be very efficient with SBDF and DC methods using a direct linear solver. For a slightly larger problem (between 700 000 to 900 000 unknowns), GM and GM-SRM can be used since these methods requires only the solution of linear systems with relatively smaller symmetric and positive definite matrices than the former saddle point problems. Nonetheless, the factorization step in the direct linear solvers may require large RAM, beyond the memory available. Therefore, iterative solvers are recommended to solve larger problems with more

than 1 million of unknowns since they require far less memory. For an exhaustive discussion on iterative solvers, we refer the reader to several main references [3, 51]. Iterative solvers like the Conjugate Gradient method can be implemented and work efficiently with GM and GM-SRM methods since the resulting linear systems involve symmetric and positive definite matrices. Though it is possible, we did not attempt computations using SBDF and DC methods with any iterative linear solver since the global matrix associated with saddle point problem is poorly conditioned and the preconditioner used, (e.g., HIPS [16]) required very long time to setup at the beginning of the execution.

In this chapter, we consider only GM and GM-SRM methods with 2^{nd}-order of accuracy to carry out all 3D computations since these methods produce a good accuracy on velocity and pressure in time with the built-in defect correction strategy. The accuracy and efficiency of several methods based on defect correction strategy, (e.g., DC, GM and GM-SRM methods) have been demonstrated in Chapter 2. The 3^{rd}-order GM and GM-SRM methods can be used to achieve sufficiently small error in time on a fixed mesh at the expense of larger CPU time. However, the global error in our 3D computations is predominantly induced by the spatial resolution since the mesh employed may not be sufficiently fine. As a result, the advantage of using higher-order in time cannot be showcased to its full potential.

5.1.1 Mesh generation

All the tetrahedral meshes for the purpose of computing 3D flows can be generated using two methods. First, it can be done using a program called *TetGen*. This program was developed by Hang Si at Weierstrass Institute for Applied Analysis and Stochastics (WIAS) [57]. *TetGen* is based on Delaunay tetrahedralizations and includes advanced features, such as mesh refinement and various adaptive strategies. This software can be used either as an executable standalone program or as a library linked to various numerical softwares for solving PDEs, (e.g., FreeFEM++, OpenFoam, etc). Another 3D mesh tool available with FreeFEM++ (only) is called *msh3*. This mesh tool has its own unique approach to generate a 3D mesh based on a function called *buildlayers*. This function allows users to extend and manipulate a 2D mesh to generate a 3D mesh by extrusion of the 2D mesh in the simplest way.

With limited computing resources, computations of 3D flows require a delicate planning since the number of unknowns can easily be overwhelming. To produce an "optimal" number of unknowns while achieving a reasonable accuracy in space, finer mesh is only required in a few specific areas. We do not consider dynamic mesh adaptation to avoid any matrix rebuild and recalculation of the required preconditioner, which may hinder the efficiency of semi-implicit methods. In our case, a finer mesh

is generated based on an educated guess, which is prescribed within a sub-region of a domain. For instance, for the 3D flow around the circular cylinder, a finer mesh is generated around and behind the cylinder to capture the formation of the boundary layer, the recirculation and vortex shedding behind the cylinder.

5.1.2 High Performance Sparse Linear Solvers

With any finite difference, finite volume and finite element methods, the number of unknowns increases dramatically as the mesh is refined, especially in 3D setting. Moreover, with finite element approximations, the unknowns increase proportionally to the degree of the polynomials (e.g., $\mathbb{P}_k$, $k = 0, 1, 2, \ldots$) used for the Lagrange basis functions. For the same physical mesh, the number of unknowns is larger if the continuous piecewise elements are replaced with the discontinuous elements. Therefore, the number of unknowns has to be controlled wisely to have a good balance between the computational efficiency and numerical accuracy of the results.

The $\mathbb{P}_2$-$\mathbb{P}_1$ element produces the smallest number of unknowns in the class of Taylor–Hood elements. On the other hand, the $\mathbb{P}_2$-$\mathbb{P}_1$ element is more expensive than the $\mathbb{P}_1$-$\mathbb{P}_1$ element used in conjuction with stabilization methods when computing 3D flows on the same mesh [34, 59]. Still, the $\mathbb{P}_2$-$\mathbb{P}_1$ element is preferable over the $\mathbb{P}_1$-$\mathbb{P}_1$ element since the highest order of accuracy of the $\mathbb{P}_2$-$\mathbb{P}_1$ method eventually makes this one competitive over $\mathbb{P}_1$-$\mathbb{P}_1$ stabilized method.

The effort to improve the convergence of iterative solvers using preconditioner is immense and well documented in the literature (see e.g., [3, 4, 5, 12, 51, 69]). The underlying concept of preconditioner is simple. Suppose that we have a linear system $\mathcal{A}x = b$ with a global matrix $\mathcal{A}$. Whenever the condition number of a matrix $\mathcal{A}$ is large [48], (i.e., $\kappa(\mathcal{A}) \gg 1$) a preconditioner $\mathcal{P}^{-1} \approx \mathcal{A}^{-1}$ is usually required in the hope of getting a new matrix $\mathcal{P}^{-1}\mathcal{A}$ with a small condition number $\kappa(\mathcal{P}^{-1}\mathcal{A}) \approx 1$, so that the linear system $\mathcal{P}^{-1}\mathcal{A}x = \mathcal{P}^{-1}b$ could be solved more easily with iterative solvers. Fewer iterations are then required to satisfy $\|r\| := \|\mathcal{P}^{-1}(\mathcal{A}x - b)\| < \text{tol}_{\text{res}}$ for a small $\text{tol}_{\text{res}} > 0$. Often, the preconditioner $\mathcal{P}^{-1}$ does not have to be evaluated explicitly. More details about the construction of preconditioners can be found in the literature mentioned above.

In this chapter, we use the library for a numerical linear solver called HIPS (Hierarchical Iterative Parallel Solver) which has a built-in preconditioner based upon multilevel Incomplete Factorization (ILU) [16, 17]. HIPS is already linked to FreeFEM++. Only two types of Krylov-based iterative solvers are available in HIPS; namely, the Generalized Minimal Residual (GMRes) and Preconditioned Conjugate Gradient (PCG). Since the matrix involved in GM and GM-SRM methods is symmetric and positive definite, the PCG method is sufficient and more efficient to solve the linear system.

There are few key parameters that have to be tuned properly to control the convergence of HIPS, among them: the number of iterations before Krylov restart, a threshold to control the fill-in in ILUT and a relative residual norm. For instance, requiring the ILUT preconditioner to be closer to the actual LU factorization of the global matrix $\mathcal{A}$ often requires larger memory and longer time for the start-up since this involves a lot of fill-in within ILU [51]. On the other hand, less fill-in in ILU results in a significant faster HIPS startup process but in a larger number of iterations to achieve the same relative residual norm.

The parameter settings that we used in our numerical experiments are obtained through trial and error. We did not attempt any intensive study on the optimal parameter values to produce the most CPU efficient and accurate solutions for HIPS. We only implement the set of parameters which, in our opinion, give an ideal gain in terms of the convergence of PCG and CPU time required for start-up. These parameter values will be provided whenever HIPS is applicable.

5.2 3D lid-driven cavity flow

This test case is sometimes known as the lid-driven flow in a cube. Several numerical experiments for the 3D lid-driven cavity can be found in [29, 30, 46, 55] and references therein. The domain is given by $\Omega = (0, 1)^3$. To build the mesh, we first set a 2D lid-driven cavity in the xz-plane. We extend the geometry to 3D in the y-direction using *msh3* with a thickness of 1 unit length to obtain a uniform tetrahedral mesh of size $50 \times 50 \times 50$. The domain is discretized in about 750 000 nearly identical tetrahedra. This gives an element diameter $0.020\,000 \leq h_K \leq 0.028\,284$. Using the $\mathbb{P}_2$-$\mathbb{P}_1$ finite element gives a total of 3 090 903 unknowns for velocity and 132 651 unknowns for pressure. The number of unknowns for the 3D lid-driven cavity easily reaches millions even with a seemingly 'coarse' mesh resolution of $50 \times 50 \times 50$. Computing this flow on fine meshes is therefore particularly challenging.

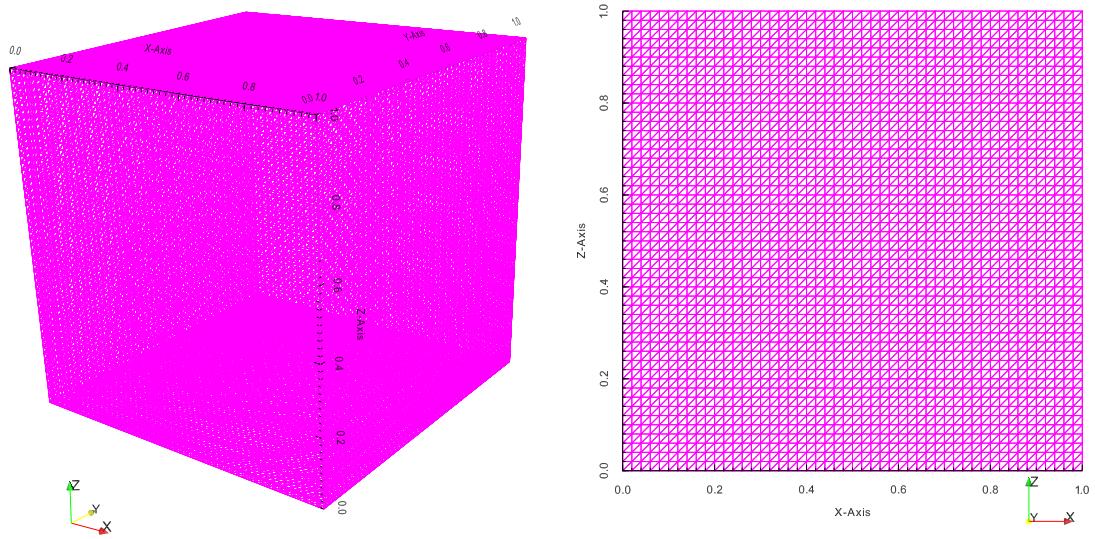

Figure 5.1: The mesh used to compute the 3D lid-driven flow: 3D mesh of the domain $\Omega \in (0,1)^3$ (left) and 2D 50×50 mesh used to generate the 3D mesh as viewed in the xz-plane (right).

For the boundary condition, we set the following:

$$\mathbf{u} = \mathbf{u}(x,y,z) = \begin{cases} (1,0,0)^\intercal, & 0 < x,y < 1 \text{ and } z = 1, \\ (0,0,0)^\intercal, & \text{on other boundaries.} \end{cases} \tag{5.2.1}$$

We mention that the velocity is prescribed to be null along the top edges, in the spirit of the boundary condition used for the 2D lid-driven cavity in Chapter 2. This type of boundary setting is called the *watertight* lid-driven cavity. We did not attempt a numerical comparison between *watertight* and *leaky* 3D lid-driven cavity flows where *leaky* means that $\mathbf{u} = (1,0,0)^\intercal$ is prescribed along the top edges. For 2D lid-driven cavity, the comparison of these two sets of boundary conditions has been done in Appendix C.1. Our preference for the *watertight* cavity setting here is solely based on the fact that more realistic numerical results were produced in the *watertight* 2D lid-driven cavity for a moderate Reynolds number, e.g., $Re = 8\,500$.

To handle the resulting millions of unknown, we use GM-2 method in conjunction with the **grad**-div splitting strategy presented in Chapter 4. To ensure the convergence of the **grad**-div splitting, we use two different approaches, i.e., the "nonlinear ansatz" or Gauss–Seidel method depending on the test cases to be presented below. In either case, the stabilization parameter $\lambda = 0.3$ is set, which is sufficient to enforce the mass conservation in this test case. Since $\kappa(\mathcal{A}) = \mathcal{O}(\lambda)$ for a fixed mesh and time step,

the resulting matrix is not so poorly conditioned. Hence, the ILUT preconditioner in HIPS requires little fill-in and less memory. The HIPS preconditioner is set as follows for this test case (refer to the FreeFEM++ manual [28] for definitions):

- Strategy use for solving – **Iterative**,
- Krylov methods – **PCG**,
- Maximum of iterations in outer iteration – **300**,
- Krylov subspace dimension in outer iteration – **50**,
- Matrix – **Symmetric**,
- Pattern of matrix – **Symmetric**,
- Relative residual norm – $\mathbf{10^{-13}}$,
- Numerical threshold in ILUT for interior domain – $\mathbf{10^{-6}}$,
- Numerical threshold in ILUT for Schur preconditioner – $\mathbf{10^{-6}}$.

All other parameters are set to default values. The given parameters produce an "optimal" balance between the CPU time for the startup and the convergence of PCG. On average, it takes about 550 seconds of CPU time to produce the HIPS preconditioner and the PCG needs at most 4 iterations to reach a relative residual norm of 10^{-13} when solving the linear systems associated with the scalar parabolic problem for the velocity and at most 3 iterations for the pressure update. To reduce the number of iterations in PCG to 2 for the velocity, we have experimented with smaller numerical threshold in ILUT, for instance 10^{-10}, but this requires more CPU time for the startup (typically $>$ 3 hours). With this larger threshold, the startup process may exhaust the computer memory; e.g., it consumes more than 40Gb of RAM, similar to what is required with the direct solver MUMPS for the same number of unknowns.

Increasing the time step and the stabilization parameter λ may also induce larger CPU time and memory requirements during the startup, since larger time step and λ produce a stiffer matrix for the scalar parabolic problem. Whenever a larger time step or a large λ is required, the numerical thresholds in ILUT have to be increased accordingly. We use the same domain, mesh, HIPS parameters and boundary conditions for the steady and unsteady flows in the 3D lid-driven cavity.

5.2.1 Steady 3D lid-driven cavity

In this section, we verify our methods by computing several steady flows in the cubic 3D lid-driven cavity, by choosing Reynolds numbers at 100, 400 and 1 000. These test

cases have been studied intensively in [30, 46, 55]. We conduct numerical verification in several ways. We make comparison with results from those articles in terms of streamlines and velocity fields projected on xy-, xz- and yz-planes.

For numerical efficiency, we apply GM-2 method with the **grad**-div splitting based on the "nonlinear ansatz" (4.4.8)-(4.4.10) presented in Section 4.4.3. With this technique, the linear systems from the momentum equation (with all velocity terms) are reduced to few scalar parabolic problems, one for each velocity component, and consists in 1 030 301 unknowns to be solved one at a time. While reaching about one million of unknowns, iterative linear solvers are preferable over direct linear solvers. Since our goal is to capture steady solutions, (i.e., let $\frac{d\mathbf{u}}{dt} \to 0$) the lack of accuracy in time due to the **grad**-div splitting at each time step does not have a significant impact as when computing unsteady flows.

We note that the constructed linear systems from the momentum equations become very stiff (or equivalently the resulting matrices are poorly-conditioned) for the iterative solver when both the time step and Reynolds number increase. It is also known that higher Reynolds numbers impose weaker dampening on the solution to reach steady state. Here, we adopt a heuristic numerical setting in order to maintain "similar stiffness" of these linear systems. For instance, for every increase by a factor of m of the Reynolds number, we decrease the time step by the same factor of m. We compute the flows at $Re = 100$, 400 and 1000 with the time steps $\tau = 0.02$, 0.005 and 0.002, respectively. We assume that we have reached a steady flow whenever $\frac{1}{\tau}\|\mathbf{u}_h^{n+1} - \mathbf{u}_h^n\|_{L^2(\Omega)} < 5 \times 10^{-5}$. We then consider $\mathbf{u}_h^{n+1}$ to be the steady state velocity field. To reduce the total CPU time, these computations are initiated using a steady solution of the same experiment with the same Reynolds number but on a coarser $30 \times 30 \times 30$ mesh. These steady solutions are then interpolated linearly on the $50 \times 50 \times 50$ mesh.

Table 5.1 shows the statistics regarding the time and number of time steps required to reach a steady state. It is observed that the number of time steps required to reach steady state increases dramatically with the Reynolds number. This is expected. We note that there are a few additional advantages when a smaller time is used to handle steady flows with high Reynolds numbers. First, smaller time step produces better stability in terms of CFL condition since GM-2 is a semi-implicit method (conditionally stable). Second, our strategy does not involve Jacobi/Gauss–Seidel iteration to ensure the convergence of **grad**-div splitting. Our numerical computation shows that a smaller time step may also help to ensure the convergence of **grad**-div splitting at each time step as well.

Table 5.1: The steady state in the lid-driven cavity.

Re	**Time step τ**	**No. of steps**	**Time unit T**	$\frac{1}{\tau}\|\mathbf{u}-\mathbf{u}_h\|_{L^2(\Omega)}$
100	0.020	338	6.760	4.93599×10^{-5}
400	0.005	1652	8.260	4.99233×10^{-5}
1000	0.002	14051	28.102	4.99983×10^{-5}

Table 5.2 provides the statistics regarding the performance of the HIPS solver for all test cases, the second column gives the total startup time which includes the CPU time (in second) required to compute the two HIPS preconditioners; i.e., one for the matrix associated with the momentum equation and another for the mass matrix associated with continuity equation. The third column in the table provides the average CPU time required to compute one time step while the last column gives us the fill-in ratio of the computed preconditioner matrices. The fill-in ratio corresponds to the number of non-zero entries in the preconditioner over the number of non-zero entries in the original matrix before factorization.

Higher fill-in ratio corresponds to more efficient ILU preconditioner and lead to better convergence of the iterative solver. The observed CPU time per time step is more or less the same among the three test cases. However, the total CPU time to achieve a steady solution increases dramatically with the highest value recorded was for $Re = 1\,000$, followed by the computations at $Re = 400$ and $Re = 100$. We also note that the fill-in ratio is the highest for the computation with smallest Reynold number. We conjecture that the largest time step $\tau = 0.02$ used in this particular test case produces a stiffer linear system despite having the smallest Reynolds number among the three cases.

Furthermore, the stiffness induced by increasing the Reynolds number is less pronounced than the stiffness induced by larger time steps. From our computational result for a fixed mesh, we suspect that the condition number of the global matrix may behave like $k(\mathcal{A}) \sim \mathcal{O}\big(\frac{\tau^\alpha}{Re^\beta}\big)$, where $0 < \beta < \alpha < 1$, but getting the exact value of α and β is still an open problem.

Table 5.2: Statistics about HIPS linear solver and preconditioner when computing the steady state solution for the lid-driven cavity.

Re	**Startup (s)**	**CPU/time step**	**Fill-in Ratio**	**Total CPU (hrs)**
100	683.34	230.64	12.1397	21.84
400	431.01	228.04	9.4989	104.76
1000	546.09	227.14	10.8785	886.69

For comparison purposes, Figure 5.2, 5.3 and 5.4 show the projected 3D streamlines for the lid-driven flows plotted on various cross-sections of the cube. Projected

streamlines on a xz-plane are defined as the streamlines for the velocity field (u, w), where $\mathbf{u} = (u, v, w)$ is the velocity field for the 3D flow. A similar definition applies to the projected streamlines on yz-planes. We observed that these streamlines are perfectly in agreement with Shankar and Deshpande [55] (shown in Figure 14), both in terms of detailed features of the projected streamlines and the location of the center for each vortex.

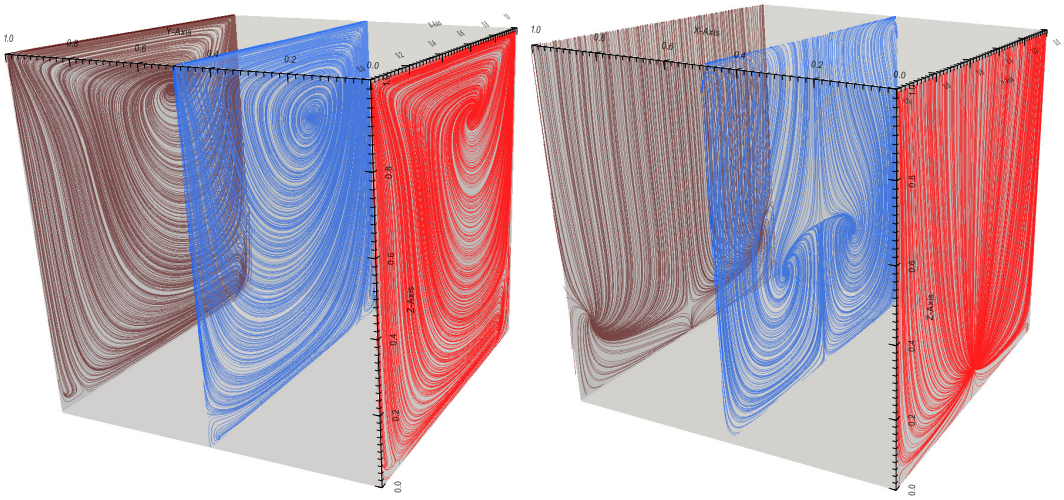

Figure 5.2: 3D lid-driven cavity at $Re = 100$: At the left, the streamlines are projected on xz-planes at $y = 0.02$ (red), $y = 0.5$ (blue) and $y = 0.98$ (brown). At the right, the streamlines are projected on yz-planes at $x = 0.02$ (red), $x = 0.5$ (blue) and $x = 0.98$ (brown).

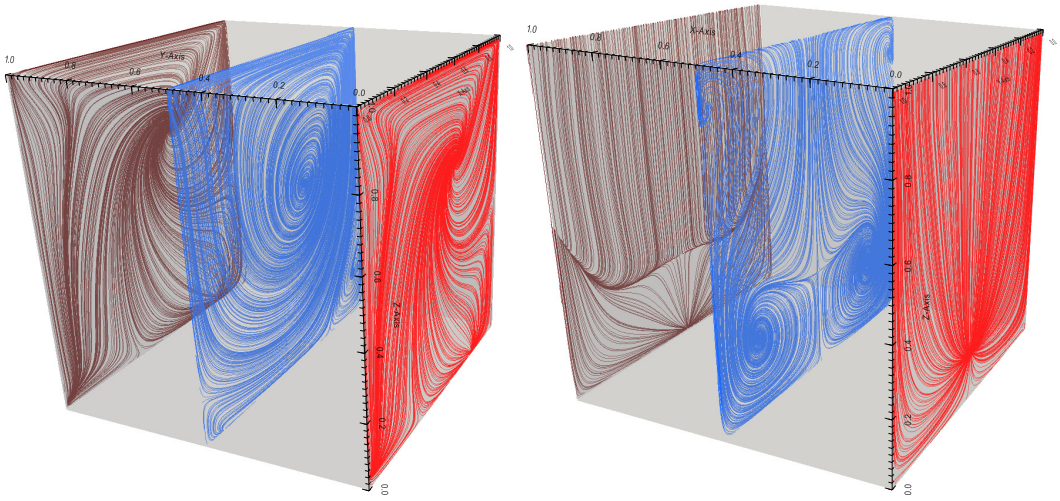

Figure 5.3: 3D lid-driven cavity at $Re = 400$: At the left, the streamlines are projected on xz-planes at $y = 0.02$ (red), $y = 0.5$ (blue) and $y = 0.98$ (brown). At the right, the streamlines are projected on yz-planes at $x = 0.02$ (red), $x = 0.5$ (blue) and $x = 0.98$ (brown).

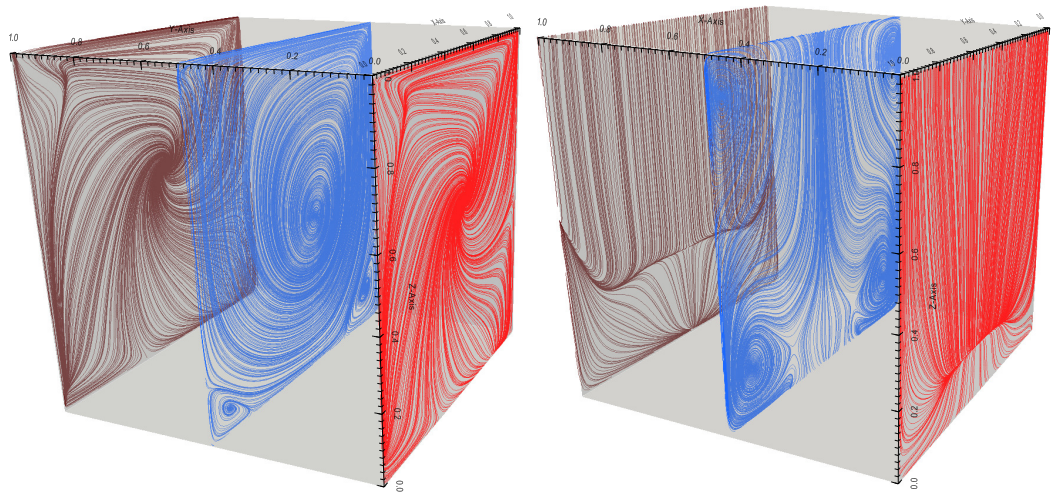

Figure 5.4: 3D lid-driven cavity at $Re = 1000$: At the left, the streamlines are projected on xz-planes at $y = 0.02$ (red), $y = 0.5$ (blue) and $y = 0.98$ (brown). At the right, the streamlines are projected on yz-planes at $x = 0.02$ (red), $x = 0.5$ (blue) and $x = 0.98$ (brown).

Figure 5.5, 5.6 and 5.7 show the velocity fields captured on the xz-plane ($y = 0.5$), yz-plane ($x = 0.5$) and xy-plane ($z = 0.5$) of the cube at $Re = 100$, 400 and 1 000,

respectively. Our solutions are in good qualitative agreement with the results documented in Pan and Glowinski [46] at $Re = 100$ (Figure 15), 400 (Figure 16) and 1 000 (Figure 17) and in Zunic et al. [72] for $Re = 100$ and 1 000. These results are also supported by other references cited therein.

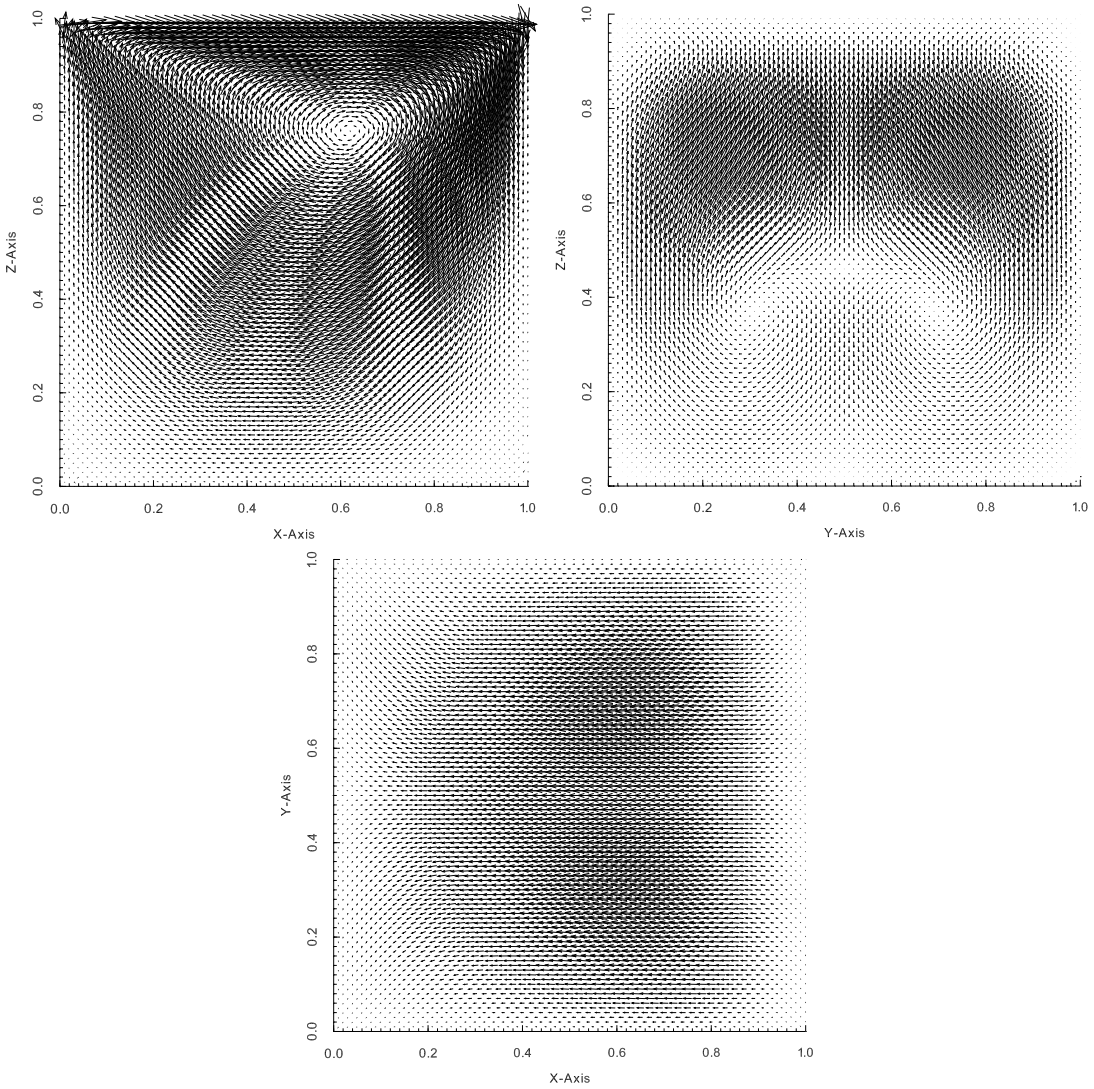

Figure 5.5: 3D lid-driven cavity at $Re = 100$: Velocity field at steady state on the midplanes at $y = 0.5$ (top left), $x = 0.5$ (top right) and $z = 0.5$ (bottom).

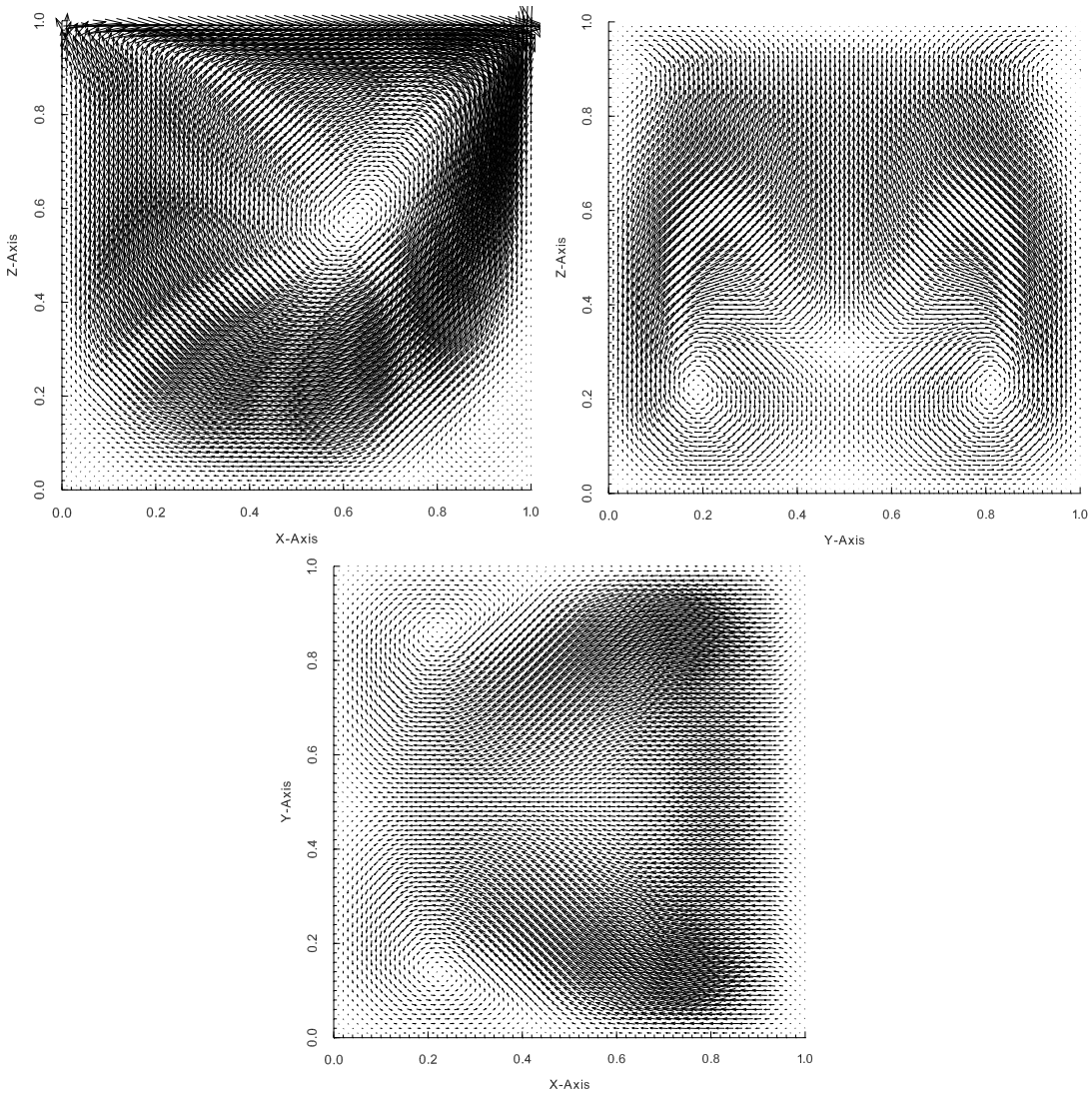

Figure 5.6: 3D lid-driven cavity at $Re = 400$: Velocity field at steady state on the midplanes at $y = 0.5$ (top left), $x = 0.5$ (top right) and $z = 0.5$ (bottom).

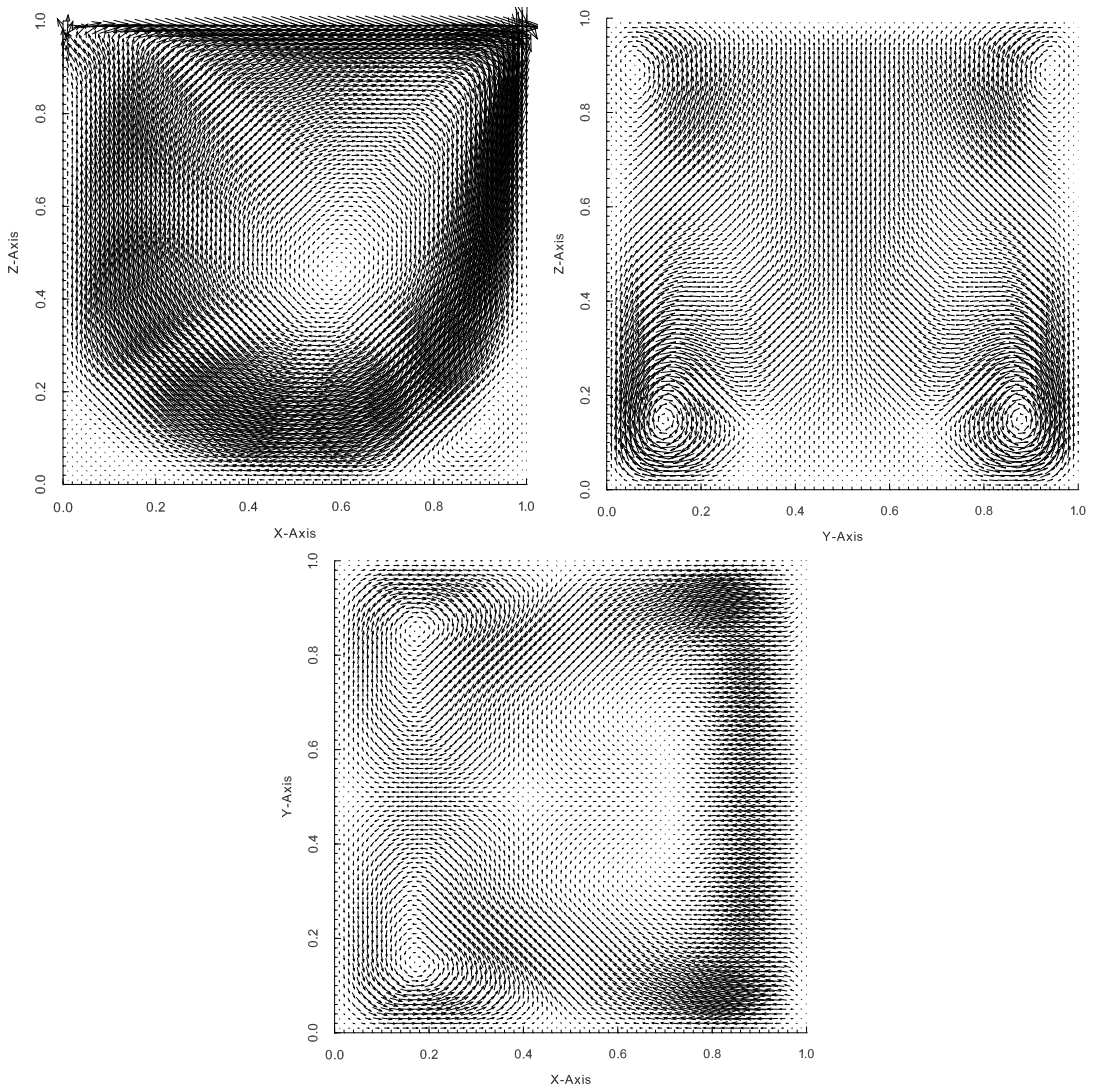

Figure 5.7: 3D lid-driven cavity at $Re = 1000$: Velocity field at steady state on the midplanes at $y = 0.5$ (top left), $x = 0.5$ (top right) and $z = 0.5$ (bottom).

The GM-2 method with **grad**-div splitting performs satisfactorily to reproduce qualitatively these steady flows. In term of performance, the global time $t \approx 28$ time units required to reach a steady state solution at $Re = 1000$ starting from the solution of Stokes equations is considerably shorter than that in [46]. For instance, their computations required $T \approx 51$ time units to reach a steady solution at $Re = 1\,000$ with a time step $\tau = 0.001$. The difference of final time T to reach a steady solution is due in part to the initial condition used since Pan and Glowinski [46]; namely, a periodic solution that requires a regularization mechanism at the upper left and right corner

of the cavity. On the other hand, the stopping criterion used in their experiment, $\frac{1}{\tau}\|\mathbf{u}_h^{n+1} - \mathbf{u}_h^n\|_{L^2(\Omega)} < 10^{-4}$, is slightly larger than the value $tol = 5 \times 10^{-5}$ that was used.

5.2.2 Unsteady 3D lid-driven cavity

Now, we turn to the unsteady case. To induce an unsteady flow, we fix $Re = 1\,970$ to produce an oscillatory flow since this Reynolds number is over the critical value for a Hopf bifurcation ($Re_{\text{cri}} \approx 1\,914$ for the cubic 3D lid-driven cavity) [39]. We use GM-2 method with a similar **grad**-div splitting to solve this problem. However, we employ the Gauss–Seidel method for a faster convergence of the **grad**-div splitting to ensure the accuracy of the solution at each time step. The algorithm is given in (4.4.2)–(4.4.4) of Section 4.4.1. To maintain a reasonable CPU time, we do at most two Gauss–Seidel iterations for each subproblem of the GM-2 method, with a time step τ of 0.005.

In this test case, the computation is initialized from a state of rest; i.e., $(\mathbf{u}, p) = (0, 0)$ at time $t = 0$ in the domain $\Omega = (0, 1)^3$ with boundary forcing given in (5.2.1). It takes about 14 days of CPU time to obtain a developed flow (but not yet periodic) on a i7-3 770 3.40 GHz desktop computer with about 15.7GB of RAM. The memory usage is considered small for such 3D computation which consists in a total of 3 090 903 unknowns for velocity and 132 651 unknowns for pressure. Here, we only present partial results in terms of the fully developed periodic flow since reproducing a fully periodic flow as in 2D requires a large time t and consequently a great deal of CPU time in sequential mode.

Figure 5.8 shows the time-evolution of the streamlines, colored by magnitude of the velocity. The streamlines illustrate the growth of the 3D primary (center) and secondary (bottom right and left) vortices starting from the state of rest. GM-2 method with **grad**-div splitting captures well the formation of the major vortex and other secondary vortices that represent the physically sound transient solutions from state of rest. At least, these flow patterns mimic closely the ones observed in the 2D cavity. Our computations have not yet reached a periodic state. A successful computation is shown to have a periodic solution with a single mode whose frequency is about 0.61 [39]. This test case requires enormous amount of CPU time to reach periodic flow. Moreover, a good accuracy in the space discretization is paramount when it comes to reproducing the right frequency for this flow, likely beyond what we used in our grid resolution. To improve this, more efficient iterative solver and preconditioner are needed to handle the exorbitant number of degrees of freedom required.

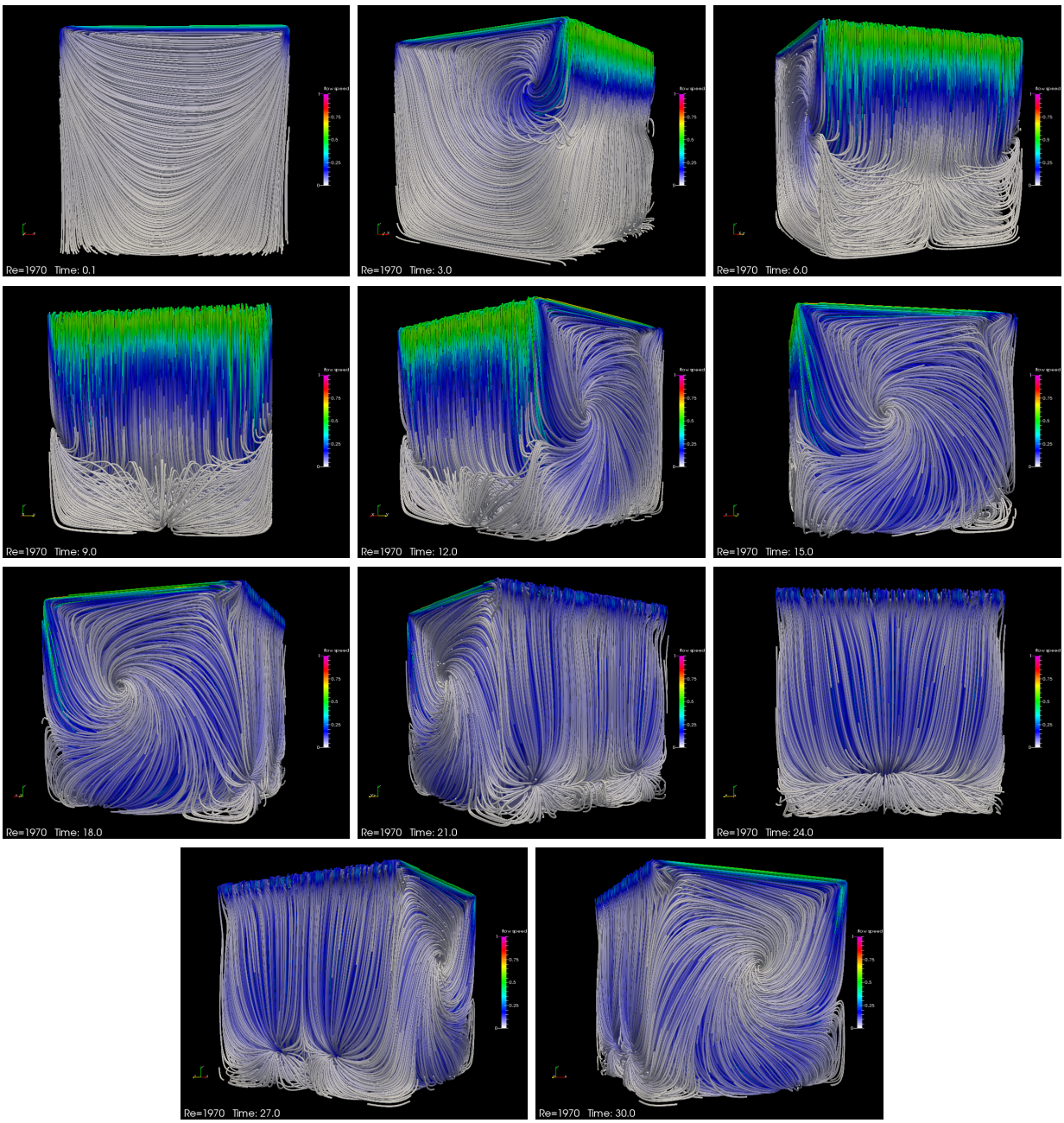

Figure 5.8: Snapshot of streamlines for 3D lid-driven cavity flow ($Re = 1\,970$) shown from $t = 0.1$ (initial flow development from the state of rest) to $t = 30$ (driven cavity flow is developed but it has not reached the periodic state) by increments of about 3 time units (the time t is increasing from the left to right, then top to bottom).

Figure 5.9 shows the velocity field projected on three midplanes; namely, xy-, xz- and yz-planes. While the solution may still be in a transient stage, the flow field is not far from the one shown in Figure 6 in [39] for $Re = 1\,970$ and in [30] for $Re = 2\,000$. The location of the primary and secondary vortices projected on the midplane and the detailed features of the upstream and downstream eddies at the corners (projected in the xz-midplane) present in our flow match qualitatively the results in these papers.

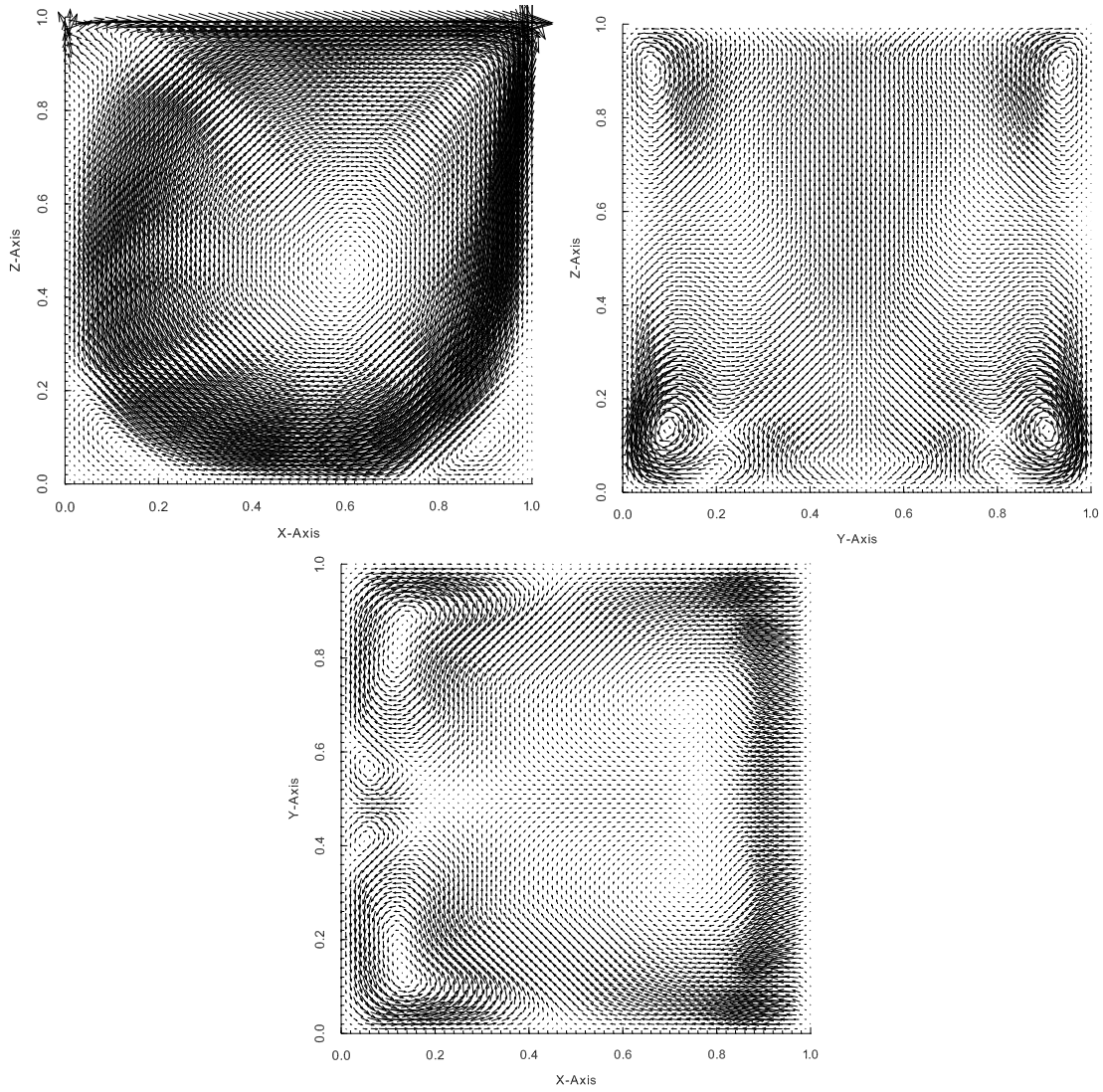

Figure 5.9: 3D lid-driven cavity at $Re = 1970$: Velocity vector during transient flow on the midplanes at $y = 0.5$ (top left), $x = 0.5$ (top right) and $z = 0.5$ (bottom).

Figure 5.10 shows the time evolution of all velocity components and pressure for

all test cases involving the 3D lid-driven cavity; all probed at point (0.2, 0.3, 0.25). For $Re = 100$, 400 and 1 000, the velocity components all reach a steady state. At $Re = 1\,970$, the flow is not yet periodic but the main portion of the transient phase is over. This can be seen from the progressive decay of the oscillations (be aware of the difference in scale from one graph to the next).

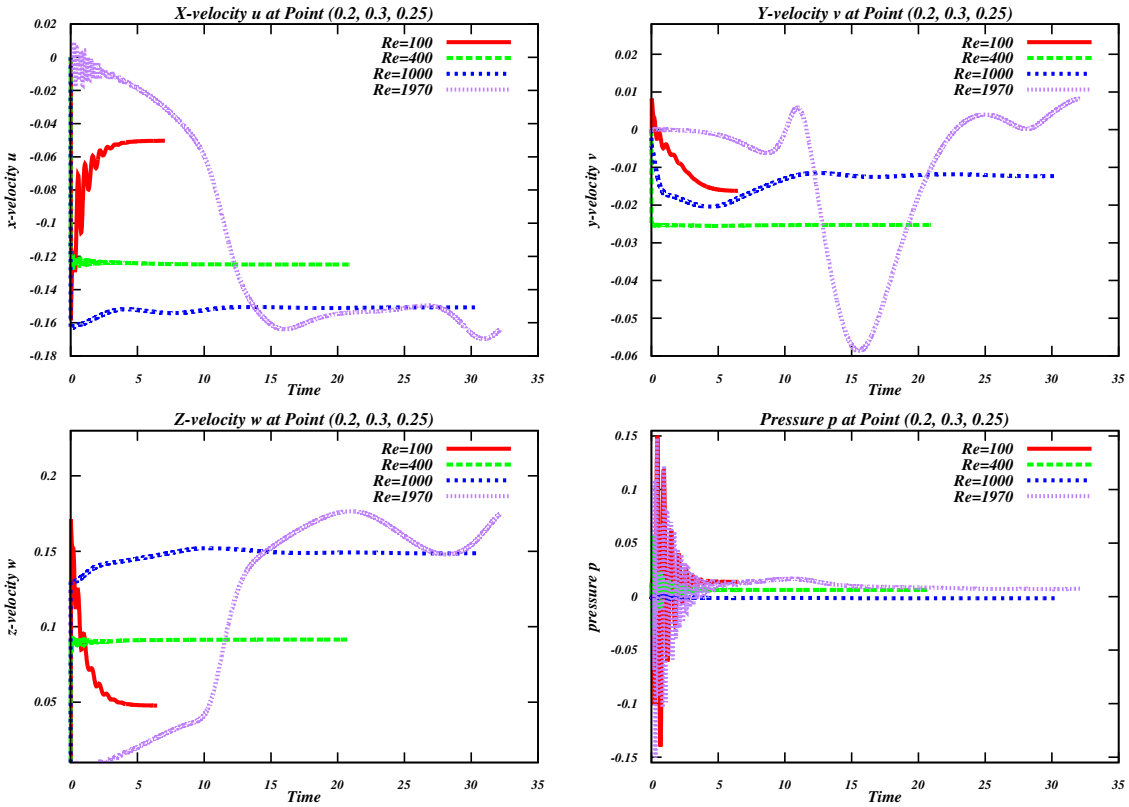

Figure 5.10: The time evolution of x-velocity u, y-velocity v, z-velocity w and pressure p at probing point (0.2, 0.3, 0.25) for $Re = 100$, 400, 1 000 (steady lid-driven flow) and $Re = 1\,970$ (unsteady lid-driven flow).

5.3 3D flow around the cylinder at $Re = 100$

To setup the geometry of this 3D test problem, we first generate a 2D rectangular-shaped domain minus a circular disk centred at the origin; $\Omega_1 = (-3, 9) \times (-3, 3) \setminus \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 < 0.25\}$. A non-uniform triangular mesh is generated with 40 elements along the edge of the rectangle in the x-direction, 20 along the y-direction and 60 along the boundary of the circular disk. This domain is then extruded in the z-direction with $z \in (-2, 2)$. This extrusion creates a 3D domain with a hollow cylinder

of diameter $D = 1$ with $\Omega = (-3, 9) \times (-3, 3) \times (-2, 2) \backslash \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 < (0.5)^2,\ z \in (-2, 2)\}$. To generate the required tetrahedral mesh, the edge along the z-axis is divided equally in 10 sub-intervals and the prismatic element is split into three tetrahedra. The 3D mesh generated has 118 500 tetrahedra with the smaller elements located near the cylinder and the element size increasing progressively away from the cylinder. This produces tetrahedra of varying diameter $0.040\,964 \leq h_K \leq 0.486\,931$. The discretization with $\mathbb{P}_2$-$\mathbb{P}_1$ finite elements produces a total of 509 040 unknowns for velocity and 22 715 for pressure. This is considered a small mesh for such 3D problems. The resulting mesh is shown on Figure 5.11.

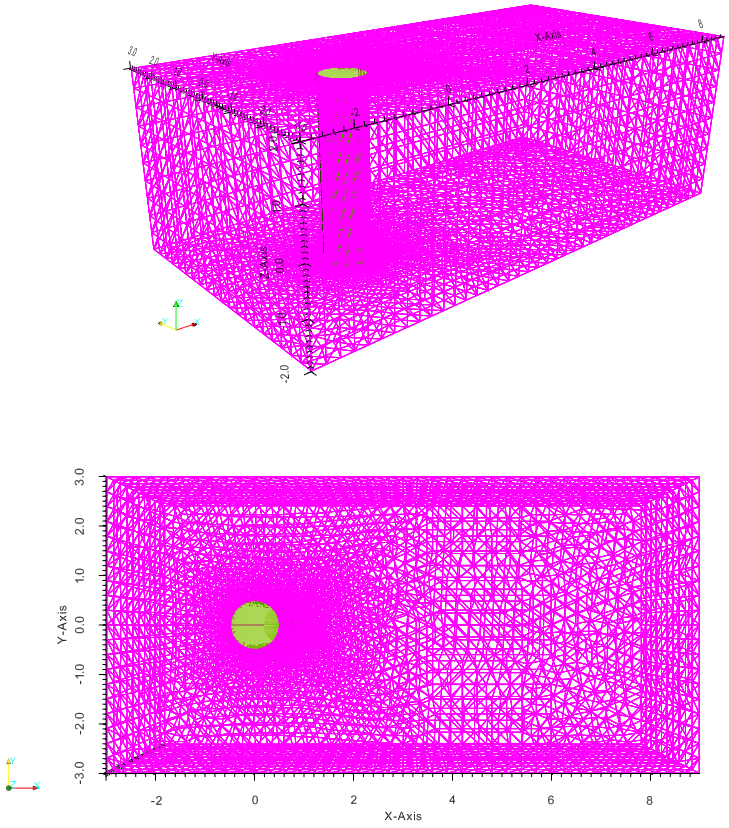

Figure 5.11: The mesh for the flow around the cylinder: 3D view from side angle (top) and 3D view from top (bottom).

Along the inflow boundary (plane at $x = -3$), $\mathbf{u} = (1, 0, 0)$ is prescribed. Along the outflow boundary (plane at $x = 9$), homogeneous Neumann boundary conditions are prescribed. No-slip velocity $\mathbf{u} = (0, 0, 0)$ is used on the surface of the cylinder. For other boundaries, vanishing normal velocity (zero flux) is imposed while allowing

natural flow tangential to the plane. GM-SRM-2 method is used with a direct solver, MUMPS, to handle the linear systems since the number of unknowns is relatively small. The stabilization parameters are set to $\alpha_1 = 200\tau$ and $\alpha_2 = 200$. With a constant time step $\tau = 0.008$, the computation of this 3D test problem requires about 7 500 time steps (total time $t = 60$ and CPU time is approximately 96 hours 50 minutes) to reach full periodic solution. The computation was initialized using the periodic solution at $Re = 100$ obtained on a coarser mesh and reinterpolated on the mesh presented above for the actual computation. The size of the computer memory needed to compute this 3D flow problem was about 16.3GB which is deemed large for about 500 000 unknowns. However, the computation is still efficient with a direct solver at the expense of computer memory. The computation of this test case was terminated at $t = 100$ (a total of 12 500 time steps) to ensure that the acquired periodic solution is reliable enough for data analysis (i.e., the lift and drag coefficients, Strouhal number, frequency, etc).

Figure 5.12 illustrates the time evolution of velocity (left), and streamlines and vorticity (right) for $t_j = \frac{j\varphi}{4}$, where $j = 0, 1, 2, \ldots, 8$. One period cycle φ is equivalent to 5.080 8 time units in this test case. Within this time window, the vortex shedding behind the 3D cylinder is fully periodic for all of the variables; e.g., velocity, pressure, lift and drag coefficients. At any given time, the velocity fields, vorticity and streamlines are nearly identical on arbitrary xy-planes taken at $-2 \leq z \leq 2$. This confirms the 2D nature of the flow, as expected for the flow around the cylinder at $Re = 100$.

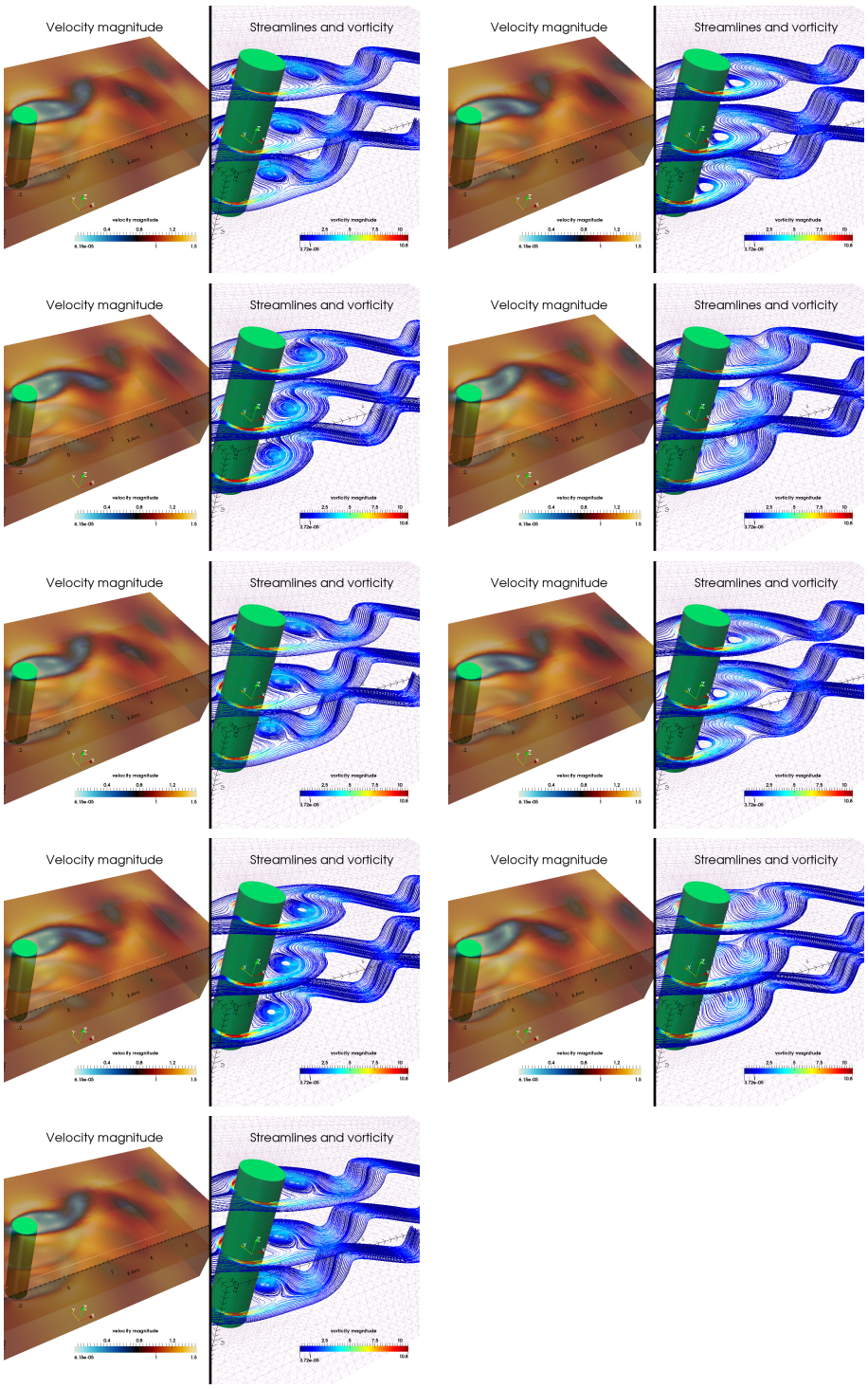

Figure 5.12: For each plot, velocity magnitude (color-coded) is shown in the left column, while both streamlines and vorticity (color-coded) are displayed in the right column. The plots are shown at every quarter period for $t_j = \frac{j\varphi}{4}$, $j = 0, 1, 2, \ldots, 7, 8$ where $\varphi = 5.080\,8$. The sequence goes from left to right, and from top to bottom.

The lift and drag along the cylinder denoted as c_l and c_d, respectively, are evaluated using the volume integration method which has been used in the previous chapter for 2D flow around the cylinder at $Re = 100$ (see [41, 54]). The only difference is that the resulting drag and lift will have to be divided by a factor of 4, the length in the z-direction of the 3D cylinder used in this test case.

Figure 5.13 shows the evolution of the lift and drag coefficients for $t \in [50, 100]$ while the computed mean, amplitude and frequency for these coefficients are summarized in Table 5.3. At first glance, we observe that the drag coefficient oscillates with a frequency twice as large as the lift coefficient. In fact, we notice a slight variation of the peak-to-peak value of the drag coefficient with the same frequency as the pulsation of the lift coefficient. This is a numerical artifact which can be reduced by taking larger stabilization parameters in GM-SRM-2; i.e., $\alpha_1 > 200\tau$ and $\alpha_2 > 200$. The choice of larger stabilization parameters, α_1 and α_2, is undesirable since over-stabilization may produce numerical inaccuracy of the overall results (in space). To compute the amplitude of the drag coefficient, we took the average of the two different peaks of the drag coefficient once a fully periodic flow is achieved (see Figure 5.13: right).

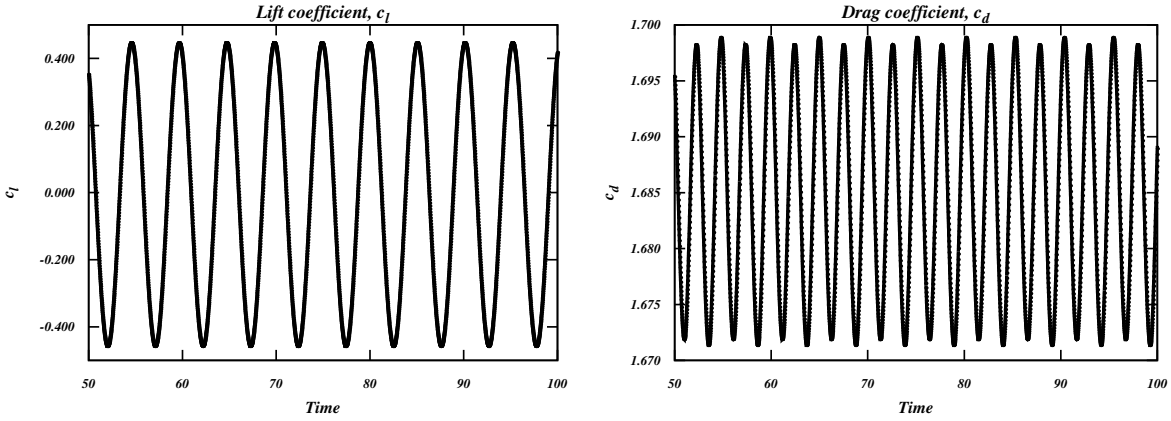

Figure 5.13: 3D flow around the cylinder at $Re = 100$: Time evolution of the lift c_l and drag c_d for $t \in [50, 100]$.

Figure 5.14 shows the time history of the three velocity components u, v and w at several probing points for $t \in [50, 100]$. The time history of the x-velocity u, y-velocity v and pressure p are monitored at $(5, 0, -2)$, $(5, 0, 0)$ and $(5, 0, 2)$. These monitoring points are chosen such that they align together and stand on a line parallel to the cylinder. The time history of the z-velocity is monitored at $(5, -2, 0)$, $(5, 0, 0)$ and $(5, 2, 0)$. These monitoring points are chosen such that they align together and form a line orthogonal to the cylinder. All x-velocity u, y-velocity v and pressure p produce similar results at all these probing points. Further, the amplitude of the variation on

the z-velocity is less than 10^{-3} at these points, i.e., at least two orders of magnitude smaller than that for the other two components. It is known that the flow around the 3D cylinder at $Re = 100$ is still two-dimensional from the literature. Our results follow these trends. For Reynolds number $Re > 180$ [49] or $Re > 200$ [59], it has been documented that 2D model may fail to represent the 3D flow realistically. Moreover, large difference is observed between 2D results and experimental data.

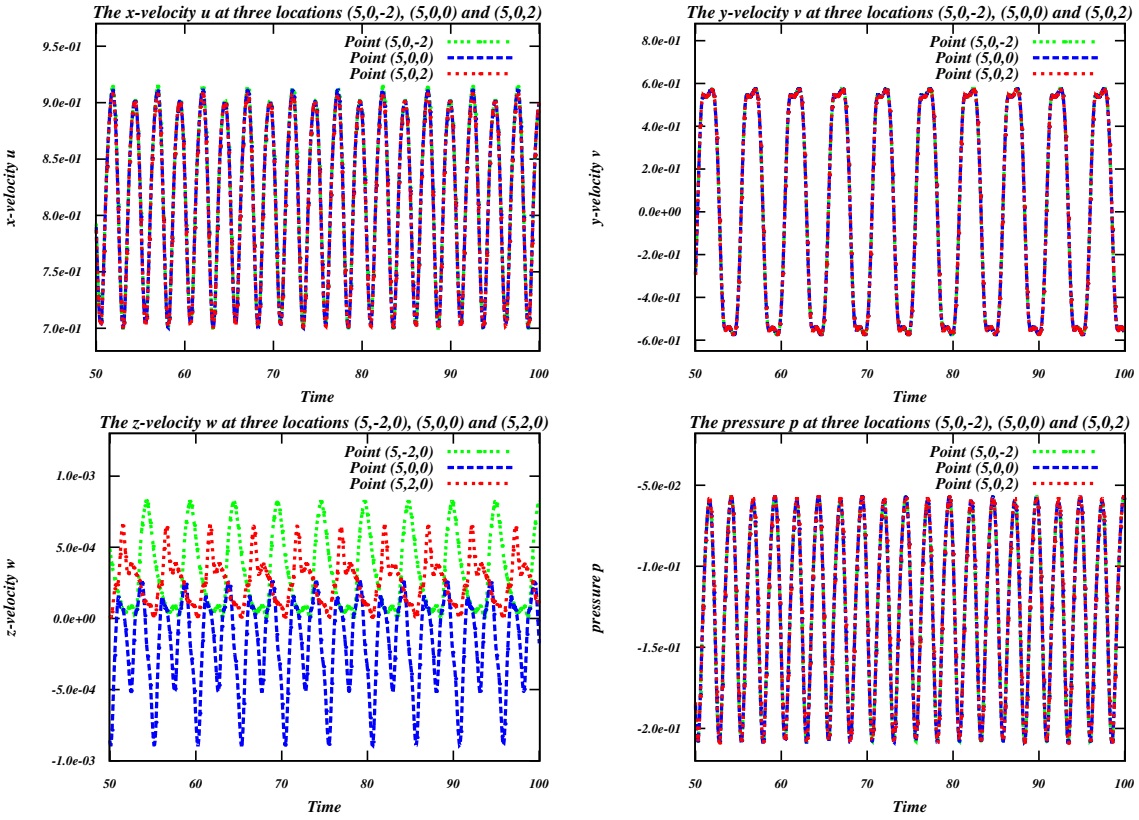

Figure 5.14: 3D flow around the cylinder at $Re = 100$: Time evolution of the x-, y-, z-velocity and pressure monitored at various locations for $t \in [50, 100]$.

The drag coefficient from our results is bigger than the values found in the literature (see Table 5.3). For instance, the computed mean drag is about 20% and 26% greater than the mean drag reported in [49] and [42], respectively. The computed mean lift is very close to zero, which suggests that our computation reproduces quite accurately the space-time symmetric (across the xz-plane or $y = 0$) vortex shedding with a single frequency of oscillation. Our frequency is about 17% bigger than the commonly agreed value of 0.168 for the flow around the cylinder at $Re = 100$ (see [21, 49]). Our lift amplitude is 28% bigger than the reported value in [42].

The coarse mesh and smaller domain used in our computation may result in these over-predicted values. In conclusion, the mesh used for the spatial discretization in our computation should be made finer to produce more accurate results on lift and drag coefficients and the Strouhal numbers. This can be done with more CPU and memory resources but we did not attempt this here.

Table 5.3: 3D flow around the cylinder at $Re = 100$: The comparison of the computed average, amplitude and frequency of the drag and lift coefficients with values from the literature.

Drag c_d	**Mean**	**Amplitude**	**Frequency**
Current results	1.6851	1.3177×10^{-2}	0.3936
Ranjani et al. [49]	1.3349	–	–
Mittal & Raghuvanshi [42]	1.4020	–	–
Lift c_l	**Mean**	**Amplitude**	**Frequency**
Current results	-5.6667×10^{-3}	0.4527	0.1968
Mittal & Raghuvanshi [42]	–	0.3550	0.1680

5.4 3D flow around the sphere at $Re = 300$

There are many ways to setup the computational domain for this test problem. Due to several technical reasons, a cylindrical domain is preferable than a rectangular domain as the external boundary for this test case. First of all, the number of unknowns can be optimized since the volume of a cylinder is smaller than that of an elongated cuboid when the width of the elongated cuboid and the diameter of a cylinder are close to each other. Second, the flow between the sphere and the outer cylindrical boundary is not subject to an artificial break-up of (axi)symmetry since the nearest distance from the surface of the sphere to the side wall of the cylinder is identical in all directions. On the other hand, a domain with a square cross-section would induce preferred directions in the shedding vortices behind the sphere. Apparently, this does not reflect the right physics of the flow. Thirdly, we noticed that the 3D flow around the sphere with a cylindrical domain has a shorter transient time to reach a fully periodic flow (not shown).

To setup the 3D geometry, we first define a 2D domain Ω_I composed of the rectangle $[0, 12] \times [0, 3]$ in the xz-plane minus the semi-circular hole centered at $(x, z) = (3, 0)$ with a radius of 0.5, as shown at the top of Figure 5.15. We fix Ω_I on the xz-plane to facilitate the boundary labelling for the top and bottom part of the 3D cylindrical domain. Small elements are prescribed along the lower boundary and around the semicircle to account for the larger variation on the solutions in these regions. To do this, we equally divide both edges on the left and right into 6 elements, top edge into

30 elements, bottom edge into 120 elements and the boundary of the semicircle into 30 elements. Next, using *msh3* the 3D cylindrical domain is generated by rotating the plane along the z-axis. The resulting mesh is shown at the bottom of Figure 5.15. The cylindrical domain generated is defined as $\Omega = \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 < 9,\ z \in (0, 12)\} \setminus \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + (z-3)^2 < 0.25\}$.

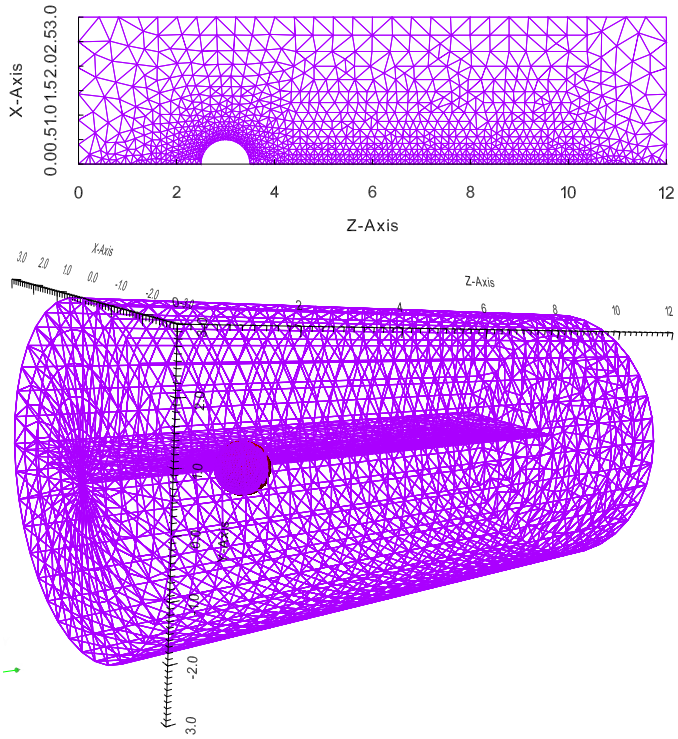

Figure 5.15: The 2D mesh for Ω_I (top) and the 3D mesh for Ω (a sphere inside the cylinder). The 2D mesh is shown lying inside the 3D cylinder (bottom).

With the above setting, a tetrahedralization is generated in FreeFEM++, which gives a total of 171 356 tetrahedra, 24 474 nodes and a varying element size $0.038\,269 \leq h_K \leq 1.000\,650$. Figure 5.16 shows the trace of the mesh on the yz-plane, xz-plane and a cross-section in the xy-direction at $z = 3$. It can be seen that the resulting mesh is fine near the sphere and in the wake behind the sphere to properly catch the boundary layer and shedding vortices. The size of the mesh increases near the inflow boundary $\Gamma_{z=0}$, outflow boundary $\Gamma_{z=12}$ and the cylindrical boundary $\Gamma_c = \{x, y, z \in \mathbb{R} \mid x^2 + y^2 = 9,\ z \in (0, 12)\}$, since the fluid velocity tends to constant values in these areas. We tried to mesh this geometry with *TetGen* but *msh3* turned out to be easier to use and sufficient to obtain a relatively nice mesh for this test case.

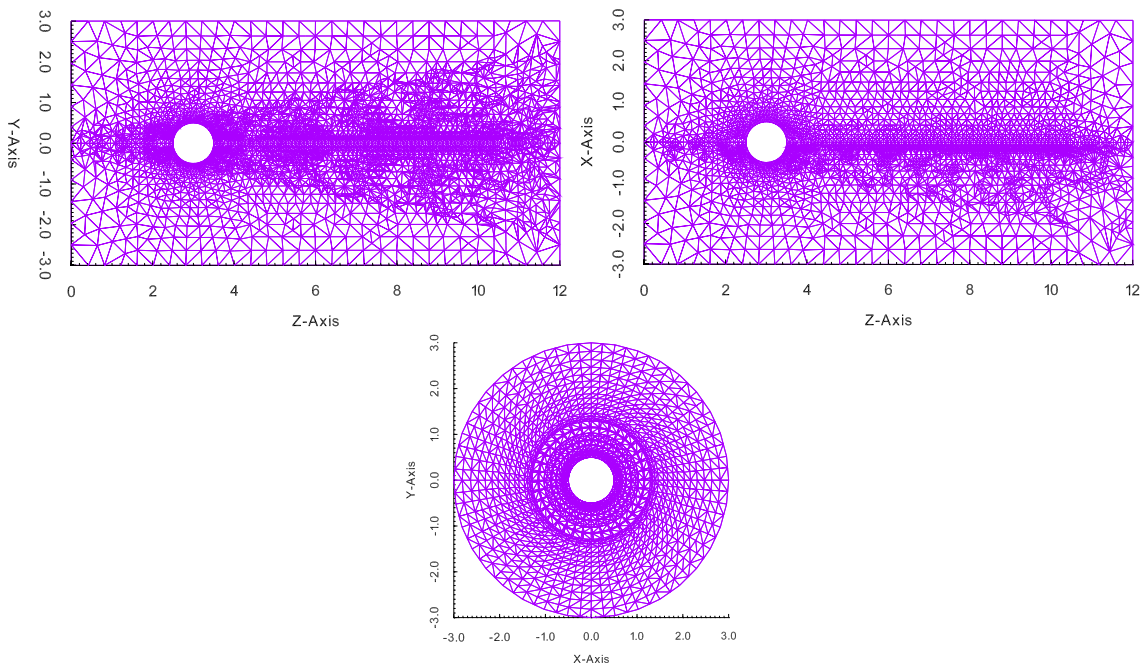

Figure 5.16: The mesh around the sphere: Views of the mesh in the yz-plane (top left), xz-plane (top right) and xy-plane at $z = 3$ (bottom).

Similar to the 3D flow around the cylinder, a fixed velocity $\mathbf{u} = (0, 0, 1)$ is prescribed along the inflow boundary $\Gamma_{z=0}$. Homogeneous Neumann boundary conditions are prescribed along the outflow boundary $\Gamma_{z=12}$. No-slip velocity $\mathbf{u} = (0, 0, 0)$ is set along the boundary of the sphere. Lastly, along the cylinder boundary Γ_c, the velocity normal to the surface is set to zero, which allows for free tangential flow along the side wall of the cylinder. We considered a Reynolds number of 300 which is slightly above the critical value for a Hopf bifurcation $Re_{\text{crit}} \in [270, 280]$ [33]. This value is higher than the one required for the 3D flow around the cylinder which is about $Re_{\text{crit}} = 70$.

The discretization with $\mathbb{P}_2$-$\mathbb{P}_1$ elements produces a total of 573 915 unknowns for the velocity and 24 474 for the pressure. The resulting number of unknowns is considered moderate for a 3D problem, hence the direct linear solver MUMPS can be still used and is CPU-efficient. We solve this problem with the GM-SRM-2 method by fixing the stabilization parameters $\alpha_1 = 50\tau$ and $\alpha_2 = 50$. This choice of the stabilization parameters produces minor oscillations (in time) of the velocity and pressure at start-up, but these oscillations are quickly damped in a few time steps. The choice of our stabilization parameters is close to optimal since by picking $\alpha_1 < 50\tau$ and $\alpha_2 < 50$, the minor oscillations generated at startup are not as quickly damped as for the optimal values. We also checked that computations with $\alpha_1 = 100\tau$ and $\alpha_2 = 100$ produce smaller Strouhal number for drag, lateral and side lift coefficients, i.e., $Str = 0.1329$,

which could be a sign of over-stabilization in GM-SRM-2 with these larger values of α_1 and α_2.

Here, we suggest an adjustment on stabilization parameters α_1 and α_2 by comparing the closeness of our Strouhal number to the one obtained in the literature. The Strouhal number is chosen because it is the least sensitive parameter and relatively easy to reproduce. This approach may not be supported by any theoretical analysis but we found that this method although non-rigorous to be very useful to guess optimal α_1 and α_2, at least for this test case.

In this test case, we do not apply the computation of lift, drag and side coefficients using volume integration technique which was proposed in [32, 54] and implemented in our earlier test cases. We remind the reader that the volume integration technique has been used in Chapter 2 to compute c_d and c_l involving 2D flows around the cylinder. The computation of c_d and c_l does not involve the volume integration technique in most of the literature. To ensure a similar comparison with the published results, we resort to the direct boundary integral technique from [33]:

$$\begin{aligned} c_d(t) &= \frac{8}{\pi U_\infty}\int_S \left[\frac{\nu}{2}\left(\frac{\partial w(t)}{\partial x}+\frac{\partial u(t)}{\partial z}\right)n_x+\frac{\nu}{2}\left(\frac{\partial w(t)}{\partial y}+\frac{\partial v(t)}{\partial z}\right)n_y \right. \\ &\quad \left. +\nu\frac{\partial w(t)}{\partial z}n_z - p(t)n_z\right] dS, \end{aligned} \tag{5.4.1}$$

$$\begin{aligned} c_l(t) &= -\frac{8}{\pi U_\infty}\int_S \left[\frac{\nu}{2}\left(\frac{\partial v(t)}{\partial x}+\frac{\partial u(t)}{\partial y}\right)n_x+\nu\frac{\partial v(t)}{\partial y}n_y \right. \\ &\quad \left. +\frac{\nu}{2}\left(\frac{\partial v(t)}{\partial z}+\frac{\partial w(t)}{\partial y}\right)n_z - p(t)n_y\right] dS, \end{aligned} \tag{5.4.2}$$

$$\begin{aligned} c_s(t) &= -\frac{8}{\pi U_\infty}\int_S \left[\nu\frac{\partial u(t)}{\partial x}n_x+\frac{\nu}{2}\left(\frac{\partial u(t)}{\partial y}+\frac{\partial v(t)}{\partial x}\right)n_y \right. \\ &\quad \left. +\frac{\nu}{2}\left(\frac{\partial u(t)}{\partial z}+\frac{\partial w(t)}{\partial x}\right)n_z - p(t)n_x\right] dS, \end{aligned} \tag{5.4.3}$$

where $\mathbf{n} = (n_x, n_y, n_z)$ is the vector normal to the boundary S of the sphere. We set the far-field flow speed to $U_\infty = 1$, the diameter to $D = 1$ and the viscosity to $\nu = \frac{1}{300}$, to get $Re = 300$ for this test case. Further, the external force is set to $\mathbf{f} = 0$.

The computations were stable with the time step $\tau = 0.005$ and were run until a fully periodic flow was observed. A periodic solution is reached for about $t = 120$ (i.e., 24 000 time steps), starting from a periodic solution at $Re = 1\,000$ computed on a coarser mesh to initiate the von Kármán alley behind the sphere. The required total CPU time was about 21 days on a i7-3 770 3.40GHz personal desktop, using 23GB of memory. The computation was terminated once it reaches $t = 140$.

Figure 5.17 shows the time evolution of the streamlines of the flow which are plotted for 2 periods of the flow.

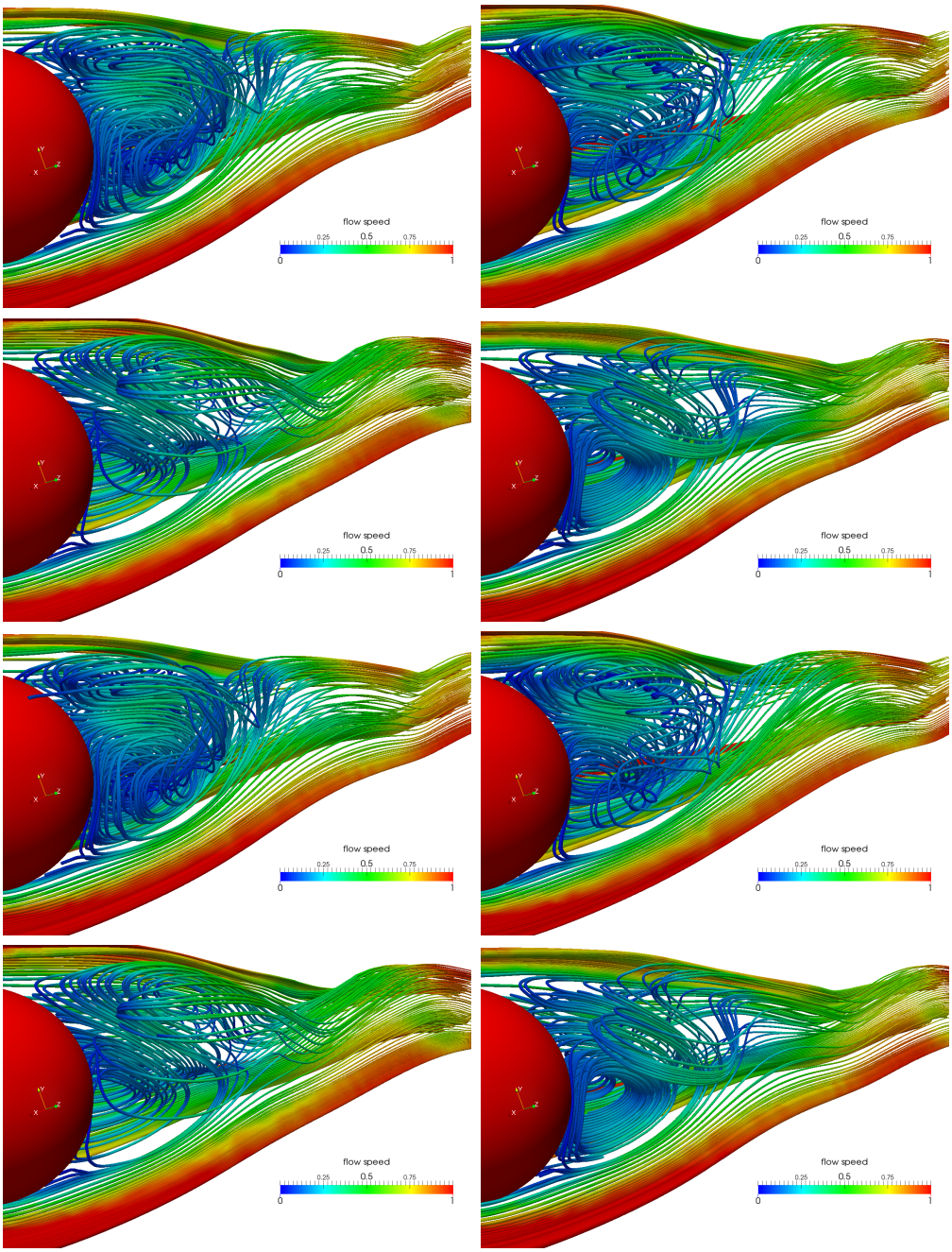

Figure 5.17: Snapshots of the streamlines for the 3D flow around the sphere ($Re = 300$), colored by the flow speed, shown at $t = \frac{j\varphi}{4}$, $j = 1, 2, \ldots, 6, 7$ (from left to right, then top to bottom), $\varphi = 7.4074$.

The flow exhibits many 3D characteristics which cannot be represented by an axisymmetric 2D flow. The streamlines are color-coded by the magnitude of the velocity. Higher fluid velocity is observed along the streamwise direction. The velocity increases moving away from the sphere in the crosswise direction and the smallest flow speed occurs behind the sphere which results in the generation of two separating vortices, as can be seen as two major closed loop and interconnected streamlines (recirculations) in this area. In the wake behind the sphere, the periodic occurence of swirling flow is observed, for instance by comparing the streamlines at $t = \frac{\varphi}{4}$ and $\frac{5\varphi}{4}$ which are identical to each other. The period φ of the flow is approximately 7.4074 time units.

We adopt the technique that we have discussed earlier (see Section 5.2.1) for the projection of streamlines on 2D planes. Figure 5.18 illustrates the planes used to plot the projected streamlines.

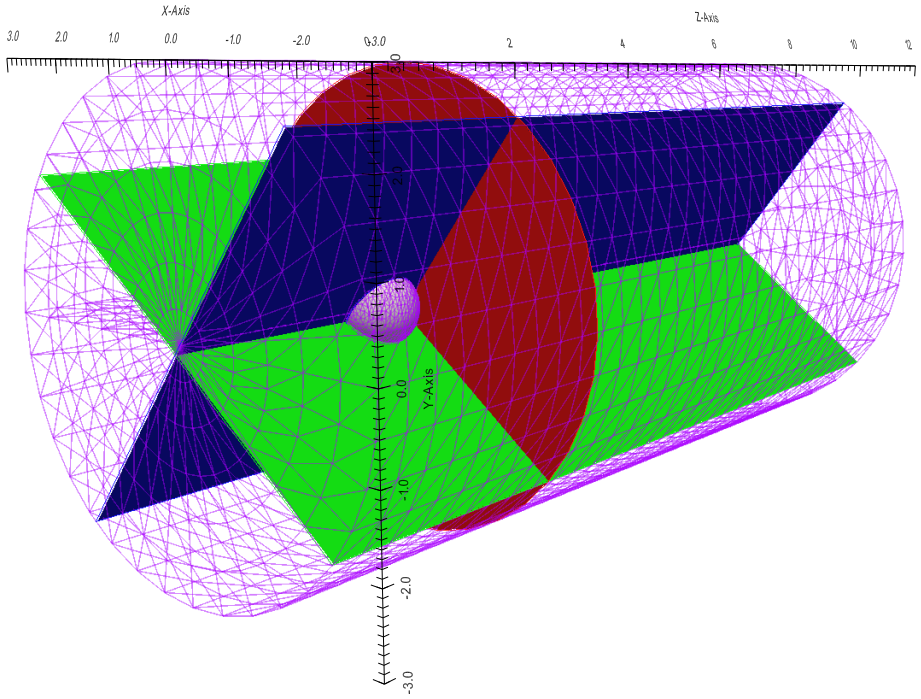

Figure 5.18: The three planes used for the projection of streamlines/streamtubes: xy-plane behind the cylinder at $z = 3.5$ (in red); both yz-plane (in blue) and xz-plane (in green) are rotated counter-clockwise of 35° around the z-axis.

The bifurcation occuring at $Re_{\text{crit}} \in [270, 280]$ results from the transformation of the axisymmetric flow for $Re < Re_{\text{crit}}$ into a non-axisymmetric flow for $Re > Re_{\text{crit}}$ but with reflexion symmetries (in space and space-time) with two symmetry planes

parallel to the z-axis. The cylindrical domain used in our experiment prevents any locking of the symmetry planes at a specified angle. By plotting the so-called rear surface-limiting streamlines on the xy-plane (see Figure 5.19), our result shows the existence of a symmetry plane with a pair of symmetric vortices on both sides of the plane. The symmetry plane is tilted with a counter-clockwise angle of about $35^\circ \pm 2^\circ$. Minor oscillations of the angle about the z-axis occurs over time with a maximal amplitude of $\pm 2^\circ$. As a result, the stagnation point (for streamlines projected on the xy-plane at $z = 3.5$) at the back of the sphere shifts slightly over time in a periodic fashion. The period of these oscillations follows the period of the lift and drag coefficients, i.e., $\varphi = 7.4074$ time units. The rear stagnation point is captured at (-0.13, 0.18, 3.5) at $t = 0$, (-0.12, 0.16, 3.5) at $t = \frac{\varphi}{4}$, (-0.14, 0.17, 3.5) at $t = \frac{\varphi}{2}$ and (-0.16, 0.22, 3.5) at $t = \frac{3\varphi}{4}$ and (-0.13, 0.18, 3.5) at $t = \varphi$ (not shown), and similar pattern repeats in a periodic fashion.

Figure 5.20 and 5.21 show the streamlines projected on the rotated yz- and xz-plane (counter-clockwise by 35°), respectively, over a single period of the flow. In addition to this rotation of 35° (see Figure 5.18), these planes are slightly adjusted within $\pm 2^\circ$ to match as closely as possible the symmetry axis in the projection plane at different times within each period. The plots in Figure 5.20 show the existence of a reflexion symmetry with axis $x = 0$ for the projected streamlines in the tilted xz-plane. Our results for the projected streamlines on the tilted yz- and xz-plane match very closely with the ones shown in Figure 25 of [33].

Table 5.4 summarizes the drag, lateral lift and side lift coefficients computed with GM-SRM-2 and the values found in the literature. The computed mean drag is underestimated within a range $16\% - 21\%$ compared to values in the literature (see [33, 50, 53, 67] and the experiment [50]), while the computed drag amplitude is about 30% smaller than the value in [67]. Our computations produce very close values for the frequencies of the drag, lateral lift, and side lift coefficients. For the lateral and side lift coefficients, we compare well to the few published values, at least within the bounds set by the numerical accuracy of our spatial discretization. The mean and amplitude for the side coefficient computed with our method are larger than zero, while these should be zero from theory. The side lift being null corresponds to the ability to reach a planar-symmetric flow in a tilted midplane, in our case with an angle of 35° counter-clockwise off the yz-plane. We have a time-variation of $\pm 2^\circ$ on the tilting angle, resulting in a side lift coefficient potentially off from zero. Nonetheless, our GM-SRM-2 method is capable to reproduce the fully periodic unsteady flows with a frequency close to values found in the literature, which shows that the time approximation with GM-SRM method is reasonably accurate to solve this 3D test case. The accuracy of the side and lateral lift coefficient can be further improved by using a finer mesh and a larger domain. Again however, finer meshes require a more efficient solver to handle larger linear systems.

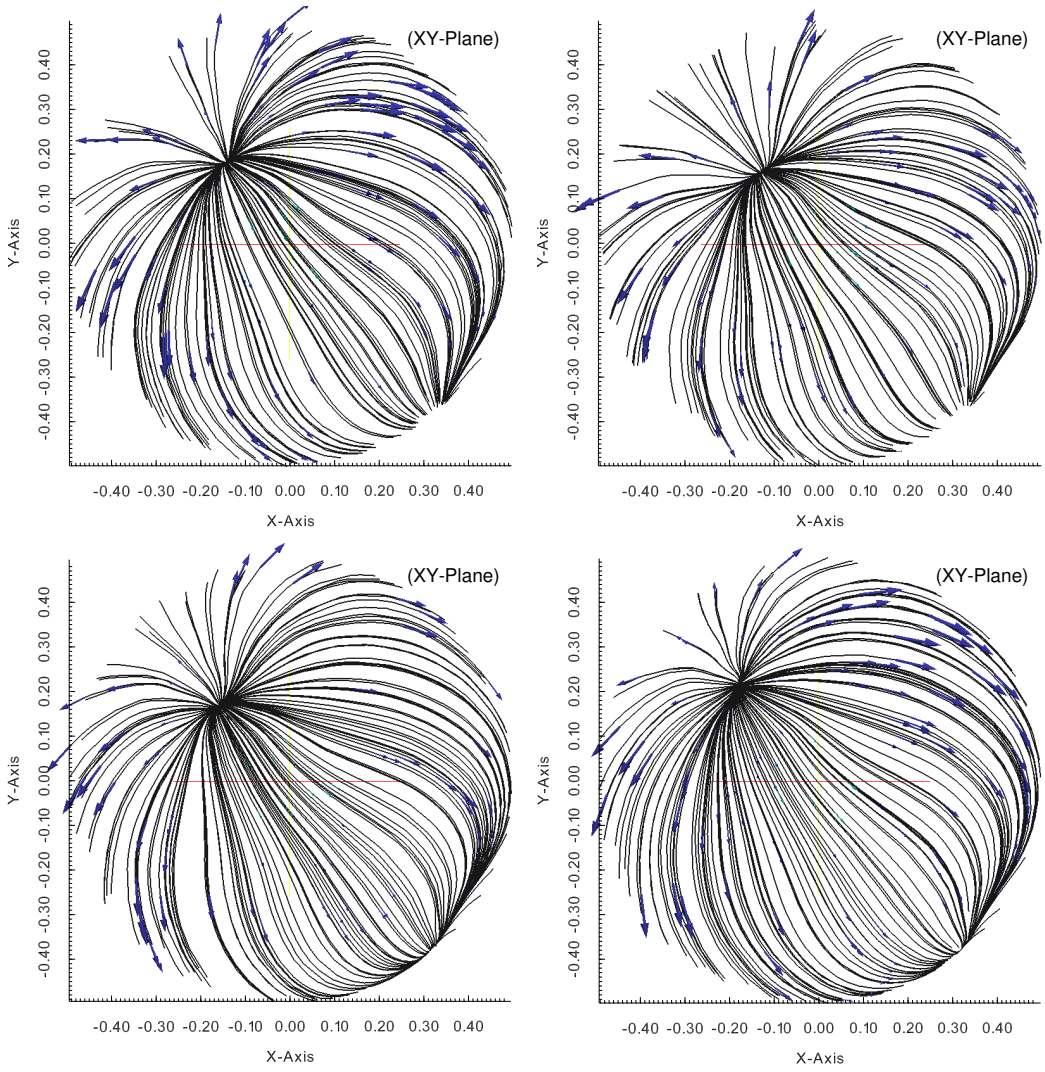

Figure 5.19: Snapshots of rear-surface limiting streamlines for the 3D periodic flow around the sphere ($Re = 300$) projected on the xy-plane at $z = 3.5$ for $t = 0$ (top left), $t = \frac{\varphi}{4}$ (top right), $t = \frac{\varphi}{2}$ (bottom left) and $t = \frac{3\varphi}{4}$ (bottom right) time units, where $\varphi = 7.4074$.

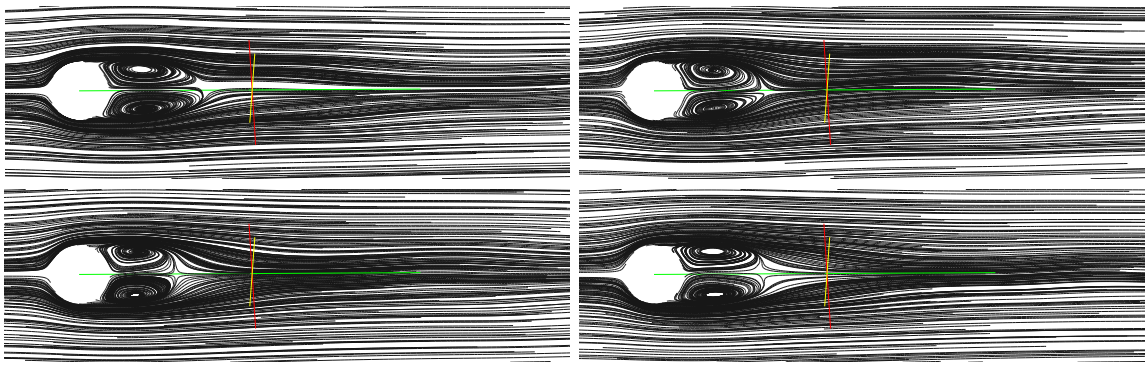

Figure 5.20: Snapshots of the streamlines for the 3D periodic flow around the sphere ($Re = 300$) projected on the xz-plane (tilted counter-clockwise by $35° \pm 2°$) for $t = 0$ (top left), $t = \frac{\varphi}{4}$ (top right), $t = \frac{\varphi}{2}$ (bottom left) and $t = \frac{3\varphi}{4}$ (bottom right) where $\varphi = 7.4074$.

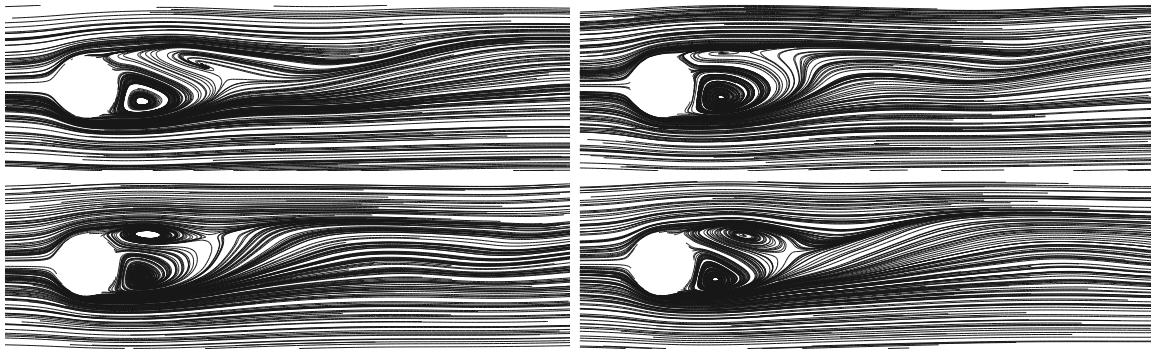

Figure 5.21: Snapshots of the streamlines for the 3D periodic flow around the sphere ($Re = 300$) projected on the yz-plane (tilted counter-clockwise by $35° \pm 2°$) for $t = 0$ (top left), $t = \frac{\varphi}{4}$ (top right), $t = \frac{\varphi}{2}$ (bottom left) and $t = \frac{3\varphi}{4}$ (bottom right) where $\varphi = 7.4074$.

Table 5.4: The average, amplitude and frequency of the drag, lateral and side lift coefficients for 3D flow around the sphere at $Re = 300$.

Drag c_d	**Mean**	**Amplitude**	**Frequency**
Current	0.5302	1.9651×10^{-3}	0.1350
Johnson & Patel [33]	0.6560	3.5000×10^{-3}	0.1370
Tomboulides [67]	0.6710	2.8000×10^{-3}	0.1360
Roos & Willmarth [50]	0.6290	–	–
Sakamoto & Haniu [53]	–	–	$0.1500 - 0.1650$
Lateral lift c_l	**Mean**	**Amplitude**	**Frequency**
Current	-1.1170×10^{-1}	1.8018×10^{-3}	0.1350
Johnson & Patel [33]	-6.9000×10^{-2}	1.6000×10^{-2}	0.1370
Tomboulides [67]	–	–	0.1360
Sakamoto & Haniu [53]	–	–	$0.1500 - 0.1650$
Side lift c_s	**Mean**	**Amplitude**	**Frequency**
Current	6.4800×10^{-2}	1.1721×10^{-2}	0.1350
Johnson & Patel [33]	0	0	–

5.5 A Comparison on the Computational Efficiency

In this section, we compare the CPU and memory requirements of all 2D and 3D test cases attemped in this thesis. We did not do efficiency comparisons as were presented in [41] since we do not have sufficient information about the convergence (in time) for our 3D test cases. Our goal in this chapter was not so much in comparing our results with reference solutions than to illustrate the potential of the methods. Since we did not carry the computation to search truly periodic flows but just far enough to obtain the general features of the flows, CPU times might not be comparable to some of the published result.

Table 5.5 summarizes all test cases done in the thesis with the respective Reynolds numbers, time-stepping methods, the presence of **grad**-div stabilization term (GD), number of unknowns, computer memory (in Mb) and CPU time (in second) involved to compute one time step. Here, the computer memory is the maximum memory required to solve the linear systems and to store all variables. Also presented in this table is the ratio of the CPU time per degree of freedom per time step, here noted as $ratio^{\oslash}$. All computation were carried out on the same personal desktop with an Intel Core i7-3 770 CPU 3.4GHz×8 cores processor.

All the test cases employ MUMPS as the linear solver except two test cases in the 3D lid-driven cavity where a GM method with **grad**-div stabilization and the iterative linear solver HIPS were used. The parameter $ratio^{\oslash}$ provides a reasonable clue on required CPU resources with different linear solvers, e.g., MUMPS and HIPS, both

for the 2D and 3D computations independently from the order of the method. For instance, we deduce that higher-order SBDF methods are the most CPU-efficient with MUMPS since their *ratio*$^{\oslash}$ is the smallest. Furthermore, the value of *ratio*$^{\oslash}$ between the 2$^{\text{nd}}$- and 3$^{\text{rd}}$-order SBDF methods are close to each other.

Table 5.5: Statistics for all test cases carried in this thesis.

Test case	Re	Method	GD	#dof $(\mathbf{u}, p)$	Mem.	CPU_τ	[1]ratio$^{\oslash}$
2D cavity	8 500	SBDF-2	No	(137 618, 17 323)	392	0.98	6.32×10^{-6}
2D cavity	8 500	SBDF-3	No	(137 618, 17 323)	392	1.31	8.45×10^{-6}
2D cavity	8 500	DC-2	No	(137 618, 17 323)	392	1.99	1.28×10^{-5}
2D cavity	8 500	DC-3	No	(137 618, 17 323)	392	3.62	2.34×10^{-5}
2D cavity	8 500	GM-2	Yes	(137 618, 17 323)	273	2.46	1.58×10^{-5}
2D cavity	8 500	GM-3	Yes	(137 618, 17 323)	273	4.19	2.70×10^{-5}
2D cavity	8 500	GM-SRM-2	Yes	(137 618, 17 323)	272	2.88	1.86×10^{-5}
2D cavity	8 500	GM-SRM-3	Yes	(137 618, 17 323)	271	4.85	3.13×10^{-5}
2D cavity	50 000	SBDFB-3	Yes	(381 242, 47 856)	1 308	2.73	6.36×10^{-6}
2D cavity	100 000	SBDFB-3	Yes	(592 602, 74 326)	2 107	4.75	7.12×10^{-6}
2D cylinder	100	SBDF-2	No	(108 474, 13 648)	307	1.00	8.19×10^{-6}
2D cylinder	100	SBDF-3	No	(108 474, 13 648)	289	1.05	8.60×10^{-6}
2D cylinder	100	DC-2	No	(108 474, 13 648)	302	2.24	1.83×10^{-5}
2D cylinder	100	DC-3	No	(108 474, 13 648)	289	3.06	2.51×10^{-5}
2D cylinder	100	GM-2	Yes	(108 474, 13 648)	214	2.71	2.22×10^{-5}
2D cylinder	100	GM-3	Yes	(108 474, 13 648)	223	3.32	2.72×10^{-5}
2D cylinder	100	GM-SRM-2	Yes	(108 474, 13 648)	223	2.61	2.14×10^{-5}
2D cylinder	100	GM-SRM-3	Yes	(108 474, 13 648)	222	3.23	2.64×10^{-5}
2D cylinder	1000	SBDFB-3	Yes	(721 760, 90 500)	2 574	5.91	7.28×10^{-6}
3D cavity (HIPS)	100, 400, 1 000	[2] GMS2-GS_2	Yes	(3 090 903, 132 651)	15 680	228.61	7.09×10^{-5}
3D cavity (HIPS)	1970	[3]GMS2-NLGD	Yes	(3 090 903, 132 651)	15 680	381.80	1.18×10^{-4}
3D cylinder	100	GM-SRM-2	Yes	(509 040, 22 715)	13 766	33.89	6.37×10^{-5}
3D sphere	300	GM-SRM-2	Yes	(573 915, 24 474)	23 058	51.75	8.65×10^{-5}

We observe that parameter *ratio*$^{\oslash}$ for all DC, GM and GM-SRM methods can be approximately scaled by the number of subproblems involved in the methods by comparing it with the *ratio*$^{\oslash}$ of the SBDF methods. For instance, we found a factor of k larger in the *ratio*$^{\oslash}$ of the k^{nd}-order DC, GM and GM-SRM than the *ratio*$^{\oslash}$ of SBDF-k method since the former methods have k subproblems per time step to deal with. Between the methods with the defect correction strategy in general, DC methods have the smallest value of *ratio*$^{\oslash}$ which are followed closely by GM and GM-SRM methods. Even though the resulting linear systems in GM and GM-SRM methods (with symmetric and positive definite global matrix) are easier to solve than the saddle point global matrix in DC methods, the *ratio*$^{\oslash}$ is still controlled by the total number of the linear system to be solved.

The 3D test problems produce a *ratio*$^{\oslash}$ about 3 to 4 times larger than for 2D problems using GM-SRM methods. This occurs due to the different sparsity pattern in the global matrix between 2D and 3D problems. 3D problems have a larger bandwidth than 2D test problems, which requires more fill-in in the matrix during the factorization step in MUMPS. The combination of GM methods with **grad**-div splitting and

[1] ratio$^{\oslash}$ = CPU_τ/dof

[2] GMS2-GS_2 is GM-2 method with **grad**-div splitting with only two Gauss–Seidel iterations

[3] GMS2-NLGD is GM-2 method with **grad**-div splitting with "nonlinear ansatz"

HIPS is very promising to compute 3D flows with a large number of unknowns which is otherwise not feasible using MUMPS.

Most of our 2D problems require reasonably small memory and can be easily handled by modern personal desktop. This memory rarely exceeds 4Gb. However, for 3D flow around cylinder and sphere, the memory required with MUMPS reaches about 14Gb and 23Gb, respectively. We should mention that for these flows the meshes were fine enough to reproduce qualitatively solutions from the literature but not quantitatively, hence more memory could be needed. For 3D lid-driven flows, computations with a direct solver fail to work as the method requires more than 64Gb, exceeding what is available on our desktop. By switching to the iterative solver HIPS, the memory requirement drops to 16Gb, for 3 millions of unknown.

Chapter 6

Conclusions and Future Works

6.1 Concluding remarks

In Chapter 1, we reviewed several time-stepping methods and found out that semi-implicit methods (e.g., CNAB method) are very competitive in terms of accuracy and efficiency when solving unsteady incompressible Navier–Stokes equations. This sets our fundamental motivation to develop more robust time-stepping methods based on their order of accuracy (in time), stability, and CPU and memory efficiency. This is possible with semi-implicit approach.

In Chapter 2, we first studied several high-order semi-implicit time-stepping methods; namely, the known method SBDF and the newly-developed/modified methods DC, GM and GM-SRM. Rigorous numerical assessment had been conducted to verify their theoretical rate of convergence. Our analysis also includes comparisons in terms of numerical error, required CPU time and numerical stability. Several parameters of interest were benchmarked with the values found in the literature. For 2D unsteady flows, SBDF and DC methods are the most efficient methods based on ratio error size over CPU time, followed by GM and GM-SRM methods.

In Chapter 3, we showed that SBDF methods with **grad**-div stabilization are more stable and accurate based on the strength of the local mass conservation while maintaining the advantage of being very efficient when computing 2D unsteady flows. Also, several nonlinear extrapolation schemes were studied which have great potential to reduce the numerical errors on velocity and pressure while bringing the accuracy of this method closer to that of BDF (fully-implicit) methods. The enhanced SBDF methods showed their robustness when computing (2D) nearly turbulent flows.

In the first part of Chapter 4 we improve the DC, GM and GM-SRM methods with

the introduction of new extrapolation formulae for the nonlinear advection term. The new extrapolation formulae further reduce the error on velocity and pressure while maintaining the theoretical order of convergence in time, and give better numerical stability depending on the test cases. We observed that a triggered numerical error propagating in time is not generated with these improved formulae. The second part of this chapter includes results that improve the accuracy and efficiency of **grad**-div splitting proposed in [24], and allow us to reduce the size of the resulting linear system in GM methods when solving larger flow problems.

In Chapter 5, we showcased the computation of 3D flows using GM and GM-SRM methods but only for 2^{nd}-order schemes. To handle the millions of degrees of freedom arising in the computation of 3D lid-driven cavity flows, a GM-2 method in conjunction with **grad**-div splitting is employed to break the resulting linear system into smaller systems. Also, an efficient iterative solver known as HIPS is used to handle these moderate size linear systems. The numerical results for the 3D lid-driven cavity at $Re = 100$, 400 and 1 000 (steady flows) produce very satisfactory qualitative results. For the 3D lid-driven cavity at $Re = 1\,970$ (unsteady flows), our results have not yet reached a fully periodic state but the partial results showed that the flow is unsteady, with the general features well reproduced. Our results for 3D flows around the cylinder at $Re = 100$ and 3D flows around the sphere produce fully periodic solution with the Strouhal number very close to the one in the literature. The qualitative estimates of lift and drag coefficients for both cases are however falling short of the values in the literature because the space discretization in our experiments is not fine enough. Nonetheless, the GM-SRM-2 method produces with good accuracy the right periodic flows; at least a good accuracy is achieved in time. The ratio CPU time over degree of freedom per time step showed that high-order SBDF methods are the most CPU-efficient (with MUMPS). The usage of computer memory can be significantly reduced by using an iterative solver, such as HIPS, to compute 3D flows. For about the same number of unknowns, the resulting global matrices in 3D problems are sparser (more null elements), with a larger bandwidth than those in 2D problems, making iterative linear solvers more competitive compared to direct solvers (e.g., MUMPS) in 3D.

6.2 Future works

It would be interesting to investigate the potential of GM and GM-SRM methods with order of accuracy larger than 3 since these methods have better stability properties than SBDF methods. The major issue however would be the efficiency since these higher-order methods ($k > 3$) require more evaluations of auxiliary variables and involve more subproblems. One way to improve efficiency is to parallelize the

computations based on the subproblems since they can be made "independent" of each other; e.g., by performing the computations of the upper subproblems one or two steps ahead of the lower subproblems (according to the dependence tree in Figure 4.1).

We could investigate the use of other mixed finite elements, in particular the conservative pair of elements for space discretization. This may result in different numerical behaviour; for instance, regarding the need for the stabilization term in the Navier–Stokes equations which is closely linked to mass conservation. Such numerical study has not been attempted.

For Taylor–Hood elements, the optimal choice for the stabilization parameter λ for GM methods, and α_1 and α_2 for GM-SRM methods, is still an open problem. This particular topic connects many aspects of the computations; e.g., minimizing the velocity and pressure error, and fulfilling mass conservation, numerical dissipations and stability when dealing with flows with high Reynolds numbers. A rigorous mathematical analysis is needed to improve the selection of λ for Navier–Stokes equations, and should be supported by numerical test cases.

For finite element methods, fancy iterative solvers play a vital role. Unfortunately, our work has paid little attention to this critical area. Such work is required to handle more degrees of freedom and larger problems. This should include domain decomposition techniques in conjunction with parallel implementations. Incorporating more efficient techniques to solve larger linear systems (e.g., multigrid methods) and the development of more specific preconditioners are other avenues. In the current computational setting, the selection of optimal parameters in MUMPS and HIPS for better numerical efficiency has been left out due to the time constraint in completing this thesis. Such work would allow our high-order time stepping methods to solve large scale industrial flow problems.

Appendix A

Fundamental Theorems on Finite Element Methods

Theorem A.0.1 (**Error estimates for Stokes equations using $\mathbb{P}_2$-$\mathbb{P}_1$ finite elements (see Ern and Guermond, [13])**). *Assume that the solution $(\mathbf{u}, p)$ to the Stokes problem (1.1.1) is sufficiently smooth, i.e., $\mathbf{u} \in [H^3(\Omega) \cap H_0^1(\Omega)]^d$ and $p \in H^2(\Omega) \cap L_0^2(\Omega)$. Then, the solution $(\mathbf{u}_h, p_h)$ of (1.1.3) with $\mathcal{X}_h$ and $\mathcal{M}_h$, as defined in (1.1.4) satisfies*

$$\forall h, \quad \|\mathbf{u} - \mathbf{u}_h\|_{1,\Omega} + \|p - p_h\|_{0,\Omega} \leq ch^2(\|\mathbf{u}\|_{3,\Omega} + \|p\|_{2,\Omega}),$$

for some constant $c > 0$. Moreover, if the Stokes problem has smoothing properties,

$$\forall h, \quad \|\mathbf{u} - \mathbf{u}_h\|_{0,\Omega} \leq ch^3(\|\mathbf{u}\|_{3,\Omega} + \|p\|_{2,\Omega}).$$

Appendix B

Useful Concepts, Definitions and Numerical Tools

B.1 Regular Triangulation

For a given set S, the diameter of S is defined by [28]

$$\text{diam}(S) = \sup\{|\mathbf{x} - \mathbf{y}| \ : \ \mathbf{x}, \mathbf{y} \in S\}.$$

The family of meshes $\{\mathcal{T}_h\}_{h>0}$ discretizing a domain Ω, is called regular if it fulfills the following conditions.

(1) $\lim_{h \downarrow 0} \max\{\text{diam} \mid T_K \in \mathcal{T}_h\} = 0.$

(2) There exist a constant $\sigma > 0$ independent of h such that

$$\frac{\rho(T_K)}{\text{diam}(T_K)} \geq \sigma, \quad \forall T_K \in \mathcal{T}_h, \tag{B.1.1}$$

where $\rho(T_K)$ are the diameter of the inscribed circle of T_K.

B.2 Newton–Raphson's method

We are interested in computing solutions $\overline{\mathbf{X}} \in \mathbb{R}^n$ of the system $\mathbf{F}(\overline{\mathbf{X}}) = 0$, where $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ is a general nonlinear function. Newton's method consists in iterating

$$\mathbf{X}^{n+1} \ = \ \mathbf{X}^n - [\mathbf{F}'(\mathbf{X}^n)]^{-1}\mathbf{F}(\mathbf{X}^n) \tag{B.2.1}$$

starting from some $\mathbf{X}^0 \in \mathbb{R}^n$. When applying Newton's method to the stationary Navier–Stokes equations, we define the variable

$$\mathbf{X} = \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix}$$

and the function

$$\mathbf{F}(\mathbf{X}) = \mathbf{F}\left(\begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} \right) = \begin{bmatrix} -\nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p - \mathbf{f} \\ \nabla\cdot\mathbf{u} + \epsilon p \end{bmatrix}. \tag{B.2.2}$$

Applying (B.2.1) to the (B.2.2), we get

$$\mathbf{F}'\left(\begin{bmatrix} \mathbf{u_k} \\ p_k \end{bmatrix} \right) \begin{bmatrix} \mathbf{u}_{k+1} - \mathbf{u}_k \\ p_{k+1} - p_k \end{bmatrix} = -\mathbf{F}\left(\begin{bmatrix} \mathbf{u_k} \\ p_k \end{bmatrix} \right)$$

or written in a component-wise form

$$\begin{aligned} -\nu\Delta\mathbf{u}_{k+1} + \mathbf{u}_{k+1}\cdot\nabla\mathbf{u}_k + \mathbf{u}_k\cdot\nabla\mathbf{u}_{k+1} + \nabla p_{k+1} &= \mathbf{f} + \mathbf{u}_k\cdot\nabla\mathbf{u}_k, & \text{(B.2.3)} \\ \nabla\cdot\mathbf{u}_{k+1} - \epsilon p_{k+1} &= 0. & \text{(B.2.4)} \end{aligned}$$

From a practical point of view, few other issues have to be considered: first is that (B.2.3) and (B.2.4) are written in variational form and second, a stopping criterion is required to assess the convergence of $\mathbf{X}^n$. One way to achieve this is by fixing $\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_{L^2(\Omega)} < tol$ where $tol > 0$ is any user-define small real number.

To write down the general formulation of Newton's method for nonstationary Navier–Stokes equations with time-stepping methods requires care. For instance, when involving multistep methods (e.g., BDF), one may simply add the time-discretization term and freeze the past step value(s) whenever the Newton's method is iterated at each time step. When using BDF-2 for time-stepping, the algorithm is written as follows: Given a proper initialization of $\mathbf{u}^0$ and $\mathbf{u}^1$, and we fix $\mathbf{u}_0^{n+2} := \mathbf{u}^{n+1}$ by convention, for $n = 0, 1, 2, \ldots$ we seek the solution $(\mathbf{u}^{n+2}, p^{n+2})$ through the convergence of Newton's iteration for some value $k \in \mathbb{N}$ (with a proper stopping criteria) using

$$\begin{aligned} \frac{3\mathbf{u}_{k+1}^{n+2} - 4\mathbf{u}^{n+1} + \mathbf{u}^n}{2\tau} - \nu\Delta\mathbf{u}_{k+1}^{n+2} + \mathbf{u}_{k+1}^{n+2}\cdot\nabla\mathbf{u}_k^{n+2} + \mathbf{u}_k^{n+2}\cdot\nabla\mathbf{u}_{k+1}^{n+2} & \\ +\nabla p_{k+1}^{n+2} - \mathbf{f}^{n+2} &= \mathbf{u}_k^{n+2}\cdot\nabla\mathbf{u}_k^{n+2}, \\ \nabla\cdot\mathbf{u}_{k+1}^{n+2} - \epsilon p_{k+1}^{n+2} &= 0. \end{aligned}$$

At convergence of Newton's method, we fix $\mathbf{u}^{n+2} := \mathbf{u}_{k+1}^{n+2}$ and $p^{n+2} := p_{k+1}^{n+2}$ for some value $k \in \mathbb{N}$ and for all $n = 0, 1, 2, \ldots$

B.3 Bound associated with the nonlinear term

We define the nonlinear term $B(\mathbf{u}) = \mathbf{u} \cdot \nabla \mathbf{u}$ where $\mathbf{u} \in \mathbb{R}^d$, $d = 2, 3$ is a vector field. Let $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^d$ are non-zero vector fields. Then following bound holds

$$\big|B(\mathbf{a}+\mathbf{b}) - B(\mathbf{a}+\mathbf{c})\big| \leq \big|(\mathbf{a}+\mathbf{b}) \cdot \nabla(\mathbf{b}-\mathbf{c})\big| + \big|(\mathbf{b}-\mathbf{c}) \cdot \nabla(\mathbf{a}+\mathbf{c})\big|. \qquad \text{(B.3.1)}$$

Proof:

$$\begin{aligned}
B(\mathbf{a}+\mathbf{b}) - B(\mathbf{a}+\mathbf{c}) &= (\mathbf{a}+\mathbf{b}) \cdot \nabla(\mathbf{a}+\mathbf{b}) - (\mathbf{a}+\mathbf{c}) \cdot \nabla(\mathbf{a}+\mathbf{c}) \\
&= \mathbf{a} \cdot \nabla \mathbf{a} + \mathbf{a} \cdot \nabla \mathbf{b} + \mathbf{b} \cdot \nabla \mathbf{a} + \mathbf{b} \cdot \nabla \mathbf{b} - \mathbf{a} \cdot \nabla \mathbf{a} - \mathbf{a} \cdot \nabla \mathbf{c} - \\
&\quad \mathbf{c} \cdot \nabla \mathbf{a} - \mathbf{c} \cdot \nabla \mathbf{c} \\
&= \mathbf{a} \cdot \nabla(\mathbf{b}-\mathbf{c}) + (\mathbf{b}-\mathbf{c}) \cdot \nabla \mathbf{a} + \mathbf{b} \cdot \nabla \mathbf{b} - \mathbf{b} \cdot \nabla \mathbf{c} + \mathbf{b} \cdot \nabla \mathbf{c} - \\
&\quad \mathbf{c} \cdot \nabla \mathbf{c} \\
&= \mathbf{a} \cdot \nabla(\mathbf{b}-\mathbf{c}) + (\mathbf{b}-\mathbf{c}) \cdot \nabla \mathbf{a} + \mathbf{b} \cdot \nabla(\mathbf{b}-\mathbf{c}) + (\mathbf{b}-\mathbf{c}) \cdot \nabla \mathbf{c} \\
&= (\mathbf{a}+\mathbf{b}) \cdot \nabla(\mathbf{b}-\mathbf{c}) + (\mathbf{b}-\mathbf{c}) \cdot \nabla(\mathbf{a}+\mathbf{c}) \\
\big|B(\mathbf{a}+\mathbf{b}) - B(\mathbf{a}+\mathbf{c})\big| &\leq \big|(\mathbf{a}+\mathbf{b}) \cdot \nabla(\mathbf{b}-\mathbf{c})\big| + \big|(\mathbf{b}-\mathbf{c}) \cdot \nabla(\mathbf{a}+\mathbf{c})\big|.
\end{aligned}$$

∎

Appendix C

Several modeling considerations

C.1 Boundary conditions for the 2D lid-driven cavity flow at $Re = 8\,500$

The domain $\Omega = (0,1)^2$ is typically used to compute 2D lid-driven cavity flows. Homogeneous Dirichlet boundary conditions with velocity $\mathbf{u}|_\Gamma = (0,0)$ is prescribed along left, right and lower boundaries. Along the upper lid, the velocity $\mathbf{u}|_\Gamma = (1,0)$ is set to provide a constant flow. The boundary settings at the corner produce discontinuous solutions for the lid-driven cavity. We consider two types of boundary conditions at the two upper corners. Figure C.1 shows the "watertight cavity" (denoted as CAVITY-1) which fixes $\mathbf{u}|_\Gamma = (0,0)$ (in blue) at the top corners while the "leaky cavity" (denoted as CAVITY-2) fixes $\mathbf{u}|_\Gamma = (1,0)$ (in red) at the top corners.

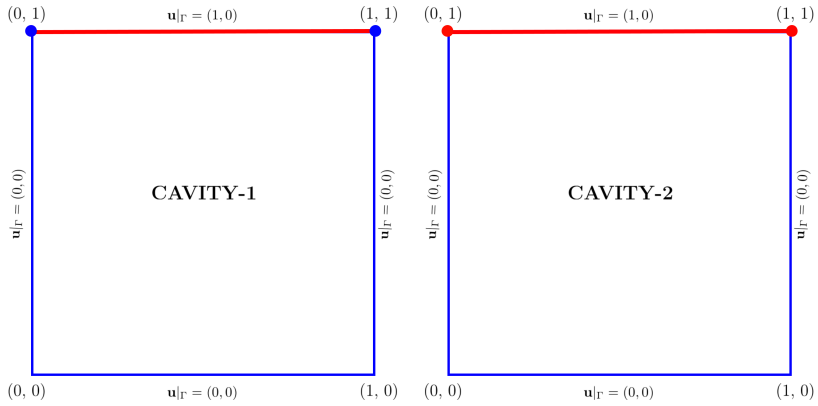

Figure C.1: Two types of domain setting for 2D lid-driven cavity: "Watertight cavity" (CAVITY-1) and "leaky cavity" (CAVITY-2).

We compute the 2D lid-driven cavity using both CAVITY-1 and CAVITY-2 and we compare the numerical results. The discretization of the domain is done using non-uniform triangular mesh with 120 triangles along each edge of the square. This results in a varying mesh size $0.006\,28 \leq h_K \leq 0.016\,60$ with 34 164 vertices and 17 323 elements. With $\mathbb{P}_2$-$\mathbb{P}_1$ elements, a total of 154 941 unknowns are generated for both velocity and pressure. We compute the 2D lid-driven cavity using different combinations of methods, boundary conditions, Reynolds numbers and time steps. All these are given as follows:

(a) CAVITY-1, BDF-2 method, $Re = 8\,000$, $\tau = 0.05$
(b) CAVITY-1, DC-2 method, $Re = 8\,000$, $\tau = 0.002\,7$
(c) CAVITY-2, BDF-2 method, $Re = 8\,000$, $\tau = 0.05$
(d) CAVITY-2, BDF-2 method, $Re = 8\,500$, $\tau = 0.01$.

Here we consider two values of Reynolds number; namely 8000 and 8500. Many authors have reported that the first Hopf bifurcation can be triggered at about $Re = 8000$ (see [2, 8, 14, 66]) while [18] reported that bifurcation is possible with a smaller critical value $Re_{cri} < 7500$. This can only be achieved for a sufficient long simulation time T. Computations are initialized from the state of rest. The time evolution of x-velocity u are monitored from time to time at three points; i.e., near bottom left $(0.2, 0.3)$, bottom right $(0.8, 0.3)$ and top right $(0.8, 0.7)$.

Figure C.2 shows the x-velocity u monitored at three points for $t \in [1\,000, 1\,200]$ using the above test cases. We observed that the x-velocity u at all monitoring points becomes fully periodic when CAVITY-1 is implemented with $Re = 8\,000$. However, both computations with CAVITY-2 produces steady solution. Even with the higher Reynolds number, $Re = 8\,500$, and smaller time step $\tau = 0.01$, we were not able to trigger flow instabilities in CAVITY-2.

In [41], we produced unsteady periodic flows with all methods for CAVITY-1 at $Re = 8\,500$ with various time steps $\tau = 0.001, 0.001\,4$ and $0.002\,5$. We tested that the periodicity is observed with CAVITY-2 at $Re = 10\,000$ and time step $\tau = 0.01$ using BDF-2 method (result not shown). The "leaky cavity" (CAVITY-2) is a type of regularized cavity which results in lower sensitivity to flow instabilities induced by high Reynolds number. On the basis of these results, we were convinced that the "watertight cavity" (CAVITY-1) is the best option to compute 2D lid-driven cavity at $Re = 8\,500$.

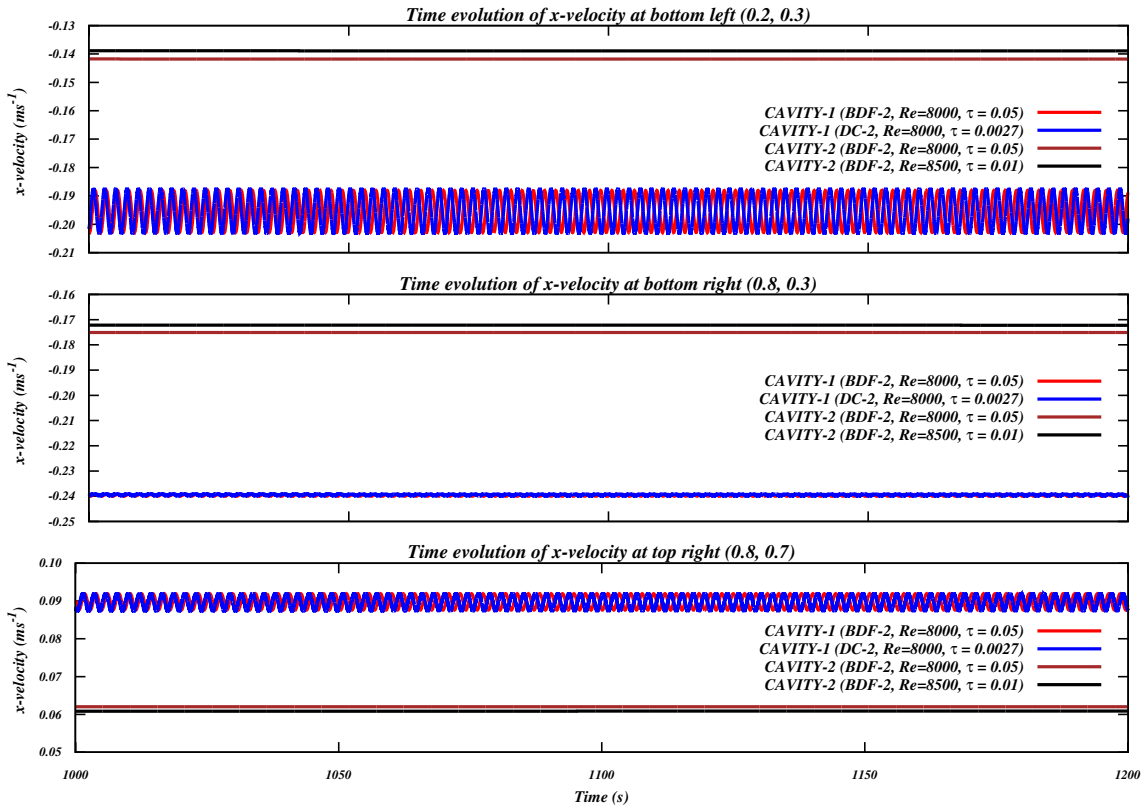

Figure C.2: Time evolution of the x-velocity u at three locations with different methods, model and boundary conditions used to compute 2D lid-driven cavity at $Re = 8\,500$.

Bibliography

[1] U M Ascher, S J Ruuth, and B T R Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

[2] F Auteri, N Parolini, and L Quartapelle. Numerical investigation on the stability of singular driven cavity flow. *Journal of Computational Physics*, 183(1):1–25, 2002.

[3] O Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1996.

[4] M Benzi, G H Golub, and J Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[5] M Benzi and A J Wathen. Some preconditioning techniques for saddle point problems. In *Model Order Reduction: Theory, Research Aspects and Applications*, pages 195–211. Springer, 2008.

[6] D Boffi, F Brezzi, and M Fortin. *Mixed Finite Element Methods and Applications*, volume 44. Springer, 2013.

[7] F Brezzi and M Fortin. *Mixed and Hybrid Finite Element Methods*, volume 15. Springer Science & Business Media, 2012.

[8] C-H Bruneau and M Saad. The 2d lid-driven cavity problem revisited. *Computers & Fluids*, 35(3):326–348, 2006.

[9] A J Chorin. On the convergence of discrete approximations to the Navier–Stokes equations. *Mathematics of Computation*, 23(106):341–353, 1969.

[10] X Dai, J Sun, and X Cheng. Error estimates for an operator-splitting method for Navier–Stokes equations: Second-order schemes. *Journal of Computational and Applied Mathematics*, 231(2):696–704, 2009.

[11] T A Davis. *Direct Methods for Sparse Linear Systems*, volume 2. SIAM, 2006.

[12] H C Elman, D J Silvester, and A J Wathen. *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*. Oxford University Press (UK), 2014.

[13] A Ern and J-L Guermond. *Theory and Practice of Finite Elements*, volume 159. Springer Science & Business Media, 2013.

[14] A Fortin, M Jardak, J J Gervais, and R Pierre. Localization of Hopf bifurcations in fluid flow problems. *International Journal for Numerical Methods in Fluids*, 24(11):1185–1210, 1997.

[15] M Fortin and R Glowinski. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-value Problems*. Elsevier, 2000.

[16] J Gaidamour and P Hénon. A parallel direct/iterative solver based on a Schur complement approach. In *Computational Science and Engineering, 2008. CSE'08. 11th IEEE International Conference on*, pages 98–105. IEEE, 2008.

[17] J Gaidamour, P Hénon, and Y Saad. Hips users guide, 2010.

[18] U K N G Ghia, K N Ghia, and C T Shin. High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, 1982.

[19] F X Giraldo. The Lagrange–Galerkin spectral element method on unstructured quadrilateral grids. *Journal of Computational Physics*, 147(1):114–146, 1998.

[20] R Glowinski and O Pironneau. Finite element methods for Navier–Stokes equations. *Annual review of fluid mechanics*, 24(1):167–204, 1992.

[21] S Goldstein. *Modern Developments in Fluid Dynamics: An Account of Theory and Experiment Relating to Boundary Layers, Turbulent Motion and Wakes*, volume 1. Clarendon Press, 1938.

[22] P M Gresho and R L Sani. *Incompressible Flow and the Finite Element Method. Volume 2: Incompressible Flow and Finite Element*. John Wiley and Sons, Inc., New York, NY (United States), 1998.

[23] J L Guermond and P D Minev. High-order artificial compressibility for the Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 37(6):2656–2681, 2015.

[24] J L Guermond and P D Minev. High-order time stepping for the Navier–Stokes equations with minimal computational complexity. *Journal of Computational and Applied Mathematics*, 2016.

[25] J L Guermond, P D Minev, and J Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44):6011–6045, 2006.

[26] Y He. The Crank–Nicolson/Adams–Bashforth scheme for the time-dependent Navier–Stokes equations with nonsmooth initial data. *Numerical Methods for Partial Differential Equations*, 28(1):155–187, 2012.

[27] Y He and W Sun. Stability and convergence of the Crank–Nicolson/Adams–Bashforth scheme for the time-dependent Navier–Stokes equations. *SIAM Journal on Numerical Analysis*, 45(2):837–869, 2007.

[28] F Hecht. Freefem++ users manual. Version 3.22, 2013.

[29] R Iwatsu, J M Hyun, and K Kuwahara. Analyses of three-dimensional flow calculations in a driven cavity. *Fluid Dynamics Research*, 6(2):91–102, 1990.

[30] R Iwatsu, K Ishii, T Kawamura, K Kuwahara, and J M Hyun. Numerical simulation of three-dimensional flow structure in a driven cavity. *Fluid Dynamics Research*, 5(3):173–189, 1989.

[31] E W Jenkins, V John, A Linke, and L G Rebholz. On the parameter choice in grad-div stabilization for the Stokes equations. *Advances in Computational Mathematics*, 40(2):491–516, 2014.

[32] V John. Reference values for drag and lift of a two-dimensional time-dependent flow around a cylinder. *International Journal for Numerical Methods in Fluids*, 44(7):777–788, 2004.

[33] T A Johnson and V C Patel. Flow past a sphere up to a Reynolds number of 300. *Journal of Fluid Mechanics*, 378:19–70, 1999.

[34] V Kalro and T Tezduyar. Parallel 3d computation of unsteady flows around circular cylinders. *Parallel Computing*, 23(9):1235–1248, 1997.

[35] J Kim and P Moin. Application of a fractional-step method to incompressible Navier–Stokes equations. *Journal of Computational Physics*, 59(2):308–323, 1985.

[36] W W Kim and S Menon. An unsteady incompressible Navier–Stokes solver for large eddy simulation of turbulent flows. *International Journal for Numerical Methods in Fluids*, 31(6):983–1017, 1999.

[37] W Kress and P Lötstedt. Time step restrictions using semi-explicit methods for the incompressible Navier–Stokes equations. *Computer methods in applied mechanics and engineering*, 195(33):4433–4447, 2006.

[38] A Leonard. Energy cascade in large-eddy simulations of turbulent fluid flows. *Advances in Geophysics*, 18:237–248, 1975.

[39] A Liberzon, Yu Feldman, and A Y Gelfgat. Experimental observation of the steady-oscillatory transition in a cubic lid-driven cavity. *Physics of Fluids (1994-present)*, 23(8):084106, 2011.

[40] P Lin. A sequential regularization method for time-dependent incompressible Navier–Stokes equations. *SIAM Journal on Numerical Analysis*, 34(3):1051–1071, 1997.

[41] K C Loy and Y Bourgault. On efficient high-order semi-implicit time-stepping schemes for unsteady incompressible Navier–Stokes equations. *Computers & Fluids*, 148:166–184, 2017.

[42] S Mittal and A Raghuvanshi. Control of vortex shedding behind circular cylinder for flows at low Reynolds numbers. *International Journal for Numerical Methods in Fluids*, 35(4):421–447, 2001.

[43] M Olshanskii, G Lube, T Heister, and J Löwe. Grad-div stabilization and subgrid pressure models for the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 198(49):3975–3988, 2009.

[44] M Olshanskii and A Reusken. Grad-div stablilization for Stokes equations. *Mathematics of Computation*, 73(248):1699–1718, 2004.

[45] O Østerby. Five ways of reducing the Crank–Nicolson oscillations. *BIT Numerical Mathematics*, 43(4):811–822, 2003.

[46] T W Pan and R Glowinski. A projection/wave-like equation method for the numerical simulation of incompressible viscous fluid flow modeled by the Navier–Stokes equations. *Computational Fluid Dynamics Journal*, 9(2):28–42, 2000.

[47] O Pironneau. On the transport-diffusion algorithm and its applications to the Navier–Stokes equations. *Numerische Mathematik*, 38(3):309–332, 1982.

[48] A Quarteroni and A Valli. *Numerical Approximation of Partial Differential Equations*, volume 23. Springer Science & Business Media, 2008.

[49] B N Rajani, A Kandasamy, and S Majumdar. Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modelling*, 33(3):1228–1247, 2009.

[50] F W Roos and W W Willmarth. Some experimental results on sphere and disk drag. *AIAA Journal*, 9(2):285–291, 1971.

[51] Y Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[52] P Sagaut. *Large Eddy Simulation for Incompressible Flows: An Introduction*. Springer Science & Business Media, 2006.

[53] H Sakamoto and H Haniu. A study on vortex shedding from spheres in a uniform flow. *Journal of Fluids Engineering*, 112(4):386–392, 1990.

[54] M Schäfer, S Turek, F Durst, E Krause, and R Rannacher. *Benchmark Computations of Laminar Flow Around a Cylinder*. Springer, 1996.

[55] P N Shankar and M D Deshpande. Fluid mechanics in the driven cavity. *Annual Review of Fluid Mechanics*, 32(1):93–136, 2000.

[56] J Shen. On a new pseudocompressibility method for the incompressible Navier–Stokes equations. *Applied Numerical Mathematics*, 21(1):71–90, 1996.

[57] H Si. Tetgen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):11, 2015.

[58] J C Simo and F Armero. Unconditional stability and long-term behavior of transient algorithms for the incompressible Navier–Stokes and Euler equations. *Computer Methods in Applied Mechanics and Engineering*, 111(1):111–154, 1994.

[59] S P Singh and S Mittal. Flow past a cylinder: Shear layer instability and drag crisis. *International Journal for Numerical Methods in Fluids*, 47(1):75–98, 2005.

[60] A Smith and D Silvester. Implicit algorithms and their linearization for the transient incompressible Navier–Stokes equations. *IMA journal of numerical analysis*, 17(4):527–545, 1997.

[61] O Soto, R Löhner, J R Cebral, and R Codina. A time-accurate implicit monolithic finite element scheme for incompressible flow problems. In *Proc. ECCOMAS CFD*, 2001.

[62] G Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.

[63] G Strang. *Linear Algebra and its Applications*. Brooks/Cole Thomson Learning Inc, 1988.

[64] E Süli and D F Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

[65] R Temam. *Navier–Stokes Equations: Theory and Numerical Analysis*, volume 343. American Mathematical Soc., 2001.

[66] G Tiesinga, F W Wubs, and A E P Veldman. Bifurcation analysis of incompressible flow in a driven cavity by the Newton–Picard method. *Journal of Computational and Applied Mathematics*, 140(1):751–772, 2002.

[67] A G Tomboulides. Direct and large-eddy simulation of wake flows: Flow past a sphere. *Previews of Heat and Mass Transfer*, 6(21):563–564, 1995.

[68] S Turek. *A Comparative Study of Some Time-Stepping Techniques for the Incompressible Navier–Stokes Equations: From Fully Implicit Nonlinear Schemes to Semi-implicit Projection Methods*. Citeseer, 1995.

[69] S Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*, volume 6. Springer Science & Business Media, 1999.

[70] D Xiu and G E Karniadakis. A semi-Lagrangian high-order method for Navier–Stokes equations. *Journal of Computational Physics*, 172(2):658–684, 2001.

[71] N N Yanenko. *The Method of Fractional Steps*. Springer, 1971.

[72] Z Zunic, M Hribersek, L Skerget, and J Ravnik. 3d lid driven cavity flow by mixed boundary and finite element method. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006*. Delft University of Technology; European Community on Computational Methods in Applied Sciences (ECCOMAS), 2006.