

UPC

CTTC

Marching faster in time:

**advanced time-integration for
incompressible flows**

Centre Tecnològic de Transferència de Calor
Departament de Màquines i Motors Tèrmics
Universitat Politècnica de Catalunya

Josep Plana Riu
Doctoral Thesis

Marching faster in time: advanced time-integration for incompressible flows

Josep Plana Riu

TESI DOCTORAL

presentada al

Departament de Màquines i Motors Tèrmics
ESEIAAT
Universitat Politècnica de Catalunya

per a l'obtenció del grau de

Doctor per la UPC

Terrassa, 10 de setembre de 2026

Marching faster in time: advanced time-integration for incompressible flows

Josep Plana Riu

Directors de la Tesi

Dr. Carles David Pérez Segarra
Dr. Assensi Oliva Llena

Tutor

Dr. Francesc Xavier Trias Miquel

Tribunal Qualificador

Dr. Carles Olet Casasayas
Universitat Politècnica de Catalunya

Dr. Roel W.C.P. Verstappen
Rijksuniversiteit Groningen

Dr. Francesco Capuano
Universitat Politècnica de Catalunya

Dr. Zhang Hao
Northeastern University

Dr. Cristóbal Cortés Gracia
Universidad de Zaragoza

Als meus pares i a la Gisi

A la Naomi

Acknowledgements

Una tesi doctoral, tot i portar un nom al davant, implica un grup molt gran de persones darrere. Qui em coneix, sap que sóc un despistat, així que començo els agraïments fent un disclaimer: espero no haver-me deixat a ningú.

Primer de tot, m'agradaria agrair als meus directors el seu suport en l'elaboració d'aquesta tesi: als professors Assensi Oliva i Carles David Pérez Segarra, per ser l'ànima del CTTC i per fer-me sentir, a mi i a tots, un més des del primer dia que vaig arribar, sent un estudiant de tercer d'aeronàutica que li agradava "això dels fluids". Des de llavors hem anat iterant per un camí que ens ha portat fins aquí. També vull agrair de tot cor al professor F. Xavier Trias, per les discussions que han enriquit infinitament aquesta tesi, però també per les discussions que portaven a parlar de quantes vegades el Duplantis batria el rècord del món de salt amb perxa... entre molts altres temes que en major o menor mesura han contribuït a completar aquesta tesi. Vas començar sent el meu tutor i un referent, però en aquests quatre anys de tesi t'has convertit en un amic.

No puc deixar-me a ningú de la família del CTTC. Començant per l'Octavi, que potser encara em demana si el meu ordinador és el hokuto, però que m'ha solucionat mil embolics amb llibreries i versions que no acabaven de rutllar. És l'últim membre del grup de CTTC bike, que tants migdies ha amenitzat pujant cap a Estenalles, o cap a Sant Llorenç Savall. Gràcies Eugenio i Jannes, per proporcionar aquests breaks tan necessaris per mantenir una bona salut mental: el rècord de la V el batrem tard o d'hora. També vull agrair al Xavi, l'Àdel, el Guillem, i el Marcial per totes les discussions sobre computació, tfa, o hpc2 que hem tingut durant tot aquest temps. Sense les vostres contribucions, el quart capítol no hauria sigut el mateix.

Tampoc em puc deixar els companys de sala que han fet que el meu pas pel CTTC hagi sigut més amè: gràcies Jordi, Jesús i Eduard! Ni tots amb qui hem compartit pauses eternes de cafè: l'Uri, el Chen, el Mahdi, la Linda, i la Sara, una de les cares noves del lab. Gràcies a tots! No puede faltar Héctor, que aunque haya pasado tiempo

en UK o en Madrid, nunca dejará de ser "friends and family" del lab.

The three months I spent at CWI in Amsterdam probably were the most productive ever, especially thanks to riding my bike daily from Haarlem to Amsterdam back and forth. Thank you Benjamin for making me feel at home in the Netherlands, for all the shared commutes that ended in a hard effort over Spaarndammerdijk or around the Ringvaart (van de Haarlemmermeerpolder, such long names you write in Dutch). I would also like to thank all the people I met there: Henrik, Rik, Syver, Toby, Robin, Dimitris, Pardeep, Anna... which enriched infinitely those three months. I will keep fond memories of "potato Europe" for the rest of my life. Dankjewel!

Tampoc puc oblidar-me dels meus amics, gràcies als quals he pogut desconnectar i abandonar per alguna estona el món del CFD i de la mecànica de fluids. Ni a María, por ser un pilar fundamental en mi vida. Haces que todo sea muy fácil. Gracias por estar siempre a mi lado.

No puc descriure com d'agraït estic amb els meus pares, la meva germana Gisela, i la resta de la meva família, per estar donant-me suport incondicional durant tota aquesta tesi. Malgrat que no entenguin pràcticament res del que hi ha escrit en aquest document, sense ells això hauria sigut completament impossible. Per descomptat, mil gràcies, Naomi. Gràcies per ser amb qui comparteixo les alegries quan tot va sobre rodes, però sobretot per ser el meu suport quan no tot va rodat. Aquesta tesi existeix també en gran part gràcies a tu.

Contents

Abstract	v
1 Introduction	1
1.1 Objectives of the Thesis	5
1.2 Outline of the Thesis	7
References	7
2 Time integration of the Navier-Stokes equations	11
2.1 Semi-discretization in space	11
2.2 Pressure-velocity coupling	12
2.2.1 Pressure correction methods	14
2.2.2 Projection methods	15
2.3 Time integration strategies for the momentum equation	17
2.3.1 Linear Multistep Methods	17
2.3.2 Multistage methods	19
2.3.3 Stability considerations	23
2.3.4 Adaptive timestepping strategies for the momentum equation . .	25
References	27
3 Self-adaptive integration of incompressible flows	35
3.1 Self-adaptive time-integration methods	36
3.1.1 Linear stability analysis of the incompressible Navier-Stokes equations	37
3.1.2 Existing methods for computing the eigenbounds of the convective and diffusive operators	41
3.1.3 Extending AlgEigCD to staggered grids	43

3.2	Self-adaptive integration for Reduced-Order Models (ROMs)	47
3.2.1	Energy-conserving Proper Orthogonal Decomposition (POD)-Galerkin ROMs	48
3.2.2	Generalizing the stability-based adaptive timestepping to ROMs .	51
3.2.3	Analytical stability analysis of ROMs	55
3.2.4	Numerical experiments	59
3.2.5	Accuracy of the eigenbound estimates	65
3.2.6	Computational performance of RedEigCD	67
3.3	Conclusions	69
	References	70
4	Exploiting repeated matrix block structure to accelerate the numerical solution of the incompressible Navier-Stokes equations	75
4.1	Introduction	75
4.2	Increasing the arithmetic intensity of sparse matrix-vector products (SpMV) 78	
4.2.1	Extension to the Poisson equation	80
4.2.2	Parallel-in-time ensemble-averaged simulations	83
4.2.3	Methods to accelerate the transition phase	86
4.2.4	Multiple parameter simulations	92
4.3	Numerical experiments	92
4.3.1	Channel flow	93
4.3.2	Rayleigh-Bénard convection	101
4.3.3	Multiple parameter simulations: airfoil with flaps and slats depleted	103
4.4	Extension to higher-order symmetry-preserving discretizations	106
4.4.1	Application to a projection method	108
4.4.2	Numerical experiments for higher-order discretizations	110
4.5	Conclusions	111
	References	113
5	Concluding Remarks	117
5.1	Future Work	118
A	Published Work	121

Abstract

The computational cost of scale-resolving simulations of incompressible flows, such as Direct Numerical Simulation (DNS) and Large-Eddy Simulation (LES), is largely dictated by the temporal integration of the incompressible Navier-Stokes equations. Traditional timestepping strategies often rely on conservative stability criteria, such as the standard Courant-Friedrichs-Lewy (CFL) condition, which typically underestimates the maximum stable timestep by neglecting the complex interplay between convective and diffusive scales. This thesis addresses the challenge of "marching faster in time" by developing a framework that integrated stability-aware adaptive timestepping with hardware-optimized sparse linear algebra techniques.

The first part of this thesis focuses on the development and extension of self-adaptive time integration methods based on the spectral analysis of the governing operators. By implementing algorithms such as AlgEigCD on staggered grids, it is shown that the maximum stable timestep can be accurately determined by tracking the evolution of the operators' eigenvalues, rather than relying on the conservative CFL condition. This approach allows for significantly larger timesteps without compromising stability or accuracy, leading to substantial reductions in computational time for incompressible flow simulations, without sacrificing the fidelity of the results.

Furthermore, this thesis extends these concepts to the realm of reduced-order modeling (ROM). Using Poincaré's separation theorem for singular values, a theoretical foundation is established to prove that the stability limits of the reduced convective operators are bounded by those of the full-order system. The resulting RedEigCD algorithm enables ROMs to sustain larger timesteps, effectively leveraging their lower dimensionality to further reduce total wall-clock time.

Finally the research addresses the hardware-level bottleneck inherent in modern CFD solvers, which are frequently limited by memory bandwidth during sparse algebraic operations. By exploiting the repeated matrix structure found in parallel-in-time

ensemble averaging simulations, sparse matrix-vector products are replaced by sparse matrix-matrix products. This transformation increases the arithmetic intensity of the computations, enabling higher performance on modern hardware architectures. The effectiveness of these combined mathematical and computational strategies is validated through a series of academic and industrial test cases, demonstrating that significant speedups can be achieved without compromising the accuracy or stability of the simulations. This work provides a practical pathway for accelerating CFD simulations in real-world applications, while also contributing to the theoretical understanding of stability in numerical methods for incompressible flows.

Introduction

“When I meet God, I am going to ask him two questions: Why relativity? And Why turbulence? I really believe he will have an answer for the first.”

– Werner Heisenberg

One could assume the reader has stood in a bridge, looking at the river flowing underneath, and wondered how the water moves, how fast it flows, and how it interacts with the surrounding environment. Or maybe the reader is an avid cyclist, and has felt the feeling of the wake of a bus overtaking them, and wished to understand the physics behind it. Or perhaps the reader is a weather enthusiast, and has been fascinated by the patterns of clouds and storms, and has wanted to predict their behavior, being one example depicted in Figure 1.1. These are just a few examples of the many phenomena that can be explained and predicted using the principles of fluid mechanics, which is the study of how fluids (liquids and gases) behave and interact with their surroundings.

While most of these observations are beautiful and fascinating, observers generally do not understand how insanely complex the underlying physics is, and how difficult it is to predict the behavior of fluids in various scenarios. Such complex that if you prove the existence of a smooth solution to the three-dimensional incompressible Navier-Stokes equations, you will win a million dollars from the Clay Mathematics Institute, as it is one of the seven Millennium Prize Problems. No one has been able (yet) to prove or disprove the existence of such solutions, and, while it is widely believed that they do exist, the proof has eluded mathematicians for decades. This highlights the complexity of the governing equations of fluid mechanics, and the challenges that arise when trying to understand and predict fluid behavior in various scenarios. Unfortunately, this thesis

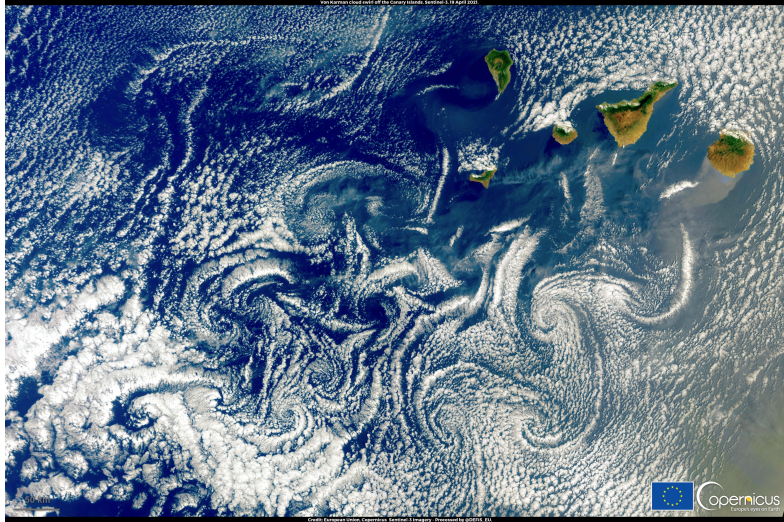


Figure 1.1: Visualization of the von Kármán vortex street behind the Canary Islands, caused by the interaction of the wind with the islands. Image taken from Copernicus Sentinel-3 imagery, processed by ESA.

does not aim to solve the Navier-Stokes existence and smoothness problem, and therefore will not be awarded a million dollars. Instead, this thesis focuses on developing advanced time-integration methods for Computational Fluid Dynamics (CFD) simulations of incompressible flows, which is a crucial step towards understanding and predicting fluid behavior in various applications, such as aerospace, automotive, civil engineering, and environmental science. By improving the efficiency and accuracy of time integration methods for incompressible flows, this thesis aims to enable faster and more efficient simulations, which can lead to better designs, improved performance, and more accurate predictions of fluid behavior under different conditions.

CFD is a branch of fluid mechanics that uses numerical methods and algorithms to solve and analyze problems involving fluid flows. It allows engineers and scientists to simulate the behavior of fluids in various scenarios, such as airflow over an aircraft wing, water flow in a pipe, or weather patterns in the atmosphere. CFD is widely used in industries such as aerospace, automotive, civil engineering, and environmental science to optimize designs, improve performance, and predict fluid behavior under different conditions.

Regardless of the specific application, CFD typically involves solving the Navier-

Stokes equations, which describe the motion of fluid substances. These equations are a set of nonlinear partial differential equations that account for the conservation of mass and momentum. These read

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \quad (1.1)$$

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \mathbf{f}, \quad (1.2)$$

where ρ is the fluid density, $\mathbf{u}$ is the velocity field, p is the pressure, $\boldsymbol{\tau}$ represents the viscous stress tensor, and $\mathbf{f}$ denotes body forces acting on the fluid. The first of these two equations corresponds to the *continuity equation*, which ensures mass conservation, while the second is the *momentum equation*, which describes the balance of forces acting on the fluid. Solving these equations accurately and efficiently is crucial for predicting fluid behavior in various engineering and scientific applications. In case of requiring the conservation of energy, an additional equation is included to account for the energy balance in the fluid. The complexity of these equations often necessitates the use of advanced numerical techniques and resources to obtain solutions for real-world problems.

The methods developed in this thesis focus on incompressible flows, which are characterized by a constant density. In such cases, the continuity equation simplifies to $\nabla \cdot \mathbf{u} = 0$. Moreover, the momentum equation can also be simplified, as the density term can be treated as a constant, and tensile and viscous forces can be more easily computed. Therefore, the Navier-Stokes equations for incompressible flows are given by

$$\nabla \cdot \mathbf{u} = 0 \quad (1.3)$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nabla \cdot (\nu \nabla \mathbf{u}) + \mathbf{f} \quad (1.4)$$

where $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity of the fluid, and μ is the dynamic viscosity. These equations form the basis for many CFD simulations of incompressible flows, and the methods developed in this thesis aim to improve the efficiency and accuracy of solving these equations in various applications.

The incompressible momentum equation is characterized by the presence of two dominant terms: the convective term, characterized by its strong nonlinearity and responsible of transporting quantities (momentum, in this case), and the viscous term, which is responsible for the diffusion of such quantities (again, momentum). The relative importance of these two terms is often quantified by the Reynolds number, which is

defined as $\text{Re} = \frac{UL}{\nu}$, where U is a characteristic velocity, L is a characteristic length scale, and ν is the kinematic viscosity. For high Reynolds numbers, the convective term dominates, leading to turbulent flows with complex dynamics. For low Reynolds numbers, the viscous term dominates, resulting in laminar flows with smooth and predictable behavior.

Every CFD simulation involves several key steps, that can be summarized in: (i) *meshing*, where the physical domain is discretized into smaller elements or control volumes; (ii) *space discretization*, where the governing equations are approximated using numerical methods such as Finite Difference Method (FDM), Finite Volume Method (FVM), or Finite Element Method (FEM); (iii) *solution*, where the resulting algebraic equations from the discretization are solved using iterative or direct solvers; (iv) *time integration*, where the transient behavior of the fluid is captured by advancing the solution in time using methods such as explicit, implicit, or semi-implicit schemes; and (v) *post-processing*, where the results are analyzed and visualized to extract meaningful insights about the fluid flow. While not all CFD simulations require all of these steps, i.e., Reynolds Averaged Navier-Stokes (RANS) simulations typically do not require time integration, as they focus on steady-state solutions, the methods developed in this thesis are focused on improving the efficiency of the time integration step for incompressible flows.

Opposite to other steps in the CFD workflow, such as space discretization, which have seen significant advancements in recent years [1–4], time integration methods for incompressible flows have remained relatively unchanged, even if some efforts have been devoted in pushing their boundaries [5–7]. As a result, many CFD practitioners still rely on traditional time-integration methods, such as explicit Runge-Kutta (RK) or implicit Backward Differentiation Formula (BDF) which may not fully exploit the potential for faster simulations. Therefore, there is a need for advanced time-integration methods that can overcome these limitations and enable faster and more efficient simulations of incompressible flows.

However, opposite to classical Ordinary Differential Equation (ODE) integration, where there is an evolution equation for each degree of freedom, in the case of incompressible flows, the presence of the pressure term, which acts as a Lagrange multiplier to enforce the divergence-free constraint, leads to a Differential-Algebraic Equation (DAE) of index-2 that must be solved simultaneously. This coupling introduces additional challenges for time integration, as it requires the development of specialized methods that can handle the pressure-velocity coupling and ensure stability while allowing for larger time steps. As a result, many traditional time-integration methods may not be

suitable for incompressible flows, and there is a need for advanced techniques that can address these challenges and enable faster simulations. This has been handled in the past by developing methods that decouple the pressure and velocity fields, such as projection methods or fractional step methods [8–10], which generally lead to the time integration of the momentum equation, while the pressure is computed separately to enforce the incompressibility constraint.

Numerical simulations of turbulent flows, characterized by high Reynolds numbers, are particularly challenging due to the wide range of scales that must be resolved, from the largest energy-containing eddies to the smallest dissipative scales [11]. This requires the use of very fine spatial and temporal resolutions, which can lead to prohibitively long simulation times. These fine resolutions lead to a very large number of degrees of freedom, which in turn leads to large linear systems that must be solved at each time step. Therefore, the use of modern systems is required for performing these simulations.

While the peak performance of these systems is continuously increasing, the efficiency of CFD simulations is generally limited not by the peak performance, but by the amount of data that must be transferred between the different levels of memory hierarchy [12], what leads to memory-bound computations. This is therefore the main bottleneck for CFD simulations, and it is crucial to develop methods that can reduce the amount of data transfer and improve the efficiency of memory usage. This can be achieved by developing algorithms that can exploit the structure of the matrices involved in the computations, such as block structures arising from ensemble averaging turbulent flow simulations, which can lead to faster simulations while still maintaining accuracy, without the need for additional hardware resources. Therefore, there is a need for advanced time-integration methods that can not only allow for larger time steps, but also reduce the computational cost and memory requirements of CFD simulations, particularly for turbulent flows.

1.1 Objectives of the Thesis

Therefore, it feels natural to ask the following research questions: (i) Can we develop advanced time-integration methods that can allow for larger time steps while maintaining stability and accuracy for incompressible flows? (ii) Can we exploit the structure of the matrices involved in the computations to improve the efficiency of memory usage and reduce data transfer?

Regarding the first research question, classical stability analysis of time-integration methods has been restricted to applying the Courant-Friedrichs-Lewy (CFL) condition

[13]. While this condition is widely used in practice, especially in commercial software such as ANSYS Fluent or OpenFOAM, it is generally a very conservative estimate of the maximum timestep that can be used in a simulation, and it does not take into account the specific properties of the governing equations or the numerical methods used for space discretization. Therefore, there is a need for more accurate and less conservative stability analysis that can provide better estimates of the maximum timestep that can be used in a simulation, allowing for faster simulations while still maintaining stability and accuracy. This has already been explored for simple multistep methods such as Adams-Bashforth (AB) methods (or parameter-dependent variants of them) [14], but it has not been explored for more methods such as RK, which is widely used in practice due to its simplicity and ease of implementation, as well as its improved stability properties compared to multistep methods.

Moreover, given the current trend towards data-driven methods in CFD, and the observation in literature of the improved stability properties of reduced order models compared to full order models [15], it is natural to ask if a linear stability analysis of the reduced governing equation can be employed to extract larger timesteps for this reduced order model. Therefore, the benefit of the reduced order model is twofold: it can provide a faster simulation due to the reduced number of degrees of freedom, and it can also allow for larger time steps due to its improved stability properties.

With regards to the second research question, the use of block structures arising from ensemble averaging in turbulent flow simulations can lead to faster simulations while still maintaining accuracy, without the need for additional hardware resources. Therefore, there is a need for advanced time-integration methods that can not only allow for larger time steps, but also reduce the computational cost and memory requirements of CFD simulations, particularly for turbulent flows. This can be achieved by developing algorithms that can exploit the structure of the matrices involved in the computations, such as block structures arising from ensemble averaging in turbulent flow simulations, which can lead to faster simulations while still maintaining accuracy, without the need for additional hardware resources.

All in all, the main objectives of this thesis can be summarized as follows:

- Develop advanced time-integration methods that can allow for larger time steps while maintaining stability and accuracy for incompressible flows, both for Full-Order Models (FOMs) and Reduced-Order Models (ROMs).
- Develop algorithms that can exploit the structure of the matrices involved in the computations to improve the efficiency of memory usage and reduce data transfer,

particularly for turbulent flow simulations.

By aiming to achieve these objectives, this thesis seeks to contribute to the advancement of time-integration methods for incompressible flows, enabling faster and more efficient simulations that can lead to better designs, improved performance, and more accurate predictions of fluid behavior under different conditions, which can have significant implications for various engineering and scientific applications.

1.2 Outline of the Thesis

The thesis is structured as follows: Chapter 2 provides a comprehensive review of the state-of-the-art in time-integration methods for incompressible flows, including both explicit and implicit schemes, as well as their stability properties and limitations. Chapter 3 presents the development of advanced time-integration methods that can allow for larger time steps while maintaining stability and accuracy for incompressible flows, both for FOM and ROM. Chapter 4 focuses on the development of algorithms that can exploit the structure of the matrices involved in the computations to improve the efficiency of memory usage and reduce data transfer, particularly for turbulent flow simulations. Finally, Chapter 5 summarizes the main findings of the thesis, discusses their implications, and outlines potential directions for future research in this area.

References

- [1] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187(1):343–368, May 2003.
- [2] F. X. Trias, O. Lehmkuhl, A. Oliva, C. D. Pérez-Segarra, and R. W.C.P. Verstappen. Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids. *Journal of Computational Physics*, 258:246–267, February 2014.
- [3] D. Santos, J. A. Hopman, C. D. Pérez-Segarra, and F. X. Trias. On a symmetry-preserving unconditionally stable projection method on collocated unstructured grids for incompressible flows. *Journal of Computational Physics*, 523, February 2025.

- [4] J. A. Hopman, D. Santos, À. Alsalti-Baldellou, J. Rigola, and F. X. Trias. Quantifying the checkerboard problem to reduce numerical dissipation. *Journal of Computational Physics*, 521:113537, January 2025.
- [5] B. Sanderse and B. Koren. Accuracy analysis of explicit Runge–Kutta methods applied to the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 231(8):3041–3063, April 2012.
- [6] B. Sanderse. Energy-conserving Runge–Kutta methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 233:100–131, January 2013.
- [7] F. Capuano, G. Coppola, L. Rández, and L. de Luca. Explicit Runge–Kutta schemes for incompressible flow with improved energy-conservation properties. *Journal of Computational Physics*, 328:86–94, January 2017.
- [8] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.
- [9] R. Témam. Sur l’approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (I). *Archive for Rational Mechanics and Analysis*, 32(2):135–153, January 1969.
- [10] R. Témam. Sur l’approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (II). *Archive for Rational Mechanics and Analysis*, 33(5):377–385, January 1969.
- [11] A.N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Doklady Akademii Nauk SSSR*, 30:9–13, 1941.
- [12] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, April 2009.
- [13] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100(1):32–74, December 1928.
- [14] F. X. Trias and O. Lehmkuhl. A Self-Adaptive Strategy for the Time Integration of Navier-Stokes Equations. *Numerical Heat Transfer, Part B: Fundamentals*, 60(2):116–134, August 2011.

- [15] Joan Baiges, Ramon Codina, and Sergio Idelsohn. Explicit reduced-order models for the stabilized finite element approximation of the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 72(12):1219–1243, August 2013.

Time integration of the Navier-Stokes equations

“Time is what keeps everything from happening at once.”
– John Wheeler

In this chapter, the time integration of the Incompressible Navier-Stokes (INS) equations is described. First, the INS equations are semi-discretized in space making use of the Method of Lines (MOL) approach, which results in a Differential-Algebraic Equation (DAE) of index 2. Then, the resulting DAE is integrated in time.

2.1 Semi-discretization in space

Discretizing Eqs.(1.3)-(1.4) in space using the MOL approach, which involves discretizing the spatial derivatives while keeping the time derivative continuous, results in

$$M\mathbf{u}_h(t) = \mathbf{g}_h(t), \quad (2.1)$$

$$\Omega \frac{d\mathbf{u}_h(t)}{dt} + C(\mathbf{u}_h(t))\mathbf{u}_h(t) = -\Omega G\mathbf{p}_h(t) + D\mathbf{u}_h(t) + \Omega \mathbf{f}_h(t), \quad (2.2)$$

where $\mathbf{u}_h(t)$ and $\mathbf{p}_h(t)$ are the discrete velocity and pressure fields, respectively, $\mathbf{f}_h(t)$ is the discrete source term, $\mathbf{g}_h(t)$ accounts for time-dependant boundary conditions, M is the discrete divergence operator, Ω is the matrix containing the cell volumes, $C(\mathbf{u}_h(t))$ is the discrete convection operator, G is the discrete gradient operator, and D is the

discrete diffusion operator. The resulting system of equations is a DAE of index 2, which can be solved using various time integration methods.

In the remainder of this chapter, no assumptions are made on the spatial discretization unless explicitly stated, and the discussion is general and applicable to any spatial discretization method, such as Finite Difference Method (FDM), Finite Volume Method (FVM), or Finite Element Method (FEM). However, the author acknowledges that the choice of spatial discretization can have a significant impact on the accuracy and stability of the numerical solution, and it is important to carefully consider the spatial discretization when solving the INS equations.

To guarantee that the resulting system can be solved, the spatial discretization needs to satisfy the inf-sup condition [1], which ensures that the discrete velocity and pressure spaces are compatible and that the resulting system of equations is well-posed. This condition is crucial for the stability and accuracy of the numerical solution, and it must be carefully considered when choosing the spatial discretization method. This condition is satisfied by using compatible discretizations, e.g., staggered grids for FDM and FVM methods [2], or Taylor-Hood elements for FEM [3]. Non-staggered grids for FDM and FVM methods, or equal-order interpolation for FEM, do not satisfy the inf-sup condition, which can lead to numerical instabilities and inaccurate solutions, such as checkerboarding [4].

2.2 Pressure-velocity coupling

Eqs.(2.1) and (2.2) are coupled through the pressure field $\mathbf{p}_h(t)$, which appears in the momentum equation yet it has no time derivative. This coupling is often referred to as the pressure-velocity coupling, and it is a key aspect of the INS equations that must be carefully handled in order to obtain accurate and stable numerical solutions.

Eq.(2.2) can be rewritten as

$$\frac{d\mathbf{u}_h(t)}{dt} = \mathbf{F}(\mathbf{u}_h(t)) - G\mathbf{p}_h(t), \quad (2.3)$$

where $\mathbf{F}(\mathbf{u}_h(t)) = -\Omega^{-1}C(\mathbf{u}_h(t))\mathbf{u}_h(t) + \Omega^{-1}D\mathbf{u}_h(t) + \mathbf{f}_h(t)$ is the discrete operator that combines the convection, diffusion, and source terms. Differentiating the continuity equation with respect to time and substituting Eq.(2.3) into the resulting equation yields

$$L\mathbf{p}_h = M\mathbf{F}(\mathbf{u}_h(t)) - \dot{\mathbf{g}}_h(t), \quad (2.4)$$

where $L = MG$ is the discrete Laplacian operator. This equation is a Poisson equation for the pressure field $\mathbf{p}_h(t)$, which must be solved at each time step in order to ensure that the velocity field remains divergence-free. Introducing this constraint into Eq.(2.3) yields

$$\frac{d\mathbf{u}_h(t)}{dt} = P\mathbf{F}(\mathbf{u}_h(t)) + GL^{-1}\dot{\mathbf{g}}_h(t), \quad (2.5)$$

where $P = I - GL^{-1}M$ is the projection operator that projects the velocity field onto the divergence-free subspace. This approach lowers the index of the DAE from 2 to 1, which implies that there is no guarantee that all constraints are satisfied [5]. Therefore, using the system of equations in the form of Eqs.(2.1) and (2.2) is often preferred in practice. Therefore, the resulting system can be written as

$$\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_h(t) \\ \dot{\mathbf{p}}_h(t) \end{bmatrix} = \begin{bmatrix} A(\mathbf{u}_h(t)) & G \\ M & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h(t) \\ \mathbf{p}_h(t) \end{bmatrix} + \begin{bmatrix} \mathbf{f}_h(t) \\ \mathbf{g}_h(t) \end{bmatrix}, \quad (2.6)$$

where $A(\mathbf{u}_h(t)) = -C(\mathbf{u}_h(t)) + D$. Given the DAE nature of the system, the pressure acts as a Lagrange multiplier to ensure that the divergence-free constraint is satisfied, as it does not have its own time evolution. This means that the pressure field must be determined at each time step in order to ensure that the velocity field remains divergence-free. Thus, standard Ordinary Differential Equation (ODE) solvers cannot be directly applied without modification [6].

Discretizing Eq.(2.6) yields a system of algebraic equations for the velocity and pressure fields at each time step. Following the notation from Perot [7], this can be expressed as

$$\begin{bmatrix} \mathcal{A}(\mathbf{u}_h^{n+1}) & G \\ M & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^{n+1} \\ \mathbf{p}_h^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_h^{n+1} \\ \mathbf{g}_h^{n+1} \end{bmatrix} + \begin{bmatrix} \Delta t^{-1}I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^n \\ \mathbf{p}_h^n \end{bmatrix}. \quad (2.7)$$

Note that this system of equations is nonlinear, as the momentum operator $\mathcal{A}(\mathbf{u}_h^{n+1})$ depends on the velocity field at the new time step. Therefore, Eq.(2.7) cannot be solved using a matrix inversion, yet requires an iterative strategy. In this context, $\mathcal{A}(\mathbf{u}_h^{n+1})$ is the linearized operator at the current iteration, which makes the system a linear saddle-point problem at each time step. The term $\mathbf{r}_h^{n+1}$ encapsulates the history-dependent terms, as this algebraic structure is generic enough to represent both multistep and multistage schemes. While the history of previous steps is encapsulated in $\mathbf{r}_h^{n+1}$ for multistep methods, for multistage methods, Eq.(2.7) is solved at each stage, and $\mathbf{r}_h^{n+1}$ contains the contributions from the previous stages.

Solving the system of equations in Eq.(2.7) is usually prohibitive, even if preconditioning techniques such as block factorization [8, 9] or augmented Lagrangian methods [10, 11] are used. Instead, various methods have been developed to handle the pressure-velocity coupling, such as projection methods [12–14] or pressure correction methods [15, 16]. Both families of methods typically involve solving a Poisson equation for the pressure field at each time step, which can be computationally expensive but is necessary to ensure the accuracy and stability of the numerical solution. These methods can be derived from the system of equations in Eq.(2.7) by means of a block LU factorization [7], which reads

$$\begin{bmatrix} \mathcal{A} & G \\ M & 0 \end{bmatrix} = \begin{bmatrix} A & 0 \\ M & -M\mathcal{A}^{-1}G \end{bmatrix} \begin{bmatrix} I & \mathcal{A}^{-1}G \\ 0 & I \end{bmatrix}. \quad (2.8)$$

Depending on how the Schur complement $M\mathcal{A}^{-1}G$ is approximated, different methods can be derived. For instance, if the Schur complement is approximated by the discrete Laplacian operator $L = MG$, then projection methods are obtained. On the other hand, if the Schur complement is approximated by a more accurate operator, such as $M\tilde{\mathcal{A}}^{-1}G$, then pressure correction methods are obtained, where $\tilde{\mathcal{A}}$ is an approximation of the momentum operator $\mathcal{A}$.

2.2.1 Pressure correction methods

Starting from the block LU factorization, Patankar and Spalding [15] first derived the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE), which key modeling step is the approximation of the Schur complement by replacing $\mathcal{A}^{-1}$ by D^{-1} , where $D = \text{diag}(\mathcal{A})$ is the diagonal of the momentum operator. This leads to a Poisson-like equation for the pressure correction, which coefficients are not fixed and instead depend on the momentum operator.

In general, SIMPLE follow this structure:

1. Calculate a predictor velocity field $\mathcal{A}\mathbf{u}_h^* = \mathbf{r}_h - G\mathbf{p}_h^n$.
2. Determine a pressure-correction step by solving $\tilde{S}(\delta\mathbf{p})_h = M\mathbf{u}_h^*$, where $\tilde{S} = MD^{-1}G$ is the approximate Schur complement.
3. Update the velocity and pressure fields using $\mathbf{u}_h^{n+1} = \mathbf{u}_h^* - D^{-1}G(\delta\mathbf{p})_h$ and $\mathbf{p}_h^{n+1} = \mathbf{p}_h^n + \alpha_p(\delta\mathbf{p})_h$, respectively, where α_p is a relaxation factor for the pressure correction.

This method has later on evolved into SIMPLE variants, such as SIMPLER [16] or SIMPLEC [17], which differ in which triangular matrix of the block LU factorization is used, or the terms that are included in the approximation of the Schur complement.

2.2.2 Projection methods

Projection (or fractional step) methods, on the other hand, are derived by approximating the Schur complement by the discrete Laplacian operator $L = MG$. This leads to a Poisson equation for the pressure field, which must be solved at each time step in order to ensure that the velocity field remains divergence-free and satisfies the incompressibility condition.

Applying Eq.(2.8) to the system of equations in Eq.(2.7) yields

$$\begin{bmatrix} \mathcal{A} & 0 \\ M & -M\mathcal{A}^{-1}G \end{bmatrix} \begin{bmatrix} I & \mathcal{A}^{-1}G \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^{n+1} \\ \mathbf{p}_h^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_h^{n+1} \\ \mathbf{g}_h^{n+1} \end{bmatrix} + \begin{bmatrix} \Delta t^{-1}I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^n \\ \mathbf{p}_h^n \end{bmatrix}, \quad (2.9)$$

which can be split into two steps as follows

$$\begin{bmatrix} \mathcal{A} & 0 \\ M & -M\mathcal{A}^{-1}G \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^* \\ \mathbf{p}_h^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_h^{n+1} + \Delta t^{-1}\mathbf{u}_h^n \\ \mathbf{g}_h^{n+1} \end{bmatrix}, \quad (2.10)$$

$$\begin{bmatrix} I & \mathcal{A}^{-1}G \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^{n+1} \\ \mathbf{p}_h^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_h^* \\ \mathbf{p}_h^{n+1} \end{bmatrix}, \quad (2.11)$$

which split Eq.(2.7) into a predictor step (Eq.(2.10)) and a projection step (Eq.(2.11)). The predictor step involves solving a linear system for the intermediate velocity field $\mathbf{u}_h^*$, which does not satisfy the incompressibility condition. The projection step then projects this intermediate velocity field onto the divergence-free subspace, which ensures that the resulting velocity field $\mathbf{u}_h^{n+1}$ satisfies the incompressibility condition.

While Eqs.(2.10)-(2.11) are exact, they are not practical to solve as they require the solution of a linear system involving the momentum operator $\mathcal{A}$ at each time step. Projection methods approximate $\mathcal{A}^{-1}$ by a simpler operator, such as $\Delta t I$ [12–14], which leads to a Poisson equation for the pressure field. In this seminal approach, the set of

equations reads

$$\mathbf{u}_h^* = \mathbf{u}_h^n + \Delta t \mathbf{r}_h^{n+1}, \quad (2.12)$$

$$L\bar{\mathbf{p}}_h^{n+1} = M\mathbf{u}_h^* - \mathbf{g}_h^{n+1}, \quad (2.13)$$

$$\mathbf{u}_h^{n+1} = \mathbf{u}_h^* - G\bar{\mathbf{p}}_h^{n+1}, \quad (2.14)$$

where $\bar{\mathbf{p}}_h^{n+1} = \Delta t \mathbf{p}_h^{n+1}$. This approximation, however, introduces splitting errors, which affect the accuracy of the numerical solution. Various strategies have been developed to reduce these splitting errors, being the most well-known the incremental pressure correction methods from Van Kan [18], which introduce the gradient of the pressure field at the previous time step into the predictor step, which reduces the splitting errors and improves the accuracy of the numerical solution. This approach leads to the following set of equations

$$\mathbf{u}_h^* = \mathbf{u}_h^n + \Delta t \mathbf{r}_h^{n+1} - G\bar{\mathbf{p}}_h^n, \quad (2.15)$$

$$L\phi_h^{n+1} = M\mathbf{u}_h^* - \mathbf{g}_h^{n+1}, \quad (2.16)$$

$$\mathbf{u}_h^{n+1} = \mathbf{u}_h^* - G\phi_h^{n+1}, \quad (2.17)$$

where $\phi_h^{n+1} = \bar{\mathbf{p}}_h^{n+1} - \bar{\mathbf{p}}_h^n$. This method is second-order accurate in time, and it is widely used in practice due to its simplicity and efficiency. However, it still suffers from splitting errors. To tackle this limitation, Chang et al. [19] eliminated splitting errors by enforcing the incompressibility constraint to machine precision. This, named exact fractional step method, reformulates the discrete momentum equation so that, by construction, the velocity field is exactly divergence-free. This is done by reducing the governing equations to a single linear system for the streamfunction, which is solved at each time step. This system becomes algebraically equivalent to the original system of equations in Eq.(2.7), and therefore it does not generate the numerical boundary layer inherent to classical projection methods.

This numerical boundary layer is originally generated by incorrectly enforcing a homogeneous Neumann boundary condition for the pressure field at the boundaries. As in the physical problem the pressure field at the boundaries has to counteract the flux of vorticity at the wall, the solution needs to transition from the incorrect boundary value to the correct inner value, thus leading to a sharp numerical boundary layer, which thickness scales with $\sqrt{\nu \Delta t}$. This was solved by Timmermans et al. [20] in classical projection methods, where the pressure update equation accounts for a non-divergence-free velocity at the boundaries, and therefore the pressure field at the boundaries is

updated in a way that accounts for the flux of vorticity at the wall, thus eliminating the numerical boundary layer and improving the accuracy of the numerical solution near the boundaries. This was shown to be the only way to eliminate the numerical boundary layer [21] in this non-exact framework.

2.3 Time integration strategies for the momentum equation

After handling the pressure-velocity coupling, the momentum equation needs to be integrated in time. As it is characterized by two very different terms in nature, the non-linear convection and the linear, stiff diffusion, various strategies have been developed to integrate the momentum equation in time. These strategies can be broadly classified into three categories: fully implicit methods, explicit methods, and Implicit-Explicit (IMEX) methods. All these strategies fall into two families: multistep methods, which use information from previous time steps to advance the solution, and multistage methods, which use intermediate stages within a single time step to advance the solution.

Starting from Eq.(2.12), projection methods require integrating in time the momentum equation to determine the value of $\mathbf{u}_h^*$. While until this point, the procedure has been left general, in this section, we tackle the time integration of the momentum equation, which is a key aspect of projection methods. The choice of time integration method can have a significant impact on the accuracy and stability of the numerical solution, and it is important to carefully consider the time integration method when solving the INS equations.

2.3.1 Linear Multistep Methods

Linear Multistep Methods (LMMs) are a family of ODE integration methods that use information from previous steps to advance to solution. By using stored values of the solution at previous time steps, LMMs can achieve higher-order accuracy without the need for intermediate stages, which can be computationally expensive. Applied to a generic ODE of the form $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$, where $\mathbf{y} \in \mathbb{R}^d$, a LMM can be expressed as

$$\sum_{j=0}^k \alpha_j \mathbf{y}^{n-j+1} = \Delta t \sum_{j=0}^k \beta_j \mathbf{f}(\mathbf{y}^{n-j+1}, t^{n-j+1}), \quad (2.18)$$

where $\mathbf{y}^{n-j}$ is the solution at t^{n-j} , and α_j and β_j are the coefficients of the method, which are determined following the order conditions for the desired order of accuracy.

In the case of $\beta_0 = 0$, the method is explicit, while if $\beta_0 \neq 0$, the method is implicit. Generally, methods are normalized such that $\alpha_0 = 1$, yet some schemes do not follow this convention [22, 23].

Applied to the momentum equation, these schemes are used to determine $\mathbf{u}_h^*$. In general,

$$\mathbf{u}_h^* = - \sum_{j=1}^k \alpha_j \mathbf{u}_h^{n-j+1} + \Delta t \sum_{j=0}^k \beta_j \mathbf{F}(\mathbf{u}_h^{n-j+1}), \quad (2.19)$$

where $\mathbf{F}(\mathbf{u}_h^{n-j+1})$ is the discrete operator that combines the convection, diffusion, and source terms at t^{n-j+1} . The choice of LMM can have a significant impact on the accuracy and stability of the numerical solution, and it is important to carefully consider the choice of LMM when solving the INS equations.

While LMMs are widely used in practice given their simplicity and efficiency, they suffer from several limitations. Unlike one-step methods, LMMs are not self-starting, which means that they require a starting procedure to generate the initial values for the first few steps. This can be done using a one-step method, such as a Runge-Kutta method, or by using a lower-order LMM for the first few steps.

Implicit LMMs Considering the most common flavours of implicit LMMs for the numerical integration of the momentum equation, Crank-Nicolson and Backward Differentiation Formula (BDF) methods are widely used in practice, especially BDF2, which forms the backbone of modern projection and pressure-correction schemes [18, 21, 24]. While these are the most relevant, other implicit LMMs have been developed, such as the Two Implicit Advanced Step-point (TIAS), which is tailored to high-order stiff problems [25], while preserving excellent properties such as A-stability. Other implicit LMMs such as BDF2 are also A-stable, yet for orders higher than 2, BDF methods are not A-stable, which can lead to stability issues when applied to stiff problems such as the INS equations. On the other hand, TIAS methods are designed to be A-stable for all orders, which makes them a promising alternative for high-order implicit time integration of the momentum equation.

Explicit LMMs Concerning their explicit counterparts, the use of Adams-Bashforth (AB) methods is widespread in practice and played a central role developing projection methods [26, 27], and remain widely used for the treatment of the nonlinear advection, even if they are unstable for pure advection cases. Other methods such as the leapfrog circumvent this issue, yet they are unstable for pure diffusion cases, which makes them less attractive for the numerical integration of the momentum equation. These second-

order explicit LMMs can be rewritten as a parameter-dependent family of methods, which includes the leapfrog and the AB2 as particular cases, and which can be tuned to improve the stability properties of the method [23]. However, these methods still suffer from stability issues when applied to stiff problems such as the INS equations, which limits their applicability in practice.

2.3.2 Multistage methods

Opposite to LMMs that rely on information from previous steps, multistage methods, such as Runge-Kutta (Runge-Kutta (RK)) methods, advance the solution from step n to $n + 1$ by introducing s intermediate stages within the current interval. Applied to a generic ODE of the form $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$, where $\mathbf{y} \in \mathbb{R}^d$, a RK method can be expressed as

$$\mathbf{y}_i = \mathbf{y}^n + \Delta t \sum_{j=1}^s a_{ij} \mathbf{f}(\mathbf{y}_j, t^n + c_j \Delta t), \quad i = 1, \dots, s, \quad (2.20)$$

$$\mathbf{y}^{n+1} = \mathbf{y}^n + \Delta t \sum_{i=1}^s b_i \mathbf{f}(\mathbf{y}_i, t^n + c_i \Delta t), \quad (2.21)$$

where $\mathbf{y}_i$ is the solution at the intermediate stage i , and a_{ij} , b_i , and c_i are the coefficients of the method, which are determined following the order conditions for the desired order of accuracy. Generally, these coefficients are arranged in the so-called Butcher tableau [28], which provides a convenient way to represent the coefficients of the method. In general, a Butcher tableau reads

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}, \quad (2.22)$$

where $A = [a_{ij}]$ is the matrix of coefficients for the intermediate stages, $\mathbf{c} = [c_1, \dots, c_s]^T$ is the vector of stage time fractions defined as $c_i = \sum_j a_{ij}$, and $\mathbf{b} = [b_1, \dots, b_s]^T$ is the vector of coefficients for the final update. While computationally more expensive than LMMs, RK methods possess the advantage of being self-starting, and allowing larger stability regions compared to LMMs, which can be beneficial for stiff problems such as the incompressible Navier-Stokes equations. For these reasons, they have been extensively used in CFD [29–31].

Applied to the momentum equation,

$$\mathbf{u}_{h,i} = \mathbf{u}_h^n + \Delta t \sum_{j=1}^s a_{ij}(\mathbf{F}(\mathbf{u}_{h,j}) - G\mathbf{p}_j), \quad i = 1, \dots, s, \quad (2.23)$$

$$\mathbf{u}_h^{n+1} = \mathbf{u}_h^n + \Delta t \sum_{i=1}^s b_i(\mathbf{F}(\mathbf{u}_{h,i}) - G\mathbf{p}_i), \quad (2.24)$$

where $\mathbf{u}_{h,i}$ is the intermediate velocity field at stage i , and $\mathbf{p}_i$ is the pressure field at stage i . Note that the pressure-velocity coupling is still present, therefore requiring to handle this coupling at each stage.

For projection methods, Sanderse [32, 33] presented a general methodology to use explicit and implicit RK for the INS. To preserve the order of accuracy, the pressure-velocity coupling needs to be handled at each stage, which can be done by solving a Poisson equation for the pressure field at each stage. This can be computationally expensive, yet it is necessary to ensure the accuracy and stability of the numerical solution. Therefore, using a s -stage RK method requires s projections and hence, at least, s solutions of a Poisson equation, making RK methods computationally more expensive than LMMs for the numerical integration of the momentum equation.

Moreover, RK methods can allow larger time steps compared to LMMs, which can offset their higher computational cost in some cases. Figure 2.1 provides a schematic representation of the projection method applied to a RK method, where the predictor and projection steps are represented by red and blue arrows, respectively. The thick black line represents the manifold defined by the continuity equation, where the velocity field is divergence-free.

Techniques such as the fast projection method [34–37] have been developed to avoid these projections except for the last stage, while preserving higher-than-second-order accuracy, compared to the original works from Le and Moin [30]. However, this comes with reduction in the stability regions, thus generating a trade-off between improved stability or faster algorithms.

Explicit Runge-Kutta methods With regards to explicit RK methods, Sanderse and Koren [32] analyzed their temporal accuracy, leading to the conclusion that the order of accuracy at least up to fourth-order is preserved for the velocity field without additional constraints, while the pressure field is a first order approximation, $\mathbf{p}_h(t) = \boldsymbol{\psi}(t) + \mathcal{O}(\Delta t)$. The pressure field can match the order of accuracy by satisfying additional order conditions, together with an additional Poisson equation solve.

Beyond classical RK formulations, specific classes of explicit RK have been developed

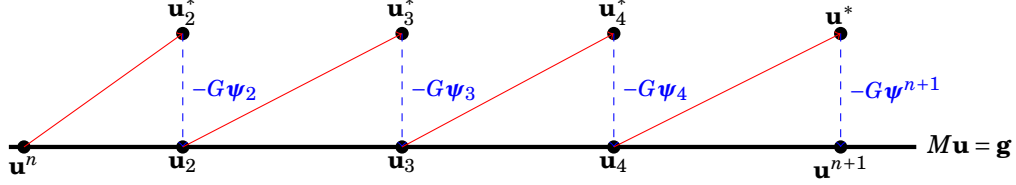


Figure 2.1: Schematic representation of the projection method applied to a RK method. The pressure-like field ψ is stated like this to denote the Lagrange multiplier nature of this variable, to distinguish it from the actual pressure field. The red arrows represent the predictor step, while the blue dashed arrows represent the projection step, where the intermediate velocity field is projected onto the divergence-free subspace by solving a Poisson equation for the pressure-like field ψ . The thick black line represents the manifold defined by the continuity equation, where the velocity field is divergence-free.

with the goal of tackling different aspects of the numerical integration of the momentum equation. For instance, to prevent spurious oscillations due to sharp gradients, Strong Stability Preserving (SSP) RK methods have been developed by preserving the convex functional properties of the spatial discretization, which ensure that the time discretization does not introduce new extrema in the solution [38]. Other formulations tackled the computational overhead of storing multiple intermediate solutions by developing low-storage RK methods [39].

While explicit RK methods are widely used for the numerical integration of the momentum equation, they suffer from stability issues, especially when applied to stiff problems such as the INS equations. Therefore, to preserve stability, explicit RK methods require small time steps, which can make them computationally expensive for long-time simulations.

Implicit Runge-Kutta methods On the other hand, implicit RK methods are a robust alternative for stiff problems where the restrictions of explicit methods can be prohibitive. Given their excellent stability properties, implicit methods such as Gauss-Legendre, Radau or Lobatto families, eliminate the Courant-Friedrichs-Lewy (CFL) restriction, thus allowing time steps not dictated by stability, but by physical time scales. However, they come at the cost of solving a large coupled nonlinear system at each time step, which is computationally expensive.

This can be alleviated using Diagonally Implicit Runge-Kutta (DIRK) methods, a subclass of implicit RK methods where $a_{ij} = 0$ for $j > i$, which allows to solve a smaller nonlinear system at each stage, thus reducing the computational cost. In the same line,

Explicit Singly DIRK (ESDIRK) methods give the additional advantage of having a constant diagonal term, $a_{ii} = \gamma$, $\forall i$, which allows to reuse the same Jacobian matrix at each stage, thus further reducing the computational cost.

Despite this stability benefits, high-order implicit RK methods frequently suffer from order reduction when applied to stiff problems such as the INS equations, which can significantly affect the accuracy of the numerical solution [40]. This is primarily due to time-dependent boundary conditions and the DAE nature of the constraints, which restricts the global convergence rate to the method's internal stage order rather than the classical order of accuracy [33, 41, 42]. Some schemes such as the Radau IIA family are stiffly accurate, which means that the solution at the last stage coincides with the solution at the new time step, which can help to mitigate order reduction [43]. In this context, Kazemi-Kamyab et al. [44] extended the analysis of order reduction to collocated frameworks, and showed that the order degradation is not inevitable by distinguishing between incremental and total pressure. This ensured a consistent pressure-velocity coupling, and therefore the high-order temporal accuracy of the method was preserved.

Implicit-Explicit Runge-Kutta methods Given the different nature of the convective and diffusive terms in the momentum equation, Implicit-Explicit (IMEX) RK methods have been developed to treat these terms differently. On the one hand, the stiff diffusive term is treated implicitly, which allows to alleviate the stability restrictions, while on the other hand, the non-linear convective term is treated explicitly, which avoids the computational cost of solving a nonlinear system at each time step. This makes IMEX RK methods a promising alternative for the numerical integration of the momentum equation, as they can provide a good balance between stability and computational cost [45]. Moreover, in the context of INS equations, specific high-order IMEX formulations have been developed to preserve the order of accuracy of the method, while handling the pressure-velocity coupling at each stage [46].

Generalizing the IMEX methods, Additive Runge-Kutta (ARK) methods provide a rigorous framework to partition the different terms in the momentum equation, and treat them with different RK methods, which can be implicit or explicit. This allows to tailor the time integration method to the specific characteristics of each term, which can further improve the stability and accuracy of the numerical solution [47]. This flexibility becomes particularly relevant in multiphysics problems, where different terms may require different treatment, or where distinct solvers are employed for different operators. Furthermore, in the context of INS equations, ARK methods have been extended to construct stiffly accurate, energy-conserving schemes that preserve kinetic

energy, a property that standard monolithic methods cannot guarantee [33].

2.3.3 Stability considerations

Integrating the momentum equation in time requires careful consideration of the stability properties of the chosen time integration method. To set the starting point, any time integration method can be described by its stability function, which characterizes the growth of errors in the numerical solution. Advancing in time an ODE can be expressed as $\mathbf{y}^{n+1} = R(z)\mathbf{y}^n$, where $R(z)$ is the stability function of the method, and z is a complex variable that depends on the time step and the eigenvalues of the Jacobian matrix of the system. For multistage methods, $R(z)$ is a rational function, while for LMMs, $R(z)$ is a matrix-valued function. The stability properties of the method can be analyzed by studying the region of the complex plane where $|R(z)| \leq 1$, which is known as the stability region. For stiff problems such as the INS equations, it is important to choose a time integration method with a large stability region that includes the negative real axis, which allows for larger time steps without sacrificing stability. Figure 2.2 shows the stability regions of different explicit RK methods, where the drawn lines represent the boundaries of the stability regions. As can be seen, explicit RK methods have limited stability regions, which can restrict the time step size for stiff problems. On the other hand, implicit methods such as BDF or Gauss-Legendre have much larger stability regions that include the entire left half of the complex plane, which allows for larger time steps without sacrificing stability [5].

As an oversimplification of the stability analysis, the time step size for explicit methods has been classically determined by means of the CFL condition, which restricts the Courant number, $C = ||\mathbf{u}||\Delta t/\Delta x$ (for uniform unidimensional grids) to be of order unity [48]. However, this condition is not sufficient to ensure stability, as it does not account for the diffusive term in the momentum equation, which states that $\Delta t \propto \Delta x^2/\nu$ to guarantee stability. These stability constraints, however, have been proved to be inaccurate [23], and more rigorous stability analyses have been developed to determine the stability limits of different time integration methods, which can be used to determine the maximum time step size for a given method and problem. These techniques depend on estimating the eigenbounds of the Jacobian matrix of the system, which can be done using techniques such as the Gershgorin circle theorem [23].

To circumvent the stability restrictions of explicit methods, implicit schemes aim for A-stability, which ensures that the numerical perturbations decay for all time step sizes given the nodes of the method are in the left half of the complex plane and

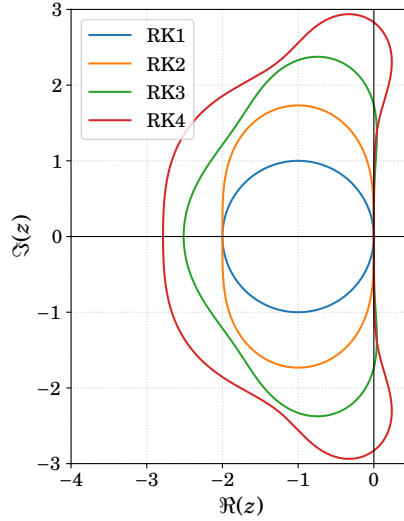


Figure 2.2: Stability regions of different time integration methods.

therefore guarantees stability for linear stiff problems regardless of the time step size. However, given the nonlinear nature of the INS, A-stability does not suffice to guarantee stability, and therefore additional properties such as L-stability, which ensures that these perturbations are rapidly damped. This property is critical for suppressing spurious numerical oscillations where the solution lacks sufficient regularity, such as in the presence of sharp gradients or discontinuities, which are common in turbulent flows [6, 49].

As A-stability and L-stability are derived from linear test problems, they do not necessarily guarantee stability for nonlinear problems such as the INS equations. This is addressed by the concept of B-stability [50], which extends the stability criteria to nonlinear problems by ensuring that the numerical solution does not grow unboundedly for any initial condition, which results in the numerical difference between two numerical solutions to decay over time, a property called contractivity [51].

All these stability criteria assume constant physical properties. However, in many practical applications, i.e., Large-Eddy Simulation (LES), the effective viscosity is neither constant nor uniform, as it depends on the local flow conditions. This can

lead to significant variations in the stability properties of the time integration method across the computational domain. This is addressed with the concept of C-stability [52], which ensures robustness for varying stiffness problems. This, applied to RK methods, requires that $b_i \geq 0$, $\forall i$, and the symplectic matrix $M_{ij} = b_i a_{ij} + b_j a_{ji} - b_i b_j$ is positive semidefinite, which ensures that the numerical solution remains stable even in the presence of non-constant-nor-uniform stiffness.

Table 2.1 summarizes the different stability concepts for time integration methods, along with their mathematical definitions and typical methods that satisfy these properties.

Table 2.1: Summary of stability concepts for time integration methods.

Concept	Mathematical definition	Typical methods
A-stability	$ R(z) \leq 1$ for $\text{Re}(z) \leq 0$	BDF, Gauss-Legendre
L-stability	$\lim_{z \rightarrow -\infty} R(z) = 0$	BDF2, SDIRK
B-stability	$\ \mathbf{w}_1^{n+1} - \mathbf{w}_2^{n+1}\ \leq \ \mathbf{w}_1^n - \mathbf{w}_2^n\ $	Radau IIA, Lobatto IIIC
C-stability	$b_i \geq 0$ and $\mathbf{x}^T M \mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^s$	Radau IIA, Lobatto IIIC

2.3.4 Adaptive timestepping strategies for the momentum equation

When setting up a numerical simulation, there is the need of selecting an appropriate time step size, which will have a significant impact on the accuracy and stability of the numerical solution. If the time step size is too large, the numerical solution may become unstable or inaccurate, while if it is too small, the simulation may take an unnecessarily long time to complete.

To address this issue, adaptive timestepping strategies have been developed, which adjust the time step size dynamically based on the local flow conditions and the desired level of accuracy. These strategies can be broadly classified into two categories: *error-based* methods, which adjust the time step size based on an estimate of the local truncation error, and *stability-based* methods, which adjust the time step size based on physical criteria such as the CFL condition or the local flow velocity.

While beneficial due to their ability to dynamically adjust the time step size based on the local flow conditions, adaptive timestepping strategies can introduce order reduction issues in LMMs, as the order conditions that lead to the coefficients of the method are derived under the assumption of a constant time step size [28]. Therefore, when the time step size is varied, the order conditions may no longer be satisfied, which can lead

to a reduction in the order of accuracy of the method. Moreover, geometric properties such as symplecticity or energy conservation can be lost when the time step size is varied, which can lead to a significant degradation of the accuracy of the numerical solution, especially for long-time simulations [53].

Error-based methods Relevant strategies for error-based adaptive timestepping are based on dynamically adjusting the time step by estimating the local truncation error T and keeping it within a user-defined tolerance ϵ . Thus, the time step size evolution is performed by a standard I-controller, which adjusts the time step size according to the following formula

$$\Delta t^{n+1} = \Delta t^n \left(\frac{\epsilon}{T} \right)^{\frac{1}{p+1}}, \quad (2.25)$$

where p is the order of accuracy of the time integration method. The most common mechanism for estimating T involves embedded RK pairs, such as the Fehlberg [54] or Dormand-Price [55] methods, which provide two approximations of the solution at each time step, one of order p and another of order $p + 1$, which can be used to estimate the local truncation error. This approach is widely used in practice due to its simplicity and efficiency, and it can be easily implemented in existing time integration schemes.

However, translating this error estimate to a new step size is challenging, as it requires careful control to avoid instabilities. Standard I-controllers can introduce rapid oscillations in the time step size, which can lead to instabilities. To mitigate this, additional signal processing techniques were used to achieve a smoother evolution of Δt [56, 57].

Specifically tailored for LES, Capuano [58] developed a minimum-dissipation adaptive timestepping strategy, specifically designed for LES, which adjusts the time step size based on the local dissipation rate of the flow, which is a key quantity in LES that characterizes the energy transfer between resolved and subgrid scales. This approach allows to dynamically adjust the time step size based on the local flow conditions, and therefore the temporal discretization error does not affect the energy cascade, which is a key aspect for the accuracy of LES simulations.

Stability-based methods Courant et al. [48] provided the starting point for stability-based adaptive timestepping by introducing the CFL condition, which approximates the eigenvalues of the convective and diffusive operators from the 1D convection-diffusion equation. While this approximation is rather inaccurate for the numerical solution of incompressible flows, it provides a first-order approximation to ensure stability in fully explicit simulations, as well as setting the starting point for more accurate

methodologies.

Building upon this theory, more rigorous stability analyses have been developed to determine the spectral radius of the Jacobian matrix of the system, which can be used to determine the maximum time step size for a given method and problem. These techniques depend on estimating the eigenbounds of the Jacobian matrix of the system, which can be done using techniques such as the Gershgorin circle theorem [23].

However, these techniques require explicitly building the Jacobian matrix, which can be computationally expensive for large systems. To address this issue, alternative approaches built upon the structure of collocated operators without explicitly building the Jacobian matrix have been developed, which provide a more efficient way to estimate the stability limits of different time integration methods [59]. In this setup, most of the computations are performed in a setup phase, while the online calculations are reduced to a sparse matrix-vector product for each operator.

These two methods have been originally developed for a second-order parameter-dependent LMM, which is widely used in practice for the numerical integration of the momentum equation. However, they can be extended to other time integration methods, such as RK methods, by analyzing the stability properties of the method and determining the maximum time step size that satisfies the stability criteria.

References

- [1] F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 8(R2):129–151, 1974.
- [2] Francis H. Harlow and J. Eddie Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids*, 8(12):2182–2189, December 1965.
- [3] C. Taylor and P. Hood. A numerical solution of the Navier-Stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73–100, January 1973.
- [4] J. A. Hopman, D. Santos, À. Alsalti-Baldellou, J. Rigola, and F. X. Trias. Quantifying the checkerboard problem to reduce numerical dissipation. *Journal of Computational Physics*, 521:113537, January 2025.

- [5] B. Sanderse. *Energy-Conserving Discretization Methods for the Incompressible Navier-Stokes Equations: Application to the Simulation of Wind-Turbine Wakes*. PhD thesis, Technische Universiteit Eindhoven, 2013.
- [6] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II*, volume 14 of *Springer Series in Computational Mathematics*. Springer, Berlin, Heidelberg, 1996.
- [7] J. Blair Perot. An Analysis of the Fractional Step Method. *Journal of Computational Physics*, 108(1):51–58, September 1993.
- [8] David Silvester, Howard Elman, David Kay, and Andrew Wathen. Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow. *Journal of Computational and Applied Mathematics*, 128(1):261–279, March 2001.
- [9] Howard Elman, Victoria E. Howle, John Shadid, Robert Shuttleworth, and Ray Tuminaro. Block Preconditioners Based on Approximate Commutators. *SIAM Journal on Scientific Computing*, 27(5):1651–1668, January 2006.
- [10] Michele Benzi and Maxim A. Olshanskii. An Augmented Lagrangian-Based Approach to the Oseen Problem. *SIAM Journal on Scientific Computing*, 28(6):2095–2113, January 2006.
- [11] Patrick E. Farrell, Lawrence Mitchell, and Florian Wechsung. An Augmented Lagrangian Preconditioner for the 3D Stationary Incompressible Navier–Stokes Equations at High Reynolds Number. *SIAM Journal on Scientific Computing*, 41(5):A3073–A3096, January 2019.
- [12] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.
- [13] R. Témam. Sur l’approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (I). *Archive for Rational Mechanics and Analysis*, 32(2):135–153, January 1969.
- [14] R. Témam. Sur l’approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (II). *Archive for Rational Mechanics and Analysis*, 33(5):377–385, January 1969.

- [15] S. V Patankar and D. B Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, October 1972.
- [16] Suhas V. Patankar. A Calculation Procedure for Two-Dimensional Elliptic Situations. *Numerical Heat Transfer*, 4(4):409–425, October 1981.
- [17] J. P. Van Doormaal and G. D. Raithby. Enhancements of the Simple Method for Predicting Incompressible Fluid Flows. *Numerical Heat Transfer*, 7(2):147–163, April 1984.
- [18] J. van Kan. A Second-Order Accurate Pressure-Correction Scheme for Viscous Incompressible Flow. *SIAM Journal on Scientific and Statistical Computing*, 7(3):870–891, July 1986.
- [19] Wang Chang, Francis Giraldo, and Blair Perot. Analysis of an Exact Fractional Step Method. *Journal of Computational Physics*, 180(1):183–199, July 2002.
- [20] L. J. P. Timmermans, P. D. Mineev, and F. N. Van De Vosse. An Approximate Projection Scheme for Incompressible Flow Using Spectral Elements. *International Journal for Numerical Methods in Fluids*, 22(7):673–688, 1996.
- [21] J. L. Guermond, P. Mineev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44):6011–6045, September 2006.
- [22] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187(1):343–368, May 2003.
- [23] F. X. Trias and O. Lehmkuhl. A Self-Adaptive Strategy for the Time Integration of Navier-Stokes Equations. *Numerical Heat Transfer, Part B: Fundamentals*, 60(2):116–134, August 2011.
- [24] John B Bell, Phillip Colella, and Harland M Glaz. A second-order projection method for the incompressible navier-stokes equations. *Journal of Computational Physics*, 85(2):257–283, December 1989.
- [25] Alessandra Nigro, Carmine De Bartolo, Francesco Bassi, and Antonio Ghidoni. Up to sixth-order accurate A-stable implicit schemes applied to the Discontinuous Galerkin discretized Navier–Stokes equations. *Journal of Computational Physics*, 276:136–162, November 2014.

- [26] Katuhiko Goda. A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *Journal of Computational Physics*, 30(1):76–95, January 1979.
- [27] J Kim and P Moin. Application of a fractional-step method to incompressible Navier-Stokes equations. *Journal of Computational Physics*, 59(2):308–323, June 1985.
- [28] Ernst Hairer, Gerhard Wanner, and Syvert P Noersett. *Solving Ordinary Differential Equations I*, volume 8 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.
- [29] Philippe R Spalart, Robert D Moser, and Michael M Rogers. Spectral Methods for the Navier-Stokes Equations with One infinite and Two Periodic Directions. *Journal of Computational Physics*, 96:297–324, 1991.
- [30] Hung Le and Parviz Moin. An improvement of fractional step methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 92(2):369–379, February 1991.
- [31] Y. Morinishi, T. S. Lund, O. V. Vasilyev, and P. Moin. Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow. *Journal of Computational Physics*, 143(1):90–124, June 1998.
- [32] B. Sanderse and B. Koren. Accuracy analysis of explicit Runge–Kutta methods applied to the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 231(8):3041–3063, April 2012.
- [33] B. Sanderse. Energy-conserving Runge–Kutta methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 233:100–131, January 2013.
- [34] Carlo De Michele, Francesco Capuano, and Gennaro Coppola. Fast-Projection Methods for the Incompressible Navier–Stokes Equations. *Fluids*, 5(4):222, December 2020.
- [35] Mokbel Karam, James C. Sutherland, and Tony Saad. Low-cost Runge-Kutta integrators for incompressible flow simulations. *Journal of Computational Physics*, 443:110518, October 2021.

- [36] Mokbel Karam and Tony Saad. On the theory of fast projection methods for high-order Navier-Stokes solvers. *Journal of Computational Physics*, 495:112557, December 2023.
- [37] Maher Eid, Mokbel Karam, and Tony Saad. On the use of fast projection methods with unsteady velocity boundary conditions. *Journal of Computational Physics*, 521:113529, January 2025.
- [38] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of Computation*, 67(221):73–85, 1998.
- [39] J.H Williamson. Low-storage Runge-Kutta schemes. *Journal of Computational Physics*, 35(1):48–56, March 1980.
- [40] Mark H. Carpenter, David Gottlieb, Saul Abarbanel, and Wai-Sun Don. The Theoretical Accuracy of Runge–Kutta Time Discretizations for the Initial Boundary Value Problem: A Study of the Boundary Error. *SIAM Journal on Scientific Computing*, 16(6):1241–1252, November 1995.
- [41] A. Ostermann and M. Roche. Runge-Kutta Methods for Partial Differential Equations and Fractional Orders of Convergence. *Mathematics of Computation*, 59(200):403–420, 1992.
- [42] Rodolfo Ruben Rosales, Benjamin Seibold, David Shirokoff, and Dong Zhou. Spatial Manifestations of Order Reduction in Runge-Kutta Methods for Initial Boundary Value Problems, December 2017.
- [43] Volker John, Gunar Matthies, and Joachim Rang. A comparison of time-discretization/linearization approaches for the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 195(44):5995–6010, September 2006.
- [44] V. Kazemi-Kamyab, A.H. Van Zuijlen, and H. Bijl. Analysis and application of high order implicit Runge–Kutta schemes to collocated finite volume discretization of the incompressible Navier–Stokes equations. *Computers & Fluids*, 108:107–115, February 2015.
- [45] Uri M. Ascher, Steven J. Ruuth, and Raymond J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2):151–167, November 1997.

- [46] S. Boscarino and L. Pareschi. On the asymptotic properties of IMEX Runge–Kutta schemes for hyperbolic balance laws. *Journal of Computational and Applied Mathematics*, 316:60–73, May 2017.
- [47] Christopher A. Kennedy and Mark H. Carpenter. Additive Runge–Kutta schemes for convection–diffusion–reaction equations. *Applied Numerical Mathematics*, 44(1):139–181, January 2003.
- [48] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100(1):32–74, December 1928.
- [49] Rolf Rannacher. Finite element solution of diffusion problems with irregular data. *Numerische Mathematik*, 43(2):309–327, June 1984.
- [50] J. C. Butcher. A stability property of implicit Runge-Kutta methods. *BIT Numerical Mathematics*, 15(4):358–361, December 1975.
- [51] J F B M Kraaijevanger. Contractivity of Runge-Kutta methods. *BIT*, 31:482–528, 1991.
- [52] Kevin Burrage and J. C. Butcher. Non-linear stability of a general class of differential equation methods. *BIT Numerical Mathematics*, 20(2):185–203, June 1980.
- [53] Ernst Hairer, Gerhard Wanner, and Christian Lubich. Structure-Preserving Implementation. In Ernst Hairer, Gerhard Wanner, and Christian Lubich, editors, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, pages 303–336. Springer, Berlin, Heidelberg, 2006.
- [54] E. Fehlberg. Klassische Runge-Kutta-Formeln fünfter und siebenter Ordnung mit Schrittweiten-Kontrolle. *Computing*, 4(2):93–106, June 1969.
- [55] J.R. Dormand and P.J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, March 1980.
- [56] G. Noventa, F. Massa, S. Rebay, F. Bassi, and A. Ghidoni. Robustness and efficiency of an implicit time-adaptive discontinuous Galerkin solver for unsteady flows. *Computers and Fluids*, 204, 2020.

- [57] Antonio Ghidoni, Francesco Carlo Massa, Gianmaria Noventa, and Stefano Rebay. Assessment of an adaptive time integration strategy for a high-order discretization of the unsteady RANS equations. *International Journal for Numerical Methods in Fluids*, 94(12):1923–1963, 2022.
- [58] Francesco Capuano, Benjamin Sanderse, Enrico Maria De Angelis, and Gennaro Coppola. A minimum-dissipation time-integration strategy for Large-Eddy Simulation of Incompressible Turbulent Flows. In *AIMETA 2017 - XXIII Congresso - Associazione Italiana Di Meccanica Teorica e Applicata Salerno*, 2017.
- [59] F. X. Trias, X. Álvarez-Farré, A. Alsalti-Baldellou, A. Gorobets, and A. Oliva. An efficient eigenvalue bounding method: CFL condition revisited. *Computer Physics Communications*, 305, December 2024.

Self-adaptive integration of incompressible flows

“It is not the strongest of the species that survives, nor the most intelligent; it is the one most adaptable to change.”

– Charles Darwin

Integrating explicitly the incompressible Navier-Stokes equations in time is a challenging task, as the stability constraints given by the convective and diffusive terms are generally very restrictive. This problem is especially true for scale-resolving simulations, which require very fine spatial discretization that, according to the Courant-Friedrichs-Lewy (CFL) condition [1], leads to very small timesteps due to the stiffness of the diffusive term.

While this problem cannot be avoided without moving to implicit (or implicit-explicit) integration, these problems can be alleviated by maximizing the timestep by performing a stability analysis of the system at every timestep [2]. This approach, called self-adaptive time-integration, has been shown to significantly improve the efficiency of explicit integration for incompressible flows, as it allows to use the largest stable timestep at every iteration. While this approach has been successfully applied to multistep integration, it has not been fully explored for multistage integration and staggered grids. Therefore, in this chapter, an extension of self-adaptive time-integration methods to multistage integration and staggered grids is presented, which allows to efficiently compute the eigenbounds of the convective and diffusive operators for these cases. Finally, the self-adaptive integration method is extended to Reduced-

Order Models (ROMs), which are commonly used for control, design, optimization, or uncertainty quantification tasks.

3.1 Self-adaptive time-integration methods

As introduced in Section 2.3.4, stability-aware self-adaptive integration methods are based on the spectral analysis of the convective and diffusive operators, which allows to determine the largest stable timestep at every iteration. The key idea is to compute the eigenbounds of the convective and diffusive operators, which are given by the spectral radius of these operators. However, computing the spectral radius of these operators can be computationally expensive, especially for large-scale simulations. Therefore, deriving efficient methods to compute the spectral radius of these operators is required.

The seminal work for self-adaptive integration for incompressible flows was developed by Courant et al. [1], which introduced the CFL condition, which provides a stability constraint for explicit integration methods. By performing a simplified spectral analysis of the one-dimensional convective and diffusive operators, they derived the well-known CFL condition,

$$\Delta t \leq \min \left(C_C \frac{\Delta x}{|u|}, C_D \frac{\Delta x^2}{\nu} \right), \quad (3.1)$$

where $C_D, C_C \leq 1$ have to be prescribed *a priori*, as well as a characteristic length of the control volume Δx . While very extensively used in most commercial solvers, this condition is based on a simplified analysis and does not take into account the full spectral properties of the operators, which can lead to overly restrictive timesteps.

To overcome these limitations, Trias and Lehmkuhl [2] proposed a self-adaptive time-integration method, EigenCD, based on the spectral analysis of the convective and diffusive operators, which allows to compute the largest stable timestep at every iteration. EigenCD is based on the Gershgorin circle theorem, which provides an efficient way to compute the eigenbounds of the convective and diffusive operators. This method, which relies in the explicit construction of the convective and diffusive operators, has been shown to significantly improve the efficiency of explicit integration for incompressible flows, as it allows to use the largest stable timestep at every iteration. Moreover, EigenCD does not rely on any specific structure of the operators, which makes it applicable to a wide range of discretizations as well as to both collocated and staggered grids. As the explicit construction of the operators can be computationally expensive, Trias et al. [3] proposed an efficient method for computing the eigenbounds of the convective and

diffusive operators based on the structure of the discrete collocated operators, which allows to compute the eigenbounds without explicitly constructing the operators. This method, called AlgEigCD, has been shown to significantly reduce the computational cost of computing the eigenbounds, while still providing accurate estimates of the largest stable timestep. However, given its nature, AlgEigCD can only be used for collocated discretizations, as it relies on the structure of the convective and diffusive operators in these grids. Therefore, in this chapter, an extension of AlgEigCD to staggered grids is presented, which allows to efficiently compute the eigenbounds of the convective and diffusive operators for staggered grids. This extension is based on a careful analysis of the structure of the operators for staggered grids, and it leverages the properties of the discrete operators to obtain an efficient algorithm for computing the eigenbounds.

3.1.1 Linear stability analysis of the incompressible Navier-Stokes equations

To integrate Eqs.(2.1)-(2.2) explicitly in time, the stability of the system needs to be analyzed. Independently of the family of time integration schemes considered, the methodology can be contracted to

$$\mathbf{u}_h^{n+1} \approx R(\Delta t \tilde{F}) \mathbf{u}_h^n, \quad (3.2)$$

where $\tilde{F} = \Omega^{-1}(D - C(\mathbf{u}))$ is the operator arising from linearizing $\mathbf{F}(\mathbf{u}_h(t))$ as $\tilde{F}(\mathbf{u}_h(t))\mathbf{u}_h(t)$ via Picard linearization, and $R(z)$ is the stability function of the given linearized scheme. Note that there is an approximation symbol instead of an equal symbol as the $n + 1$ value requires projecting the solution back into the incompressible space, which relies on the solution of the Poisson equation. However, the projection step in a projection method is implicit and does not affect the stability region in a zero-divergence framework.

Following linear stability analysis theory for differential equations [4], the linear method is stable if

$$\|R(\Delta t \tilde{F})\|_2 \leq 1. \quad (3.3)$$

As explained in [4], this requirement can be simplified to

$$|R(\Delta t \lambda_F)| \leq 1, \quad (3.4)$$

where λ_F is the largest (in magnitude) eigenvalue of $\tilde{F}$. Therefore, the maximum stable timestep is such that for $z_{max} = \Delta t \|\lambda_F\|_2$, it fulfills the equation $\|R(z_{max})\|_2 = 1$. By doing so, the eigenvalue $\Delta t \lambda_F$ is located at the boundary of the stability region, as

depicted in Figure 3.1. Hence, the self-adaptive timestep can be computed as

$$\Delta t = \frac{z_{max}}{\|\lambda_F\|_2}, \quad (3.5)$$

where z_{max} is obtained by solving the equation $\|R(z)\|_2 - 1 = 0$ using a bisection method along the axis defined by the origin of coordinates ($z = 0 + 0i$) and λ_F every timestep. Therefore, the goal of any adaptive timestepping method is to calculate the eigenbounds of $\tilde{F}$ so that the maximum stable timestep is computed using Eq. (3.5).

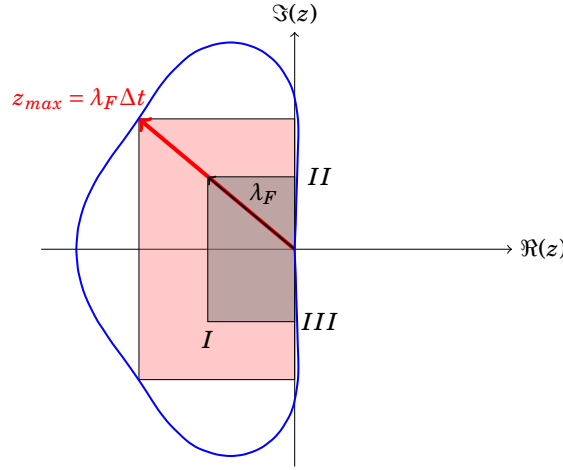


Figure 3.1: Schematic illustration of Bendixson's rectangle (gray) for the eigenvalues of $\tilde{F}$. *I*: $\Re(\lambda_F)$, *II*: $\Im(\lambda_F)$, *III*: $-\Im(\lambda_F)$. The blue curve represents the boundary of the stability region for a third-order Runge-Kutta scheme.

The methodology for computing the eigenbounds of $\tilde{F}$ assumes the discretization is symmetry-preserving [5, 6]. The reader is referred to [3, Appendix B] for a detailed discussion on the generalization to non-symmetry-preserving discretizations, which relies on splitting the convective term into a skew-symmetric part and a symmetric part, and then applying the same methodology to both parts separately. This makes the method applicable to flux limiting and upwinding schemes, which are very well spread in the whole Computational Fluid Dynamics (CFD) community. However, the focus of this chapter is on symmetry-preserving discretizations, as they are widely used in scale-resolving simulations, and they have very good stability properties, which makes them a good candidate for self-adaptive time-integration methods.

Symmetry-preserving discretizations are a family of schemes that preserve the symmetry properties of the continuous operators at the discrete level [5]. This is usually achieved by ensuring that the diffusive operator is discretized as a symmetric matrix and corresponds to the composition of the gradient and divergence operators, resulting in a negative semi-definite matrix, the convective operator corresponds to a skew-symmetric matrix, and the discrete gradient and divergence operators are dual to each other, i.e., $-\Omega M^T = G$. In physical terms, these properties ensure that the only dissipation of kinetic energy comes from the diffusive term, while the convective term only redistributes the kinetic energy without creating or destroying it. On the other hand, the div-grad compatibility ensures that the pressure-gradient work and the velocity divergence cancel each other, preventing any additional energy source or sink.

Given the diffusive operator is a symmetric negative semi-definite matrix, its eigenvalues are real and non-positive, and therefore will lie on the negative real axis. On the other hand, given the convective operator is a skew-symmetric matrix, its eigenvalues are purely imaginary, and therefore will lie on the imaginary axis. Therefore, the eigenvalues of $\tilde{F}$ will lie in the left half of the complex plane, as their real part will be non-positive. Given the subadditivity of the spectral radius, i.e., $\rho(A + B) \leq \rho(A) + \rho(B)$, the eigenbound of $\tilde{F}$ lies, according to Bendixson's inequality [7], in the rectangle defined by the eigenbounds of the convective and diffusive operators, similarly to the one depicted in Figure 3.1, which is denominated Bendixson's rectangle.

The stability region is determined by the stability function of the time integration scheme (Figure 2.2). For multistep methods, their stability functions correspond to matrices. For instance, a second-order Linear Multistep Method (LMM) ($\kappa 1L2$) [2, 5] defined as $(\kappa + 1/2)\mathbf{y}^{n+1} - 2\kappa\mathbf{y}^n + (\kappa - 1/2)\mathbf{y}^{n-1} = \Delta t[(1 + \kappa)f(\mathbf{y}^n) - \kappa f(\mathbf{y}^{n-1})]$ has a stability function given by

$$\begin{pmatrix} \mathbf{u}_h^{n+1} \\ \mathbf{u}_h^n \end{pmatrix} = \begin{pmatrix} \frac{2\kappa + z(1+\kappa)}{\kappa + 1/2} & -\frac{\kappa - 1/2 + \kappa z}{\kappa + 1/2} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u}_h^n \\ \mathbf{u}_h^{n-1} \end{pmatrix}, \quad (3.6)$$

where $z = \Delta t \lambda_F$. Therefore, evaluating $|R(z)|$ corresponds to solving an eigenvalue problem for the matrix defined by the stability function. This can be easily generalized to the κ -dependent family of multistep methods, which include the second-order Adams-Bashforth (AB) ($\kappa = 0.5$) and the leapfrog ($\kappa = 0$) methods as particular cases. This has been depicted in Figure 3.2, where the stability regions for some multistep methods from the $\kappa 1L2$ family are depicted, together with the stability regions for some Runge-Kutta (RK) methods up to $s = 4$ for comparison purposes. As can be seen in Figure 3.2, the stability regions for the multistep methods are generally smaller than those for the RK methods, which means that they will require smaller timesteps to ensure stability.

However, the computational cost-per-step for multistep methods is generally lower than that for RK methods, which can make them more efficient for certain problems. Therefore, the choice of the time integration scheme should be based on a careful analysis of the stability properties of the system and the computational cost of the scheme.

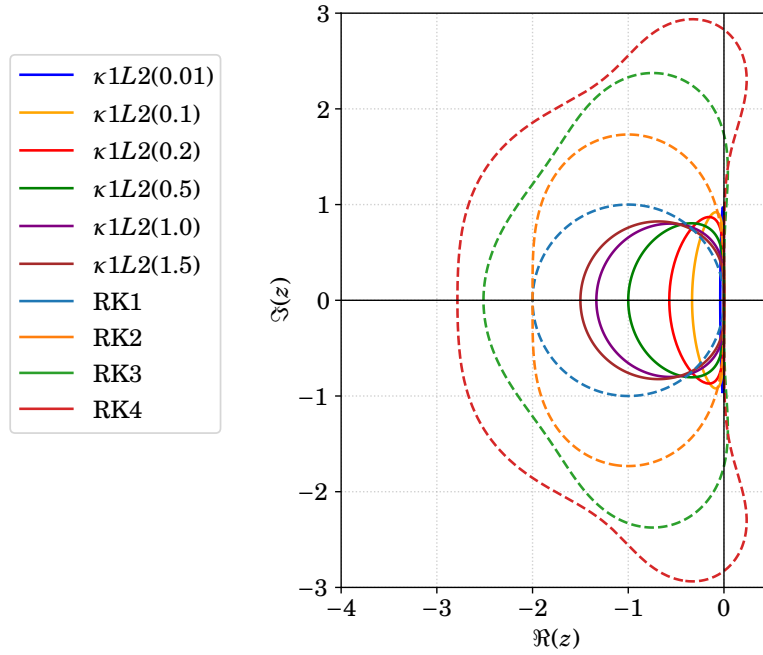


Figure 3.2: Stability regions for some multistep methods from the $\kappa 1L2$ family. Stability regions for RK up to $s = 4$ are depicted with dashed lines for comparison purposes.

On the other hand, the stability function of a general RK method is given by a rational function. Following the notation from the Butcher's tableau (Eq.(2.22)), the stability function for an explicit RK method can be written as

$$R(z) = 1 + z\mathbf{b}^T(I - zA)^{-1}\mathbf{e}_s, \quad (3.7)$$

where $\mathbf{e}_s$ is a vector of ones of length s , where s is the number of stages of the method. Therefore, evaluating $|R(z)|$ corresponds to solving a linear system of equations for the matrix defined by the stability function. However, for explicit RK methods, the matrix

$I - zA$ is lower triangular with ones in the diagonal, which means that the linear system can be solved efficiently using forward substitution, which yields a polynomial function.

Moreover, for explicit RK methods such that the number of stages is equal to the order of the method, up to $s = 4$, the stability function can be written as a polynomial of degree s as follows

$$R(z) = 1 + \sum_{j=1}^s \frac{z^j}{j!}. \quad (3.8)$$

Therefore, evaluating $|R(z)|$ corresponds to evaluating a polynomial of degree s , which makes the evaluation of the stability function for explicit RK methods very efficient.

3.1.2 Existing methods for computing the eigenbounds of the convective and diffusive operators

According to the literature, the most commonly used method for computing the eigenbounds of the convective and diffusive operators is the Gershgorin circle theorem, which reads as follows.

Theorem 1 (Gershgorin circle theorem, [8]). *Let $A \in \mathbb{R}^{n \times n}$ be a square matrix. Then, every eigenvalue λ of A lies within at least one of the Gershgorin disks defined by*

$$D_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq R_i\}, \quad (3.9)$$

where a_{ii} is the i -th diagonal element of A , and $R_i = \sum_{j \neq i} |a_{ij}|$ is the sum of the absolute values of the non-diagonal elements in the i -th row. Therefore, the spectral radius of A can be estimated as

$$\rho(A) \leq \max_{1 \leq i \leq n} (|a_{ii}| + R_i). \quad (3.10)$$

This method is based on the fact that the eigenvalues of a matrix are located within the union of the Gershgorin disks, which are defined by the diagonal elements of the matrix and the sum of the absolute values of the non-diagonal elements in each row. Therefore, by computing the diagonal elements and the sum of the absolute values of the non-diagonal elements for each row, an estimate of the spectral radius can be obtained. This method is very efficient, as it only requires computing the diagonal elements and the sum of the absolute values of the non-diagonal elements for each row, which can be done in linear time with respect to the size of the matrix. This is depicted schematically in Figure 3.3, where the eigenvalues of the matrix A are located within the union of the

Gershgorin disks. The Bendixson rectangle is also depicted, which gives some insight on the method's accuracy.

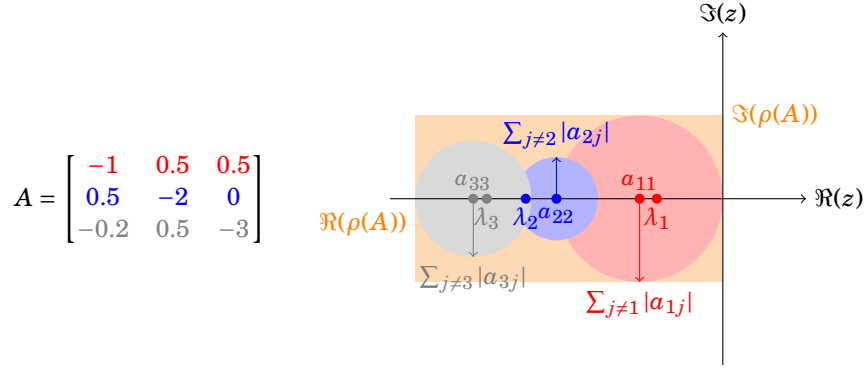


Figure 3.3: Schematic illustration of the Gershgorin circle theorem. The eigenvalues of the matrix A are located within the union of the Gershgorin disks. λ_1, λ_2 , and λ_3 are the eigenvalues of A , while a_{11}, a_{22} , and a_{33} are the diagonal elements of A . The red, blue, and gray circles represent the Gershgorin disks corresponding to the first, second, and third rows of A , respectively.

EigenCD [2] built on top of the Gershgorin circle theorem to compute the eigenbounds of the convective and diffusive operators, which allows to compute the largest stable timestep at every iteration. This method relies on the explicit construction of the convective and diffusive operators, to which Theorem 1 is applied to compute the eigenbounds. This core idea makes it applicable to a wide range of discretizations as well as to both collocated and staggered grids, as it does not rely on any structure of the operators. However, the explicit construction of the convective and diffusive operators can be computationally expensive, especially for large-scale simulations, as it requires explicitly constructing the matrices corresponding to these operators, which can be very large and sparse.

To circumvent this issue, Trias et al. [3] proposed an efficient method for computing the eigenbounds of the convective and diffusive operators based on the properties of the discrete operators, which allows to compute the eigenbounds without explicitly constructing the operators. This method, called AlgEigCD, relied on the offline construction of a matrix which purely depends on geometrical information rather than on the flow field. Afterwards, in an online stage, this pre-computed matrix together with the flow field information allows applying Theorem 1 to compute the eigenbounds of the convective and diffusive operators without explicitly constructing these operators. This

method has been shown to significantly reduce the computational cost of computing the eigenbounds, while still providing accurate estimates of the largest stable timestep. However, AlgEigCD is only applicable to collocated grids, as it relies on the structure of the convective and diffusive operators for collocated grids. Therefore, in this chapter, an extension of AlgEigCD to staggered grids is presented, which allows to efficiently compute the eigenbounds of the convective and diffusive operators for staggered grids. This extension is based on a careful analysis of the structure of the operators for staggered grids, and it leverages the properties of the discrete operators to obtain an efficient algorithm for computing the eigenbounds.

3.1.3 Extending AlgEigCD to staggered grids

The derivation of AlgEigCD in [3] relies on the structure of the convective and diffusive operators for a collocated grid. However, for a staggered grid, the structure of these operators is different, which prevents the direct application of this method. Therefore, in this section, an extension of AlgEigCD to staggered grids is presented, which allows to efficiently compute the eigenbounds of the convective and diffusive operators for staggered grids. This extension is based on a careful analysis of the structure of the operators for staggered grids, and it leverages the properties of the discrete operators to obtain an efficient algorithm for computing the eigenbounds, which preserves the core idea of the original AlgEigCD method, i.e., the offline construction of a matrix which purely depends on geometrical information rather than on the flow field, and an online stage where this pre-computed matrix together with the flow field information allows applying Theorem 1 to compute the eigenbounds of the convective and diffusive operators without explicitly constructing these operators.

Before delving into the details of the extension of AlgEigCD into staggered grids, it is important to settle the base theorem used for AlgEigCD to simplify the estimates of the eigenbounds of operators to simple algebraic operations. To preserve the idea from [3], the same property is used

Theorem 2 (Theorem 1 in [3]). *Let $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{m \times n}$ be rectangular matrices and $m \geq n$. Then, the square matrices $AB \in \mathbb{R}^{n \times n}$ and $A^T B^T \in \mathbb{R}^{m \times m}$ have the same eigenvalues except for the zero-valued ones.*

In terms of the diffusive operator, a second-order symmetry-preserving operator can be built by concatenating two first-order operators [5, 9], which leads to

$$D = KAS, \tag{3.11}$$

where $K \in \mathbb{R}^{N_v \times N_F}$ is a differencing matrix from staggered faces (N_F) to staggered cells (N_v), $\Lambda \in \mathbb{R}^{N_F \times N_F}$ corresponds to a diagonal matrix containing the diffusivities at the staggered faces, and $S \in \mathbb{R}^{N_F \times N_v}$ is a differencing matrix from staggered cells to staggered faces. Thus, from [3] it follows that a family of α -dependent matrices can be built using Theorem 2 such that their eigenbound is the same as the one from the diffusive term (Eq. (3.11)). Hence, applying Theorem 2 with $A = K\Lambda^\alpha$ and $B = \Lambda^{1-\alpha}S$,

$$\Lambda^\alpha K^T S^T \Lambda^{1-\alpha}. \quad (3.12)$$

Therefore, the Gershgorin circle theorem can be applied straightaway to Eq. (3.12) as it will have the same eigenbound as the diffusive term, which leads to

$$\rho(D) \leq \max\{\text{diag}(\Lambda^\alpha) \circ |K^T S^T| \text{diag}(\Lambda^{1-\alpha})\}, \quad (3.13)$$

where $\circ$ denotes the Hadamard product.

The benefit of using this method instead of the Gershgorin circle theorem straightaway to D is that $|K^T S^T|$ can be build in a pre-process (offline) stage. Thus, every timestep, assuming Λ varies as in a LES framework, only a sparse matrix-vector product (SpMV) and possibly (if $\alpha \neq 0$) an elementwise vector-vector product (axty) should be performed, opposite to needing to rebuild the matrix every step, as for EigenCD [2].

With regards to the convective term, its construction in a staggered framework is defined generally as

$$C(\mathbf{u})\mathbf{u} = K((\Pi\mathbf{u} + \mathbf{y}_I) \circ (A\mathbf{u} + \mathbf{y}_A)), \quad (3.14)$$

where $\Pi \in \mathbb{R}^{N_F \times N_v}$ interpolates the fluxes to the staggered faces, and $A \in \mathbb{R}^{N_F \times N_v}$ interpolates the velocities to the staggered facs. Moreover, $\mathbf{y}_I$ and $\mathbf{y}_A$ apply the boundary conditions to the interpolated fields to account for them. As $\mathbf{y}_A$ does not have an associated volume, it can be dropped for the stability analysis. However, the interpolated boundary conditions vector $\mathbf{y}_I$ is associated with faces and thus it cannot be dropped. Therefore, the convective operator can be rewritten as

$$C(\mathbf{u}) = KFA, \quad (3.15)$$

where $F = \text{diag}(\Pi\mathbf{u} + \mathbf{y}_I) \in \mathbb{R}^{N_F \times N_F}$ is a diagonal matrix containing the fluxes at the staggered faces. Trias et al. [3] showed that using Theorem 2 straight onto this definition does not provide any benefits, as more entries are considered in the estimate. Instead, they proposed a set of transformations to obtain a more efficient estimate. This method

relies on the following theorems.

Theorem 3 (Perron-Frobenius (PF) theorem, [10, 11]). *Given an irreducible matrix $A \in \mathbb{R}^{n \times n}$ such that $[A]_{ij} \geq 0$, $\forall i, j$, it has a unique real and positive largest (in magnitude) eigenvalue $r \in \mathbb{R}^+$ and corresponding eigenvector with only positive entries, i.e.,*

$$A\mathbf{v} = r\mathbf{v} \Rightarrow |\lambda| < r \text{ and } v_i > 0, \forall i \in \{1, \dots, n\}, \quad (3.16)$$

where λ denotes any eigenvalue of A except r , and r is the Perron-Frobenius eigenvalue.

Theorem 4 (Wielandt's theorem, [12]). *Given a matrix A satisfying Theorem 3, and a matrix $B \in \mathbb{R}^{n \times n}$ such that $|[B]_{ij}| \leq [A]_{ij}$, $\forall i, j$. Then, any λ^B satisfies $|\lambda^B| \leq r$, where r is the PF eigenvalue of A .*

Theorem 5 (Nikiforov's second Lemma, [13]). *Let $A \in \mathbb{R}^{n \times n}$ be an irreducible, i.e. it is not similar via a permutation of a block upper triangular matrix, and a non-negative symmetric matrix $R \in \mathbb{R}^{n \times n}$ such that $[R]_{ii} = \sum_{j=1}^n [A]_{ij}$ and $[R]_{ij} = 0$ if $i \neq j$. Then,*

$$\rho\left(R + \frac{1}{q-1}A\right) \geq \frac{q}{q-1}\rho(A), \quad (3.17)$$

where the equality holds iff $[R]_{ii} = [R]_{jj}$ for any combination of i and j .

Hence, Theorem 4 may be used to relate the eigenbound of the convective operator and such from

$$A^C \equiv |K||F||A|, \quad (3.18)$$

which corresponds to a symmetric positive matrix. Thus, by construction of A^C , its off-diagonal terms correspond to $2|C|$ as follows, $A^{C,off} = A^C - \text{diag}(\text{diag}(A^C)) = 2|C|$, while the rowsums of $A^{C,off}$ are equal to the diagonal terms of A^C given the skew-symmetry of C . Thus, $A^{C,off}$ satisfies Theorem 3, which implies that it has a unique largest eigenvalue. Hence, applying Theorem 4, this can relate the eigenvalues of $|C|$ and those from C . Thus, Theorem 5 can be applied to $A^{C,off}$ and $\text{diag}(\text{diag}(A^C))$ such that, with $q = 2$, the left-hand side becomes $\rho(A^C)$. This leads to

$$\rho(A^C) \stackrel{\text{Thm 5}}{\geq} 2\rho(A^{C,off}) = 4\rho(|C|) \stackrel{\text{Thm 4}}{\geq} 4\rho(C), \quad (3.19)$$

as in [3], yet with the matrices having a different definition given their staggered nature. Now, applying Eq.(3.19) to the convective term, together with Theorem 2 and Gershgorin

circle theorem, this leads to

$$\rho(C) \leq \frac{1}{4} \max\{\text{diag}(|F|^\alpha) \circ \|K\|^T |A|^T \text{diag}(|F|^{1-\alpha})\}. \quad (3.20)$$

However, as presented in [3], the relevant operators are scaled by Ω^{-1} . Therefore, the eigenbounds of both convective and diffusive operators read

$$\rho(\Omega^{-1}D) \leq \max\{\text{diag}(\Lambda^\alpha) \circ |K^T \Omega^{-1} S^T| \text{diag}(\Lambda^{1-\alpha})\}, \quad (3.21)$$

$$\rho(\Omega^{-1}C) \leq \frac{1}{2} \max\{\text{diag}(|F|^\alpha) \circ \|K\|^T \Omega^{-1} |A|^T \text{diag}(|F|^{1-\alpha})\}. \quad (3.22)$$

The value of α can be optimized to minimize the eigenbounds, which can be done efficiently by performing a line search in an interval defined by the user. In order to provide a general estimate, a test with a two-dimensional shear-layer roll-up in a square domain of side 2π has been performed [14], where the eigenbounds of the convective and diffusive operators have been computed for different values of α and for different grid sizes. The results are depicted in Figure 3.4, where it can be seen that the optimal value of α is around 0, which corresponds to the original definition of AlgEigCD for collocated grids.

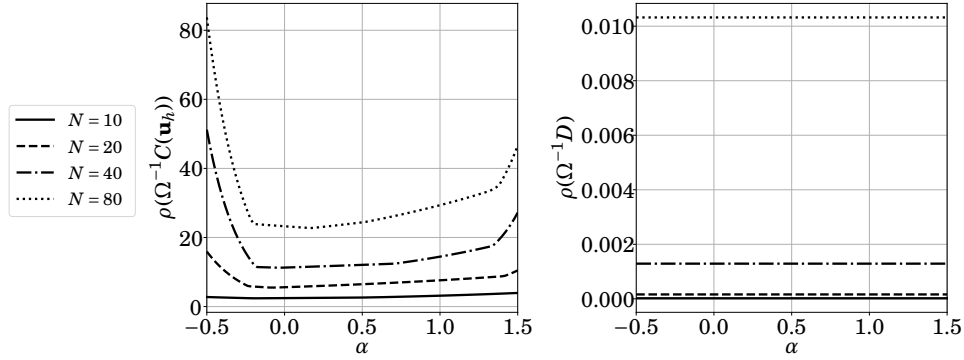


Figure 3.4: Eigenbounds of the convective (left) and diffusive (right) operators for different values of α . The optimal value of α is around 0, which corresponds to the original definition of AlgEigCD for collocated grids.

Therefore, Eqs.(3.21)-(3.22) can be rewritten as

$$\rho(\Omega^{-1}D) \leq \max\{|K^T \Omega^{-1} S^T| \text{diag}(\Lambda)\}, \quad (3.23)$$

$$\rho(\Omega^{-1}C) \leq \frac{1}{2} \max\{|K|^T \Omega^{-1} |A|^T| \text{diag}(|F|)\}. \quad (3.24)$$

3.2 Self-adaptive integration for ROMs

Scale-resolving simulations, i.e., Direct Numerical Simulation (DNS) and Large-Eddy Simulation (LES), are generally prohibitively expensive for many practical applications, especially for tasks that require many simulations such as control, design, optimization, or uncertainty quantification [15]. ROMs try to alleviate the costs by projecting the high-dimensional system onto a low-dimensional subspace, which is spanned by a few basis functions [16–18]. The methodology for constructing the basis depends on the characteristics of the problem. For instance, for a parameter-based problem, the basis is built based on a greedy Reduced Basis algorithm [19,20]. For a time-dependent problem, the basis is built based on the Proper Orthogonal Decomposition (POD) of a matrix of snapshots of the solution [21–23].

While stability-aware adaptive step control has been thoroughly studied for the high-dimensional system (the Full-Order Model (FOM)), it has not been fully explored for ROMs. While several contributions have addressed stabilization techniques for ROMs based on spectral considerations [24], the use of adaptive timestepping for ROMs has been limited to extending the classical CFL condition, yet no spectral analysis has been performed to determine the stability limits of the ROMs. In this section, a novel approach for self-adaptive integration of ROMs based on a spectral analysis of the ROM's Jacobian matrix is presented. It is shown that this approach can significantly improve the stability, while preserving their accuracy and also reducing computational costs.

Even though lots of work has been done on the construction of ROMs applied to CFD: closure models [25, 26], structure-preserving operators [27, 28], hyper-reduction techniques [29, 30], or stabilization [31–33]; little effort has been put to finetune the timestep to maximize the performance of the ROM. Sanderse [15] indicated the use of the Gershgorin circle theorem as a possible way to determine the stability limits of ROMs, yet no tests were performed to validate this approach. In this section, the vision from this paper is realized in an intrusive context, i.e., the ROM is constructed by projecting the governing equations onto a low-dimensional subspace, by implementing a

stability-based adaptive timestepping. This aligns with efforts in other fields, such as structural mechanics, where extensive analysis of the stability conditions for projection-based ROMs has been performed [34].

Therefore, in this section, the goal is to bridge the gap between self-adaptive timestepping techniques tailored for scale-resolving CFD simulations [2, 3] and ROMs, as according to Bach et al. [34], it should be possible to generally obtain larger timesteps compared to the FOM. This is supported by recent developments across various ROM paradigms, which have demonstrated the ability to maintain stability and qualitative physics at temporal resolutions coarser than the FOM [35–38]. Thus, the potential benefit of this approach is twofold: (i) it can significantly reduce the computational cost of ROMs by allowing for larger timesteps, and (ii) it can improve the stability and accuracy of ROMs by ensuring that the timestep is chosen based on the spectral properties of the ROM's Jacobian matrix. This extension, however, should not break the leitmotif of reduced-order modeling [39], which is to avoid online FOM-size operations, which break the efficiency of the ROM.

3.2.1 Energy-conserving POD-Galerkin ROMs

The ROM considered in this section follows the ODE-based projection approach [40], where the PDE is first discretized in space, and afterwards is projected to the subspace obtained with POD [15]. This approach, called "discretize first, then project", is depicted in Figure 3.5 with thick lines. The alternative approach, "project first, then discretize", is depicted with fine lines. Depicted in dashed lines is the approach taken to calculate the snapshots, which are used to build the POD basis.

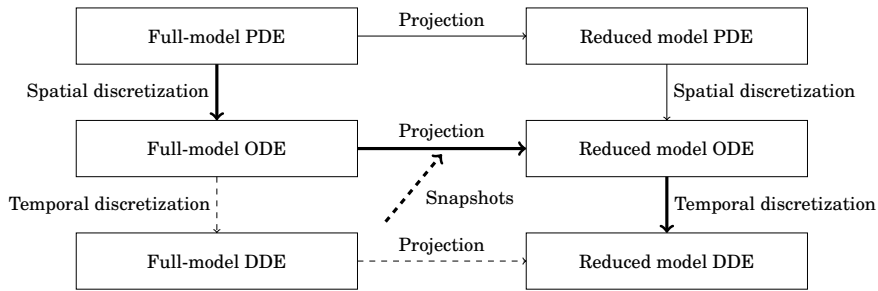


Figure 3.5: This section's approach for energy-conserving POD-Galerkin ROMs. Adapted from [15]. The thick lines indicate the steps that are performed with this approach, while the fine lines indicate the steps that are not performed.

Let the velocity vector be defined as $\mathbf{u} \in \mathbb{R}^{N_v}$. Its POD-Galerkin approximation is obtained by the Ansatz

$$\mathbf{u}_c(t) \approx \mathbf{u}_r(t) = \Phi \mathbf{a}(t), \quad (3.25)$$

where $\Phi \in \mathbb{R}^{N_v \times M}$ is the projection matrix, and $\mathbf{a} \in \mathbb{R}^M$ is the coefficient vector, where M is the number of modes of the ROM. The projection matrix is obtained from the so-called snapshot matrix X , which contains K snapshots of the velocity field, i.e., $X = [\mathbf{u}_c(t_1), \mathbf{u}_c(t_2), \dots, \mathbf{u}_c(t_K)] \in \mathbb{R}^{N_v \times K}$, which are obtained from a numerical solution of Eqs.(2.1)-(2.2) (also known as the FOM).

Construction of the POD basis The POD basis arises from the solution of the eigenvalue problem [41]

$$R \Phi_j = \lambda_j \Phi_j, \quad (3.26)$$

where R is a correlation matrix between the velocity field and Φ_j is the j -th column of the POD basis. Namely, by definition,

$$R = \frac{1}{T} \int_0^T \mathbf{u}(t) \mathbf{u}(t)^T dt, \quad (3.27)$$

where T is the total time of sampling. As Δt is non-constant when performing self-adaptive timestepping in the FOM, the integral form of Eq.(3.27) does not reduce to the classical $R = X X^T$ correlation matrix. Instead, the integral yields a weighted correlation matrix, where the weights are given by the time steps used to sample the snapshots. Introducing $\Delta \in \mathbb{R}^{K \times K}$ as a diagonal matrix containing $\Delta t_k/T$ in its k -th diagonal entry, the correlation matrix can be written as

$$R = X \Delta X^T = \tilde{X} \tilde{X}^T, \quad (3.28)$$

where $\tilde{X} = X \Delta^{1/2}$ is the weighted snapshot matrix. Therefore, the POD basis can be obtained from the Singular Value Decomposition (SVD) of $\tilde{X}$, which is computationally more efficient than solving the eigenvalue problem for R .

To consider the weighted orthonormality condition assumed by Sanderse [15], the weighted and unweighted problems are related by $\Phi = \Omega^{-1/2} \hat{\Phi}$, where $\hat{\Phi}$ is the unweighted POD basis obtained from the SVD of $\tilde{X}$. Therefore the snapshot matrix is also related in such a way that $\hat{X} = \Omega^{1/2} \tilde{X} = \Omega^{1/2} X \Delta^{1/2}$. Hence $\hat{\Phi}$ is a product of the SVD of $\hat{X}$, i.e.,

$$\hat{X} = \hat{\Phi} \Sigma \Psi^*. \quad (3.29)$$

Hence, the columns of $\hat{\Phi}$, denoted by $\hat{\Phi}_j$ correspond to the eigenvectors of $\hat{X}\hat{X}^T$. Thus, the procedure to obtain Φ is as follows: (i) obtain snapshots of the velocity field to build X and as a consequence $\hat{X}$, (ii) perform the SVD of $\hat{X}$ to obtain $\hat{\Phi}$ and as a consequence Φ , (iii) truncate Φ to the given number of modes M . For more details on the construction of the POD basis, see [15].

Projection of the FOM Given Φ and the Ansatz, both mass and momentum equations from Eqs.(2.1)-(2.2) are projected onto the subspace spanned by Φ . Concerning mass conservation, this is preserved automatically if the snapshots are divergence-free ($MX_j = 0$), regardless of $\mathbf{a}$ [15]. For the momentum equation, the projection will consider homogeneous boundary conditions. Otherwise, the reader is referred to Rosenberger and Sanderse [27] for a method that considers inhomogeneous boundary conditions. The projection of the momentum equation yields

$$\frac{d\mathbf{a}}{dt} = -\Phi^T C(\Phi\mathbf{a}_c)\Phi\mathbf{a} + \Phi^T D\Phi\mathbf{a} + \Phi^T \mathbf{f}, \quad (3.30)$$

where the weighted orthonormality condition $\Phi^T \Omega \Phi = I$ has been used. Here, $\mathbf{a}$ stands for the convected velocity in the reduced space while $\mathbf{a}_c$ stands for the convecting velocity in the reduced space. This linearization of the convective term, in explicit integration, follows a Picard linearization, where the convecting velocity is fixed to the latest available solution. This difference between the convecting and convected velocity is, therefore, a consequence of this linearization, and ensures the framework remains generalizable to non-autonomous transport equations, where the convecting velocity is not necessarily the solution of the system, yet a prescribed function of space and time. The pressure term has been dropped as it is exactly zero due to the divergence-free condition of the snapshots, as shown by

$$\Phi^T G \mathbf{p} = -\Phi^T \Omega M^T \mathbf{p} = (M \Omega \Phi)^T \mathbf{p} = 0. \quad (3.31)$$

The procedure to build the reduced convective term, $\Phi^T C(\Phi\mathbf{a}_c)\Phi\mathbf{a}$ depends on operations at the FOM level in an online phase, which break the efficiency of the ROM. This is circumvented following the works of Stabile and Rozza [39], which rewrote the convective term as

$$\Phi^T C(\Phi\mathbf{a}_c)\Phi\mathbf{a} \equiv \underbrace{[\Phi^T C(\Phi_1)\Phi]}_{C_{r,1}} C_{r,2} \dots C_{r,M}(\mathbf{a}_c \otimes \mathbf{a}) = C_r(\mathbf{a}_c \otimes \mathbf{a}) = \left(\sum_{i=1}^M a_{c,i} C_{r,i} \right) \mathbf{a}. \quad (3.32)$$

With this approach, all FOM-size operations are performed offline to construct $C_{r,i}$, which keeps the complexity of online operations at ROM level. This allows to rewrite Eq.(3.30) as

$$\frac{d\mathbf{a}}{dt} = -C_r(\mathbf{a}_c \otimes \mathbf{a}) + D_r \mathbf{a} + F_{r,0}, \quad (3.33)$$

where $D_r = \Phi^T D \Phi$ and $F_{r,0}$ is a constant term that arises from the projection of source terms. This algorithm results in a computational complexity of $\mathcal{O}(M^3)$. While not being the most efficient approach, usually the required number of modes M is low enough so that performing operations of this complexity is acceptable. In addition, the use of hyperreduction techniques in case of higher values of M should be considered to reduce the complexity to $\mathcal{O}(M^2)$ for the convective term.

Following the procedure from Rosenberger and Sandeher [27], which include non-homogeneous boundary conditions, the ROM can be rewritten as

$$\frac{d\mathbf{a}}{dt} = -C_r(\mathbf{a}_c \otimes \mathbf{a}) - C_{\text{hom,bc}}(\mathbf{a} \otimes \mathbf{a}_{bc}) + D_r \mathbf{a} + F_{r,0}, \quad (3.34)$$

where $C_{\text{hom,bc}}$ is the matrix containing the reduced contribution of the boundary conditions to the convective term, and $\mathbf{a}_{bc}$ are the coefficients of the boundary condition at time t [27]. Therefore, the linear contribution from the convective term in $\mathbf{a}$ is given by

$$C_l(t) = \frac{\partial C_{\text{hom,bc}}(\mathbf{a} \otimes \mathbf{a}_{bc})}{\partial \mathbf{a}} = C_{\text{hom,bc}}(\mathbf{I} \otimes \mathbf{a}_{bc}(t)) = [C_{\text{hom,bc},1} \ C_{\text{hom,bc},2} \ \dots \ C_{\text{hom,bc},M}] \mathbf{a}_{bc}(t) = \sum_{i=1}^{M_{bc}} a_{bc,i}(t) C_{\text{hom,bc},i}. \quad (3.35)$$

3.2.2 Generalizing the stability-based adaptive timestepping to ROMs

As for stability-based adaptive timestepping for FOMs, the main idea is to compute the eigenbounds of the system's linearized operators. Since the convection operator depends on the current velocity, its eigenbounds need to be computed at each timestep. The diffusion operator, however, is constant in time for DNS, so its eigenbounds can be computed once at the beginning of the simulation.

To preserve the efficiency gain achieved by the adaptive timestepping, the computational costs of this eigenbound computation should be negligible compared to the cost of performing a single timestep. Therefore, the computational cost should not exceed the algebraic complexity of the ROM, which is $\mathcal{O}(M^3)$ for the convective term. In a hyper-

reduction framework, this reduces to $\mathcal{O}(M^2)$, which is the cost of performing a single timestep. Therefore, the eigenbound computation should not exceed a computational complexity of $\mathcal{O}(M^2)$ in its online phase.

The exact computation of the eigenvalues of a matrix in $\mathbb{R}^{M \times M}$ has a computational cost of $\mathcal{O}(M^3)$, which is not acceptable for our purposes [42–44]. As an alternative, Gershgorin’s circle theorem can be applied to the reduced operators as for EigenCD [2], which has a computational cost of $\mathcal{O}(M^2)$ for dense matrices, as it only requires the computation of the sum of the absolute values of the entries of the matrix. This estimate can be quite inaccurate and hence result in timesteps significantly smaller than the largest stable timestep based on the exact eigenbounds. Another alternative is AlgEigCD, which leverages the structure of the convective and diffusive operators. The concept behind AlgEigCD is approximating the eigenbounds of the operators by finding a factorization $H = AB$ such that the eigenbound of $A^T B^T$ can be approximated efficiently. The reduced operators, however, have the form $H = \Phi^T H_F$, where H_F is the FOM operator. Following AlgEigCD, the eigenbound of this matrix H can be approximated by the eigenbound of $H^* = (\Phi^T A_F)^T (B_F \Phi)^T$, where $A_F B_F$ is a factorization of H_F . In contrast to H , H^* is a matrix of FOM dimension, which has an eigenbound estimation more expensive than that of H . Thus, such a transfer does not result in an efficient method, as it breaks the leitmotif of reduced-order modeling, which is to avoid online FOM-size operations. Therefore, a new approach that exploits the structure of the reduced operators is presented in this section, which allows to obtain an efficient estimation of the eigenbounds of the reduced operators without performing any FOM-size operation.

Concerning the diffusive operator, the application of the exact calculation of the eigenvalues is straightforward, as the diffusive operator at the ROM level is built as $\Phi^T D \Phi$, and it is a real symmetric matrix. Thus, its eigenvalues are real, and the eigenbound $\rho(D_r)$ can be exactly computed in an offline stage, as a computational complexity of $\mathcal{O}(M^3)$ is acceptable in the offline phase. On the other hand, the convective term is going to be split into the linear and the quadratic contribution. With respect to the latter, as defined in Eq.(3.33), the quadratic contribution to the convective term is given by $C_r(\mathbf{a}_c \otimes \mathbf{a})$. Its Jacobian matrix obtained via Picard linearization is given by

$$J_C = \frac{\partial C_r(\mathbf{a}_c \otimes \mathbf{a})}{\partial \mathbf{a}} = \sum_{i=1}^M a_{c,i} C_{r,i}, \quad (3.36)$$

which yields a skew-symmetric Jacobian iff $C_{r,i}$ is skew-symmetric for all i . Note that $\mathbf{a}_c$

changes at each timestep, but $C_{r,i}$ does not as it only depends on Φ_i , which is constant in time. Therefore, the eigenbound of the Jacobian of the quadratic contribution to the convective term can be approximated by exploiting that eigenbounds are subadditive, which yields

$$\rho(J_C) \leq \sum_{i=1}^M |a_{c,i}| \rho(C_{r,i}), \quad (3.37)$$

where $\rho(C_{r,i})$ can be exactly computed offline. This approach is efficient as it only requires the computation of the sum of the absolute values of the entries of $C_{r,i}$, which has a computational cost of $\mathcal{O}(M)$, as it corresponds to a dot product. This complexity is negligible compared to the $\mathcal{O}(M^3)$ cost of performing a single timestep, which is the cost of the ROM.

On the other hand, the linear contribution is in general neither symmetric nor skew-symmetric, as it depends on the boundary conditions. Therefore, the eigenvalues of the Jacobian of the linear contribution, $C_l(t)$, can be complex and time-dependent. As the contribution does not have a predefined structure, it is split into its symmetric and skew-symmetric part, which are defined as

$$C_{\text{hom,bc},i}^S = \frac{1}{2}(C_{\text{hom,bc},i} + C_{\text{hom,bc},i}^T), \quad (3.38)$$

$$C_{\text{hom,bc},i}^\Omega = \frac{1}{2}(C_{\text{hom,bc},i} - C_{\text{hom,bc},i}^T), \quad (3.39)$$

and will contribute to the real and imaginary part of the global eigenbound, respectively. Following the same approach as for the quadratic contribution, the eigenbound of the Jacobian of the linear contribution can be approximated by

$$\rho(C_l^S(t)) \leq \sum_{i=1}^{M_{bc}} |a_{bc,i}(t)| \rho(C_{\text{hom,bc},i}^S), \quad (3.40)$$

$$\rho(C_l^\Omega(t)) \leq \sum_{i=1}^{M_{bc}} |a_{bc,i}(t)| \rho(C_{\text{hom,bc},i}^\Omega), \quad (3.41)$$

where $\rho(C_{\text{hom,bc},i}^S)$ and $\rho(C_{\text{hom,bc},i}^\Omega)$ are exactly computed offline.

Hence, this new approach to estimate the eigenbounds used to determine the largest

stable timestep for the ROM is given by

$$\Re(\lambda_F) \leq \tilde{\Re}(\lambda_F) := \rho(D_r) + \sum_{i=1}^{M_{bc}} |a_{bc,i}(t)| \rho(C_{\text{hom},bc,i}^S), \quad (3.42)$$

$$\Im(\lambda_F) \leq \tilde{\Im}(\lambda_F) := \sum_{i=1}^M |a_{c,i}| \rho(C_{r,i}) + \sum_{i=1}^{M_{bc}} |a_{bc,i}(t)| \rho(C_{\text{hom},bc,i}^\Omega). \quad (3.43)$$

With all these ingredients, the stability-aware adaptive timestepping method for ROMs, RedEigCD, can be summarized in Algorithm 1, where T is the total time of the simulation.

Algorithm 1 Stability-aware self-adaptive timestep method for ROMs (RedEigCD)

Offline stage:

- 1: Compute Φ from snapshots.
- 2: Compute D_r as $\Phi^T D \Phi$.
- 3: Compute $C_{r,i}$ as in Eq.(3.33) for $i = 1, \dots, M$.
- 4: Compute $C_{l,i}$ as in Eq.(3.35) for $i = 1, \dots, M_{bc}$.
- 5: ★ Compute $\rho(D_r)$ and $\rho(C_{r,i})$ for $i = 1, \dots, M$.
- 6: ★ Compute $\rho(C_{l,i}^S)$ and $\rho(C_{l,i}^\Omega)$ for $i = 1, \dots, M_{bc}$.

Online stage:

- 1: Set $t = 0$, $\mathbf{a} = \mathbf{a}_0$, $\Delta t = \Delta t_0$.
- 2: **while** $t < T$ **do**
- 3: ★ Compute $\tilde{\Re}(\lambda_F)$ and $\tilde{\Im}(\lambda_F)$ as in Eq.(3.43).
- 4: ★ Compute z_{max} with the bisection method.
- 5: ★ Compute Δt as in Eq.(3.5).
- 6: Advance Eq.(3.34) in time with the chosen time integration scheme.
- 7: $t = t + \Delta t$

★ Steps required for an adaptive time integration with RedEigCD.

The operational differences between the proposed framework and classical stability-aware adaptive timestepping methods for FOMs are displayed in Table 3.1. The main difference is that the eigenbound estimation for the ROM is performed at a much lower computational cost compared to the FOM, which allows to perform stability-aware adaptive timestepping for ROMs without breaking their efficiency. In addition, the use of the exact eigenvalues for the diffusive term at the ROM level allows to obtain a more accurate estimation of the eigenbounds, which can result in larger stable timesteps compared to using Gershgorin's circle theorem.

Table 3.1: Description and comparison of computational complexity (offline vs. online) for EigenCD, AlgEigCD, and the proposed RedEigCD method.

Method	Description	Online	Offline
EigenCD [2]	Applies Gershgorin to the explicit $D_r, C_r(\mathbf{a})$ operators. Inaccurate for dense matrices in a reduced framework.	$\mathcal{O}(M^2)$	0
AlgEigCD [3]	Relies on the structure of $D_r, C_r(\mathbf{a})$ to reduce the eigenvalue computation to a matrix-vector product of FOM size.	$\mathcal{O}(N_F)$	$\mathcal{O}(N_F)$
RedEigCD (proposed)	Uses the tensorial decomposition to simplify the eigenvalue calculation for $C_r(\mathbf{a})$. Uses the exact eigenvalues for D_r .	$\mathcal{O}(M)$	$\mathcal{O}(M^3)$

3.2.3 Analytical stability analysis of ROMs

In practice, it has been observed that ROMs are stable for larger timesteps than the FOM. As noted by Baiges et al. [45], timestep stability limits for ROMs can exceed Courant numbers of 1, opposite to the FOM. Intuitively, cutting off some of the highest spatial frequencies, which are represented by the smallest singular values, would imply removing some of the most rapid modes. These are the modes that determine the system's stiffness, and thus are the most restrictive modes in terms of stability. By eliminating these modes, the stability constraint is more relaxed, and thus larger timesteps can be taken.

Bach et al. [34] analyzed the stability of projection-based ROMs compared to FOMs for linear elasticity problems which are linear problems with real symmetric matrices (like diffusion), where the timestep at the ROM level is demonstrated to be at least as large as the timestep at the FOM level, yet usually larger. To provide a mathematical explanation for this phenomenon, the Poincaré separation theorem was used, which reads as follows:

Lemma 1 (Poincaré separation theorem [46, 47]). *Let $V \in \mathbb{R}^{m \times k}$, where $k \ll m$, be a matrix with orthonormal column vectors, i.e., $V^T V = I_k$. Let $A \in \mathbb{R}^{m \times m}$ be a real symmetric matrix. Then, the eigenvalues of $V^T A V$ separate the eigenvalues of A such that*

$$\lambda_i \leq \mu_i \leq \lambda_{m-k+i}, \quad i \leq k, \quad (3.44)$$

where λ_i denotes the eigenvalues of A , and μ_i denotes the eigenvalues of $V^T A V$, ordered in non-decreasing order.

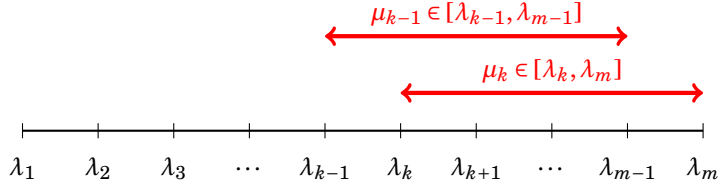


Figure 3.6: Graphical representation of the range of reduced eigenvalues μ_i with respect to the full eigenvalues λ_i according to the Poincaré separation theorem.

At the limit, $i = k$, the value μ_k denotes the eigenbound of $V^T A V$. According to Lemma 1, μ_k is bounded by λ_m , which is the eigenbound of A . For the sake of clarity, this has been represented graphically in Figure 3.6. This theorem can be applied straightforwardly to the diffusive operator and shows that $\rho(D_r) \leq \rho(D)$, as the diffusive operator corresponds to a real symmetric matrix, and Φ is an Ω -orthonormal matrix. Therefore, Lemma 1 holds as its result is not altered if the basis fulfills $V^T \Omega V = I_k$ instead of $V^T V = I_k$ [34].

In contrast, the convective operator corresponds to a non-symmetric matrix, and thus Lemma 1 does not hold. Therefore, the stability of ROMs for larger timesteps than FOMs cannot be explained by the Poincaré separation theorem. Therefore, the work of Bach et al. [34] is extended to the case of ROMs with non-symmetric operators. To do so, the work of Rao [48] is employed, which extends the Poincaré separation theorem to singular values instead of eigenvalues, and thus holds even for the case of non-symmetric matrices.

Lemma 2 (Poincaré separation theorem for singular values [48]). *Let $A \in \mathbb{R}^{m \times n}$. Let $B \in \mathbb{R}^{m \times k}$ and $C \in \mathbb{R}^{n \times r}$ be orthonormal matrices, i.e. $B^T B = I_k$ and $C^T C = I_r$. The i -th singular value of A is denoted by σ_i in increasing order, as for the Poincaré separation theorem. Moreover, $\tilde{\sigma}_i$ denotes the i -th singular value of $B^T A C$. Then, the following inequality holds:*

$$\sigma_i \leq \tilde{\sigma}_i \leq \sigma_{\tilde{m}-i+1}, \text{ for } 1 \leq i \leq \min(r, k), \quad (3.45)$$

where $\tilde{m} = \min(m, n)$.

Let, then, A be a square matrix and $B = C$. In the current case, $A = C(\mathbf{u})$, and $B = \Phi$. Hence, this theorem can be applied to the convective term by making use of the fact

that the singular values of a real skew-symmetric matrix correspond to the absolute values of the eigenvalues, which suffices to estimate the eigenbound of the operator, as defined by Lemma 3.

Lemma 3. *Let $A \in \mathbb{C}^{n \times n}$ be a normal matrix (i.e. $A^*A = AA^*$). Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of A , listed with algebraic multiplicity. Let $\sigma_1, \dots, \sigma_n$ be the singular values of A (the nonnegative square roots of the eigenvalues of A^*A), also listed with multiplicity. Then, the singular values are the absolute values of the eigenvalues:*

$$\{\sigma_1, \dots, \sigma_n\} = \{|\lambda_1|, \dots, |\lambda_n|\}. \quad (3.46)$$

Proof. As A is a normal matrix, it can be diagonalized with a unitary matrix U , which is a set of orthonormal eigenvectors such that $U^*U = I$, and a diagonal matrix Λ , which has the eigenvalues of A as entries, i.e. $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ [49, pp. 101–103]. Therefore,

$$A = U\Lambda U^*. \quad (3.47)$$

Then, A^*A can be written as

$$A^*A = (U\Lambda U^*)^*(U\Lambda U^*) = U\Lambda^*U^*U\Lambda U^* = U\Lambda^*\Lambda U^*, \quad (3.48)$$

where $\Lambda^*\Lambda = \text{diag}(\lambda_1^*\lambda_1, \dots, \lambda_n^*\lambda_n) = \text{diag}(|\lambda_1|^2, \dots, |\lambda_n|^2)$. Therefore,

$$A^*A = U\text{diag}(|\lambda_1|^2, \dots, |\lambda_n|^2)U^*. \quad (3.49)$$

Then, the i -th column of U , $\mathbf{u}_i$, is going to be an eigenvector of A^*A with eigenvalue $|\lambda_i|^2$, since $A^*A\mathbf{u}_i = \mathbf{u}_i|\lambda_i|^2$. By definition, the singular values of A are the eigenvalues of A^*A . Therefore, it shows that Eq.(3.46) holds and, with a proper reordering,

$$\sigma_i = |\lambda_i|, \quad \forall i. \quad (3.50)$$

□

Hence, the inequality can be rewritten for $r = k$, $m = n$ with $i = k$ to estimate the eigenbound of the convective term, which reads as follows,

$$\sigma_k \leq \tilde{\sigma}_k \leq \sigma_{m-k+1} \leq \sigma_m, \quad (3.51)$$

as graphically represented in Figure 3.7.

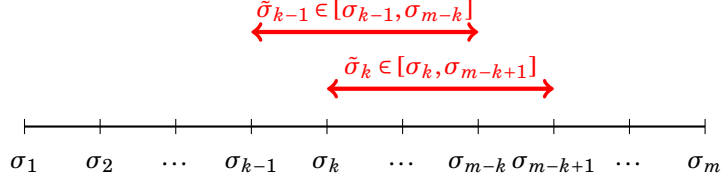


Figure 3.7: Graphical representation of the range of reduced singular values for $i = k$ and $i = k - 1$, in red, according to the Poincaré separation theorem for singular values.

Compared to Lemma 1, Eq.(3.51) does include the inequality $\sigma_{m-k+1} \leq \sigma_m$, as Lemma 2 bounds the singular values of the reduced matrix with σ_k and σ_{m-k+1} as shown in Fig.3.6, yet by the assumed order of the singular values $\sigma_{m-k+1} \leq \sigma_m$. As stated by Lemma 3, for a normal matrix such as the convective operator, given its skew-symmetry, results in terms of singular values can be translated to eigenvalues. Thus, the eigenvalues of $C(\mathbf{u})$ are greater than those of $C_r(\mathbf{a})$. Therefore, $\rho(C_r) \leq \rho(C)$.

As Eq.(3.43) indicates, the magnitude of the relevant eigenvalue in the ROM is determined by the contributions of both diffusive and convective term. Thus, the following theorem can be stated.

Theorem 6. Let Δt^{ROM} be the largest stable timestep for the ROM given a state $\mathbf{a}$, and Δt^{FOM} the largest stable timestep for a state $\mathbf{u}$ that is in the span of Φ , e.g. $\mathbf{u} = \Phi \mathbf{a}$, both using the same explicit time integration scheme. Then, $\Delta t^{\text{ROM}} \geq \Delta t^{\text{FOM}}$.

Proof. Let $\lambda_{F,\text{FOM}} = \rho(D) + i\rho(C(\mathbf{u}))$ be the eigenvalue of the FOM, and $\lambda_{F,\text{ROM}}$ be calculated as in Eq.(3.43). As shown above, $\rho(D_r) \leq \rho(D)$ (Lemma 1). Moreover, given $\mathbf{u} = \Phi \mathbf{a}$ is the FOM equivalent of $\mathbf{a}$, $\rho(C_r) \leq \rho(C)$ (Lemma 2). Therefore, $\|\lambda_{F,\text{FOM}}\| \geq \|\lambda_{F,\text{ROM}}\|$, where $\|\cdot\|$ denotes the complex modulus. Let Δt be calculated as in Eq.(3.5). As the stability region of the explicit time integration scheme is fixed, and Δt is inversely proportional to $\|\lambda_F\|$, then $\Delta t^{\text{ROM}} \geq \Delta t^{\text{FOM}}$. $\square$

Remark 1 (Applicability to time-integration). *Strictly speaking, Theorem 6 guarantees $\Delta t^{\text{ROM}} \geq \Delta t^{\text{FOM}}$ only when the convective field is identical for both FOM and ROM (i.e. $\mathbf{u} = \Phi \mathbf{a}$). In a practical time-integration scenario, the ROM trajectory $\mathbf{u}_r = \Phi \mathbf{a}(t)$ will deviate from the FOM solution $\mathbf{u}(t)$. Consequently, a rigorous inequality cannot be guaranteed for $t > 0$.*

However, Theorem 6 serves as a strong theoretical proxy for the stability behavior of the ROM. Since the projection Φ inherently removes the high-frequency modes associated

with the largest eigenvalues of $C(\mathbf{u})$, the inequality $\rho(C_r) < \rho(C)$ is expected to hold robustly in practice. The theorem thus provides the structural justification for the increased stability limits observed in numerical experiments, even in the presence of trajectory divergence.

3.2.4 Numerical experiments

The proposed RedEigCD method is applied to two different fluid flow test cases. First of all, the robustness of the method in a case with periodic boundary conditions, namely the shear-layer roll-up, is presented. Moreover, the evolution of the maximum stable Δt for different numbers of modes is presented, and compared with the maximum stable Δt using AlgEigCD. In the second case, the influence of non-homogeneous boundary conditions on the preservation of stability is considered for the unsteady wake behind an actuator disk [15]. Both test cases have been run using the classical explicit fourth-order RK scheme applied to a two-dimensional incompressible flow solver [14, 50]. Both methods have been run for $M = [16, 32, 64, 128, 200]$ modes. Table 3.2 summarizes the main characteristics of the test cases.

Table 3.2: Summary of the methods considered in this work.

Case	Method	Description / Purpose
A	FOM+AlgEigCD	Full-order model with staggered AlgEigCD (3.1.3); used as reference.
B	ROM+RedEigCD	ROM with proposed self-adaptive timestep method.
C	ROM+constant	ROM with constant Δt ; used to compare accuracy of ROM+RedEigCD.
D	Best ROM	Best approximation ROM (Eq. (3.53)) with constant Δt ; used to verify $\Delta t_{\text{ROM}} > \Delta t_{\text{FOM}}$ throughout simulation.

Shear-layer roll-up In this case, the roll-up of a shear layer has been simulated in a square domain of side length 2π with full-periodic boundary conditions. The initial condition is given by

$$u_0(x, y) = 1 + \begin{cases} \tanh\left(\frac{y-\pi/2}{\delta}\right), & y \leq \pi, \\ \tanh\left(\frac{3\pi/2-y}{\delta}\right), & y > \pi, \end{cases} \quad v_0(x, y) = \epsilon \sin(x), \quad (3.52)$$

where $\delta = \pi/15$ is the initial thickness of the shear layer, and $\epsilon = 1/20$ is the amplitude of the perturbation. The Reynolds number is set to $Re = 1000$. The FOM discretization used for this case employs 100×100 finite volumes, with variable timestep computed using AlgEigCD (Section 3.1.3). The snapshots are collected for $t = 0$ to $t = 20$ from Case A (Table 3.2). Figure 3.8 shows the decay of the singular values for this case.

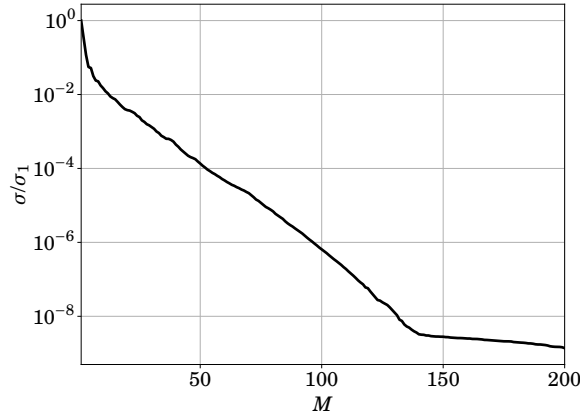


Figure 3.8: Decay of the singular values for the shear-layer roll-up case for $Re = 1000$ up to $M = 200$.

Moreover, note that even for the largest number of modes considered, $M = 200$, where the singular values have decayed up to $\mathcal{O}(10^{-9})$, the number of degrees of freedom has notably reduced compared to those of the FOM, which is 3×10^4 , considering both velocity components and the pressure. The simulation has been run for Case B (ROM+RedEigCD) with the number of modes previously stated and the evolution of Δt is compared with Case A.

Figure 3.9 (left) shows that the adaptive timestepping method is effective in providing a larger timestep for the FOM than for the ROM, as expected from Theorem 6, as even for the largest number of modes considered, the timestep for the ROM and the timestep for the FOM are of similar value, leading to a ratio of around 1 for the whole simulation. Depending on the current flow physics and the number of modes, this timestep can be up to 9 times larger than for the FOM, which significantly accelerates the simulation by notably reducing the number of required timesteps. Moreover, the ratio of timesteps converges to 1 for increasing the ROM dimension, as expected from Theorem 6, as the eigenvalues of the ROM operators interlace with those of the FOM. Physically, this

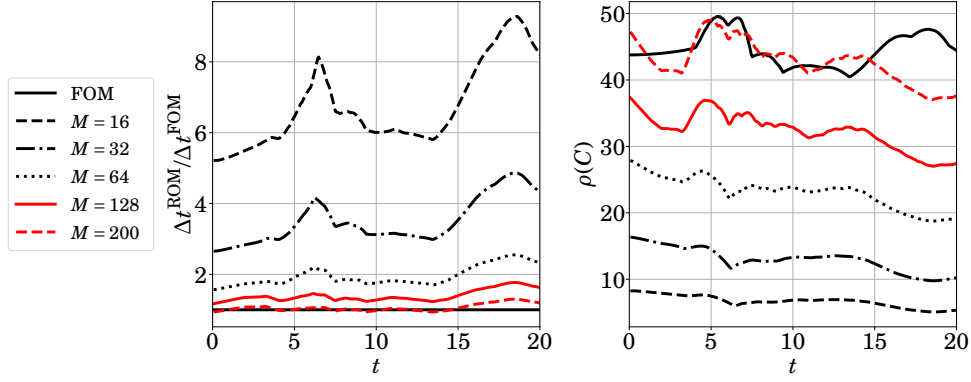


Figure 3.9: Timestep ratio between Case B and Case A up to $t = 20$ (left) and evolution of the imaginary eigenbound estimate (right) for the roll-up of a shear-layer at $Re = 1000$ for different numbers of modes.

is driven by the inclusion of higher-index POD modes, which capture steeper spatial gradients. Therefore, the ROM's stability constraints monotonically approach the FOM limit.

Concerning the time-evolution of $\rho(C)$ (Figure 3.9 right), the expected behaviour as theoretically derived in Section 3.2.3 is observed, as the eigenbound of the reduced convective term steadily increases with the number of modes. For $M = 200$, however, in some points the imaginary eigenbound for the ROM is larger than that of the FOM. The reason for this mismatch with theory reported for $M = 200$ is as follows. While Section 3.2.3 considers that the given matrix at FOM level and its reduced counterpart are evaluated at the equivalent state vector $\mathbf{u}$. However, in a ROM simulation at a given t , the reduced velocity $\mathbf{u}_r(t) = \Phi \mathbf{a}(t)$ is not identical to its FOM counterpart $\mathbf{u}(t)$, and therefore both velocity fields lead to non-identical convective operators. Given these differences, Lemma 2 acts as an approximation, and thus the eigenbound of the ROM can be larger than that of the FOM at some points.

Therefore, to test if the Δt at ROM level using the equivalent state vector is always larger than the one for the FOM actually holds, the best approximation ROM [15] is computed, which is defined as the ROM obtained by projecting the FOM snapshots onto the subspace spanned by the first M POD modes, i.e.,

$$\mathbf{a}_{best} = \Phi^T \Omega \mathbf{u}(t), \quad (3.53)$$

which should fulfill Theorem 6 throughout the whole simulation as both $C(\mathbf{u})$ and $C_r(\mathbf{a}_{best})$ are evaluated at the equivalent state vector. Hence, the reduced convective and diffusive operators for every snapshot have been constructed and the eigenbounds using the RedEigCD method have been computed. As shown in Figure 3.10, the result from Theorem 6 is verified, as the timestep for the best approximation ROM is always larger than the one for the FOM, even for $M = 200$. Moreover, the timestep ratio for the best approximation ROM converges to 1 for increasing the number of modes, as expected from Theorem 6, as the eigenvalues of the ROM operators interlace with those of the FOM. Physically, this is driven by the inclusion of higher-index POD modes, which capture steeper spatial gradients. Therefore, the ROM's stability constraints monotonically approach the FOM limit.

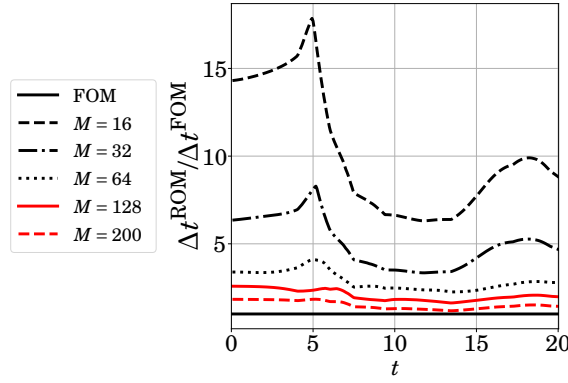


Figure 3.10: Evolution of the timestep ratio for the best approximation ROM (Eq.(3.53)) for the roll-up of a shear-layer at $Re = 1000$ for different numbers of modes.

Figure 3.11 shows that the errors for Case B and Case D have the same magnitude regardless of the number of modes ($M = 16$, left; $M = 200$, right). Thus, the use of the proposed stability-aware adaptive timestepper does not alter the accuracy of the ROM compared to using a constant timestep, as the error is dominated by the projection error, which is the same for both cases. In this case, Δt for Case C is set to 0.01, as in the test case from [15]. To isolate the temporal accuracy of the stability-controlled ROM from the spatial truncation error, the reduced error relative to the projected FOM $\mathbf{a}_{best}$ is computed, which is defined as

$$\varepsilon(\mathbf{a}_{test}(t)) = \frac{\|\mathbf{a}_{test}(t) - \mathbf{a}_{best}(t)\|_2}{\|\mathbf{a}_{best}(t)\|_2}, \quad (3.54)$$

where $\mathbf{a}_{\text{test}}$ is the state vector of the ROM with either RedEigCD or constant Δt . As shown in Figure 3.11, the error for both cases is of the same order of magnitude, which confirms that the use of the proposed stability-aware adaptive timestep method does not alter the accuracy of the ROM compared to using a constant timestep, as the error is dominated by the projection error, which is the same for both cases. Moreover, it can be observed that for $M = 16$ (left), the error is larger than for $M = 200$ (right), as expected from the decay of the singular values shown in Figure 3.8, as more modes are included in the ROM and thus more spatial frequencies are captured, which leads to a smaller projection error.

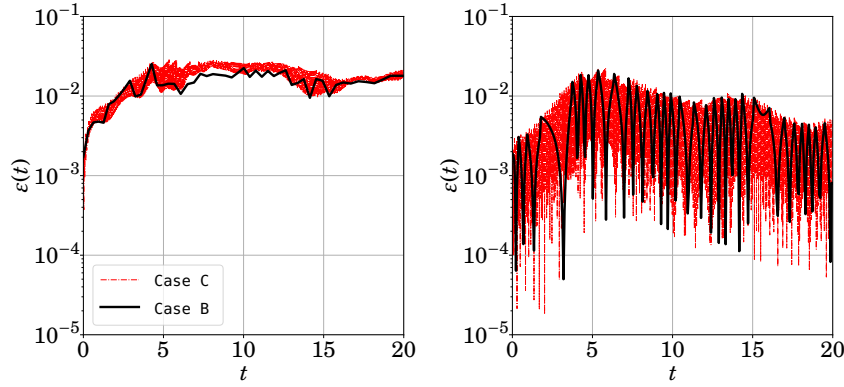


Figure 3.11: Evolution of the relative error with respect to the best approximation for Case B and Case C for $M = 16$ (left) and $M = 200$ (right) for the roll-up of a shear-layer at $\text{Re} = 1000$.

Flow through an actuator disk In this case, the air flow around a wind turbine is modelled as in [27]. The geometrical domain is $[0, 10] \times [-2, 2]$, with a time-dependent prescribed inlet velocity at the left boundary, which is given by

$$u(0, y, t) = \cos(\alpha(t)), \quad v(0, y, t) = \sin(\alpha(t)), \quad \alpha(t) = \frac{\pi}{6} \sin\left(\frac{t}{2}\right). \quad (3.55)$$

The other boundaries correspond to an outflow condition [14, Section 6.4.2]- The wind turbine has been modelled using an actuator disk of diameter 1, placed at $(2, 0)$, which corresponds to a momentum sink modelled as a constant force $f = 0.25$ acting in the negative x -direction. The initial conditions for this case are

$$u_0(x, y) = 1, \quad v_0(x, y) = 0. \quad (3.56)$$

The Reynolds number is 100. The FOM discretization used for this case employs 200×80 finite volumes, with variable timestep computed using AlgEigCD (Case A). The numerical tests performed correspond to those of Table 3.2. Snapshots have been collected for $t = 0$ to $t = 8\pi$ from Case A. Figure 3.12 shows the decay of the singular values for this case, which follow the same trend as in the previous case, with a notable decay up to $M = 200$.

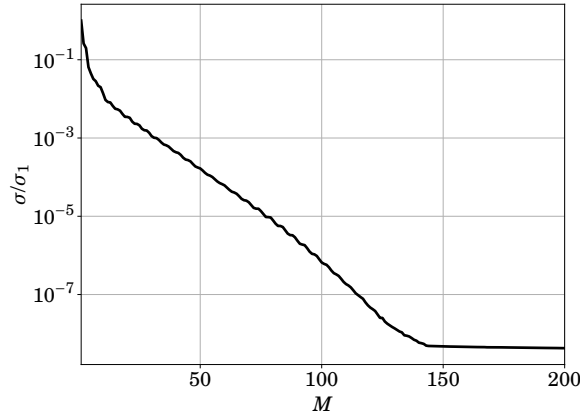


Figure 3.12: Decay of the singular values for the flow through an actuator disk case for $\text{Re} = 100$ up to $M = 200$.

The new stability-aware adaptive timestepper is effective in increasing the timestep while maintaining stable solutions for non-homogeneous boundary conditions as shown in Figure 3.13 (left), where the timestep ratio between Case B and Case A is shown. For this case, depending on the flow conditions and number of modes, the timestep for the ROM can be up to 40 times larger than for the FOM, thus leading to significantly faster simulations. Moreover, as expected from Section 3.2.3, the timestep ratio converges to 1 for increasing the ROM dimension, as the eigenvalues of the ROM operators interlace with those of the FOM. As it can be observed in Figure 3.13 (right), the eigenbound of the whole $F = -C_r(\mathbf{a}_c) + C_l + D_r$ steadily increases with the number of modes, as expected from Section 3.2.3. Opposite to the shear-layer case, for $M = 200$ the eigenbound of the ROM is always smaller than that of the FOM, which is consistent with the theoretical results from Section 3.2.3. Note that in this case, the presence of non-homogeneous boundary conditions contributes for both real and imaginary eigenbounds, this being the reason why the whole eigenbound is depicted in Figure 3.13 (right) instead of only

the imaginary part as in the shear-layer case.

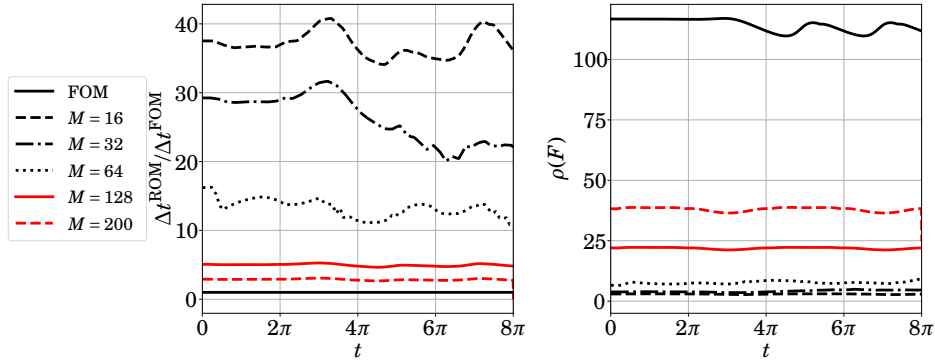


Figure 3.13: Timestep ratio between Case B and Case A up to $t = 8\pi$ (left) and evolution of the eigenbound estimate (right) for the numerical simulation of a wind field of $\text{Re} = 100$.

The magnitude of the ROM error, computed with Eq.(3.54), is not affected by the variable Δt compared to the Case C, where Δt is set to $4\pi/200$ according to the results from [15], as shown in Figure 3.14 for $M = 16$ (left) and $M = 200$ (right).

3.2.5 Accuracy of the eigenbound estimates

A crucial aspect of RedEigCD is the accuracy of the eigenbound estimates, as they directly influence the stability and thus the maximum timestep allowed. This has been studied for both the test cases from Section 3.2.4 by comparing the eigenbound estimates obtained from RedEigCD with those obtained from a direct computation of the eigenvalues of the reduced operators, which is feasible for the ROM dimensions considered in this work. The results have also been compared with the eigenbound estimates obtained by applying Gershgorin's circle theorem directly to $a_i C_{r,i}$, which would replicate EigenCD at the ROM level, and thus would be the most straightforward way to compute the eigenbound estimates for the ROM. As shown in Figure 3.15, the eigenbound estimates obtained from RedEigCD are in good agreement with those obtained from direct eigenvalue computation, while the estimates obtained from Gershgorin's circle theorem are notably more conservative, especially for an increasing number of modes, which is consistent with the expected deterioration of the estimates from Gershgorin's circle theorem for increasing matrix dimensions in dense matrices. The accuracy of the eigenbound estimates from RedEigCD is crucial to ensure that the adaptive timestep method

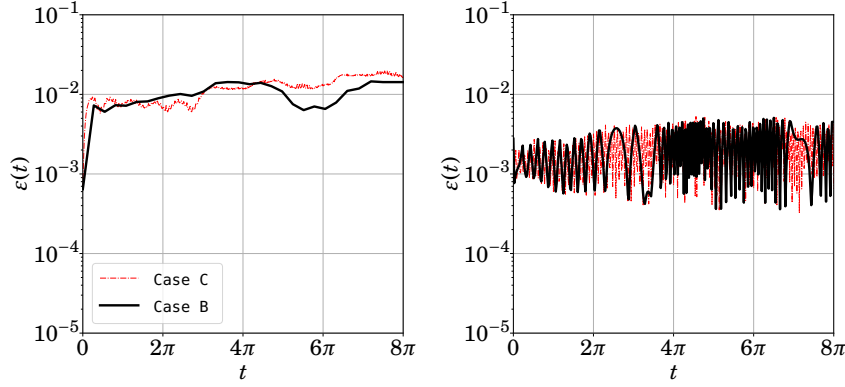


Figure 3.14: Evolution of the relative error with respect to the best approximation for Case B and Case C for $M = 16$ (left) and $M = 200$ (right) for the numerical simulation of a wind field of $Re = 100$.

is effective in providing a larger timestep for the ROM than for the FOM, as shown in Figures 3.9 and 3.13.

To compute an overall error between the eigenbound estimates from RedEigCD and those from direct eigenvalue computation, the relative error is computed as

$$\varepsilon_{\text{est}} = \frac{\|\lambda_{a_i C_{r,i}} - \lambda_{\text{est}}\|_{\infty}}{\|\lambda_{a_i C_{r,i}}\|_{\infty}}, \quad (3.57)$$

where $\lambda_{a_i C_{r,i}}$ are the eigenvalues of $a_i C_{r,i}$ obtained from direct eigenvalue computation, and λ_{est} are the eigenbound estimates obtained from RedEigCD or Gershgorin's circle theorem. The results show that the relative error for RedEigCD is around a 15% larger for the shear-layer case and a 35% larger for the actuator case, while for the Gershgorin's circle theorem, these errors are around a 353% and a 288%, respectively. Therefore, estimating the eigenbounds for both cases using RedEigCD leads to a notably smaller error compared to using Gershgorin's circle theorem, which is crucial to ensure the effectiveness of the adaptive timestep method in providing a larger timestep for the ROM than for the FOM, on top of the fact that RedEigCD's computational complexity is limited to $\mathcal{O}(M)$ instead of Gershgorin's $\mathcal{O}(M^2)$.

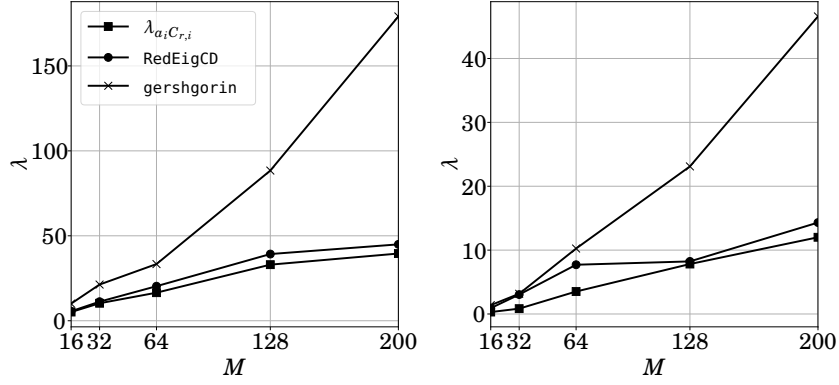


Figure 3.15: Comparison of the eigenbound estimates obtained from RedEigCD, Gershgorin's circle theorem, and direct eigenvalue computation for the roll-up of a shear-layer at $Re=1000$ (left) and the flow through an actuator disk at $Re=100$ (right) for different numbers of modes.

3.2.6 Computational performance of RedEigCD

Assessing the performance of RedEigCD in terms of wall-clock time requires a careful analysis of the online computational efficiency as well as the offline cost of the method. While a complexity analysis is presented in Table 3.1, it is crucial to validate these theoretical estimates with empirical measurements. To do so, the wall-clock time for both the offline and online stages of RedEigCD is measured and compared against the baseline methods (EigenCD and AlgEigCD) for a range of ROM dimensions M .

The online computational efficiency of the RedEigCD algorithm is evaluated by comparing the total online wall-clock time of the ROM against the FOM. A total speed-up factor, $\mathcal{S}$ has been defined as the product of the reduction in cost per timestep and the increase in the average timestep size, which is expressed as

$$\mathcal{S} = \frac{T_{\text{FOM}}}{T_{\text{ROM}}} \frac{\Delta t^{\text{ROM}}}{\Delta t^{\text{FOM}}}, \quad (3.58)$$

where T_{FOM} and T_{ROM} are the average wall-clock time per timestep for the FOM and ROM, respectively. Figure 3.16 shows the speed-up factor $\mathcal{S}$ for both test cases as a function of the number of modes. The results indicate that the proposed RedEigCD method can achieve significant speed-ups compared to the FOM, especially for lower ROM dimensions where the increase in timestep size is more pronounced. For these

lower dimensions, the computational complexity of $\mathcal{O}(M^3)$ is acceptable and leads to wall-clock times that are independent of the number of modes. For the larger numbers, however, hyper-reduction would be preferable as the high complexity of the ROM starts penalizing. For the shear-layer case, overall speed-ups of up to 800 times are observed for $M = 16$, while for the actuator case, this figure exceeds 3000 times for $M = 16$. However, even for $M = 200$, a notable speed-up is still observed, which highlights the effectiveness of the proposed method in accelerating ROM simulations while maintaining stability.

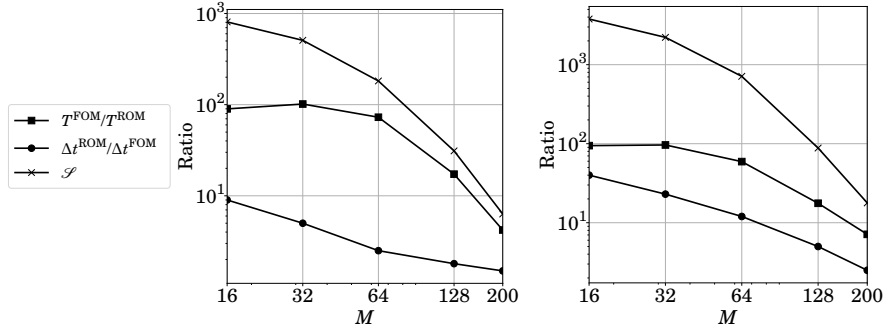


Figure 3.16: Speed-up factor $\mathcal{S}$ for the roll-up of a shear-layer at $\text{Re}=1000$ (left) and the flow through an actuator disk at $\text{Re}=100$ (right) for different numbers of modes, where $\mathcal{S}$ is defined as the product of the reduction in cost per timestep (WCT) and the increase in the average timestep size (Δt).

On the other hand, regarding the offline costs, a cost metric $\mathcal{C}$ is defined as the equivalent number of ROM timesteps required to perform the offline computations of RedEigCD, which is expressed as

$$\mathcal{C} = \frac{T_{\text{offline}}}{T_{\text{ROM}}}, \quad (3.59)$$

where T_{offline} is the total wall-clock time for the offline computations of RedEigCD, and T_{ROM} is the average wall-clock time per timestep for the ROM. For the most expensive case ($M = 200$), the offline cost of RedEigCD is equivalent to approximately 20 ROM iterations. While for extremely short iterations this overhead may be significant, for production runs involving thousands of timesteps, the cost of the offline stage of RedEigCD is negligible compared to the overall simulation time, especially considering the significant speed-ups achieved during the online stage.

3.3 Conclusions

In this chapter, self-adaptive timestepping methods for incompressible flows have been presented. These methods are based on a linear stability analysis of the momentum equation, which is built upon the estimation of the eigenbounds of the diffusive and convective term. The proposed methods in the literature, EigenCD and AlgEigCD, have been reviewed, with the latter being extended to the case of staggered grids.

Stability-aware adaptive timestepping methods are crucial for accelerating the solution of incompressible flow problems. While classical CFL conditions provide a useful guideline for selecting the timestep, they are usually overconservative, and lack the ability to cross the information from both convective and diffusive terms. The methods presented in this chapter provide a more accurate estimation of the stability limits, and the information from both operators is combined to compute the maximum stable timestep, which usually exceeds the one obtained from classical CFL conditions.

Moreover, a new method, RedEigCD, has been proposed to estimate the eigenbounds of the reduced operators in ROMs. The stability properties of ROMs have been analyzed by making use of the Poincaré separation theorem for singular values, which has been used to show that the eigenvalues of the reduced convective operator are smaller than those of the FOM, thus leading to larger stable timesteps for the ROM compared to the FOM. The proposed RedEigCD method has been applied to two different test cases, where it has been shown that it is effective in providing larger timesteps for the ROM than for the FOM while maintaining stable solutions. Moreover, it has been shown that the use of RedEigCD does not alter the accuracy of the ROM compared to using a constant timestep. Finally, the computational performance of RedEigCD has been assessed, showing significant speed-ups compared to the FOM while maintaining stability.

On top of this, an analytical foundation on the stability properties of ROMs has been provided, which sets a mathematical background to understand why ROMs can be more stable than FOMs, and thus why they can allow for larger timesteps. This is a crucial aspect to understand the behavior of ROMs, and to design new methods that can further enhance their stability properties. This analysis, however, is not restricted to projection-based ROMs applied to incompressible flows, but can be extended to any sort of projection-based ROM. The results from this chapter therefore provide a strong theoretical and practical basis for the use of stability-aware adaptive timestepping methods in ROMs for incompressible flows, which can significantly accelerate the simulations while maintaining stability and accuracy.

However, the proposed RedEigCD method is developed based on ROMs with computational complexity that scales with the cube of the number of modes, which can be prohibitive for large ROM dimensions. Therefore, future work will focus on extending the proposed method to a hyper-reduced version, which would allow to estimate the eigenbounds of the reduced operators with a computational complexity that is independent of the number of modes. This is observed with a decay of the overall speed-ups obtained for $M > 64$ in both test cases. Moreover, future work will also focus on extending the proposed method to other types of ROMs, such as those based on machine learning techniques and operator inference (OpInf), which are becoming increasingly popular in the field of CFD.

References

- [1] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100(1):32–74, December 1928.
- [2] F. X. Trias and O. Lehmkuhl. A Self-Adaptive Strategy for the Time Integration of Navier-Stokes Equations. *Numerical Heat Transfer, Part B: Fundamentals*, 60(2):116–134, August 2011.
- [3] F. X. Trias, X. Álvarez-Farré, A. Alsalti-Baldellou, A. Gorobets, and A. Oliva. An efficient eigenvalue bounding method: CFL condition revisited. *Computer Physics Communications*, 305, December 2024.
- [4] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II*, volume 14 of *Springer Series in Computational Mathematics*. Springer, Berlin, Heidelberg, 1996.
- [5] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187(1):343–368, May 2003.
- [6] F. X. Trias, O. Lehmkuhl, A. Oliva, C. D. Pérez-Segarra, and R. W.C.P. Verstappen. Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids. *Journal of Computational Physics*, 258:246–267, February 2014.
- [7] Ivar Bendixson. Sur les racines d’une équation fondamentale. *Acta Mathematica*, 25(0):359–365, 1902.

- [8] S Gerschgorin. über die Abgrenzung der Eigenwerte einer Matrix. *Bulletin de l'Académie des Sciences de l'URSS: Classe des sciences mathématiques et na*, pages 749–754, 1931.
- [9] B. Sanderse, R.W.C.P. Verstappen, and B. Koren. Boundary treatment for fourth-order staggered mesh discretizations of the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 257:1472–1505, January 2014.
- [10] Oskar Perron. Zur Theorie der Matrices. *Mathematische Annalen*, 64(2):248–263, June 1907.
- [11] G. Frobenius. über Matrizen aus nicht negativen Elementen. *Sitzung der physikalisch-mathematischen Classe*, pages 456–477, 1912.
- [12] F. R. Gantmakher. *Applications of the Theory of Matrices*. Dover Publications, Mineola, N.Y., 2005.
- [13] Vladimir Nikiforov. Chromatic number and spectral radius. *Linear Algebra and its Applications*, 426(2-3):810–814, October 2007.
- [14] B. Sanderse. *Energy-Conserving Discretization Methods for the Incompressible Navier-Stokes Equations: Application to the Simulation of Wind-Turbine Wakes*. PhD thesis, Technische Universiteit Eindhoven, 2013.
- [15] B. Sanderse. Non-linearly stable reduced-order models for incompressible flow with energy-conserving finite volume methods. *Journal of Computational Physics*, 421, November 2020.
- [16] C. Leblond, C. Allery, and C. Inard. An optimal projection method for the reduced-order modeling of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 200(33):2507–2527, August 2011.
- [17] Ricardo Reyes and Ramon Codina. Projection-based reduced order models for flow problems: A variational multiscale approach. *Computer Methods in Applied Mechanics and Engineering*, 363:112844, May 2020.
- [18] Aviral Prakash and Yongjie Jessica Zhang. Projection-based reduced order modeling and data-driven artificial viscosity closures for incompressible fluid flows. *Computer Methods in Applied Mechanics and Engineering*, 425:116930, May 2024.

- [19] Alfio Quarteroni and Gianluigi Rozza. Numerical solution of parametrized Navier–Stokes equations by reduced basis methods. *Numerical Methods for Partial Differential Equations*, 23(4):923–948, July 2007.
- [20] Andrea Manzoni. An efficient computational framework for reduced basis approximation and *a posteriori* error estimation of parametrized Navier–Stokes flows. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(4):1199–1226, July 2014.
- [21] John L. Lumley. The structure of inhomogeneous turbulence. In *Atmospheric Turbulence and Wave Propagation*, pages 166–178. Moscow, 1967.
- [22] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. I. Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, October 1987.
- [23] Philip Holmes, John L. Lumley, and Gal Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, 1 edition, October 1996.
- [24] David Amsallem and Charbel Farhat. Stabilization of projection-based reduced-order models. *International Journal for Numerical Methods in Engineering*, 91(4):358–377, 2012.
- [25] Zhu Wang, Imran Akhtar, Jeff Borggaard, and Traian Iliescu. Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237–240:10–26, September 2012.
- [26] Shady E. Ahmed, Suraj Pawar, Omer San, Adil Rasheed, Traian Iliescu, and Bernd R. Noack. On closures for reduced order models—A spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, 33(9):091301, September 2021.
- [27] H. Rosenberger and B. Sanderse. No pressure? Energy-consistent ROMs for the incompressible Navier-Stokes equations with time-dependent boundary conditions. *Journal of Computational Physics*, 491:112405, October 2023.
- [28] R. B. Klein and B. Sanderse. Energy-conserving hyper-reduction and temporal localization for reduced order models of the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 499:112697, February 2024.

- [29] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T. Patera. An ‘empirical interpolation’ method: Application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, November 2004.
- [30] Saifon Chaturantabut and Danny C. Sorensen. A State Space Error Estimate for POD-DEIM Nonlinear Model Reduction. *SIAM Journal on Numerical Analysis*, 50(1):46–63, January 2012.
- [31] Mansoor Ahmed and Omer San. Stabilized principal interval decomposition method for model reduction of nonlinear convective systems with moving shocks. *Computational and Applied Mathematics*, 37(5):6870–6902, November 2018.
- [32] Giovanni Stabile, Francesco Ballarin, Giacomo Zuccarino, and Gianluigi Rozza. A reduced order variational multiscale approach for turbulent flows. *Advances in Computational Mathematics*, 45(5-6):2349–2368, December 2019.
- [33] Xi Li, Youcai Xu, and Minfu Feng. A Pressure-Stabilized Continuous Data Assimilation Reduced Order Model for Incompressible Navier–Stokes Equations. *Journal of Scientific Computing*, 103(1):10, April 2025.
- [34] C. Bach, L. Song, T. Erhart, and F. Duddeck. Stability conditions for the explicit integration of projection based nonlinear reduced-order and hyper reduced structural mechanics finite element models, 2018.
- [35] Vedant Puri, Aviral Prakash, Levent Burak Kara, and Yongjie Jessica Zhang. Snf-rom: Projection-based nonlinear reduced order modeling with smooth neural fields. *Journal of Computational Physics*, 532:113957, 2025.
- [36] Suraj Pawar, Omer San, Adil Rasheed, and Shady Vedantam. A neural network-based projection-based rom for incompressible flows with coarsened time-stepping. *Journal of Computational Physics*, 523:113615, 2025.
- [37] Omer San and Romit Maulik. Extreme learning machine for reduced order modeling of turbulent geophysical flows. *Physical Review E*, 97(4):042322, 2018.
- [38] Kevin Carlberg, Matthew Barone, and Harbir Antil. Galerkin v. least-squares Petrov-Galerkin projection in nonlinear model reduction. *Journal of Computational Physics*, 330:693–734, 2017.

- [39] Giovanni Stabile and Gianluigi Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier–Stokes equations. *Computers & Fluids*, 173:273–284, September 2018.
- [40] Peter Benner, Serkan Gugercin, and Karen Willcox. A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems. *SIAM Review*, 57(4):483–531, January 2015.
- [41] Karl Kunisch and Stefan Volkwein. Optimal snapshot location for computing POD basis functions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(3):509–529, May 2010.
- [42] J. H. Wilkinson. The QR Algorithm for Real Symmetric Matrices with Multiple Eigenvalues. *The Computer Journal*, 8(1):85–87, April 1965.
- [43] F. J. Corbató. On the Coding of Jacobi’s Method for Computing Eigenvalues and Eigenvectors of Real Symmetric Matrices. *Journal of the ACM*, 10(2):123–125, April 1963.
- [44] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1 edition, 1997.
- [45] Joan Baiges, Ramon Codina, and Sergio Idelsohn. Explicit reduced-order models for the stabilized finite element approximation of the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 72(12):1219–1243, August 2013.
- [46] G. W. Stewart. *Matrix Algorithms: Volume II: Eigensystems*. Society for Industrial and Applied Mathematics, January 2001.
- [47] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley Series in Probability and Statistics. Wiley, Hoboken (N.J.), 3rd ed edition, 2019.
- [48] C. Radhakrishna Rao. Separation theorems for singular values of matrices and their applications in multivariate analysis. *Journal of Multivariate Analysis*, 9(3):362–377, September 1979.
- [49] Charles R. Johnson and Roger A. Horn. Unitary equivalence and normal matrices. In *Matrix Analysis*, pages 65–118. Cambridge University Press, Cambridge, 1985.
- [50] B. Sanderse. INS2D, 2018.

Exploiting repeated matrix block structure to accelerate the numerical solution of the incompressible Navier-Stokes equations

“Science may be regarded as a maximum yielding of thoughts with a minimum of expense.”

– Ernst Mach

In this chapter, a novel approach to accelerate the numerical solution of the incompressible Navier-Stokes equations is presented. This approach is based on the idea of increasing the arithmetic intensity of the computations by exploiting the repeated matrix block structure that appears when ensemble averaging multiple simulations, similarly to that one appearing in domains with symmetries or repeated geometrical patterns. This way, the data of the sparse matrix can be reused across the different simulations, increasing the arithmetic intensity of the computations and leading to significant speed-ups compared to performing the simulations sequentially.

4.1 Introduction

Numerical solutions of the incompressible Navier-Stokes equations are characterized by a few algebraic operations, or their stencil-based counterparts, that are independent of

the spatial discretization used (Finite Volume Method (FVM), Finite Element Method (FEM), Discontinuous Galerkin (DG), Finite Difference Method (FDM), etc.): the dot product (dot), the sparse matrix-vector product (SpMV), the linear combination of vectors (axpy), and the elementwise product of vectors (axty). These operations form the building blocks of the methods used to advance the solution in time.

While its computational cost is implementation-dependent, SpMV usually becomes one of the bottlenecks of these simulations, given the big amount of data movement it requires. Despite the computational power suffices, the memory bandwidth of the hardware is not enough to feed the computational units, and the performance of the simulation is limited by the data movement. This is known as being *memory-bound*.

How efficiently the data is used is defined by the roofline model [1], which states that the performance of a kernel is limited by the minimum between the peak performance of the hardware and the product of the arithmetic intensity (I) of the kernel and the memory bandwidth. The arithmetic intensity is defined as the number of floating-point operations performed per byte of data moved. For all algebraic operations, this value is usually low, making it memory-bound. Therefore, addressing this memory-boundedness is crucial to reach the full potential of modern systems. According to the roofline model, increasing the arithmetic intensity of the computations is one of the ways to achieve this. Figure 4.1 depicts a simplified version of the roofline model, where the memory-bound and compute-bound regions are shown. The goal of this chapter is to push the arithmetic intensity of the computations to the right, closer to the compute-bound region, as indicated with the SpMVs being replaced by sparse matrix-matrix products (SpMMs).

Given the definition of I , it is straightforward to consider two ways to increase it: either by increasing the number of floating-point operations performed, or by reducing the amount of data moved. The former is limited by the operation performed; e.g., in a axpy operation, the number of floating-point operations is fixed by the size of the vectors. This is why the latter is the most common approach to increase the arithmetic intensity of a kernel.

With the goal of breaking the strong scaling limits of different implementations, the literature has considered multiple strategies to increase the arithmetic intensity of the computations. In particular, of the SpMV. Greathouse and Daga [2] improved the performance of the SpMV by optimizing the loads of the sparse matrix, which led to speed-ups compared to the original algorithms. In the same line of increasing the arithmetic intensity, Krasnopolsky [3] extended on the work from Makarashvili et al. [4], which proposed exploiting the ergodicity of the flow to perform ensemble-averaged simulations in different devices. Krasnopolsky's approach consists of performing multiple simula-

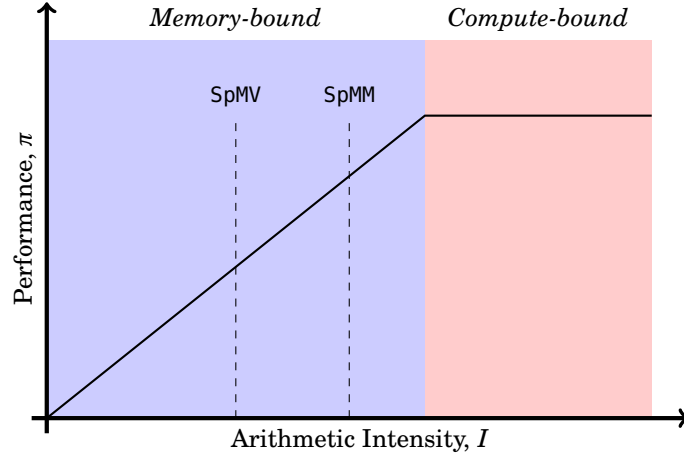


Figure 4.1: Simplified version of a roofline model in which the memory-bound (blue) and the compute-bound (red) regions are depicted. The goal of this chapter is represented in pushing the arithmetic intensity of the computations to the right, closer to the compute-bound region, by replacing all SpMV operations by SpMM.

tions of the same flow, with perturbed initial conditions, in the same device. This way, the data of the sparse matrix can be reused across the different simulations, increasing the arithmetic intensity of the computations. The results showed that this approach can lead to significant speed-ups compared to performing the simulations sequentially. This is performed by rearranging the m flow states into a dense matrix with m columns, and performing the computations on this dense matrix, leading to a SpMM instead of a SpMV. The use of this SpMM is the key to increase the arithmetic intensity of the computations, as it allows to reuse the data of the sparse matrix across the different simulations. This approach is particularly effective for large-scale simulations, where the cost of data movement is the most significant.

While Krasnopolsky [3] showed the potential of this approach, this generalization was just applied to the numerical solution of the Poisson equation, which is a substep of the overall simulation. Later works by Alsalti-Baldellou et al. [5, 6] extended this approach to the overall simulation in the context of simulations with domains with symmetries or repeated patterns by making use of the block structure of the matrices when properly ordered. These studies led to significant speed-ups compared to the original implementations due to the increase in the arithmetic intensity, as well as a reduction in the memory footprint due to the reduction of the matrices' size, both for

CPU and GPU-accelerated implementations. However, the potential of this approach for general simulations, without the benefits of the reduced memory footprint that appears when exploiting geometrical patterns, has not been explored yet. This is what is going to be explored in this chapter.

4.2 Increasing the arithmetic intensity of SpMVs

According to the roofline model [1], the performance (π) extracted from a kernel is not limited by the peak performance of the hardware. Instead, it is defined as

$$\pi = \min(\pi_{\text{peak}}, I\beta), \quad (4.1)$$

where π_{peak} is the peak performance of the hardware, and β is the memory bandwidth. For sparse algebra operations, such as the SpMV, the arithmetic intensity is usually low, making them memory-bound, i.e., the performance being limited by $I\beta$. Therefore, increasing the arithmetic intensity of these operations is crucial to reach the full potential of modern systems.

Let us consider a sparse matrix which structure reads $A = (I_s \otimes \tilde{A}) \in \mathbb{R}^{sn \times sm}$, where I_s is the identity matrix of size s , $\tilde{A} \in \mathbb{R}^{n \times m}$ is a sparse matrix. Essentially, A is a block diagonal matrix with s blocks of $\tilde{A}$. Thus, given s vectors $\tilde{\mathbf{x}} \in \mathbb{R}^m$, which can be arranged as a dense matrix $\mathbf{x} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_s)$, it is straightforward to see that the product $A\mathbf{x}$ can be computed as a SpMM as

$$A\mathbf{x} = \tilde{A}(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_s). \quad (4.2)$$

While the identity from Eq. (4.2) represents the same exact number of floating-point operations as performing a single SpMV for A , the arithmetic intensity of the SpMM is higher than that of the SpMV due to the reuse of the data of the sparse matrix $\tilde{A}$ across the different vectors $\tilde{\mathbf{x}}_i$. According to the roofline model (see Figure 4.1), this increase in the arithmetic intensity increases the performance of the kernel, as it moves it closer to the compute-bound region, without any additional computational cost.

The speed-up from transforming SpMV into SpMM is theoretically bounded by the ratio of performance of the two kernels,

$$P_{m, \text{SpMM}} = \frac{\pi_{m, \text{SpMM}}}{\pi_{\text{SpMV}}}, \quad (4.3)$$

which, assuming the bandwidth is identical for both kernels, can be expressed as the ratio of the arithmetic intensities. According to [5], for a sparse matrix in CSR format,

the arithmetic intensity of the SpMM, with a sparse matrix of n_c columns, n_r rows, $\text{nnz}(A)$ non-zeros, and a dense matrix with m rhs, can be expressed as

$$I_{\text{SpMM}}(m) = \frac{(2\text{nnz}(A) + 1)m}{8\text{nnz}(A) + 4\text{nnz}(A) + 4(n_r + 1) + 8m(n_r + n_c + 1)}, \quad (4.4)$$

while the arithmetic intensity of the SpMV can be expressed as $I_{\text{SpMV}} = I_{\text{SpMM}}(1)$. Thus, the upper-bound of the speed-up from transforming SpMV into SpMM can be expressed as

$$P_{m,\text{SpMM}}^{ub} = \frac{I_{\text{SpMM}}(m)}{I_{\text{SpMV}}} = \frac{m[3(\text{nnz}(A) + n_r + 1) + 2n_c]}{3\text{nnz}(A) + (1 + 2m)(n_r + 1) + 2mn_c}. \quad (4.5)$$

On the other hand, the lower-bound of the speed-up is computed assuming zero temporal locality when accessing the input vector coefficients, which is set by replacing $8n_c m$ with $8\text{nnz}(A)m$ in the denominator of Eq. (4.4), leading to

$$P_{m,\text{SpMM}}^{lb} = \frac{m[5\text{nnz}(A) + 3(n_r + 1)]}{(3 + 2m)\text{nnz}(A) + (1 + 2m)(n_r + 1)}. \quad (4.6)$$

Figure 4.2 depicts an example with 7 and 13 non-zeros per row in which it can be observed that denser matrices, which are generally related to higher-order discretization schemes, provide improved potential speed-ups in the SpMM as the amount of data transferred with the matrix is bigger. Therefore, the increase in the arithmetic intensity is more significant, leading to a higher potential speed-up. In general, the speed-up from transforming SpMV into SpMM increases with the number of rhs (m) and the density of the matrix (nnz/row), as both factors contribute to increasing the arithmetic intensity of the computations. The lower-bound, however, presents marginal differences with different number of non-zeros per row, as it assumes zero temporal locality when accessing the input vector coefficients, which is not affected by the density of the matrix.

While this repeated matrix block structure naturally appears in simulations with domains with symmetries or repeated patterns with a proper mesh ordering [5, 6], it can be artificially introduced in general simulations by performing multiple simulations of the same flow, without the benefits of the reduced memory footprint that appears when exploiting these geometrical patterns [5]. This can be forced in multiple ways. On the one hand, a *multiple parameter* simulation, in which a set of parameters of the simulation are varied across the different simulations, can be performed. This is useful for sensitivity analysis or uncertainty quantification studies that do not require changing the mesh or the domain of the simulation. On the other hand, if a turbulent

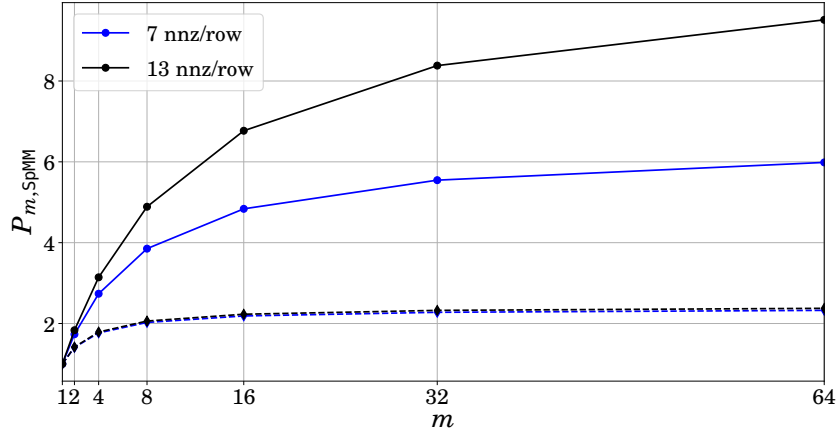


Figure 4.2: Upper-bound (continuous) and lower-bound (dashed) of the speed-up from transforming SpMV into SpMM as a function of the number of rhs (m) for a sparse matrix with $n_c = n_r = 10^6$ for 7 nnz/row (blue) and 13 nnz/row (black) for up to 64 rhs.

flow presents a statistically steady-state, the ergodicity of the flow can be exploited to perform *ensemble-averaged* simulations, where multiple flow states are simultaneously simulated with slightly perturbed initial conditions. Both ways allow to artificially introduce the repeated matrix block structure in the computations, leading to an increase in the arithmetic intensity of the computations and, therefore, to a potential speed-up compared to performing the simulations sequentially.

4.2.1 Extension to the Poisson equation

While the use of SpMV is present in multiple steps of the numerical solution of the incompressible flows (e.g., the advection term, the diffusion term, etc.), the most computationally expensive step is usually the solution of the Poisson equation for the pressure. This is due to the fact that it requires solving a linear system of equations, which is usually done by iterative methods that require multiple SpMV operations. Therefore, the numerical solution of the Poisson equation is going to be the most benefitted from the increase in the arithmetic intensity of the computations. This is why an estimate of the speed-up for this operation is crucial to understand the potential benefits of this approach.

Namely, by definition, the speed-up in the Poisson equation solution given m rhs is

$$P_{m,\text{Poisson}} = \frac{mT_{\text{Poisson},1}}{T_{\text{Poisson},m}}, \quad (4.7)$$

where $T_{\text{Poisson},1}$ is the time to solve the Poisson equation for a single rhs, and $T_{\text{Poisson},m}$ is the time to solve the Poisson equation for m rhs. This wall-clock time may be split into two different contributions. First of all, the contribution that benefits of the increased arithmetic intensity, which corresponds to the time spent in the SpMV/SpMM operations ($T_{\text{Poisson}}^{\text{SpMV}}$ and $T_{\text{Poisson},m}^{\text{SpMM}}$, respectively). Second, the contribution that does not benefit of the increased arithmetic intensity, which corresponds to the time spent in the rest of the operations of the Poisson equation solution ($T_{\text{other},1}$ and $T_{\text{other},m}$, respectively). These other terms are assumed to grow linearly with m , as they are not affected by the increase in the arithmetic intensity. Thus, $T_{\text{other},m} = mT_{\text{other},1}$. Let χ be the fraction of the wall-clock time for the 1-rhs case devoted to SpMV. Thus, the wall-clock time for the 1-rhs case is expressed as

$$T_{\text{Poisson},1} = T_{\text{Poisson},1}^{\text{SpMV}} + \underbrace{T_{\text{other},1}}_{(1-\chi)T_{\text{Poisson},1}} = \frac{1}{\chi} T_{\text{Poisson},1}^{\text{SpMV}}. \quad (4.8)$$

On the other hand, the wall-clock time for the m -rhs case is expressed as

$$T_{\text{Poisson},m} = T_{\text{Poisson},m}^{\text{SpMM}} + \underbrace{T_{\text{other},m}}_{m(1-\chi)T_{\text{Poisson},1}} = T_{\text{Poisson},m}^{\text{SpMM}} + \frac{m(1-\chi)}{\chi} T_{\text{Poisson},1}^{\text{SpMV}}. \quad (4.9)$$

Introducing Eqs.(4.8)-(4.9) into Eq.(4.7), the speed-up in the Poisson equation solution can be expressed as

$$P_{m,\text{Poisson}} = \frac{P_{m,\text{SpMM}}}{\chi + (1-\chi)P_{m,\text{SpMM}}}. \quad (4.10)$$

Figure 4.3 depicts the upper-bound and lower-bound of the speed-up in the numerical solution of the Poisson equation as a function of the number of rhs (m) for a sparse matrix with $n_c = n_r = 10^6$ and 7 non-zeros per row, which represents the classical second-order Laplacian operator, for $\chi = 0.8$ and $\chi = 0.5$ for up to 64 rhs. It can be observed that the speed-up in the Poisson equation solution is significantly lower than the speed-up in the SpMM compared to the SpMV, as it is limited by the contribution of the other operations that do not benefit of the increase in the arithmetic intensity. However, it can also be

observed that, for $\chi = 0.8$, which represents a case where 80% of the wall-clock time is devoted to the SpMV operations, a speed-up of around 3 can be achieved for 64 rhs, while for $\chi = 0.5$, which represents a case where 50% of the wall-clock time is devoted to the SpMV operations, a speed-up of around 1.8 can be achieved for 64 rhs, which corresponds to less than the lower-bound for $\chi = 0.8$. This highlights the importance of the contribution of the SpMV operations in the overall performance of the Poisson equation solution, as well as the potential benefits of increasing the arithmetic intensity of these operations by transforming them into SpMM.

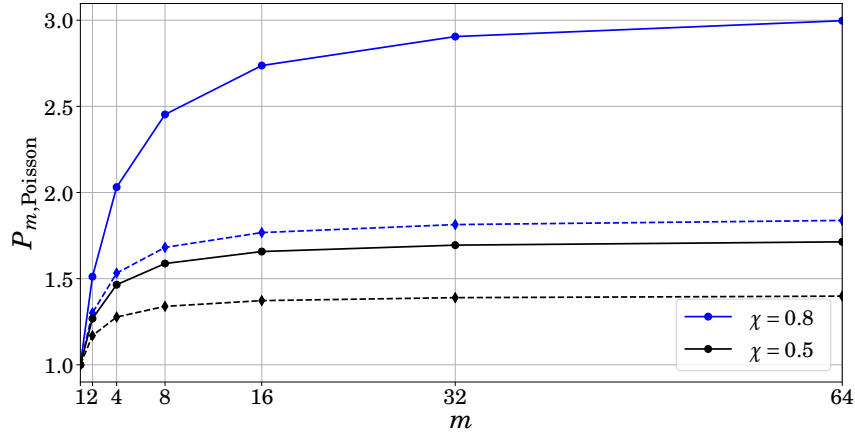


Figure 4.3: Upper-bound (continuous) and lower-bound (dashed) of the speed-up induced in the numerical solution of the Poisson equation by transforming SpMV into SpMM as a function of the number of rhs (m) for a sparse matrix with $n_c = n_r = 10^6$ and 7 non-zeros per row for $\chi = 0.8$ (blue) and $\chi = 0.5$ (black) for up to 64 rhs.

These results also highlight the fact that the use of SpMV-heavy preconditioners such as multigrid methods, which are commonly used in the numerical solution of the Poisson equation, can significantly increase the potential benefits of this approach, as they increase the contribution of the SpMV operations in the overall performance of the Poisson equation solution. Therefore, the use of SpMV-heavy preconditioners is crucial to achieve significant speed-ups by transforming SpMV into SpMM.

4.2.2 Parallel-in-time ensemble-averaged simulations

Numerical simulations of turbulent flows, if the boundary conditions are not time-dependent, usually present a characteristic structure: a transition phase followed by a statistically steady-state phase. During the transition phase, the flow evolves from the initial conditions to a statistically steady-state, which is usually characterized by a significant change in the flow properties. On the other hand, during the statistically steady-state phase, statistics are collected to characterize the flow properties. These statistics are usually collected by performing time-averaging of the flow properties, which requires a long simulation time to achieve convergence. This is why the transition phase is usually not considered in the analysis of the flow properties, as it does not contribute to the statistics collected during the statistically steady-state phase.

However, the statistically steady-state phase is ergodic, i.e., the time-averaged statistics are equal to the ensemble-averaged statistics. This means that, instead of performing a single long simulation to collect the statistics, multiple shorter simulations with slightly perturbed initial conditions can be performed simultaneously, and the statistics can be collected by performing ensemble-averaging across these simulations provided the different flow states are statistically independent. This approach is known as parallel-in-time ensemble-averaged simulations, and it allows to significantly reduce the time required to collect the statistics of the flow properties, as it allows to perform multiple simulations simultaneously.

Let the transition time be defined as T_T , and the averaging time be defined as T_A . Thus, the time required to perform a single simulation to collect the statistics is $T = T_T + T_A$. On the other hand, if m simulations are performed simultaneously, the time required to collect the statistics is $T' = T_T + T_A/m$, as the averaging time is reduced by a factor of m due to the ensemble-averaging, ensuring that T_A/m is larger than a single time unit [3]. Moreover, according to Makarashvili et al. [4], for the ergodic ensemble to yield the same statistics as the time-averaged one, the different ensembles should be statistically independent, which can be achieved by perturbing the initial conditions of the simulations. Given these conditions, for a relevant statistic α , the time-averaged statistic $\langle \alpha \rangle_T$ and the ensemble-averaged statistic $\langle \alpha \rangle_E$ are defined as

$$\langle \alpha \rangle_T = \frac{1}{T_A} \int_{T_T}^{T_T+T_A} \alpha(t) dt, \quad (4.11)$$

$$\langle \alpha \rangle_E = \frac{1}{m} \sum_{i=1}^m \langle \alpha_i \rangle_T = \frac{1}{m} \sum_{i=1}^m \frac{1}{T_A/m} \int_{T_T}^{T_T+T_A/m} \alpha_i(t) dt, \quad (4.12)$$

where α_i is the statistic α for the i -th simulation. Thus, if the different simulations are statistically independent, the ensemble-averaged statistic is equal to the time-averaged one, i.e., $\langle \alpha \rangle_E = \langle \alpha \rangle_T$. This means that, by performing multiple simulations simultaneously with slightly perturbed initial conditions, the same statistics can be collected in a shorter time compared to performing a single long simulation.

This situation leads to identical divergence, gradient and Laplacian operators across the different simulations, which will lead to the repeated matrix block structure described in the previous section, allowing to perform the computations on a dense matrix with m columns, leading to a SpMM instead of a SpMV, and therefore increasing the arithmetic intensity of the computations.

Applying Eqs.(2.1)-(2.2) in a collocated framework without time-dependent boundary conditions, following the notation from [7], yields

$$M\mathbf{u}_s = \mathbf{0}_c, \quad (4.13)$$

$$\Omega \frac{d\mathbf{u}_c}{dt} + C(\mathbf{u}_s)\mathbf{u}_c = -\Omega G_c \mathbf{p}_c + D\mathbf{u}_c + \Omega \mathbf{f}_c, \quad (4.14)$$

where $\mathbf{u}_c, \mathbf{p}_c$ are the velocity and pressure fields defined at the cell center, respectively; G_c is the discrete cell-to-cell gradient operator, $\mathbf{u}_s = \Gamma_{c \rightarrow s} \mathbf{u}_c$ is the velocity defined at the cell faces, where $\Gamma_{c \rightarrow s}$ is the cell-to-face interpolation operator, and $C(\mathbf{u}_s) = M \text{diag}(\mathbf{u}_s) \Gamma_{c \rightarrow s}$ [8]. The rest of the operators are defined as in Eqs.(2.1)-(2.2), and the time-dependent terms have been dropped for simplicity.

In the framework with m simultaneous flow states, Eqs.(4.13)-(4.14) can be expressed as

$$\tilde{M}\mathbf{U}_s = \tilde{\mathbf{0}}_c, \quad (4.15)$$

$$\tilde{\Omega} \frac{d\mathbf{U}_c}{dt} + \tilde{M}\tilde{\mathbf{U}}_s \tilde{\Gamma} \mathbf{U}_c = -\tilde{\Omega} \tilde{G}_c \mathbf{P}_c + \tilde{D}\mathbf{U}_c + \tilde{\Omega} \mathbf{F}_c, \quad (4.16)$$

where $\mathbf{U}_c = (\mathbf{u}_{c,1}, \mathbf{u}_{c,2}, \dots, \mathbf{u}_{c,m})$ and $\mathbf{P}_c = (\mathbf{p}_{c,1}, \mathbf{p}_{c,2}, \dots, \mathbf{p}_{c,m})$ are the velocity and pressure fields defined at the cell center for the m flow states, respectively; $\tilde{M} = I_m \otimes M$, $\tilde{\Omega} = I_m \otimes \Omega$, $\tilde{D} = I_m \otimes D$, $\tilde{G}_c = I_m \otimes G_c$, and $\tilde{\mathbf{0}}_c = (\mathbf{0}_c, \mathbf{0}_c, \dots, \mathbf{0}_c)$; and $\tilde{\Gamma}$ is a block diagonal matrix with m blocks of $\Gamma_{c \rightarrow s}$. Moreover, $\tilde{\mathbf{U}}_s$ is the block diagonal matrix containing the velocity defined at the cell faces for the m flow states, which is computed by applying the cell-to-face interpolation operator to the velocity defined at the cell centers for each flow state. These definitions end up in a block structure of the matrices, which allows to perform the computations on a dense matrix with m columns, leading to a SpMM instead of a SpMV,

and therefore increasing the arithmetic intensity of the computations.

The speed-up obtained in an iteration by using the improved SpMM compared to using m SpMV is defined as

$$P_{m,\text{iter}} = \frac{mT_{\text{iter},1}}{T_{\text{iter},m}}, \quad (4.17)$$

where $T_{\text{iter},1}$ is the wall-clock time per iteration for a single flow state, and $T_{\text{iter},m}$ is its counterpart for m flow states. Hence, if the same split and assumptions as those for the Poisson equation speed-up are performed, the wall-clock time per iteration for the m flow states case is $T_{\text{iter},m} = T_{\text{iter},m}^{\text{SpMM}} + mT_{\text{iter},1}^L$, where the superscript L denotes the contributions that are assumed to grow linearly with m and do not benefit of the increase in the arithmetic intensity. If $\Theta = T_{\text{iter},1}^{\text{SpMV}}/T_{\text{iter},1}$ is defined as the weight of the SpMV operations in the wall-clock time per iteration for the 1-flow state case, the speed-up per iteration can be expressed as

$$P_{m,\text{iter}} = \frac{1}{1 - \Theta + \Theta P_{m,\text{SpMM}}^{-1}}. \quad (4.18)$$

Obtention of uncorrelated flow states The methodology relies on the fact that the different flow states are uncorrelated. According to the works of Nikitin [9], the growth of disturbances in turbulent flows is generally a wall effect, and it is defined as

$$\varepsilon^2(t) = \frac{1}{V} \int_V |\Delta \mathbf{u}|^2 dV, \quad (4.19)$$

where V is the domain volume and $\Delta \mathbf{u}$ is the vector of disturbances, which is defined as the difference between the velocity field of a flow state and the velocity field of a reference flow state. This growth is exponential in time until approximately $t^+ = 10^3$, being t^+ the elapsed time normalized with the viscous time scale $t_\tau = l_\tau/u_\tau$, where l_τ and u_τ are the viscous length and velocity scales, respectively. After this time, disturbances remain constant at approximately $\varepsilon \approx 10^{-1}$. This exponential growth corresponds to the classical behaviour of chaotic systems ruled by the Lyapunov exponent [10]. According to Badii et al. [11], the correlation among different flow states will decay exponentially with the generalized Lyapunov exponent for 1D cases, which indicates that eventually the flow states will become uncorrelated. Later on, Mendes et al. [12] extended these results to 2D cases.

Even if perturbations lead to uncorrelated flow states, the average solution is expected to behave similarly given the existence of a weak global attractor [13, 14].

If the solutions in the weak global attractor are strongly continuous with respect to the initial conditions, the different flow states will converge to the same solution [13]. Therefore, even if initial conditions are perturbed and solutions become uncorrelated, all will lead to the same average solution, which is the one that is expected to be obtained by performing a single long simulation, given the ergodicity of the flow.

4.2.3 Methods to accelerate the transition phase

As defined in Eq.(4.12), every flow state will be integrated for $T_T + T_A/m$ time units, which means that the transition phase will be performed m times. In order to quantify how relevant this transition phase is, Krasnopolsky [3] defined the *times ratio* as

$$\beta = \frac{T_T}{T_A}, \quad (4.20)$$

which represents the ratio between the transition time and the averaging time. For a given β , the speed-up from performing parallel-in-time ensemble-averaged simulations compared to performing a single long simulation can be expressed as

$$P_m = \frac{1 + \beta}{m + \beta} \frac{5m}{5m - 3\theta(m - 1)}, \quad (4.21)$$

where θ is the weight of the Poisson equation in the whole iteration. The speed-up calculation assumes the wall-clock time per iteration is constant throughout the whole simulation.

Krasnopolsky [15] proposed a strategy to reduce the time required to perform the transition phase. In this strategy, a single flow state was first run until the flow is developed, which is cheaper to compute than actually running m simultaneous flow states. Later on, this flow state was used as the initial condition for m simultaneous flow states, to which a perturbation was applied to ensure that they become uncorrelated [9]. This strategy allows to significantly reduce the time required to perform the transition phase, as it is performed in a cheaper way than actually running m simultaneous flow states. However, this strategy does not allow to reduce the time required to perform the transition phase, as it actually requires to perform the transition phase for a single flow state in the objective mesh, which is not negligible for cases with low β . Therefore, a strategy that allows to reduce the time required to perform the transition phase is crucial to achieve significant speed-ups for cases with low β . This could be circumvented by performing the transition phase in a cheaper way than actually running m simultaneous

flow states in the objective mesh, which is the strategy proposed in this thesis.

Assume the simulation is started with a cheaper procedure than the actual simulation, e.g., with a coarser mesh. After the flow has developed and the simulations are expected to be independent in the coarse mesh (T_D time units), the mesh is refined to the objective mesh, and the flow is developed until the actual transition time (T_T time units), after which the statistics are collected for T_A/m time units, similarly to Kransopolsky [3]. Figure 4.4 depicts this strategy, which allows to significantly reduce the time required to perform the transition phase, as it is performed in a coarser mesh, which is cheaper to compute. This strategy is particularly effective for cases with low β , as it allows to significantly reduce the time required to perform the transition phase, which is a significant part of the total simulation time for these cases.

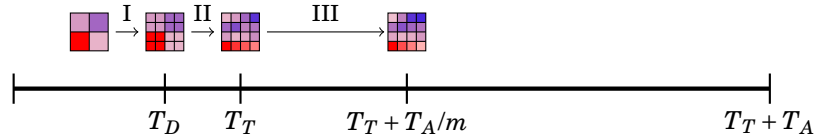


Figure 4.4: Proposed ensemble averaging strategy in which the case is run until T_D in a coarser mesh, after which the mesh is refined and the case is run until T_T to develop the flow in the objective mesh, after which the statistics are collected for T_A/m time units. The following phases are defined: *I*: mapping, *II*: developing, *III*: averaging.

Assume that the average timestep as well as the average wall-clock time per iteration are the same for both phases *II* and *III* (Figure 4.4), and are defined $\overline{\Delta t}_m^F$ and $\overline{T}_{\text{iter},m}^F$, respectively, where F stands for the fine mesh. Similarly, $\overline{\Delta t}_m^C$ and $\overline{T}_{\text{iter},m}^C$ are defined as the average timestep and the average wall-clock time per iteration for the coarse mesh, respectively. Hence, for an arbitrary case with m flow states, let T_m be the wall-clock time required to perform the whole simulation, which reads

$$T_m = N_D T_D + N_{DT} T_{DT} + \frac{N_A}{m} T_A = \frac{\overline{T}_{\text{iter},m}^C}{\overline{\Delta t}_m^C} T_D + \frac{\overline{T}_{\text{iter},m}^F}{\overline{\Delta t}_m^F} (T_{DT} + T_A/m), \quad (4.22)$$

where $T_{DT} = T_T - T_D$ is the time required to perform the transition phase in the fine mesh, and N_P is the number of iterations required for any given phase $P = \{D, DT, A\}$. Now, let $T_D = \gamma T_{DT}$, and $f_m^X = \overline{T}_{\text{iter},m}^X / \overline{\Delta t}_m^X$ be defined as the cost per time unit for the mesh $X = \{F, C\}$. Thus, the wall-clock time required to perform the whole simulation

can be expressed as

$$T_m = \frac{T_T}{1+\gamma} \left[f_m^C \gamma + f_m^F \left(1 + \frac{1+\gamma}{m} \beta \right) \right]. \quad (4.23)$$

By definition, the speed-up from performing the whole simulation with the proposed strategy compared to performing a single long simulation is defined as T_1/T_m , which is expressed as

$$P_m = \frac{1 + \delta_1 \gamma + (1+\gamma)\beta}{m(\delta_m \gamma + 1) + (1+\gamma)\beta} \frac{m f_1^F}{f_m^F}, \quad (4.24)$$

where $f_m^C = \delta_m f_m^F$ has been defined as the cost per time unit for the coarse mesh, which is expressed as a fraction of the cost per time unit for the fine mesh. If $\gamma = 0$, Eq. (4.24) reduces to the single mesh speed-up from Krasnopolsky [3].

Defining the *effective times ratio* as $\tilde{\beta} = \frac{1+\gamma}{1+\delta_1 \gamma} \beta$, and the *rhs scale-up* as $\Phi = \frac{1+\delta_m \gamma}{1+\delta_1 \gamma}$, and assuming $\overline{\Delta t}_m^F = \overline{\Delta t}_1^F$, Eq. (4.24) can be expressed as

$$P_m = \frac{1 + \tilde{\beta}}{\Phi m + \tilde{\beta}} P_{m, \text{iter}}^F, \quad (4.25)$$

which is similar to the single mesh speed-up from Krasnopolsky [3], but with an effective times ratio and a rhs scale-up that take into account the use of the coarse mesh for the transition phase. This expression highlights the fact that, by using a cheaper procedure for the transition phase, the effective times ratio is increased, which leads to an increase in the speed-up from performing parallel-in-time ensemble-averaged simulations compared to performing a single long simulation. Eq.(4.25) is characterized by three parameters: $P_{m, \text{iter}}^F$, Φ , and δ_m . Note that δ_m^{-1} indicates how faster the coarse mesh is in advancing in time compared to the fine mesh, considering both the bigger timestep that preserves the stability of the integration and the faster iterations due to a reduced problem size. Hence, this factor will be strongly related to the refinement factor of the mesh. It can be assumed that $\delta_m^{-1} \approx r_f^4$ for a 3D case, where r_f is the refinement factor of the mesh, as the cost per time unit is expected to grow with the number of cells in the mesh, which grows with r_f^3 , and with the inverse of the timestep, which grows with r_f .

Let $\Pi = \delta_1^{-1}$ be defined as the speed-up of the transition phase in the coarse mesh compared to the fine mesh. Therefore, the effective times ratio can be rewritten as $\tilde{\beta} = \frac{\Pi(1+\gamma)}{\gamma+\Pi} \beta$. Figure 4.5 depicts the ratio between the effective times ratio and the original times ratio as a function of the transition phase speed-up for the coarse mesh (Π) for different values of γ .

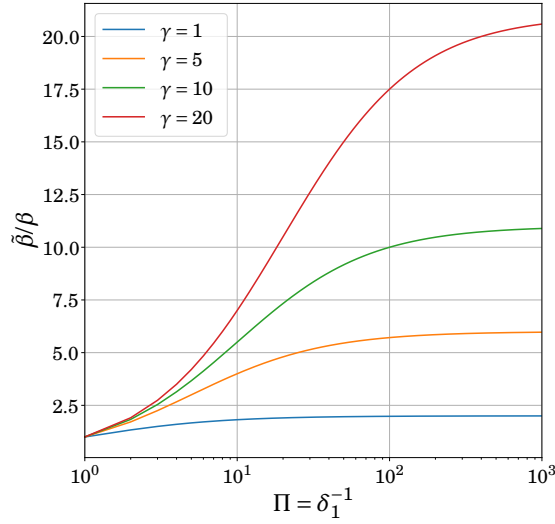


Figure 4.5: Ratio between $\tilde{\beta}$ and β as a function of the transition phase speed-up for the coarse mesh (Π) for different values of γ .

It can be observed that, for any given γ , the effective times ratio is increased compared to the original times ratio β for any $\Pi > 1$, which means that the use of a cheaper procedure for the transition phase leads to an increase in the speed-up from performing parallel-in-time ensemble-averaged simulations compared to performing a single long simulation. Moreover, the increase in the effective times ratio is more significant for lower values of γ , which means that the use of a cheaper procedure for the transition phase is particularly effective for cases with low β , as it allows to significantly reduce the time required to perform the transition phase, which is a significant part of the total simulation time for these cases. On the other hand, the selection of γ generates a trade-off for the user, as a smaller value will ensure bigger convergence to a statistically steady-state before the start of the averaging phase, but it will also lead to a smaller increase in the effective times ratio, and therefore to a smaller speed-up from performing parallel-in-time ensemble-averaged simulations compared to performing a single long simulation.

Introducing the definition of the iteration speed-up for the fine mesh $P_{m,iter}^F$ into Eq.

(4.25) yields

$$P_m = \frac{1 + \tilde{\beta}}{\Phi m + \tilde{\beta}} \frac{1}{1 - \Theta + \Theta(P_{m, \text{SpMM}}^F)^{-1}}, \quad (4.26)$$

which relevant parameters are defined in Table 4.1.

Table 4.1: Summary of the relevant parameters for the speed-ups and the parallel-in-time ensemble-averaged simulations.

Speed-ups and times		Parallel-in-time parameters	
Parameter	Definition	Parameter	Definition
Poisson speed-up, $P_{m, \text{Poisson}}$	Eq. (4.7)	Times ratio, β	T_A/T_T
Iteration speed-up, $P_{m, \text{iter}}$	Eq. (4.18)	Transition ratio, γ	T_D/T_{DT}
Simulation speed-up, P_m	Eq. (4.25)	Simulation pace, f_m	$\overline{T}_m/\overline{\Delta t}_m$
Coarse transition time, T_D	Fig. 4.4	Pace ratio, δ_m	f_m^C/f_m^F
Fine transition time, T_{DT}	Fig. 4.4	Transition speed-up, Π	δ_1^{-1}
Transition time, T_T	Fig. 4.4	Effective times ratio, $\tilde{\beta}$	$\frac{\Pi(1+\gamma)}{\gamma+\Pi} \beta$
Averaging time, T_A	Fig. 4.4	RHS scale-up, Φ	$\frac{1+\delta_m\gamma}{1+\delta_1\gamma}$

A *priori* estimations on the overall speed-up of the methods, it is assumed that $\Phi = 1$ for all m . This hypothesis indicates that the effect of the mesh refinement does not depend on m ($\delta_1 = \delta_m$, $\forall m$). This assumption is validated experimentally in Section 4.3. With these results and using Eq.(4.26), the speed-up from performing parallel-in-time ensemble-averaged simulations compared to performing a single long simulation can be bounded so that the performance obtained by the implementation should lay between those bounds.

To exemplify the effects in terms of speed-up, it will be assumed that $\Theta = 0.65$, i.e., 65% of the wall-clock time per iteration for the 1-flow state case is devoted to performing SpMV operations, which is a reasonable assumption for an incompressible flow solver given the dependence on Krylov-subspace methods for the solution of the Poisson equation, which are SpMV-heavy. Assuming $r_f = 2$, which is a common refinement factor for mesh refinement, $\delta_m^{-1} \approx 16$ is obtained, which means that the coarse mesh is 16 times faster in advancing in time compared to the fine mesh. Moreover, $\gamma = 5$ is assumed, which means that the transition phase in the coarse mesh is 5 times longer than the transition phase in the fine mesh. Considering a discretization with 7 non-zeros per

row, the upper-bound of the speed-up is presented in Table 4.2 for different values of β and m . Note that these correspond to the worst-case scenario in terms of discretization, as a denser matrix will provide better results. However, it can be observed that, even for this worst-case scenario, significant speed-ups can be achieved for low values of β ($\beta = 1$ and $\beta = 2$ have speed-ups of 14% and 26%, respectively), while for higher values of β , the speed-ups are even better, even though it is expected given the fact that the transition phase is a smaller part of the total simulation time for higher values of β . Moreover, the best speed-up for each value of β is achieved for different values of m , which highlights the importance of selecting the optimal number of flow states to perform the parallel-in-time ensemble-averaged simulations, as it can significantly affect the overall speed-up achieved by this approach.

Table 4.2: Estimated upper-bound of the speed-up from performing parallel-in-time ensemble-averaged simulations compared to performing a single long simulation for different values of β and m , considering $\Theta = 0.65$, $\delta_m^{-1} = 16$, and $\gamma = 5$, for a matrix with 7 non-zeros per row. The best speed-up for each value of β is highlighted in bold.

m		β					
		1	2	5	10	20	40
16		0.52	0.78	1.18	1.46	1.66	1.78
8		0.81	1.07	1.40	1.58	1.69	1.75
4		1.06	1.26	1.44	1.53	1.58	1.60
2		1.14	1.22	1.29	1.31	1.33	1.34
1		1.00	1.00	1.00	1.00	1.00	1.00

A more favorable condition has been tested by considering a denser matrix with 27 non-zeros per row, which represents a discretization which includes not only the first neighbors normal to the faces but also the first neighbors in the diagonal, which is a common discretization for the Laplacian operator. The results are presented in Table 4.3, which shows that the speed-ups are significantly larger compared to the case with 7 non-zeros per row, as the increase in the arithmetic intensity is more significant for a denser matrix. Leveraging this denser matrix leads to speed-ups exceeding 100% for the most favorable case ($\beta = 40$ and $m = 16$), which highlights the potential of this approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation, especially for cases with high β and for denser matrices.

Table 4.3: Estimated upper-bound of the speed-up from performing parallel-in-time ensemble-averaged simulations compared to performing a single long simulation for different values of β and m , considering $\Theta = 0.65$, $\delta_m^{-1} = 16$, and $\gamma = 5$, for a matrix with 27 non-zeros per row. The best speed-up for each value of β is highlighted in bold.

m		β					
		1	2	5	10	20	40
16		0.63	0.93	1.42	1.75	1.99	2.14
8		0.94	1.25	1.64	1.85	1.98	2.05
4		1.19	1.41	1.62	1.72	1.77	1.80
2		1.22	1.30	1.37	1.40	1.42	1.42
1		1.00	1.00	1.00	1.00	1.00	1.00

4.2.4 Multiple parameter simulations

Parametric studies such as the numerical computation of $C_L - \alpha$ curve on an airfoil, or the effect of different thermal conductivities in a wall, usually require a number of simulations to be run independently, i.e., the user launches multiple simulations with a given angle of attack α , or with different thermal conductivities λ , respectively; in order to compute the relevant parameters and generate the curves.

In general, let a flow simulation in which an arbitrary parameter ξ is varied to perform a parametric study, $\xi = \{\xi_1, \xi_2, \dots, \xi_m\}$, where ξ_i is the value of the parameter ξ for the i -th simulation. If the different simulations share the same mesh and the same operators, which is usually the case for these parametric studies, the computations can be performed on a dense matrix with m columns, leading to a SpMM instead of a SpMV, and therefore increasing the arithmetic intensity of the computations.

This approach is similar to the one described for the parallel-in-time ensemble-averaged simulations, but instead of performing multiple simulations with different initial conditions, multiple simulations with different parameters are performed simultaneously. This approach allows to significantly reduce the time required to perform parametric studies compared to performing multiple independent simulations, as it allows to leverage the increase in arithmetic intensity provided by the SpMM.

4.3 Numerical experiments

In order to validate the theoretical results presented in the previous sections, the incompressible Navier-Stokes equations are discretized in space following a finite-volume

symmetry-preserving discretization [7, 16], so that the properties of the continuous operators are preserved in the discrete operators, to ensure that no additional numerical errors are introduced by the discretization. The semi-discretized equations are integrated in time using Heun’s third-order Runge-Kutta methods and applied to the incompressible Navier-Stokes as in Sanderse and Koren [17]. Timestep selection will be calculated adaptively based on the AlgEigCD framework developed by Trias et al. [8], as described in Chapter 3.

The method has been implemented in the in-house code Termofluids Algebraic (tfa), in which the calculations are carried out using the ² framework [18], which allows the portability of the code to multiple architectures as well as integrating it with external libraries such as HYPRE [19] or CHRONOS [20], while still having its own solvers and kernels. The runs for these numerical experiments have been performed in the MareNostrum 5 General Purpose Partition (MN5-GPP) supercomputer at the Barcelona Supercomputing Center (BSC). In this case, the numerical solution of the Poisson equation is performed using an adaptive Algebraic Multigrid (aAMG) preconditioned-conjugate gradient method [21] provided by CHRONOS. The configuration used was tailored for Computational Fluid Dynamics (CFD) problems and consisted of a PMIS coarsening [22], an extended+I interpolation [23], and one pre- and post-smoothing steps using a lightweight factored sparse approximate inverse with five nonzeros per row.

The techniques previously presented have been applied to a turbulent planar channel flow, and a Rayleigh-Bénard natural convection case as two examples of academic flows that are relevant given the reproducibility of the results in previous publications and benchmarks, thus leading to a verification of the method, where a parallel-in-time approach is applied. Finally, the method has been applied as well to an industrially relevant case such as an airfoil with both flaps and slats depleted, in a multiple parameter framework, in order to prove the applicability of the method not only to academic cases but also to industrially relevant cases, and to show the potential of the method to significantly reduce the time required to perform parametric studies compared to performing multiple independent simulations.

4.3.1 Channel flow

First, the methodology has been tested in a turbulent planar channel flow with $Re_\tau = 180$. The method has been tested with a 160^3 grid with a hyperbolic tangent refinement in the y -direction with $\gamma = 1.5$, which ensures a theoretical y^+ at the wall of 0.65.

Moreover, the test has been run using both of the proposed frameworks in this paper, with and without mesh refinement. In the case in which the refinement was applied, the refinement factor used was $r_f = 2$, leading to a 80^3 mesh before the mapping is applied.

The simulation was run for 352 half-channel-height time-units, δ/u_τ , with a 32-time-unit transition period, determined from probe data observations in the core of the flow, showing steady-state behaviour had been reached. In the case using the mesh refinement, the coarse mesh has been run for 30 time units before interpolating to the fine mesh. Note that these 32 time units exceed notably the expected time according to the results from Nikitin [9], which sets the time to have independent fields to $t^+ = 10^3$, which corresponds to 5.6 time units given the ratio between the viscous time scale and the half-channel-height time scale. Therefore, the different flow states are expected to be uncorrelated. The statistics have been collected for 320 time units, which means that $\beta = 10$. A CFL (calculated with respect to the maximum stable timestep according to AlgEigCD) of 0.95 has been used, which ensures that the timestep is as big as possible while ensuring stability of the integration.

The simulation has been run in 64 cores of a node of MN5-GPP in order to set a reasonable load of 64k cells per CPU core, so that the cores are loaded enough to perform efficiently. The parallelization strategy used has been the combination of MPI+OpenMP.

Figure 4.6 depicts the mean velocity profile in wall units and the RMS streamwise velocity for the turbulent planar channel flow case for 1, 2, 4, and 8 flow states. It can be observed that the results are consistent across the different number of flow states, which indicates that the different flow states are uncorrelated and that the statistics collected are consistent with each other.

The tested conditions in terms of T_D , T_T , and T_A lead to $\beta = 10$ and $\gamma = 15$, which will be used to verify Eq. (4.26). Table 4.4 summarizes the average time-steps, wall-clock times per iteration, and δ_m for both the coarse and fine meshes and all the simulated flow states. It can be observed that the average time-step is similar for both meshes, which also validates the assumption of $\overline{\Delta t}_m^F = \overline{\Delta t}_1^F$ used in the speed-up estimation. Moreover, the cost per time unit for the coarse mesh is significantly lower than the cost per time unit for the fine mesh, which leads to a δ_m of approximately 6×10^{-2} , which means that the coarse mesh is approximately 16 times faster in advancing in time compared to the fine mesh, as expected from the approximation $\delta_m^{-1} = r_f^4$. This significant difference in the cost per time unit between the coarse and fine meshes highlights the potential of using a cheaper procedure for the transition phase to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation, especially for cases with low β , as it allows

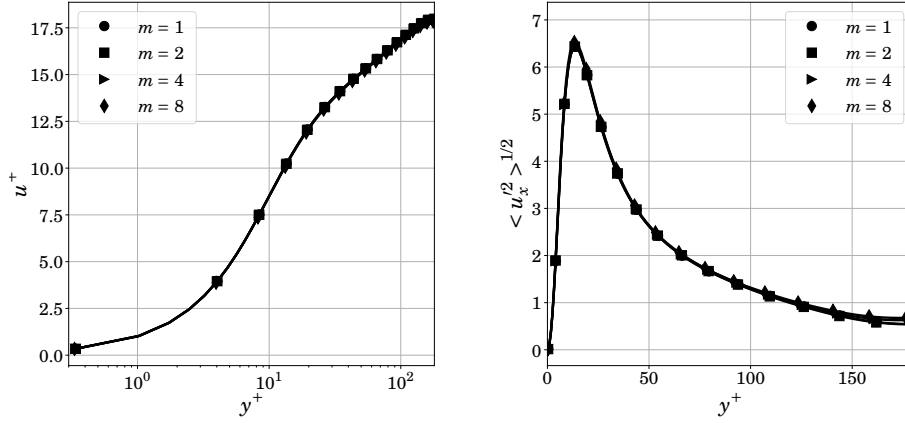


Figure 4.6: Mean velocity profile in wall units (left) and RMS streamwise velocity (right) for the turbulent planar channel flow case for 1, 2, 4, 8 rhs.

to significantly reduce the time required to perform the transition phase, which is a significant part of the total simulation time for these cases.

Table 4.4: Average time-steps, wall-clock times per iteration [s], and δ_m for both the coarse and fine meshes and all the simulated flow states for a turbulent planar channel flow of $\text{Re}_\tau = 180$.

m	$\overline{\Delta t}_m^C$ [s]	$\overline{T}_{\text{iter},m}^C$ [s]	$\overline{\Delta t}_m^F$ [s]	$\overline{T}_{\text{iter},m}^F$ [s]	δ_m (Π)
1	2.66×10^{-3}	0.1167	1.30×10^{-3}	0.9152	0.0624 (16.018)
2	2.538×10^{-3}	0.1651	1.286×10^{-3}	1.3267	0.0630 (15.858)
4	2.438×10^{-3}	0.2554	1.739×10^{-3}	2.2454	0.0812 (12.320)
8	2.340×10^{-3}	0.4795	1.092×10^{-3}	4.2377	0.0528 (18.943)

On the other hand, the speed-ups for the whole simulation given the aforementioned conditions are depicted in Figure 4.7, which shows that the speed-ups for the Poisson equation, for the iteration, and for the whole simulation are consistent with the theoretical results, which validates the method. Moreover, it can be observed that the speed-ups for the Poisson equation are significantly higher than the speed-ups for the iteration and for the whole simulation, which is expected given that the Poisson equation is the most computationally expensive part of the iteration, and therefore benefits more from the increase in arithmetic intensity provided by the SpMM. The speed-ups for the iteration

and for the whole simulation are also significant, which highlights the potential of this approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation, especially for cases with high β and for denser matrices. Moreover, the predicted results given the theoretical speed-up estimations are consistent with the obtained results, which validates the theoretical analysis and highlights the importance of selecting the optimal number of flow states to perform the parallel-in-time ensemble-averaged simulations, as it can significantly affect the overall speed-up achieved by this approach.

Figure 4.7 also shows that the speed-ups for the case with mesh refinement are significantly higher than the speed-ups for the case without mesh refinement, which increased an 11.22%, a 33.02% and 60.01% for $m = 2$, $m = 4$ and $m = 8$, respectively, for the whole simulation. This significant increase in the speed-ups for the case with mesh refinement highlights the potential of using a cheaper procedure for the transition phase to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation.

Moreover, the test case has been run for denser matrices in order to test the method for higher-order space discretizations, which generally have denser sparsity patterns compared to FVM just using the first neighbors. Under these circumstances, the method has been tested in a framework considering only the first neighbor cells, which will have 7 non-zeros per row (7p); considering the first and second neighbours of the cell, which will have 13 non-zeros per row (13p); and considering all neighbors of the cell, including the diagonal neighbors, which will have 27 non-zeros per row (27p). Figure 4.8 shows a two-dimensional representation of the stencil for the 7p, 13p, and 27p discretizations, where the shaded cells represent the first neighbors in all representations, which corresponds to case 7p.

These tests have been performed in a single time unit in the fine 160^3 mesh so that the presented time for the different elements to run has had enough time steps so that the average value is representative. Moreover, a timer has been added to monitor the time devoted to performing SpMM operations in CHRONOS, so that not only the speed-up present in the Poisson solution and the whole iteration is represented, but also the speed-up present in the kernel itself. Given the distribution of sizes and sparsity patterns inherent to aAMG strategies, the presented results as well as the bounds are weighted by the fraction of time of each of these different types of matrices in the whole solution of the Poisson equation, so that the results are representative of the overall speed-up for the solution of the Poisson equation.

Figure 4.9 depicts the speed-ups for the SpMM operations for the 7p, 13p, and 27p

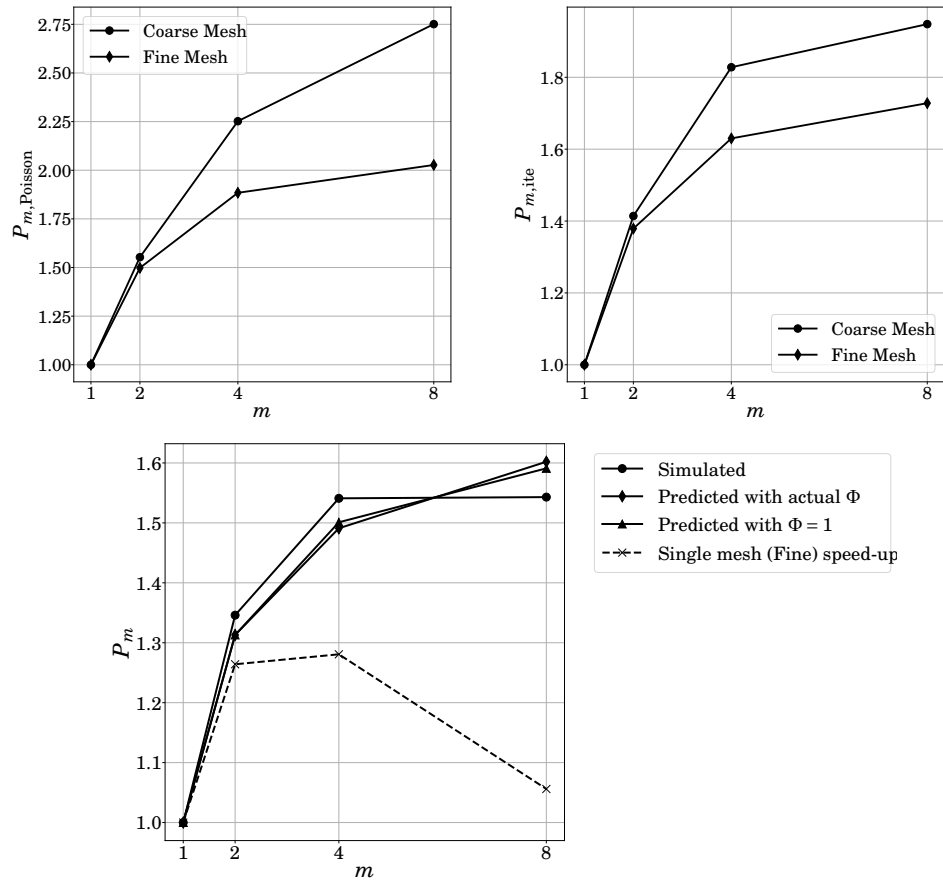


Figure 4.7: Speed-ups for the Poisson equation (top left), for the iteration (top right), and for the whole simulation (bottom) for a turbulent planar channel flow of $Re_\tau = 180$ with mesh refinement.

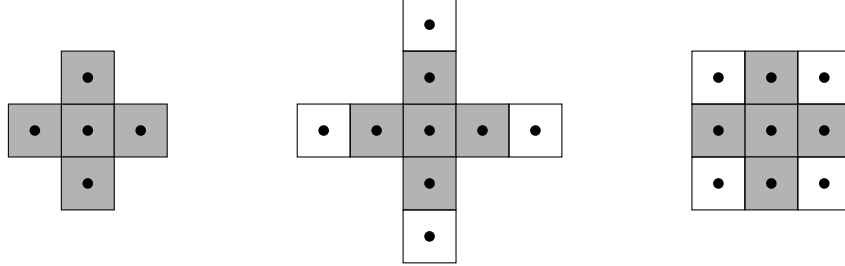


Figure 4.8: Two-dimensional representations of the stencil for 7p (left), 13p (center), and 27p (right) discretizations. The shaded cells represent the first neighbors in all representations.

discretizations in the Poisson equation resolution for a turbulent planar channel flow of $Re_\tau = 180$ with a 160^3 mesh. The dashed lines represent the theoretical bounds for the speed-ups given the theoretical analysis, while the markers represent the obtained results. As expected from the theoretical results, no slow-down is obtained for the tested range of m in the three cases. Moreover, the speed-ups for the SpMM operations are significantly higher for the denser matrices, which is expected given the increase in arithmetic intensity provided by the SpMM. The speed-ups for the 27p discretization are significantly higher than the speed-ups for the 7p and 13p discretizations, which highlights the potential of this approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation, especially for denser matrices.

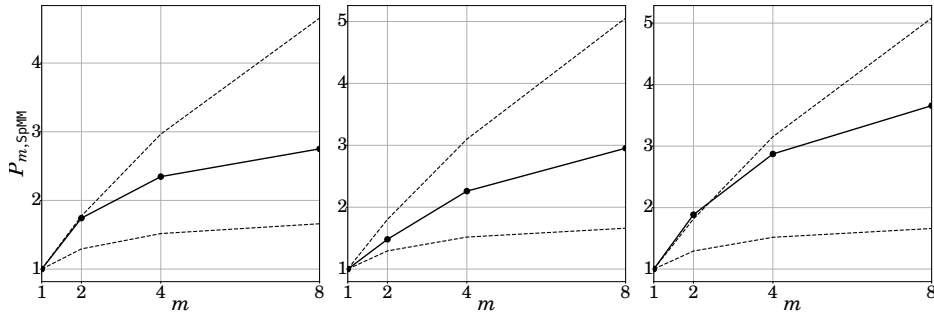


Figure 4.9: Speed-ups for the SpMM operations for the 7p (left), 13p (center), and 27p (right) discretizations in the Poisson equation resolution for a turbulent planar channel flow of $Re_\tau = 180$ with a 160^3 mesh. The dashed lines represent the theoretical bounds for the speed-ups given the theoretical analysis, while the markers represent the obtained results.

The implementation performance has been validated by representing the roofline plot of the execution for the different discretizations, as shown in Figure 4.10, where the arithmetic intensity and the achieved performance for the different discretizations are represented. It can be observed that the achieved performance also increases with the number of non-zeros per row, which is expected given the increase in arithmetic intensity, as it allows to better leverage the computational capabilities of the hardware. The achieved performance for the 27p discretization is significantly higher than the achieved performance for the 7p and 13p discretizations, as expected given the increased density of the matrix. Moreover, the achieved performance for all cases practically reaches the roofline, which indicates that the implementation is efficient and that the performance is limited by the computational capabilities of the hardware rather than by the implementation itself, which makes the code suitable to perform efficiently in large-scale simulations and to leverage the potential of the proposed approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation.

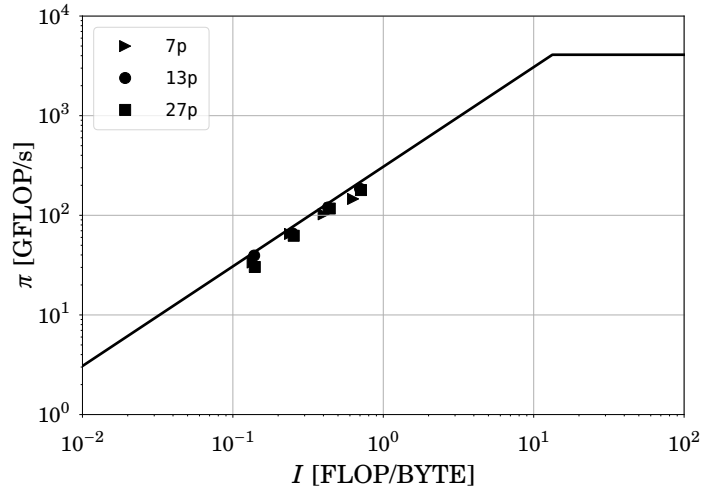


Figure 4.10: Roofline plot of the SpMM kernel for the different discretizations for a turbulent planar channel flow of $Re_\tau = 180$ with a 160^3 mesh.

With regards to the extension to the solution of the Poisson equation using the aAMG preconditioned Conjugate Gradient method implemented in CHRONOS, speed-up figures are obtained and compared with the theoretical bounds given by Eq.(4.10)

with $\chi \approx 0.8$, as the aAMG preconditioner is SpMV-heavy. The results are depicted in Figure 4.11, which shows that the speed-ups for the SpMM operations translate to the whole solution of the Poisson equation, overlapping with the theoretical bounds until $m = 4$. However, as a big workload is performed in the preconditioner, which does not depend much on the original discretization, the speed-ups for the solution of the Poisson equation do not significantly vary among the three discretization cases tested.

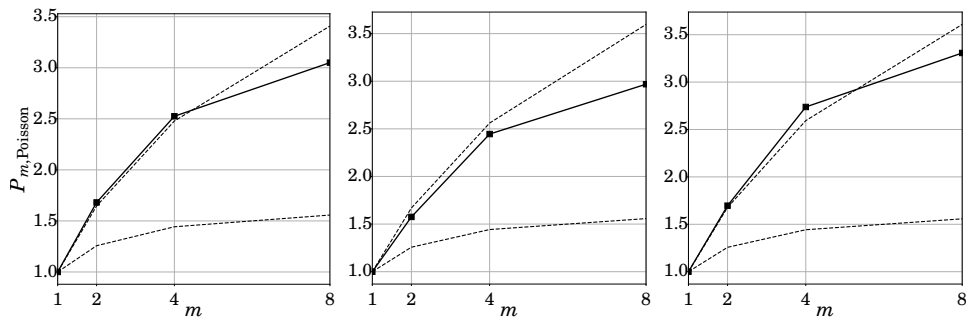


Figure 4.11: Speed-ups for the numerical solution of the Poisson equation for the 7p (left), 13p (center), and 27p (right) discretizations for a turbulent planar channel flow of $Re_\tau = 180$ with a 160^3 mesh. The dashed lines represent the theoretical bounds for the speed-ups given the theoretical analysis, while the markers represent the obtained results.

Eventually, the results have been extended to the whole iteration in Figure 4.12, where the benefits of the increased arithmetic intensity result in speed-ups of around a 100% for $m = 4$ and $m = 8$. While smaller for $m = 2$, the speed-ups for this case are also significant in the three tested discretizations.

By doing so, by means of Eq. (4.26), the speed-up for the whole simulation can be estimated for the different discretizations, which is depicted in Figure 4.13, where the continuous lines represent the estimated speed-ups given an in-flight mesh refinement strategy, following the parameters obtained in the numerical experiments, while the dashed lines represent the estimated speed-ups without mesh refinement. It can be observed that the estimated speed-ups for the whole simulation are significantly higher for the case with mesh refinement compared to the case without mesh refinement, which aligns with the results obtained in Figure 4.7 and highlights the potential of using a cheaper procedure for the transition phase to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation, as the benefits in terms of speed-up grow for approximately a 10% for $m = 2$, a 40% for $m = 4$, and a 60% for $m = 8$ when using the mesh refinement.

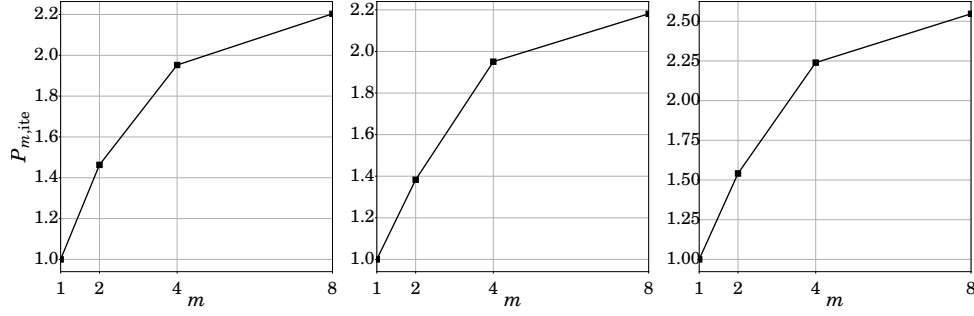


Figure 4.12: Speed-ups in a time step computation for the 7p (left), 13p (center), and 27p (right) discretizations for a turbulent planar channel flow of $Re_\tau = 180$ with a 160^3 mesh.

Moreover, the estimated speed-ups for the whole simulation are significantly higher for the denser matrices, as follows from the previous analysis.

4.3.2 Rayleigh-Bénard convection

To test the robustness of the method when additional transport equations are included, the method has been tested in a Rayleigh-Bénard convection setup, with $Ra = 10^9$ and $Pr = 0.71$. The method has been tested in a $160 \times 320 \times 64$ mesh, with a domain periodic in the z -direction. In the y -direction, the mesh has been refined with a hyperbolic tangent function with $\gamma = 1.5$.

In order to let the flow develop to a steady-state, which will make the relevant data more uniform through the iterations, the simulation has been run for 50 time units. Note that this is not enough to obtain a statistically steady state [24] and thus the presented results only aim at wall-clock times and speed-ups. For the 7p case, a mesh refinement from a $80 \times 160 \times 32$ mesh to the $160 \times 320 \times 64$ mesh has been applied after 30 time units, in order to test the values of δ_m obtained by adding this additional transport equation.

Similarly to the channel flow test case, the simulations here presented have been run in 64 cores of a node of MN5-GPP supercomputer, which actually preserve the same workload per CPU core as in the channel flow case.

As a full simulation letting the flow develop has not been performed, only actual figures for Poisson and iteration speed-up will be presented for a multi-mesh set-up. On the other hand, estimations for the whole simulation can be extrapolated by setting

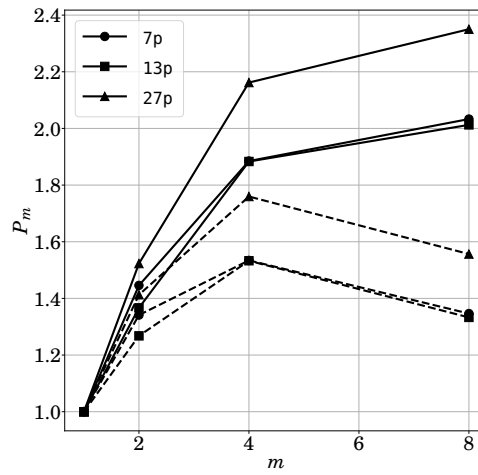


Figure 4.13: Speed-ups for the whole simulation for the 7p, 13p, and 27p discretizations for a turbulent planar channel flow of $Re_\tau = 180$. The continuous lines represent the estimated speed-ups given an in-flight mesh refinement strategy, following the parameters obtained in the numerical experiments, while the dashed lines represent the estimated speed-ups without mesh refinement.

a transition time T_T of 50 time units, as it shows to be sufficient time to develop the flow, and an averaging time T_A of 400 time units, set by Demou and Grioriadis [25] in previous simulations of equivalent Rayleigh-Bénard convection simulations, which leads to $\beta = 8$. In the estimations, T_D will be set to 40 time units, so that the flow can redevelop in the finer mesh. Therefore, the estimations will be performed with $\beta = 8$ and $\gamma = 4$, which gives smaller benefits in terms of speed-up compared to the channel flow case, which used $\beta = 10$ and $\gamma = 15$.

Figure 4.14 depicts the speed-ups for the Poisson equation, for the iteration, and for the whole simulation for a Rayleigh-Bénard convection of $Ra = 10^9$ and $Pr = 0.71$ with mesh refinement. It can be observed that the speed-ups for the Poisson equation are the most significant, which is expected given the amount of SpMV operations performed in a SpMV-heavy aAMG preconditioner for the CG solver. The speed-ups for the iteration and for the whole simulation are also significant, which highlights the potential of this approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation. It is important to note that the speed-ups for the whole simulation are estimated based on the theoretical analysis and the obtained speed-ups for the iteration, as the full simulation has not been performed. Similarly to Figure 4.7, the approximation $\Phi = 1$ results in a good estimation of the speed-ups for the whole simulation.

Opposite to the channel flow test case, the lower value of β for this case leads to a lower potential speed-up, which is expected from the theoretical analysis. However, the presence of the in-flight mesh refinement strategy allows to significantly increase the speed-ups for the whole simulation with a factor of around 30% for $m = 4$, which corresponds to the optimal number of flow states for this case, and a factor of around 45% for $m = 8$, which is the maximum number of flow states tested in this work. The higher the number of flow states, the higher the benefits of using a cheaper procedure for the transition phase, thus penalizing notably the single phase approach.

4.3.3 Multiple parameter simulations: airfoil with flaps and slats depleted

The method has been tested as well in an unstructured mesh to validate the performance of the method in an industrially relevant case. In order to do so, the numerical simulation of the flow at $Re=10^7$ around a 30P30N airfoil is presented for a single mesh approach. Instead of performing an ensemble average, in this case the multiple rhs are devoted to simulate multiple flow states arising from different angles of attack α . Thus, performing these simulations simultaneously allows to significantly reduce the time required to

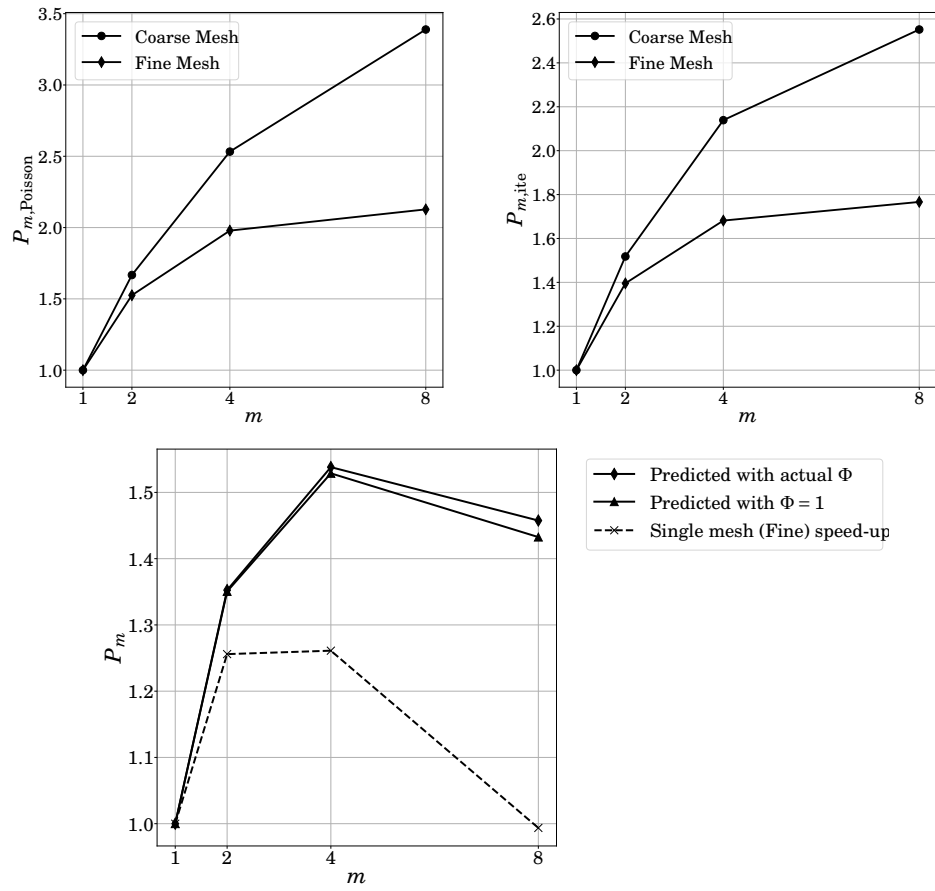


Figure 4.14: Speed-ups for the Poisson equation (top left), for the iteration (top right), and for the whole simulation (bottom) for a Rayleigh-Bénard convection of $Ra = 10^9$ and $Pr = 0.71$ with mesh refinement.

perform parametric studies compared to performing multiple independent simulations, as it allows to leverage the increase in arithmetic intensity provided by the SpMM.

The simulated domain is $150c \times 150c \times 2c$, where c is the chord length of the airfoil, in a mesh of 14M cells (Figure 4.15). The test case has been run for 1, 2, 4, and 8 flow states, which correspond to simulating multiple angles of attack simultaneously.

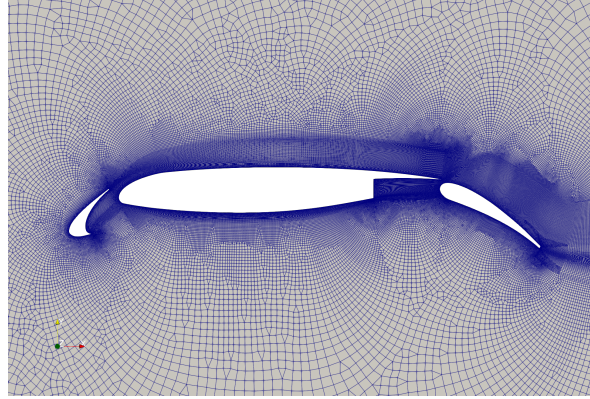


Figure 4.15: Mesh for the simulation of the flow around a 30P30N airfoil.

The simulation has been run for 100 time steps using a third-order Runge-Kutta method with a CFL of 0.95, which guarantees that the timestep is as big as possible while ensuring stability of the integration. The simulation has been run in 112 cores of a node of MN5-GPP supercomputer, which corresponds to a workload of around 115k cells per CPU core, which is a reasonable workload for an unstructured mesh, in a MPI-only parallelization strategy.

Figure 4.16 depicts the speed-ups for the Poisson equation, for the iteration, and for the SpMM kernel for a the flow around a 30P30N airfoil at $Re = 10^7$. It can be observed that the speed-ups for all the tested elements are significant, which highlights the potential of this approach to significantly reduce the time required to perform parametric studies compared to performing multiple independent simulations, as it allows to leverage the increase in arithmetic intensity provided by the SpMM. Moreover, the speed-ups per iteration directly translate to the speed-ups for the whole simulation, since multiple parameter simulations do not have a common transition phase among all the flow states, and therefore the speed-ups for the iteration are a good estimation of the speed-ups for the whole simulation. The speed-ups for the SpMM kernel are significantly higher than the speed-ups for the Poisson equation and for the iteration, which is expected given

that the SpMM kernel benefits more from the increase in arithmetic intensity provided by performing multiple simulations simultaneously compared to performing multiple independent simulations.

4.4 Extension to higher-order symmetry-preserving discretizations

The denser operators used in Section 4.3 for the channel flow case are manufactured just to test the method for matrices that are denser than the ones obtained with a first-neighbor finite-volume discretization, which is the one used in the rest of the numerical experiments. However, these denser matrices can be representative of the matrices obtained with higher-order symmetry-preserving discretizations, which generally have denser sparsity patterns compared to FVM just using the first neighbors. Therefore, the method can be extended to higher-order symmetry-preserving discretizations, which would allow to significantly increase the accuracy of the simulations while still leveraging the potential of this approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation.

According to Trias [26], a fourth-order three-dimensional Laplacian discretization can be obtained by extending the fourth-order one-dimensional Laplacian discretization to multiple dimensions by defining the residual matrix as

$$R = -\frac{1}{24} T_{cs}^T T_{cs}, \quad (4.27)$$

where T_{cs} is the cell-to-face incidence matrix. By doing so, a fourth-order symmetry-preserving discretization can be obtained as

$$\tilde{L} = (I + R)L(I + R). \quad (4.28)$$

Recalling the definition of $L = MG$, a high-order divergence operator can be defined as $\tilde{M} = (I + R)M$ and a high-order gradient operator can be defined as $\tilde{G} = G(I + R)$, which ensures that the properties of the continuous operators are preserved in the discrete operators, following the leitmotif of symmetry-preserving discretizations.

Therefore, this same exact deconvolution can be applied to the convective term, leading to

$$\tilde{C} = (I + R)C(I + R), \quad (4.29)$$

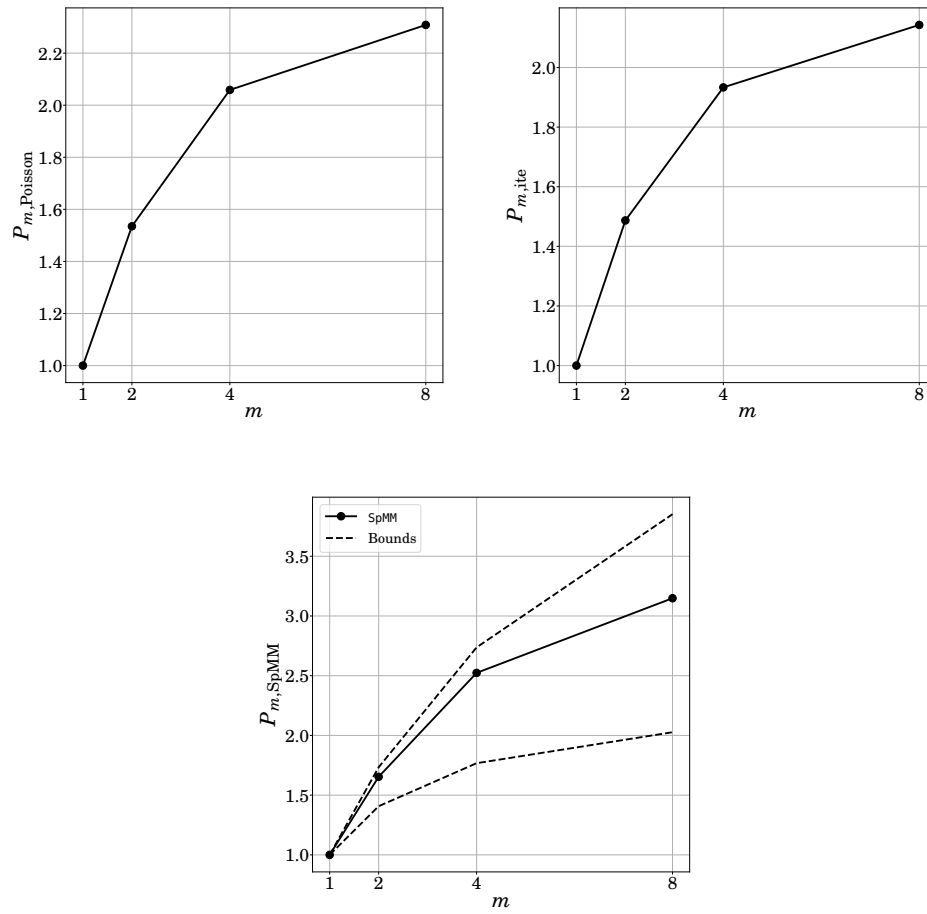


Figure 4.16: Speed-ups for the Poisson equation (top left), for the iteration (top right), and for the SpMM kernel (bottom) for a the flow around a 30P30N airfoil at $\text{Re} = 10^7$.

which, similarly to the diffusive term, extends its accuracy to fourth-order. Recalling the structure of the convective term, $C = MU_s\Pi$, a high-order interpolation can be defined as $\tilde{\Pi} = \Pi(I + R)$. Therefore, these new higher-order operators will have a greater number of non-zeros per row compared to the original operators, which, according to Figure 4.2, leads to bigger gains in terms of arithmetic intensity, and therefore significant increases in the speed-up, both in the SpMM kernel as well as the whole simulation.

Their sparsity patterns can straightforwardly be obtained by introducing the second-order stencil of the cell neighbors into the stencil of the current cell as depicted in Figure 4.17, which shows a two-dimensional representation of the stencil for the second-order divergence operator, fourth-order divergence operator, second-order gradient operator and cell-to-face interpolator, and fourth-order gradient operator. The red arrows represent the expansion of the stencil from the second-order to the fourth-order discretization. It can be observed that the stencil for the fourth-order discretization includes the stencil for the second-order discretization, which is expected given that the fourth-order discretization is obtained by applying an exact deconvolution to the second-order discretization, which ensures that the properties of the continuous operators are preserved in the discrete operators, as the symmetry-preserving properties are preserved by the extension to higher-order discretizations.

Its extension to three dimensions is straightforward by including the neighbors in the z -direction as well, which would lead to a significant increase in the number of non-zeros per row compared to the second-order discretization. Table 4.5 summarizes the number of non-zeros per row for the second-order and fourth-order discretizations for the divergence operator, gradient operator, and cell-to-face interpolator in 1D, 2D, and 3D for structured and unstructured meshes. These much denser matrices lead to a significant increase in the arithmetic intensity of the computations, and therefore to a significant increase in the potential speed-ups provided by this approach, as shown in the previous numerical experiments.

4.4.1 Application to a projection method

Projection methods, as seen in Section 2.2.2, depend on the numerical solution of a Poisson equation for the pressure, which reads as

$$L\mathbf{p}_c = \mathbf{M}\mathbf{u}_s^*, \quad (4.30)$$

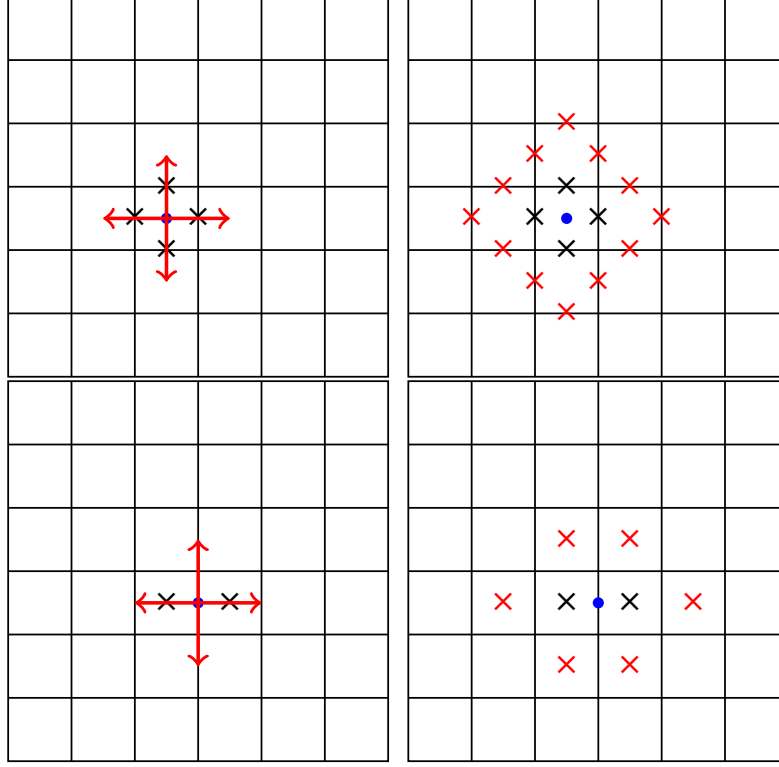


Figure 4.17: Two-dimensional representation of the stencil for the second-order divergence operator (top left), fourth-order divergence operator (top right), second-order gradient operator and cell-to-face interpolator (bottom left), and fourth-order gradient operator (bottom right). The red arrows represent the expansion of the stencil from the second-order to the fourth-order discretization.

Table 4.5: Number of non-zeros per row for the second-order and fourth-order discretizations for the divergence operator, gradient operator, and cell-to-face interpolator in 1D, 2D, and 3D for structured and unstructured meshes.

	M	$\tilde{M}$	G, Π	$\tilde{G}, \tilde{\Pi}$	L	$\tilde{L}$	C	$\tilde{C}$
$1D$	2	4	2	4	3	5	2	4
$2D^{tri}$	3	9	2	6	4	19	3	18
$2D^{quad}$	4	16	2	8	5	25	4	24
$3D^{hex}$	6	36	2	12	7	63	6	62

which arises from the projection step $\mathbf{u}_s = \mathbf{u}_s^* - G\mathbf{p}_c$. If the high-order operators are considered, Eq. (4.30) can be rewritten as

$$\tilde{L}\tilde{\mathbf{p}}_c = \tilde{M}\mathbf{u}_s^*, \quad (4.31)$$

which arises from the projection step $\tilde{\mathbf{u}}_s = \mathbf{u}_s^* - \tilde{G}\tilde{\mathbf{p}}_c$. Introducing the definitions for the high-order operators and simplifying $I + R$, Eq.(4.31) recovers

$$L(I + R)\tilde{\mathbf{p}}_c = M\mathbf{u}_s^*, \quad (4.32)$$

from which is straightforward to show that the outcome from the Poisson equation for both second- and fourth-order schemes is going to be the same, as the high-order approximation of $\mathbf{p}_c$ is obtained by applying the exact deconvolution to the second-order approximation, i.e., $\mathbf{p}_c = (I + R)\tilde{\mathbf{p}}_c$. Therefore, the Laplacian used for the Poisson solution is going to be the second-order Laplacian, which is the one used in the rest of the numerical experiments, and therefore the speed-ups for the Poisson solution are not expected to vary between the second-order and fourth-order discretizations, while preserving the reduced computational cost of the second-order Laplacian compared to its fourth-order counterpart. On the other hand, the convective and diffusive terms are going to be computed with the fourth-order discretization, which is expected to lead to a significant increase in the accuracy of the simulations.

4.4.2 Numerical experiments for higher-order discretizations

In order to test the method for higher-order discretizations, the numerical simulation of a turbulent planar channel flow of $\text{Re}_\tau = 180$ with a 128^3 mesh has been performed using the second-order and fourth-order symmetry-preserving discretizations described in the previous section. Given the additional computational cost for the higher-order discretizations, the simulation has been run for two time units, which is enough to have a representative average time per iteration for the different elements. The simulation has been run under the same workload per CPU core as the previous channel flow test case, which corresponds to 64k cells per CPU core, in an MPI-only parallelization strategy.

Two discretization strategies have been tested: the second-order symmetry-preserving (SP2) [7], and the fourth-order symmetry-preserving (SP4) [26] discretizations. The speed-ups for the calculation of the convective and diffusive terms for the second-order and fourth-order discretizations are depicted in Figure 4.18. It can be observed that the

speed-ups for the SP4 discretization are significantly higher than the speed-ups for the SP2 discretization, which is expected given the increase in arithmetic intensity provided by the denser matrices of the SP4 discretization compared to the SP2 discretization. As the density of the convective term is smaller than that of the diffusive term (Table 4.5), the speed-ups for the convective term are smaller than the speed-ups for the diffusive term, which is expected given the increase in arithmetic intensity provided by the denser matrices of the diffusive term compared to the convective term.

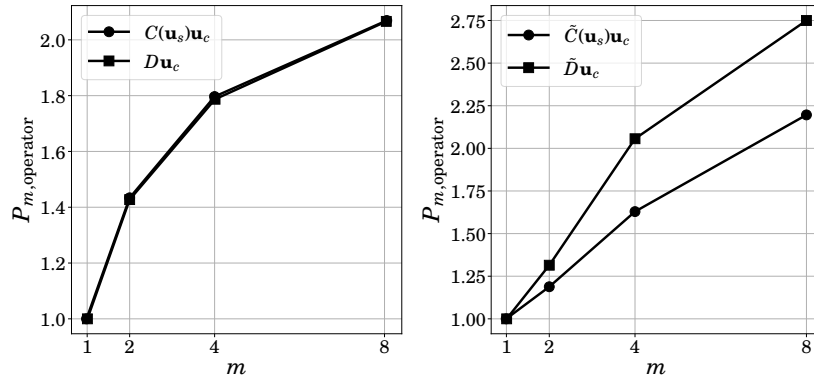


Figure 4.18: Speed-ups in the calculation of the convective and diffusive terms for the second-order (left) and fourth-order (right) discretizations in numerical solution of a turbulent planar channel flow of $\text{Re}_\tau = 180$ with a 128^3 mesh.

These results indicate that the method can be extended to higher-order symmetry-preserving discretizations, which would allow to significantly increase the accuracy of the simulations while still leveraging the potential of this approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation, as the increase in arithmetic intensity provided by the denser matrices of the higher-order discretizations leads to significantly higher speed-ups for the calculation of the convective and diffusive terms compared to the second-order discretization.

4.5 Conclusions

In this chapter, a method to perform parallel-in-time ensemble-averaged simulations by running multiple flow states simultaneously has been presented. The method extends

on the work from Krasnopolsky [3], which focused on incrementing the arithmetic intensity in the Poisson equation resolution, to the whole iteration by transforming all SpMV operations into SpMM. This allows to potentially increase the speed-ups for the whole iteration and therefore for the whole simulation.

The method relies on the ergodicity of turbulent flows, which allows the replace the long time average required to obtain the flow statistics by an ensemble average, which can be obtained by running multiple flow states simultaneously. By doing so, the method allows to leverage the increase in arithmetic intensity provided by performing multiple simulations simultaneously compared to performing multiple independent simulations, which leads to significant speed-ups for the Poisson equation resolution, for the iteration, and for the whole simulation compared to performing a single long simulation.

As the main drawback of this approach is the transition phase required to let the flow develop until a statistically steady-state is reached, an in-flight mesh refinement strategy has been proposed to significantly reduce the time required to perform the transition phase, which is a significant part of the total simulation time for cases that take a long time to develop, i.e., natural convective flows. By doing so, the method allows to significantly increase the speed-ups for the whole simulation compared to performing either a single long simulation or parallel-in-time ensemble-averaged simulations without mesh refinement, without penalizing the accuracy as the flow is re-developed in the finer mesh.

The method has been tested in three different test cases: a turbulent planar channel flow, a Rayleigh-Bénard convection, and the flow around a 30P30N airfoil. The results show that the method allows to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation, with speed-ups of of around 50% for the whole simulation for the channel flow and the Rayleigh-Bénard cases, and of around 100% for the whole simulation for the airfoil case. Moreover, the results show that the speed-ups for the Poisson equation resolution are significantly higher than the speed-ups for the iteration and for the whole simulation, which is expected given the amount of SpMV operations performed in a SpMV-heavy aAMG preconditioner for the CG solver. The speed-ups for the iteration and for the whole simulation are also significant, which highlights the potential of this approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation.

Given the additional benefits observed for denser matrices, the method has been extended to higher-order symmetry-preserving discretizations, which generally have

denser sparsity patterns compared to classical second-order symmetry-preserving discretizations. As this higher-order discretization does not affect the Laplacian used for the Poisson solution, the speed-ups for the Poisson solution are not expected to vary between the second-order and fourth-order discretizations, while preserving the reduced computational cost of the second-order Laplacian compared to its fourth-order counterpart. On the other hand, as the convective and diffusive terms are computed with the fourth-order discretization, the speed-ups for the calculation of these terms improved by a 20% for the convective term and a 70% for the diffusive term, compared to the second-order discretization.

The conclusion after these results is that the method presented in this chapter allows to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation, with significant speed-ups for the Poisson equation resolution, for the iteration, and for the whole simulation. Moreover, the method can be extended to higher-order symmetry-preserving discretizations, which would allow to significantly increase the accuracy of the simulations while still leveraging the potential of this approach to significantly reduce the time required to perform parallel-in-time ensemble-averaged simulations compared to performing a single long simulation.

References

- [1] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, April 2009.
- [2] Joseph L. Greathouse and Mayank Daga. Efficient Sparse Matrix-Vector Multiplication on GPUs Using the CSR Storage Format. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, volume 2015-January, pages 769–780. IEEE Computer Society, January 2014.
- [3] Boris I. Krasnopol'sky. An approach for accelerating incompressible turbulent flow simulations based on simultaneous modelling of multiple ensembles. *Computer Physics Communications*, 229:8–19, August 2018.
- [4] Vakhtang Makarashvili, Elia Merzari, Aleksandr Obabko, Andrew Siegel, and Paul Fischer. A performance analysis of ensemble averaging for high fidelity turbulence simulations at the strong scaling limit. *Computer Physics Communications*, 219:236–245, October 2017.

- [5] Àdel Alsalti-Baldellou, Xavier Álvarez-Farré, Guillem Colomer, Andrey Gorobets, Carlos David Pérez-Segarra, Assensi Oliva, and F. Xavier Trias. Lighter and faster simulations on domains with symmetries. *Computers and Fluids*, 275, May 2024.
- [6] Àdel Alsalti-Baldellou, Carlo Janna, Xavier Álvarez-Farré, and F. Xavier Trias. A Multigrid Reduction Framework for Domains with Symmetries. *SIAM Journal on Scientific Computing*, 46(6):B860–B883, December 2024.
- [7] F. X. Trias, O. Lehmkuhl, A. Oliva, C. D. Pérez-Segarra, and R. W.C.P. Verstappen. Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids. *Journal of Computational Physics*, 258:246–267, February 2014.
- [8] F. X. Trias, X. Álvarez-Farré, A. Alsalti-Baldellou, A. Gorobets, and A. Oliva. An efficient eigenvalue bounding method: CFL condition revisited. *Computer Physics Communications*, 305, December 2024.
- [9] N. V. Nikitin. Disturbance growth rate in turbulent wall flows. *Fluid Dynamics*, 44(5):652–657, October 2009.
- [10] Laurence Keefe, Parviz Moin, and John Kim. The dimension of attractors underlying periodic turbulent Poiseuille flow. *Journal of Fluid Mechanics*, 242:1–29, 1992.
- [11] R. Badii, K. Heinzelmann, P. F. Meier, and A. Politi. Correlation functions and generalized Lyapunov exponents. *Physical Review A*, 37(4):1323–1328, February 1988.
- [12] C. F. O. Mendes, R. M. Da Silva, and M. W. Beims. Decay of the distance autocorrelation and Lyapunov exponents. *Physical Review E*, 99(6):062206, June 2019.
- [13] A. Cheskidov and C. Foias. On global attractors of the 3D Navier–Stokes equations. *Journal of Differential Equations*, 231(2):714–754, December 2006.
- [14] Matheus Cheque Bortolan, Alexandre Nolasco de Carvalho, Pedro Marín-Rubio, and José Valero. Weak global attractor for the 3D-Navier-Stokes equations via the globally modified Navier-Stokes equations. February 2024.

- [15] B. I. Krasnopolsky. Optimal Strategy for Modelling Turbulent Flows with Ensemble Averaging on High Performance Computing Systems. *Lobachevskii Journal of Mathematics*, 39(4):533–542, May 2018.
- [16] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187(1):343–368, May 2003.
- [17] B. Sanderse and B. Koren. Accuracy analysis of explicit Runge–Kutta methods applied to the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 231(8):3041–3063, April 2012.
- [18] X. Álvarez, A. Gorobets, F. X. Trias, R. Borrell, and G. Oyarzun. HPC2—A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD. *Computers and Fluids*, 173:285–292, September 2018.
- [19] Robert D Falgout, Jim E Jones, and Ulrike Meier Yang. The Design and Implementation of hypre, a Library of Parallel High Performance Preconditioners. In A.M. Bruaset and A. Tveito, editors, *Numerical Solution of Partial Differential Equations on Parallel Computers. Lecture Notes in Computational Science and Engineering*, volume 51, pages 267–294. Springer, Berlin, Heidelberg, 2006.
- [20] Giovanni Isotton, Matteo Frigo, Nicolò Spiezia, and Carlo Janna. Chronos: A General Purpose Classical AMG Solver for High Performance Computing. *SIAM Journal on Scientific Computing*, 43(5):C335–C357, January 2021.
- [21] Victor A. Paludetto Magri, Andrea Franceschini, and Carlo Janna. A Novel Algebraic Multigrid Approach Based on Adaptive Smoothing and Prolongation for Ill-Conditioned Systems. *SIAM Journal on Scientific Computing*, 41(1):A190–A219, January 2019.
- [22] Hans De Sterck, Ulrike Meier Yang, and Jeffrey J. Heys. Reducing Complexity in Parallel Algebraic Multigrid Preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27(4):1019–1039, January 2006.
- [23] Hans De Sterck, Robert D. Falgout, Joshua W. Nolting, and Ulrike Meier Yang. Distance-two interpolation for parallel algebraic multigrid. *Numerical Linear Algebra with Applications*, 15(2-3):115–139, March 2008.
- [24] F. X. Trias, A. Gorobets, M. Soria, and A. Oliva. Direct numerical simulation of a differentially heated cavity of aspect ratio 4 with Rayleigh numbers up to 10^{11} -

Part I: Numerical methods and time-averaged flow. *International Journal of Heat and Mass Transfer*, 53(4):665–673, 2010.

- [25] Andreas D. Demou and Dimokratis G. E. Grigoriadis. Direct numerical simulations of Rayleigh–Bénard convection in water with non-Oberbeck–Boussinesq effects. *Journal of Fluid Mechanics*, 881:1073–1096, December 2019.
- [26] F Xavier Trias. Symmetry-preserving discrete deconvolutions. *ERCRAFT Bulletin*, 2025.

Concluding Remarks

“I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing on the seashore... whilst the great ocean of truth lay all undiscovered before me.”

– Isaac Newton

This thesis has addressed the fundamental challenge of computational efficiency in incompressible flow simulations by replacing traditional, overconservative timestepping methods with a rigorous framework for stability-aware adaptive timestepping and hardware-optimized sparse linear algebra. The central ambition of this work, which was *marching faster in time* has been realized by bridging the gap between mathematical stability theory and high-performance computing implementation, enabling significant speedups in simulations of incompressible flows without sacrificing accuracy or stability.

A primary achievement of this research is the development and extension of self-adaptive time-integration methods for the incompressible Navier-Stokes equations. By implementing algorithms like AlgEigCD on staggered grids, this work demonstrated that the classical Courant-Friedrichs-Lewy (CFL) condition often fails to capture the true stability limits of the system. Through rigorous eigenvalue analysis, it was shown that simulations can safely utilize significantly larger timesteps than those prescribed by the CFL condition, leading to substantial reductions in computational time, without compromising the stability or accuracy of the results. This finding challenges the conventional wisdom in computational fluid dynamics and opens new avenues for efficient simulation strategies.

Furthermore, this thesis expanded these concepts into the domain of Reduced-Order

Models (ROMs). With the introduction of the RedEigCD algorithm, this work provided a novel mechanism for estimating stability limits within the reduced subspace. A key theoretical contribution was the application of the Poincaré separation theorem, which establishes a rigorous connection between the eigenvalues of the full system and those of the reduced model, indicating that they are interlaced, thus ensuring that the full system's eigenbound is always greater than or equal to the reduced system's eigenbound. This insight not only validates the use of reduced-order models for stability analysis but also provides a theoretical foundation for their application in adaptive timestepping strategies. This explains the inherent ability of ROMs to sustain larger timesteps than the Full-Order Model (FOM), and establishes a reliable foundation for their use in rapid design and optimization tasks.

Parallel to these mathematical advancements, the research addressed the memory-boundedness of Computational Fluid Dynamics (CFD) simulations, which sets a fundamental limit on the performance of CFD simulations in modern hardware. By identifying and exploiting repeated matrix block structures present in parallel-in-time ensemble simulations, the computational kernel was shifted from a sparse matrix-vector product to a more efficient sparse matrix-matrix product. This shift, combined with the use of the self-adaptive timestepping methods developed in this thesis, led to significant performance improvements, as it increased the arithmetic intensity of the computations, allowing for better utilization of modern hardware capabilities. Validated against academic and industrial scenarios, this approach demonstrated that numerical speedup can be achieved without compromising the accuracy or stability of the simulations, thus providing a practical pathway for accelerating CFD simulations in real-world applications.

5.1 Future Work

While this thesis has made significant strides in advancing the state of the art in adaptive timestepping and efficient linear algebra for incompressible flow simulations, there remain several promising avenues for future research. In the line of adaptive timestepping for the incompressible Navier-Stokes equations and reduced-order models, future work consists of:

- Study the capabilities of the AlgEigCD algorithm in energy-conserving explicit time integration methods, such as Relaxation Runge-Kutta (RRK), which are commonly used in compressible flow simulations. This would extend the capabilities of `tfa` to perform machine-accurate energy-conserving simulations of incompressible flows,

and would also provide insights into the stability properties of RRK methods, which are currently not well understood in the context of incompressible flows.

- Extend the RedEigCD algorithm to hyperreduction methods, which are commonly used in reduced-order modeling to further reduce the computational cost. This would involve developing a rigorous framework for estimating stability limits in the context of hyperreduction, and would provide a more comprehensive understanding of the stability properties of reduced-order models. This would allow preserving the efficiency of hyperreduction while ensuring the stability of the reduced-order models, thus enabling their use in a wider range of applications.
- Investigate the effect of adaptive timestepping on the kinetic energy budgets of the simulations. This would allow determining the numerical dissipation introduced by the adaptive timestepping methods, and compare the results with the numerical dissipation introduced by traditional fixed timestepping methods. This would provide insights into the trade-offs between computational efficiency and physical fidelity in the context of adaptive timestepping, and would help to further optimize the performance of the algorithms developed in this thesis.

In the line of efficient linear algebra for parallel-in-time ensemble simulations, future work consists of:

- Extend the results presented in this thesis to modern GPU-accelerated hardware. This would involve fine tuning the sparse matrix-matrix product kernel on GPU architectures, and optimizing it for performance. This would allow leveraging the computational power of modern GPUs to further accelerate CFD simulations, and would provide insights into the challenges and opportunities of implementing efficient linear algebra algorithms on GPU architectures.
- Perform long term simulations of turbulent flows in unstructured grids using the methods developed in this thesis. This would allow validating the performance and accuracy of the algorithms in more complex and realistic scenarios, and would provide insights into the challenges of simulating turbulent flows with adaptive timestepping and efficient linear algebra methods. This would also help to identify potential limitations of the methods developed in this thesis, and would provide guidance for further improvements and optimizations.
- Implement methodologies to determine transition times in parallel-in-time ensemble simulations. This would involve developing algorithms to analyze, on runtime,

the convergence of the simulations and determine when the simulations have reached a statistically steady state. This would allow optimizing the computational resources by terminating simulations that have already converged, and would provide insights into the dynamics of turbulent flows and the convergence properties of parallel-in-time ensemble simulations.

Published Work

Journal publications

1. J. Plana-Riu, F.X. Trias, À. Alsaltí-Baldellou, X. Álvarez-Farré, G. Colomer, and A. Oliva. *Exploiting repeated matrix block structures for more efficient CFD on modern supercomputers*. *Computer Physics Communications*, 326:110232 (2026).
2. J. Plana-Riu, H. Rosenberger, B. Sanderse, F.X. Trias. *Stable self-adaptive timestepping for Reduced Order Models for incompressible flows*. *Journal of Computational Physics*, 115104 (2026).

Conference publications

As a first author

1. J. Plana-Riu, F.X. Trias, À. Alsaltí-Baldellou, A. Oliva. *Strategies for increasing the arithmetic intensity on ensemble averaged parallel-in-time simulations*. In: 34th Parallel Computational Fluid Dynamics Conference (ParCFD 2023), Cuenca, Ecuador, 2023.
2. J. Plana-Riu, E. Schillaci, G. Colomer, J. Rigola. *Induction heating finite-volume simulations for cooking applications*. In: 14th International Conference on Computational Heat and Mass Transfer (ICCHMT 2023), Düsseldorf, Germany, 2023.

3. J. Plana-Riu, F.X. Trias, C.D. Pérez-Segarra, A. Oliva. *On self-adaptive Runge-Kutta schemes with improved energy-conservation properties*. In: 14th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements (ETMM14), Barcelona, Spain, 2023.
4. J. Plana-Riu, F.X. Trias, C.D. Pérez-Segarra, A. Oliva. *Cost-vs-accuracy analysis of self-adaptive time-integration methods*. In: 10th International Symposium on Turbulence, Heat and Mass Transfer (THMT10), Rome, Italy, 2023.
5. J. Plana-Riu, F.X. Trias, A. Oliva. *Beyond classical stability analysis on Runge-Kutta schemes: positivity and phase preservation*. In 14th ECCOMAS Workshop on Direct and Large-Eddy Simulation (DLES14), Erlangen, Germany, 2024.
6. J. Plana-Riu, F.X. Trias, À. Alsalti-Baldellou, G. Colomer, A. Oliva. *Performance analysis of parallel-in-time techniques in modern supercomputers*. In: 9th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2024), Lisbon, Portugal, 2024.
7. J. Plana-Riu, F.X. Trias, À. Alsalti-Baldellou, G. Colomer, A. Oliva. *Ensemble averaging parallel-in-time approach for industrial LES*. In: 9th European Thermal Sciences Conference (EUROTHERM 2024), Bled, Slovenia, 2024.
8. J. Plana-Riu, F.X. Trias, G. Colomer, À. Alsalti-Baldellou, X. Álvarez-Farré, A. Oliva. *Enabling lighter and faster simulations with repeated matrix blocks*. In: 35th Parallel Computational Fluid Dynamics Conference (ParCFD 2024), Bonn, Germany, 2024.
9. J. Plana-Riu, D. Santos, F.X. Trias, A. Oliva. *High-order symmetry-preserving discretizations: application to repeated matrix block structures*. In: 15th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements (ETMM15), Dubrovnik, Croatia, 2025.

As a co-author

1. Y.X. Chen, J. Zheng, O. Sanmartí, J. Plana-Riu, J. Castro. *A comparative numerical analysis of 3D and 2D vertical falling films*. In: 14th International Conference on Computational Heat and Mass Transfer (ICCHMT 2023), Düsseldorf, Germany, 2023.

2. J. Zheng, Y.X. Chen, J. Plana-Riu, O. Sanmartí, J. Castro. *Falling film characteristic numerical modelling and experimental validation*. In: 10th International Symposium on Turbulence, Heat and Mass Transfer (THMT10), Rome, Italy, 2023.
3. M. Mosqueda-Otero, À. Alsalti-Baldellou, X. Álvarez-Farré, J. Plana-Riu, G. Colomer, F.X. Trias, A. Oliva. *A portable algebraic implementation for reliable overnight industrial LES*. In: 35th Parallel Computational Fluid Dynamics Conference (ParCFD 2024), Bonn, Germany, 2024.
4. D. Santos, J.A. Hopman, J. Plana-Riu, F.X. Trias. *A mimetic finite volume method on collocated grids for incompressible flows: application to thermal magnetohydrodynamics*. In: 15th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements (ETMM15), Dubrovnik, Croatia, 2025.
5. M. Mosqueda-Otero, À. Alsalti-Baldellou, J. Plana-Riu, X. Álvarez-Farré, G. Colomer, F.X. Trias, A. Gorobets, A. Oliva. *A portable algebraic implementation for reliable industrial LES*. In: 15th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements (ETMM15), Dubrovnik, Croatia, 2025.