

IMPERIAL

Imperial College London

Department of Aeronautics

Implicit Time-Stepping for Scale-Resolved Industrial Flow Simulations using Spectral/hp Elements

Henrik Wüstenberg

February 2025

1st Supervisor: Prof. Spencer Sherwin
Department of Aeronautics
Imperial College London

2nd Supervisor: Prof. Joaquim Peiró
Department of Aeronautics
Imperial College London

3rd Supervisor: Prof. David Moxey
Department of Engineering
King's College London

Submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in
Aeronautics of Imperial College London and the Diploma of Imperial College London.

Abstract

Accurate numerical simulations of unsteady fluid flows require scale-resolving simulations which are, however, limited in applications due to high computational costs. Since computational resources are limited, cost and time constraints require algorithmic improvements instead. Scale-resolving simulations typically rely on explicit time-stepping schemes due to low computational cost per time step. In this work, we investigate an implicit time-stepping scheme that increases the computational cost for each time step, but permits larger time steps to reduce the time-to-solution. In particular, we study a velocity correction scheme that is commonly employed for solving the incompressible Navier-Stokes equations. The semi-implicit velocity correction scheme uses implicit diffusion and explicit advection treatment. The explicit advection treatment becomes a performance bottleneck for high Reynolds number flows around complex geometries, because of a CFL-type condition that constrains the algorithm's stability to small time steps. We explore a linear-implicit velocity correction scheme which removes the CFL limitation. The linear-implicit scheme uses a linearisation of the advection operator and, hence, preserves the linear structure of the semi-implicit algorithm. We perform a comparison of both velocity correction schemes and look into their stability, accuracy and computational performance. Our investigation includes canonical problems and a two-dimensional cylinder flow for verification. Further, we investigate turbulent flows at high Reynolds number and around challenging geometries based on Formula 1 race cars. We find that the linear-implicit scheme allows strong improvements in the stability allowing up to 100-times larger time step sizes on the most complex 3D geometry. The influence on the accuracy with larger time step sizes varies and all cases show negligible differences up to at least 10-times larger time step sizes. Additionally, the stability improvement provides leverage for the computational performance enabling 2-fold speed-up with minor loss in accuracy and 10-fold speed-up in time-to-solution with stronger losses in the accuracy.

Contents

1	Introduction	13
1.1	Motivation	13
1.2	Literature Review	15
1.3	Outline	19
2	Numerical solution of the incompressible Navier-Stokes equations	21
2.1	The incompressible Navier-Stokes equations	21
2.2	Velocity correction schemes	22
2.2.1	Derivation of the velocity correction scheme	22
2.2.2	The semi- and linear-implicit formulations	24
2.2.3	Implementation via weak form	26
2.3	Spatial discretisation via spectral/hp elements	29
2.3.1	Spectral/hp elements	31
2.3.2	Matrix assembly	36
2.3.3	Stabilising under-resolved simulations	37
2.3.4	CFL condition	40
2.4	Solving the linear system	41
2.4.1	Approach for building the matrix	41
2.4.2	Linear system solver	43
2.4.3	Preconditioning	44
2.4.4	Freezing the time-dependent matrix	46
2.5	Extension to quasi-3D Problems	47
2.5.1	Decoupling the linear-implicit advection operator	49
2.5.2	An algorithm for hybrid parallelisation	52

2.6	The Nektar++ software framework	55
3	Stability and accuracy of the linear-implicit scheme	58
3.1	Stability analysis	59
3.1.1	Linear analysis of time-stepping schemes	59
3.1.2	Non-linear stability analysis for velocity correction schemes	60
3.2	Accuracy of the scheme	61
3.2.1	Review of error estimates	61
3.3	Verification of the implementation	62
3.3.1	2D steady Kovasznay flow	62
3.3.2	Manufactured solution: verifying temporal accuracy	66
3.4	Stability and accuracy on a canonical problem	68
3.4.1	2D unsteady cylinder flow	68
3.5	Stability and accuracy on complex geometries	75
3.5.1	The Imperial Front Wing geometry	76
3.5.2	Numerical setup for the IFW	79
3.5.3	2D Imperial Front Wing	81
3.5.4	3D extruded Imperial Front Wing	92
3.6	Numerical characteristics of the 2.5D formulation	108
4	Performance	111
4.1	Benchmark setup	111
4.1.1	Benchmark test case	112
4.1.2	Hardware and software considerations	112
4.2	Performance analysis	113
4.2.1	Speeding-up simulations with implicit time-stepping	113
4.2.2	Algorithmic insight and bottlenecks	115
4.2.3	Algorithmic improvements	118
5	Conclusions	126
5.1	Summary	126
5.2	Conclusion	127
5.3	Future work	129
A	Manufactured solution: Sympy script	140

Acknowledgements

First of all, I want to thank my supervisors for their constant patience and support. I had a unique chance to be supported by three brilliant academics. Spencer Sherwin constantly has the bigger picture in mind and guided me whenever I found myself lost in the depth of my research. Joaquim Peiró always had an open-ear for my issues and doubts as a PhD student and spent his time correcting my misunderstandings as well as my papers and thesis. As for David Moxey, I remember every meeting as he has been there to give advice for the difficult decisions and helped me to see through complex issues with ease. All in all, my supervision trio spans different levels of experience in academia and covers a variety of knowledge on CFD in general and spectral/hp elements in particular. I could not have been more lucky than with this supervision. Thank you all for helping me become a PhD.

I further want to extend my gratitude to the support during my secondments around Europe. Thank you to Ramon Codina who guided me through the complexities of the finite element method during my stay at CIMNE in Barcelona and thanks to the Kmins coffee group. I must also thank Niclas Jansson for an entertaining stay at the KTH in Stockholm, with great office mates, and for sharing his vision for high-performance computing. Finally, the McLaren CFD methodology team, headed by Julien Hoessler, has introduced me to the industry side of CFD and, more importantly, to a great team spirit at work.

Beyond world-class academics and engineers, I could not have been more lucky than with the great community of Aero PhD students at Imperial. The basement offices 141A and 141C certainly feel like a second home. Special thanks to Chi Hin, the London mafia: Parv, Kaloyan and Alex, and our sports enthusiasts for Body Pump, Yoga and Running. I would not have made it without all of you supporting me.

Outside of the university, I appreciate my friends and family for being available and helping out with ample distractions from the day-to-day work. In particular, Tanita for tirelessly standing

by my side, supporting me with great cooking and even greater espresso. Thanks also to my parents Jürgen and Ute as well as my brother's family: Jannik, Josi and Malte for providing a beautiful retreat far away from London. Finally, there are too many friends that helped me through this, but I want to give special thanks to Tom and Sunny. They continuously helped me distract from the everyday troubles with good chatter and video games.

I want to highlight that this PhD experience was only possible through the Marie Skłodowska-Curie program or more formally: This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 955923. Computational resources were supported in part through the Imperial College Research Computing Service (DOI: 10.14469/hpc/2232).

Henrik Wüstenberg

February 2025

Statement of Originality

The work presented hereafter is based on research carried out by the author at the Department of Aeronautics of Imperial College London and it is all the author's own work except where otherwise acknowledged. No part of the present work has been submitted elsewhere for another degree or qualification.

Henrik Wüstenberg

February 2025

Copyright declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution 4.0 International Licence (CC BY).

Under this licence, you may copy and redistribute the material in any medium or format for both commercial and non-commercial purposes. You may also create and distribute modified versions of the work. This on the condition that you credit the author.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

List of Figures

1.1	Time evolution of the lift coefficient for the extruded Imperial Front Wing geometry.	14
2.1	The algorithm of the velocity-correction scheme to compute pressure and velocity at the new time $t^{n+1} = (n + 1)\Delta t$.	28
2.2	Overview of the mapping between collapsed, reference and Cartesian coordinate systems for simplex element types. Adapted from Moxey et al. [9].	32
2.3	Overview of the one-dimensional basis functions for a <i>nodal</i> Lagrange polynomial and a <i>modal</i> Legendre polynomial for polynomial order $P = 5$.	33
2.4	Schematic illustration of matrix assembly from elemental matrices M^e into global block matrix with boundary-interior decomposition M_g .	37
2.5	The full matrix $\mathbf{A}$ and the statically condensed Schur complement matrix $\mathbf{S}$ for a pressure Poisson problem on a 4x4 domain of quadrilateral elements. The condition number κ shows an order of magnitude difference for this canonical mesh.	42
2.6	The distribution of energy across discrete Fourier modes on various industrial flow problems at high Reynolds number.	50
2.7	The mean-implicit algorithm for a quasi-3D velocity-correction scheme. The flowchart shows only few distinct modes to clearly highlight the building blocks of the algorithm. Note that each wavenumber k has two associated half modes (sine and cosine). However, practical simulations typically use $N_z \gg 10$, see figure 2.6.	54
2.8	Top-down structure of the Nektar++ framework. it shows the connection between applications-specific solvers with the core libraries, fundamentals in LibUtilities and third-party dependencies like Boost and Gslib .	56
3.1	The region of stability lies outside the highlighted regions. Comparison of BDF time-stepping schemes for order 1, 2 and 3.	60

3.2	A visualisation of the streamlines in the steady-state Kovasznay flow highlighting the structure of the recirculating flow. Boundaries of the four quadrilateral elements are highlighted in white, streamlines coloured by x-velocity magnitude u and the pressure with a greyscale.	63
3.3	A comparison of the stability measured in time step size Δt that achieves convergence to the steady-state Kovasznay flow.	65
3.4	The time convergence order on a manufactured solution with varying time step size Δt . A comparison of the semi-implicit and linear-implicit scheme where the implicit scheme uses two different advection velocity approaches.	67
3.5	Setup of the cylinder test case highlighting the mesh structure with quadrilateral elements and the cylinder location in green colour. The boundary conditions on all boundaries are highlighted and the cylinder wall itself uses a no-slip condition.	70
3.6	Estimation of the CFL number for different time step sizes and the time-averaged value for the 2D cylinder. The comparison for the semi-implicit and linear-implicit prediction at $\Delta t = 1\Delta t_{\text{CFL}}$ overlay almost exactly.	72
3.7	The time-averaged lift coefficient $\overline{C}_l$ and drag coefficient $\overline{C}_d$ for the cylinder case with increasing time step size. The blue area highlights the a 1% deviation from the semi-implicit prediction. The averaging interval is $T \in [250, 1200]t^*$	73
3.8	Power spectral density (PSD) of the lift coefficient signal in the interval $T = [200, 1200]$. Comparing the frequency spectrum for different time step sizes.	74
3.9	The imperial front wing (IFW) geometry; a F1 front wing based on the McLaren MP4-17D race car. The slice location at $y = 250$ mm for the 2D and 3D test case is highlighted in blue.	77
3.10	Slice of the imperial front wing (IFW) geometry for our investigations in 2D and 3D, and used in [29, 32, 87].	78
3.11	Overview of the computational domain, boundary conditions and meshing approach for the two-dimensional Imperial Front Wing case.	83
3.12	Estimation of the CFL number for different time step sizes and the time-averaged value for the 2D IFW.	84
3.13	The time-averaged lift coefficient $\overline{C}_{l,\text{RMS}}$ and drag coefficient $\overline{C}_d$ for the 2D IFW case with increasing time step size. The averaging interval is $T \in [30, 35]t^*$	86

3.14	The time-averaged pressure coefficient C_p for the 2D IFW case showing only the main plane. Comparison of increasing time step size effect averaging over the interval $T \in [15, 25]t^*$	88
3.15	The time-averaged skin-friction coefficient C_f for the main plane of the 2D IFW case. Comparison of increasing time step size effect using an averaging interval of $T \in [15, 25]t^*$	89
3.16	The power spectral density (PSD) of the lift coefficient using the semi-implicit scheme with strong pressure and linear-implicit with weak pressure formulation. .	90
3.17	The power spectral density (PSD) of the lift coefficient comparing the strong and weak pressure formulation with both semi-implicit and linear-implicit scheme at the same time step size $\Delta t = \Delta t_{\text{CFL}}$	92
3.18	Overview of the computational domain, boundary conditions and meshing approach for the extruded Imperial Front Wing case.	94
3.19	Distribution of wall units $\Delta x^+, \Delta y^+, \Delta z^+$ for the time-averaged flow field on all wing profiles for the extruded IFW.	96
3.20	The skin friction coefficient C_f evaluated on all the surface of all three wing profiles. The main plane reaches up to $1x/c$, the first flap up to $1.3x/c$ and the second flap until $1.8x/c$	97
3.21	Power spectral density of the total lift coefficient using an interval of $10t^*$. Comparison of data from Slaughter et al. [32] and a reference simulation with semi-implicit scheme and $\Delta t = 1\Delta t_{\text{CFL}}$	98
3.22	Variation of CFL estimates in space and time for the eIFW geometry based on the time-averaged velocity field.	99
3.23	Time evolution of the lift coefficient for the extruded Imperial Front Wing geometry using the semi-implicit scheme and a time step size $\Delta t = \Delta t_{\text{CFL}}$	100
3.24	The time-averaged lift coefficient $\overline{C}_l$ and drag coefficient $\overline{C}_d$ for the 3D eIFW case with increasing time step size. We present the sum of the coefficients at the top along with the contribution from each wing. The averaging interval uses $10t^*$ and we show an error of 1% relative to the time-averaged semi-implicit coefficients. .	100
3.25	The time-averaged lift coefficient $\overline{C}_l$ and drag coefficient $\overline{C}_d$ with increasing time step size. We present the coefficients individually for the contributions of each wing element. The averaging interval uses $10t^*$ and we show an error of 1% relative to the time-averaged semi-implicit coefficients.	102

3.26	Time-averaged pressure coefficient $\overline{C}_p$ for all wing elements comparing the influence of larger time step sizes. Locations with transition to turbulence are highlighted with black boxes for the	104
3.27	Time-averaged skin friction coefficient $\overline{C}_f$ for all wing elements comparing the influence of larger time step sizes. Locations with transition to turbulence are highlighted with black boxes.	105
3.28	Power spectral density of the total lift coefficient using an interval of $10t^*$. Comparison of time step size influence $\Delta t = 1\Delta t_{\text{CFL}}$	107
3.29	The time-averaged energy for the wave numbers k highlighting the large gap between mean mode $k = 0$ and higher frequency $k > 0$ modes for the cylinder flow with $Re = 500$	110
4.1	Speed-up and number of iterations comparing the semi-implicit scheme with time step size $\Delta t(\overline{\text{CFL}} \approx 1)$ against the linear-implicit scheme with increasing time step size.	114
4.2	Heat map of execution time spent in the three sections of the velocity-correction scheme algorithm: explicit forcing, pressure solve and velocity solve. Additionally, we show timings of building and solving the linear systems. We show absolute time spent in seconds per 100 time steps as well as in percentage of total execution time. Comparing semi-implicit and linear-implicit scheme at increasing time step size. .	117
4.3	Speed-up and number of iterations comparing the effect of preconditioner freezing for 1, 10, 100 steps for the linear-implicit scheme's velocity problem and with increasing time step size.	120
4.4	Heat map of execution time spent in the three sections of the velocity-correction scheme algorithm: explicit forcing, pressure solve and velocity solve. Additionally, we show timings of building and solving the linear systems. We show absolute time spent in seconds per 100 time steps as well as in percentage of total execution time. Comparing semi-implicit and linear-implicit scheme at increasing time step size. .	121
4.5	Speed-up and number of iterations comparing the effect of advection freezing for 1, 10, 100, 1000 steps for the linear-implicit scheme's velocity problem and with increasing time step size.	124

4.6	Heat map of execution time spent in building and solving the linear systems. We show absolute time spent in seconds per 100 time steps as well as in percentage of total execution time. Investigation into the effect of advection freezing.	125
-----	---	-----

List of Tables

2.1	Definition of reference domains for one, two and three dimensional element types. Adapted from Moxey et al. [9].	31
3.1	An overview of all the test cases investigated for the stability and accuracy of the linear-implicit velocity-correction scheme. The table shows, from left to right, the domain's dimension, Reynolds number, degrees of freedom of the assembled matrix and the polynomial order of the velocity space P_u . Additionally, we highlight the stability in terms of the maximum stable time step size $\Delta t_{\text{CFL}}^{\text{Explicit}}$ with the semi-implicit scheme and the possible increase in time step size using the linear-implicit scheme denoted by $\Delta t_{\text{CFL}}^{\text{Implicit}}$. An indication of accuracy is measured by the difference in time-averaged lift coefficient $\overline{C}_l$ between the semi-implicit prediction and linear-implicit prediction at a time step size $\Delta t = 10\Delta t_{\text{CFL}}$	59
3.2	Stability comparison of the Nektar++ implementation of the linear-implicit against the semi-implicit scheme and results from reference [39]. Additionally, we add the effect of the choice of advection velocity $\tilde{\mathbf{u}}$, extrapolated (Ext.) or updated (Upd.), approach. Data presents the H_1 error of the x-velocity component u for increasing time step size Δt	64
3.3	Mesh convergence for the time-averaged lift and drag coefficients on the 2D cylinder flow. We use data from the interval $T \in [200, 1000]$	70
3.4	Overview of reported time-averaged force coefficients and distinct peaks in the PSD of the force coefficients from literature for the 3D extruded IFW geometry. .	79
3.5	Prediction of the mean drag coefficient $\overline{C}_d$ and RMS of the lift coefficient $\overline{C}_{l,\text{RMS}}$ at increasing time step size. Comparison of the semi-implicit and linear-implicit scheme.	109

Chapter 1

Introduction

1.1 Motivation

Computational Fluid Dynamics (CFD) is a versatile tool in the scientific and engineering community. The range of application ranges from climate and environmental science to biological studies of blood flow all the way to engineering problems in the transport and energy sector. In particular in the transport sector, the automotive and aerospace industries rely on CFD tools to develop efficient vehicles for air, land, sea and space. They require CFD tools to be computationally cheap such that rich optimisation processes are feasible to identify an efficient and effective design.

Scale-resolving simulations are at the high-fidelity end of the CFD toolset [1]. They include the spectrum from highly-accurate direct numerical simulations (DNS) which resolve all scales of the turbulent flow to large eddy simulations (LES) which focus on the energetic content of the turbulent scales [2]. A DNS provides direct insight into the chaotic nature of turbulence, however, it is computationally infeasible for industrial design applications. In contrast, a LES approach provides insight into the larger structures of the flow while the dissipation is not fully resolved. Thus, LES enable a computationally feasible approach for accurate predictions of unsteady flow phenomena such as macro-scale vortices and massive separation regions [3].

In practise, the CFD method of choice tends to low-fidelity Reynolds-averaged Navier-Stokes (RANS) simulations. A RANS approach immediately predicts the time-averaged flow field where it employs a turbulence model for the dynamic effects of turbulent scales. The turbulence model requires careful choosing as their reliability is limited to a range of applications see references

[4, 5] for a discussion on the range of applications. Additionally, the RANS approach typically performs poorly for highly unsteady flows [3]. Consequently, there is a need for computationally feasible and scale-resolving simulations.

The time evolution of a scale-resolving simulation typically goes through an initial transient before reaching a quasi-steady state as shown in figure 1.1. This is a consequence of using an initial guess based on either an analytic expression, for example a uniform flow field, or a precursor RANS solution. The flow field then evolves over time into a turbulent flow. In practise, the quasi-steady state is most relevant for sampling statistics of the physics and, hence, it is desirable to minimise the computational resources to reach this state.

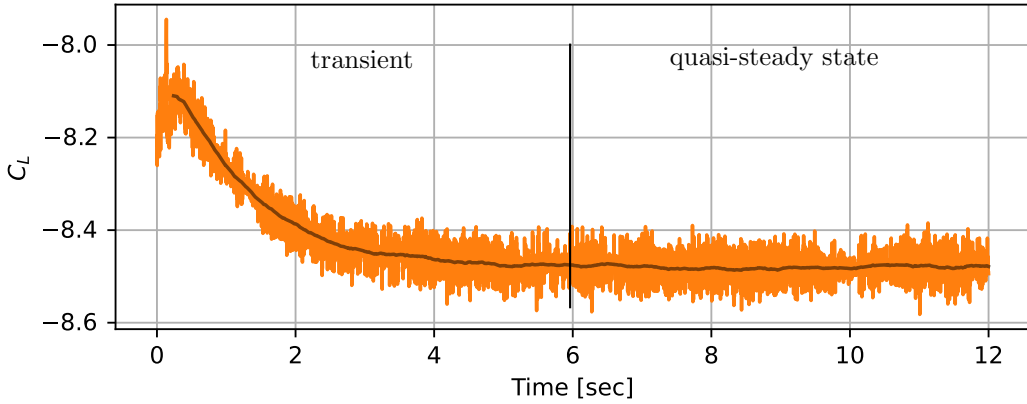


Figure 1.1: Time evolution of the lift coefficient for the extruded Imperial Front Wing geometry.

The computational resources required to reach the quasi-steady state mainly depend on the spatial resolution, the time-stepping scheme as well as the computational efficiency of the implementation. We assume that the spatial resolution is well adapted to the problem at hand and assume it fixed. Instead, we investigate the possibility to reduce computational resources through an effective time-stepping scheme.

A common time-stepping scheme in CFD software for scale-resolving simulations of incompressible flows is the family of fractional step schemes see for example the NeK5000 [6], NekRS [7], Neko [8] or Nektar++ [9] codes. They typically employ a semi-implicit time-stepping scheme where the diffusion term is treated implicitly and the advection term explicitly. This explicit treatment of advection leads to low computational costs per time step, however, it also enforces a Courant-Friedrichs-Lewy (CFL) type stability condition which limits the largest stable time step size. The CFL stability limit becomes particularly restrictive for complex geometries and

high Reynolds number flows due to fine boundary layer meshes. Therefore, a semi-implicit time-stepping schemes requires a large amounts of time steps for both, reaching the quasi-steady state and sampling statistics. Consequently, the time-stepping scheme becomes a performance bottleneck for challenging simulations.

We are investigating computational improvements through an implicit time-stepping scheme [10]. An implicit time-stepping scheme removes the CFL stability restriction of the algorithm and, hence, allows larger time step sizes. A larger time step size allows reaching the quasi-steady state in fewer steps and potentially reduces the computational resources. The implicit treatment of the advection term does, however, incur larger computational cost per time step. This leads to a trade-off between the semi-implicit and implicit algorithms for computational efficiency. Our first research question investigates this: does an implicit time-stepping scheme allow a reduction in computational resources for reaching the quasi-steady state?

The second research question focusses on sampling statistics during the quasi-steady state. The time step size does not only determine the stability and performance of the time-stepping scheme but also influences the accuracy in time. We will investigate whether an implicit scheme further allows reducing the computational costs for gathering statistics of a turbulent flow. The second research question is: does an implicit time-stepping scheme allow sampling accurate statistics of a complex turbulent flow through larger time step sizes?

1.2 Literature Review

We investigate scale-resolving simulations for the incompressible Navier-Stokes equations using implicit large eddy simulation (iLES). The large eddy simulation (LES) methodology aims to resolve the larger scales of turbulent flows and to employ a model for the unresolved smaller scales, also known as subgrid scales (SGS), in order to make the simulations computationally feasible. Sagaut [11] gives a detailed overview of the LES methodology and introduces various approaches for SGS models. The explicit definition of a SGS model that separates the larger and subgrid scales is known as explicit LES (eLES). Explicit SGS models include, for example, the Smagorinsky [12], dynamic Smagorinsky [13] or Vreman [14] models. In contrast to eLES, the iLES approach implicitly defines a model through the numerical method. Hence, the spatial resolution of the computational mesh defines the separation between large and small scales. This allows iLES to flexibly define the fidelity of the method. Grinstein et al. [15] describe the notion of iLES modelling approaches in detail.

The motivation to choose iLES for complex geometries is the robustness with spatial refinement. The work by Dairay et al. [16] compares iLES and eLES approaches for predicting the dissipation spectra of the Taylor-Green vortex benchmark. They investigate the Smagorinsky [12] and dynamic Smagorinsky [13] models as well as an implicit approach, namely spectral vanishing viscosity [17]. Their conclusion shows that the explicit models are sensitive to numerical errors showing a distortion in the kinetic energy spectra in the high and mid-range wavenumber regime. In contrast, the implicit approach with spectral vanishing viscosity shows a robust prediction of the spectra without distortions in the resolved wavenumbers. This robustness is valuable for industrial applications as it reduces the amount of case-dependent tuning. In practice, the spectral vanishing viscosity is tuned such that the numerical dissipation of the iLES approximately mimics the physical dissipation of the incompressible Navier-Stokes equations.

We consider methods that solve the incompressible Navier-Stokes equations in the primitive variables formulation using velocity and pressure. There are broadly two approaches to solve the system of equations. On the one hand, the monolithic, or coupled, approach uses, for example, an Uzawa algorithm, originally proposed in Arrow et al. [18]. The Uzawa algorithm solves the saddle point system of pressure and velocity simultaneously. This approach is numerically stable and accurate, however, relies on robust preconditioners for practical applications, see Karniadakis and Sherwin [19] for a detailed discussion. On the other hand, projection methods, also known as fractional step or splitting schemes, are an approach where the system of equations is split into a velocity and an elliptic pressure problem. They were first introduced in Chorin [20] with a correction step which imposes the divergence-free condition for velocity through an elliptic pressure problem. In contrast to the coupled approach, splitting schemes introduce a splitting error which requires consistent pressure boundary conditions in order to ensure time accuracy, see Karniadakis et al. [21] for a discussion on higher-order splitting schemes.

In this work we consider fractional step schemes. Guermond and Shen [22] give an overview of fractional step schemes with a division into pressure correction, velocity correction and consistent splitting schemes. Following their overview, we employ a rotational velocity correction scheme where the diffusion term is transformed to enforce the divergence-free property within the pressure Poisson problem. The rotational form was initially proposed in the work of Karniadakis et al. [21] where they enable the mathematical framework for higher-order in time with splitting schemes. The most typical form of the rotational velocity correction scheme uses a second-order semi-implicit discretisation with implicit treatment of the diffusion term and explicit treatment of the advection term. This avoids the severe time step restriction of the diffusion operator with

maximum eigenvalue growth $\lambda_{\max} \approx \mathcal{O}(P^4)$ [19]. In contrast, the advection term is less severely restricted with $\lambda_{\max} \approx \mathcal{O}(P^2)$. The semi-implicit form is widely employed in scale-resolving CFD software for incompressible Navier-Stokes including Nek5000 [6], NekRS [7], Neko [8] and Nektar++ [9]. The popularity comes from the computationally efficient setup with only a Poisson and Helmholtz problems both of which have a positive-definite matrix and constant matrix coefficients leading to low computational cost per time step.

The explicit advection treatment in semi-implicit velocity correction schemes enforces a Courant-Friedrichs-Levy (CFL) stability condition on the time step size. We consider industrial applications from the Formula 1 industry. The external aerodynamic flows of Formula 1 race cars consider Reynolds numbers of the order $\text{Re} \approx \mathcal{O}([10^5, 10^6])$, see references [23–25]. For these large Reynolds numbers, the advection term in the incompressible Navier-Stokes equations becomes dominant. Additionally, the geometries involve curved geometries and surface singularities, such as sharp corners, further challenging the numerical stability of algorithms. The combination of high Reynolds number flow and complex geometries cause a significant restriction on the time step size through the CFL condition, see for example references [26–34]. The time resolution enforced by the CFL condition requires time steps of the order $\Delta t \propto \mathcal{O}([10^{-5}, 10^{-6}])$. Using a second order scheme, the small time step size leads to negligibly small temporal errors of the order $\epsilon_t \leq \mathcal{O}([10^{-10}, 10^{-12}])$.

There are a plethora of methods to remove the CFL constraint and form an unconditionally stable time-stepping scheme for the incompressible Navier-Stokes equations. We refer to fully-implicit approaches as the straightforward application of a linearisation to the non-linear problem, for example using a Newton method or Picard iterations, see reference [10]. On the one hand, these approaches are effective for numerical stability and are applicable to any partial differential equation. On the other hand, they typically have a large computational overhead for the inner iterations making them less interesting for the optimised velocity correction schemes. An alternative are semi-Lagrangian formulations which allow stabilising the advection term. These forms are efficiently implemented in weather and climate modelling, for example, Loft et al. [35] combine them with a spectral element method. Xiu et al. [36] discuss the strong and auxiliary form of the semi-Lagrangian approach highlighting the advantage of the auxiliary form for numerical stability. The semi-Lagrangian however requires solving an additional equation for the advection term. Lin et al. [37] describes an energy-stable method that corrects the advection term to avoid unbounded growth of the energy. This method follows the algorithmic structure of the semi-implicit velocity correction scheme, however, doubles the computational work in order

to achieve unconditional stability. Further, an update of the matrices is required depending on a free and problem-dependent parameter that incurs further cost per time step. References [22, 38, 39] propose linearising the advection operator with an explicit advection velocity to form a linear-implicit velocity correction scheme. This similarly removes the CFL constraint, however, requires little change to the structure of the algorithm. Nevertheless, it introduces time-dependent matrix coefficients which adds an computational overhead, however, does not require solving any additional linear systems.

In view of the current literature, the linear-implicit scheme with linearised advection operator appears an ideal choice for avoiding the high cost of the initial transient and potentially low impact on the accuracy in the quasi-steady state. Additionally, the linear-implicit scheme's algorithmic structure follows the semi-implicit algorithm which simplifies the implementation and potentially leads to a minimal computational overhead.

The spatial discretisation for scale-resolving simulations tends towards higher-order methods from the continuous or discontinuous Galerkin family see for example references [6–9, 40, 41]. In this work we employ a continuous Galerkin approach and combine it with a higher-order finite element type method. These higher-order methods combine the geometric flexibility of the finite element method with the exponential convergence of spectral methods [19]. Classic examples are the spectral element method (SEM) and spectral/hp element method [42]. In this work, we focus on complex geometries which require unstructured meshes for accurate resolution of curved surfaces and boundary layer meshes. The spectral/hp element method is particularly well-suited for unstructured meshes, because it enables tensor contractions for simplex-type elements making them computationally efficient. Therefore, the spectral/hp element method provides an effective discretisation for iLES simulations of challenging problems.

We implement the linear-implicit scheme within the Nektar++ framework, see also references [9, 43]. The framework provides an efficient implementation of the spectral/hp element method capable of fast evaluation of operators using tensor contractions. Additionally, Nektar++ allows higher-order meshing capabilities using the meshing tool NekMesh [44]. NekMesh enables discretisation of challenging geometries using unstructured meshes and allows curvature of mesh elements near surfaces [45]. The framework further uses grouping methods to increase the throughput of linear algebra operators [46] and various implementations of stabilisation methods for high Reynolds number simulations based on the spectral vanishing viscosity (SVV) approach [17, 47]. This leads to an overall strong framework for challenging iLES simulations that enables us to test the linear-implicit scheme on canonical and complex geometries at large-scale.

The problems of interest are first of all canonical test cases for the incompressible Navier-Stokes problems. This includes steady and unsteady problems in two dimensions including a manufactured flow, Kovasznay flow [48] and cylinder flow at Reynolds number $Re = 500$. We use the canonical test cases to verify the implementation and identify numerical characteristics in comparison to a previous study by Dong and Shen [39].

Furthermore, we want to extend this study to challenging problems which represent high Reynolds number flows on complex geometries. While these problems are typical for the automotive industry, we focus on a test case based on a Formula 1 race car. A Formula 1 race car uses wings as aerodynamic devices with high lift to drag ratio which provide high downforces at the front and rear of the car, see the work by Toet [49] for more details. The front wing is particularly important as it provides around 30% of the downforce of the race car and effects all aerodynamic devices, air intakes and cooling downstream. We use the open geometry Imperial Front Wing [25] which represents the front wing of the McLaren 17-D race car. The geometry has been studied previously by Pegrum [23] in an experimental campaign to identify the vortex system and numerical studies by Lombard [24] and Buscariolo et al. [34] with the Nektar++ framework validating the scale-resolving simulations against experimental data. These studies however lacked an analysis of the laminar-turbulent transition and the boundary layer flow close to the wing profiles. Slaughter et al. [32] performed a well-resolved iLES and analyse the characteristics of an extruded slice of the Imperial Front Wing. Their work shows a detailed analysis of the three-dimensional turbulent flow describing integral forces, time-averaged surface quantities and spectral analysis. This rich analysis makes the test case an ideal benchmark for our study of the linear-implicit scheme for challenging problems.

1.3 Outline

We introduce the numerical method, that is time-stepping schemes and spatial discretisation, in chapter 2. The method includes both the semi-implicit and linear-implicit formulations of the velocity correction scheme, an introduction to the spectral/hp element method including integration and differentiation rules, basis functions, CFL estimates and the spectral vanishing viscosity. A discussion of the linear-implicit schemes stability and accuracy follows in chapter 3. The stability and accuracy discussion reviews theoretical results for the stability and error estimates for accuracy. These numerical characteristics are tested on canonical problems including a manufactured solution, steady Kovasznay flow and 2D cylinder flow. An implementation for

quasi-3D, or 2.5D, spatial discretisation is further tested on a cylinder flow. Finally, the linear-implicit scheme's stability, accuracy and performance are tested for a high Reynolds number flow around a multi-element wing in ground effect using a 2D and 3D setup. Next, we discuss the computational performance of the linear-implicit scheme in chapter 4 and compare it against the semi-implicit scheme. The performance analysis further identifies algorithmic bottlenecks and discusses improvements. A summary and conclusions of this work follows in chapter 5.

Chapter 2

Numerical solution of the incompressible Navier-Stokes equations

2.1 The incompressible Navier-Stokes equations

The motion of an incompressible Newtonian fluid in a Eulerian frame of reference is governed by the conservation of momentum and mass equations

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \nu \nabla^2 \mathbf{u} = f, \quad (2.1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2.2)$$

$$\mathbf{u}|_{\Gamma_D} = \mathbf{w}_D, \quad (2.3)$$

$$\left. \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \right|_{\Gamma_N} = \mathbf{w}_N. \quad (2.4)$$

where $\mathbf{u}$ is the d -dimensional velocity field ($d = 2$ or 3), p the scalar pressure field (divided by density $\rho = 1$), ν the kinematic viscosity, f is an external body force, $\mathbf{w}_D$ is the velocity Dirichlet boundary condition and $\mathbf{w}_N$ is the velocity Neumann boundary condition.

We consider solving the set of equations in a volume of fluid $\Omega \in \mathcal{R}^d$ an open bounded domain with bounding surface $\Gamma = \partial\Omega$. Bounding surfaces with Dirichlet boundary condition are specified with Γ_D and those with Neumann conditions with Γ_N . Our aim is to advance

the equations in time from an initial time t^0 to a final time T . The discretised time becomes $t^n = n\Delta t$ with time step index n and the time step size Δt . We denote a variable at time t^n with $\mathbf{u}^n = \mathbf{u}(t^n)$. We are concerned with advancing the primitive variables from a given time instance $(\mathbf{u}^n, p^n)$ to the next instance $(\mathbf{u}^{n+1}, p^{n+1})$. The above equations are supplemented with an initial condition $\mathbf{u}(t^0, \mathbf{x}) = \mathbf{u}_0(\mathbf{x})$ with $\mathbf{x} = (x, y, z) \in \Omega$ that complies with the conditions (2.2) and (2.3). Next, we consider the specific time-marching scheme, known as a splitting scheme, that we consider to advance the Navier-Stokes equations in time from $t^n \rightarrow t^{n+1}$.

2.2 Velocity correction schemes

The general approach in splitting schemes, also known as projection methods [22], decouples the dependent variables in the monolithic formulation of the incompressible Navier-Stokes equations. In particular, velocity-correction schemes separate the solution into two major steps. The first step evaluates the advection term and solves for the new pressure p^{n+1} . The second step solves for the new velocity vector $\mathbf{u}^{n+1}$. This decoupling introduces an intermediate velocity $\tilde{\mathbf{u}}$ field that satisfies the incompressibility condition. It is used to solve for the new pressure using an extrapolation from previous time steps. With the new pressure available, the schemes solve a linear system for each scalar component of the velocity vector $\mathbf{u}$. A commonly employed form of this scheme is known as semi-implicit. The semi-implicit scheme uses an implicit-explicit (IMEX) approach and treats the diffusion term implicitly and advection explicitly.

2.2.1 Derivation of the velocity correction scheme

We begin with the time discrete form of equation (2.1). Approximating the time derivative with the backwards differentiation formula (BDF), the operator becomes

$$\frac{\partial \mathbf{u}}{\partial t} \approx \frac{1}{\Delta t}(\gamma \mathbf{u}^{n+1} - \hat{\mathbf{u}}). \quad (2.5)$$

The J th order BDF determines the term $\hat{\mathbf{u}}$ and factor γ such that

$$\hat{\mathbf{u}} = \sum_{q=0}^{J-1} \alpha_q \mathbf{u}^{n-q} = \begin{cases} \mathbf{u}^n, & \gamma = 1, \quad \text{if } J = 1, \\ 2\mathbf{u}^n - \frac{1}{2}\mathbf{u}^{n-1}, & \gamma = \frac{3}{2}, \quad \text{if } J = 2. \end{cases} \quad (2.6)$$

This leads to the semi-discrete form of equation (2.1)

$$\frac{1}{\Delta t}(\gamma \mathbf{u}^{n+1} - \hat{\mathbf{u}}) + \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} + \nabla p^{n+1} - \nu \nabla^2 \mathbf{u}^{n+1} = f^{n+1}. \quad (2.7)$$

We split the semi-discrete equation into two substeps by introducing an intermediate velocity $\tilde{\mathbf{u}}$ into the time derivative as

$$\frac{1}{\Delta t}(\gamma \tilde{\mathbf{u}} - \hat{\mathbf{u}}) + \nabla p^{n+1} + \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} = f^{n+1}, \quad (2.8)$$

$$\frac{\gamma}{\Delta t}(\mathbf{u}^{n+1} - \tilde{\mathbf{u}}) - \nu \nabla^2 \mathbf{u}^{n+1} = 0. \quad (2.9)$$

We further assume that the intermediate velocity satisfies the incompressibility condition and normal Dirichlet boundary conditions with

$$\nabla \cdot \tilde{\mathbf{u}} = 0, \quad (2.10)$$

$$\mathbf{n} \cdot \tilde{\mathbf{u}}|_{\Gamma_D} = \mathbf{n} \cdot \mathbf{w}_D. \quad (2.11)$$

The standard formulation of the velocity correction scheme in equations (2.8) and (2.9) achieves only first-order in time. Karniadakis et al. [21] find that higher order in time is possible with consistent pressure boundary conditions. They show that the rotational diffusion operator enforces the divergence-free condition and leads to second-order, or higher, temporal accuracy. The diffusion operator enforces incompressibility through the rotational form with

$$\nabla^2 \mathbf{u} = -\nabla \times \nabla \times \mathbf{u} + \underbrace{\nabla(\nabla \cdot \mathbf{u})}_{=0} = -\nabla \times \nabla \times \mathbf{u}. \quad (2.12)$$

For consistency, we add and subtract the rotational diffusion operator to both equations in the split form as

$$\frac{1}{\Delta t}(\gamma \tilde{\mathbf{u}} - \hat{\mathbf{u}}) + \nabla p^{n+1} + \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} + \nu \nabla \times \nabla \times \mathbf{u}^{n+1} = f^{n+1}, \quad (2.13)$$

$$\frac{\gamma}{\Delta t}(\mathbf{u}^{n+1} - \tilde{\mathbf{u}}) - \nu \nabla^2 \mathbf{u}^{n+1} - \nu \nabla \times \nabla \times \mathbf{u}^{n+1} = 0. \quad (2.14)$$

Moreover, the explicit advection and rotational diffusion terms are treated explicitly and are extrapolated to J th order. The extrapolation is denoted with the superscript $\mathbf{u}^{*n+1}$ and is

defined as

$$\mathbf{u}^{\star n+1} = \sum_{q=0}^{J-1} \beta_q \mathbf{u}^{n-q} = \begin{cases} \mathbf{u}^n & \text{if } J = 1, \\ 2\mathbf{u}^n - \mathbf{u}^{n-1} & \text{if } J = 2. \end{cases} \quad (2.15)$$

Note that the non-linear term is extrapolated after evaluating the vector product which we denote with $[\mathbf{u} \cdot \nabla \mathbf{u}]^{\star n+1}$.

We substitute the extrapolation from equation (2.15) in equation (2.13) for the non-linear advection and diffusion term, and reach a strong form equation for updating the pressure to the next time step as

$$\nabla p^{n+1} = -\frac{1}{\Delta t}(\gamma \tilde{\mathbf{u}} - \hat{\mathbf{u}}) - [\mathbf{u} \cdot \nabla \mathbf{u}]^{\star n+1} - \nu \nabla \times \nabla \times \mathbf{u}^{\star n+1} + f^{n+1}, \quad (2.16)$$

$$\nabla \cdot \tilde{\mathbf{u}}^{n+1} = 0, \quad (2.17)$$

$$\mathbf{n} \cdot \tilde{\mathbf{u}}^{n+1}|_{\Gamma_D} = \mathbf{n} \cdot \mathbf{w}_D^{n+1}. \quad (2.18)$$

with consistent pressure boundary conditions of the form

$$\frac{\partial p^{n+1}}{\partial \mathbf{n}} = \mathbf{n} \cdot \left[-\frac{\gamma}{\Delta t} \mathbf{w}^{n+1} - [\mathbf{u} \cdot \nabla \mathbf{u}]^{\star n+1} - \nu \nabla \times \nabla \times \mathbf{u}^{\star n+1} + f^{n+1} \right] \quad (2.19)$$

To find the new velocity field $\mathbf{u}^{n+1}$, we derive an equation that includes the updated pressure p^{n+1} . The velocity equation follows from summing up the split form of equations (2.13) and (2.14). The summation leads to an approximate Navier-Stokes operator

$$\frac{\gamma}{\Delta t} \mathbf{u}^{n+1} + \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} - \nu \nabla^2 \mathbf{u}^{n+1} = \frac{1}{\Delta t} \hat{\mathbf{u}} - \nabla p^{n+1} + f^{n+1}, \quad (2.20)$$

$$\mathbf{u}^{n+1}|_{\Gamma_D} = \mathbf{w}_D^{n+1} \quad (2.21)$$

This operator forms the basis for the new velocity $\mathbf{u}^{n+1}$ which corrects velocity for the new pressure. In combination, the equations (2.16) and (2.20) form a coupled system for the new pressure and velocity $(p^{n+1}, \mathbf{u}^{n+1})$.

2.2.2 The semi- and linear-implicit formulations

The velocity equation in (2.20) couples the velocity components (u, v, w) through the implicit advection term. One approach to decouple the velocities is treating the advection term explicitly. This leads to the semi-implicit velocity equation using an implicit diffusion operator and an

explicit advection term. The semi-implicit equation forms three scalar Helmholtz problems for each velocity component of the form

$$\frac{\gamma}{\Delta t} \mathbf{u}^{n+1} - \nu \nabla^2 \mathbf{u}^{n+1} = \frac{1}{\Delta t} \hat{\mathbf{u}} - [\mathbf{u} \cdot \nabla \mathbf{u}]^{*n+1} - \nabla p^{n+1} + f^{n+1}. \quad (2.22)$$

The semi-implicit scheme has the advantage of having a symmetric and positive-definite matrix with no time dependency of the matrix coefficients. It does however treat advection explicitly which restricts the maximum time step size Δt with a CFL condition.

An alternative form of the velocity equation (2.20) treats the advection term implicitly using a linearised advection term for unconditional stability. This form was initially proposed by Simo and Armero [38] for a coupled scheme and a pressure correction scheme. Later Guermond and Shen [22] proposed the linear-implicit form for a velocity correction scheme and Dong and Shen [39] implement the scheme and test the stability on steady and unsteady flow problems in 2D.

The key idea of the linear-implicit scheme is to linearise the advection operator in equation (2.20) as $[\mathbf{u} \cdot \nabla \mathbf{u}]^{n+1} \approx [\tilde{\mathbf{u}} \cdot \nabla \mathbf{u}]^{n+1}$. This is essentially a Picard iteration for the velocity problem, however, using only a single iteration to reduce computational costs. The choice of advection velocity $\tilde{\mathbf{u}}$ varies as we discuss below. This decouples all components of the velocity field into scalar advection-diffusion-reaction (ADR) problems. At the same time it enables an implicit treatment of advection leading to a velocity equation of the form

$$\frac{\gamma}{\Delta t} \mathbf{u}^{n+1} - \nu \nabla^2 \mathbf{u}^{n+1} + \tilde{\mathbf{u}} \cdot \nabla \mathbf{u}^{n+1} = \frac{\gamma}{\Delta t} \hat{\mathbf{u}} - \nabla p^{n+1} + f^{n+1}. \quad (2.23)$$

This linear system decouples each velocity component equivalently to the semi-implicit form above. Since the ADR matrix includes an advection operator, it does not have the favourable properties of the Helmholtz matrix and instead has indefinite and time-dependent coefficients. Note that the linear-implicit scheme remains linear in the time step and does not require any inner iteration strategy.

There exist two different approaches for the advection velocity $\tilde{\mathbf{u}}$ in the literature. On the one hand, Simo and Armero [38] propose a J th order extrapolation of the previous velocity solutions as

$$\tilde{\mathbf{u}} = \sum_{q=0}^{J-1} \beta_q \mathbf{u}^{n-q}, \quad (2.24)$$

using the coefficients β_q defined in equation (2.15). This approach is straightforward to implement since the previous solution are required for the BDF scheme. We refer to the linear-implicit

scheme with this advection velocity as **extrapolated**. On the other hand, Dong and Shen [39] introduce updated information into the advection velocity and use the intermediate velocity $\tilde{\mathbf{u}}^{n+1}$ for the advection velocity. The intermediate velocity is a residual of the pressure equation (2.16) that leads to

$$\tilde{\mathbf{u}} = \sum_{q=0}^{J-1} \frac{\alpha_q}{\gamma} \mathbf{u}^{n-q} - \frac{\Delta t}{\gamma} (\nabla p^{n+1} + [\mathbf{u} \cdot \nabla \mathbf{u}]^{*n+1} + \nu \nabla \times \nabla \times \mathbf{u}^{*n+1} - \mathbf{f}^{n+1}). \quad (2.25)$$

When we use this advection velocity, the linear-implicit scheme is referred to as **updated**. Section 3.3.1 provides a comparison of both advection velocities identifying their influence on stability and accuracy for the steady Kovasznay flow.

2.2.3 Implementation via weak form

We implement the semi-implicit and linear-implicit velocity correction schemes using the spectral/hp element method. The spectral/hp element method, or another Galerkin-type discretisation, uses the weak form of the above equation for pressure and velocity problems. Therefore, we transform the respective equations using the inner product for the spaces of each basis function.

Pressure Poisson equation

For the pressure equation, we take the L2-inner product of equation (2.16) with the derivative of the basis function $\nabla \phi$ and rearrange for the pressure

$$\int_{\Omega} \nabla \phi \cdot \nabla p^{n+1} = \int_{\Omega} \nabla \phi \cdot \left(\frac{\hat{\mathbf{u}}}{\Delta t} - \frac{\gamma}{\Delta t} \tilde{\mathbf{u}}^{n+1} - [\mathbf{u} \cdot \nabla \mathbf{u}]^{*n+1} - \nu \nabla \times \nabla \times \mathbf{u}^{*n+1} + \mathbf{f}^{n+1} \right) \quad (2.26)$$

$$\forall \phi \in H^1(\Omega).$$

As we pointed out above, the work of Karniadakis et al. [21] found that consistent pressure boundary conditions are required for higher-order in time. Therefore two approaches exist for the pressure boundary conditions via integration by parts. The approaches are referred to as weak and strong pressure form.

The weak form of the pressure equation follows when we integrate the intermediate velocity $\tilde{\mathbf{u}}^{n+1}$ and the diffusion term by parts and apply the divergence theorem. The divergence of the intermediate velocity is zero following the assumption in equation (2.17). Also, the divergence of the diffusion term, that is the curl operator, is analytically zero in the continuous space.

Therefore, the weak pressure equation is

$$\begin{aligned} \int_{\Omega} \nabla \phi \cdot \nabla p^{n+1} &= \int_{\Omega} \nabla \phi \cdot \left(\frac{\hat{\mathbf{u}}}{\Delta t} - [\mathbf{u} \cdot \nabla \mathbf{u}]^{*n+1} + f^{n+1} \right) \\ &\quad - \int_{\Gamma} \mathbf{n} \cdot \left[\phi \left(\frac{\gamma}{\Delta t} \mathbf{w}^{n+1} + \nu \nabla \times \nabla \times \mathbf{u}^{*n+1} \right) \right]. \end{aligned} \quad (2.27)$$

The strong pressure form results from applying integration by parts to the right hand side of equation (2.26). As above, the divergence removes the diffusion term and the intermediate velocity from the volumetric integral. The resulting pressure equation is

$$\begin{aligned} \int_{\Omega} \nabla \phi \cdot \nabla p^{n+1} &= - \int_{\Omega} \phi \nabla \cdot \left(\frac{\hat{\mathbf{u}}}{\Delta t} - [\mathbf{u} \cdot \nabla \mathbf{u}]^{*n+1} - f^{n+1} \right) \\ &\quad - \int_{\Gamma} \mathbf{n} \cdot \left[\phi \left(\frac{\gamma}{\Delta t} \mathbf{w}^{n+1} - \frac{\hat{\mathbf{u}}}{\Delta t} + [\mathbf{u} \cdot \nabla \mathbf{u}]^{*n+1} + \nu \nabla \times \nabla \times \mathbf{u}^{*n+1} - f^{n+1} \right) \right]. \end{aligned} \quad (2.28)$$

Note that the extrapolation of explicit advection and diffusion terms varies depending on using the semi-implicit or linear-implicit velocity correction scheme in both, the weak and strong, pressure formulations. The semi-implicit scheme uses a J th order extrapolation for the explicit advection and rotational diffusion term using equation (2.15). In contrast, the linear-implicit scheme uses a first order extrapolation independent of the order J of the Backwards difference formula. These terms are denoted with the superscript a^{*n+1} .

Velocity equations

The velocity fields use the basis ψ for approximation with $\psi \in H_0^1(\Omega)^d$. The weak form for the semi-implicit and linear-implicit velocity problems transform the second order diffusion operator into the Laplacian using integration by parts

$$\int_{\Omega} \psi \nabla^2 \mathbf{u}^{n+1} = - \int_{\Omega} \nabla \psi \cdot \nabla \mathbf{u}^{n+1} + \int_{\Gamma} \mathbf{n} \cdot [\psi (\nabla \cdot \mathbf{u}^{n+1})]. \quad (2.29)$$

The boundary term is zero due to the incompressibility condition (2.2).

We implement the semi-implicit scheme applying the L2 inner product with equation (2.22), dividing by the kinematic viscosity ν , and using the Laplacian for diffusion

$$\int_{\Omega} \nabla \psi \cdot \nabla \mathbf{u}^{n+1} + \lambda \int_{\Omega} \psi \mathbf{u}^{n+1} = \frac{1}{\nu} \int_{\Omega} \psi \left(\frac{\hat{\mathbf{u}}}{\Delta t} - [\mathbf{u} \cdot \nabla \mathbf{u}]^{*n+1} - \nabla p^{n+1} + \mathbf{f}^{n+1} \right), \quad (2.30)$$

where $\lambda = \frac{\gamma}{\nu \Delta t}$.

The linear-implicit scheme follows the same idea with only the advection term being treated implicitly. Taking the L2 inner product with the basis ψ and equation (2.23), dividing by the kinematic viscosity ν , and using the Laplacian leads to the weak form

$$\begin{aligned} \int_{\Omega} \nabla \psi \cdot \nabla \mathbf{u}^{n+1} + \lambda \int_{\Omega} \psi \mathbf{u}^{n+1} + \int_{\Omega} \frac{1}{\nu} \psi \tilde{\mathbf{u}}^{n+1} \cdot \nabla \mathbf{u}^{n+1} \\ = \frac{1}{\nu} \int_{\Omega} \psi \left(\frac{\hat{\mathbf{u}}}{\Delta t} - \nabla p^{n+1} + \mathbf{f}^{n+1} \right). \end{aligned} \quad (2.31)$$

Algorithms

We implement the above weak-form problems in Nektar++ [9] to leverage the spectral/hp element framework. Figure 2.1 shows the algorithm to advance the Navier-Stokes equations in time for a three dimensional problem. Within this work, we compare the semi-implicit and linear-implicit algorithm's stability, accuracy and performance. Assuming the algorithm is initialised with an appropriate restart procedure, both algorithms initially evaluate the explicit advection and forcing terms. The second step is solving for the new pressure p^{n+1} with either the strong form equation (2.28) or the weak form in equation (2.27). Finally, we update each velocity component with a scalar Helmholtz problem for the semi-implicit or an ADR problem for the linear-implicit algorithm. Having obtained $\mathbf{u}^{n+1}$ and p^{n+1} , we continue from the top.

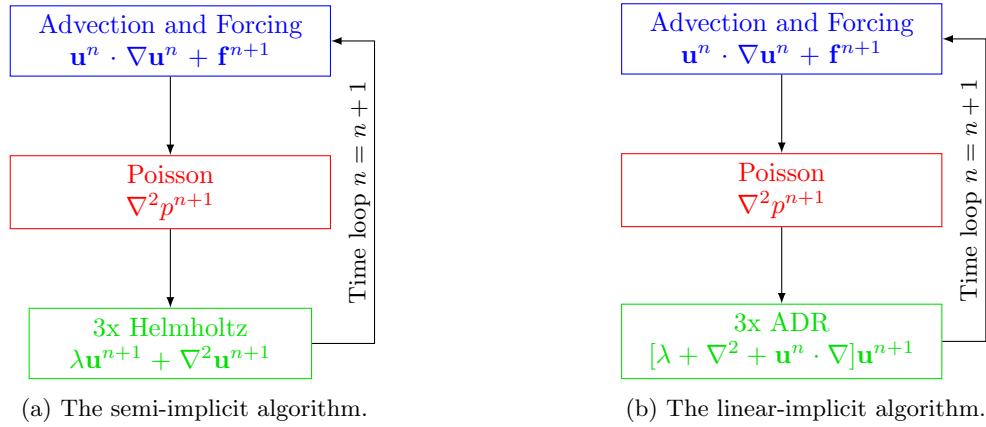


Figure 2.1: The algorithm of the velocity-correction scheme to compute pressure and velocity at the new time $t^{n+1} = (n + 1)\Delta t$.

Next, we look at the spatial approximation for the above weak form of the pressure Poisson and the velocity Helmholtz or ADR problems.

2.3 Spatial discretisation via spectral/hp elements

This section is largely based on the work in Karniadakis and Sherwin [19].

We use the spectral/hp element method to discretise the domain Ω into tessellated subdomains Ω^e . We consider tessellations that create a set of non-overlapping subdomains such that

$$\Omega = \cup_e^{N_e} \Omega^e. \quad (2.32)$$

Each subdomain Ω^e is also known as an element since the tessellation creates a mesh of N_e geometric elements that approximate the geometry of the domain Ω .

We consider the numerical solution of partial differential equations (PDE) with an abstract operator $\mathcal{L}u = 0$ where u is the solution. The operator is defined in the weak sense and, generally, $u|_{\Omega}$ must be smooth and have at least a first-order derivative. Therefore, we require $u|_{\Omega}$ lies in the Sobolev space $W^{1,2}(\Omega)$. The space is formally defined as

$$H^1(\Omega) := W^{1,2}(\Omega) = \{u \in L^2(\Omega) \mid D^\alpha u \in L^2(\Omega) \forall |\alpha| \leq 1\}. \quad (2.33)$$

Assuming a linear operator for $\mathcal{L}$ in the weak form, we express the problem as: *find* $u \in H^1(\Omega)$ *such that*

$$a(u, v) = \ell(v) \quad \forall v \in H^1(\Omega). \quad (2.34)$$

The operator $a(\cdot, \cdot)$ is a bilinear form and $\ell(\cdot)$ is a linear form as is common for the finite element method.

We solve the problem in equation (2.34) for the spatial domain Ω with a finite-dimensional subspace $\mathcal{U}_N \subset H^1(\Omega)$. In this work, we use a continuous Galerkin approach where continuity across element boundaries is enforced and select

$$\mathcal{U}_N = \{u \in C^0(\Omega) \mid u|_{\Omega^e} \in \mathcal{P}_P(\Omega^e)\}. \quad (2.35)$$

The polynomial space $\mathcal{P}_P$ within an element Ω^e defines polynomials of degree P with C^0 continuity across elemental boundaries. This leads to the finite-dimensional form of the weak operator in equation (2.34): *find* $u^\delta \in \mathcal{U}_N$ *such that*

$$a(u^\delta, v^\delta) = \ell(v^\delta) \quad \forall v^\delta \in \mathcal{U}_N. \quad (2.36)$$

The function space $H^1(\Omega)$ satisfies Neumann boundary conditions naturally. We enforce homogeneous Dirichlet boundary conditions by requiring that $\mathcal{U}_N \subset H_0^1(\Omega)$ where

$$H_0^1(\Omega) := \{v \in H^1(\Omega) \mid v(\Gamma) = 0\}. \quad (2.37)$$

Further, we treat inhomogeneous Dirichlet boundary conditions via lifting. Therefore, we decompose our solution into a homogeneous and inhomogeneous solution of the form

$$u^\delta = u^H + u^D, \quad (2.38)$$

where the inhomogeneous contribution satisfies the Dirichlet boundary condition $u^D(\Gamma) = w_D(\mathbf{x})$. Finally, we solve the bilinear form in equation (2.36) for the homogeneous solution u^H and add the inhomogeneous contribution u^D as a forcing term. The lifting approach has the advantage of reducing the size of the linear system, however, requires a numbering system to reorder the matrix which is not required for strong Dirichlet boundary conditions, see Karniadakis and Sherwin [19, §2.2.1.3] for a detailed discussion.

In our case the bilinear form in equation (2.36) is the left-hand side operator of either the pressure Poisson or velocity problems defined in the strong form in equations (2.26), (2.22) and (2.23). The trial functions $u^\delta \in \mathcal{U}_N$ on the other hand approximate the pressure p and the velocity components (u, v, w) within each elemental region Ω^e .

The finite-dimensional solution for the domain Ω is then represented with a weighted sum of trial functions as

$$u^\delta(x) = \sum_p \hat{u}_p \Phi_p(x), \quad (2.39)$$

where $\Phi_p(x) \in \mathcal{U}_N$ are global trial, or *basis*, functions. We find the solution u^δ by solving a linear system for the coefficients $\hat{u}_p$. The global solution however takes contributions from each elemental domain Ω^e . We hence follow the approach of defining piecewise polynomial basis functions $\phi \in \mathcal{P}_P$ on each element and assembling them to form the global solution.

It is further important to consider the specific finite element spaces for approximating the incompressible Navier-Stokes equations. The primitive variable formulation requires defining functional spaces for velocity and pressure. Since the momentum equation (2.1) couples both velocity and pressure, the functional spaces must satisfy a compatibility condition known as Ladyzhenskaya–Babuška–Brezzi (LBB), or inf-sup, condition, see references [50, 51] for details. The inf-sup condition requires a restriction of the pressure space relative to velocity in order to

exclude parasitic modes. This ensures stability and uniqueness of discrete solutions u^δ to the incompressible Navier-Stokes equations. A number of approaches are available including bubble functions or Taylor-Hood approximation spaces [52]. In this work, we employ Taylor-hood spaces which use basis functions with $\psi \in \mathcal{P}_P$ for the velocity space and $\phi \in \mathcal{P}_{P-1}$ for the pressure space.

In the following we first consider the numerical approximation of functions inside an element Ω^e using the spectral/hp element method. Next, we consider the assembly of all elements into a linear system, discuss the issue of under-resolved simulations and introduce the Courant-Friedrichs-Levy (CFL) stability condition.

2.3.1 Spectral/hp elements

The standard approach for finite elements automates numerical operations using a map to a reference, also *standard*, element Ω_{st} . This reduces the memory consumption because information can be reused which favours data locality for computational performance, see also reference [53]. The parametric mapping $\chi^e : \Omega_{st} \rightarrow \Omega^e$ defines the Cartesian coordinate $\mathbf{x} = \chi^e(\xi)$ in terms of the reference coordinate system ξ . The standard element is generally defined as $\Omega_{st} \in [-1, 1]^d$ where d is the dimension of the element.

Name	Domain definition
Segment	$\mathcal{S} = \{\xi_1 \in [-1, 1]\}$
Quadrilateral	$\mathcal{Q} = \{\xi \in [-1, 1]^2\}$
Triangle	$\mathcal{T} = \{\xi \in [-1, 1]^2 \mid \xi_1 + \xi_2 \leq 0\}$
Hexahedron	$\mathcal{H} = \{\xi \in [-1, 1]^3\}$
Prism	$\mathcal{R} = \{\xi \in [-1, 1]^3 \mid \xi_1 \leq 1, \xi_2 + \xi_3 \leq 0\}$
Pyramid	$\mathcal{P} = \{\xi \in [-1, 1]^3 \mid \xi_1 + \xi_3 \leq 0, \xi_2 + \xi_3 \leq 0\}$
Tetrahedron	$\mathcal{A} = \{\xi \in [-1, 1]^3 \mid \xi_1 + \xi_2 + \xi_3 \leq -1\}$

Table 2.1: Definition of reference domains for one, two and three dimensional element types. Adapted from Moxey et al. [9].

The type of element varies with the dimension d and we consider segments in 1D, quadrilateral and triangular elements in 2D and hexahedral, tetrahedral, prismatic and pyramid elements in 3D. The simplex elements are formed from quadrilateral or hexahedral elements with one or more collapsed edges. This requires an additional coordinate system η and map $\Upsilon : \Omega_\eta \rightarrow \Omega_{st}$ with coordinates $\eta \in [-1, 1]^d$. The collapsed domain Ω_η is either that of a quadrilateral in 2D or a hexahedral in 3D. Also, the mapping Υ varies depending on the simplex type. Table 2.1 summarises the domain definitions for each element type. A definition of the maps Υ for each

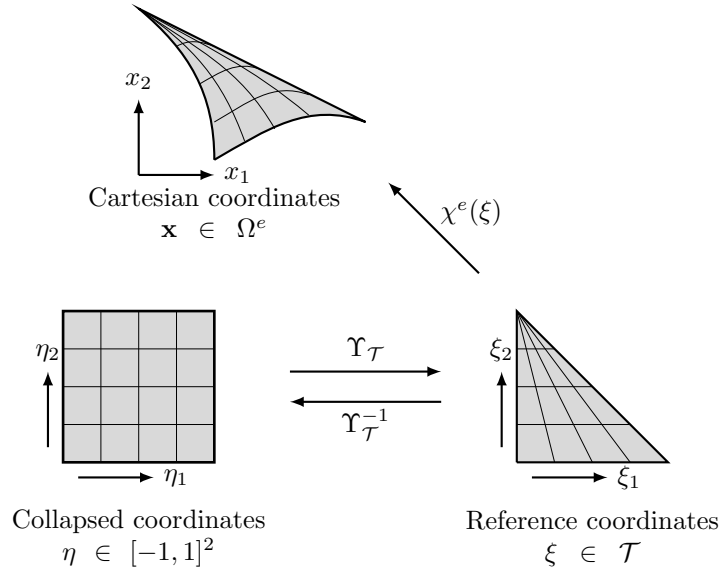


Figure 2.2: Overview of the mapping between collapsed, reference and Cartesian coordinate systems for simplex element types. Adapted from Moxey et al. [9].

element type is available in Moxey et al. [9].

Figure 2.2 shows an example mapping for a Triangle element. The mapping begins with a reference Quadrilateral element in collapsed coordinates which is mapped via Υ to the simplex in the reference coordinate system. Finally, the mapping χ^e transforms coordinates to the Cartesian location of the element Ω^e within the domain Ω .

We evaluate the discrete function u^δ using the mapping to the reference element introduced above as

$$u^\delta(\mathbf{x}) = \sum_{p,q,r}^{N_m} \hat{u}_{pqr} \phi_{pqr} [(\chi^e)^{-1}(\mathbf{x})], \quad (2.40)$$

which is a summation over all $N_m = P + 1$ modes of the polynomial basis function ϕ in three dimensions. The three-dimensional polynomial basis is a tensor product of one-dimensional basis functions ϕ as

$$\phi_{pqr}(\xi) = \phi_p(\xi_1) \phi_q(\xi_2) \phi_r(\xi_3). \quad (2.41)$$

In the following we discuss the choice of basis ϕ and essential numerical operations such as integration and differentiation.

Choice of polynomial basis functions

The choice of an ideal basis ϕ aims at an efficient implementation for solving the pressure and velocity problems introduced in section 2.2.3. Additionally, they should have favourable matrix properties in order to scale the solution to large and challenging problems. We choose polynomial bases ϕ from the family of Jacobi polynomials. Jacobi polynomials have orthogonality properties allowing a low-degree of coupling within elemental matrices. This leads to favourable condition numbers of elemental matrices which makes the solution of linear systems computationally efficient.

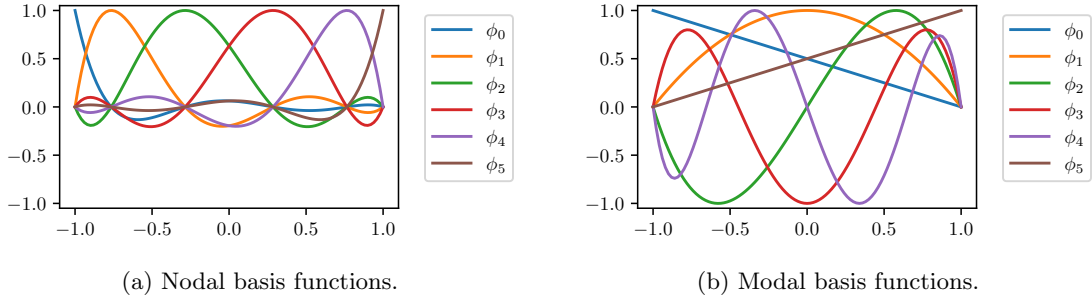


Figure 2.3: Overview of the one-dimensional basis functions for a *nodal* Lagrange polynomial and a *modal* Legendre polynomial for polynomial order $P = 5$.

The higher-order computational fluid dynamics community uses Lagrange and Legendre polynomials extensively for basis functions see references [6–8, 40–42]. Lagrange polynomials offer a *nodal* approach where each polynomial has a value at one node, or degree of freedom (DoF), see figure 2.3a. The combination of Lagrange polynomials and higher-order finite elements is also known as the spectral element method (SEM). The nodal approach has collocation properties leading to a diagonal mass matrix that is cheaply evaluated. Further, the nodal basis is amenable to tensor contractions, also known as *sum factorisation*, for quadrilateral and hexahedral elements. Tensor contractions decrease the amount of operations for function evaluations significantly, see discussion below.

The Legendre basis on the other hand uses *modal*, or *hierarchical*, basis functions as shown in figure 2.3b. These basis functions use polynomials of increasing order. The modes follow a hierarchy where the first and last modes are boundary modes of linear order. Boundary modes are the only modes that are non-zero on the left or right boundary, respectively. Intermediate modes in the hierarchy are *interior* modes. Interior modes are strictly zero on the boundaries and non-zero inside the element Ω^e . The distinction between boundary and interior modes makes the

modal basis amenable to boundary-interior decomposition which enables substructuring, or static condensation, see also section 2.4.1. Additionally, the Legendre basis allows tensor contractions for not only quadrilateral and hexahedral elements but also elements with one or more collapsed edges. This includes triangles in two dimensions and prisms, pyramids and tetrahedron in three dimensions.

In this work, we employ Legendre polynomials to approximate the functional space $\mathcal{U}_N$ defined in equation (2.35). Similar to the nodal basis, the modal basis also allows tensor contractions. While the nodal basis enables tensor contractions on quadrilateral and hexahedral elements, the modal basis additionally permits tensor contractions for simplexes [54]. Currently, approximating complex geometries with only hexahedral elements is an open problem in mesh generation [55]. Instead, complex problems typically require unstructured meshes with simplex elements to approximate challenging geometries. Therefore, we choose a modal basis that allows efficient evaluation for all element types above.

Numerical evaluation of integrals and differentiation

The solution of the pressure and velocity problems in section 2.2 requires numerical integration and differentiation. We employ Gaussian quadrature for the evaluation of integrals. Gaussian quadrature approximates the integrand $u(\xi)$ with, for example, Legendre or Lagrange polynomials using Q points ξ_i . The approximation for an integral in (one-dimensional) standard space becomes a summation of the form

$$\int_{-1}^1 u(\xi) d\xi \approx \sum_{i=0}^{Q-1} w_i u(\xi_i), \quad (2.42)$$

where w_i are quadrature weights and ξ_i are zeros of the polynomial.

The key of Gaussian quadrature is choosing the weights w_i and zeros ξ_i , also known as *quadrature*. Instead of the straightforward choice of equispaced points, Gaussian quadrature uses Gauss-type points. The Gauss-type points include classical Gauss points, Gauss-Radau and Gauss-Lobatto-Legendre (GLL) point distributions. While classical Gauss points exclude the endpoints $\xi = \{-1, 1\}$, Gauss-Radau points include one and GLL points both endpoints, respectively. The advantage of Gauss points is that they enable exact integration, up to machine precision, for polynomial bases of order $P \leq 2Q - 3$. For implementation purposes, it is useful

to formulate this limit as the minimum number of quadrature points for exact integration as

$$Q_{\min} \geq \frac{P+3}{2}. \quad (2.43)$$

We typically employ GLL points, however, use Gauss-Radau points for simplex-type elements where one or multiple edges are collapsed.

The integration for the velocity correction scheme, that is the weak form of pressure and velocity problems in section 2.2.3, requires integration of an inner product of two polynomial bases. The polynomials are of equal or lesser order and their product produces a second order polynomial which requires $Q_{\min} \geq P + \frac{3}{2}$ quadrature points. The second order polynomials are typical for linear problems as in the pressure Poisson and velocity Helmholtz operators which motivates choosing this quadrature. Still, the advection term produces a third order polynomial which requires a quadrature $Q_{\min} \geq \frac{3P}{2} + \frac{3}{2}$ for exact integration, up to machine precision. In practise, we typically choose exact integration up to second order polynomials for a reduction in computational cost. This, however, can lead to aliasing errors when a non-linear term is present, for example, the advection term. We describe dealiasing strategies in section 2.3.3.

Using GLL points for the one-dimensional integration, we extend the formulation in equation (2.41) to the local element Ω^e . Here, we include the change of coordinates via the mapping χ with

$$\int_{\Omega^e} u(\mathbf{x}) d\mathbf{x} = \int_{\Omega_{st}} u((\chi^e)^{-1}(\mathbf{x})) |\mathbf{J}|^{-1} |\partial\xi|. \quad (2.44)$$

Note that the change of coordinates requires evaluating the determinant of the Jacobian $\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \xi}$.

We use numerical differentiation to determine the derivative at a given point ξ_i in physical space. The function $u(\xi)$ is represented with a Lagrangian polynomial ϕ such that we exploit the collocation property and determine the derivative with

$$\left. \frac{\partial u(\xi)}{\partial \xi} \right|_{\xi_i} = \sum_{j=0}^{Q-1} d_{ij} u(\xi_j). \quad (2.45)$$

Similar to numerical integration, we employ Gauss-type points to choose zeros ξ_i and construct the differentiation matrix d_{ij} using the Jacobi Formulae given in [19, Appendix C]. The local formulation of the derivative uses the chain rule for the change of coordinates and requires

evaluating the resulting Jacobian $\mathbf{J}$ as

$$\nabla u = \frac{\partial \xi}{\partial \mathbf{x}} \frac{\partial u}{\partial \xi} = \mathbf{J} \nabla_{\xi} u. \quad (2.46)$$

Leveraging tensor contractions

Another favourable property of spectral methods are tensor contractions as initially recognised by Orszag [56]. The tensor product of the polynomial basis functions in equation (2.41) allows tensor contractions, also known as sum-factorisation. For example, we consider the evaluation of the three-dimensional basis in standard space

$$u^{\delta}(\xi_1, \xi_2, \xi_3) = \sum_p^P \sum_q^P \sum_r^P \hat{u}_{pqr} \phi_p(\xi_1) \phi_q(\xi_2) \phi_r(\xi_3). \quad (2.47)$$

The evaluation cost are of order $\mathcal{O}(P^3)$ for one degree of freedom and, assuming $N_{\text{DoF}} = P$, an $\mathcal{O}(P^6)$ operation for each element.

We reduce the costs by exploiting tensor contractions. The order $\mathcal{O}(P^3)$ summation in equation (2.47) is reduced to

$$u^{\delta}(\xi_1, \xi_2, \xi_3) = \sum_p^P \phi_p(\xi_1) \underbrace{\left\{ \sum_q^P \phi_q(\xi_2) \underbrace{\left[\sum_r^P \hat{u}_{pqr} \phi_r(\xi_3) \right]}_{f_{pq}} \right\}}_{f_p} \quad (2.48)$$

where the sub-summations f_{pq} and f_p effectively reduce the operation count per degree of freedom to order $\mathcal{O}(3P)$. This leads to a total operation count for all DoF of order $\mathcal{O}(P^4)$ instead of $\mathcal{O}(P^6)$.

2.3.2 Matrix assembly

We assemble all elemental operations detailed above into a linear system of equations. The process requires an assembly operator $\mathcal{A}$ that connects all elemental matrices to form a block matrix. We differ between local u_l and global u_g coefficients where local means per element and global refers to the coefficient as part of the assembled block matrix. The assembly operator describes a mapping of the form

$$u_g = \mathcal{A} u_l. \quad (2.49)$$

Note however that the combination of local vertex degrees of freedom makes the operation non-symmetric. That is

$$u_l \neq \mathcal{A}^T u_g. \quad (2.50)$$

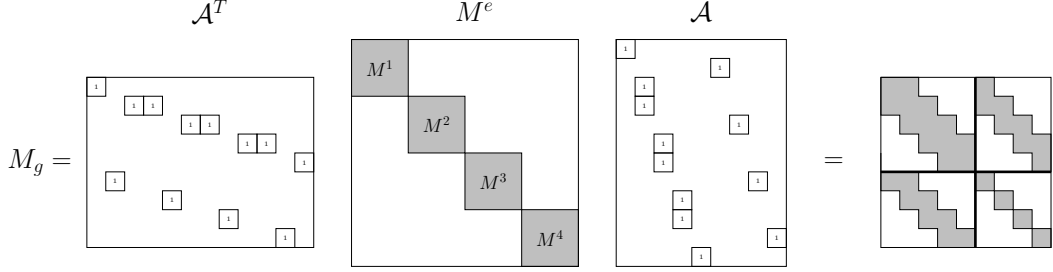


Figure 2.4: Schematic illustration of matrix assembly from elemental matrices M^e into global block matrix with boundary-interior decomposition M_g .

The particular assembly operator for this work uses a boundary-interior decomposition. It assembles the block matrix system into the form presented in figure 2.4. We use this assembly operator to enable static condensation of the matrix system, see section 2.4.1 for more details.

2.3.3 Stabilising under-resolved simulations

The computational cost of high-fidelity simulations, in particular on challenging geometries, limit the feasible number of degrees of freedom N_{DoF} . This effectively constraints the spatial accuracy through the number of mesh elements N_e , their element size h as well as the polynomial order P . The finite accuracy of the unknown variables $\mathbf{u}$ and the multiscale nature of turbulent flows leads to a separation into resolved $\mathbf{u}_H$ and unresolved scales $\mathbf{u}_h$. In practise, the separation line between these scales is ideally in the Kolmogorov scales. However, this is not always possible due to the limited computational resources. As a consequence, the unresolved scales are more energetic and interact with the resolved scales. This is an issue due the non-linear advection term which enables scattering energy from the unresolved scales to the resolved scales and, hence, causing an unbounded growth of energy which, finally, destabilises the algorithm.

There are broadly two approaches to mitigating this instability. The first approach is known as explicit LES (eLES), see reference [11] for a detailed discussion. It uses an explicit model to describe the unresolved scales and their interaction with the resolved scales. Typical candidates are Smagorinsky type models with constant [12] or dynamic coefficients [13]. The other approach is an implicit LES (iLES) also referred to as model-free approach, see reference [15] for details.

An iLES approach attempts to reduce the contamination of the resolved scales by dampening the unresolved scales and their interaction. Candidates, also referred to as filters or stabilisations, include spectral vanishing viscosity (SVV), see Tadmor [17], or gradient jump penalty (GJP) in Moura et al. [57] and Kou et al. [58].

In this work, we employ the Discontinuous Galerkin (DG) mimicking SVV kernel (SVV-DG) from Moura et al. [47]. SVV-DG mimics the favourable dissipation properties of the DG framework. In particular, the upwind type behaviour which removes the effect of wave reflections at mesh expansions. This often occurs at the edge of refinement regions, traps energy locally between high- and low-resolution elements and leads to an unbounded growth of energy. The SVV-DG kernel uses SVV coefficients optimised to fit the DG-dissipation behaviour with a polynomial order $P - 2$. The SVV coefficients for polynomial order $P \in [3, 10]$ are tabulated in Moura et al. [47].

The kernel $\mathcal{Q}^\star$ applies a scaling to the diffusion operator of the form

$$\int_{\Omega^e} \nabla \phi \cdot \mathcal{Q}^\star [\nabla u], \quad (2.51)$$

where $\mathcal{Q}^\star = \mathcal{T}^{-1} \mathcal{Q} \mathcal{T}$ is the matrix operator that applies the filter coefficients $\mathcal{Q}$. The kernel incorporates a mapping $\mathcal{T}$ to the orthonormal set of Legendre basis functions which are popular in modal DG approaches. The diagonal matrix $\mathcal{Q}$ contains the filter coefficients given in Moura et al. [47]. The SVV-DG stabilisation is part of the Laplacian matrix, as shown in equation (2.51), and is applied to the velocity Helmholtz equation (2.30) or ADR equation (2.31) as

$$\mathcal{Q}^\star [\nabla u] = \mathcal{T}^{-1} \mathcal{Q} [\nabla u] \mathcal{T}. \quad (2.52)$$

A robust outflow boundary condition

The finite computational resources further require truncating the domain Ω . The truncation necessarily removes part of the physical processes. In external aerodynamics, this typically appears in moderate to high Reynolds number flows where energetic wake structures move far downstream without significant dissipation. To avoid building large computational meshes that cover the entire range until dissipation of these structures, domains are typically truncated. The truncation potentially leads to trapped energy on the outflow boundaries when energetic structures reach the outflow.

Dong et al. [59] describes a robust boundary condition that stabilises the outflow boundary

for truncated domains. The boundary condition allows the inflow of kinetic energy at the outflow and limits the stress on the boundary to avoid unbounded growth. They further show that this boundary condition allows severe truncation of the domain without influence on the physics.

The boundary condition applied to the rotational velocity-correction scheme above is applied to the pressure and velocity separately. On the outflow boundary Ω_o , the pressure and velocity boundary conditions are, respectively,

$$p_{\text{HO}}^{n+1} = \nu \mathbf{n} \cdot \nabla \mathbf{u}^{*n+1} \cdot \mathbf{n} - \frac{1}{2} |\mathbf{u}^{*n+1}|^2 S_0(\mathbf{n} \cdot \mathbf{u}^{*n+1}), \quad (2.53)$$

$$\mathbf{n} \cdot \nabla \mathbf{u}^{n+1} = \frac{\partial \mathbf{u}^{n+1}}{\partial \mathbf{n}}_{\text{HO}} = \frac{1}{\nu} \left[p^{n+1} \mathbf{n} + \frac{1}{2} |\mathbf{u}^{*n+1}|^2 S_0(\mathbf{n} \cdot \mathbf{u}^{*n+1}) - \nu (\nabla \cdot \mathbf{u}^{*n+1} \mathbf{n}) \right]. \quad (2.54)$$

where the scalar function $S_0(\mathbf{n} \cdot \mathbf{u})$ is defined as

$$S_0(\mathbf{n} \cdot \mathbf{u}) = \frac{1}{2} \left(1 - \tanh \frac{\mathbf{n} \cdot \mathbf{u}}{U_0 \delta} \right). \quad (2.55)$$

This is a smoothed step function with characteristic velocity scale $U_0 = 1$ and a tuning parameter δ . The tuning parameter defines a step function with $\delta = 0$ and a smooth step function with $\delta > 0$. The subscript *HO* indicates the use of the outflow boundary condition which we use for complex problems in section 3.5.3 and 3.5.4.

Dealiasing techniques

Dealiasing is a broad term that tackles aliasing errors in numerical methods. These errors arise either due to the physics of the problem or the numerical approximation and potentially lead to unstable algorithms. We discussed under-resolution, or *physical aliasing*, in section 2.3.3 and introduce the iLES approach with a DG-SVV kernel that stabilises physical aliasing errors. Further aliasing effects occur due to approximations in the numerical method also referred to as *numerical aliasing*. Numerical aliasing summarises aliasing errors that occur in the evaluation of integrals via Gaussian quadrature, see appendix A in reference [60] for a detailed description.

The issue of numerical aliasing is prominent in higher-order methods. An increase in polynomial order requires an increase in quadrature points for exact integration, up to machine precision, see section 2.3. In practise, exact integration becomes computationally expensive due to the large amounts of DoF required for non-linear terms. In order to reduce computational cost, it is common to use exact integration for linear terms, see references [26–34]. As a consequence, nonlinear terms, such as the advection operator, potentially destabilise the algorithm.

Mengaldo et al. [61] investigate two strategies to mitigate the aliasing effect of the advection term. One approach uses higher quadrature locally for the evaluation of the explicit advection term. The second approach uses a higher quadrature Q globally for the polynomial basis functions such that the non-linearity is exactly integrated. In this work, we use the second approach with higher quadrature. Note that we equally apply higher quadrature for both the linear-implicit and semi-implicit scheme in the results in chapter 3.

The required quadrature for the basis functions ϕ uses exact quadrature for a third order polynomial. Following Mengaldo et al. [61], the number of quadrature points for a second order non-linearity is $Q = P + \frac{P_{\text{adv}}}{2} + \frac{3}{2}$ where P_{adv} is the polynomial order of the non-linearity. The linearised advection operator and explicit advection operator in the velocity correction schemes use an advection velocity based on the velocity field $\mathbf{u}$, thus, $P_{\text{adv}} = P$. Consequently, the quadrature is $Q = \frac{3P}{2} + \frac{3}{2}$. Note that for a curved mesh, that is a non-linear mapping from reference to local element, an increase in quadrature is required to avoid geometric aliasing.

2.3.4 CFL condition

The numerical stability of an explicit time-stepping scheme depends on the eigenvalue spectrum of the discretised partial differential equation. If the eigenvalues lie outside of the stability region of the time integration scheme, the algorithm is unstable. For the Navier-Stokes equations, this condition is known as Courant-Friedrichs-Lewy (CFL) condition.

The CFL condition describes a stability limit that depends on the time step size Δt , the mesh spacing h and a constant c based on the eigenvalues of the matrix. For spectral/hp element method, the constant $c = CP^N$ varies with the polynomial order P of the basis. Moreover, the constant CP^N scales with the maximum eigenvalue λ_{max} that is $CP^N \propto \lambda_{\text{max}}$. Numerical stability requires the CFL number to satisfy

$$CFL = \frac{CP^N}{h} \Delta t \leq CFL_{\text{crit}}. \quad (2.56)$$

The practical limit CFL_{crit} depends critically on the temporal discretisation. For the incompressible Navier-Stokes equations, the limit depends particularly on the explicit or implicit treatment of the diffusion and advection term.

The diffusion operator's maximum eigenvalues grow with $\lambda_{\text{max}} \propto \mathcal{O}(P^4)$ which severely restricts the time step size. Therefore, most time-stepping schemes use a semi-implicit formulation with implicit diffusion and explicit advection term. In contrast, the advection operator shows

an eigenvalue growth of $\lambda_{\max} \propto \mathcal{O}(P^2)$ which makes explicit treatment more practical. Still, the CFL restriction becomes limiting for high Reynolds number flows over complex geometries where intermediate polynomial orders $P \in [3, 5]$ and small mesh spacings h are required to resolve physical flow features. For these cases, the maximum eigenvalues $\lambda_{\max}$ become large enough such that the explicit treatment of the advection term limits the time step size.

Karniadakis and Sherwin [19] describe an estimator for the CFL condition using the spectral/hp element method with a C0 continuous modified basis. They estimate the elemental CFL number CFL^e based on the approximate maximum eigenvalue of the elemental matrix. The estimation uses

$$CFL^e \approx \max_{e=1, N_e} |V^e| \frac{c_\lambda P^2}{h} \Delta t \quad (2.57)$$

where V^e is the elemental velocity in the reference space, $c_\lambda = 0.2$ is a constant, P is the polynomial order of the element and h is a measure for the element size.

Using the local CFL estimator in equation (2.57) and the semi-implicit formulation, we find a practical stability limit for a time-averaged value of $CFL_{\text{crit}} \in [0.1, 1.0]$. In comparison, the linear-implicit velocity correction scheme shows a practical limit of $CFL_{\text{crit}} \in [100, 200]$.

2.4 Solving the linear system

We evolve the incompressible Navier-Stokes equations in time by solving the velocity correction scheme algorithm in figure 2.1. The algorithm updates the pressure via the weak form Poisson problem in equation (2.27) and the velocity components via the Helmholtz and ADR problems in equation (2.30) and (2.31), respectively. Upon discretising the weak forms with the spectral/hp element method, we have $d + 1$ scalar matrix problems of the form

$$\mathbf{A}x = b \quad (2.58)$$

which we solve for the new pressure p^{n+1} and velocity components $(u^{n+1}, v^{n+1}, w^{n+1})$. Next, we discuss two approaches for building the matrix $\mathbf{A}$ and an efficient solver to find the solution x .

2.4.1 Approach for building the matrix

The classic approach in solving linear systems consists of building the left-hand side matrix $\mathbf{A}$, also known as *full* matrix. This approach incurs practical limitations on the problem size. Consider figure 2.5a, we show an example matrix for a 4x4 quadrilateral mesh and a polynomial

order $P = 6$ for a Poisson problem. Non-zero matrix entries are highlighted in black. The example matrix has 279841 entries of which only 3321 entries or 1.1% have a nonzero coefficient. This highlights the nature of the matrix size scaling with $\mathcal{O}(N_{\text{DoF}}^2)$ where N_{DoF} is the number of global degrees of freedom (DoF). In practise, solvers use sparse matrices which store only non-zero matrix coefficients along with their row indices reducing the memory requirements and, hence, increasing the possible problem size.

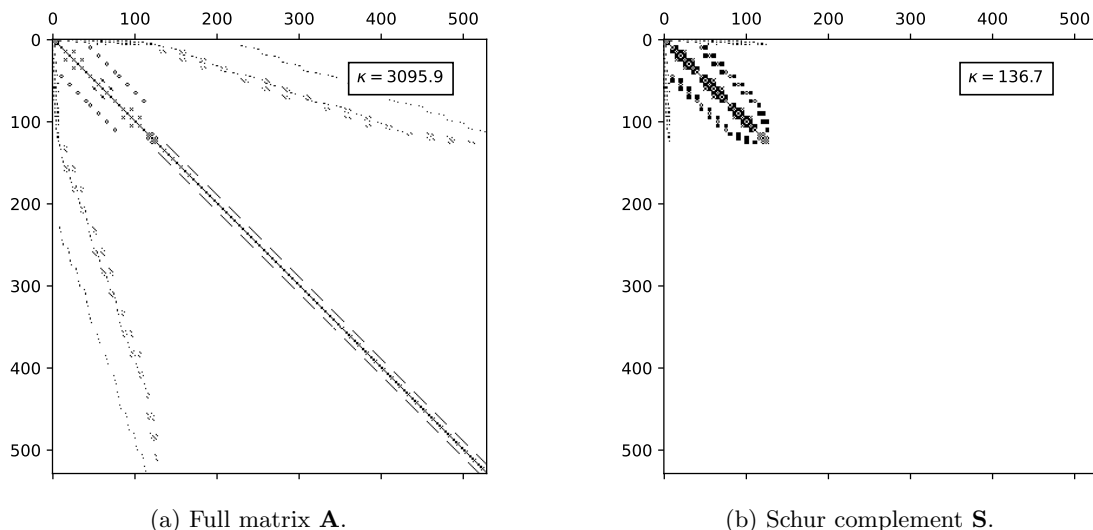


Figure 2.5: The full matrix $\mathbf{A}$ and the statically condensed Schur complement matrix $\mathbf{S}$ for a pressure Poisson problem on a 4x4 domain of quadrilateral elements. The condition number κ shows an order of magnitude difference for this canonical mesh.

The matrices for the splitting scheme become unreasonably large and ill-posed for scale-resolving simulations on large-scale unstructured meshes. Further improvements are possible with a transformation of the full matrix $\mathbf{A}$ using substructuring, or static condensation [19]. Substructuring exploits the boundary-interior decomposition of the basis functions to decouple interior and boundary DoF. The statically condensed matrix is called the Schur complement $\mathbf{S}$. It reduces the matrix size, that is memory requirements, and, additionally, reduces the condition number κ that is the ratio of the maximum to minimum eigenvalue. The condition number is relevant for solving the linear system iteratively, see section 2.4.2.

For substructuring, we decompose the original linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ into a boundary and interior subsystem of the form

$$\begin{bmatrix} \mathbf{A}_b & \mathbf{A}_c^T \\ \mathbf{A}_c & \mathbf{A}_i \end{bmatrix} \begin{bmatrix} \mathbf{x}_b \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}_b \\ \mathbf{f}_i \end{bmatrix} \quad (2.59)$$

where subscripts b and i denote the boundary and interior DoFs respectively. The subscript c denotes the coupling matrices. This coupling can effectively be removed by transforming the matrix system in equation (2.59) into a Schur complement $\mathbf{S}$ of the form

$$\mathbf{S}\mathbf{x}_b = \mathbf{f}_S = \mathbf{f}_b - \mathbf{A}_c^T \mathbf{A}_i^{-1} \mathbf{f}_i \quad (2.60)$$

where the Schur complement $\mathbf{S} = [\mathbf{A}_b + \mathbf{A}_c^T \mathbf{A}_i^{-1} \mathbf{A}_c]$ and $\mathbf{f}_S$ is a short-hand for the forcing term. Finally, we update the interior DoF directly evaluating the expression

$$\mathbf{x}_i = \mathbf{A}_i^{-1} \mathbf{f}_i - \mathbf{A}_i^{-1} \mathbf{A}_c^T \mathbf{x}_b. \quad (2.61)$$

Note that while building the Schur complement requires inverting the interior-interior matrix $\mathbf{A}_i$, solving the system in equation (2.60) is the major effort for finding the solution $\mathbf{x}$.

2.4.2 Linear system solver

The straightforward approach for solving the matrix problem in equation (2.60) is a direct solver. Direct solvers provide an exact solution, up to machine accuracy by inverting the matrix $\mathbf{S}$. Standard methods such as the LU or Cholesky decomposition require communication between all parallel partitions causing a large communication overhead. Optimised solvers such as the XXT library in reference [62] minimise the communication with a sparse parallel Cholesky decomposition. Still, the parallel efficiency constraints scaling on three-dimensional meshes when moving from hundreds to thousands of processors. Instead, we consider iterative solvers using Krylov subspace methods which require less communication and scale efficiently on distributed-memory high performance computers to exascale [8].

On the one hand, the discrete Poisson and Helmholtz problems are symmetric positive-definite matrices with constant coefficients. There are several efficient iterative solvers available for this type of linear system [63, 64]. We employ a conjugate gradient (CG) [65] solver for these matrix systems. On the other hand, the advection operator in the ADR problem forms a non-symmetric matrix. Two prominent options are the GMRES algorithm which is widely employed due to its robustness [10, 42, 66] and the Biconjugate gradient stabilised (BiCGStab) algorithm which is cheaper in communication costs and memory requirements. The communication, a consequence of global reductions, are constant for BiCGStab with four or two, if fused for each iteration. In comparison, the global reductions for GMRES grow linearly with the number of iterations.

Further, the BiCGStab algorithm uses a constant amount of memory whereas the GMRES algorithm requires more memory depending on the number of iterations. We employ the GMRES algorithm for the robustness on ill-conditioned problems and consider restarting the algorithm every m steps to reduce the memory consumption.

An iterative solver aims to reduce the residual of the solution r to be approximately zero after N_{it} iterations. The residual at the k th iteration is defined as

$$\mathbf{r}^k = \mathbf{f}_S - \mathbf{S}\mathbf{x}_b^k. \quad (2.62)$$

Standard criteria measure the relative reduction of the residual's L_2 norm at the k th iteration as

$$\|\mathbf{r}^k\|_2 < \tau \|\mathbf{r}^0\|_2 = \tau \|\mathbf{f}_S\|_2, \quad (2.63)$$

where τ is a user-defined tolerance. We find that this estimator is problem-dependent and requires fine-tuning for different geometries. Instead we employ the *absolute* criterion for all tests

$$\|\mathbf{r}^k\|_2 < \tau. \quad (2.64)$$

2.4.3 Preconditioning

The number of iterations required to satisfy the stopping criteria above depends on the conditioning of the Schur complement $\mathbf{S}$. The condition number κ , that is the ratio of the largest to the smallest eigenvalue, indicates the conditioning of the matrix. Ideally the condition number is close to 1, however, in practical problems the Schur complement is ill-conditioned with $\kappa \gg 1$ and the iterative solvers require large number of iterations to converge. An effective method to further improve the conditioning is applying a preconditioner $\mathbf{M}$ that approximates the Schur complement as

$$\mathbf{M}^{-1} \approx \mathbf{S}^{-1}. \quad (2.65)$$

We consider two different preconditioners for the pressure and the velocity problems.

Pressure preconditioner

The pressure system is preconditioned with a Jacobi approach, that is the diagonal of the matrix system. In the context of substructuring, this preconditioner uses the inverse of the diagonal

components of the Schur complement as

$$\mathbf{M}^{-1} = \mathbf{S}_{ii}^{-1}. \quad (2.66)$$

The Jacobi preconditioner is cheap in construction, however, most effective for diagonally-dominant matrices.

Velocity preconditioner

The velocity system is usually ill-conditioned when we apply it to high Reynolds number flows on complex geometries. This is a particular issue for three-dimensional problems where the coupling of vertex and edge modes with face modes leads to severe ill-conditioning of the matrix. In order to improve the conditioning, we consider the low-energy block preconditioner for substructured solvers introduced in reference [67]. The preconditioner transforms the Schur complement $\mathbf{S}$ from the original basis to a low-energy basis. The low-energy basis reduces the coupling of vertex and edge modes from face modes leading to a more diagonally-dominant Schur complement. This makes the transformed matrix amenable to block preconditioning and Sherwin and Casarin [67] suggest potentially sublinear growth of the condition number using this approach. Therefore, we use this favourable preconditioner for the velocity problem.

Reference [67] proposes the preconditioner for elliptic systems, in particular, for Helmholtz problems such as the semi-implicit scheme's velocity equation (2.22). We employ the preconditioner for the linear-implicit scheme as well, however, construct the matrix based on the Helmholtz system and ignore the advection matrix for preconditioner construction.

The low-energy block preconditioning requires three essential steps:

1. construct the transformation matrix $\mathbf{R}$ for change of basis,
2. build the block preconditioner in the low-energy basis,
3. transform the Schur complement $\mathbf{S}$ to the new basis.

The transformation matrix $\mathbf{R}$ transforms the Schur complement from the original basis $\mathbf{S}_1$ to the low-energy basis as

$$\mathbf{S}_2 = \mathbf{R}\mathbf{S}_1\mathbf{R}^T. \quad (2.67)$$

We construct the transformation using the boundary-interior decomposition of the modal basis

function as

$$\mathbf{R} = \begin{bmatrix} \mathbf{I} & \mathbf{R}_{ve} & \mathbf{R}_{vf} \\ 0 & \mathbf{I} & \mathbf{R}_{ef} \\ 0 & \mathbf{I} & \mathbf{I} \end{bmatrix}. \quad (2.68)$$

Here the modification of vertex modes by edge and face modes is applied through $\mathbf{R}_{ve}$ and $\mathbf{R}_{vf}$, respectively. Similarly, the edge mode modification uses $\mathbf{R}_{ef}$. Next, we construct the block preconditioner based on the transformed Schur complement $\mathbf{S}_2$ as

$$\mathbf{M}^{-1} = \begin{bmatrix} (\mathbf{S}_2^{-1})_{vv} & 0 & 0 \\ 0 & (\mathbf{S}_2^{-1})_{eb} & 0 \\ 0 & 0 & (\mathbf{S}_2^{-1})_{ef} \end{bmatrix}. \quad (2.69)$$

Finally, we transform the Schur complement $\mathbf{S}_1$ in the original basis to the low-energy basis. This last step is redundant, within the Nektar++ framework, however, it is necessary to construct the linear system and the preconditioner independently.

2.4.4 Freezing the time-dependent matrix

The main difference between the semi-implicit and linear-implicit scheme is the advection operator in the velocity matrix, compare equation (2.22) and (2.23). The linear-implicit velocity system has time-dependent matrix coefficients and hence requires re-computation of the preconditioner and ADR matrix at every step. The re-computation is the major cost overhead compared to the semi-implicit scheme, see chapter 4. Additionally, we observe that the iteration counts for the velocity system are stable for long physical time periods. This suggests using a freezing strategy where we do not re-compute the preconditioner and/or the ADR matrix for a number of steps N_{freeze} in order to reduce the computational cost.

Preconditioner freezing

The preconditioner requires a few operations transforming the Schur complement and constructing the low-energy block preconditioner. We consider freezing the preconditioner for N_{freeze} time steps and investigate the effect on the performance. Fortunately, the preconditioner does not influence the stability or accuracy of the scheme and, hence, the number of steps is freely chosen based on computational performance.

Advection freezing

Dong and Shen [39] suggest a freezing approach to reduce the computational overhead. Assuming we have computed the velocity for a time step u^{n+0} and that we want to freeze the advection matrix for N_{freeze} steps. The computation of $u^{n+N_{\text{freeze}}}$ uses simple algebraic modifications for the freezing approximation

$$\tilde{\mathbf{u}}^{n+N_{\text{freeze}}} \cdot \nabla \mathbf{u}^{n+N_{\text{freeze}}} \approx \tilde{\mathbf{u}}^{n+0} \cdot \nabla \mathbf{u}^{n+N_{\text{freeze}}} + (\tilde{\mathbf{u}}^{n+N_{\text{freeze}}} - \tilde{\mathbf{u}}^{n+0}) \cdot \nabla \mathbf{u}^{*n+N_{\text{freeze}}}.$$

The first term on the right hand side is the approximated advection using the advection velocity of the previous time step $\tilde{\mathbf{u}}^{n+0}$. The second term adds forcing for the deficit in advection between the frozen velocity $\tilde{\mathbf{u}}^{n+0}$ and the current velocity $\tilde{\mathbf{u}}^{n+N_{\text{freeze}}}$. The extrapolation is performed with as defined in equation (2.15).

The advection freezing strategy modifies the velocity problem in equation (2.23) to the frozen velocity system for time step t^{n+1} as

$$\frac{\gamma}{\Delta t} \mathbf{u}^{n+1} - \nu \nabla^2 \mathbf{u}^{n+1} + \tilde{\mathbf{u}}^{n+0} \cdot \nabla \mathbf{u}^{n+1} = f^{n+1} + \frac{\gamma}{\Delta t} \hat{\mathbf{u}} - \nabla p^{n+1} - (\tilde{\mathbf{u}}^{n+N_{\text{freeze}}} - \tilde{\mathbf{u}}^{n+0}) \cdot \nabla \mathbf{u}^{*n+1}. \quad (2.70)$$

The implementation follows analogous to equation (2.31).

Advection freezing is not demonstrated in [39], so, we extend there study and test the performance for challenging problems.

2.5 Extension to quasi-3D Problems

This section is based on the work in [68].

The simulation of geometries with one homogeneous direction enables straightforward parallelisation [69]. The idea is based on expanding the solution into the z -direction, but decouple solutions along discrete z -coordinates. The physical-space velocity is therefore transformed with a Fourier expansion

$$\underbrace{\mathbf{u}(x, y, z)}_{\text{Physical space}} = \sum_{k=0}^{N_z} \underbrace{\left[\hat{u}_k(x, y), \quad \hat{v}_k(x, y), \quad \hat{w}_k(x, y) \right]^T}_{\text{Fourier space}} \phi_k(z) \phi(x, y), \quad (2.71)$$

where ϕ_k is the Fourier basis and $\phi(x, y)$ represents any Galerkin-type basis function which we will omit in the following extension.

Nektar++ uses an harmonic expansion for the Fourier basis with a real cosine/sine series

$$\phi_k(z) = \cos(\beta_k z) + \sin(\beta_k z) \quad (2.72)$$

where the wave number is $\beta_k = 2\pi k/L_z$ and L_z is the domain length in z -direction. The Fourier coefficients u_k are decomposed into a corresponding coefficient $\hat{u}_k^c$ and $\hat{u}_k^s$ for the cosine and sine mode, respectively. The superscripts will be used for brevity to refer to either of the half modes where necessary.

The linear-implicit scheme presented above requires some adaptation for the extension to the Fourier basis. Considering the harmonic expansion in equation (2.71), the first- and second-order derivatives for scalar velocity components are

$$\nabla u = \sum_k [\nabla_{xy}, -\beta_k] \hat{u}_k^s \cos(\beta_k z) + [\nabla_{xy}, +\beta_k] \hat{u}_k^c \sin(\beta_k z), \quad (2.73)$$

$$\nabla^2 u = \sum_k (\nabla_{xy}^2 - \beta_k^2) \hat{u}_k^c \cos(\beta_k z) + (\nabla_{xy}^2 - \beta_k^2) \hat{u}_k^s \sin(\beta_k z), \quad (2.74)$$

where the two-dimensional derivative is denoted as $\nabla_{xy} = [\partial_x, \partial_y]$. The first-order derivative highlights the coupling of cosine and sine modes in equation (2.73) where the cosine coefficient $\hat{u}_k^c = -\beta_k \hat{u}_k^s$ and the sine coefficient is $\hat{u}_k^s = \beta_k \hat{u}_k^c$. Consequently, it is beneficial to hold both coefficients $\hat{u}_k^c, \hat{u}_k^s$ in the memory of each processor to avoid communication when evaluating derivatives.

The influence of the quasi-3D approach on solving the velocity-correction scheme becomes clear upon substitution of the Fourier basis. The pressure equation (2.26) becomes

$$\begin{aligned} \sum_{k=0}^{N_z} \left[[\nabla_{xy}^2 - \beta_k^2] \hat{p}_k^{c,n+1} = -\frac{1}{\Delta t} \left(\gamma \hat{\mathbf{u}}_k^{c,n+1} - \sum_q \alpha_q \hat{\mathbf{u}}_k^{c,n-q} \right) \right. \\ \left. - [\mathbf{u} \cdot \nabla \mathbf{u}]_k^{c,n} - \nu \nabla \times \hat{\omega}_k^{c,n} + \hat{\mathbf{f}}_k^{c,n+1} \right] \cos(\beta_k z), \end{aligned} \quad (2.75)$$

effectively modifying the Laplacian to a Helmholtz equation with substituted Fourier basis due to the second-order derivative in the diffusion operator (2.74). Note that we only discuss the cosine mode formulation here, because it is equivalent to the sine mode. As for the explicit right-hand side terms, the first and last term are independently evaluated. Further, the rotational diffusion operator only couples within a full mode and as such its implementation is straightforward. The non-linear advection term however couples the complete Fourier space.

The advection term is the product of two Fourier expansions. One for the advection velocity $\mathbf{u}_\ell$ and another for the advected (scalar) field u_k . The product hence leads to a convolution matrix of the form

$$\left(\sum_{\ell=0}^{N_z} \mathbf{u}_\ell \phi_\ell \right) \cdot \left(\sum_{k=0}^{N_z} \nabla u_k \phi_k \right) = \sum_{\ell=0}^{N_z} \sum_{k=0}^{N_z} \mathbf{u}_\ell \cdot \nabla u_k \phi_{\ell+k} =$$

$k \backslash \ell$	0	1	2	$\dots$	N_z
0	$\mathbf{u}_0 \cdot \nabla u_0$	$\mathbf{u}_1 \cdot \nabla u_0$	$\mathbf{u}_2 \cdot \nabla u_0$	$\dots$	$\mathbf{u}_{N_z} \cdot \nabla u_0$
1	$\mathbf{u}_0 \cdot \nabla u_1$	$\mathbf{u}_1 \cdot \nabla u_1$	$\mathbf{u}_2 \cdot \nabla u_1$	$\dots$	$\mathbf{u}_{N_z} \cdot \nabla u_1$
2	$\mathbf{u}_0 \cdot \nabla u_2$	$\mathbf{u}_1 \cdot \nabla u_2$	$\mathbf{u}_2 \cdot \nabla u_2$	$\dots$	$\mathbf{u}_{N_z} \cdot \nabla u_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
N_z	$\mathbf{u}_0 \cdot \nabla u_{N_z}$	$\mathbf{u}_1 \cdot \nabla u_{N_z}$	$\mathbf{u}_2 \cdot \nabla u_{N_z}$	$\dots$	$\mathbf{u}_{N_z} \cdot \nabla u_{N_z}$

(2.76)

The matrix (2.76) contains N_z^2 terms to compute. Each term incorporates 6 matrix-vector products for the 2D gradients in the x - y plane of each variable as well as 9 vector-vector products to form the advection terms. This rapidly growing number of computations becomes unfeasible even at modest number of modes N_z . Reference [69] instead proposes to evaluate the advection terms in physical space by first transforming the variables back to physical space using an inverse discrete Fourier transformation (DFT). Then, the entire advection term requires 9 matrix-vector products independent of the number of modes N_z . This approach requires communicating the coefficients $\hat{u}_k$ for the DFT, which eases the computational burden on each processor for an increase in communication which potentially limits the scalability of the algorithm.

2.5.1 Decoupling the linear-implicit advection operator

The above strategy works for the explicit advection terms, however, not for an implicit advection term. Considering the velocity problem in equation (2.31), the linearised advection terms couple Fourier modes and should not be evaluated in physical space to avoid additional DFTs and their expensive communication. Based on the literature survey of industrial flow problems in figure 2.6, we suggest avoiding this communication by choosing only part of the Fourier modes to be

implicit.

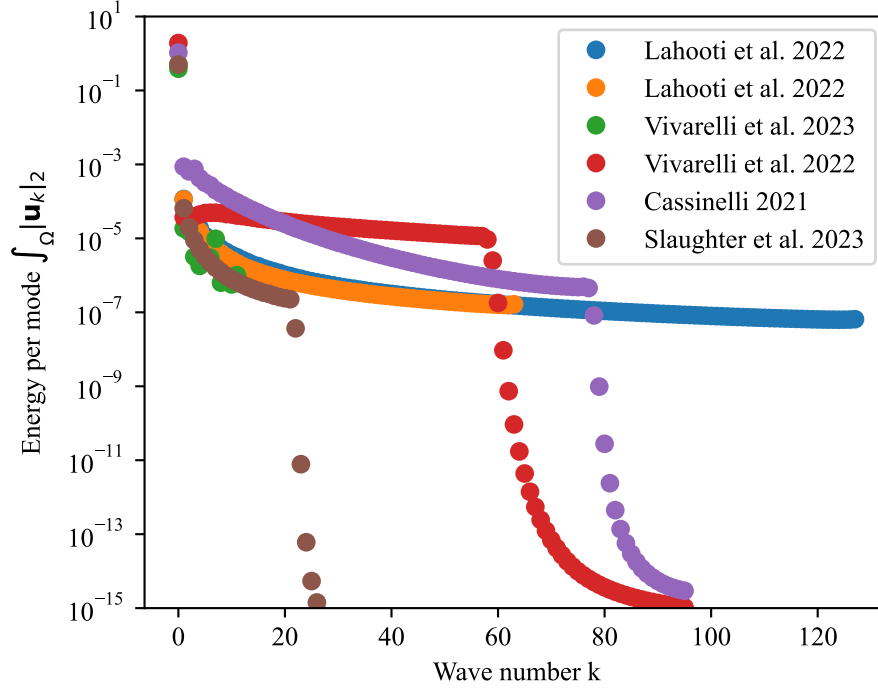


Figure 2.6: The distribution of energy across discrete Fourier modes on various industrial flow problems at high Reynolds number.

Figure 2.6 shows the distribution of energy across discrete Fourier modes in various industrial flow problems. The distribution of energy across the Fourier modes in high Reynolds number simulations shows a distinct pattern: the majority of the energy is contained in the mean mode $k = 0$. The simulations usually require higher-frequency modes for resolving spanwise fluctuations, however, the additional modes contain only a small amount of the energy. Reference [32] shows quasi-3D simulations of a Formula 1 front wing geometry at a Reynolds number of 2.2×10^5 . The brown curve of the energy distribution across the modes in figure 2.6 highlights the gap in magnitude from the mean-mode ($k = 0$) to higher modes ($k > 0$). Note that the energy is plotted with logarithmic vertical axis and shows an energy ratio of mean mode to any higher frequency mode of $\frac{E_0}{E_k} \geq \mathcal{O}(10^3)$. References [30, 70–72] show similar variations for high Reynolds number simulations of turbulent flows and a similar margin in the energy ratio between $[\mathcal{O}(10^3), \mathcal{O}(10^4)]$. Thus, we propose to treat only the energetic mean mode ($k = 0$) implicit. The higher-frequency modes ($k > 0$) are treated explicit in order to build an efficient and scalable algorithm.

Formally, this operator requires that

$$\hat{\mathbf{u}}_{\ell=0}^n \cdot \nabla u_k^{n+1}, \quad \forall k \geq 0, \quad (2.77)$$

$$\hat{\mathbf{u}}_\ell^n \cdot \nabla u_{k=0}^{n+1}, \quad \forall \ell \geq 0. \quad (2.78)$$

The first assumption corresponds to the first column in (2.76) where $\ell = 0$ and the second assumption to the first row where $k = 0$. Next, we expand both assumptions to reveal the effect on the advection operators. The first assumption (2.77) gives an operator of the form

$$\begin{aligned} \hat{\mathbf{u}}_{\ell=0} \cdot \nabla u_k^{n+1} = & \sum_{k=0}^{N_z} \left[\left(\hat{u}_0^c \partial_x + \hat{v}_0^c \partial_y \right) \hat{u}_k^{c,n+1} - \left(\beta_k \hat{w}_0^c \right) \hat{u}_k^{s,n+1} \right] \cos(\beta_k z) \\ & + \left[\left(\hat{u}_0^s \partial_x + \hat{v}_0^s \partial_y \right) \hat{u}_k^{s,n+1} + \left(\beta_k \hat{w}_0^s \right) \hat{u}_k^{c,n+1} \right] \sin(\beta_k z). \end{aligned} \quad (2.79)$$

In contrast, the row-wise assumption (2.78) leads to

$$\begin{aligned} \hat{\mathbf{u}}_\ell \cdot \nabla u_{k=0}^{n+1} = & \sum_{\ell=0}^{N_z} \left[\left(\hat{u}_\ell^c \partial_x + \hat{v}_\ell^c \partial_y \right) \hat{u}_0^{c,n+1} \right] \cos(\beta_\ell z) \\ & + \left[\left(\hat{u}_\ell^s \partial_x + \hat{v}_\ell^s \partial_y \right) \hat{u}_0^{s,n+1} \right] \sin(\beta_\ell z). \end{aligned} \quad (2.80)$$

where the z -component vanishes, because $\beta_{k=0} = 0$.

Now we can substitute both assumptions as well as the explicit contributions of the advection operator into the strong form velocity equation (2.20). For brevity, we will only consider the cosine mode here.

$$\begin{aligned} \sum_{k=0}^{N_z} \left[\left[\nabla_{xy}^2 + (\lambda - \beta_k^2) + \left[\hat{u}_0^c, \hat{v}_0^c \right]^T \cdot \nabla_{xy} \right] \hat{\mathbf{u}}_k^{c,n+1} \right. \\ - \beta_k \hat{w}_0^c \hat{\mathbf{u}}_k^{s,n+1} \\ + \left[\hat{u}_k^c, \hat{v}_k^c \right]^T \cdot \nabla_{xy} \hat{\mathbf{u}}_0^{c,n+1} \\ = \frac{1}{\nu} \left[- \left[\hat{\mathbf{u}}_{\ell>0} \cdot \nabla \hat{\mathbf{u}}_{k>0} \right]^n \right. \\ \left. + \sum_q \frac{\alpha_q}{\Delta t} \hat{\mathbf{u}}_k^{c,n-q} - \nabla \hat{p}_k^{c,n+1} + \hat{\mathbf{f}}_k^{c,n+1} \right] \cos(\beta_k z) \end{aligned} \quad (2.81)$$

where $[\hat{\mathbf{u}}_{\ell>0} \cdot \nabla \hat{\mathbf{u}}_{k>0}^n]$ represents the explicit advection terms that are not covered by assumptions (2.77) and (2.78). The equivalent term for the sine mode is easily derived switching cosine and

sine components as well as substituting $-\beta_k \hat{w}_0 \hat{\mathbf{u}}_k^{s,n+1} = \beta_k \hat{w}_0 \hat{\mathbf{u}}_k^{c,n+1}$.

2.5.2 An algorithm for hybrid parallelisation

The implementation of the quasi-3D algorithm requires a few adaptations for higher performance. We follow the hybrid parallelisation strategy proposed by Bolis et al. [73] which combines the Fourier expansion in the z -direction with a spectral/hp element discretisation for the xy -plane. The approach enables a parallelisation across the Fourier modes as well as in the two-dimensional mesh. The Fourier coefficients are distributed across processors such that each processor holds, ideally, a single full mode. Additionally, the modal coefficients $\hat{\mathbf{u}}_k$ on each plane can be distributed across distinct processors via a mesh decomposition to further reduce the problem size. With that distributed setup, each process holds the data for one or more full modes $[\hat{p}_k^c, \hat{p}_k^s, \hat{\mathbf{u}}_k^c, \hat{\mathbf{u}}_k^s]$ for one partition of the mesh. With this hybrid strategy, the algorithm scales efficiently exploiting all three dimensions.

The pressure and velocity coefficients at the next time step are computed by solving the linear systems for the cosine and sine mode *sequentially*. Despite having both modes available on a processor, this minimises the rank of the system improving the condition number as well as reducing the complexity of the implementation. Interestingly, sequential solving has no influence on the pressure equation (2.75), it requires however changes to the velocity equation (2.81) in order to decouple the modes.

In the current form, the velocity update in (2.81) for the update of the cosine mode $\hat{\mathbf{u}}_k^{c,n+1}$ depends on the corresponding sine mode $\hat{\mathbf{u}}_k^{s,n+1}$ as well as the mean cosine mode $\hat{\mathbf{u}}_{k=0}^{c,n+1}$. The sine mode would require building a coupled matrix system that solves for the full mode simultaneously. We instead use half-mode matrices and, thus, take this term explicit to decouple cosine and sine modes. Similarly, the mean cosine mode would require solving the mean mode $k = 0$ first and, subsequently, solving the higher modes $k > 0$ in parallel. This would effectively double the computational time for each solve. To avoid this cost overhead, we evaluate this advection

contribution explicitly as well. The decoupled half-mode velocity equation becomes

$$\begin{aligned}
& \sum_{k=0}^{N_z} \left[\left[\nabla_{xy}^2 + (\lambda - \beta_k^2) + \frac{1}{\nu} \left[\hat{u}_0^c, \hat{v}_0^c \right]^T \cdot \nabla_{xy} \right] \hat{\mathbf{u}}_k^{c,n+1} \right. \\
&= \frac{1}{\nu} \left[- \left[\hat{\mathbf{u}}_{\ell>0} \cdot \nabla \hat{\mathbf{u}}_{k>0} \right]^n - \beta_k \hat{w}_0 \hat{\mathbf{u}}_k^{s,n} + \left[\hat{u}_k^c, \hat{v}_k^c \right]^T \cdot \nabla_{xy} \hat{\mathbf{u}}_0^{c,n} \right. \\
&\left. \left. + \sum_q \frac{\alpha_q}{\Delta t} \hat{\mathbf{u}}_k^{c,n-q} - \nabla \hat{p}_k^{c,n+1} + \hat{\mathbf{f}}_k^{c,n+1} \right] \right] \cos(\beta_k z). \tag{2.82}
\end{aligned}$$

In this form, the velocity update requires N_z solves of a two-dimensional linear ADR problem for the coefficients $\hat{\mathbf{u}}_k^{n+1}$ of each half mode.

Next, we transform the modal problems (2.75) and (2.82) into the weak form for solution with Galerkin-type approximations. For brevity, we omit the summation and Fourier basis ϕ_k . Taking the L_2 -inner product of the k -th pressure problem (2.75) with the gradient of the basis $\nabla \phi(x, y)$ and integrating by parts, the weak form pressure equation is

$$\begin{aligned}
\int_{\Omega} \nabla \phi \cdot \nabla \hat{p}_k^{c,n+1} - \beta_k^2 \int_{\Omega} \phi \hat{p}_k^{c,n+1} &= \int_{\Omega} \nabla \phi \cdot \left(\sum_q \frac{\alpha_q}{\Delta t} \hat{\mathbf{u}}_k^{c,n-q} - [\mathbf{u} \cdot \nabla \mathbf{u}]_k^{c,n} + \hat{\mathbf{f}}_k^{n+1} \right) \\
&\quad - \int_{\Gamma} \mathbf{n} \cdot \left[\phi \left(\frac{\gamma}{\Delta t} \hat{\mathbf{u}}_k^{c,n+1} + \nu \nabla \times \hat{\omega}_k^{c,n} \right) \right], \tag{2.83} \\
\forall \phi &\in H^1(\Omega).
\end{aligned}$$

Similarly, the k -th velocity equation results from the L_2 inner product with the basis $\psi(x, y)$. Here we integrate the diffusion term by parts leading to

$$\begin{aligned}
& \int_{\Omega} \frac{\gamma}{\nu \Delta t} \psi \hat{\mathbf{u}}_k^{c,n+1} + \int_{\Omega} \nabla \psi \cdot \nabla \hat{\mathbf{u}}_k^{c,n+1} + \frac{1}{\nu} \int_{\Omega} \psi \left[\hat{u}_0^c, \hat{v}_0^c \right]^T \cdot \nabla_{xy} \hat{\mathbf{u}}_k^{c,n+1} \\
&= \frac{1}{\nu} \int_{\Omega} \psi \left[- \left[\hat{\mathbf{u}}_{\ell>0} \cdot \nabla \hat{\mathbf{u}}_{k>0} \right]^n - \beta_k \hat{w}_0 \hat{\mathbf{u}}_k^{s,n} \right. \\
&\quad \left. + \left[\hat{u}_k^c, \hat{v}_k^c \right]^T \cdot \nabla_{xy} \hat{\mathbf{u}}_0^{c,n} + \sum_q \frac{\alpha_q}{\Delta t} \hat{\mathbf{u}}_k^{c,n-q} \right. \\
&\quad \left. - \nabla \hat{p}_k^{c,n+1} + \hat{\mathbf{f}}_k^{c,n+1} \right] \\
&\quad + \int_{\Gamma} \mathbf{n} \cdot \psi \nabla \hat{\mathbf{u}}_k^{c,n+1}, \tag{2.84} \\
\forall \psi &\in H_0^1(\Omega)^d.
\end{aligned}$$

The matrix systems are discretised with a spectral/hp element method for efficient evaluation

of the tensor-product basis. Each matrix problem for the pressure (2.83) and velocity (2.84) is statically condensed via a boundary-interior decomposition to improve the condition number, see section 2.4.1. The Helmholtz matrix for pressure is symmetric and positive definite which makes it amenable to solve via a parallel direct solver. We employ the XXT solver by Tufo and Fischer [62] which is based on a parallel Cholesky decomposition. The ADR matrices are non-symmetric and indefinite which requires a solver for general matrices. Since the matrix coefficients are also time-dependent, we use an iterative approach with the restarted Generalised Minimum Residual (GMRES) method and precondition the system with the inverse of the diagonal of the matrix.

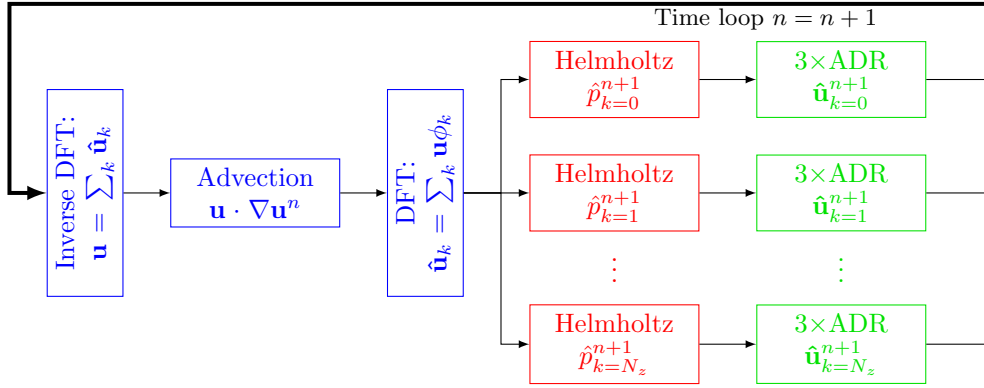


Figure 2.7: The mean-implicit algorithm for a quasi-3D velocity-correction scheme. The flowchart shows only few distinct modes to clearly highlight the building blocks of the algorithm. Note that each wavenumber k has two associated half modes (sine and cosine). However, practical simulations typically use $N_z \gg 10$, see figure 2.6.

Figure 2.7 presents the algorithm to advance the mean-mode-implicit velocity correction scheme to the next time step. After setting initial and boundary equations, it evaluates the advection term in physical space and transforms between Fourier and physical space via a DFT. With the explicit advection term available on each processor, the new cosine and sine coefficients for pressure and velocity are computed with full modes distributed to distinct processors. Each mode k involves solving first a Helmholtz problem for the pressure and subsequently three ADR problems for each velocity component of the cosine mode. This process is repeated to solve for the sine mode. Finally, an inverse DFT is employed to communicate the updated solution and transform it back to physical space where the cycle restarts with the next explicit advection term.

2.6 The Nektar++ software framework

This section gives an overview of the Nektar++ framework. A detailed description is available in Cantwell et al. [43] and Moxey et al. [9].

Nektar++ is a framework for solving partial differential equations (PDE) using the spectral/hp element method introduced in section 2.3. It has been successfully used for large-scale simulations for the compressible and incompressible Navier-Stokes equations using 2D, 3D and quasi-3D discretisations, see references [24, 30, 32, 34, 70, 71]. The integrated preprocessing tool NekMesh [44] further enables handling computational meshes for challenging geometries and provides accurate representations of surfaces through CAD connectivity see reference [45] for details. The framework is written in C++ and relies on a software abstraction layer using factory patterns as well as C++11-native smart pointers. The user and developer guides [74] contain an in-depth introduction to the design of Nektar++.

The work by Cantwell et al. [43] gives an overview of Nektar++’s capability for large scale simulations. They present almost ideal scaling of the framework up to thousands of cores in parallel. More recent work by Lindblad et al. [75] shows strong scaling on the Archer2 high performance computer (HPC) up to one hundred thousand cores. The framework’s suitability for handling the typically large data sets of high-fidelity simulations are introduced by Moxey et al. [9]. Nektar++ uses in-situ processing and parallel input/output (IO). In-situ processing allows running post-processing steps during the simulation such that the parallel computation produces derived quantities, such as vorticity and Reynolds stresses, during the simulation and makes them readily available for analysis. Reading and writing large data sets in parallel is enabled through the hierarchical data format version 5 (HDF5) and the data is written into block-unstructured VTK [76] files such that visualisations are possible with free tools such as ParaView [77]. In summary, this motivates the usage of the framework for high-fidelity simulations of simple and complex geometries and should provide an appropriate framework to investigate the linear-implicit scheme.

The framework is split into two structures: a **library** for developers of numerical methods and **solvers**, for application-specific users of the software. Figure 2.8 shows a top-down structure of the framework with the **solvers** at the top and an interface to the core libraries via **SolverUtils**. **SolverUtils** includes features such as a generic time-stepping algorithm using the general linear method (GLM) which allows implicit, explicit and mixed implicit-explicit schemes. The core Nektar++ libraries include a generic interface to spatial discretisations using continuous

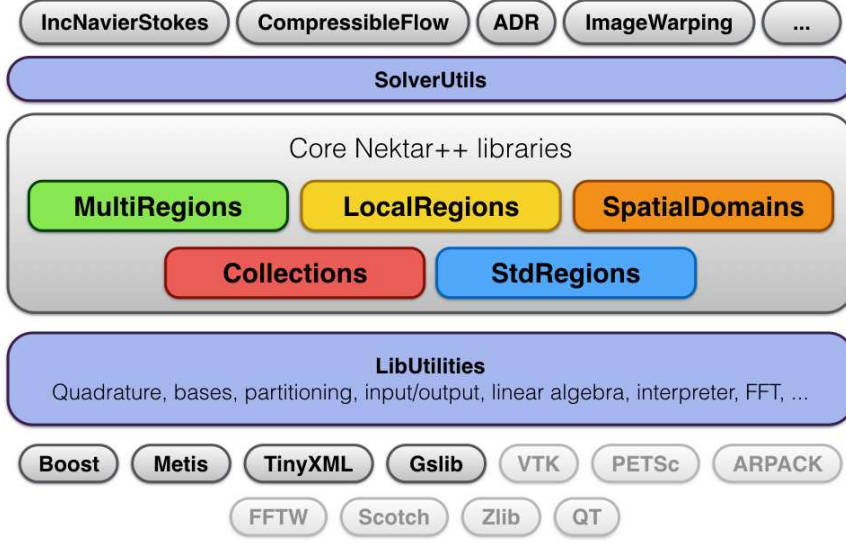


Figure 2.8: Top-down structure of the Nektar++ framework. it shows the connection between applications-specific solvers with the core libraries, fundamentals in **LibUtilities** and third-party dependencies like **Boost** and **Gslib**.

Galerkin (CG) and discontinuous Galerkin (DG) approaches with equal or mixed polynomial order. The sublibrary **LibUtilities** holds fundamental functionality such as definitions for basis functions ϕ , linear algebra operations, and others as introduced in sections 2.3 and 2.4.2. Further the library contains functionality for pre- and post-processing. Pre-processing includes generating higher-order meshes from a given linear mesh and optimising the mesh with curvature to represent complex geometries with **NekMesh**. The utility **FieldConvert** enables post-processing for conversion between dataformats or computing derived quantities from a given solution, for example, the wall shear stress or vorticity.

The **solvers** at the top of figure 2.8 use library functions to realise solvers for a given PDE. The framework is most-developed for incompressible and compressible Navier-Stokes equations, advection-diffusion-reaction problems and electro-cardiological modelling. Through the generic time-stepping algorithms it is possible to solve steady and unsteady problems.

This work uses the incompressible Navier-Stokes solver, namely **IncNavierStokesSolver**, which is implemented with the semi-implicit velocity correction scheme that we introduced in section 2.2. We extend the implementation with the linear-implicit velocity correction scheme. This requires generating a new class for the **IncNavierStokesSolver**.

One challenge of this change was the time-dependent ADR matrix for the linear-implicit scheme. It requires handling references to the linear system of the ADR matrices and deleting them to avoid memory leakage. Therefore, we implemented a logic inside the incompressible solver directory and utility functions in the `MultiRegions` library. The functions specifically delete only the time-varying elemental matrices and keep time-independent matrices in memory to maximise performance.

We conduct all our tests with the Nektar++ framework version 5.5.0.

Chapter 3

Stability and accuracy of the linear-implicit scheme

This chapter revisits theoretical and investigates practical limitations for the stability and accuracy of the linear-implicit velocity-correction scheme. We re-visit theoretical limitations based on a literature review and check both research questions. This involves finding the largest stable time step size with the semi-implicit scheme and the possible increase using the linear-implicit scheme. This allows identifying the stability margin for the initial transient and evaluating the influence of larger time step sizes on the resolution of the initial transient and the quasi steady-state, see chapter 1.

We consider a range of test cases for studying the behaviour of the linear-implicit scheme in incompressible flows. The tests verify the algorithm, help identify numerical characteristics, and give insight into the scheme's accuracy. On the one hand, canonical examples include two dimensional cases such as steady Kovasznay flow, a time-varying manufactured solution and cylinder flow at a low Reynolds number. On the other hand, challenging problems with complex geometries, that is a Formula 1 car's front wing, and high Reynolds number flow. The challenging problems have large meshes ranging in size from 1 million degrees of freedom (DoF) up to 100 million DoF.

The overview in table 3.1 lists all cases and highlights key features. The Reynolds number indicates whether the flow is laminar as for the Kovasznay and cylinder flow or turbulent, see the Imperial Front Wing (IFW) cases. The DoF and polynomial order to provide a reference for the scale of each problem. Further, we provide an overview of the stability with the maximum

Table 3.1: An overview of all the test cases investigated for the stability and accuracy of the linear-implicit velocity-correction scheme. The table shows, from left to right, the domain’s dimension, Reynolds number, degrees of freedom of the assembled matrix and the polynomial order of the velocity space P_u . Additionally, we highlight the stability in terms of the maximum stable time step size $\Delta t_{\text{CFL}}^{\text{Explicit}}$ with the semi-implicit scheme and the possible increase in time step size using the linear-implicit scheme denoted by $\Delta t_{\text{CFL}}^{\text{Implicit}}$. An indication of accuracy is measured by the difference in time-averaged lift coefficient $\bar{C}_l$ between the semi-implicit prediction and linear-implicit prediction at a time step size $\Delta t = 10\Delta t_{\text{CFL}}$.

Case name	Re. number	DoF	P_u	$\Delta t_{\text{CFL}}^{\text{Explicit}}$	$\times \Delta t_{\text{CFL}}^{\text{Implicit}}$	$\Delta \bar{C}_d(10\Delta t_{\text{CFL}})$
2D Kovasznay	40	0.4×10^3	9	$1e-2$	$\times 10^6$	-
2D Cylinder	500	104.2×10^3	6	$1e-3$	$\times 80$	$\leq 0.3\%$
2D IFW	2.2×10^5	0.9×10^6	6	$5e-6$	$\times 100$	$\leq 1\%$
3D eIFW	1.7×10^5	54.7×10^6	4	$1e-5$	$\times 100$	$\leq 1\%$

stable time step size for the semi-implicit scheme $\Delta t_{\text{CFL}}^{\text{Explicit}}$ and the respective increase using the linear-implicit scheme denoted as $\times \Delta t_{\text{CFL}}^{\text{Implicit}}$. Finally, we indicate the change in accuracy with the difference in time-averaged lift coefficient at an order of magnitude increase in time step size.

3.1 Stability analysis

We discuss theoretical limits for the stability of the linear-implicit time-stepping scheme. Stability generally means that any numerical oscillations in the discrete equations are damped and do not lead to unbounded growth of the energy of the system. The opposite case of an *unstable* scheme typically shows a solution that is *diverging* from a physical solution. Stability analysis for the discretisation of the time derivative in section 2.2, also known as linear analysis, is available in literature [78]. The employed backwards-difference formula (BDF) is straightforward to analyse. A non-linear analysis specifically for the incompressible Navier-Stokes equations was first presented in reference [38] by Simo and Armero. They check whether the energy of the system is decreasing, or stable, for the time-discrete formulation. This section briefly reviews both analysis in the context of velocity correction schemes.

3.1.1 Linear analysis of time-stepping schemes

We aim for an implicit time-stepping scheme in order to achieve unconditional stability that is being able to use arbitrary large time steps Δt to reach the *solution* faster. The BDF schemes are well-established for that, because they are linear multi-step schemes that are straightforward to implement. Furthermore, BDF up to second order are A-stable which implies that their region of

stability covers the entire left half of the complex plane. Figure 3.1 shows the region of *instability* for the BDF formulas up to 3rd order. We see that a BDF formula of order 3 does not cover the entire left half plane. This implies that the scheme is not unconditionally stable. For that reason, we are using the 2nd order BDF (BDF2) for the time-stepping. Note that the stability regions are based on the linear test equation $\partial u / \partial t = \lambda u$.

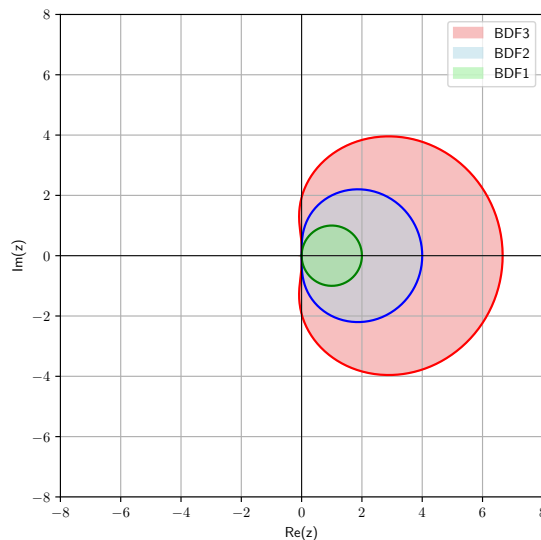


Figure 3.1: The region of stability lies outside the highlighted regions. Comparison of BDF time-stepping schemes for order 1, 2 and 3.

3.1.2 Non-linear stability analysis for velocity correction schemes

Simo and Armero [38] build an analysis for time-stepping schemes applied directly to the incompressible Navier-Stokes system. They analyse the energy stability of the time-discrete algorithm. This checks whether the discrete kinetic energy E is consistent with the energy of the continuous equations, that is decaying due to viscosity over time. The analysis applies equally to monolithic and splitting, or fractional step, schemes.

Dong and Shen proof energy stability for the linear-implicit velocity correction scheme [39]. They, however, simplified the scheme to use the standard form for the diffusion term as well as a first-order BDF approximation. The analysis of velocity correction schemes is complemented by analysis on the second-order BDF for the Stokes equations, which exclude the non-linear term [22]. The analysis in reference [79] by Chen and Shen includes the non-linear term. They proof stability for the 1st order scheme with non-linear term and the 2nd order scheme without non-

linear term. Whether the proof equally applies to the 2nd order, linear-implicit and rotational velocity correction scheme described in section 2.2 is suggested by the works above, but not proven.

Further, the above analysis did not consider the time- and space-discrete (fully-discrete) system. Guermond et al. [80] consider the fully-discrete system for the 2nd order rotational velocity-correction scheme, however, without the non-linear term. They point out that the optimal error estimates hold for the first and second order scheme assuming the inf-sup condition is satisfied.

In summary, the analysis on velocity-correction schemes stretches over almost 30 years. The proof of stability for the fully-discrete velocity correction scheme, for second order in time and including the non-linear term is not established due to the intricacies of the non-linear and rotational diffusion term, and the BDF2 approximation. Nevertheless, the accumulated analysis indicates that the scheme is likely stable and 2nd order accurate. We provide numerical evidence for the stability on canonical problems in section 3.4 and complex problems in section 3.5 below.

3.2 Accuracy of the scheme

This section briefly summarises the formal conclusions on velocity correction schemes from the literature. There is a range of analysis on the time-discrete and space-continuous and, more recently, fully-discrete velocity correction scheme. In addition, we verify the accuracy of our implementation with the method of manufactured solutions.

3.2.1 Review of error estimates

The temporal error of the scheme is influenced by the time discretisation using the BDF, extrapolation, the splitting scheme and the spatial discretisation. The BDF scheme's truncation error gives the formal order of convergence, however, the splitting scheme influences the temporal error [21]. While the truncation error is easily derived for a BDF scheme of varying order, the splitting scheme adds complexity to the analysis. The error analysis for the velocity-correction scheme with 2nd order BDF is shown in reference [22]. They find that the order of convergence in the time-continuous case for velocity $\mathbf{u}$ is $\mathcal{O}(\Delta t^2)$ and for pressure p is $\mathcal{O}(\Delta t^{3/2})$ in the L_2 norm.

The analysis by Chen and Shen [79] further shows that 1st and 2nd order convergence for velocity and pressure is possible in the fully-discrete case. They derive error estimates for the 1st

order velocity correction scheme with non-linear term and 2nd order estimates without non-linear term. Finally, they use Legendre polynomials to approximate a simple domain with manufactured solution and provide evidence for second-order convergence. They show that both, velocity and pressure, have second order convergence in the L_∞ norm, however, only velocity shows 2nd order convergence in the L_2 norm. Note that the L_∞ norm provides an upper limit for the accuracy.

All in all, the theoretical analysis does not cover the exact scheme that we are employing: a rotational velocity correction scheme with non-linear terms and 2nd order extrapolation. Next, we verify the stability and accuracy of our implementation using the spectral/hp element discretisation.

3.3 Verification of the implementation

This section investigates practical stability limits and the achievable accuracy for canonical test problems. We consider two-dimensional problems on small-scale quadrilateral meshes to confirm stability and accuracy in this order. The stability check uses steady Kovasznay flow based on the work by Kovasznay [48]. Further, we verify the temporal accuracy of our implementation with a manufactured solution.

3.3.1 2D steady Kovasznay flow

Canonical problems with analytical solutions are ideal for validating the implementation. The Kovasznay flow represents a laminar recirculation zone behind an array of cylinders, see reference [48]. Figure 3.2 shows the streamlines of the solution. For this study, we are investigating a domain $\Omega \in [-0.5, 1.0] \times [-0.5, 0.5]$ filled with quadrilateral elements. The aim is to reproduce the stability results in [39] to verify our implementation and identify steady-state stability characteristics.

The analytic solution in [48], however written in conservative variables, is

$$u = 1 - e^{-\lambda x} \cos(2\pi y), \quad (3.1)$$

$$v = \frac{-\lambda}{2\pi} e^{-\lambda x} \sin(2\pi y), \quad (3.2)$$

$$p = \frac{1}{2} (1 - e^{-2\lambda x}), \quad (3.3)$$

$$\lambda = \frac{1}{2\nu} - \sqrt{\frac{1}{4\nu^2} + 4\pi^2}, \quad (3.4)$$

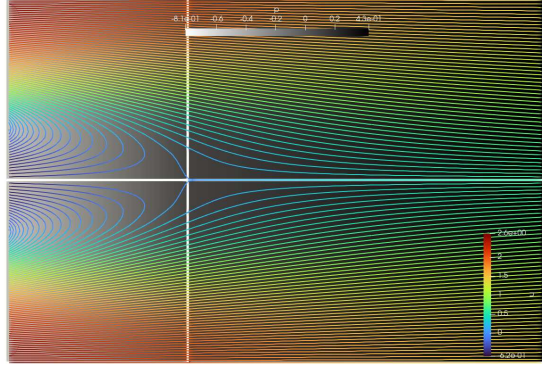


Figure 3.2: A visualisation of the streamlines in the steady-state Kovasznay flow highlighting the structure of the recirculating flow. Boundaries of the four quadrilateral elements are highlighted in white, streamlines coloured by x-velocity magnitude u and the pressure with a greyscale.

where the kinematic viscosity $\nu = \frac{1}{40}$ for a valid solution. We choose a high polynomial order of $P = 9$ for both velocity and pressure spaces to have negligible spatial errors. The time integration uses second order $J = 2$ for all results presented below. The boundary conditions are Dirichlet type enforcing the exact solution for velocity and higher-order pressure boundary conditions, see the boundary integral in equation (2.26).

Dong and Shen [39] show H_1 errors up to $\Delta t = 0.45$ in the original paper where NaN implies that no data is available. Overall the linear-implicit scheme extends the stability up to $45\times$ larger Δt as compared to the semi-implicit scheme. Further, we find good agreement between our implementation of the linear-implicit scheme and the reference data in the literature [39]. The H_1 errors are of the same order of magnitude for almost every time step size and show little variation suggesting good agreement.

Knowing that we can reproduce the original results, we extended our study to time steps $\Delta t > 0.45$ and investigate when, or if, the scheme becomes unstable. Comparing the linear-implicit scheme for the extrapolated and updated advection velocity approach, we found that they behave similar up to a time step size of $\Delta t = 0.45$. For larger time step sizes, both approaches remain stable, however, the updated approach gives a result with large errors. This is possibly linked to a state where the linear-implicit scheme does not cause an unbounded growth in energy, but finds a non-physical solution that is different from the Kovasznay flow stated in equation (3.1). This is an unexpected result, because the Kovasznay flow is a steady-state problem and the time-stepping error should not change the solution. The extrapolated approach does not show this behaviour despite choosing a time step several orders of magnitude larger than the upper

Δt	Semi-implicit	Dong and Shen [39]	Upd. Linear-implicit	Ext. Linear-implicit
0.008	$3.522e-8$	$2.600e-8$	$3.048e-8$	$3.518e-8$
0.01	$3.650e-8$	$2.768e-8$	$3.140e-8$	$3.645e-8$
0.02	∞	$3.616e-8$	$3.851e-8$	$4.390e-8$
0.03	∞	$4.403e-8$	$4.597e-8$	$5.074e-8$
0.04	∞	$5.126e-8$	$5.297e-8$	$5.668e-8$
0.05	∞	$5.799e-8$	$5.972e-8$	$6.222e-8$
0.06	∞	$6.431e-8$	$6.613e-8$	$6.723e-8$
0.07	∞	$7.033e-8$	$7.233e-8$	$7.192e-8$
0.08	∞	$7.609e-8$	$7.836e-8$	$7.634e-8$
0.1	∞	$8.705e-8$	$9.004e-8$	$8.460e-8$
0.12	∞	$9.745e-8$	$1.014e-7$	$9.227e-8$
0.14	∞	$1.075e-7$	$1.125e-7$	$9.951e-8$
0.16	∞	$1.172e-7$	$1.234e-7$	$1.064e-7$
0.18	∞	$1.268e-7$	$1.343e-7$	$1.130e-7$
0.2	∞	$1.363e-7$	$1.450e-7$	$1.194e-7$
0.25	∞	$1.603e-7$	$1.719e-7$	$1.344e-7$
0.3	∞	$1.856e-7$	$1.988e-7$	$1.485e-7$
0.35	∞	$2.150e-7$	$2.264e-7$	$1.616e-7$
0.4	∞	$2.593e-7$	$2.555e-7$	$1.741e-7$
0.45	∞	$4.267e-7$	$2.929e-7$	$1.859e-7$
0.5	∞	NaN	$1.689e+3$	$1.971e-7$
0.8	∞	NaN	$3.217e+2$	$2.549e-7$
1	∞	NaN	$1.971e+2$	$2.867e-7$
10	∞	NaN	$6.383e+1$	$6.878e-7$
100	∞	NaN	$1.102e+2$	$8.993e-7$
1,000	∞	NaN	$2.151e+2$	$9.326e-7$
10,000	∞	NaN	$5.993e+1$	$9.361e-7$

Table 3.2: Stability comparison of the Nektar++ implementation of the linear-implicit against the semi-implicit scheme and results from reference [39]. Additionally, we add the effect of the choice of advection velocity $\tilde{\mathbf{u}}$, extrapolated (Ext.) or updated (Upd.), approach. Data presents the H_1 error of the x-velocity component u for increasing time step size Δt .

limit of the updated approach. This difference for both approaches suggests that the updated approach introduces an error proportional to the time step size Δt by using the residual term as the advection velocity, cf. equation (2.25).

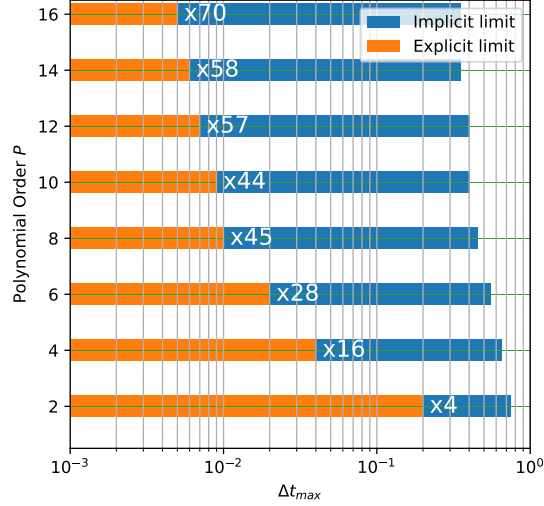


Figure 3.3: A comparison of the stability measured in time step size Δt that achieves convergence to the steady-state Kovasznay flow.

Considering the *updated* linear-implicit scheme, we further investigated the stability limit with polynomial refinement. Figure 3.3 shows the maximum stable time step size for the semi-implicit scheme (orange) and the updated linear-implicit scheme (blue). The semi-implicit scheme shows a strong reduction with increasing polynomial order P limiting the time step size from 0.2 at low polynomial order to 0.005 at a high polynomial order $P = 16$. Within the same polynomial order range, the linear-implicit scheme also shows a decrease from $\Delta t = 0.75$ at $P = 2$ to $\Delta t = 0.35$ at $P = 16$. Thus, the semi-implicit scheme sees a constraint of a factor of 40 between the low and high polynomial order while the linear-implicit scheme a factor of about 2. These results hint at a potentially low dependency of stability and spatial refinement for the linear-implicit scheme. Additionally, figure 3.3 shows the stability margin, that is the ratio of maximum Δt for both schemes, which highlights the strong increase in stability at larger polynomial order. Note that these results are mesh- and problem-dependent and we will investigate below how these characteristics extend to two- and three-dimensional unsteady flow problems.

3.3.2 Manufactured solution: verifying temporal accuracy

The method of manufactured solution allows testing an algorithm with a generic analytical solution. Since time-varying analytic solutions to the incompressible Navier-Stokes equations are rare, we employ a manufactured solution to test the temporal accuracy of our implementation.

We apply the method of manufactured solutions to study the convergence order in time. The employed analytic solution is non-physical. It is created by substituting functions for each dependent variable u, v, p into the Navier-Stokes equations and evaluating each derivative to find an analytic forcing function $\mathbf{f} = \{f_x, f_y\}$ that balances the equations. The forcing terms along with Dirichlet boundary conditions and initial conditions based on the analytical functions for each variable are applied to the domain $\Omega \in [0, 2] \times [-1, 1]$. This domain is discretised with four quadrilateral elements and using a polynomial order of $P = 15$ which makes temporal errors dominant. Finally, we predict the solution for an entire period of the time-varying functions that is up to the final time $t_f = 2\pi$.

We employ the python library `sympy` [81] to construct a two- or three-dimensional unsteady analytic solution which satisfies the momentum equations and is divergence-free. The tool also helps in testing variations which may represent physically relevant solutions. A basic script to construct manufactured solutions is available in appendix A.

Finally, we choose the following two-dimensional problem formulation

$$u = \frac{A\pi}{a} \cos(ax) \cos(\pi y) \sin(bt), \quad (3.5)$$

$$v = A \sin(ax) \sin(\pi y) \sin(bt), \quad (3.6)$$

$$p = A \sin(ax) \sin(\pi y) \cos(bt), \quad (3.7)$$

where $A = 2$, $a = \pi$, $b = 1$ and the kinematic viscosity $\nu = 0.1$. Assuming the solution is divergence-free, we substitute it into the momentum equations in (2.1) to find forcing terms $\mathbf{f}$ that balance the Navier-Stokes equations.

We observe the order of convergence by varying the time step size Δt and comparing the maximum of the L_2 and L_∞ error norms over the entire interval $T \in [0, t_f]$. The order of convergence is determined with a polynomial fit to the the maximum norms for each time step size. Figure 3.4 shows the convergence of the x-velocity component u (solid line) and pressure p (dotted line). Considering the semi-implicit scheme, we observe second-order convergence for the u velocity component and a slightly lower-order of 1.8 for the pressure. This observation is

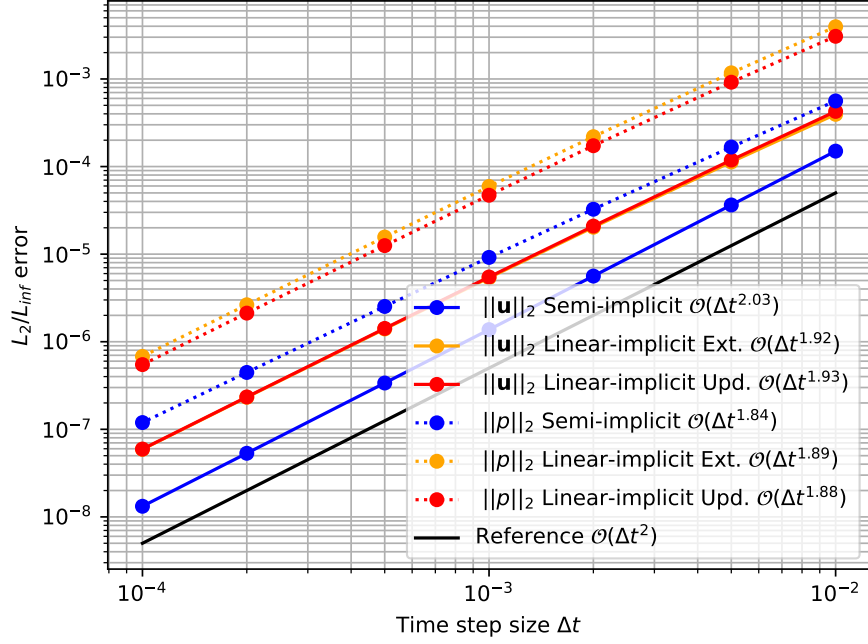


Figure 3.4: The time convergence order on a manufactured solution with varying time step size Δt . A comparison of the semi-implicit and linear-implicit scheme where the implicit scheme uses two different advection velocity approaches.

inline with the error analysis in reference [22]. Further, the linear-implicit scheme shows almost identical behaviour independent of the choice of advection velocity $\tilde{\mathbf{u}}$ which has been suggested in reference [38].

A comparison of the semi-implicit against the linear-implicit schemes reveals that the linearised advection operator changes the error constant. We observe a shift in both L_2 and L_∞ errors by a factor of approximately 5. This shift increases the error independent of the time step size Δt . Since the polynomial order is high, this is likely triggered by a change in the temporal error constant due to the implicit treatment of the advection operator. Nevertheless, we observe second-order accuracy in time for the linear-implicit scheme independent of the choice of advection velocity.

In summary, the canonical stability and accuracy tests highlight numerical characteristics of the linear-implicit scheme. First of all, we reproduce the stability limits from the work of Dong and Shen [39] when using the updated advection velocity for the Kovasznay flow. Moreover, the extrapolated advection velocity shows truly unconditional stability for this canonical problem with time step sizes tested up to $\Delta t = 10^4$. Considering the accuracy, the advection velocity does

not show a significant impact on the L_2 and L_∞ norms of the velocity component u . Therefore, we use the linear-implicit scheme with extrapolated advection velocity for all following tests.

3.4 Stability and accuracy on a canonical problem

We consider a two dimensional cylinder flow to check our research questions of reaching the quasi-steady state with larger time step sizes and sampling statistics with larger time step sizes. Similar to the verification in the previous section 3.3, we compare the linear-implicit and semi-implicit velocity correction schemes to find insight into the numerical characteristics and best-practise for complex simulations.

3.4.1 2D unsteady cylinder flow

The flow around a cylinder in two dimensions and at a Reynolds number $Re = 500$ is characterised by a laminar boundary layer flow that periodically sheds into a Von Kármán vortex street. The two-dimensional setup constraints the flow to not show the irregular shedding observed in 3D. This classic benchmark problem for incompressible flow allows resolving the flow accurately in space and, hence, isolating the temporal error. It further has the advantage that reference [39] uses the same test case for testing stability and accuracy of the linear-implicit scheme.

Therefore, we begin with establishing a mesh that is converged in spatial resolution using the semi-implicit velocity correction scheme, see section 2.2, as a baseline. This serves as reference for further tests of stability and accuracy. We measure stability as the maximum stable time step size Δt where we define stable as giving physical predictions for at least $T = 1000t^*$ without diverging. The non-dimensional time unit $t^* = \frac{U}{D}t$ uses the freestream velocity $U = 1$ and cylinder diameter $D = 1$. It defines the time for a structure to convect along the characteristics length of the geometry.

The reference for measuring stability improvements is the maximum stable time step size using the semi-implicit scheme Δt_{CFL} where CFL indicates that this time step corresponds to an elemental CFL number $\text{CFL}^e \approx 1$. Note that we use the CFL estimator introduced in section 2.3.4. Each simulation is restarted from either a uniform flow field for the mesh convergence study below, or an initialised cylinder flow at $t = 200t^*$ for the stability study. The initialised flow field at $t = 200t^*$ has transitioned into periodic vortex shedding.

We measure the accuracy using the integral drag and lift coefficients around the surface of

the cylinder. They are defined as

$$C_d = \frac{F_x}{\frac{1}{2}\rho U^2 A}, \quad C_l = \frac{F_y}{\frac{1}{2}\rho U^2 A}, \quad (3.8)$$

respectively. In particular, we consider the time-averaged drag coefficient $\overline{C_d}$ using the time-averaged force $\overline{F_x}$ in the x-direction. For the lift coefficient, we consider the root-mean-square (RMS) of the vertical force $F_{y,\text{RMS}}$. Both coefficients are normalised by the area $A = DL$ using the cylinder diameter $D = 1$ and length $L = 1$. Note that the density $\rho = 1$ in the non-dimensional formulation of the governing equations (2.1).

Description of the cylinder case

The cylinder uses a domain $\Omega = [-10, 40] \times [-10, 10]$ with unit diameter. The Reynolds number $Re = 500$ with freestream velocity U and unit density $\rho = 1$ defines the kinematic viscosity for this case as $\nu = \frac{UD}{Re} = 0.002$. Figure 3.5 highlights the geometry and illustrates the mesh structure. We employ the meshing software Gmsh [82] to construct a structured mesh with 2880 quadrilateral elements. The boundary conditions for this case are uniform inflow, robust outflow using equations (2.53) and (2.54), a periodic upper and lower boundary, and no-slip conditions on the cylinder surface.

The quadrature for the basis functions uses exact quadrature following the dealiasing strategy in section 2.3.3. The number of quadrature points is $Q = P + \frac{P_{\text{adv}}}{2} + \frac{3}{2}$ where $P_{\text{adv}} = P$ is the polynomial order of the nonlinearity. While the reference [39] does not state whether the mesh is curved or not, we use a non curved mesh and, hence, no additional increase in quadrature is necessary. Both the linear-implicit and semi-implicit scheme use the same quadrature.

Reproducing reference data from Dong and Shen [39]

We approximate the cylinder flow with a mesh of quadrilaterals as introduced above in figure 3.5. The structure of the mesh follows the finest mesh from reference [39]. Next, we vary the polynomial order P within the range $P \in [2, 6]$ for pressure and velocity approximation spaces to reproduce the reference result. Note that we use equal polynomial order for velocity and pressure spaces $P_u = P_p = P$. We use the semi-implicit scheme with the time step size $\Delta t = 0.001$ which is stable for all polynomial orders. Note that this is different from reference [39] where the smallest reported time step is doubled with $\Delta t = 0.002$.

Table 3.3 shows the time-averaged drag and RMS lift coefficient, respectively. The averaging

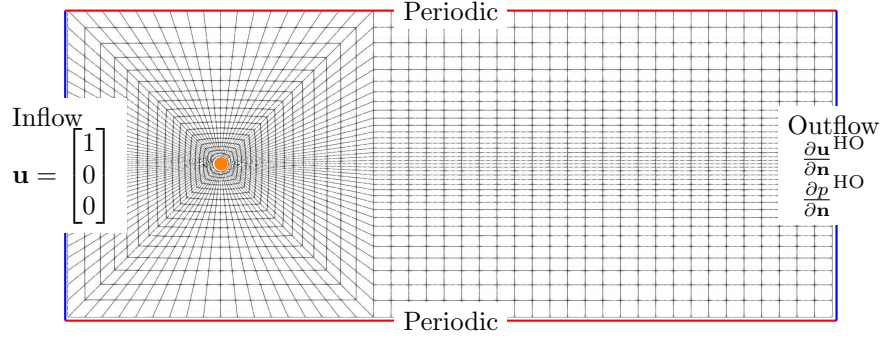


Figure 3.5: Setup of the cylinder test case highlighting the mesh structure with quadrilateral elements and the cylinder location in green colour. The boundary conditions on all boundaries are highlighted and the cylinder wall itself uses a no-slip condition.

Table 3.3: Mesh convergence for the time-averaged lift and drag coefficients on the 2D cylinder flow. We use data from the interval $T \in [200, 1000]$.

Polynomial order P	$\overline{C}_{l,\text{RMS}}$	$\overline{C}_d$
2	0.47923	1.08656
3	0.71154	1.27701
4	0.84862	1.45267
5	0.86037	1.47841
6	0.85149	1.46851
Dong and Shen[39]	0.851	1.466

interval uses $T \in [200, 1000]$ which is well into the periodic vortex shedding for all resolutions. Clearly, the error drops rapidly with polynomial order. At the coarsest resolution with $P = 2$, the error in the lift is 43.7% relative to the reference. However, the error drops with increasing polynomial order to 16.4%, 0.3%, -1.1%, -0.06% for polynomial orders $P = 3, 4, 5, 6$, respectively. Interestingly there is an overshoot behaviour in the error for $P = 5$ indicated by an increase in the error and a negative sign. Still, we observe a convergence towards the reference lift and drag values with polynomial order. At the highest resolution, each coefficient varies by less than 0.1% from the reference suggesting that this result is closely matching the study of Dong and Shen [39].

The following stability and accuracy investigations use the mesh with a polynomial order $P = 6$ and are restarted from the solution $(\mathbf{u}, p)$ at $t = 200t^*$.

Stability limitations

The mesh convergence study shows the maximum stable time step size for the semi-implicit scheme is $\Delta t_{\text{CFL}} = 0.001$. This time step size also corresponds to a time-averaged $\overline{\text{CFL}}^e \approx 1$. The time-averaged CFL uses the CFL estimator from section 2.3.4 which provides the maximum estimated CFL^e considering all elements Ω^e , see equation (2.57). We next identify stability limits with the linear-implicit scheme by increasing the time step size Δt starting with Δt_{CFL} to establish the largest stable time step size. Additionally, we observe the influence on CFL^e estimates.

The stability overview in table 3.1 shows that the linear-implicit scheme with extrapolated advection velocity allows time steps up to $\Delta t_{\text{max}} = 80\Delta t_{\text{CFL}}$. The maximum local CFL^e appears consistently downstream of the cylinder. The exact location oscillates with the period of the vortex shedding between the upper and lower edge of the cylinder. This results appears reasonable since the largest velocity magnitude $\|u\|_2$ appears within the shedding vortices and it coincides with fine to intermediate mesh refinement.

Figure 3.6 shows the time evolution of the CFL^e estimates for different time step sizes. The plot shows the oscillations around the time-averaged estimate which is highlighted with a dashed line and written on top of the line. The semi-implicit and linear-implicit prediction for $\Delta t = 1\Delta t_{\text{CFL}}$ coincide and show variations in the CFL estimate in the range $\text{CFL}^e \in [0.8, 1.2]$. Further, the mean CFL shows almost direct scaling with the time step size increase Δt_{CFL} . This suggests that larger time steps potentially have minor influence since the flow features do not appear to change significantly. We will further discuss this in the accuracy comparison below.

In comparison, Dong and Shen [39] found that even larger time steps are possible using the linear-implicit scheme with updated advection velocity. They report a time step size study in Table 7 [39] for a qualitatively equivalent mesh, however, with a polynomial order $P = 3$. Their study shows a $\Delta t_{\text{max,ref}} = 40\Delta t_{\text{CFL,ref}}$ where $\Delta t_{\text{CFL,ref}} = 0.003$. It is interesting to note that they find a similar increase in the stability limit using the linear-implicit scheme despite using the updated advection velocity and a coarser mesh.

Influence of temporal accuracy

This section discusses the influence of larger time step size on the accuracy of predictions. On the one hand, we discuss integral quantities such as time-averaged drag and lift coefficients. On the other hand, we consider a spectral analysis of the time-resolved signal of the lift coefficient

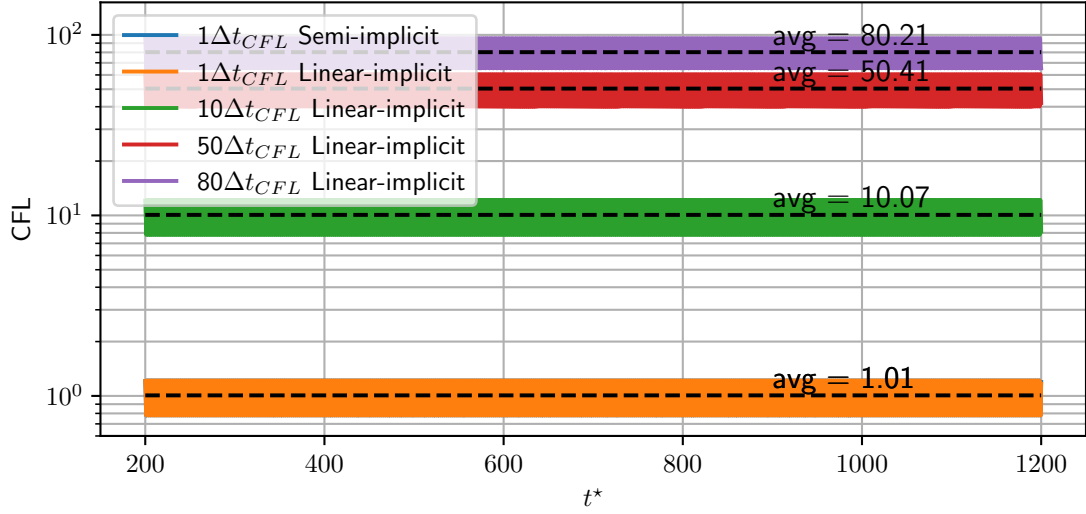
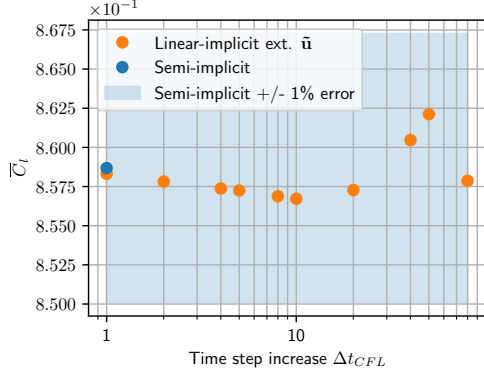


Figure 3.6: Estimation of the CFL number for different time step sizes and the time-averaged value for the 2D cylinder. The comparison for the semi-implicit and linear-implicit prediction at $\Delta t = 1\Delta t_{CFL}$ overlay almost exactly.

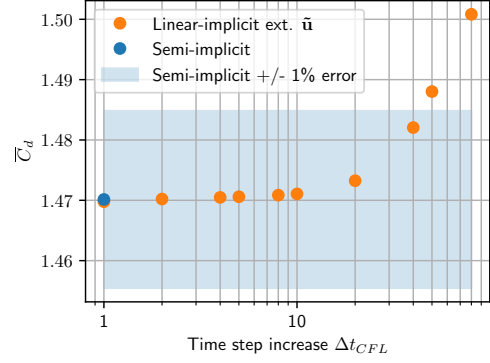
for insight into dispersion characteristics.

We begin the discussion with a comparison of the RMS of the lift coefficient and time-averaged drag coefficient in figure 3.7. The coefficients use a time interval of $950t^*$ for averaging or the RMS, respectively. Figures 3.7a and 3.7b show the lift and drag coefficients for the linear-implicit prediction at increasing Δt , and along with the semi-implicit prediction. The results indicate almost identical predictions with up to an order of magnitude increase in the time step size. The maximum difference against the semi-implicit prediction for $10\Delta t_{CFL}$ is 0.06% in drag and 0.22% in lift. For larger time step increases, the deviations grow stronger in both coefficients. The drag coefficient is consistently over-predicted with a maximum deviation at $80\Delta t_{CFL}$ of 2.09%. For the lift coefficient, the trend is not as consistent with an increase up to $50\Delta t_{CFL}$ after which it reduces again. The lift coefficient reaches a peak deviation from the semi-implicit of 0.4%.

We additionally show the bounds for a 1% error compared to the semi-implicit prediction in figures 3.7a and 3.7b. This illustrates the magnitude of any differences in the time-averaged predictions. It becomes clear that the time step increase has only a minor error for the cylinder case. It is likely that the high spatial resolution reduces the effect of error interactions between temporal and spatial error terms. As a consequence, we can identify that the temporal error is minor up to an order of magnitude increase in the time step size or an estimated CFL number $CFL^e \leq 10$. When further increasing the time step size, an error in the time-averaged and integral



(a) Time-averaged RMS lift coefficient $\overline{C}_{l,\text{RMS}}$.



(b) Time-averaged drag coefficient $\overline{C}_d$.

Figure 3.7: The time-averaged lift coefficient $\overline{C}_l$ and drag coefficient $\overline{C}_d$ for the cylinder case with increasing time step size. The blue area highlights the a 1% deviation from the semi-implicit prediction. The averaging interval is $T \in [250, 1200]t^*$.

quantities growth stronger which indicates that the temporal error becomes more dominant. Nevertheless, it is encouraging to see a minor impact on the accuracy for more than an order of magnitude increase in the time step size.

Next, we look into the influence of increasing time step size on the dispersion of signals in the in the frequency domain. We decompose the lift coefficient C_l over the interval $T = [200, 1200]t^*$ applying a discrete Fourier transformation (DFT). The frequency is non-dimensionalised for the cylinder case using the Strouhal number defined as

$$St = \frac{fD}{U}. \quad (3.9)$$

Figure 3.8 shows the magnitude of each frequency, also known as power spectral density (PSD), of the lift signal. We identify the shedding frequency of the most energetic vortices through the peak with the lowest Strouhal number. Using the semi-implicit prediction as reference, the most energetic vortices have a Strouhal number of about $St = 0.228$. The frequency corresponds well to the curve fit by Barkley and Henderson [83] which is reported up to $Re = 1000$ in the work of Wen and Lin [84]. Subsequent peaks with strictly decreasing energy appear at $St = \{0.68, 1.14, 1.59\}$. These less energetic peaks may be related to the vortex cascade, however, there is no strong evidence for that in experimental work on two-dimensional cylinder flow, see Wen and Lin [84]. It is possible that the subsequent peaks appear as a result of the numerical scheme. For Strouhal numbers $St > 10$, the energy continues to decrease until it reaches a noisy

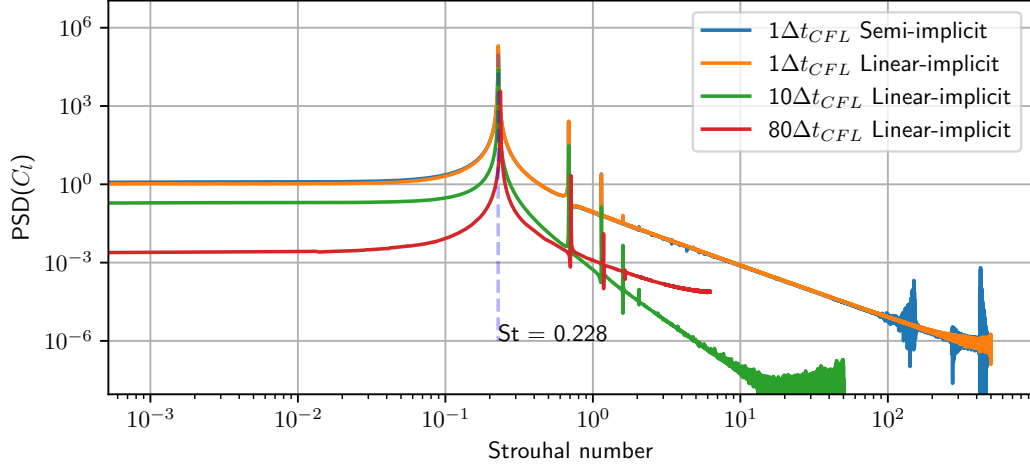


Figure 3.8: Power spectral density (PSD) of the lift coefficient signal in the interval $T = [200, 1200]$. Comparing the frequency spectrum for different time step sizes.

spectrum which indicates the limit of the mesh resolution.

We next turn our attention to the influence of increasing time step size. Figure 3.8 show the PSD of the lift coefficient for the linear-implicit scheme with increasing time step size up to the stability limit. These highlight the effect of large time step sizes on the lift prediction and the resolution of high frequency structures. At $1\Delta t_{CFL}$, the implicit scheme captures frequencies up to the respective Nyquist criteria of $St_{nyq.} = 1/(2\Delta t) = 500$. When we increase the time step to 10 and $80\Delta t_{CFL}$, the spectrum is cut-off at $St_{nyq.} = 50$ and 6.25, respectively. From the shift in the cut-off Strouhal number, it becomes clear that larger time step sizes constraint the range of scales that are resolved. Additionally, we observe that the largest time step size $80\Delta t_{CFL}$ does not resolve the spectrum down to the noisy range. Instead the constraint becomes severe enough to cut part of the spectrum and increases the energy within the resolved Strouhal numbers.

In figure 3.8, we see that the impact on the vortex shedding frequency is minor with essentially the same Strouhal number $St \approx 0.228$ for 1 and $10\Delta t_{CFL}$. At $80\Delta t_{CFL}$, the frequency begins to shift slightly to $St = 0.236$ which shows an influence on the vortex shedding. This shift in the frequency range is likely a result of the increase in CFL. Having a larger CFL number leads to faster travelling waves which influences the transition from boundary to shear flow and finally the vortex shedding.

Lastly, we observe a reduction in the magnitude of the PSD with increasing time step size. While both schemes with $1\Delta t_{CFL}$ show an almost identical spectrum, the larger time step size

$10\Delta t_{\text{CFL}}$ leads to a reduction in the amplitudes of the peak by an order of magnitude. Additionally, the gradient in the energy for $St > 0.228$ is steeper than for $1\Delta t_{\text{CFL}}$. For the largest time step size $80\Delta t_{\text{CFL}}$, the effect is even stronger with three orders of magnitude reduction in the PSD amplitudes. The gradient beyond the peak is shallower compared to $10\Delta t_{\text{CFL}}$. The reduction in energy for larger time step sizes is a direct result of the increase in numerical diffusion with larger time step sizes which impacts the entire frequency spectrum.

We find a strong increase in stability of almost two orders of magnitude is possible using the linear-implicit velocity correction scheme. Interestingly an upper stability limit still exists with the linear-implicit scheme which did not show for the steady Kovasznay flow in section 3.3.1. The origin of the upper stability limit is not clear. We further find that the impact of larger time step sizes on the accuracy is negligible up to a 10-fold increase over the semi-implicit schemes stability limit. We found a minor influence on time-averaged lift and drag coefficients as well as the power spectral density. Finally, choosing time steps near the upper stability limit leads to larger differences in the time-averaged and time-resolved quantities. In particular the time-averaged drag coefficient shows a consistent increase with larger time step sizes while the PSD indicates a cut-off near the dominant frequencies of the vortex shedding. All in all, the results show that a temporal over-resolution is appearing and even a 10-fold increase in the time step has minor effects on the accuracy of the simulation.

3.5 Stability and accuracy on complex geometries

The study on canonical problems in section 3.4 shows the possible stability improvements and effects on the accuracy at time step sizes above the $\text{CFL} \approx 1$ limit. We found time step increase of up to $200\Delta t_{\text{CFL}}$ still give stable predictions for long-time integrations. Additionally, we found that the accuracy is only affected for $\Delta t \gg 10\Delta t_{\text{CFL}}$. In this section, we turn our attention to problems with practical geometries and high Reynolds number flows which, in combination, lead to a challenging test case for accurate numerical simulations. We gather numerical evidence for the stability and accuracy of the linear-implicit velocity correction scheme when applied to such challenging test cases.

The geometries for the study is an inverted wing in ground effect. We detailed there significance for the automotive industry and in particular for Formula 1 (F1) race cars in the introduction, see chapter 1. These aerodynamic devices are critical for the front and rear of race cars where they produce the necessary downforce, that is negative lift. Race car designs use

wings due to their high lift-to-drag ratios which help with achieving fast lap times [85].

The main wing element generates most of the downforce, however, large chord lengths lead to flow separation which reduce the lift-to-drag ratio. Instead, adding one or multiple flaps allows increasing the lift, in particular at low speeds, without strong reductions in the lift-to-drag ratio [86]. The boundary layer flow between multicomponent wings is, however, complex in nature. The upstream element sheds a boundary layer which interacts with the downstream element and, at the same time, is re-energised by the flow in the gap between both elements. We consider this challenging test case ideal for establishing the suitability of the linear-implicit scheme in a practical setting.

In the following, we introduce the particular inverted wing geometry and introduce physical parameters of the simulations.

3.5.1 The Imperial Front Wing geometry

The object of study is the Imperial Front Wing (IFW). The IFW is a multicomponent inverted wing in ground effect and derives from the McLaren MP4-17D race car's front wing. This particular geometry has never been raced, but is based on the 2002 and 2003 F1 car. Since the IFW is openly accessible in the repository [25], it is amenable for scientific investigation and has been studied extensively in references [23, 24, 29, 32, 34, 87].

Figure 3.9 shows the full Imperial Front Wing (IFW) geometry available in reference [25]. The IFW is a multi-element and inverted wing in ground effect. The largest element, known as mainplane, has a chord length $c = 0.25$ m which serves as reference length. The IFW further has two flaps, namely, the flap 1 and the flap 2. The height above the ground normalised by the chord length h/c defines the distance from the moving ground to the end plate.

Overall, the IFW incorporates challenging geometric features such as curved surfaces in the stream- and span-wise directions as well as high Reynolds number flow. This leads to complex boundary layer physics, interactions between the elements of the wing and a vortex system forming in the wake. Next, we briefly review the studies on this geometry and describe the context of our study.

The experimental campaign in reference [23] focusses on the vortex system forming behind the wing. They investigate the influence of ride height h/c and a rotating wheel behind the front wing using laser-smoke flow visualisation, 2D particle image velocimetry (PIV), total pressure probes and hot-wire anemometry. The main conclusion for the IFW without wheel are the identification of a system of four co-rotating vortices and a merging of these vortices when a wheel is present.

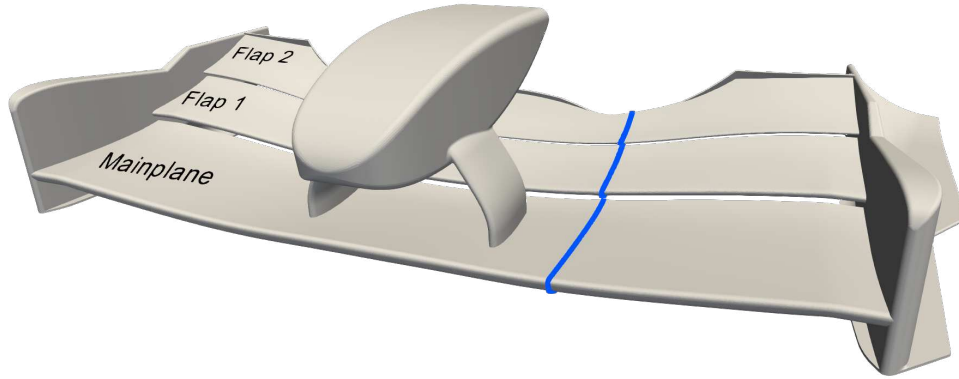


Figure 3.9: The imperial front wing (IFW) geometry; a F1 front wing based on the McLaren MP4-17D race car. The slice location at $y = 250$ mm for the 2D and 3D test case is highlighted in blue.

This initial investigation was later followed by a numerical investigation in reference [24] where Nektar++ is setup to reproduce the experiment with an implicit large eddy simulation (iLES). Lombard [24] reproduces the experimental results reproducing the four vortices of the IFW. He further investigates the variation with ride height h/c and finds a shift in the relative strength of each vortex. Similarly, the merging of the vortices is qualitatively reproduced with the iLES approach. His conclusion however mentions that a study of the required numerical resolution is necessary to confirm the accuracy of the results.

Reference [34] studied the IFW using iLES with Nektar++ and compared results against an experimental campaign. The ride height is fixed to $0.36h/c$ using the height provided in the regulations for F1 cars up to the year 2005. They find good agreement between PIV measurements of the energetic vortical structures and numerical predictions using Nektar++. Buscariolo et al. additionally use pressure-sensitive oil to visualise the boundary layer flow on the mainplane and flaps of the IFW. By comparing the flow visualisation against iLES predictions with varying resolution, they identify the minimum resolution to accurately predict transition to turbulence and re-attachment of the boundary layer on the first and second flap. This gives a qualitative comparison of the boundary flow and required resolution, but lacks insight into the transition phenomena and interaction between the three elements of the IFW.

This gap is further investigated in reference [32]. Slaughter et al. use a 2D slice of the original geometry highlighted in blue in figure 3.10 and extrude it with a quasi-3D approach. A detailed description of the quasi-3D approach is available in section 2.5. The extruded geometry removes

the geometric complexity of the full geometry, but allows a computationally feasible study of the boundary layer flow with high resolution.

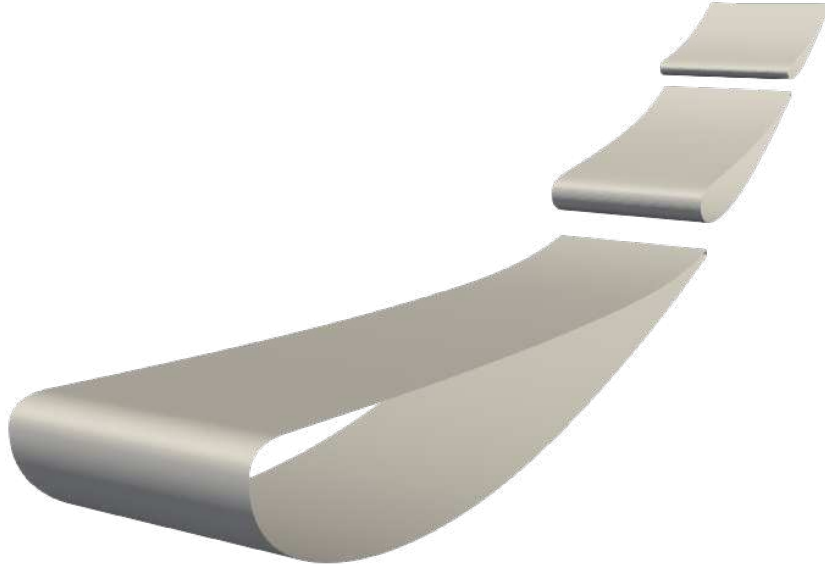


Figure 3.10: Slice of the imperial front wing (IFW) geometry for our investigations in 2D and 3D, and used in [29, 32, 87].

The study in [32] identifies laminar separation bubbles (LSB) occurring on the mainplane and flap 1 suction sides causing the transition to turbulence. These LSBs are thinned due to the ground effect on the suction side. Further, they identify a bypass transition on the flap 2 pressure side. The bypass, or wake transition, is likely induced from the upstream shedding off the trailing edges and the shear layers from mainplane and flap 1 suction sides. Overall, they report time-averaged force coefficients $C_l = -8.33$ and $C_d = 0.17$. Additionally, the spectral analysis highlights energetic peaks in the power spectral density (PSD) of the total force at Strouhal number $St = 30, 40, 60, 140$ and 200 . Note that the Strouhal number is the frequency f normalised by the freestream velocity and chord length $St = f \frac{c}{U}$. These transition patterns, force coefficients and spectral peaks are reproduced in subsequent studies as well, see table 3.4 for an overview.

Reference [29] uses the high-order discontinuous Galerkin solver Horses3D and compares an explicit LES (eLES) with Vreman model [14] against an iLES approach for the same geometry.

They find good agreement against the reference results in [32] matching the time-averaged wall-shear stress and pressure distributions for all elements. Their results further show a better agreement in the wall shear stress and pressure coefficient using the iLES approach rather than the eLES with Vreman model. Similarly, reference [87] uses the finite volume solver CharLES and identify the transition on all elements. They investigate the spatial resolution and find that a $\Delta y^+ \approx 2.5$ is sufficient to capture the LSB, however, note that the boundary layer growth rate and the number of elements in the LSB height is essential to resolve the transition with a finite-volume approach.

The mean force coefficients in table 3.4 show good agreement in the lift coefficients with differences up to 4.2% and, for drag, 4.1% difference.

Table 3.4: Overview of reported time-averaged force coefficients and distinct peaks in the PSD of the force coefficients from literature for the 3D extruded IFW geometry.

Source	$\overline{C}_l$	$\overline{C}_d$	PSD peaks in St
[32] Slaughter et al.	-8.33	0.17	30, 40, 60, 140, 200
[29] Ntoukas et al.	-8.6821	0.1725	23, 33, 50
[87] O’Sullivan and Ferrer	-8.212	0.177	25, 38, 53, 115, 158

As for the spectral analysis, [29] reports distinct peaks in the PSD of the total lift force at $St = 23, 33$ and 50. However, they do not find the higher frequency peaks from reference [32] at $St = 140$ and 200. Reference [87] finds peaks for Strouhal numbers $St = 25, 38, 53, 115$ and 158 with the most-refined mesh. We conclude that three lower peaks are commonly found in the Strouhal number range $St = [20, 60]$ for the extruded IFW slice (eIFW).

These studies on the 3D extruded IFW provide a challenging benchmark for identifying the characteristics of the linear-implicit scheme. It allows us to compare large- and small-scale phenomena at increasing time step sizes, observe the influence on the accuracy and discuss them in the context of cross-validated results from the literature. We will particularly aim at identifying stability limits, measured by the time step size, which are essential for reducing the time-to-solution as well as their impact on reproducing the transition phenomena, force coefficients and spectral analysis.

3.5.2 Numerical setup for the IFW

We investigate the flow over the extruded IFW slice and identify numerical characteristics of the linear-implicit scheme in comparison to the reference semi-implicit scheme. The initial investigation uses a 2D test case of the slice which serves as a low cost benchmark for finding

numerical characteristics, see the results in section 3.5.3. Having identified the characteristics in 2D, informs the study for the large-scale 3D simulations in section 3.5.4. In the following we will describe the preprocessing for both, 2D and 3D, simulations and highlight differences were relevant.

In comparison to the canonical cylinder problem, the IFW slice has complex geometric features involving boundary layers and curved surfaces in close proximity of each other. We generate the 2D and 3D meshes in figure 3.11 and 3.18 using the NekMesh industrial pipeline [45]. The pipeline creates an initial linear finite-element mesh using the commercial software Star-CCM+ and a provided computer-aided design (CAD) file. The linear mesh uses refinement zones in the volume and creates a prism layer on the boundary of the wing elements with a single prism, or quadrilateral in 2D, layer. Next, NekMesh reconstructs CAD and mesh connectivities ensuring compliance between the original geometry and the linear mesh. The reconstruction includes the linear vertex nodes as well as higher-order edge and face nodes to achieve a high surface quality. Subsequent steps apply curvature to the mesh following the curved surfaces of the geometry to represent important surfaces such as the leading edges accurately and avoid fishnet patterns. Finally, NekMesh splits the single prism element isoparametrically into a user-defined number of prism layers for fine-tuning the boundary layer meshes. Green et al. [44] describe the higher-order mesh generation in detail. These high-quality meshes enable an accurate simulation of the original geometry with minimal disturbances from the mesh.

The physical parameters for the stability and accuracy tests are a Reynolds number $Re = 2.2 \times 10^5$ in 2D and $Re = 1.72 \times 10^5$ in 3D, see table 3.1. Both Reynolds numbers use the main plane chord length $c = 0.25$ m as a reference and a farfield velocity $U = 1$. The kinematic viscosity ν follows from the above Reynolds numbers. The 2D case uses the same Reynolds number as in reference [32] and the 3D setup uses a different Reynolds number. We consider the impact of this difference on the physics as marginal.

All simulations are advanced from an initial Reynolds-averaged Navier-Stokes (RANS) solution. We compute the RANS solution using the commercial Software Star-CCM+ [88] and a $k - \omega$ turbulence model and interpolate it onto the higher-order mesh using the Nektar++ tool FieldConvert. Using the RANS initial guess, we advance the simulation for at least 20 convective time units t^* . A convective time unit is a non-dimensional time defined with $t^* = \frac{U}{c}t$ based on the physical time t it takes to convect a structure over the main chord c with the farfield velocity $U = 1$.

The accuracy comparison uses drag and lift force coefficients for comparison. Those are

defined equivalently to equation (3.8), however, the area is $A = c\ell_y$ with $\ell_y = 1$ for the 2D case and $\ell_y = 0.2c = 0.05$ m for the 3D case. Moreover, we use the pressure and skin friction coefficient to investigate the near-wall flow structure. The pressure coefficient is

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho U^2 A} \quad (3.10)$$

with the freestream pressure $p_\infty = 0$. Note that the non-dimensional formulation uses a unit density $\rho = 1$ and unit farfield velocity, see section 2.1. The skin friction coefficient is

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho U^2 A}, \quad (3.11)$$

where τ_w is the wall-shear stress using the standard definition

$$\tau_w = \nu \left. \frac{\partial u}{\partial y} \right|_{y=0}. \quad (3.12)$$

3.5.3 2D Imperial Front Wing

The 2D Imperial Front Wing (IFW) case aims at resolving the turbulent flow close to the aerodynamic surfaces where the transition to turbulence and integral forces are most sensitive. Since we do not have reference data for the 2D case, we compare predictions from the semi-implicit scheme against the linear-implicit scheme with the same and larger time step sizes.

An overview of the computational domain and mesh is available in figure 3.11. We highlight the computational domain in figure 3.11a which shows the dimensions in the stream-wise and vertical directions. The stream-wise and vertical length use the chord length c for scale and both are defined such that blockage effects are negligible. Further, the boundary conditions for the case are a uniform inflow profile, a moving floor at the bottom, slip boundary conditions at the top and robust outflow boundary conditions following equations (2.53) and (2.54). Note that pressure uses higher-order Neumann boundary conditions on all boundaries except for the outflow. The higher-order Neumann conditions follow from the boundary integral term in equations (2.28) or (2.27) for the semi-implicit and linear-implicit formulation, respectively. The wing profile surfaces use no-slip boundary conditions.

Figures 3.11b and 3.11c show the far- and near-field of the computational mesh. The unstructured mesh uses triangular elements for the volume, or rather surface, and quadrilateral elements for the boundary layer mesh on the wing profiles. We use coarse refinement regions

around the wing profiles and for the wake with finer mesh elements closer to the wing. While the wake still experiences stronger diffusion due to the low refinement downstream of the wings, the near-field uses finer mesh elements.

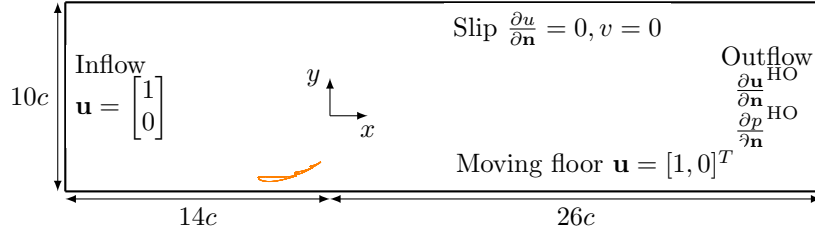
The boundary layer on the wing uses ten layers with an initial height of $71\mu m$ leading to a wall unit of $\Delta y^+ = 0.13$. The boundary layer growth rate is set to 1.4. The setup follows the work of Slaughter et al. [32] which shows that it resolves the three-dimensional laminar and turbulent boundary layer flow. The floor boundary layer, not shown in figure 3.11c, does not require as high a resolution and uses five layers in order to resolve the thin laminar boundary layer.

The study begins with an overview of the identified stability limit and continues with a comparison of accuracy with increasing time step size Δt up to this limit. We compare accuracy using the time-averaged lift and drag coefficients, their power spectral density (PSD) and pressure and skin friction coefficients. The integral coefficients hint at the overall resolution of the turbulent flow. Meanwhile, the PSD gives insight into the spatio-temporal scales of the flow and their resolution. Further, surface quantities such as the pressure C_p and skin friction coefficient C_f give insight into the resolution of the transition to turbulence on the wing profiles.

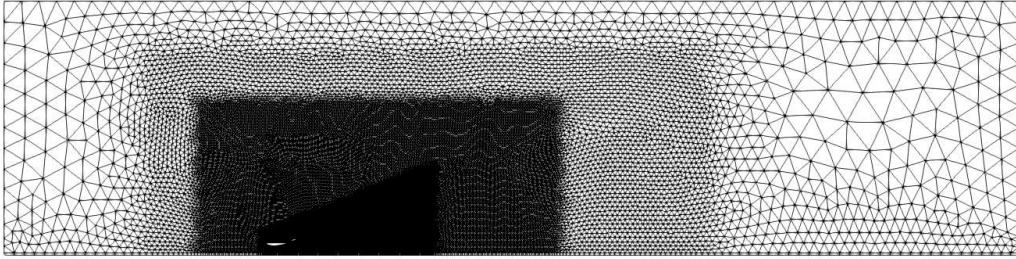
Practical stability limitations

The stability tests follow the procedure described for the cylinder flow above in section 3.4.1. For the 2D IFW, we initialise the flow from a Reynolds-averaged Navier-Stokes (RANS) as described above in section 3.5.2 and advance the solution for at least $40t^*$. At this point in time, the solution advanced through the initial transient and reached the quasi-steady state. With this setup, we investigate the first research question of reaching the quasi steady-state faster through larger time step sizes, however, without investigating the actual performance.

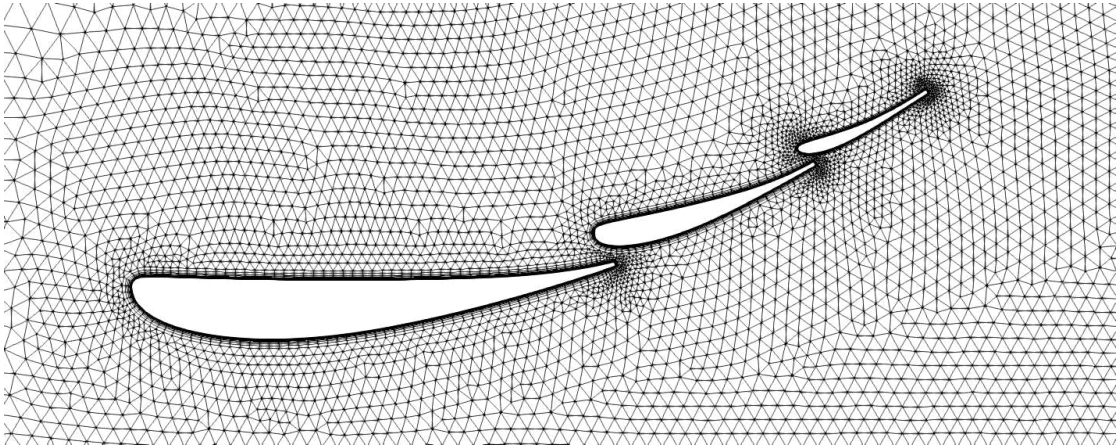
Figure 3.12 shows the CFL estimate for both schemes at various time step sizes. The semi-implicit scheme is stable up to a time step size $1\Delta t_{\text{CFL}} = 5 \times 10^{-6}$. Interestingly, the velocity field for this time step reaches a maximum CFL larger than 1 with $\overline{CFL} = 1.8$ when time-averaged over $40t^*$. While we would expect the semi-implicit scheme to be only stable up to a CFL of 1, it is possible that the CFL estimator does not accurately account for the curvature of mesh elements close to the surface or the geometric singularity at the trailing edge, see also section 2.3.4. We further observe that the largest CFL values are located at the leading and trailing edge of the wing profiles which is where the flow reaches the highest velocity magnitude and fine boundary layers. This is particularly critical at the trailing edge where the kink in the geometry



(a) Computational domain and boundary conditions with $2\times$ scaled wing profiles for visibility.



(b) The farfield mesh with refinement zones around the wing profiles.



(c) The nearfield mesh highlighting the boundary layer mesh.

Figure 3.11: Overview of the computational domain, boundary conditions and meshing approach for the two-dimensional Imperial Front Wing case.

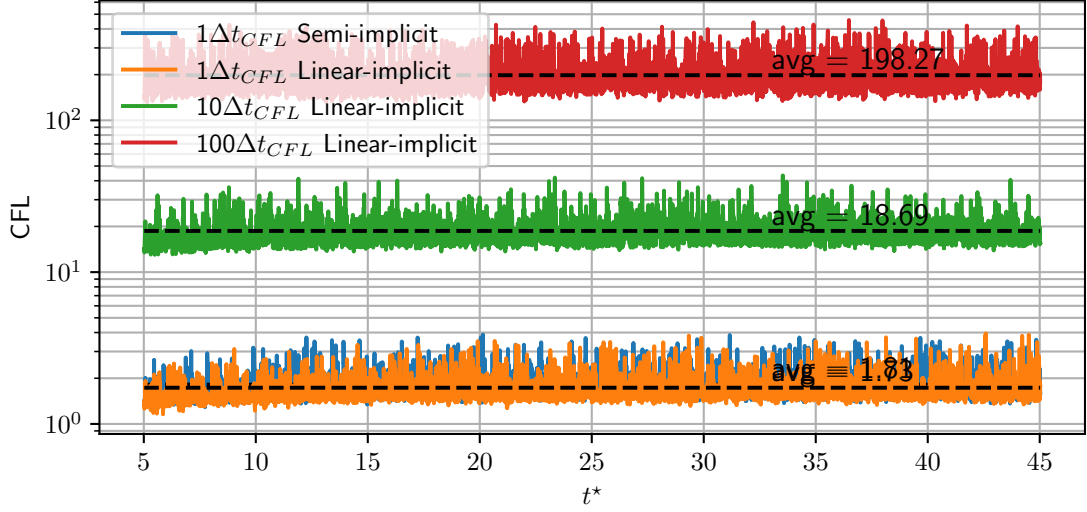


Figure 3.12: Estimation of the CFL number for different time step sizes and the time-averaged value for the 2D IFW.

leads to a strong flow component orthogonal to the boundary layer elements. Therefore, we expect the largest CFL numbers around this location.

In contrast, figure 3.12 shows the linear-implicit scheme is stable for time step sizes of $\{1, 10, 100\}\Delta t_{CFL}$. The increase in time-averaged CFL is almost linear for an order of magnitude increase with $\overline{CFL} = 18.69$. At the largest time step, the time-averaged $\overline{CFL} = 198.27$ is two orders of magnitude larger than at the stability limit of the semi-implicit scheme. This increase however shows a superlinear scaling of the CFL estimate suggesting that the turbulent flow is reaching a larger maximum velocity due to the large time step size. We will investigate this in the accuracy comparison below.

It is interesting to note that the increase in the relative time step size of $100\Delta t_{CFL}$ is even larger than for the cylinder case in section 3.4.1 where we observe $80\Delta t_{CFL}$. In comparison, the cylinder setup does not use the spectral vanishing viscosity while having a flow at Reynolds number 500. It is possible that the spatial error of any under-resolved scales of the simulation are interacting with larger temporal errors through choosing larger time step sizes. This leads to instabilities in the algorithm which decrease the stability margin for the cylinder setup.

Overall, the stability improvement through the linear-implicit scheme is encouraging. We would generally not expect an upper stability limit with the linear-implicit scheme as the literature review in section 3.1 shows. Still, we find an upper stability limit at about $100\Delta t_{CFL}$ or

a time-averaged $\overline{CFL} \approx 200$. The origin of this stability limit requires a more in-depth study of the test cases which is outside the scope of this work.

The next section discusses the influence of choosing larger time steps and, hence having larger temporal errors, on the accuracy of the flow field.

Influence of temporal accuracy

We investigate the accuracy with varying time step size for both, the semi-implicit and linear-implicit, schemes. As mentioned above, we extend the simulations for $40t^*$ where the flow is in a quasi-steady state. In this state, the integral forces vary only due to the chaotic nature of the turbulent flow and we sample them for time-averaging and spectral decomposition. Our results are reduced to forces acting on the main plane in order to focus the discussion on the numerical features of the time-stepping schemes rather than the complexity of the geometry.

We compare the overall accuracy of capturing lift and drag forces in figure 3.13. Both, lift and drag coefficients, show a similar trend with increasing time step size: the prediction is almost equivalent for the semi-implicit and linear-implicit scheme for time steps up to $10\Delta t_{CFL}$. The difference in this range relative to the semi-implicit prediction with $\Delta t = \Delta t_{CFL}$ reaches a maximum of 0.6% and 1.1% for lift and drag, respectively. Further, increasing the time step size to an time-averaged $\overline{CFL} > 18.4$ shows larger differences for the lift coefficient of 1.9% at $20\Delta t_{CFL}$ and 3.5% at the largest time step size $100\Delta t_{CFL}$. Note that the drag coefficient shows larger relative differences for these time steps of 4.8% and 6.8% for $20\Delta t_{CFL}$ and $100\Delta t_{CFL}$, respectively.

The above results show a minor influence of the increasing time step size on the time-averaged forces. This result is interesting as it suggests that large increases in the time step size are possible without strong influence on the resolution of the turbulent flow. Considering figure 3.13, it indicates that the spatial resolution for this problem is higher than the temporal resolution up to a time step size $\Delta t = 10\Delta t_{CFL}$. For larger time step sizes $\Delta t > 10\Delta t_{CFL}$, we observe a stronger difference in predicting the forces. These differences are likely stronger due to the temporal and spatial errors having a similar magnitude and, hence, their interactions becoming stronger. While the numerical error is difficult to predict, sensitive surface quantities such as the pressure and skin friction coefficient give insight into the transition to turbulence which effect the forces.

Figure 3.14 shows the time-averaged pressure coefficient $\overline{C_p}$ for the semi-implicit scheme at the limiting time step size and the linear-implicit scheme for increasing time step sizes. We observe that both time-stepping schemes give almost identical predictions in figure 3.14a for the

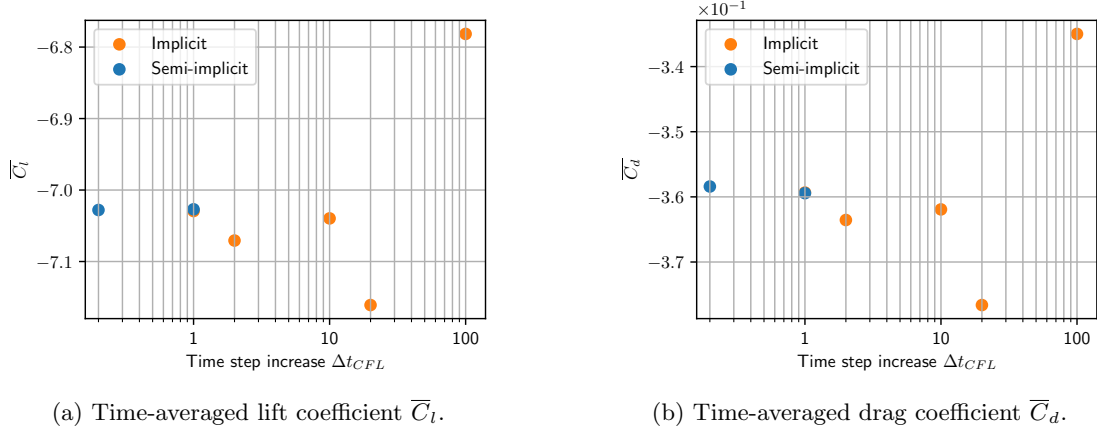


Figure 3.13: The time-averaged lift coefficient $\overline{C}_{l,RMS}$ and drag coefficient $\overline{C}_d$ for the 2D IFW case with increasing time step size. The averaging interval is $T \in [30, 35]t^*$.

pressure side of the main plane where $C_p > 0$. This is expected, because the flow on the top side of the main plane is simply laminar flow without transition turbulence. On the suction side, where $C_p < 0$, we see a significant difference between the linear-implicit prediction with the largest time step size $100\Delta t_{CFL}$ and all other predictions. At the largest time step size, the flow predicts a lower magnitude on the negative peak around $x/c = 0.05$. Moreover, the pressure close to the transition at $x/c = 0.4$ follows a different trend. Figure 3.14b enlarges the area around the transition location and indicates that for $100\Delta t_{CFL}$ the pressure does not drop which is a good indicator of the transition to turbulence. Besides, we find minor differences in the prediction of the pressure coefficient for all other time step sizes $\Delta t < 100\Delta t_{CFL}$.

We further investigate the influence of the time step size on the transition using the skin friction coefficient in figure 3.15. In particular figure 3.15a, shows a similar trend in the skin friction for all time step sizes except for $100\Delta t_{CFL}$. At the largest time step size, the skin friction is under-predicted along the suction side visible in the range $x/c \in [0.0, 0.8]$. The zoom in figure 3.15b further shows the difference around the transition location. For a time step size $\Delta t = 100\Delta t_{CFL}$, we do not see the skin friction coefficient reaching zero which indicates that this large time step predicts a laminar flow along the length of the main plane.

The smaller time step sizes show a different characteristic. They reach zero magnitude in the skin friction at $x/c = 0.35$ and a strong increase in C_f at $x/c = 0.41$. This is typical for the transition from laminar to turbulent flow showing an increase in friction. Interestingly, the skin friction reduces for $x/c > 0.41$ and reaches zero a second time at $x/c = 0.48$. Further downstream, the skin friction increases and reaches a maximum of $C_f = 0.004$ at $x/c = 0.53$. This double zero

in the skin friction suggests an initial transition, followed by a re-laminarisation which is followed again by a second transition to turbulence. This effect does not appear physical. It is possible that the resolution is insufficient to resolve the transition accurately and any instabilities that naturally lead to a transition are damped through the spectral vanishing viscosity.

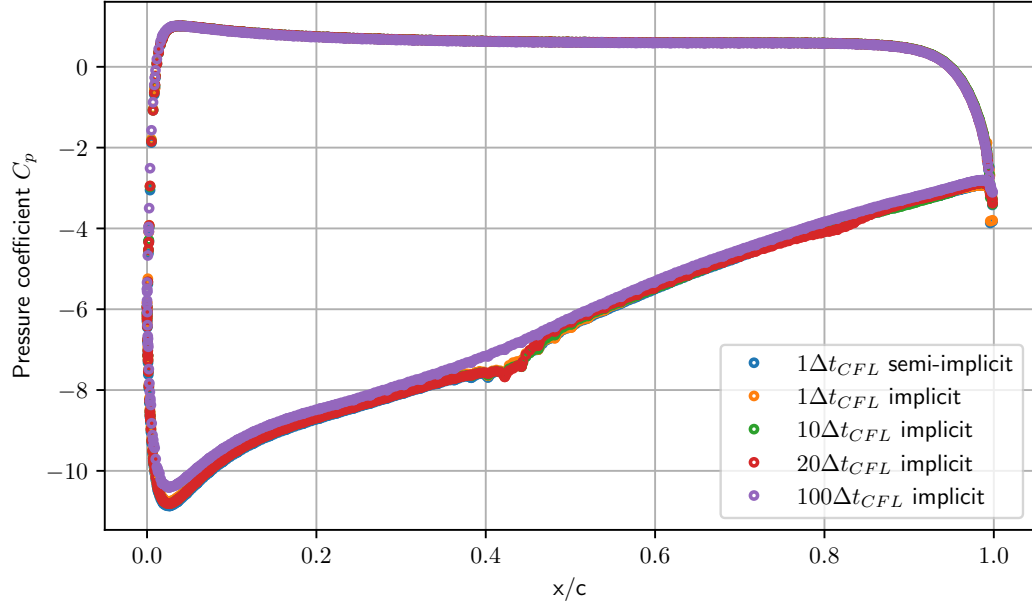
Additionally, it is worth noting that the prediction with linear-implicit scheme and $\Delta t = 20\Delta t_{\text{CFL}}$ shows the same re-laminarisation and double transition. However, it transitions further upstream at $x/c = 0.46$ compared to the prediction with $\Delta t < 20\Delta t_{\text{CFL}}$ at $x/c = 0.48$. Also, it shows a significantly higher skin friction peak downstream of the second transition with reaching up to $C_f = 0.01$, that is more than twice the skin friction of smaller time steps.

The accuracy of predictions with linear-implicit scheme and $\Delta t < 20\Delta t_{\text{CFL}}$ consistently predict similar surface quantities as well as forces on the main plane. This supports the above claim that the temporal error is insignificant up to $10\Delta t_{\text{CFL}}$. $20\Delta t_{\text{CFL}}$ shows an overall larger skin friction and pressure coefficient compared to predictions with $\Delta t < 20\Delta t_{\text{CFL}}$. The increase in both surface quantities explains the increase in the magnitude of the time-averaged lift and drag forces discussed above. $100\Delta t_{\text{CFL}}$ does not predict transition to turbulence as indicated through skin friction and pressure distribution on the main plane. This explains the strong reduction in the magnitude of both lift and drag force discussed above. Further insight into how the large time steps influence the physics are possible through frequency analysis.

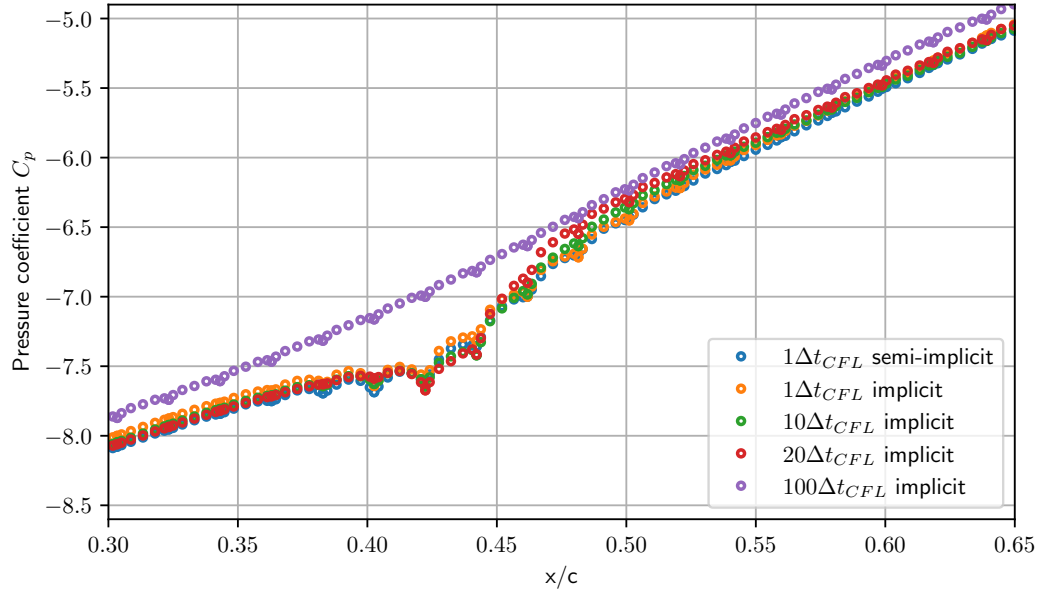
The power spectral density (PSD) gives insight into the spatial and temporal resolution of flow structures. We particularly apply Welch's method [89] in order to filter the raw Fourier-transformed signal and significant characteristics. Figure 3.16 shows the decomposition of the raw lift coefficient over a period of $T \in [30, 35]t^*$.

We observe that the semi-implicit and linear-implicit scheme behave differently for $\Delta t = \Delta t_{\text{CFL}}$. The PSD appears similar for $St < 60$ with a maximum in the energy around $St = 10$ and a peak at $St \in [40, 50]$. For $St > 50$, both schemes give a different prediction. On the one hand, the semi-implicit scheme predicts a third peak at $St = 65$ and a turbulent spectrum with a decay rate of $-5/3$ up to $St = 500$. For $St > 500$, the semi-implicit prediction decays sharply and reaches the white noise level at $St \approx 2000$. On the other hand, the linear-implicit prediction does not show a third peak and decays stronger for $St > 50$ with a rate of $-30/3$. The linear-implicit PSD reaches the white noise level at $St \approx 1500$.

It is interesting that both schemes give the same prediction up to $St = 50$, but differ in the decay of energy at higher frequencies. The rate of decay shows a strong difference of factor 6. This hints at a different dissipation characteristic of both time-stepping schemes.

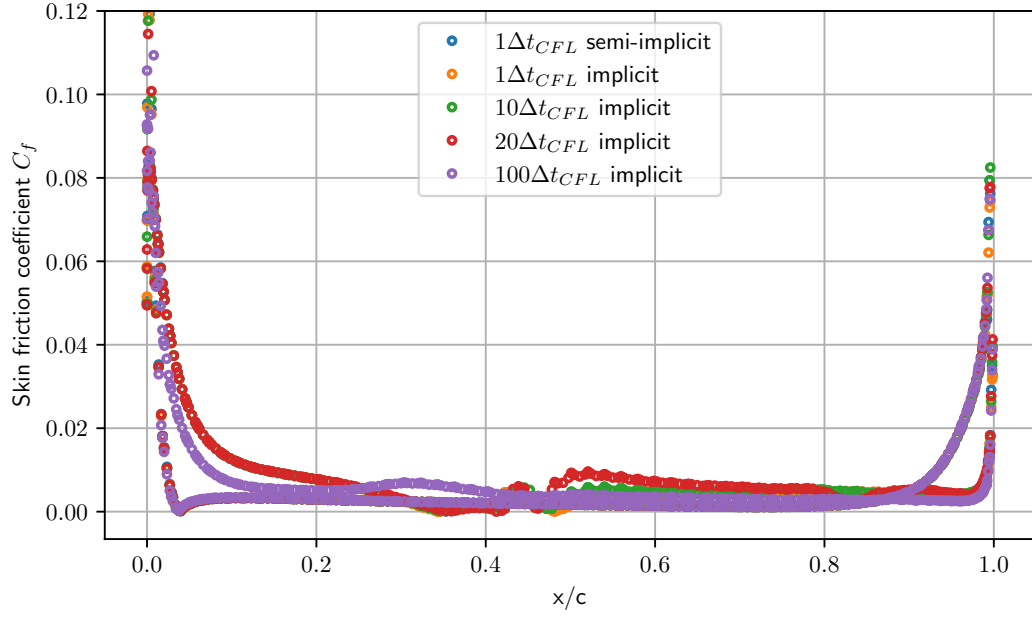


(a) Time-averaged pressure coefficient C_p .

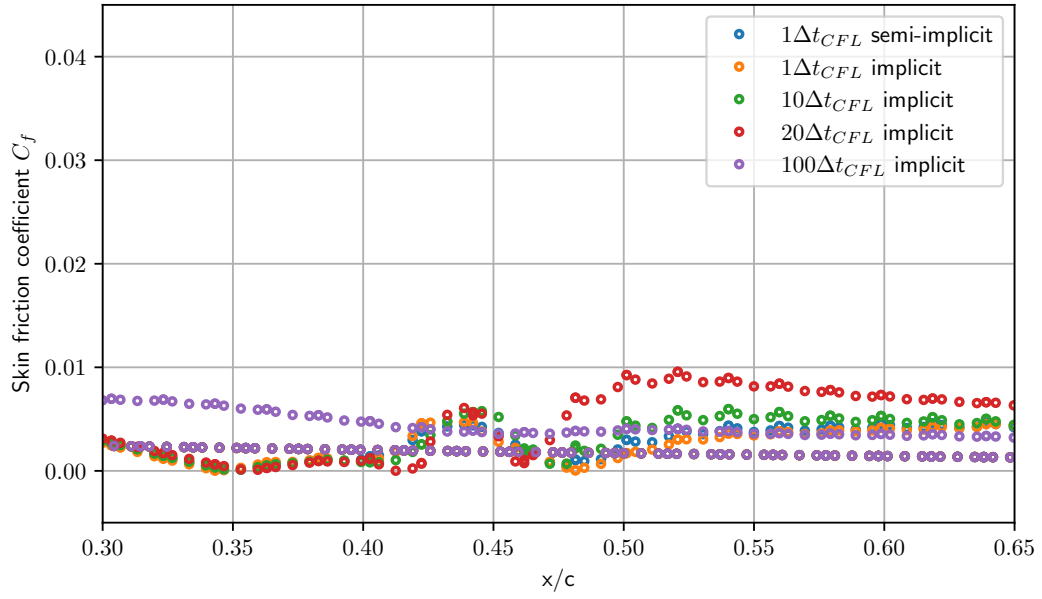


(b) Zoom around the transition location.

Figure 3.14: The time-averaged pressure coefficient C_p for the 2D IFW case showing only the main plane. Comparison of increasing time step size effect averaging over the interval $T \in [15, 25]t^*$.



(a) Time-averaged skin friction coefficient C_f .



(b) Zoom around the transition location.

Figure 3.15: The time-averaged skin-friction coefficient C_f for the main plane of the 2D IFW case. Comparison of increasing time step size effect using an averaging interval of $T \in [15, 25]t^*$.

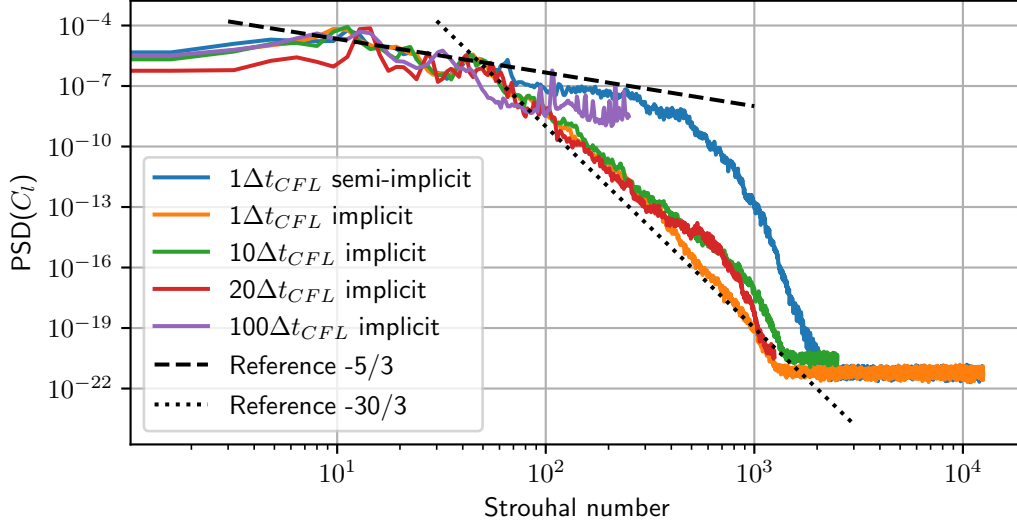


Figure 3.16: The power spectral density (PSD) of the lift coefficient using the semi-implicit scheme with strong pressure and linear-implicit with weak pressure formulation.

The linear-implicit scheme predicts equivalent PSDs for 1 and $10\Delta t_{CFL}$. Subtle differences for $20\Delta t_{CFL}$ where pronounced peaks appear at low frequencies including $St = 15, 30, 40$. $100\Delta t_{CFL}$ is similar at low frequencies, but cannot resolve frequencies for $St > 250$ due to the sampling frequency of $St = 500$ and cutting-off using the Nyquist criterion. Additionally, the large Δt leads to a hyper viscosity type effect with a reduction in energy for $St > 50$ compared to smaller $\Delta t < 100\Delta t_{CFL}$. Also, we find high energy content close to the Nyquist frequency introducing peaks in the intermediate frequency range of $St \in [40, 250]$.

The low-frequency spectrum is similarly resolved for both schemes and time steps $\Delta t < 20\Delta t_{CFL}$. For higher frequency content, we see a stronger dissipation of the linear-implicit scheme with a rate of $-30/3$ compared to the semi-implicit scheme with $-5/3$. Moreover, this dissipation is independent of the time step size increase. Additionally, we observe that more than an order of magnitude increase in the time step size introduces secondary frequencies for this problem. Likely, these appears due to the re-laminarisation at $\Delta t = 20\Delta t_{CFL}$ that we discussed for figure 3.15. In particular, the largest time step $\Delta t = 100\Delta t_{CFL}$ does not resolve the dissipative scales of the problem and introduces energy peaks close to the Nyquist frequency.

In summary, we find that the linear-implicit scheme gives similar predictions to the semi-implicit scheme up to $\Delta t = 10\Delta t_{CFL}$. For these time step sizes, the physical behaviour of the solution appears equivalent in terms of the forces and the transition to turbulence. Further

increases in the time step size introduce more dissipation and eventually influence the turbulent flow sufficiently to predict a laminar flow without transition to turbulence. We further found that the linear-implicit scheme shows a distinct dissipation behaviour with a stronger energy decay in the PSD. Despite this fundamental difference, the predictions of time-averaged forces and surface quantities are not significantly influenced. Next, we will briefly investigate the origin of the different dissipation behaviour between both schemes.

Effect of the pressure formulation

The mismatch in the PSD of the linear-implicit and semi-implicit in figure 3.16 motivate an investigation into the diffusion and dispersion behaviour of the linear-implicit scheme. This analysis however is an involved process. Instead, we find that the pressure formulations, introduced in section 2.2.3, have an effect on the dissipation at higher frequencies.

Figure 3.17 shows the PSD of both time-stepping schemes with weak and strong pressure formulation. All predictions use the same time step size $\Delta t = 5 \times 10^{-6}$ and a Taylor-Hood expansion in space with polynomial order $P = 5$ for the velocity and $P = 4$ for the pressure space. The power spectrum is equally resolved for the four schemes up to a Strouhal number $St = 50$. For $St > 50$, we observe that the behaviour is similar independent of the semi-implicit or linear-implicit scheme. It does however show a distinct behaviour for the weak and strong pressure formulation. The strong pressure form shows a turbulent spectrum similar to the $-5/3$ power law of isotropic turbulence. Further, the strong form shows a decay for $St > 100$ due to the spectral vanishing viscosity dampening high frequency energy. In contrast, the weak pressure formulation shows a consistent decay for $St > 50$ with a steeper slope of $-30/3$. Still, the different formulations reach the white noise level of machine accuracy at similar Strouhal numbers in the range $St \in [1200, 2200]$.

The comparison suggests that the pressure formulation influences the dissipation behaviour of the velocity-correction scheme. To best of the authors knowledge, this issue is not discussed in the literature and more insight would be valuable. We also note that the semi-implicit scheme uses extrapolation for the advection and diffusion terms in both pressure formulations while the linear-implicit scheme does not extrapolate either term to second order.

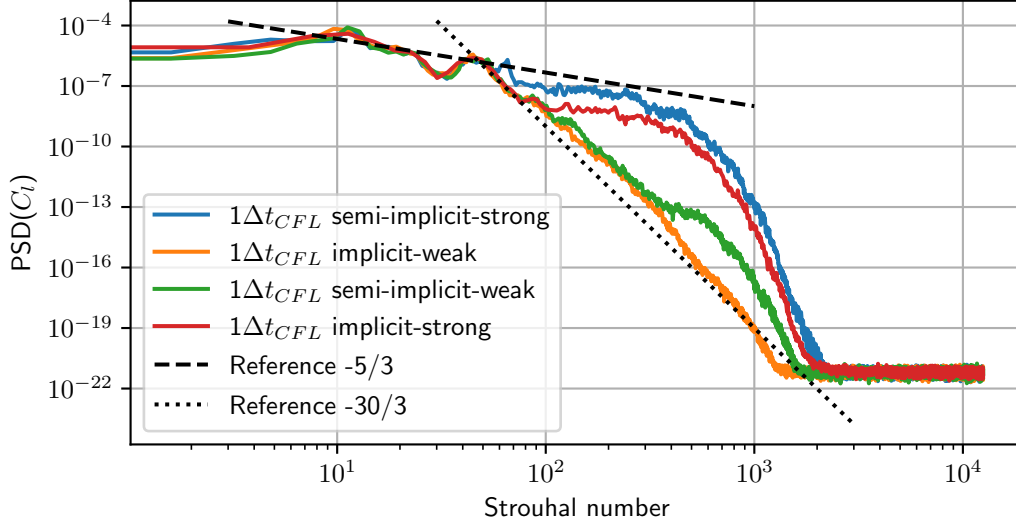


Figure 3.17: The power spectral density (PSD) of the lift coefficient comparing the strong and weak pressure formulation with both semi-implicit and linear-implicit scheme at the same time step size $\Delta t = \Delta t_{CFL}$.

3.5.4 3D extruded Imperial Front Wing

We extend the above study to a 3D extrusion of the 2D slice, see figure 3.10. Our efforts focus on resolving the laminar, transitional and turbulent flow around the wing elements. The computational domain in figure 3.18a shows the dimensions of the volume. The lengths are largely based on the work by Slaughter et al. [32]. Following their span-wise sensitivity study, we chose a span-wise domain length of $\ell_y = 0.2c = 0.05$ m. Slaughter et al. show that $0.2c$ is sufficient to resolve all flow features except for the far wake which is not the aim of our analysis.

The inflow uses a uniform velocity profile with farfield velocity $U = 1$. Similarly, the lower wall is moving at farfield velocity to emulate the relative motion of a moving car over a stationary road. The ride height follows the reference data in [32] with $h = 0.25c$. We further apply a slip boundary condition for the top wall assuming negligible blockage effects with a height of $12c$. The side walls use slip boundary conditions. O’Sullivan and Ferrer [87] show that slip boundaries influence the flow near the side walls and effect the integral forces. Still, their effect on the transition to turbulence is negligible across most of the span. Finally, the outflow uses the robust outflow boundary conditions for velocity using equation (2.54). The pressure uses the higher-order Neumann boundary conditions for all boundaries except for the outflow where we apply the robust boundary condition in equation (2.53).

The meshing approach focusses on the nearfield flow characteristics. On the one hand, we resolve the boundary flow with transition to turbulence accurately and refine the nearfield heavily. On the other hand, we use low resolution for the farfield wake. The moving floor uses a layer of prism elements in order to refine a laminar separation bubble that was reported by Slaughter et al. [32]. Similarly, the wing profiles use a layer of prismatic elements to resolve the turbulent boundary layer. Figure 3.18d shows the fine span-wise refinement on the main plane wing. We use an almost homogeneous refinement across the span resolve the fine scales.

The boundary layer mesh on the main plane uses a maximum height of 2.5 mm. The finest prismatic layer measures 90 μm in height and uses a growth rate of 1.2 over 10 elements. This fine meshing setup follows the resolution used in Slaughter et al. [32]. Figure 3.19b below shows that the setup is sufficient to resolve the turbulent boundary layer with vertical wall units $\Delta z^+ < 1$.

Overall the mesh is highly resolved in order to minimise the effect of spatial errors. Similar to the previous studies in this chapter, this better isolates the effect of temporal errors and enables an analysis of the influence of larger time step sizes. Next, we discuss how accurately we reproduce the reference results.

Reproducing reference data in [32]

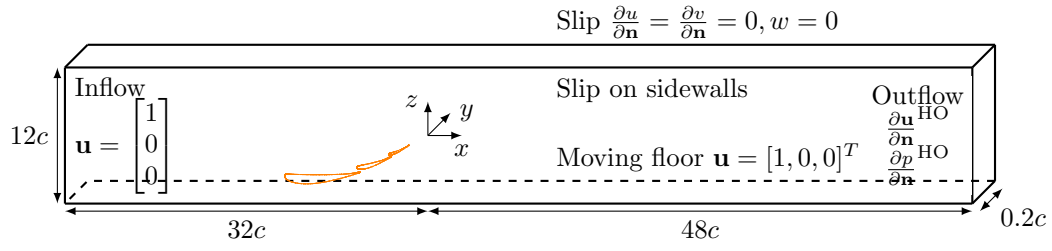
The eIFW investigation ideally resolves the transition to turbulence for the three wing elements as described in the reference. In order to validate our mesh, we estimate the spatial resolution of the boundary layer using wall units for each coordinate direction. The wall units are defined through the mean friction velocity $u_\tau = \sqrt{\frac{\tau_w}{\rho}}$ as

$$\Delta[x^+, y^+, z^+] = \frac{u_\tau \Delta[x, y, z]}{\nu}, \quad (3.13)$$

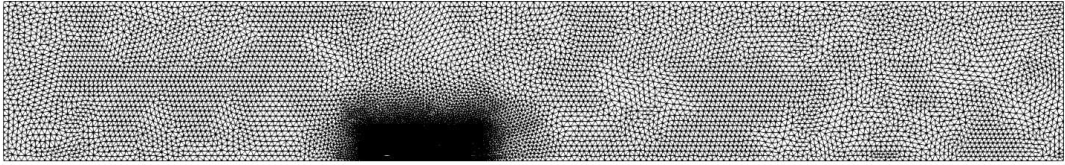
where $\Delta[x, y, z]$ are the nominal spacings between quadrature points.

We approximate the nominal spacings for the x - and y -coordinate by finding the nearest vertex on a wing's surface and taking the x - and y -distance to that vertex. The vertical distance Δz is just the height of the initial boundary layer element. The three distances $\Delta[x, y, z]$ represent the spacing of the linear mesh. We however use a higher-order approximation with Gauss-Lobatto-Legendre distributed points in between the vertices. Therefore, we scale the distance by the polynomial order P .

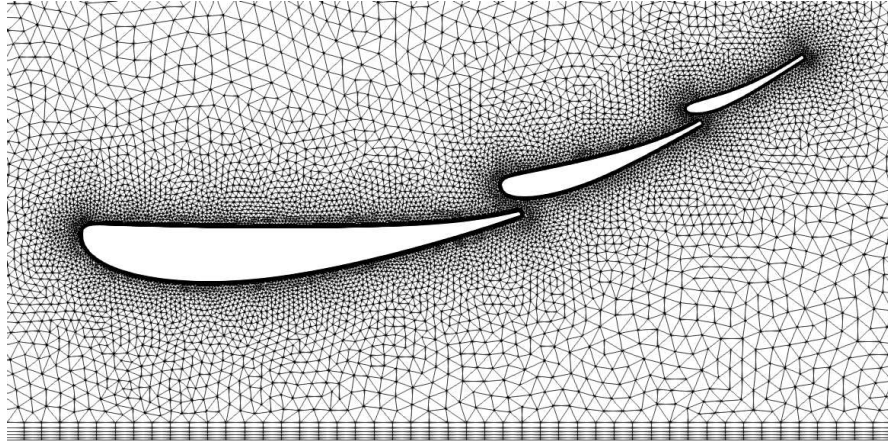
Figure 3.19 presents the estimated wall units along with empirical limits for LES by Georgiadis et al. [3]. We find good agreement for the streamwise and vertical wall units with resolution well



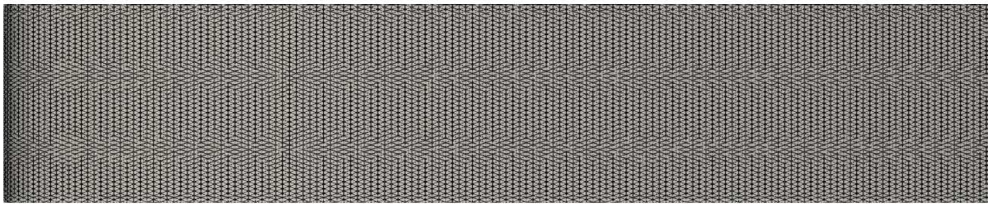
(a) Computational domain and boundary conditions with $4\times$ scaled wing profiles for visibility.



(b) The farfield mesh with refinement zone around the wing profiles.



(c) The nearfield mesh highlighting the boundary layer mesh on the wings and floor.



(d) The surface mesh of the main plane wing element.

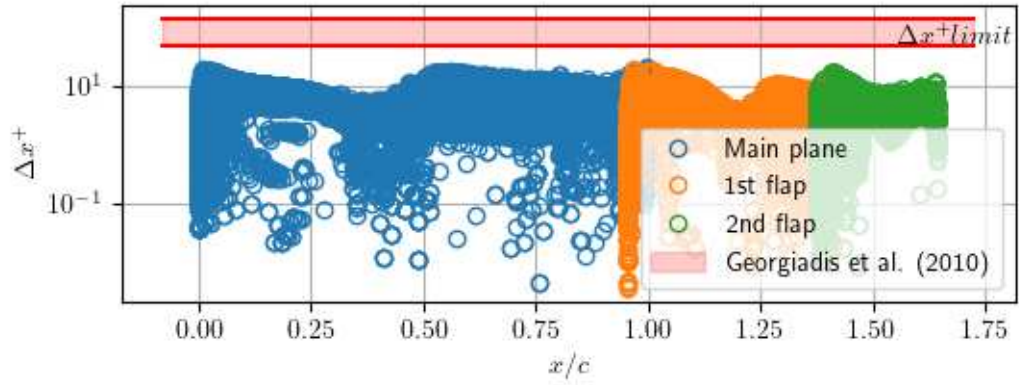
Figure 3.18: Overview of the computational domain, boundary conditions and meshing approach for the extruded Imperial Front Wing case.

below the empirical limits. For the spanwise wall unit Δy^+ , we find that the leading and trailing edge of all wing elements reach a maximum that extends into the limits. Nevertheless, the leading edge flow is laminar and does not require too high resolution and the trailing edge flow does not have an attached boundary layer flow, but a recirculation region. Therefore, the mismatch is acceptable in these regions and we conclude a good resolution of the boundary layer using our meshing approach. We note that the wall unit limitation are based on flat-plate geometries and are not necessarily good indicators for our case with highly anisotropic turbulence along multiple curved surfaces.

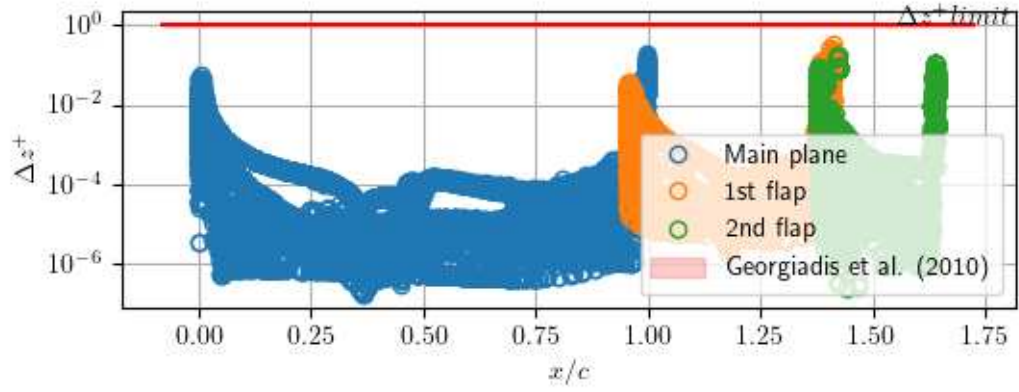
We further confirm capturing the transition process on the wing profiles against reference data by Slaughter et al. [32]. The comparison uses the skin friction coefficient C_f in equation (3.11) and takes the time-average. The time-averaged skin friction coefficient gives insight into the sensitive transition phenomena and highlights the transition location with $\overline{C_f} = 0$ indicating zero wall shear stress.

Figure 3.20 shows the mean skin friction coefficient for our most-refined *Mesh D* compared to the reference results. We use the semi-implicit scheme with the maximum stable time step size $\Delta t = 1\Delta t_{\text{CFL}} = 1 \times 10^{-5}$. In comparison, the reference simulation used an even smaller time step size of $0.4\Delta t_{\text{CFL}}$. Overall, we observe good agreement in the skin friction coefficient. For the main plane, we see strong similarity in the initial peak, transition location around $x/c = 0.5$ and the following increase in skin friction due to the turbulent boundary layer. Additionally, the transition on the first flap at $x/c = 1.25$ is well matched with a slight under-prediction of the friction after the transition to turbulence. Finally, the bypass transition on the upper surface of the second flap is equally well-predicted. The overall reproduction of sensitive flow phenomena is encouraging.

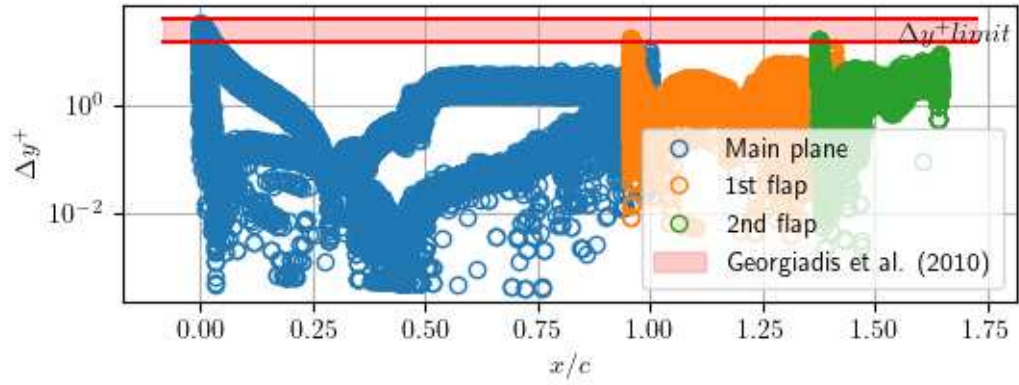
Figure 3.21 shows the power spectral density (PSD) of the lift coefficient for both, the validation data and a prediction with the semi-implicit scheme. The PSD clearly shows three energetic frequencies for the semi-implicit prediction. The peaks are located at $St = \{20, 30, 40\}$ and are partially matching the reference data of Slaughter et al. Slaughter et al. report the peaks with equivalently increasing magnitude at $St = \{30, 40, 60\}$. It is interesting that a similar mismatch is found in the works by Ntoukas et al. [29] and O’Sullivan and Ferrer [87], see also table 3.4. Further, the semi-implicit prediction does not show the higher Strouhal number peaks reported by Slaughter et al. and O’Sullivan and Ferrer. We conclude that while we have a mismatch with the reference data by Slaughter et al., our results still lie within the range of reported Strouhal peaks of the literature for this problem which confirms the resolution of our approach.



(a) Streamwise wall unit Δx^+ .



(b) Vertical wall unit Δz^+ .



(c) Spanwise wall unit Δy^+ .

Figure 3.19: Distribution of wall units Δx^+ , Δy^+ , Δz^+ for the time-averaged flow field on all wing profiles for the extruded IFW.

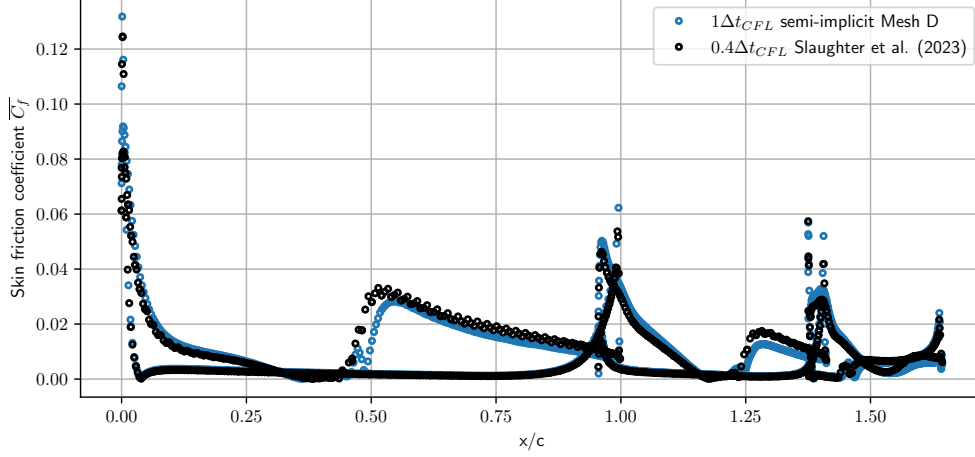


Figure 3.20: The skin friction coefficient C_f evaluated on all the surface of all three wing profiles. The main plane reaches up to $1x/c$, the first flap up to $1.3x/c$ and the second flap until $1.8x/c$.

Having established a mesh that sufficiently resolves the transition to turbulence, we next investigate the stability and accuracy of the linear-implicit scheme.

Stability limitations

The stability study follows the same procedure as for the 2D IFW described in section 3.5.3. For the eIFW, the largest stable time step size for $\overline{\text{CFL}} \approx 1.4$ is $\Delta t_{\text{CFL}} = 1 \times 10^{-5}$. Our tests use the linear-implicit scheme with increasing Δt and an initial condition based on the RANS solution for this mesh. A time step is assumed stable if it predicts the flow over a period of at least $20t^*$ after which the solution typically reached a quasi-steady state. After running several simulations in a trial-and-error fashion, we established stability limits for both schemes.

We again employ the CFL estimator described in section 2.3.4 for a local (per-element) CFL number. Figure 3.22a shows the variation of the CFL estimate for the entire domain. The spatial maximum on the eIFW geometry lies at the leading and trailing edges of all three elements. We find that the maximum among these six regions lies however at the trailing edge of the main plane. The turbulent flow reaches the highest velocity along the main plane and interacts with the trailing edge prism layer. Due to the fine boundary layer refinement this leads high CFL numbers and, similar to the 2D geometry, leads to the upper stability limit for this geometry.

Moreover, figure 3.22b illustrates the spatial maximum over a long-time integration along with the time-averaged CFL. We observe that the semi-implicit scheme is stable up to $1\Delta_{\text{CFL}}$ which leads to a time-averaged $\overline{\text{CFL}} = 1.4$. As for the 2D case, the simulation with an explicit

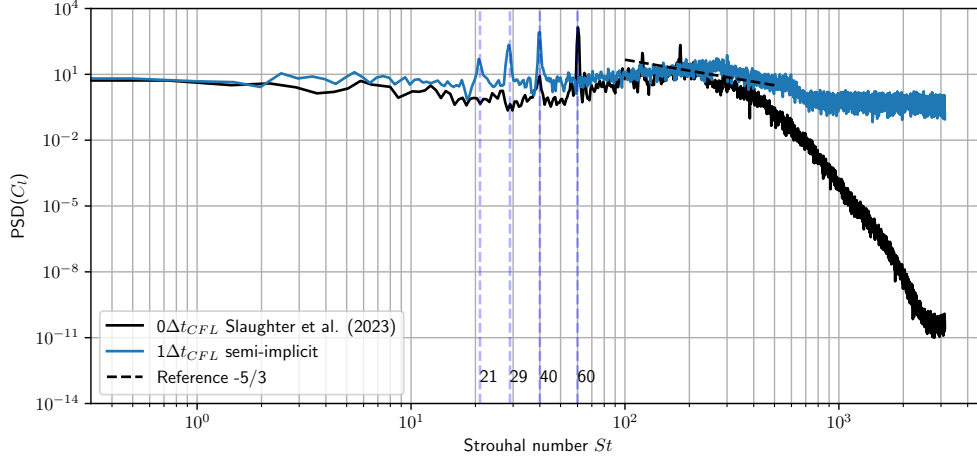


Figure 3.21: Power spectral density of the total lift coefficient using an interval of $10t^*$. Comparison of data from Slaughter et al. [32] and a reference simulation with semi-implicit scheme and $\Delta t = 1\Delta t_{CFL}$.

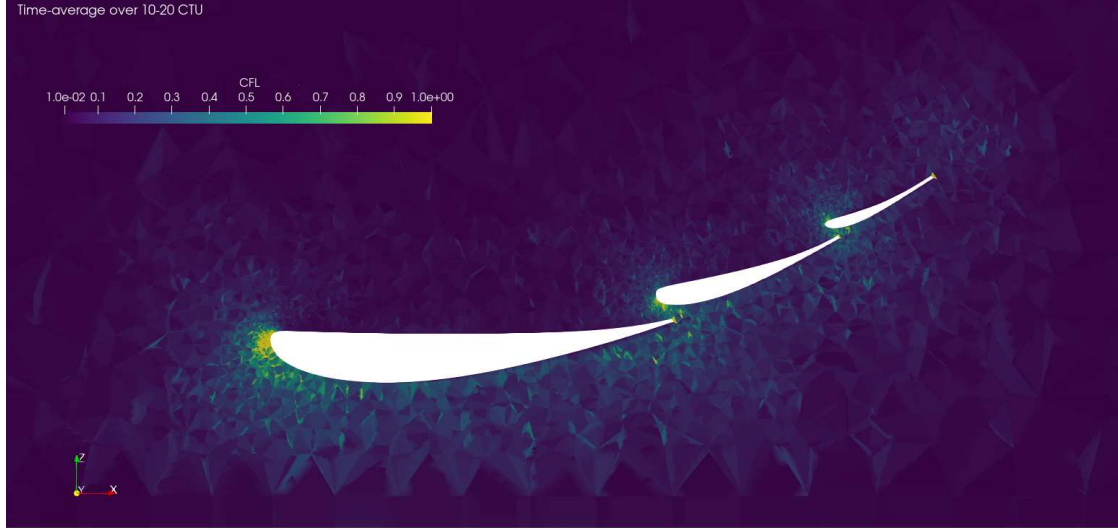
advection term is stable for a CFL slightly greater than 1. On the one hand, the curvature of the mesh biases the mapping from standard space velocity to the local velocity and includes inaccuracy into the CFL estimate. On the other hand, the spectral vanishing viscosity operator dampens instabilities in the flow which could allow a CFL number $CFL > 1$.

The linear-implicit scheme allows up to two orders of magnitude increase that is $100\Delta_{CFL}$. Further, the increments in time step size Δt show a similar increment in the CFL for $10\Delta_{CFL}$. The CFL-increment, however, becomes stronger for $50\Delta_{CFL}$ and $100\Delta_{CFL}$ with about a 16% stronger increase in time-averaged CFL compared to the Δt increase. Since the time step and spatial discretisation do not change, the stronger increase in CFL suggest higher velocity magnitudes.

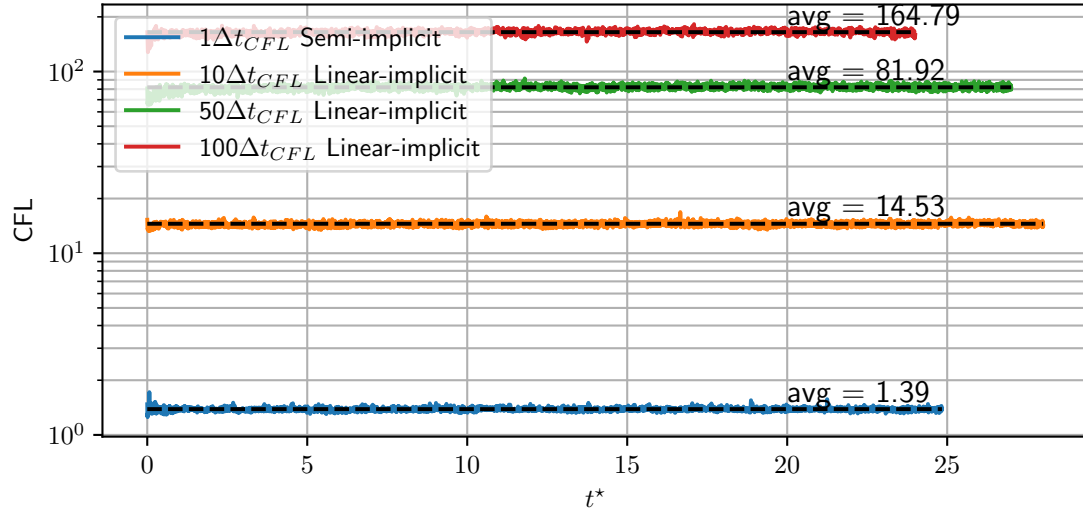
Accuracy with increasing time step size

This section discusses the influence of temporal accuracy on the prediction of the turbulent flow across the complex geometry. Similarly to the 2D IFW simulation, our discussion focuses on the main plane of the three-component wing, because the entire wing includes a plethora of phenomena appearing and would distract from the discussion of temporal accuracy. Nevertheless, we highlight severe changes on the first and second flap where relevant.

We advance the simulations with both schemes and all time step sizes until we reach the quasi-steady state. The quasi-steady state is defined visually investigating the raw signal and the moving average. We show this approach for the semi-implicit scheme with $\Delta t = \Delta t_{CFL}$ in



(a) Time-averaged CFL estimate for the mid-plane at $y = 0.2c$ using the linear-implicit scheme with a time step size of $10\Delta t_{CFL}$.



(b) Estimation of the CFL number for different time step sizes and the time-average for the eIFW.

Figure 3.22: Variation of CFL estimates in space and time for the eIFW geometry based on the time-averaged velocity field.

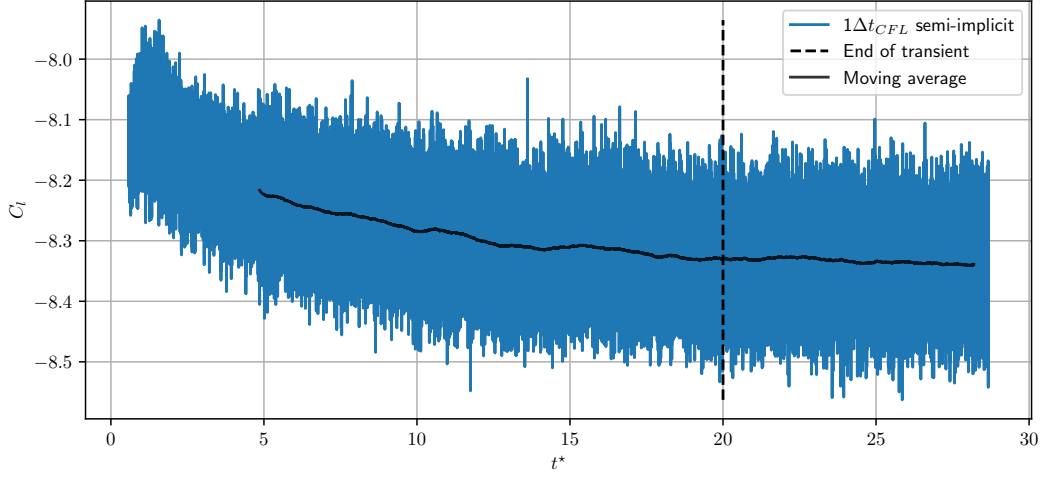
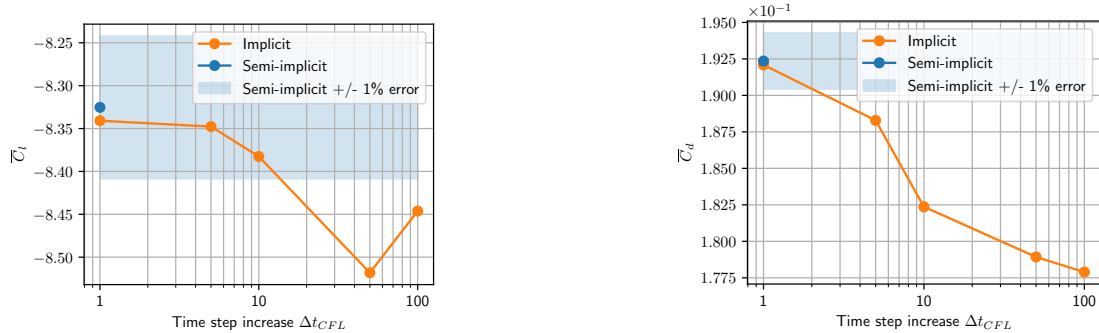


Figure 3.23: Time evolution of the lift coefficient for the extruded Imperial Front Wing geometry using the semi-implicit scheme and a time step size $\Delta t = \Delta t_{CFL}$.

figure 3.23. In this case, we assume the quasi-steady state from $t = 20t^*$. Consequently, we begin averaging at this point. The averaging duration is based on the reference studies [29, 32, 87] which use $8t^*$.



(a) Time-averaged lift coefficient $\overline{C}_l$ for the sum of all wings.

(b) Time-averaged drag coefficient $\overline{C}_d$ for the sum of all wings.

Figure 3.24: The time-averaged lift coefficient $\overline{C}_l$ and drag coefficient $\overline{C}_d$ for the 3D eIFW case with increasing time step size. We present the sum of the coefficients at the top along with the contribution from each wing. The averaging interval uses $10t^*$ and we show an error of 1% relative to the time-averaged semi-implicit coefficients.

Beginning with the integral force coefficients, we compare the time-averaged lift and drag coefficients in figures 3.24 and 3.25. Figure 3.24 shows the sum of forces over all wing elements whereas figure 3.25 presents contributions from each wing element. Additionally, we show 1%

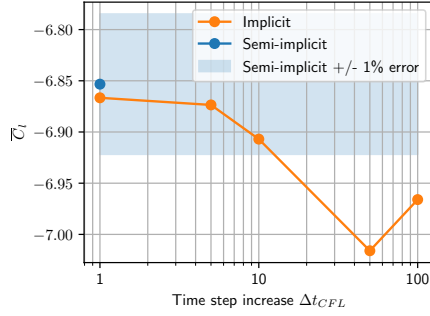
error intervals relative to the mean semi-implicit prediction which serves as a reference accuracy for engineering applications.

The sum of all wing forces in figure 3.24 indicates a decreasing trend for the lift coefficient with increasing time step size up to $\Delta t = 50\Delta t_{\text{CFL}}$. Interestingly the lift coefficient increases again for the largest time step size of $100\Delta t_{\text{CFL}}$. In contrast, the drag coefficient shows a different trend. The drag decreases with increasing time step size with a particular strong decrease from $5\Delta t_{\text{CFL}}$ to $10\Delta t_{\text{CFL}}$. Overall, we observe differences in the drag coefficient up to 7.7% measured against the semi-implicit reference value.

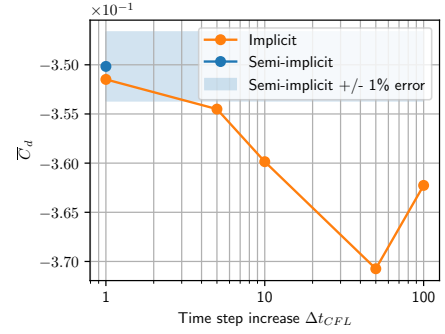
The individual contributions in figure 3.25 give more insight into the trends for the sum of the forces. The lift coefficient's trend is consistent for all wing elements. In particular, the main plane and first flap trends agree with the sum of the lift force. Moreover, it is encouraging that an order of magnitude increase in the time step size still has only a small influence on the lift coefficient. In contrast, the drag trend is more intricate as the main plane contributes with negative drag whereas both flaps increase the drag force. The main plane wing shows a similar decrease with increasing time step size as does the lift force. However, both flaps show a strong peak in the drag force. On the one hand, the first flap shows a strong deviation from the trend at $50\Delta t_{\text{CFL}}$ with a strong peak in the drag that is not present for $100\Delta t_{\text{CFL}}$. On the other hand, the second flap is consistent with a strong decrease in drag only for $100\Delta t_{\text{CFL}}$.

The force coefficients show a consistent trend among all wing profiles: consistent deviation from the semi-implicit prediction up to $50\Delta t_{\text{CFL}}$ and an inversion in that trend at $100\Delta t_{\text{CFL}}$. Next, we look at the pressure distribution and skin friction around the wing profiles to gain insight into the laminar-turbulent transition and the influence of the time step size. Note that we intentionally show only a few selected cases as to declutter the overall result.

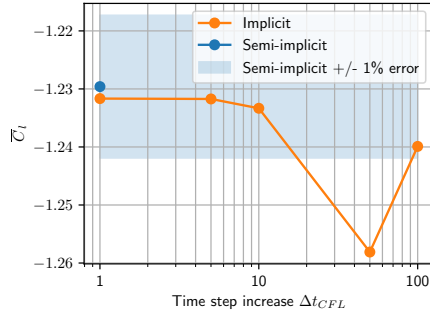
Figure 3.26a compares the time-averaged pressure coefficient $\overline{C_p}$ with an averaging window $[15, 20]t^*$. Clearly, the pressure side, where $\overline{C_p} > 0$, is equally predicted for all time step sizes and matches the reference data. In contrast, the suction side with $\overline{C_p} < 0$ shows distinct pressure distributions. The minimum at $x/c = 0.025$ with -10.746 in reference [32]. All time step sizes match this minimum with less than 2% variation. Downstream, the variations stay similar up to $x/c = 0.35$ where large differences in the pressure distribution form. Figure 3.26b shows this region closely. The semi-implicit prediction almost plateaus at $x/c = 0.45$ with a pressure coefficient of $\overline{C_p} = -7.25$ and shows a stark decrease in pressure magnitude at $x/c = 0.5$. This decrease hints at a strong deceleration of the flow and is likely the region of laminar-turbulent transition. The reference data does not plateau as significantly and shows a similar strong



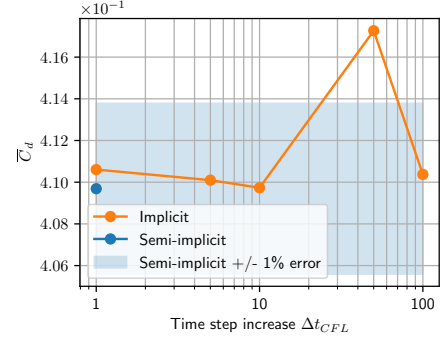
(a) Time-averaged lift coefficient $\overline{C}_l$ for the main plane wing.



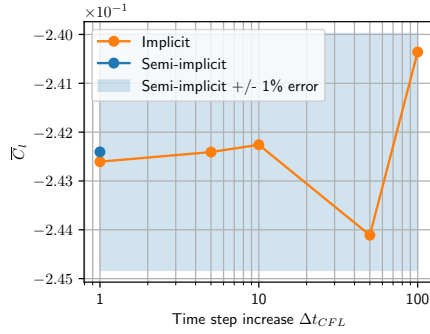
(b) Time-averaged drag coefficient $\overline{C}_d$ for the main plane wing.



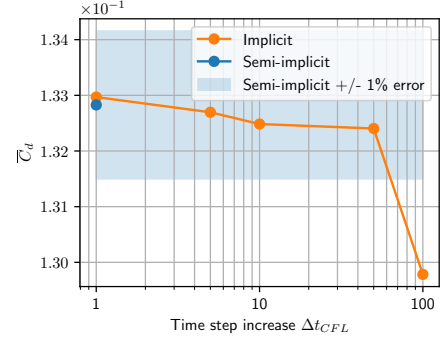
(c) Time-averaged lift coefficient $\overline{C}_l$ for the first flap.



(d) Time-averaged drag coefficient $\overline{C}_d$ for the first flap.



(e) Time-averaged lift coefficient $\overline{C}_l$ for the second flap.



(f) Time-averaged drag coefficient $\overline{C}_d$ for the second flap.

Figure 3.25: The time-averaged lift coefficient $\overline{C}_l$ and drag coefficient $\overline{C}_d$ with increasing time step size. We present the coefficients individually for the contributions of each wing element. The averaging interval uses $10t^*$ and we show an error of 1% relative to the time-averaged semi-implicit coefficients.

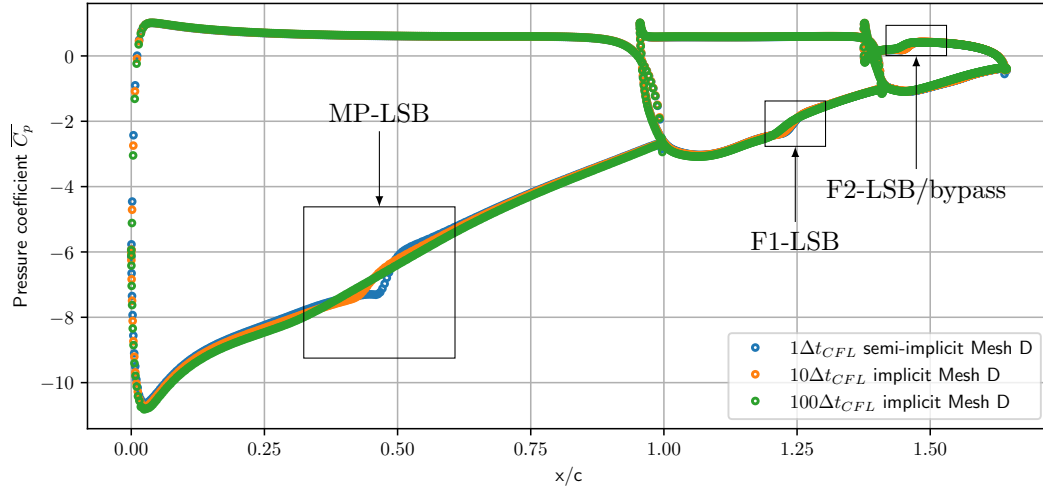
decrease in pressure magnitude upstream at $x/c = 0.44$. Moreover, the linear-implicit predictions for $10\Delta t_{\text{CFL}}$ and $50\Delta t_{\text{CFL}}$ show the pressure decrease at $x/c = 0.43$ and $x/c = 0.40$, respectively. The prediction with $100\Delta t_{\text{CFL}}$ does not show the sharp decrease in pressure magnitude. In conclusion, these cases indicate a shift in the transition location upstream with increasing time step size and no transition for the largest time step size.

Skin friction in figure 3.27a shows the main plane friction and includes all points along the span. Good agreement for all cases on the forward peak in the skin friction for $x/c < 0.05$ and for the pressure side. Distinct predictions on the suction side of the main plane. Upstream in the range $x/c = [0.05, 0.25]$, the predictions using a time step size $\Delta t < 100\Delta t_{\text{CFL}}$ consistently predict higher skin friction than the reference, and the case $100\Delta t_{\text{CFL}}$ gives smaller skin friction. Further downstream, the differences between all time step sizes become more nuanced.

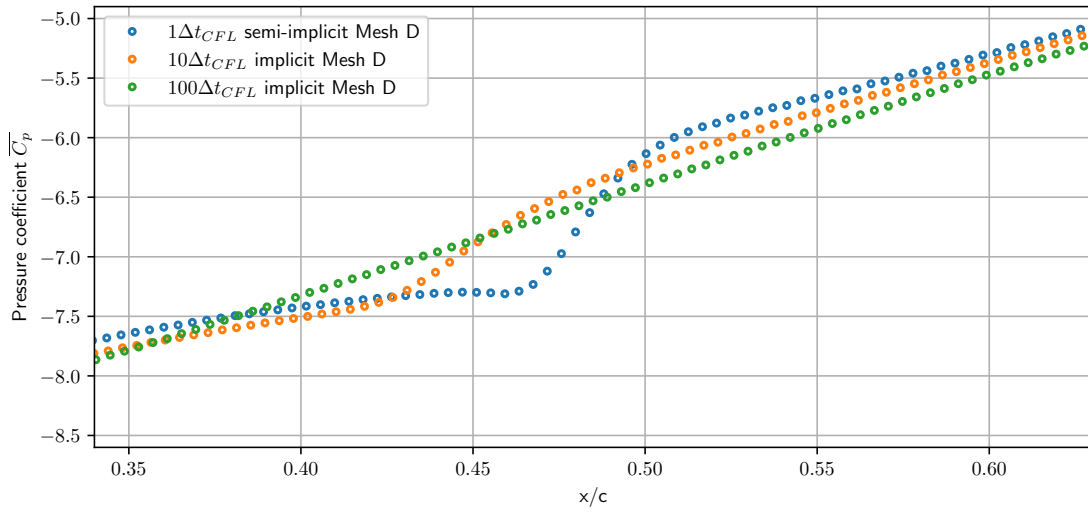
The semi-implicit prediction with $\Delta t = \Delta t_{\text{CFL}}$ predicts zero skin friction at $x/c = 0.46$ and shows a re-laminarisation, similar to the 2D IFW, with a second zero at $x/c = 0.49$ and a transition to turbulence downstream indicated by a sharp increase in skin friction. It is possible that the re-laminarisation leads to the plateau in the pressure coefficient $\overline{C_p}$, because the flow stays in a state of low skin friction where it is neither accelerated or decelerated significantly. This coincides with the same location as the pressure plateau in $x/c = [0.45, 0.49]$. Moving downstream, the semi-implicit prediction shows a transition to turbulence at $x/c = 0.5$ with a sharp increase in skin friction to $\overline{C_f} = 0.04$. The trend of this transition follows the reference data, however, the transition location is upstream at $x/c = 0.43$ for the reference and the skin friction peak is $\overline{C_f} = 0.033$. Note that this follows the observations for the pressure coefficient above.

Analogous to the semi-implicit prediction, the linear-implicit scheme with $10\Delta t_{\text{CFL}}$ and $50\Delta t_{\text{CFL}}$ follow the reference data set more closely. The prediction at $10\Delta t_{\text{CFL}}$ shows transition at $x/c = 0.43$ and the strong increase in time-averaged skin friction peaking at $\overline{C_f} = 0.038$. A larger time step size of $50\Delta t_{\text{CFL}}$ moves the transition location further upstream to $x/c = 0.4$ and reduces the peak to $\overline{C_f} = 0.028$. The trend strongly changes for the largest time step size $\Delta t = 100\Delta t_{\text{CFL}}$. We do not observe zero skin friction along the entire suction side of the main plane which shows that the flow does not transition to turbulence. Instead the laminar flow does not have a sharp increase in the skin friction, but a shallow rise in the range $x/c = [0.3, 0.45]$ where the skin friction reaches a peak of $\overline{C_f} = 0.02$.

In summary, we observe that the smallest time step size with semi-implicit scheme shows a re-laminarisation which influences the mean skin friction and pressure distribution. The re-

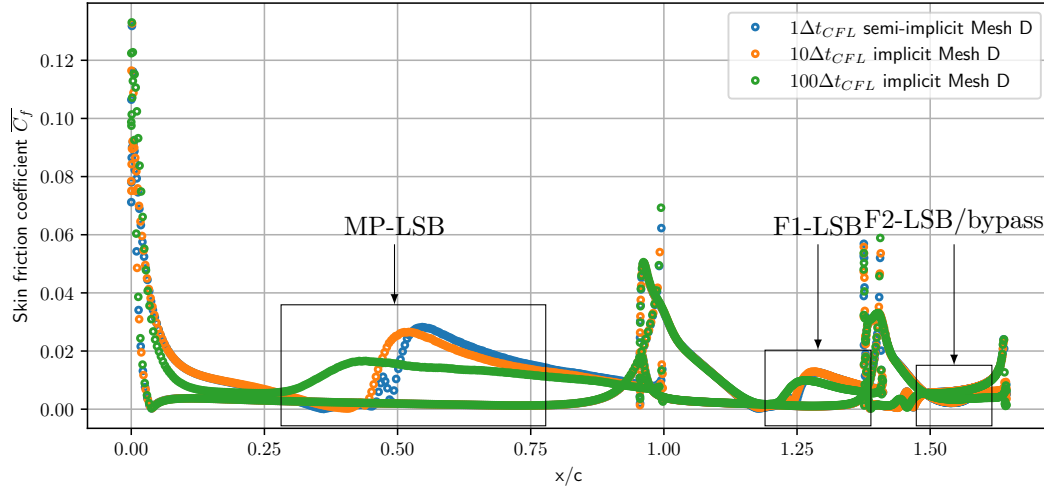


(a) Time-averaged pressure coefficient $\overline{C_p}$.

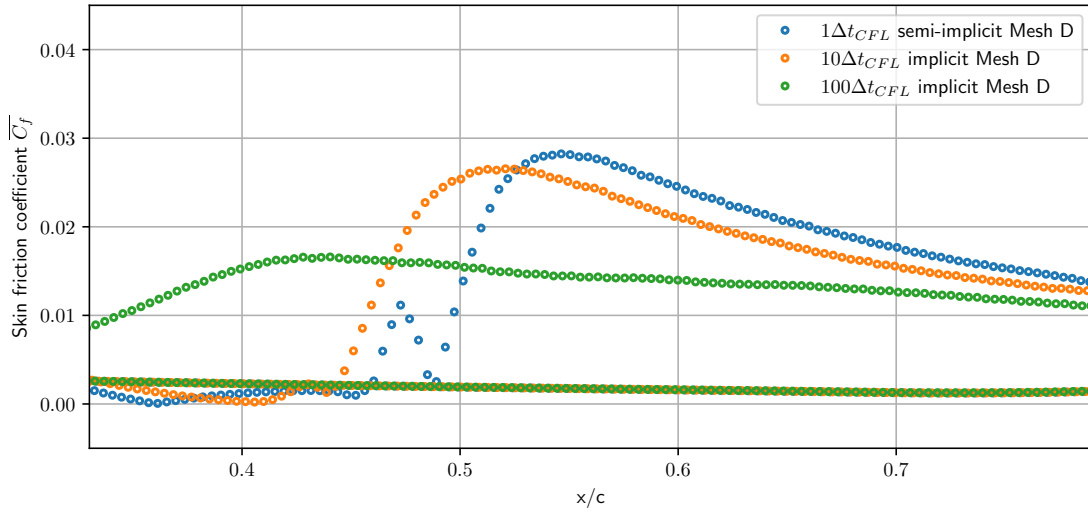


(b) Zoom of the pressure distribution around the transition location on the main plane wing.

Figure 3.26: Time-averaged pressure coefficient $\overline{C_p}$ for all wing elements comparing the influence of larger time step sizes. Locations with transition to turbulence are highlighted with black boxes for the .



(a) Time-averaged skin friction coefficient $\overline{C_f}$.



(b) Zoom around the transition location on the main plane wing.

Figure 3.27: Time-averaged skin friction coefficient $\overline{C_f}$ for all wing elements comparing the influence of larger time step sizes. Locations with transition to turbulence are highlighted with black boxes.

laminarisation is not a physical phenomena, but rather a numerical artifact. The kink in the main plane skin friction is likely an effect of insufficient averaging of the laminar separation bubble (LSB) as it is not present in the reference data, see figure 3.20.

The linear-implicit prediction with $10\Delta t_{\text{CFL}}$ closely matches the reference semi-implicit prediction showing only a slight mismatch in the mean transition location. The change in transition location leads to a decrease in the drag coefficient due to higher friction and increase in the lift coefficient as the pressure distribution indicates stronger suction on the wings.

The largest time step size of $100\Delta t_{\text{CFL}}$ does show a different laminar-turbulent transition process possibly without a laminar separation bubble (LSB). The result is a decrease in the drag and increase in the lift as we observed earlier in figures 3.24 and 3.25. It appears that there is an accuracy threshold when the numerical diffusion of the scheme is sufficient to affect the major physical phenomena of the flow.

The PSD in figure 3.28 compares the influence of time step size and numerical scheme on the dominant scales of the simulation. The initial comparison in figure 3.21 shows that the semi-implicit scheme with $1\Delta t_{\text{CFL}}$ resolves three significant Strouhal numbers at $St = \{20, 30, 40\}$. In comparison, the linear-implicit scheme with the same time step size resolves the same three frequency peaks. However, we observe a strong decrease in the PSD for Strouhal $St > 40$ which is not present in the semi-implicit PSD. This effect appears similar to the findings in two dimensions discussed above for figure 3.17 which show that the formulation of the pressure Poisson problem leads to different energy distributions in the frequency space. Apparently, this effect of the pressure form extends to complex, three-dimensional and turbulent flows.

The time step influence becomes most obvious for the cut-off Strouhal number, following the Nyquist criteria. This cut-off is at $St_{\text{nyq.}} = 900$ for $10\Delta t_{\text{CFL}}$ which lies near the edge of the resolved frequencies for the semi-implicit prediction around $St = 1000$. We further observe a shift in the frequency peaks at $10\Delta t_{\text{CFL}}$. The most energetic frequencies appear at $St = \{15, 20, 30\}$. On the one hand, this suggests that the shift in cut-off frequency leads to an accumulation of energy in the lower frequency range $St < St_{\text{nyq.}}$ and lead to a redistribution of energy into the lower Strouhal numbers. On the other hand, it appears that the effect of the pressure form discussed in the previous paragraph causes a decrease in the PSD at around $St = 30$ and removes the $St = 40$ peak observed at $1\Delta t_{\text{CFL}}$.

A further increase in time step size to $100\Delta t_{\text{CFL}}$ causes a cut-off at $St_{\text{nyq.}} = 125$. Consequently, the cut-off lies well-within the range of resolved frequencies of the turbulent flow and close to the peak frequencies. We observe a different set of peak frequencies at $St = \{20, 25, 80, 90\}$.

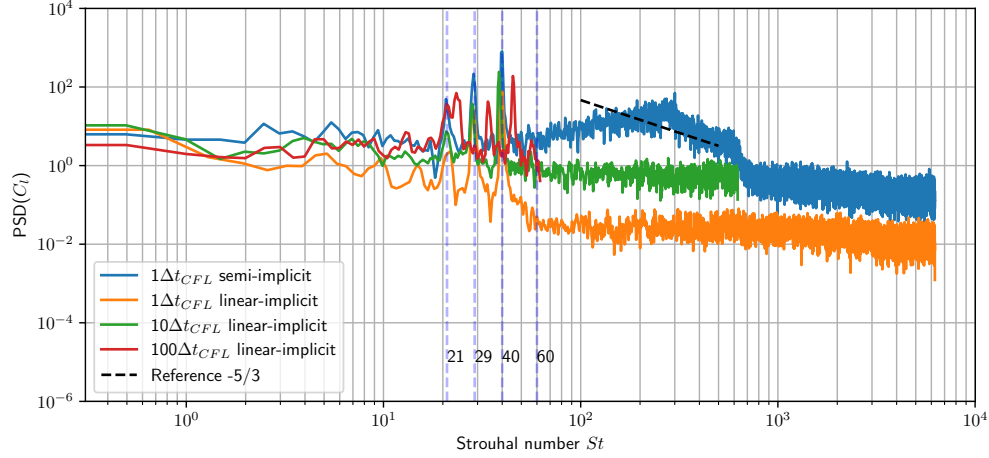


Figure 3.28: Power spectral density of the total lift coefficient using an interval of $10t^*$. Comparison of time step size influence $\Delta t = 1\Delta t_{CFL}$.

This change in the spectrum correlates with the change in flow structure that the skin friction coefficient reveals in figure 3.27a. The severe change in time step redistributes the energy and results in an overall increase in the PSD within the range $St \in [10, 105]$. Interestingly, we nevertheless find a pressure form effect with strong dissipation around $St = 30$ similar to the PSD for $10\Delta t_{CFL}$.

In summary, the linear-implicit and semi-implicit scheme provide equivalent integral force, skin friction and pressure coefficients for the same time step size. However, there is a change in the dissipation behaviour at higher frequencies that we observe in the PSDs of this test case and the 2D IFW that is discussed above, see figure 3.17. The dissipation effect further influences the comparison at increasing time step size.

We find a small effect in accuracy for a 10-fold increase in time step size. It leads to a change in integral force coefficients by less than 3% driven by a shift in transition location of less than 5% on the main plane. Interestingly, we observe a stronger change in the power spectral density of the lift signal where the peak frequencies shifted to the lower frequency range. Further, we find that a 100-fold increase in time step size has a profound effect on the turbulent boundary layer flow changing the laminar-turbulent transition characteristic and peak frequencies in the PSD. It requires further investigation to identify which physical phenomena are insufficiently resolved at this large time step.

3.6 Numerical characteristics of the 2.5D formulation

We introduce an extension of the velocity correction scheme and the spectral/hp element method for a 2.5D, or quasi-3D, formulation. The 2.5D formulation employs a Fourier expansion in the third dimension which, due to linearity, allows decoupling of the Fourier modes. We exploit the decoupled modes for a hybrid parallelisation where message passing interface (MPI) ranks decompose the two dimensional mesh and the Fourier expansion, see [73]. Note that we further decompose the Fourier modes into half modes which make up either the sine or cosine contribution. We detail the full formulation in section 2.5. We compare the stability and accuracy for the semi-implicit and linear-implicit algorithms using the formulation and results in reference [68].

We investigate the stability of the scheme on a cylinder test case with a Reynolds number of $Re = 500$ large. At this Reynolds number, an unstable Floquet mode excites three-dimensional flow [83]. The two dimensional mesh, polynomial order $P = 6$ and boundary conditions are equivalent to the case described in section 3.4.1. The difference for a quasi-3D formulation is the spanwise direction. We use a domain length of $\ell_z = 1D$ where D is the unit diameter of the cylinder. The Fourier expansion uses 8 full modes to resolve the dynamics in the z -direction and employs periodic boundary conditions implicitly.

Using the quasi 3D cylinder, we verify that the algorithm gives stable predictions with increasing time step size Δt . Based on the maximum stable time step size with the semi-implicit scheme $\Delta t_{\text{CFL}} = 3 \times 10^{-3}$, we measure the stability improvement as the increase in time step size with the linear-implicit scheme.

The stability tests restart from an initial solution that transitioned to periodic vortex shedding after $200t^*$. Restarting from this solution, we advance the simulation for another $1000t^*$. In order to measure accuracy and stability of the predictions, we measure the lift and drag forces F_y and F_x , respectively, and compare results based on the time-averaged drag coefficient and the RMS of the lift coefficient in equation (3.8).

Table 3.5 lists drag and lift predictions for the semi-implicit and linear-implicit schemes. The semi-implicit scheme gives stable predictions up to a maximum time step size of $\Delta t_{\text{CFL}} = 0.003$. In contrast, the linear-implicit scheme provides stable predictions up to $10\Delta t_{\text{CFL}}$ which is an order of magnitude increase in the stability. For $\Delta t > 10\Delta t_{\text{CFL}}$, the linear-implicit scheme accumulates energy on the high-frequency modes and becomes unstable after few time steps.

Besides the stability improvement, table 3.5 gives a comparison of the prediction accuracy for drag and lift coefficients. The lift coefficient seems to be well predicted with both schemes for

Table 3.5: Prediction of the mean drag coefficient $\overline{C}_d$ and RMS of the lift coefficient $\overline{C}_{l,\text{RMS}}$ at increasing time step size. Comparison of the semi-implicit and linear-implicit scheme.

Δt	Semi-implicit		Linear-implicit	
	$\overline{C}_d$	$\overline{C}_{l,\text{RMS}}$	$\overline{C}_d$	$\overline{C}_{l,\text{RMS}}$
0.003	0.9927	0.2835	1.0022	0.3051
0.006	(unstable)	(unstable)	1.0217	0.3261
0.012	(unstable)	(unstable)	1.0635	0.3626
0.024	(unstable)	(unstable)	1.1268	0.4266
0.03	(unstable)	(unstable)	1.1667	0.5103
0.06	(unstable)	(unstable)	(unstable)	(unstable)
0.12	(unstable)	(unstable)	(unstable)	(unstable)

equal time step sizes Δt_{CFL} . The difference is of the same order as the temporal accuracy. For larger time step sizes, we observe an increase in the RMS of the lift coefficient with increasing magnitude compared to the lift coefficient at Δt_{CFL} . In particular for $\Delta t > 4\Delta t_{\text{CFL}}$, the difference in $\overline{C}_{l,\text{RMS}}$ becomes an order of magnitude larger than the time discretisation error would suggest for a first-order scheme. This is possibly due to insufficient spatial discretisation which causes an interaction of spatial and temporal error terms.

Figure 3.29 shows the time-averaged distribution of kinetic energy across the Fourier modes with wavenumber k . It shows an energy gap in terms of the ratio of modal energy E_k between the mean mode $k = 0$ and any higher frequency modes of $\frac{E_{k=0}}{E_{k=1}} \approx 7.2 \times 10^2$. This is in good agreement with our assumption of having the majority of the energy in the mean mode ($k = 0$) and motivated by the dataset presented in figure 2.6. It is encouraging to observe a significant gap in the energy between mean and first mode, with the stability improvement described above, even for this small Reynolds number of 500. Therefore, this scheme is potentially useful in stabilising simulations outside of the range of advection-dominated flows.

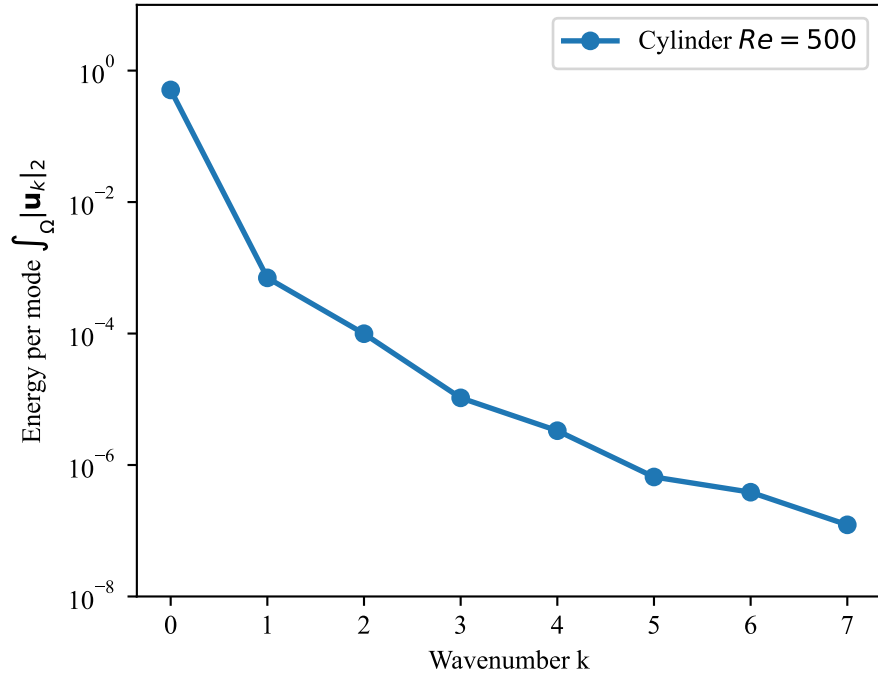


Figure 3.29: The time-averaged energy for the wave numbers k highlighting the large gap between mean mode $k = 0$ and higher frequency $k > 0$ modes for the cylinder flow with $Re = 500$.

Chapter 4

Performance

This chapter begins with a description of the setup used for analysing the performance of the scheme. It discusses the computational resources available and the software packages used for compilation. Further, it introduces the main test case for this study and provides insight into the possible speed-up, algorithmic bottlenecks and improvements to the algorithm for the linear-implicit scheme.

4.1 Benchmark setup

Our analysis aims for an estimated speed-up of the linear-implicit scheme over the semi-implicit scheme. The speed-up becomes possible through a larger stability margin of the implicit scheme which allows larger time step sizes and, hence, fewer steps-to-solution N^{sol} . Under the assumption, that the linear-implicit scheme's time-to-solution will always be less or equal than the semi-implicit's, the steps-to-solution provides an upper-bound for the maximum speed-up S^{max} as

$$S^{\text{max}} = \frac{N_{\text{semi-implicit}}^{\text{sol}}}{N_{\text{linear-implicit}}^{\text{sol}}} = \frac{\Delta t_{\text{linear-implicit}}}{\Delta t_{\text{semi-implicit}}}, \quad (4.1)$$

where $N_{\Delta t} = t^*/\Delta t$ is the number of steps required for a given time interval t^* . Note that t^* is a case-specific convective time unit as defined in chapter 3 for the different cases.

The practical speed-up S , however, becomes more intricate as it depends on the differences in the linear-implicit and semi-implicit algorithm. This includes the evaluation of explicit forcing, evaluating matrix operators, and cost-per-iteration for the GMRES and conjugate gradient algorithms multiplied by the overall number of iterations required to solve the linear systems.

These metrics are problem-dependent and unknown a priori.

Instead, we measure the time-to-solution T^{sol} for the semi-implicit and linear-implicit scheme, and determine the practical speed-up S as their ratio with

$$S = \frac{T_{\text{semi-implicit}}^{\text{sol}}}{T_{\text{linear-implicit}}^{\text{sol}}}. \quad (4.2)$$

Further, we estimate T^{sol} using the averaged wall-time per time step $\overline{T_{\Delta t}}$ and scale it to a fixed time interval with physical significance for the problem, for example one t^* . The definition follows as

$$T^{\text{sol}} = \overline{T_{\Delta t}} N_{\Delta t}, \quad (4.3)$$

We employ the C++ standard library's `std::chrono::steady_clock` to estimate the wall-time for a time step $T_{\Delta t}$.

The test further includes timers to identify algorithmic bottlenecks for both schemes. This combined analysis gives an overview of the linear-implicit scheme's computational performance and possible improvements.

4.1.1 Benchmark test case

We focus the performance analysis on a challenging rather than a canonical problem. This enables identifying performance characteristics for ill-conditioned matrices on large-scale meshes and include communication issues when using hundreds of CPUs in parallel.

The benchmark problem is the 3D extruded Imperial Front Wing (eIFW) introduced in section 3.5.1. The mesh represents a complex, three-dimensional geometry with a high Reynolds number flow which includes typical features such as ill-conditioning of matrices as well as a large number of elements requiring efficient parallel communication between processors. As such, it shares features of typical industrial flow problems, see for example references [32–34], and provides an interesting benchmark for the linear-implicit algorithm on a challenging geometry. The next section will discuss the hardware and software configuration.

4.1.2 Hardware and software considerations

The performance tests use the Imperial College London research computing service [90] infrastructure. We use the HX1 High Performance Computing (HPC) cluster which uses two Intel Xeon Platinum 8358 (Ice Lake) processors per node or a total of 64 physical cores per node. All

presented results use fully-occupied nodes on the HX1 cluster. The nodes are interconnected with NVidia ConnectX-6 HDR200 InfiniBand technology allowing low latency communication for large-scale parallel programs.

The linear-implicit and semi-implicit schemes are implemented in the open-source Nektar++ framework using version 5.5.0, see section 2.6 for details. We compile the code with Intel compiler version 2021.4.0 using Intel’s message passing interface (MPI) referred to with the acronym `iimpi` and the Intel Math Kernel Library (MKL). For parallel in-out operations, Nektar++ employs version 5 of the hierarchical data format (HDF) providing an efficient configuration for parallel execution. Further, we optimise the throughput of operations such as the advection operators with advanced vector extensions (AVX512).

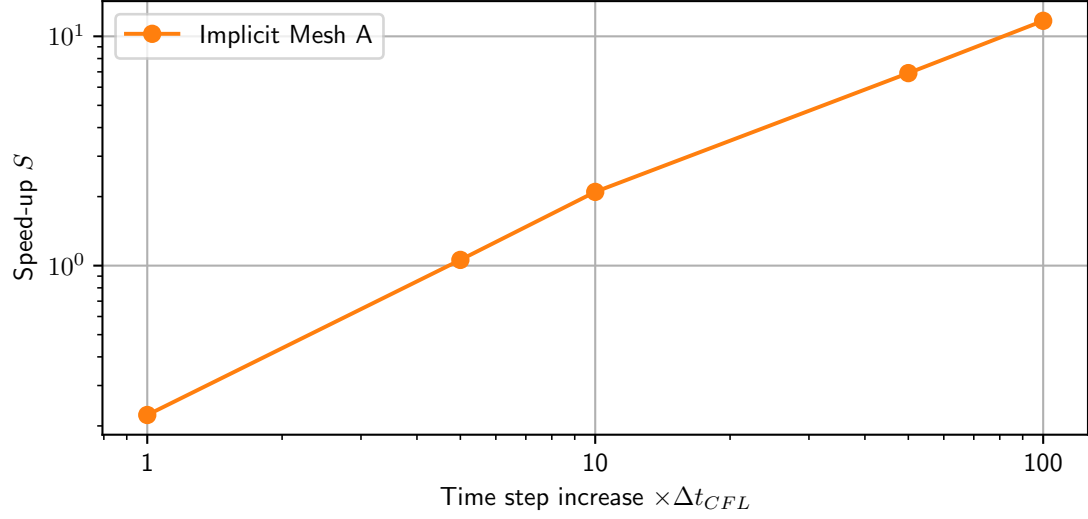
4.2 Performance analysis

4.2.1 Speeding-up simulations with implicit time-stepping

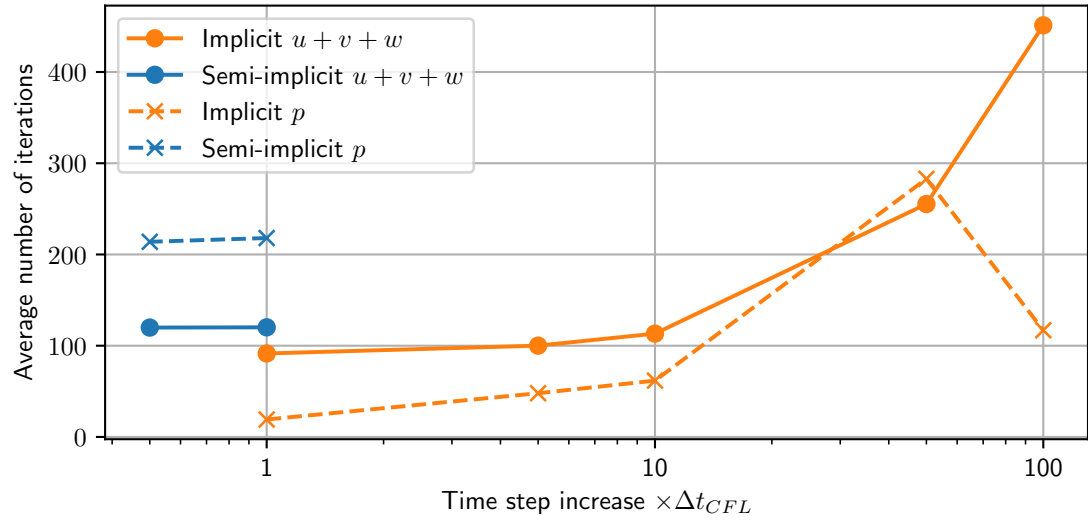
We investigate the performance of both schemes at increasing time-step sizes measuring the time-to-solution T^{sol} . The discussion in section 3.5.4 revealed a stability margin of theoretical speed-up $S^{\text{max}} = 100$ using the linear-implicit scheme on the benchmark eIFW case. This allows time steps up to $\Delta t = 10^{-3}$ and consequently 250 steps per t^* .

In order to measure the practical speed-up reliably, we aim for sufficient time steps $N_{\Delta t}$ for estimating the average wall-time per step, and allow a reasonable total wall-time for the given computational resources. We chose either one t^* or a minimum number of 1000 steps as a heuristic size that allows sufficient collection of statistics. Since 1000 steps might cover a different physical time interval of the flow, we initialise the timing tests from a statistically-steady flow solution. The flow on eIFW reaches a statistically-steady state after a time of $10t^*$. We used the semi-implicit scheme with a time step $\Delta t = 10^{-5}$ to initialise this solution. This setup allows a fair comparison of both schemes without the computational overhead of the initial transient. Next we analyse the practical speed-up possible when employing the linear-implicit scheme. We again use the notation of relative time step size increase where the reference $\Delta t_{\text{CFL}} = 10^{-5}$ corresponds to the stability limit of the semi-implicit scheme with $\text{CFL} \approx 1$.

Figure 4.1a shows the speed-up S relative to the semi-implicit prediction with time step size Δt_{CFL} . The trend shows a few interesting observations. First of all, the linear-implicit scheme is slower for the reference time step size $1\Delta t_{\text{CFL}}$. Further, the speed-up increases almost linearly up



(a) Speed-up measured as the time-to-solution ratio of semi-implicit scheme against the linear-implicit scheme.



(b) Iteration count for the pressure problem and the sum of all three velocity problems plotted against time step size Δt .

Figure 4.1: Speed-up and number of iterations comparing the semi-implicit scheme with time step size $\Delta t(\overline{CFL} \approx 1)$ against the linear-implicit scheme with increasing time step size.

to about $10\Delta t_{\text{CFL}}$, and begins flattening in the range 10 to $100\Delta t_{\text{CFL}}$, that is up to the stability limit. At the reference Δt_{CFL} , the linear-implicit scheme is slower than the semi-implicit scheme by a factor of $S \approx 0.22$. Increasing the time step size up to a factor of 10 allows a speed-up $S \approx 2.1$ and further increasing to the limit allows a maximum speed-up of $S \approx 11.6$. Note that the break-even point between both algorithms lies at about a 5-fold increase in time step size. In summary, the speed-up is almost linear in the range 1 to $10\Delta t_{\text{CFL}}$ and slightly sub-linear for 10 to $100\Delta t_{\text{CFL}}$.

The next section provides insight into the practical speed-up and discusses bottlenecks of the linear-implicit algorithm.

4.2.2 Algorithmic insight and bottlenecks

The linear-implicit and semi-implicit algorithm are broadly split into a pressure and three velocity problems, see also figure 2.1. Each problem further breaks down into three major steps: setting-up the linear system $\mathbf{A}$, setting up the preconditioner $\mathbf{M}^{-1}$, and solving the preconditioned system $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$ using either conjugate gradients or GMRES. We analyse which of these components consumes the most time, how that changes with increasing the time step size Δt and identify bottlenecks in the linear-implicit algorithm. In particular, we investigate numerical characteristics such as the iterations required by the linear system solvers and compare those with timings and the above discussion on practical speed-ups.

The velocity iteration counts in figure 4.1b shows two interesting trends that align with the practical speed-up. The linear increase in speed-up S from 1 to $10\Delta t_{\text{CFL}}$ leads to an increase in the average iteration count of 7 GMRES iterations per velocity problem. For the same time step increment, the pressure iterations almost double with a factor 1.76 gain. It is interesting to observe that the speed-up is almost linear for this time step increment while we see an increase in the work per time step. Particularly, the increase in pressure iterations appears to have no strong effect on the time-to-solution. In contrast, the iteration counts in the range 10 to $100\Delta t_{\text{CFL}}$ show a 4-fold climb in GMRES iterations for the velocity problems. This upwards trend in the velocity iterations hints at the ADR problem becoming stiffer for large time step sizes. While GMRES iterations counts are not an exact measure of the condition number, one can estimate the matrix conditioning from iteration counts as shown by Saad [66]. It appears that large time steps exponentially increase the stiffness of the velocity problem close to the stability limit and, thus, reduce the possible speed-up for time steps $\Delta t > 10\Delta t_{\text{CFL}}$.

Surprisingly, the pressure iterations in figure 4.1b show a stark increase for a 50-fold increment

in time step size by almost 5-times the pressure iterations. However, the pressure iterations reduce again for a $100\Delta t_{\text{CFL}}$. This result is unexpected and is possibly linked to a change in the physical phenomena as we pointed out in the accuracy discussion in section 3.5.4.

The iteration counts give a first view into the upper limit of the speed-up using the linear-implicit scheme. However, it does not explain the lower performance of the linear-implicit scheme at a time step size $1\Delta t_{\text{CFL}}$ where the iteration counts for the semi-implicit's Helmholtz and the linear-implicit's ADR velocity problems are almost equivalent. Next, we further look into timing individual parts of the linear-implicit algorithm and compares them against the semi-implicit performance to identify bottlenecks of the algorithm.

We compare timings of the semi-implicit and linear-implicit scheme in figure 4.2. The heat map uses colour based on the timings relative to the respective total execution time and displays the **absolute** timing for 100 time steps in seconds above. The left-hand side of figure 4.2 shows that both schemes spent the majority of their time in solving the pressure and velocity problems. Explicit forcing terms and boundary conditions are a minority of the time-to-solution and require the same absolute time in all cases. This is expected since explicit terms solely depend on the efficiency of operator evaluations. Instead we focus our attention on comparing the pressure and velocity update timings for both schemes at the same time step size $1\Delta t_{\text{CFL}}$.

Considering the pressure solve, the semi-implicit scheme uses 14.4 seconds per 100 time steps whereas the linear-implicit scheme requires 8.5 seconds. Interestingly, the ratio between both times of $\frac{14.4}{8.5} \approx 1.7$ is different from the ratio of average pressure iterations, see figure 4.1b, for both schemes $\frac{224}{35} \approx 6.4$. Since the algorithm here is equivalent for both schemes, one would expect a similar execution time. One possible explanation for this is the high standard deviation in the pressure iteration count which is 293 and 99 for the semi-implicit and linear-implicit scheme, respectively. Hence, the variation in both averages is 2 to 3 times larger than the average themselves showing that the average iteration count is insufficiently representing the work done for the pressure solve. It is interesting to note that these large variations are likely due to the diagonal preconditioner employed for the pressure system. We chose the preconditioner here for generality, however, it is well-known that the elliptic Poisson problem is ideally preconditioned with a multigrid-type approach, see [91–93].

The major part of the time-to-solution in both algorithms is required solving for the three velocity components u , v and w . The linear-implicit algorithm spends almost 93% of the execution time building the ADR matrix, the preconditioner and solving the system. In contrast, the semi-implicit algorithm uses 59% of the time-to-solution for solving the three Helmholtz

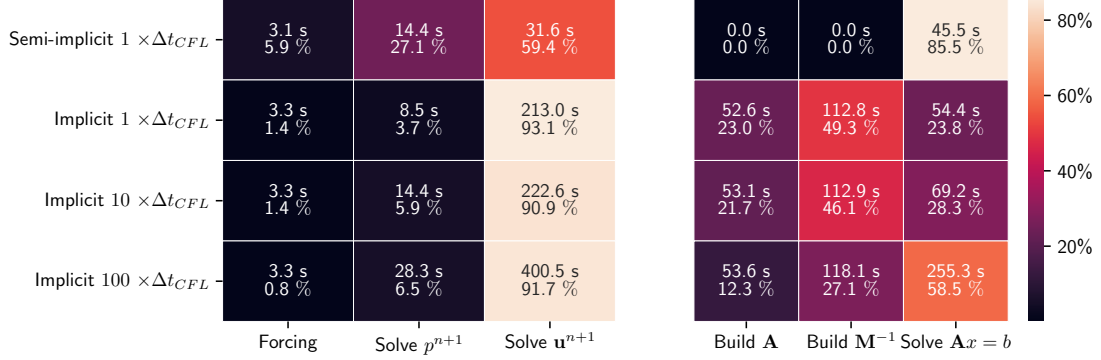


Figure 4.2: Heat map of execution time spent in the three sections of the velocity-correction scheme algorithm: explicit forcing, pressure solve and velocity solve. Additionally, we show timings of building and solving the linear systems. We show absolute time spent in seconds per 100 time steps as well as in percentage of total execution time. Comparing semi-implicit and linear-implicit scheme at increasing time step size.

matrices. The difference becomes clear if we compare the absolute time for 100 steps showing a factor of $\frac{213}{31.9} \approx 6.7$ difference for updating the velocity. The right-hand side of figure 4.2 gives further insight into the difference between both algorithms: building matrix $\mathbf{A}$ and the associated preconditioner $\mathbf{M}^{-1}$. The semi-implicit algorithm uses more than 85% of the time to solve the Helmholtz and Poisson problems. The linear-implicit scheme, however, uses only 24% of the execution time for solving the system and an additional 23% and 49% to re-build the velocity matrices and preconditioner, respectively.

In conclusion, the above analysis shows that the linear-implicit scheme is slower at the same time step size $1\Delta t_{CFL}$ due to re-building the velocity matrices $\mathbf{A}$ and the preconditioner $\mathbf{M}^{-1}$. We consider re-building either of the matrix and preconditioner only every n th step in the next section 4.2.3. In comparison to re-building, solving the linear systems is a minor cost-overhead compared to the semi-implicit scheme. Considering also the high iteration counts for pressure in figure 4.1b, we conclude a reduction in iterations through a more effective preconditioner would benefit both schemes. Next, we analysis the behaviour of the linear-implicit scheme at increasing time step size Δt .

The strongest trend with increasing time step size is more time spent solving the linear systems for velocity and pressure. We observe a shallow increase for $10\Delta t_{CFL}$ of 27% in time per 100 steps which coincides with the average increase in velocity iterations of 24%, see figure 4.1b. Similarly, the stronger increase from 10 to $100\Delta t_{CFL}$ shows a 3.98-fold increase in the average iteration count and a similar 3.68 times higher execution time for 100 steps. Considering

building the matrix $\mathbf{A}$ and the preconditioner $\mathbf{M}^{-1}$, the linear-implicit scheme shows essentially the same time spent independent of the time step size Δt with, however, a slightly upwards trend towards the larger time step size. The similarity in the build timings is due to the same amount of work required for evaluating mass, diffusion and advection operators for constructing the elemental matrices. Further, the slight increase is likely due to the matrix inversion required for the elemental Schur complement matrices $\mathbf{S}^e$.

All in all, the analysis shows that the linear-implicit scheme has a computational overhead of factor 5 compared to the semi-implicit scheme. The major contribution to this overhead is re-building the linear system and preconditioner. Fortunately, these contributions are almost independent of the time step size Δt despite forming the Schur complement matrix. We attempt reducing the re-building costs in the next section 4.2.3. Other contributions are the GMRES solver for inverting the ADR matrix and the ill-conditioned pressure problem. The GMRES solver is more expensive than the conjugate gradients method due to the orthogonalisation using a modified Gram-Schmidt process which requires additional communication for each iteration. Further, the diagonal preconditioner for the pressure system cannot bind the iteration count effectively and a multigrid-type preconditioner is desirable.

4.2.3 Algorithmic improvements

The discussion above in figure 4.2 revealed that building the preconditioner and linear system consumes up to 75% of the linear-implicit scheme's execution time. This motivates our investigation into freezing the preconditioner $\mathbf{M}^{-1}$ and the linear system $\mathbf{A}$ as described in section 2.4.4. Freezing means we are re-building either the preconditioner or the linear system only every n th time step instead of re-building at every time step. Consequently, this potentially reduces the cost by a factor of $1/n$ and potentially leads to an even larger speed-up.

Preconditioner freezing has the advantage of solely influencing the convergence of the iterative GMRES solver, that is the iteration counts. Hence, we expect an ideal operating point where the reduced work in re-building the preconditioner balances with an increase in iteration counts. In comparison, freezing the advection operator in the linear system requires adding the advection deficit, see the description in section 2.4.4, which additionally influences the error term of the linear-implicit scheme. Nevertheless, the advection freezing does make updating the preconditioner obsolete as well and, hence, has the potential to reduce up to 75% of the overall execution time for the linear-implicit scheme.

Freezing the preconditioner

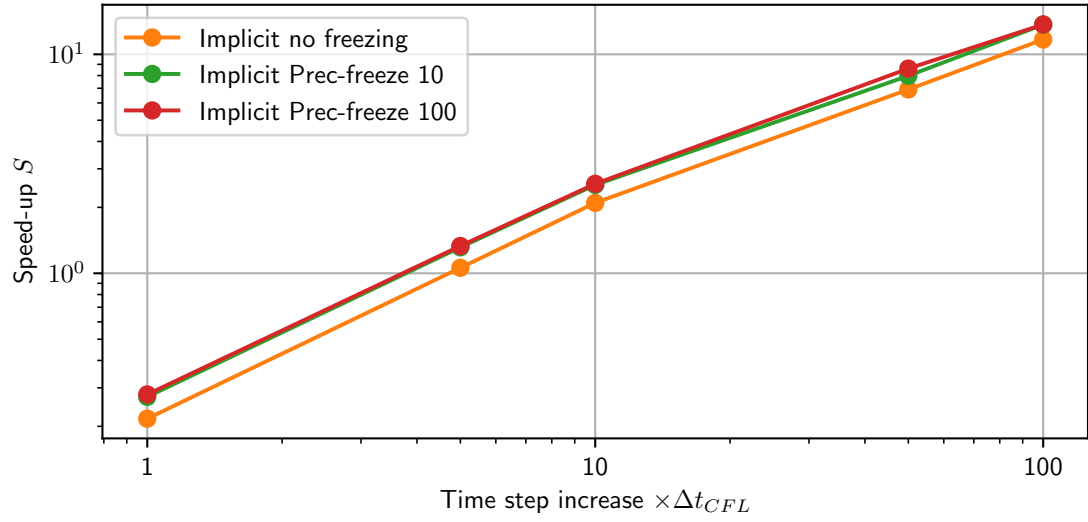
We investigate re-building for the low-energy block preconditioner that is applied to the ADR matrix of the velocity problem, see section 2.4.3. The preconditioner requires costs for setting up the low-energy basis preconditioner as well as the transformation of the original Schur complement $\mathbf{S}$ into the low-energy basis. These costs add up to almost 50% of the total execution time in the linear-implicit scheme due to re-building the matrix at every time step. We compare the effect of preconditioner freezing for the entire range of time step sizes 1 to $100\Delta t_{\text{CFL}}$ and highlight bottlenecks using a specific time step size of $10\Delta t_{\text{CFL}}$.

Figure 4.3a compares the practical speed-up against the time step size and the sum of velocity iterations for the linear-implicit scheme with 1, 10 and 100 freezing steps. The speed-up shows an improvement of 25.8% and 29.0% in the speed-up at the smallest time step $1\Delta t_{\text{CFL}}$ for 10 and 100 freezing steps, respectively. The trend stays similar for larger time steps with 10 and 100 freezing steps increasing the speed-up by 20.8% and 22.2% at $10\Delta t_{\text{CFL}}$, and an speed-up advantage of 16.9% and 17.1% at a 100-fold increase in Δt , respectively. Hence, the speed-up is most effective at smaller time steps and becomes less effective closer to the stability limit.

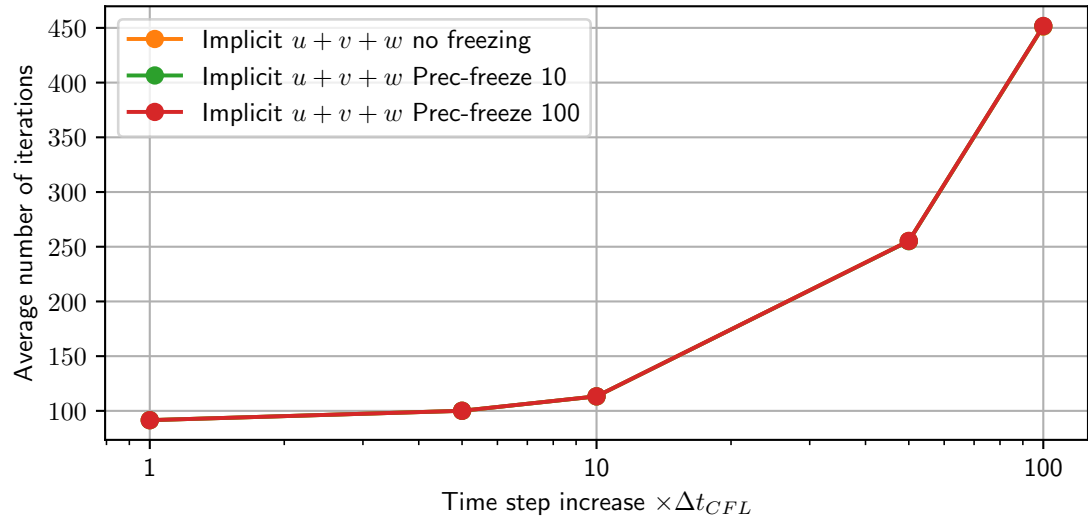
This decreasing improvement of preconditioner freezing is interesting when taking the average velocity iterations, an indicator for the work of solving the linear system, in figure 4.3b into account. The average iteration count is equivalent for all time step increases and, particularly, independent of the number of freezing steps. This appears reasonable since the flow is in a statistically steady state with only minor variations due to the chaotic nature of a turbulent flow. Thus, the conditioning of the velocity problem does not vary significantly.

This initial analysis shows that the preconditioner freezing does stagnate at around 100 freezing steps and decreases in effectiveness at larger Δt without increasing the iteration counts. Further, for a given time step of $10\Delta t_{\text{CFL}}$, we observe an improvement in the range 21% to 22% in the total execution time. Since we expect a reduction of up to 50% of execution time, we next investigate the case $10\Delta t_{\text{CFL}}$ in detail and identify bottlenecks which limit the improvement to around 20%.

Figure 4.4 compares timings for major components of the velocity update process similar to the previous section 4.2.2. We find a constant build time for the linear system $\mathbf{A}$ for 1, 10 and 100 freezing steps as this is independent of the preconditioner freezing. Further, the preconditioner freezing for 10 steps reduces the time spent building the preconditioner $\mathbf{M}^{-1}$ dramatically from 114 seconds to 19 seconds by a factor of 6. Increasing the freeze steps reduces the time further



(a) Speed-up measured as the time-to-solution ratio of semi-implicit scheme against the linear-implicit scheme.



(b) Iteration count for the sum of all three velocity problems.

Figure 4.3: Speed-up and number of iterations comparing the effect of preconditioner freezing for 1, 10, 100 steps for the linear-implicit scheme's velocity problem and with increasing time step size.

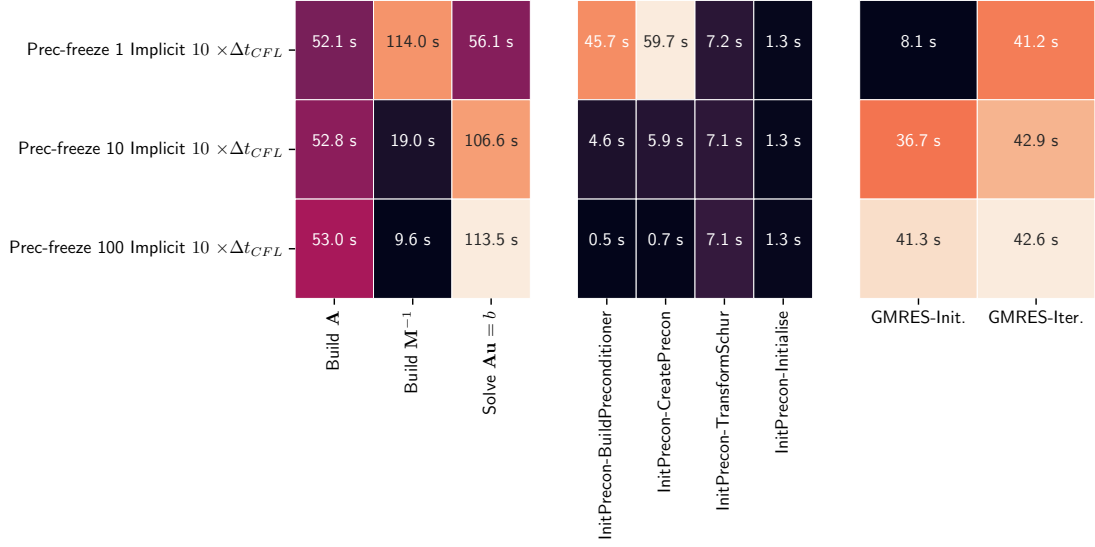


Figure 4.4: Heat map of execution time spent in the three sections of the velocity-correction scheme algorithm: explicit forcing, pressure solve and velocity solve. Additionally, we show timings of building and solving the linear systems. We show absolute time spent in seconds per 100 time steps as well as in percentage of total execution time. Comparing semi-implicit and linear-implicit scheme at increasing time step size.

to 9.6 seconds or an overall reduction by a factor of 11.9. The strong reduction here suggest a stronger speed-up than observed above in figure 4.3a. However, the freezing leads to an increase in the time spent solving the linear systems for velocity. The solving time increase by a factor of 1.9 and 2.0 for 10 and 100 freezing steps, respectively. Consequently, the overall improvement in solving the velocity systems is from 222.2 seconds without freezing to 178.4 seconds with 10 freezing steps that is a 24.5% improvement in the velocity solve. For 200 freezing steps, the velocity solves reduce to 176.1 seconds and a 26.2% improvement. This speed-up of the velocity solve is of similar magnitude as the practical speed-up observed in the discussion above.

Moreover, the central heat map in figure 4.4 shows timings for separate parts of building the preconditioner. We observe that building and transforming the preconditioner to the low-energy basis is a major component of the total build time for M^{-1} . These functions reduce by one and even two orders of magnitude when we re-build the preconditioner only every 10 or 100 steps, respectively. Additionally, the third and fourth columns show the time spent to transform the Schur complement matrix to the low-energy basis and assembling it into a sparse matrix structure. Since we are re-building the linear system, and thus the Schur complement, we cannot avoid these costs. In kind, the time spent for these operations 8.4 seconds give the stagnation

in improving the performance through preconditioner freezing. The time spent for re-building the preconditioner without freezing is dominant with 105.4 seconds. However, it reduces to the same order of magnitude at 10 freezing steps with 10.5 seconds and for 100 freezing steps reduces to a negligible 1.2 seconds. Thus, the 10 freezing steps reduces the overall cost for re-building strongly, however, below this the fixed cost for modifying the Schur complement become dominant and more freezing steps have a minor effect on reducing the the total time spent for the build preconditioner part of the algorithm.

We further investigated the increase in time to solve the three linear system for velocity. The right hand side heat map in figure 4.4 provides averaged timings for, on the one hand, the initialisation of the GMRES routine including allocation of temporary storage and building the initial residual r_0 . On the other hand, the second timer measures the Arnoldi process and Givens rotations which are the major cost per iteration. The first observation is that the time for solving the linear system does not match the sum of both timers. Without freezing, the mismatch is 6.8 seconds or about 12% of the solve time. With freezing, this mismatch increases to 27 and 29.6 seconds for 10 and 100 steps, respectively. We identified that the average timings do not accurately represent the time spent in these functions, because there is variations of up to two orders of magnitude. In particular, the large increase in the initialisation step of GMRES, going from no freezing to 10 or 100 freezing steps, is surprising, since the cost for this part of the algorithm are usually minor. We found that the all-to-all communication for the initial residual accounts for a majority of the time. These observations hint at a load imbalance between the 512 ranks used for these simulations. It is possible that the mesh decomposition strategy created uneven partitions which cause some ranks to take a significant amount of time while others are idling.

All in all, the preconditioner freezing has the potential to improve the practical speed-up. Interestingly, the potential speed-up is limited due to the required transformation of the sub-structured linear system to the low-energy space and possibly load imbalance.

Freezing the advection matrix

Next, we look at re-building the linear system including the preconditioner only every n th step. Figure 4.5a shows a consistent speed-up for all time steps with advection freezing. A freezing for two steps gives an almost equivalent increase in the speed-up. Additional freezing steps show diminishing returns when approaching a number of freezing steps $N_{\text{freeze}} > 100$. It is intriguing to see a shift in the performance break-even point with the semi-implicit scheme to around Δt_{CFL}

if we freeze for 100 time steps.

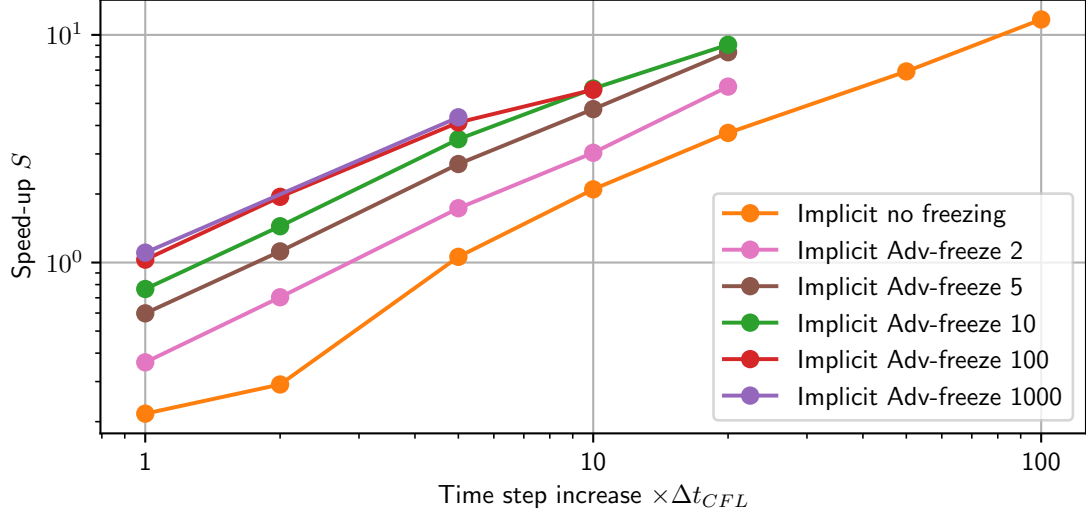
Moreover, we observe that the upper stability limit changes with advection freezing. Independent of the number of freezing steps, we find the upper limit at $20\Delta t_{\text{CFL}}$. This is likely due to the extrapolation in the linearised advection operator. For longer freezing steps, the linearised operator introduces larger errors from the current time step size which consequently reduce the stability margin. Nevertheless, it is surprising that there is a sharp stability limit for $20\Delta t_{\text{CFL}}$ as we expected a varying change in the stability margin.

We show the averaged velocity iterations in figure 4.5b. The GMRES iterations show almost no difference due to the freezing of the advection matrix. Independent of the number of freezing steps N_{freeze} , the average iteration counts and, hence, the work per time step are consistent. This coincides with the observed speed-up S trends for advection freezing in figure 4.5a where we see essentially the same linear trend independent of freezing. This hints at the matrix conditioning for the ADR problem and shows that the condition number is independent of the advection velocity over a longer physical time interval.

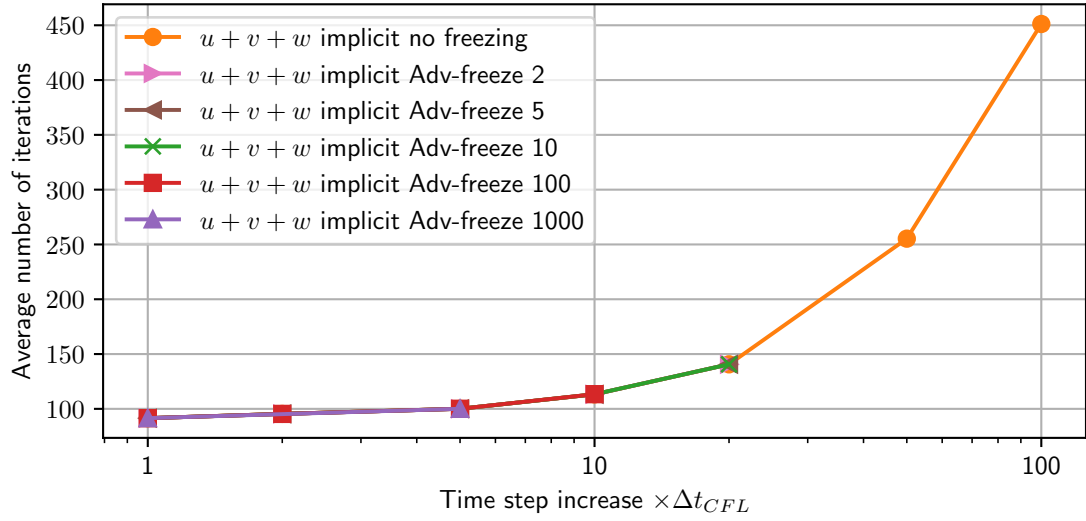
We want to further look into the algorithms bottlenecks when using advection freezing. The right hand side of figure 4.6 shows again the overview of the linear-implicit velocity correction scheme’s major components. We observe that the forcing terms show a slight increase once we use freezing. This is due to the additional evaluation of an advection term for the advection deficit. Nevertheless, this is a constant cost for any $N_{\text{freeze}} > 1$. The pressure shows variations that we assume independent of the freezing as commented on above in section 4.2.2. Moreover, the velocity solve sees a dramatic decrease by a factor of 6.1. This increase is close to the break-even with the semi-implicit scheme that discussed above. Hence, it explains that the linear-implicit scheme has similar cost per step as does the semi-implicit scheme.

The left hand side of figure 4.6 gives insight into the decrease of the execution time. Building the linear system $\mathbf{A}$ and the preconditioner $\mathbf{M}^{-1}$ almost linearly decrease with the freezing steps N_{freeze} . We measure a 10- and 100-fold decrease in building both. Interestingly this makes the time for solving the system significant for $N_{\text{freeze}} = 10$ and dominant for $N_{\text{freeze}} = 100$. Note that the solve time decreases with the freezing steps by 37.1% from no freezing to $N_{\text{freeze}} = 10$. This is unexpected, because the average iteration counts do not show a difference and there is no indication of decreasing runtime.

In summary, the potential of advection freezing for pure computational performance is large. We find that the linear-implicit scheme becomes competitive against the semi-implicit scheme at even the same time step size. Using the advection freezing for accurate simulations, however,



(a) Speed-up measured as the time-to-solution ratio of semi-implicit scheme against the linear-implicit scheme.



(b) Iteration count for the sum of all three velocity problems.

Figure 4.5: Speed-up and number of iterations comparing the effect of advection freezing for 1, 10, 100, 1000 steps for the linear-implicit scheme's velocity problem and with increasing time step size.

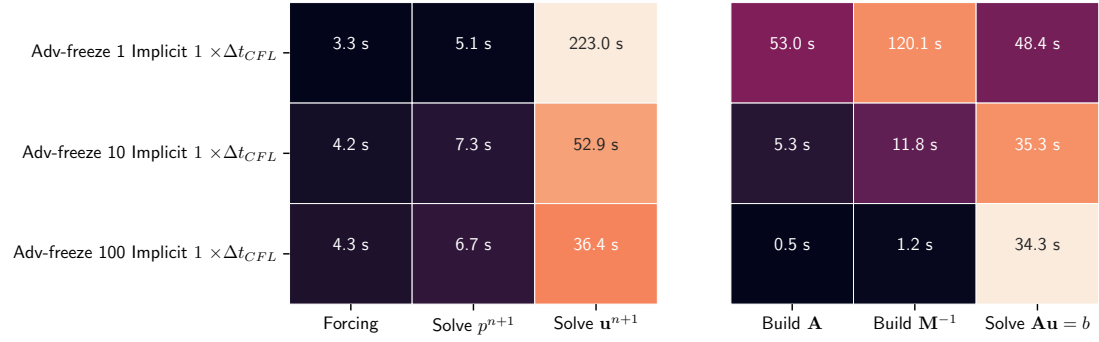


Figure 4.6: Heat map of execution time spent in building and solving the linear systems. We show absolute time spent in seconds per 100 time steps as well as in percentage of total execution time. Investigation into the effect of advection freezing.

requires an investigation into the effect on the accuracy when using small and large number of freezing steps which is not part of this work.

Chapter 5

Conclusions

5.1 Summary

In this work, we investigate a linear-implicit velocity correction scheme for scale-resolving simulations. We aim to improve the computational performance and numerical stability of velocity correction schemes for challenging test cases. The challenge comes from the combination of complex geometries with curved surfaces and high Reynolds number flow which enforce a strict Courant-Friedrichs-Levy (CFL) condition. This CFL condition constrains explicit velocity correction schemes to small time steps which in-turn limits their computational performance.

The limitation in computational performance in practise leads to a longer time-to-solution. This means it takes more computational resources to, first, reach the quasi-steady state and, second, sample statistics in the quasi-steady state. We investigate two research questions to reduce the time-to-solution. The first question focusses on reaching the quasi-steady state faster by trading an implicit time-stepping scheme that enables larger time step sizes against accuracy in time. The second question focusses on sampling statistics in the quasi-steady state. We study sampling statistics using an implicit scheme and larger time step sizes and compare those predictions against a semi-implicit scheme with smaller time step size and compare their accuracy in resolving integral force coefficients and surface quantities such as the skin friction coefficient.

We investigate the above research questions using a linear-implicit velocity correction scheme. The linear-implicit scheme allows increasing the time step size through an implicit treatment of the advection term. The implicit treatment uses a single Picard iteration for the velocity substep of the velocity correction scheme. The single Picard iteration linearises the scheme such that no

inner iterations are required. Still, the matrix coefficients become time-dependent which incurs larger computational cost per time step, however, an overall shorter time-to-solution through larger time step sizes. We study the linear-implicit scheme to explore the question on problems ranging from low to high complexity.

We implement and verify the linear-implicit scheme for the incompressible Navier-Stokes solver inside the Nektar++ framework. For verification, we study the stability and accuracy of the algorithm using manufactured solutions and compare against error estimates. Additionally, we use steady and unsteady canonical problems, including a 2D cylinder flow at a Reynolds number of 500, to verify the linear-implicit scheme against literature. Finally, with inspiration from the Formula 1 industry, we test the schemes suitability for real-world applications. The complex geometries and particularly challenging meshes required for Formula 1 aerodynamic devices, such as the front wing, use high Reynolds number flows where $\text{Re} = \mathcal{O}(10^5)$. These simulations encompass challenges for the numerical stability, accuracy and computational performance when simulating complex turbulent flows. Therefore, the study gives insight into the characteristics of the linear-implicit scheme for scale-resolving simulation of industrial geometries.

5.2 Conclusion

Our investigations consider scale-resolving simulations of complex geometries and high Reynolds number flow. We compare a semi-implicit velocity correction scheme with explicit advection treatment which is limited by the CFL condition to a time step size with $\text{CFL} \leq 1$, referred to as Δt_{CFL} . We study a linear-implicit velocity correction scheme which is stable for $\text{CFL} \gg 1$ and shows potential reductions in computational cost with minor influence on accuracy. In the following, we discuss findings for the 2D, quasi-3D and 3D implementations and consider numerical stability, accuracy and computational performance.

We find that the linear-implicit advection treatment enables up to two orders of magnitude larger time steps in 2D and 3D. This observation applies equally to canonical 2D cylinder flow at low Reynolds number and the highly complex simulations of a Formula 1 front wing geometry. Interestingly, the largest time step size corresponds to a CFL number $\text{CFL} \approx 100$ and that largely independent of the geometry and Reynolds number. Further increasing the time step size consistently lead to numerical instabilities and identified an upper stability limit. In contrast, the quasi-3D implementation of the scheme does not enable significant increases in the stability. We find that this is the result of a span-wise advection term that cannot be linearised when using

a half plane splitting of the Fourier modes into decoupled imaginary and real parts. Instead, a split into full Fourier modes with coupled imaginary and real parts potentially enables similar gains as we find for the 2D and 3D implementation. Overall, the linear-implicit scheme increases the numerical stability strongly providing potential for reducing the computational cost, first, to reach the quasi-steady state and, second, to sample statistics of forces and other quantities of interest.

We find that increasing the time step size to $10 \times \Delta t_{\text{CFL}}$ does not significantly change sensitive flow physics such as the laminar-turbulent transition. For example, we find that the impact on the cylinder and complex flows is a consistent shift in integral force coefficients, and transition location which highlights the minimal influence of the temporal error in this range. Further increasing the time step size to $\Delta t \gg 10\Delta t_{\text{CFL}}$ leads to stronger differences in the structure of the flow through dominant temporal errors. The 2D and 3D challenging front wing problems show that large time step sizes lead to a change in the laminar-turbulent transition and, consequently, strong changes in integral force coefficients. It becomes evident that the temporal error is usually small compared to the spatial error which allows larger time step sizes. However, as the linear-implicit scheme enables significant increases in the time step size, we reach temporal errors of a similar or larger magnitude as the spatial error which shows stronger effects on the flow physics.

We further find that the linear-implicit scheme allows strong reduction in computational resources. Within the Nektar++ framework we follow a substructuring approach which requires building matrices. The linear-implicit scheme shows a computational overhead of factor 5 compared to the baseline semi-implicit scheme. This overhead results mainly from re-building the time-dependent linear system and preconditioner. Despite the computational overhead, the linear-implicit scheme allows a 2-times speed-up at $10\Delta t_{\text{CFL}}$ and up to an order of magnitude speed-up for $100\Delta t_{\text{CFL}}$. The analysis shows that further computational gains are possible up to about $10\Delta t_{\text{CFL}}$ by using preconditioner freezing leading to about 20% greater speed-up. Additionally, we find that freezing the advection matrix enables a 5-times greater speed-up bringing the linear-implicit scheme's performance on par with the baseline semi-implicit performance at $1\Delta t_{\text{CFL}}$.

In conclusion, we emphasise the potential of the linear-implicit velocity correction scheme for high Reynolds number simulations of challenging geometries. Our investigations highlight that the linear-implicit scheme provides additional numerical stability which can be leveraged for an effective reduction in time-to-solution. Considering the first research question, we find that the computational cost for reaching the quasi-steady state are significantly reduced by using up to

two orders of magnitude larger time step sizes leading to a 10-fold reduction in time-to-solution. Furthermore, the answer to our second question shows that the computational resources for sampling statistics in the quasi-steady state are reduced by factor 2 using an order of magnitude larger time step size with the linear-implicit scheme.

5.3 Future work

The investigation of this work on the linear-implicit velocity correction scheme highlight key characteristics of the algorithm. Further investigation should identify the current upper stability limit of the linear-implicit scheme. Larger time steps should theoretically be possible and a truly unconditionally stable is highly sought after for academic and industrial applications. Still, the linear-implicit scheme shows an upper stability limit at about two orders of magnitude increase in the time step size corresponding to a CFL number $CFL > 100$. While non-linear stability analysis predicts unconditional stability for this scheme, it seems non-linearities in the problem lay a practical stability limit on the linear-implicit scheme. Our studies attempted to rule out major stability hurdles such as dealiasing for the non-linear term and curved geometries, spectral vanishing viscosity for under-resolved simulations, Taylor-Hood approximation spaces for the inf-sup, or LBB, condition. We do not explore the possible stabilisation of penalty terms such as a divergence penalty or gradient jump penalisation. Further improvements in stability may be possible through a Picard iteration of the velocity ADR problems. The current linearised advection operator is equivalent to a single Picard iteration. Multiple iterations potentially improve the stability of the algorithm. A performant method that circumvents the upper stability limit further improves the suitability of scale-resolving simulation to large Reynolds number flows on highly complex geometries for CFD practitioners.

The current comparison of accuracy focusses on integral force coefficients and the surface distribution of pressure and skin friction. These are good indicators for larger changes in the flow structure. Still, there is potentially more fine-grained changes within the boundary layer on the wing elements. A further comparison based on Reynolds stresses close to the wall and the turbulent kinetic energy in the wake would give more insight into the effect of larger time step sizes on the turbulent flow around complex geometries.

The stability margin for the extruded Imperial Front Wing geometry shows a margin of factor 100. This factor is case-dependent and it would be interesting to investigate the effect of varying the polynomial order P on the stability margin. We investigate the variation for the Kovasznay

flow and find an increase with larger polynomial order P , see figure 3.3. This however would be interesting to see for a complex problem such as the extruded Imperial Front Wing.

The performance analysis in this framework focusses on central processing unit (CPU) hardware. It is however becoming common practise to accelerate large-scale simulations using graphics processing units (GPU). The design of GPUs is tailored for highly parallel workloads. Hence, they are well suited for the core workloads of large-scale simulations where we solve large linear systems for the pressure and velocity problems using matrix-vector products of the form Ax . An investigation into the performance on GPU hardware would be insightful for the performance of the linear-implicit scheme on current and future hardware of high performance computing systems.

The algorithmic bottlenecks through matrix re-building pointed out in this work highlight the major computational overhead of the linear-implicit algorithm. It also highlights the potential of a matrix-free implementation. A matrix-free kernel removes the necessity to build matrices and, ideally, preconditioners. This removes major bottlenecks in the linear-implicit algorithm and pushes its performance to almost the same degree as the semi-implicit scheme whilst enabling improved stability. It further allows using accelerators such as graphics processing units (GPU) effectively. GPUs leverage larger computational throughput in comparison to central processing units (CPU). A matrix-free implementation is particularly amenable as it reduces the loads and stores required compared to matrix-based solvers. Further studies should explore the performance of the semi- and linear-implicit algorithms using matrix-free operators and conduct roofline analysis to identify potential differences.

Further, we suggest investigating the scalability of both algorithms and, in particular, the iterative solvers. The advection-diffusion-reaction matrix requires a solver for general matrices. This work explores the generalised minimum residual (GMRES) solver. GMRES requires more communication compared to the conjugate gradients (CG) solver used for the symmetric positive-definite matrices of the Poisson and Helmholtz matrices. The amount of communication in GMRES is particularly growing with the number of iterations due to the Gram-Schmidt orthogonalisation. An alternative method would be the biconjugate gradient stabilised (BiCGStab) method as a scalable algorithm suitable for large-scale simulations with the linear-implicit scheme.

Adaptive time-stepping further offers many use cases for CFD tools in practice. It allows stretching the temporal accuracy by a defined margin whilst using as much performance gain as possible. Potential sensors could be the maximum Courant-Friedrichs-Levy (CFL) estimated for the flow at a given time step or the number of iterations required to solve the velocity problems.

We emphasize that an adaptive approach for the linear-implicit velocity correction scheme adds to the automation of CFD.

Bibliography

- [1] Jeffrey P. Slotnick et al. “Enabling the Environmentally Clean Air Transportation of the Future: a Vision of Computational Fluid Dynamics in 2030”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 372.2022 (Aug. 13, 2014).
- [2] S. B. Pope. *Turbulent Flows*. Cambridge: Cambridge University Press, 2000.
- [3] Nicholas J. Georgiadis et al. “Large-Eddy Simulation: Current Capabilities, Recommended Practices, and Future Research”. In: *AIAA Journal* 48.8 (Aug. 2010), pp. 1772–1784.
- [4] Philippe R. Spalart. “Philosophies and Fallacies in Turbulence Modeling”. In: *Progress in Aerospace Sciences* 74 (2015), pp. 1–15.
- [5] Karthik Duraisamy et al. “Turbulence Modeling in the Age of Data”. In: *Annual Review of Fluid Mechanics* 51. Volume 51, 2019 (2019), pp. 357–377.
- [6] Elia Merzari et al. “Toward Exascale: Overview of Large Eddy Simulations and Direct Numerical Simulations of Nuclear Reactor Flows with the Spectral Element Method in Nek5000”. In: *Nuclear Technology* 206.9 (Sept. 2020), pp. 1308–1324.
- [7] Paul Fischer et al. “NekRS, a GPU-Accelerated Spectral Element Navier-Stokes Solver”. In: *Parallel Computing* 114 (2022), p. 102982.
- [8] Niclas Jansson et al. “Neko: A Modern, Portable, and Scalable Framework for High-Fidelity Computational Fluid Dynamics”. In: *CoRR* abs/2107.01243 (2021).
- [9] David Moxey et al. “Nektar++: Enhancing the Capability and Application of High-Fidelity Spectral/hp Element Methods”. In: *Computer Physics Communications* 249. Journal Article (2020), p. 107110.
- [10] David E. Keyes et al. “Implicit Solvers for Large-Scale Nonlinear Problems”. In: *Journal of Physics: Conference Series* 46 (Sept. 2006), pp. 433–442.

- [11] Pierre Sagaut. *Large eddy simulation for incompressible flows an introduction*. Third edition. Scientific computation. Berlin ; Springer-Verlag, 2006.
- [12] J. Smagorinsky. “General Circulation Experiments with the Primitive Equations: I. The Basic Experiment”. English. In: *Monthly Weather Review* 91.3 (1963), pp. 99–164.
- [13] M. Germano et al. *A Dynamic Subgrid-Scale Eddy Viscosity Model*. 1991.
- [14] A. W. Vreman. “An Eddy-Viscosity Subgrid-Scale Model for Turbulent Shear Flow: Algebraic Theory and Applications”. In: *Physics of Fluids* 16.10 (Oct. 2004), pp. 3670–3681.
- [15] Fenando F. Grinstein et al. *Implicit large eddy simulation : computing turbulent fluid dynamics*. Cambridge: Cambridge University Press, 2007.
- [16] Thibault Dairay et al. “Numerical Dissipation Vs. Subgrid-Scale Modelling for Large Eddy Simulation”. In: *Journal of Computational Physics* 337 (2017), pp. 252–274.
- [17] Eitan Tadmor. “Convergence of Spectral Methods for Nonlinear Conservation Laws”. In: *SIAM Journal on Numerical Analysis* 26.1 (1989), pp. 30–44.
- [18] K Arrow et al. *Studies in Nonlinear Programming*. Stanford University Press, Stanford, 1958.
- [19] George Karniadakis and Spencer Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. 2nd ed. Oxford University Press, 2013. Chap. 2.
- [20] Alexandre Joel Chorin. “Numerical Solution of the Navier-Stokes Equations”. In: *Mathematics of Computation* 22.104 (1968), pp. 745–762.
- [21] George Em Karniadakis et al. “High-Order Splitting Methods for the Incompressible Navier-Stokes Equations”. In: *Journal of Computational Physics* 97.2 (1991), pp. 414–443.
- [22] J. Guermond and J. Shen. “Velocity-Correction Projection Methods for Incompressible Flows”. In: *SIAM Journal on Numerical Analysis* 41.1 (2003), pp. 112–134.
- [23] Johnathan Mark Pegrum. “Experimental Study of the Vortex System Generated by a Formula 1 Front Wing”. PhD Thesis. Imperial College London, 2006.
- [24] Jean-Eloi William Lombard. “A High-Fidelity Spectral/hp Element LES Study of Formula 1 Front-Wing and Exposed Wheel Aerodynamics”. PhD Thesis. Imperial College London, 2017.
- [25] Filipe Fabian Buscariolo. *Imperial Front Wing (IFW) geometry*. Aug. 2019.

- [26] Martin Karp et al. “Large-Scale Direct Numerical Simulations of Turbulence Using GPUs and Modern Fortran”. In: *The International Journal of High Performance Computing Applications* 37.5 (Feb. 2023), pp. 487–502.
- [27] Daniele Massaro et al. “Direct Numerical Simulation of the Turbulent Flow Around a Flettner Rotor”. In: *Scientific Reports* 14.1 (2024), p. 3004.
- [28] T. C. L. Fava et al. “Transition to Turbulence on a Rotating Wind Turbine Blade at $Re_c = 3 \times 10^5$ ”. In: *Journal of Fluid Mechanics* 999.A54 (2024), A54.
- [29] Gerasimos Ntoulas et al. “A Comparative Study of Explicit and Implicit Large Eddy Simulations Using a High-Order Discontinuous Galerkin Solver: Application to a Formula 1 Front Wing”. In: *Results in Engineering* 25 (2025), p. 104425.
- [30] Guglielmo Vivarelli et al. *High-Order Spectral/hp Compressible and Incompressible Comparison of Transitional Boundary-Layers Subject to a Realistic Pressure Gradient and High Reynolds Number*. June 2022.
- [31] Mohsen Lahooti et al. “LES/DNS Fluid-Structure Interaction Simulation of Non-Linear Slender Structures in Nektar++ Framework”. In: *Computer Physics Communications* 282 (2023), p. 108528.
- [32] James Slaughter et al. “Large Eddy Simulation of an Inverted Multi-Element Wing in Ground Effect”. In: *Flow, Turbulence and Combustion* 110.4 (Feb. 2023), pp. 917–944.
- [33] Gianmarco Mengaldo et al. “Industry-Relevant Implicit Large-Eddy Simulation of a High-Performance Road Car via Spectral/hp Element Methods”. In: *SIAM Rev.* 63.4 (Jan. 2021), pp. 723–755.
- [34] Filipe F. Buscariolo et al. “Spectral/hp Element Simulation of Flow Past a Formula One Front Wing: Validation against Experiments”. In: *Journal of Wind Engineering and Industrial Aerodynamics* 221 (2022), p. 104832.
- [35] Richard D. Loft et al. “Terascale Spectral Element Dynamical Core for Atmospheric General Circulation Models”. In: *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing*. SC '01. Denver, Colorado: Association for Computing Machinery, 2001, p. 18.
- [36] D. Xiu et al. “Strong and Auxiliary Forms of the Semi-Lagrangian Method for Incompressible Flows”. In: *Journal of Scientific Computing* 25.1 (2005), pp. 323–346.

- [37] Lianlei Lin et al. “An Energy-Stable Scheme for Incompressible Navier-Stokes Equations with Periodically Updated Coefficient Matrix”. In: *Journal of Computational Physics* 418 (2020), p. 109624.
- [38] J. C. Simo and F. Armero. “Unconditional Stability and Long-Term Behavior of Transient Algorithms for the Incompressible Navier-Stokes and Euler Equations”. In: *Computer Methods in Applied Mechanics and Engineering* 111.1 (1994), pp. 111–154.
- [39] S. Dong and J. Shen. “An Unconditionally Stable Rotational Velocity-Correction Scheme for Incompressible Flows”. In: *Journal of Computational Physics* 229.19 (2010), pp. 7013–7029.
- [40] E. Ferrer et al. “HORSES3D: A High-Order Discontinuous Galerkin Solver for Flow Simulations and Multi-Physics Applications”. In: *Computer Physics Communications* 287 (2023), p. 108700.
- [41] F. D. Witherden et al. “PyFR: an Open Source Framework for Solving Advection–diffusion Type Problems on Streaming Architectures Using the Flux Reconstruction Approach”. In: *Computer Physics Communications* 185.11 (2014), pp. 3028–3040.
- [42] Michel O. Deville et al. *High-Order Methods for Incompressible Fluid Flow*. Vol. 9. Cambridge university press, 2002.
- [43] C. D. Cantwell et al. “Nektar++: an Open-Source Spectral/hp Element Framework”. In: *Computer Physics Communications* 192 (2015), pp. 205–219.
- [44] M. D. Green et al. “NekMesh: an Open-Source High-Order Mesh Generation Framework”. In: *Computer Physics Communications* 298 (2024), p. 109089.
- [45] Kaloyan S. Kirilov et al. “High-Order Curvilinear Mesh Generation from Third-Party Meshes”. In: *Proceedings of the 2024 International Meshing Roundtable (IMR)*. SIAM, 2024. Chap. High-Order Curvilinear Mesh Generation From Third-Party Meshes, pp. 93–105.
- [46] D. Moxey et al. “Optimising the Performance of the Spectral/hp Element Method with Collective Linear Algebra Operations”. In: *Computer Methods in Applied Mechanics and Engineering* 310 (2016), pp. 628–645.
- [47] Rodrigo C. Moura et al. “Spatial Eigenanalysis of Spectral/hp Continuous Galerkin Schemes and Their Stabilisation via DG-Mimicking Spectral Vanishing Viscosity for High Reynolds Number Flows”. In: *Journal of Computational Physics* 406 (2020), p. 109112.

- [48] L. I. G. Kovasznay. “Laminar Flow Behind a Two-Dimensional Grid”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 44.1 (1948), pp. 58–62.
- [49] W. Toet. “Aerodynamics and Aerodynamic Research in Formula 1”. In: *The Aeronautical Journal* 117.1187 (2013), pp. 1–26.
- [50] Ivo Babuška. “Error-Bounds for Finite Element Method”. In: *Numerische Mathematik* 16.4 (1971), pp. 322–333.
- [51] Franco Brezzi. “On the Existence, Uniqueness and Approximation of Saddle-Point Problems Arising from Lagrangian Multipliers”. In: *Publications des séminaires de mathématiques et informatique de Rennes* S4 (1974), pp. 1–26.
- [52] Franco Brezzi and Richard S. Falk. “Stability of Higher-Order Hood-Taylor Methods”. In: *SIAM J. Numer. Anal.* 28.3 (June 1991), pp. 581–590.
- [53] David Moxey et al. “Efficient Matrix-Free High-Order Finite Element Evaluation for Simplicial Elements”. In: *SIAM Journal on Scientific Computing* 42.3 (2020), pp. C97–C123.
- [54] Spencer J. Sherwin. “Hierarchical hp Finite Elements in Hybrid Domains”. In: *Finite Elements in Analysis and Design* 27.1 (1997), pp. 109–119.
- [55] Nico Pietroni et al. “Hex-Mesh Generation and Processing: a Survey”. In: *ACM Trans. Graph.* 42.2 (Oct. 2022).
- [56] Steven A. Orszag. “Spectral Methods for Problems in Complex Geometries”. In: *Numerical Methods for Partial Differential Equations*. Academic Press, 1979, pp. 273–305.
- [57] Rodrigo C. Moura et al. “Gradient Jump Penalty Stabilisation of Spectral/hp Element Discretisation for Under-Resolved Turbulence Simulations”. In: *Computer Methods in Applied Mechanics and Engineering* 388 (2022), p. 114200.
- [58] Jiaqing Kou et al. “Jump penalty stabilization techniques for under-resolved turbulence in discontinuous Galerkin schemes”. In: *Journal of Computational Physics* 491 (2023), p. 112399.
- [59] S. Dong et al. “A Robust and Accurate Outflow Boundary Condition for Incompressible Flow Simulations on Severely-Truncated Unbounded Domains”. In: *Journal of Computational Physics* 261 (2014), pp. 83–105.
- [60] Esteban Ferrer. “An interior penalty stabilised incompressible discontinuous Galerkin-Fourier solver for implicit large eddy simulations”. In: *Journal of Computational Physics* 348 (2017), pp. 754–775.

- [61] G. Mengaldo et al. “Dealiasing Techniques for High-Order Spectral Element Methods on Regular and Irregular Grids”. In: *Journal of Computational Physics* 299 (2015), pp. 56–81.
- [62] H. M. Tufo and P. F. Fischer. “Fast Parallel Direct Solvers for Coarse Grid Problems”. In: *Journal of Parallel and Distributed Computing* 61.2 (2001), pp. 151–177.
- [63] Howard C. Elman et al. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford University Press, USA, 2006.
- [64] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. JHU press, 2013.
- [65] Magnus Rudolph Hestenes, Eduard Stiefel, et al. *Methods of Conjugate Gradients for Solving Linear Systems*. Vol. 49. 1. NBS Washington, DC, 1952.
- [66] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.
- [67] Spencer J. Sherwin and Mario Casarin. “Low-Energy Basis Preconditioning for Elliptic Substructured Solvers Based on Unstructured Spectral/hp Element Discretization”. In: *Journal of Computational Physics* 171.1 (2001), pp. 394–417.
- [68] Henrik Wüstenberg et al. “Efficient Implicit Time-Stepping for High Reynolds Number Flows Around Complex Geometries”. In: *Direct and Large Eddy Simulation 14 Proceedings*. Mar. 2025.
- [69] G. E. Karniadakis. “Spectral Element-Fourier Methods for Incompressible Turbulent Flows”. In: *Computer Methods in Applied Mechanics and Engineering* 80.1 (1990), pp. 367–380.
- [70] Mohsen Lahooti et al. *Under-Resolved Direct Numerical Simulation of NACA0012 at Stall*. June 2022.
- [71] G Vivarelli et al. “On the Effect of Curvature and Compressibility in Laminar Boundary Layers over Fan Blades”. In: 2023.
- [72] Andrea Cassinelli. “A Spectral/hp Element DNS Study of Flow Past Low-Pressure Turbine Cascades and the Effects of Inflow Conditions”. PhD thesis. Imperial College London, Mar. 2021.
- [73] A. Bolis et al. “An Adaptable Parallel Algorithm for the Direct Numerical Simulation of Incompressible Turbulent Flows Using a Fourier Spectral/hp Element Method and MPI Virtual Topologies”. In: *Computer Physics Communications* 206 (2016), pp. 17–25.
- [74] Spencer J. Sherwin et al. *Nektar++ Spectral/hp Element Framework*. Ed. by Chris Cantwell. June 16, 2025. URL: <https://www.nektar.info/>.

- [75] Daniel Lindblad et al. “Nektar++: Development of the Compressible Flow Solver for Jet Aeroacoustics”. In: *Computer Physics Communications* 300 (2024), p. 109203.
- [76] Will Schroeder et al. *The Visualization Toolkit (2nd ed.): An Object-Oriented Approach to 3D Graphics*. USA: Prentice-Hall, Inc., 1998.
- [77] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Clifton Park, NY, USA: Kitware, Inc., 2015.
- [78] C. Hirsch. *Numerical Computation of Internal and External Flows. Volume 1, Fundamentals of Computational Fluid Dynamics*. 2nd ed. Amsterdam ; Butterworth-Heinemann, 2007.
- [79] Feng Chen and Jie Shen. “Stability and Convergence Analysis of Rotational Velocity Correction Methods for the Navier-Stokes Equations”. In: *Advances in Computational Mathematics* 45.5 (2019), pp. 3123–3136.
- [80] J. Guermond et al. “Error Analysis of Fully Discrete Velocity-Correction Methods for Incompressible Flows.” In: *Mathematics of Computation* (77 2008), pp. 1387–1405.
- [81] Aaron Meurer et al. “SymPy: Symbolic Computing in Python”. In: *PeerJ Computer Science* 3 (Jan. 2017), e103.
- [82] Christophe Geuzaine and Jean-François Remacle. “Gmsh: a 3-D Finite Element Mesh Generator with Built-In Pre- and Post-Processing Facilities”. In: *Int. J. Numer. Meth. Engng.* 79.11 (Sept. 2009), pp. 1309–1331.
- [83] Dwight Barkley and Ronald D. Henderson. “Three-Dimensional Floquet Stability Analysis of the Wake of a Circular Cylinder”. In: *Journal of Fluid Mechanics* 322 (1996), pp. 215–241.
- [84] C.-Y. Wen and C.-Y. Lin. “Two-dimensional vortex shedding of a circular cylinder”. In: *Physics of Fluids* 13.3 (Mar. 2001), pp. 557–560.
- [85] Thomas Schütz. *Hucho-Aerodynamik Des Automobils: Strömungsmechanik, Wärmetechnik, Fahrdynamik, Komfort. Strömungsmechanik, Wärmetechnik, Fahrdynamik, Komfort*. Springer Vieweg Wiesbaden, Sept. 12, 2013.
- [86] Xin Zhang et al. “Ground Effect Aerodynamics of Race Cars”. In: *Appl. Mech. Rev* 59.1 (Jan. 2006), pp. 33–49.

- [87] Andrew S. O’Sullivan and Esteban Ferrer. “Large Eddy Simulations Strategies for Formula 1 Front Wings,Benefits and Challenges Compared to RANS Modeling”. In: *SSRN preprint* (Feb. 12, 2025).
- [88] Siemens Digital Industries Software. *Simcenter STAR-CCM+, version 2021.1*. Version 2021.1. 2021.
- [89] P. Welch. “The Use of Fast Fourier Transform for the Estimation of Power Spectra: a Method Based on Time Averaging over Short, Modified Periodograms”. In: *IEEE Transactions on Audio and Electroacoustics* 15.2 (1967), pp. 70–73.
- [90] Imperial College London. *Reserach Computing Service*.
- [91] David Silvester et al. “Efficient Preconditioning of the Linearized Navier-Stokes Equations for Incompressible Flow”. In: *Journal of Computational and Applied Mathematics* 128.1 (2001), pp. 261–279.
- [92] Claudio Canuto et al. “Finite-Element Preconditioning of G-Ni Spectral Methods”. In: *SIAM J. Sci. Comput.* 31.6 (Jan. 2010), pp. 4422–4451.
- [93] Pedro D. Bello-Maldonado and Paul F. Fischer. “Scalable Low-Order Finite Element Preconditioners for High-Order Spectral Element Poisson Solvers”. In: *SIAM J. Sci. Comput.* 41.5 (Jan. 2019), S2–S18.

Appendix A

Manufactured solution: Sympy script

A common way of debugging a PDE solver is using manufactured solutions. A manufactured solution is an analytical solution to a PDE that does not necessary represent a physical phenomena. Still they are a useful method to validate a code.

The following script uses the python library sympy to construct a manufactured solution for the three-dimensional, unsteady and incompressible Navier-Stokes equations.

```

from sympy import *

# Setup symbols
x, y, z, t, A, a, b, c, d, PI, Kinvis = symbols('x-y-z-t-A-a-b-c-d-PI-Kinvis')
init_printing(use_unicode=True)

# Define a solution
u = 1/a * sin(a*x) * cos(b*y) * cos(c*z) * sin(d*t)
v = 1/b * cos(a*x) * sin(b*y) * cos(c*z) * sin(d*t)
w = -2/c * cos(a*x) * cos(b*y) * sin(c*z) * sin(d*t)
p = A * sin(a*x) * sin(b*y) * sin(c*z) * cos(d*t)
print("Input solution is:")
print("u:", u, "\nv:", v, "\nw:", w, "\np:", p)

# Navier-Stokes terms
diffusionx = Kinvis * (u.diff(x).diff(x) + u.diff(y).diff(y) + u.diff(z).diff(z))
diffusiony = Kinvis * (v.diff(x).diff(x) + v.diff(y).diff(y) + v.diff(z).diff(z))
diffusionz = Kinvis * (w.diff(x).diff(x) + w.diff(y).diff(y) + w.diff(z).diff(z))
gradpx = p.diff(x)
gradpy = p.diff(y)
gradpz = p.diff(z)
advectionx = u * u.diff(x) + v * u.diff(y) + w * u.diff(z)
advectiony = u * v.diff(x) + v * v.diff(y) + w * v.diff(z)
advectionz = u * w.diff(x) + v * w.diff(y) + w * w.diff(z)
dudt = u.diff(t)
dvdt = v.diff(t)
dwdt = w.diff(t)

# The manufactured forcing terms
print("n-Forcing-terms-that-balance-each-momentum-equation\n")
fx = dudt + advectionx + gradpx - diffusionx
fy = dvdt + advectiony + gradpy - diffusiony
fz = dwdt + advectionz + gradpz - diffusionz
print("fx:", simplify(fx), "\nfy:", simplify(fy), "\nfz:", simplify(fz))

# Double-check momentum balance and divergence-free condition
checkx = fx - (dudt + advectionx + gradpx - diffusionx)
checky = fy - (dvdt + advectiony + gradpy - diffusiony)
checkz = fz - (dwdt + advectionz + gradpz - diffusionz)
print("x-momentum:", simplify(checkx), "\ny-momentum:", simplify(checky),
"\nz-momentum:", simplify(checkz))

divfree = u.diff(x) + v.diff(y) + w.diff(z)
print("div-free:", simplify(divfree))

```