Monolithic multigrid methods for high-order discretizations of time-dependent PDEs

by

© Razan Abu-Labdeh

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Ph.D of Science.

Department of Mathematics and Statistics

Memorial University

September 2023

St. John's, Newfoundland and Labrador, Canada

Abstract

A currently growing interest is seen in developing solvers that couple high-fidelity and higher-order spatial discretization schemes with higher-order time stepping methods for various time-dependent fluid plasma models. These problems are famously known to be stiff, thus only implicit time-stepping schemes with certain stability properties can be used. Of the most powerful choices are the implicit Runge-Kutta methods (IRK). However, they are multi-stage, often producing a very large and nonsymmetric system of equations that needs to be solved at each time step. There have been recent efforts on developing efficient and robust solvers for these systems. We have accomplished this by using a Newton-Krylov-multigrid approach that applies a multigrid preconditioner monolithically, preserving the system couplings, and uses Newton's method for linearization wherever necessary. We show robustness of our solver on the single-fluid magnetohydrodynamic (MHD) model, along with the (Navier-)Stokes and Maxwell's equations. For all these, we couple IRK with higher-order (mixed) finite-element (FEM) spatial discretizations. In the Navier-Stokes problem, we further explore achieving more higher-order approximations by using nonconforming mixed FEM spaces with added penalty terms for stability. While in the Maxwell problem, we focus on the rarely used E-B form, where both electric and magnetic fields are differentiated in time, and overcome the difficulty of using FEM on curved domains by using an elasticity solve on each level in the non-nested hierarchy of meshes in the multigrid method.

With all my love, to my parents and husband.

Lay summary

Numerous real world applications can be described using mathematical models making understanding them crucial in enhancing our lives. For many, the exact solution is often either unknown or very expensive to calculate. Numerical methods are used instead to find an accurate approximation. These are all applied on a discrete formulation of the original model, achieved by applying discretization schemes. Many time-dependent problems of interest are especially difficult to solve since they require these schemes to have certain stability properties. Although some powerful choices exist, they are rarely used since their application produces a system of equations that is very large and difficult to solve. This leads to a clear lack of studies developing solvers for these types of systems.

The goal of this thesis is to find accurate approximations while spending the least amount of computational time and with minimal memory requirements utilizing the available parallel hardware. Several numerical schemes can be developed and carefully combined to do so. One common technique, proven to be effective, is using multiple levels of mesh hierarchy in what are called multigrid methods. Optimization of these methods and the use of specific discretization schemes give us the opportunity to build solvers that find increasingly accurate approximate solutions.

Generally when solvers are developed, robustness is shown by testing their performance in solving more than one mathematical problem. One group of most interest is fluid flow problems due to their vast real world applications in aerodynamics, oceanography, meteorology, biology and many more areas. These models are famously tricky to solve exactly, or numerically using only a single technique. In addition, they might be modelled on curved domains adding to their numerical difficulty. Manipulation of the multigrid process included in the solver can adjust for this. The overall goal of this thesis is to develop an effective and robust numerical solver for such time-dependent problems with highly stable discretizations.

Acknowledgements

I would like to dedicate this space to thank those who have been my continuous source of support throughout my graduate studies journey.

I would like to express my deepest appreciation to my supervisor Dr. Scott MacLachlan for his many years of guidance. Through him, I was introduced to the world of multigrid methods which I am grateful for. His support, kindness and feedback along the way have been crucial in the accomplishment of my degree. Graduating is bitter-sweet as I will definitely miss our weekly chats. I truly couldn't have asked for a better mentor.

My extreme gratitude encompasses my family since without them I would not be where I am today. My parents, Dr. Abdel-Rahman and Manal, your love and encouragement over the years of distance learning (and before) has meant to me more than words can express. A sincere extension of my thanks is to my husband Abedalsalam for all his love, patience and constant uplift of my spirits. Without you my journey, both now and in the future, would be so lonely. I also greatly thank my sisters and brother for all your support and encouragement. I am blessed to have you all in my life.

Many thanks to the head of the mathematics department at Memorial University, Dr. JC Loredo-Osti, for giving me the multiple opportunities of teaching several undergraduate courses in the past couple of years. I extend my thanks to the examiners of this thesis for their time and comments. I wish to acknowledge the financial assistance provided by the School of Graduate Studies, Department of Mathematics and Statistics, and Natural Sciences and Engineering Research Council of Canada, in the form of graduate fellowships and teaching assistantships.

Statement of contribution

The work presented in Chapter 3 is the result of a collaboration between Razan Abu-Labdeh, Patrick E. Farrell and Scott P. MacLachlan. All algorithms included were developed by all parties, while the program coding used in the research and writing of the work was done by Razan. Supervision and editing of the written chapter was done by Scott. Additional publication editing of the written chapter was done by Patrick. This work is published as *Monolithic multigrid for implicit Runge-Kutta discretizations of incompressible fluid flow* in the Journal of Computational Physics, 2023.

The work presented in Chapter 4 is the result of a collaboration between Razan Abu-Labdeh, Patrick E. Farrell, Robert C. Kirby and Scott P. MacLachlan. All algorithms included were developed by all parties, while the program coding used in the research and writing of the work was done by Razan. Supervision and editing of the written chapter was done by Scott.

The work presented in Chapter 5 is the result of a collaboration between Razan Abu-Labdeh, Patrick E. Farrell, Scott P. MacLachlan and Eric C. Cyr. All algorithms included were developed by all parties, while the program coding used in the research and writing of the work was done by Razan. Supervision and editing of the written chapter was done by Scott.

Table of contents

List of tables

List of figures

Chapter 1

Introduction

Typically, mathematical models are derived from real world physical applications in many fields, such as medicine, physics, geology, and engineering. Understanding these models is vital since they affect many aspects of our day to day lives such as the travel we use, the climate predictions we make, and the technology we rely on. These models often take the form of partial differential equations (PDEs). In practice, numerical methods are often applied to systems of equations that are a discrete formulation of these considered continuum mathematical models. In time-dependent PDEs, discretization methods are applied to both the space and time functions to produce the fully discretized system of equations. Out of the many choices of temporal discretization schemes, Runge-Kutta methods are considered one of the most powerful schemes to use. Since many of the PDEs of interest are stiff problems, among the many Runge-Kutta classifications, implicit Runge-Kutta (IRK) methods must be applied. Although interest has grown in developing numerical algorithms and solvers for various stiff PDEs with an IRK discretization, there is still a lack of efficient solvers. The main goal of this thesis is to develop a novel numerical solver that is both efficient and robust for stiff PDEs with high-order IRK temporal and mixed finite-element discretizations resulting in a saddle-point structured system.

Fluid flow models such as time-dependent (Navier-)Stokes and magnetohydrodynamics (MHD) are all examples of stiff PDEs with important real applications. Although no universal definition exists of stiff differential equations, they are known to be difficult to solve numerically due to the requirement of using a smaller time step size than accuracy requires in order to avoid instability of the temporal discretization scheme. This makes stability of the numerical scheme chosen a crucial difficulty when dealing with these types of systems. In 1928, the first analysis of the instability of temporal discretization schemes and the need for a restricted time step size was done on hyperbolic PDEs by Courant, Friedrichs, and Lewy [11]. It has been shown that, because of this strict limitation on the time step size, explicit methods cannot be efficiently used as they are conditionally stable since they have a limited domain of stability. Instead, implicit methods are preferred since they tend to have unlimited stability domains making them unconditionally stable [30, 3]. For all implicit time stepping methods, including IRK, there exist many types of stability. It is usually sufficient to use A-stable schemes for many stiff problems, but for more complicated PDEs stricter stability requirements may be needed such as L-stable schemes to better attenuate any numerical noise introduced when applying these schemes. In this thesis, when considering IRK schemes, we require they at least be A-stable methods. Both A- and L-stability are discussed in detail in later chapters.

Finite element methods (FEM) [7, 13] are among the most popular spatial discretization schemes used. They depend on defining integral forms of the PDE at hand using the basis set of selected approximation spaces. Depending on whether the approximation space is a subset of the solution space or not, finite-element methods can be categorized into conforming or non-conforming. There exists a wide variety of both types to choose from depending highly on the problem at hand [21, 13]. In PDEs with multiple variables in different solution spaces, mixed finite-element methods are used, where a combination of approximation spaces are needed. Mixed FEM is famously used in problems with a saddle-point structure such as those considered in this thesis. The difficulty in mixed FEM lies in ensuring this combination satisfies the inf-sup stability condition [13, 16]. Usually in incompressible fluid flow problems when the nonconforming discontinuous Galerkin spaces are used, additional stabilization "penalty" terms must be included [9, 12]. More details on the specific formulations of these terms used in this thesis are found in Chapter 4.

Discrete formulations of PDEs are produced using various types of discretization schemes. Because these formulations are an approximation to the continuum problem, there are natural errors in the approximations to solutions. Thus, the chosen discretization scheme(s) must be carefully designed to introduce the least amount of error, resulting in the closest possible approximation. Any one of these schemes, either temporal and spatial, can be categorized by their *order*, which is an upper bound on this error that depends on the step size taken, either temporally or spatially. The higher the order of a discretization scheme, the more accurate the approximation is. For time-dependent PDEs, stable high-order time discretization schemes need to be coupled with stable high-order spatial discretization schemes for even more accurate approximations. We do note that although there is wide knowledge of higher-order FEM discretizations on stationary saddle-point problems [15, 18, 19] and, simultaneously, of higher-order temporal discretizations with lower-order spatial discretization [17], to our knowledge there is little comparable research on the analysis of using higher-order discretization in both space and time.

The systems that are produced from a full discretization process at each time step are usually very large, making the use of direct methods impractical due to their extremely high computational cost. Iterative methods, such as Krylov methods, can be used instead and FGMRES is a common choice for IRK discretized systems due to their nonsymmetry. This method allows the incorporation of a preconditioner that can differ with each application. The main goal of preconditioning is to change a system of equations into an equivalent but easier to solve system, to further accelerate the convergence of any chosen iterative method. Multigrid methods, developed in the late 1960's [8, 26, 6], can be seen as a powerful family of preconditioners. Based on the design of the multigrid preconditioner, one of two general techniques are followed: either block or monolithic preconditioning. When block preconditioning in systems with a RK discretization, a multigrid cycle is applied to each stage separately to approximately solve the linear subsystems. Since the application of the preconditioner happens to each block individually, the system of equations is decoupled. For parabolic PDEs with IRK discretization, many block multigrid preconditioners have been developed [31, 23, 24, 10]. Sometimes, when using these preconditioners, one (or more) complicated Schur complement blocks need to be approximated which is a major disadvantage that arises in many problems with a saddle-point structure, such as fluid flow problems discretized with mixed finite element methods. So, an alternate approach is to preserve the coupling of the system by applying the preconditioner to all stages at once, which is done by monolithic preconditioning. Recent interest has increased in studying these types of solvers [4, 25, 28].

When using preconditioned methods, a clever design of the preconditioner is essential to achieve an overall efficient solver. In monolithic multigrid preconditioners, this includes carefully choosing the correct relaxation scheme. Various families of relaxation methods have been known to give excellent results, including (but not limited to) Braess-Sarazin [5], Uzawa [22] and Vanka [29] schemes. In all our work, we use a Vanka-type relaxation scheme in the monolithic multigrid process following work found in [1, 14, 2, 20, 27]. Unlike in Braess-Sarazin (and by extension Uzawa) where an update is calculated simultaneously for all grid points, Vanka relaxation depends on dividing the domain into patches and performing a local solve on each patch with the update then accumulating these updates sequentially. We note that our work is the first using monolithic multigrid with Vanka relaxation for IRK discretized saddle-point problems.

The work presented in this thesis has contributed in filling a noticeable gap in the current research by successfully developing a solver for stiff PDEs with IRK discretizations. This was done through what can be seen as a three step extension. First, the solver is developed and proven to be efficient for several A-stable families of IRK schemes on four different types of fluid flow problems varying in complexity. Although high-order A- and L-stable IRK schemes were used, only a lower-order conforming mixed FEM was chosen. Secondly, this solver was extended to solve the incompressible Navier-Stokes equations with higher-order non-conforming mixed FEM spaces while using even higher-order L-stable IRK discretizations. Thirdly, modifications were made to the multigrid transfer operators and an elasticity solve was utilized to show our solver is also effective in dealing with domains with curved boundaries in both two and three dimensions. This is shown by applying it to the well-known difficult to approximate Maxwell's equations. We specifically note that our solver has been developed to be optimally parallelizable over several cores and parallel numerical results are included throughout this thesis.

The overview of this thesis is as follows:

In Chapter 2, we introduce background information on mathematical concepts used in the work. Krylov methods, geometric multigrid, and discretization schemes are discussed.

In Chapter 3, a Newton-Krylov-Multigrid solver is developed for several types of incompressible fluid flow models with an implicit Runge-Kutta discretization. Emphasis on the use of Vanka type relaxation is made. In all models, a lower-order mixed spatial discretizations is used. The robustness of the solver is detailed through numerical results and a comparison of the effect of different stability properties and number of stages of a few implicit Runge-Kutta schemes is presented. This work is published as *Monolithic multigrid for implicit Runge-Kutta discretizations of incompressible fluid flow* in the Journal of Computational Physics, 2023.

In Chapter 4, an extension of the solver developed in Chapter 3 is made to higher-order spatial and temporal discretization spaces. The focus of testing of this solver is the incompressible Navier-Stokes equations. Non-conforming mixed finite element spaces are used and the introduction of interior penalty terms is needed to enforce the incompressibility constraint. Although the RadauIIA Runge-Kutta method is exclusively used, the findings can be generalized to other higher-order L-stable implicit Runge-Kutta schemes such as LobattoIIIC.

In Chapter 5, the same solver from Chapter 3 is also extended to Maxwell's equations on a disk and sphere. Special care must be taken in the multigrid process since the hierarchy of meshes is non-nested. The main difficulty arises in accurately approximating the elements near the curved boundary of the domain. An elasticity solve is needed to appropriately handle the curvature.

In Chapter 6, a conclusion of the full thesis is made and topics for future work are proposed.

Chapter 1 References

[1] J. H. Adler, T. Benson, E. C. Cyr, P. E. Farrell, S. MacLachlan, and R. Tuminaro. Monolithic multigrid for magnetohydrodynamics. *SIAM J. Sci. Comput.*, 43(5):S70–S91, 2021.

[2] J. H. Adler, T. R. Benson, E. C. Cyr, S. P. MacLachlan, and R. S. Tuminaro. Monolithic multigrid methods for two-dimensional resistive magnetohydrodynamics. *SIAM Journal on Scientific Computing*, 38(1):B1–B24, 2016.

[3] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. SIAM, 1998.

[4] T. Boonen, J. Van lent, and S. Vandewalle. An algebraic multigrid method for high order time-discretizations of the div-grad and the curl-curl equations. *Applied Numerical Mathematics*, 59(3):507–521, 2009.

[5] D. Braess and R. Sarazin. An efficient smoother for the Stokes problem. *Applied Numerical Mathematics*, 23(1):3–19, 1997.

[6] A. Brandt and O. E. Livne. *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition.* SIAM, 2011.

[7] S. C. Brenner, L. R. Scott, and L. R. Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

[8] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial.* SIAM, 2000.

[9] E. Burman and P. Hansbo. Edge stabilization for the generalized stokes problem: a continuous interior penalty method. *Computer Methods in Applied Mechanics and Engineering*, 195(19-22):2393–2410, 2006.

[10] H. Chen. A splitting preconditioner for the iterative solution of implicit Runge-Kutta and boundary value methods. *BIT*, 54(3):607–621, 2014.

[11] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1):32–74, 1928.

[12] J. Douglas and T. Dupont. Interior penalty procedures for elliptic and parabolic Galerkin methods. In *Computing Methods in Applied Sciences: Second International Symposium December 15–19, 1975*, pages 207–216. Springer, 2008.

[13] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics.* Oxford University Press, USA, 2014.

[14] P. E. Farrell, Y. He, and S. P. MacLachlan. A local Fourier analysis of additive Vanka relaxation for the Stokes equations. *Numerical Linear Algebra with Applications*, 28(3):e2306, 2021.

[15] P. E. Farrell, L. Mitchell, L. R. Scott, and F. Wechsung. A Reynolds-robust preconditioner for the Scott-Vogelius discretization of the stationary incompressible Navier-Stokes equations. *The SMAI Journal of Computational Mathematics*, 7:75–96, 2021.

[16] V. Girault and P.-A. Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms*, volume 5. Springer Science & Business Media, 2012.

[17] J.-L. Guermond and P. Minev. High-order time stepping for the incompressible Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 37(6):A2656–A2681, 2015.

[18] V. John, A. Linke, C. Merdon, M. Neilan, and L. G. Rebholz. On the divergence constraint in mixed finite element methods for incompressible flows. *SIAM review*, 59(3):492–544, 2017.

[19] V. John and G. Matthies. Higher-order finite element discretizations in a benchmark problem for incompressible flows. *International Journal for Numerical Methods in Fluids*, 37(8):885–903, 2001.

[20] V. John and L. Tobiska. Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 33(4):453–473, 2000.

[21] R. C. Kirby, A. Logg, M. E. Rognes, and A. R. Terrel. Common and unusual finite elements. In *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*, pages 95–119. Springer, 2012.

[22] J. Maitre, F. Musy, and P. Nigon. A fast solver for the Stokes equations using multigrid with a Uzawa smoother. In *Advances in Multi-Grid Methods: Proceedings of the conference held in Oberwolfach, December 8 to 13, 1984*, pages 77–83. Springer, 1985.

[23] K.-A. Mardal, T. K. Nilssen, and G. A. Staff. Order-optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDEs. *SIAM Journal on Scientific Computing*, 29(1):361–375, 2007.

[24] M. M. Rana, V. E. Howle, K. Long, A. Meek, and W. Milestone. A new block preconditioner for implicit Runge-Kutta methods for parabolic PDE. *SIAM J. Sci. Comp.*, 43(5):S475–S495, 2021.

[25] E. Rosseel, T. Boonen, and S. Vandewalle. Algebraic multigrid for stationary and time-dependent partial differential equations with stochastic coefficients. *Numer. Linear Algebra Appl.*, 15(2-3):141–163, 2008.

[26] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Elsevier, 2000.

[27] S. Turek. *Efficient solvers for incompressible flow problems: An algorithmic and computational approache*, volume 6. Springer Science & Business Media, 1999.

[28] J. Van Lent and S. Vandewalle. Multigrid methods for implicit Runge–Kutta and boundary value method discretizations of parabolic PDEs. *SIAM Journal on Scientific Computing*, 27(1):67–92, 2005.

[29] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65(1):138–158, 1986.

[30] G. Wanner and E. Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

[31] M. Wathen and C. Greif. A scalable approximate inverse block preconditioner for an incompressible magnetohydrodynamics model problem. *SIAM Journal on Scientific Computing*, 42(1):B57–B79, 2020.

Chapter 2

Background

This chapter presents the mathematical concepts covering the several numerical methods that make up our solver. First, we discuss Newton's method, which is the linearization technique we use on the nonlinear problems considered in this thesis. Then, in section 2.2, we present details on the different methods applied to the resulting linear system. This includes FGMRES, Chebyshev, multigrid and Vanka schemes. Finally, we detail the discretization methods of interest where finite-element spatial discretization is covered in section 2.3 and Runge-Kutta temporal discretization in section 2.4.

2.1 Newton's method

As mentioned earlier, we apply numerical methods to a system of equations that are formed from the discretization of a continuous differential equation. Depending on the original problem, this system can either be linear or nonlinear. If it is nonlinear, we usually transform it into a linear system by using a linearization method. In our work, for all nonlinear models considered, we use Newton's method as the linearization technique.

Start by considering $F(\mathbf{x}) = 0$ to be a nonlinear system of equations, and let $\delta\mathbf{x}^{(0)} := \mathbf{x}^{(1)} - \mathbf{x}^{(0)}$ where $\mathbf{x}^{(0)}$ is an initial guess for the solution. Using Taylor's series with a truncation of all terms of second order or higher, we have the following form:

$$F\left(\mathbf{x}^{(1)}\right) \approx F\left(\mathbf{x}^{(0)}\right) + J\left(\mathbf{x}^{(0)}\right)\delta\mathbf{x}^{(0)} = 0,$$

where $J\left(\mathbf{x}^{(0)}\right)$ is the Jacobian matrix at the current approximation. So by solving the linear system (also called *Newton equations*), $J\left(\mathbf{x}^{(0)}\right)\delta\mathbf{x}^{(0)} = -F\left(\mathbf{x}^{(0)}\right)$ for $\delta\mathbf{x}^{(0)}$, we can find the next approximate Newton guess $\delta\mathbf{x}^{(0)} + \mathbf{x}^{(0)} = \mathbf{x}^{(1)}$. Usually, $\mathbf{x}^{(1)}$ is a more accurate approximation than $\mathbf{x}^{(0)}$, which means that $||\mathbf{x} - \mathbf{x}^{(1)}|| < ||\mathbf{x} - \mathbf{x}^{(0)}||$. If $\mathbf{x}^{(1)}$ does not have the desired accuracy, we apply Newton's iteration again using $\mathbf{x}^{(1)}$ to find $\mathbf{x}^{(2)}$ and so on achieving a set of approximations $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \ldots$ that become continually closer to the exact solution $\mathbf{x}$. In general, we can define Newton's method on $F(\mathbf{x}) = 0$ iteratively with k denoting the current iteration along with an initial guess $\mathbf{x}^{(0)}$ as:

1. Solve $J\left(\mathbf{x}^{(k)}\right)\delta\mathbf{x}^{(k)} = -F\left(\mathbf{x}^{(k)}\right)$ for $\delta\mathbf{x}^{(k)}$.
2. Set $\delta\mathbf{x}^{(k)} + \mathbf{x}^{(k)} = \mathbf{x}^{(k+1)}$.
3. Check accuracy of approximation $\mathbf{x}^{(k+1)}$.

We note that $\delta\mathbf{x}^{(k)}$ is called the *Newton search direction.*

Newton's method is very powerful as it has very rapid convergence (twice as fast as Picard's method for example) making it a commonly used linearization technique. Despite this property, in many practical problems $\mathbf{x}$ is very large (several million entries) thus making finding the solution of the linearized Newton equations at each iteration very expensive. In such cases, instead of solving the linear system exactly using a direct method, we can approximate the solution using a numerical method [40]. This leads to the Newton-iterative method called the *inexact Newton method* [15]. As for any iterative method, we must define an appropriate stopping criterion (also called *stopping tolerance*), denoted by a set of non negative forcing terms $\{\eta_k\}$ (these can be all the same value η). In order for the inexact Newton method to converge, these forcing terms must all be less than 1. Stopping tolerances demonstrate the acceptable amount of error in the current approximation. After each iteration, the error is calculated, if it is smaller than the tolerance then the iterations stop and the desired approximate solution is reached. In inexact Newton's method, instead of using error, we measure how close (or far) we are from the exact solution by finding the *residual* of the system after each iteration. Through good choices of the forcing terms, the level of accuracy of Newton's method is maintained [15]. In inexact Newton's method, the new update approximation is defined as previously but a residual, $\mathbf{r}^k$, of $\delta\mathbf{x}^{(k)}$ is introduced in the linear system at each iteration:

$$J\left(\mathbf{x}^{(k)}\right)\delta\mathbf{x}^{(k)} + \mathbf{r}^{(k)} = -F\left(\mathbf{x}^{(k)}\right), \text{ where } \frac{||\mathbf{r}^{(k)}||}{||F\left(\mathbf{x}^{(k)}\right)||} \leq \eta_k.$$

In our solver, we apply inexact Newton's method by using a combination of FGMRES and multigrid (detailed in the next section) as the iterative method to approximate the solution to the linearized system.

Since the inexact Newton's process itself is iterative, carefully chosen stopping tolerance(s) must be set to know when convergence is reached. We emphasize that although there are existing techniques to define this tolerance, there is no universal theory that specifies the correct value(s) to set for all problems. In general, the stopping tolerance is decided based on the individual problem at hand and the chosen iterative method. In all problems considered in this thesis, we either set the values of $\{\eta_k\}$ to be (equal) fixed nonlinear tolerances chosen by experimental hand-tuning, or set them to be dependant on the physical mesh discretization size, or we use the Eisenstat-Walker stopping tolerance [18]. Eisenstat-Walker provides two choices for selecting the forcing term set where, in both choices, finding each η_k for $k = 1, 2\dots$ depends heavily on $F\left(\mathbf{x}^{(k)}\right)$ and $F\left(\mathbf{x}^{(k-1)}\right)$ with a given starting term $\eta_0 \in [0, 1)$. These are: for $k = 1, 2, \dots$

- choose either

$$\eta_k = \frac{||F\left(\mathbf{x}^{(k)}\right) - F\left(\mathbf{x}^{(k-1)}\right) - J\left(\mathbf{x}^{(k-1)}\right)\delta\mathbf{x}^{(k)}||}{||F\left(\mathbf{x}^{(k-1)}\right)||}$$

  or

$$\eta_k = \frac{|||F\left(\mathbf{x}^{(k)}\right)|| - ||F\left(\mathbf{x}^{(k-1)}\right) + J\left(\mathbf{x}^{(k-1)}\right)\delta\mathbf{x}^{(k)}|||}{||F\left(\mathbf{x}^{(k-1)}\right)||}$$

  .

- for $\gamma \in (0, 1]$ and $\omega \in (1, 2]$, choose $\eta_k = \gamma\left(\frac{||F\left(\mathbf{x}^{(k)}\right)||}{||F\left(\mathbf{x}^{(k-1)}\right)||}\right)^{\omega}$.

For each choice, a certain lower threshold criteria is set for all forcing terms. This is done to avoid oversolving the nonlinear system meaning produced approximations after a certain iteration are barely moving closer to exact solution, which happens when the forcing term is too small.

2.2 Iterative methods and multigrid

2.2.1 Iterative methods

After applying (inexact) Newton's method, we are left with a linear system of equations to solve numerically. We simplify the notation in this section for ease of discussion, but we note that, in the following, the system matrix A is the Jacobian matrix from above, $\mathbf{u}$ is the unknown Newton direction and $\mathbf{f}$ represents $-F\left(\mathbf{x}^{(k)}\right)$. Thus, the linear of system we consider is written as $A\mathbf{u} = \mathbf{f}$ with

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & \dots & a_{1n} \\ a_{21} & a_{22} & \ddots & & a_{2n} \\ \vdots & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & \dots & a_{nn} \end{bmatrix}, \ \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{bmatrix}, \ \text{and } \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix}. \tag{2.1}$$

Applying a direct method like Gauss elimination to the above system when A is dense has a computational cost $\mathcal{O}(n^3)$ where n is the dimension of the matrices in (2.1). Even though this cost is reduced for problems with a sparse A matrix, the cost will never be as good as $\mathcal{O}(n)$. Unlike direct methods, the number of operations involved in finding a solution of the system using iterative methods is not fixed before hand. Usually, the goal is to keep iterating until a reasonable approximation of the exact solution is reached. When that happens, we say a numerical method has *converged.* Measuring convergence depends on error introduced in the approximate solution from the numerical method used. After each iteration the method takes, we can compute the error in the updated approximation by calculating how far it is from the exact solution. After some iterations of a convergent iterative method, the error decreases to an acceptable amount.

Denoting $\hat{\mathbf{u}}$ to be an approximation to the exact solution $\mathbf{u}$, iterative methods find

$\hat{\mathbf{u}}$ using a series of updates, called iterations or sweeps, on an initial guess. Keeping in mind that our goal is to find an accurate approximation, the error in $\hat{\mathbf{u}}$ is $\mathbf{e} = \mathbf{u} - \hat{\mathbf{u}}$. Often, $\mathbf{u}$ is unknown and so it is hard to find the error this way. Instead, we can use the residual $\mathbf{r} = \mathbf{f} - A\hat{\mathbf{u}}$ as a proxy to measure the size of the error. This leads to an important relationship between residual and error described in the *residual equation*

$$\mathbf{r} = A\mathbf{u} - A\hat{\mathbf{u}} = A\mathbf{e},$$

which now we can use to understand the error.

One class of iterative methods is Richardson iterations, with

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \omega_{k+1}\mathbf{r}^{(k)},$$

where $\mathbf{u}^{(k)}$ is the approximation from the current iteration, $\mathbf{u}^{(k+1)}$ is the approximation in the next iteration, and ω_{k+1} is the scalar weight of the $(k+1)^{th}$ iteration. Recursively, we can write this as

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(0)} + \sum_{i=1}^{k+1} c_i A^i \mathbf{r}^{(0)},$$

where $\mathbf{r}^{(0)}$ is the residual in the initial guess and the set $\{c_i\}$ are all constants depending on the iterative method. (Note: Richardson method is not used in this thesis and is just mentioned here purely for concept introduction.)

There is a huge variety in iterative methods. The effectiveness of a certain iterative method depends highly on the problem at hand to be solved. One of the main classes is called *stationary iterative methods.* In these methods, the matrix A is split into $A = M - N$ making the linear system of equations equivalent to $M\mathbf{u} = N\mathbf{u} + \mathbf{f}$. Iteratively, it can be written in the form

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + M^{-1}\left(\mathbf{f} - A\mathbf{u}^{(k)}\right),$$

where the matrix M^{-1} is called the *preconditioner.* Two common examples of stationary iterative methods are Jacobi and Gauss-Seidel. Considering the common splitting $A = D - L - U$, where D is the diagonal matrix of A and L and U are strictly lower and upper triangular matrices respectively, then in Jacobi $M = D$ making the iteration of the form $\mathbf{u}^{(k+1)} = \mathbf{u}^k + D^{-1}\left(\mathbf{f} - A\mathbf{u}^k\right)$ while in Gauss-Seidel $M = D - L$ making the iteration $\mathbf{u}^{(k+1)} = \mathbf{u}^k + (D-L)^{-1}\left(\mathbf{f} - A\mathbf{u}^k\right)$. While these two methods are very useful building blocks, for many of the more complicated differential equations they are highly unlikely to be used on their own since they tend to be slower to converge compared to other iterative methods and they can become very expensive as the problem size grows, having computational cost at least $\mathcal{O}(n)$ per iteration accumulating to a total cost of at least $\mathcal{O}(n^2)$ over all iterations performed. However they are still used in combination with other iterative methods such as, mentioned in the next section, in multigrid methods.

Alternatively, a widely popular class of non-stationary iterative methods are called *Krylov methods*. These are constructed using a *Krylov subspace*. Given a residual vector $\mathbf{r}^{(0)}$ and a matrix A, the k-dimensional Krylov subspace is

$$K_k(A, \mathbf{r}^{(0)}) = \text{span}\{\mathbf{r}^{(0)}, A^1\mathbf{r}^{(0)}, A^2\mathbf{r}^{(0)}, \ldots, A^{k-1}\mathbf{r}^{(0)}\}.$$

Based on the system matrix properties, there exists a variety of Krylov techniques that aim to find the best possible approximation of the form $\mathbf{u}^{(k)} - \mathbf{u}^{(0)} \in K_k(A, \mathbf{r}^{(0)})$, where $\mathbf{u}^{(k)} - \mathbf{u}^{(0)} \in \mathbb{R}^k$. They differ based on the optimality condition an approximation of this form must satisfy. Two of these methods are the generalized minimum residual method (GMRES) and Chebyshev iteration [37, 19]. We present a detailed explanation of these next.

GMRES method

In GMRES, of all possible $\mathbf{u}^{(k)}$ solutions of the form $\mathbf{u}^{(k)} - \mathbf{u}^{(0)}$, the vector must also minimize $||\mathbf{f} - A\mathbf{u}^{(k)}||$. Consider the matrix $R_k = [\mathbf{r}^{(0)}, A^1\mathbf{r}^{(0)}, A^2\mathbf{r}^{(0)}, \ldots, A^{k-1}\mathbf{r}^{(0)}]$, then we can write

$$\mathbf{u}^{(k)} = \mathbf{u}^{(0)} + R_k\mathbf{y}^{(k)},$$

where $\mathbf{y}^{(k)} \in \mathbb{R}^k$. Since $\mathbf{f} - A\mathbf{u}^{(k)} = \mathbf{f} - A(\mathbf{u}^{(0)} + R_k\mathbf{y}^{(k)}) = \mathbf{r}^{(0)} - AR_k\mathbf{y}^{(k)}$, then finding $\mathbf{u}^{(k)}$ that minimizes $||\mathbf{f} - A\mathbf{u}^k||$ is equivalent to finding $\mathbf{y}^{(k)}$ that minimizes $||\mathbf{r}^{(0)} - AR_k\mathbf{y}^{(k)}||$. In other words, we solve for $\mathbf{y}^{(k)}$ in the following least-squares problem

$$\min_{\mathbf{u}^{(k)} - \mathbf{u}^{(0)} \in K_k(A, \mathbf{r}^{(0)})} ||f - A\mathbf{u}^{(k)}|| = \min_{\mathbf{y}^{(k)}} ||\mathbf{r}^{(0)} - AR_k\mathbf{y}^{(k)}||. \tag{2.2}$$

Notice that $\mathbf{y}^{(k)}$ is a solution to the following normal equations

$$R_k^T A^T A R_k \mathbf{y}^{(k)} = R_k^T A^T \mathbf{r}^{(0)}.$$

In some systems, A might be ill-conditioned, so too is $A^T A$, making it difficult to solve these equations. Additionally, if A is very large, as is the usual case, then solving these normal equations for each iteration can be very costly. Hence, the formulation of these normal equations are usually avoided by using the Arnoldi algorithm. This algorithm produces a set of orthonormal vectors, denoted as $\{\mathbf{q}_1, \mathbf{q}_2, \dots\}$, that is used in the above least-squares problem.

With induction, it can be easily proven that given $\mathbf{q}_1 = \frac{\mathbf{r}^{(0)}}{||\mathbf{r}^{(0)}||}$, then

$$\text{span}\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{k+1}\} = \text{span}\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, A^2\mathbf{r}^{(0)}, \dots, A^k\mathbf{r}^{(0)}\}.$$

Meaning the Arnoldi vector set forms a basis for $K_k(A, \mathbf{r}^{(0)})$. Next, we denote the $n \times k$ matrix $Q_k = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k]$, and the upper Hessenberg matrix of size $(k+1) \times k$ with entries h_{ij} as $H_{k+1,k}$. A general upper Hessenberg matrix, $(H_{l,m})_{ij}$, with dimension $l \times m$ contains entries

$$(H_l)_{ij} = \begin{cases} h_{ij}, \ j = 1, 2 \dots, l, \ i = 1, 2, \dots, \min(j+1, l) \\ 0, \ \text{otherwise}. \end{cases}$$

Now, after the k^{th} iteration in the Arnoldi algorithm we reach the following matrix equation

$$AQ_k = Q_{k+1} H_{k+1,k}. \tag{2.3}$$

Now, if we use the Q_k as the basis for our Krylov space in place of R_k, we can write

$$\mathbf{u}^{(k)} = \mathbf{u}^{(0)} + Q_k \mathbf{y}^{(k)}.$$

So, using (2.3), the minimization problem (2.2) now becomes

$$\min_{\mathbf{u}^{(k)} - \mathbf{u}^{(0)} \in K_k(A, \mathbf{r}^{(0)})} ||\mathbf{f} - A\mathbf{u}^{(k)}|| = \min_{\mathbf{y}^{(k)}} ||Q_{k+1}^T \mathbf{r}^{(0)} - H_k \mathbf{y}^{(k)}||.$$

Due to the definition $\mathbf{q}_1 = \frac{\mathbf{r}^{(0)}}{||\mathbf{r}^{(0)}||}$ and the orthonormalty of the columns of Q_k, we have $Q_{k+1}^T \mathbf{r}^{(0)} = \beta \mathbf{e}_1^{(k+1)}$, where β is a constant and $\mathbf{e}_1^{(k+1)}$ is a column vector of size

$k+1$ with only the first entry being 1 and the rest are 0. Additionally, we can make use of the QR factorization of the Hessenberg matrix, which is $H_k = U_k P_k$ where U_k is a $(k+1)\times(k+1)$ orthogonal matrix and P_k is a $(k+1)\times k$ upper triangular matrix. Finally, these definitions all lead to the minimization problem

$$\min_{\mathbf{u}^{(k)}-\mathbf{u}^{(0)}\in K_k(A,\mathbf{r}^{(0)})}||\mathbf{f}-A\mathbf{u}^{(k)}|| = \min_{\mathbf{y}^{(k)}}||\beta U_k^T\mathbf{e}_1^{(k+1)} - P_k\mathbf{y}^{(k)}||, \tag{2.4}$$

where $P_k\mathbf{y}^{(k)}$ is a $(k+1)\times k$ column with a zero in the last entry. Picking the first k entries of $\mathbf{y}^{(k)}$ such that $P_k\mathbf{y}^{(k)}$ matches $\beta U_k^T\mathbf{e}_1^{(k+1)}$ results in $\min_{\mathbf{y}^{(k)}}||\beta U_k^T\mathbf{e}_1^{(k+1)} - P_k\mathbf{y}^{(k)}|| = |\nu_{k+1}|$ where ν_{k+1} is the last entry of $\beta U_k^T\mathbf{e}_1^{(k+1)}$. The GMRES method continues to iterate until $|\nu_{k+1}|$ satisfies a specified stopping tolerance. We note that the matrix-vector products in the Arnoldi algorithm contribute to the (dominating) computational cost of GMRES after k iterations to be $\mathcal{O}(k^2n)$ which is cheaper than classical iterative methods, such as Jacobi and Gauss-Seidel both with complexity $\mathcal{O}(n^2)$, since in practice the GMRES converges in a number of iterations that can be much smaller than the dimensions of the problem (i.e $k \ll n$). In our work, we specifically use a class of preconditioned GMRES method called flexible GMRES (FGMRES) as the outer Krylov method in the solver where the preconditioner used is a multigrid cycle. In FGMRES, the details presented for GMRES above are the same but applied to a right preconditioned system $AM^{-1}M\mathbf{u} = \mathbf{f}$. Notice using right preconditioning does not change the residual is the Krylov space definition. The "flexibilty" in FGMRES refers to the ability of changing M after each iteration of the method depending on how accurate the approximation is after each iteration. If M is fixed to be the same for all iterations in FGMRES, then right-preconditioned GMRES and FGMRES are mathematically equivalent. However, they differ in their implementation. Where classical right-preconditioned GMRES preforms an extra application of the preconditioner at the end, FGMRES instead stores the set of matrix-vector products in the Arnoldi process. Since we are not restricted by memory in the work done in this thesis and to avoid an extra application of our expensive preconditioner, we prefer FGMRES as our outer Krylov method over right-preconditioned GMRES.

Chebyshev iteration

Let us revisit the iterative method in the form $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \omega_{k+1}\mathbf{r}^{(k)}$. If we assume that the system matrix A is diagonalizable with n eigenvalues denoted as λ_i in $A\mathbf{v}_i =$

$\lambda_i \mathbf{v}_i$, then we can write the error after the k^{th} iteration as

$$\mathbf{e}^{(k)} = \sum_{i=1}^{n} c_i p_k(\lambda_i)\mathbf{v}_i.$$

In the above, the polynomial is defined as $p_k(\lambda_i) = \prod_{j=1}^{k}(1 - \omega_j \lambda_i)$ and the coefficients c_i are in the expansion of the initial error $\mathbf{e}^{(0)} = \sum_{i=1}^{n} c_i \mathbf{v}_i$. This defines *polynomial methods.*

Since our goal in any numerical method is to minimize the error after the iteration, here this means to minimize the maximum of the polynomial on the eigenvalues,

$$||\mathbf{e}^{(k)}|| \leq (\max_{1 \leq i \leq n} |p_k(\lambda_i)|) \sum_{i=1}^{n} |c_i| ||\mathbf{v}_i||.$$

To achieve this, Chebyshev polynomials are an excellent choice [1].

The Chebyshev iteration requires some knowledge of the spectrum of the matrix A beforehand. We begin by defining the ellipse with the center θ, foci points at $\theta \pm \gamma$, a small semi axis length b, and a large semi axis length a. To apply the Chebyshev iteration, this ellipse must exclude the origin while completely including the spectrum of A. We note that if A is symmetric and positive definite, then the length $b = 0$ and hence the spectrum is included inside the interval $[\theta - \gamma, \theta + \gamma]$. Chebyshev polynomials, denoted as $T_k(z)$, were originally derived on the interval $[-1, 1]$, but can be extended by translating these polynomials to $[\theta - \gamma, \theta + \gamma]$ and rescaling them such that $T_k(0) = 1$ [24, 26]. On the interval $|z| \leq 1$ we define $T_k(z) = \cos(k \arccos(z))$, while if z is outside $[-1, 1]$ they are defined as $T_k(z) = \cosh(k \text{ arccosh}(z))$. This results in the scaled translated polynomials

$$p_k(z) = \frac{T_k(\frac{\theta - z}{\gamma})}{T_k(\frac{\theta}{\gamma})}.$$

We notice that as the vertices of the ellipse get closer, the smaller $\max_z |p_k(z)|$ becomes. This means that for the best Chebyshev iteration performance in the SPD case we can choose $\theta - \gamma = \lambda_{min}(A)$ and $\theta + \gamma = \lambda_{max}(A)$. Note this is not necessarily the best bounds for our usage of Chebyshev iterations within multigrid methods; to our knowledge there is no concrete theory on a best choice that fits a certain class of linear systems in this setting. Thus, in this thesis, we either use experimentally hand-tuned (interval) bounds or default chosen bounds set by the software used for iterative solvers (PETSc).

From the discussion above, the knowledge of the spectrum bounds beforehand requires the number of iterations be fixed beforehand as well since the weights $\{\omega_j\}_{j=1}^{k}$ defined in $p_k(\lambda_i)$ are different from those in $p_{k+1}(\lambda_i)$ [1]. So the classical two-term Chebyshev recursive algorithm is replaced by a three-term algorithm. The detailed step-by-step derivation of the three-term recursion version of Chebyshev iteration can be found in [25].

In our solver, Chebyshev polynomials are used to accelerate the convergence of the relaxation scheme in the multigrid preconditioner (discussed in the next section).

2.2.2 Geometric multigrid

As mentioned earlier, one of the most powerful families of numerical schemes for the solution of linear systems are the multigrid methods. Following the naming of these methods, multiple levels of division of the domain are used to approximate the solution of a differential equation accurately. Because of their high efficiency, they are one of the fastest iterative methods developed to this day for a wide range of discretized elliptic (and other) differential equations. Depending on the technique followed in defining the levels in the multigrid method, there are several classifications. In all the work presented here, we focus on using geometric multigrid methods (GMG), where the geometric information of the problem at hand is used in defining the multigrid components. Other classes not discussed in this thesis include algebraic multigrid (AMG) [6], "black-box" multigrid [16], and adaptive multigrid methods [5].

Any error can be described as a combination of waves with varying oscillations, typically categorized into high frequency (called *oscillatory*) components and low frequency (called *smooth*) components. Applying Jacobi or Gauss-Seidel to approximations with such error vectors, we notice that after several iterations the oscillatory components are nearly eliminated (also referred in the literature as "dampened" or "relaxed") while the smooth components are not significantly affected. This behaviour is called the *smoothing property*. Both Jacobi and Gauss-Seidel effectively damp oscillatory components of the error after only a small number of iterations making them excellent choices as smoothing schemes in multigrid methods. These iterations are called *relaxation sweeps.*

For convenience of presenting details of a multigrid method, we will consider a simple one-dimensional Poisson test problem:

$$\begin{aligned} -u^{''}(x) &= f, && \text{in } \Omega = (0,1), \\ u(0) &= u(1) = 0. \end{aligned} \tag{2.5}$$

Multigrid methods are applied to the discrete formulation of (2.5), so for simplicity we discretize the above using the finite-difference discretization method. To do this, we divide the domain Ω into the finite set of $n+1$ spatial points $\{x_i\}_{i=0}^{n}$ such that $0 = x_0 < x_1 < \cdots < x_{n-1} < x_n = 1$. In the simplest case, this leads to the uniform mesh (or grid) with *mesh size* $h = \frac{1}{n}$. Notice that the grid points are $x_i = ih$. Denoting the approximation as $u_i \approx u(x_i)$ and $f_i = f(x_i)$, the (central) finite-difference approximation of (2.5) is

$$\begin{aligned} \frac{1}{h^2}\Big(-u_{i+1} + 2u_i - u_{i-1}\Big) &= f_i, && \text{for } 1 \le i \le n-1 \\ u_0 = u_n &= 0. \end{aligned} \tag{2.6}$$

This leads to the system of equations $Au = f$ of the form (2.1), where

$$A = \begin{bmatrix} \frac{2}{h^2} & \frac{-1}{h^2} & 0 & & \dots & 0 \\ \frac{-1}{h^2} & \frac{2}{h^2} & \frac{-1}{h^2} & 0 & \dots & 0 \\ \vdots & \vdots & & & & \vdots \\ & & & & & \vdots \\ 0 & \dots & \dots & 0 & \frac{-1}{h^2} & \frac{2}{h^2} \end{bmatrix}$$

Since the boundary conditions at the endpoints, $u_0 = u_n = 0$, are implicitly enforced here, A is an $(n-1) \times (n-1)$ matrix. **Remark**: since we do not use finite-difference discretization in any of our research in this thesis, and it is only included for explanation purposes here, we will not include the details of this discretization scheme. For details on finite-difference discretization see [23, 47].

Denoting ω as a real weight constant, a weighted Jacobi iteration for example on the system of equations above is $u^{(k+1)} = u^{(k)} + \omega D^{-1}\left(f - Au^{(k)}\right)$. Using the definition of the error after each iteration and the residual equation, we can write the error formula for each Jacobi iteration as $e^{(k+1)} = e^{(k)} - \omega D^{-1} A e^{(k)}$. As stated, after a few Jacobi iterations, the oscillatory components of the error are dampened leaving only smooth components. Since no amount of further iterations will effectively reduce the smooth errors, we rarely use Jacobi (or Gauss-Seidel) alone. Using multiple levels of grids as in the multigrid process can effectively dampen these components.

Two important observations arise: smooth errors can be represented well without significant loss of information on a coarser grid, and low-frequency error components become high-frequency on a coarser grid which can be further dampened by a smoothing scheme. These are the main ideas of multigrid methods. We define a coarser uniform mesh, with a mesh size H, referred to as the coarse mesh with the original h-sized mesh as the fine mesh. The coarse mesh is constructed using a coarsening factor where the intervals in each spatial direction of the fine mesh is multiplied by this factor. Typically, the coarsening factor is either 2 or 3 (most popularly 2 making $H = 2h$). See Figure 2.1 for an example of the one dimensional mesh coarsening. **Note**: we introduce subscripts in the remainder of this section to indicate which mesh we are referring to.

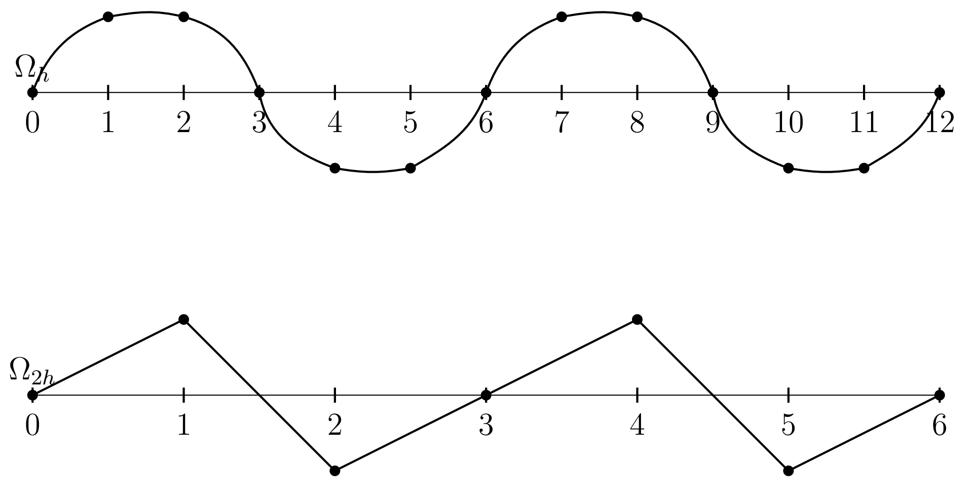


Figure 2.1: An error appearing smooth on a 13 grid points fine mesh and oscillatory on coarser meshes of 7 grid points.

Now that we can easily represent errors on a coarser grid, we can use this to increase the accuracy of the approximation on the fine grid, in other words "correcting" the approximation. This defines the second main concept of multigrid methods known as coarse-grid correction (CGC). Since in most problems of interest the exact solution is unknown and hence the error cannot be directly computed, we use the residual equation in the CGC process. Meaning instead of smoothing $A_H u_H = f_H$ on the coarse grid, we substitute the residual from the fine grid in place of f_H, where $A_H u_H = f_H$ is the approximation of the discrete system on the coarse grid. So, we would be applying Jacobi for example as a relaxation method to the system $A_H u_H = r_H$, where r_H is the residual in $A_h u_h = f_h$ represented on the coarse grid.

A natural question that comes to mind is how do the meshes communicate with each other to transfer the different components between them? For this, we use transfer operators defined as restriction and interpolation. Restriction, denoted as I_h^H, transfers information (usually vectors) from the fine grid to the coarse grid, while interpolation, denoted as I_H^h, transfers from the coarse grid to the fine grid. There are several types of transfer operators to choose from depending on the discretization method used in the problem. A usual choice is to set interpolation as the (scaled) transpose of restriction. So using the meshes in Figure 2.1 with $H = 2h$, no matter what the transfer operator choices are we can say

$$\text{Restriction: } I_h^{2h} v^h = v^{2h} \text{ and Interpolation: } I_{2h}^h v^{2h} = v^h,$$

where v^{2h} and v^h are vectors on Ω^{2h} and Ω^h, respectively.

The last thing left to figure out is how do we define the coarse-grid operator A_H. There are two methods for this: either rediscretizing the continuum problem on the coarse mesh or using the Galerkin operator, [27], where $A_H = I_h^H A_h I_H^h$. For some discretizations of differential equations, the two methods are equivalent (as the case for most of the problems considered in this thesis), for others they differ and hence rediscretization is usually used (as the case for the problems considered in Chapter 4).

Now that the different components that make up multigrid are defined, the two-grid multigrid algorithm for the system $A\mathbf{u} = \mathbf{f}$ is as follows:

Two-grid algorithm: $TG(A_h, f_h, u_h^k, \nu_1, \nu_2) \to u_h^{k+1}$:

1. Pre-smoothing: Relax ν_1 times on $A_h u_h = f_h$ to get the approximation $\hat{u}_h^k$.

2. CGC process:

    (a) Find the residual on the fine grid: $r_h = f_h - A_h \hat{u}_h^k$.

    (b) Restrict the residual to the coarse grid: $r_H = I_h^H r_h$.

    (c) Solve the coarse-grid problem $A_H u_H = r_H$ directly.

    (d) Interpolate the solution to the fine grid: $e_h = I_H^h u_H$.

    (e) Correct the approximation with the interpolation: $\hat{u}_h^k + e_h \to \hat{u}_h^k$.

3. Post-smoothing: Relax ν_2 times on $A_h \hat{u}_h^k = f_h$ to get the approximation u_h^{k+1}.

In the algorithm above, ν_1 and ν_2 are the number of sweeps of the pre- and post-smoothing schemes, respectively. These can differ but are usually the same, and in general no more than 3 sweeps are used. We note that the pre- and post-smoothing schemes themselves can also be chosen to be different but almost always are the same. The study of Local Fourier analysis (LFA), [42], can often be used to find the best parameters included in all components of a multigrid method. This indicates that due to the many components involved in multigrid, optimization of the scheme is always possible depending on the problem at hand. For example, for some problems, Jacobi or Gauss-Seidel will need to be replaced by better relaxation techniques as will be discussed in the next subsection.

To study the rate of convergence of multigrid methods in general (the two-grid method in particular) we can use the two-grid error propagation operator

$$MG = S_h^{\nu_2}(I - I_H^h A_H^{-1} I_h^H A_h) S_h^{\nu_1},$$

where S_h and I are the relaxation operator on the fine mesh and coarse mesh identity matrix, respectively.

Multigrid methods have optimal complexity in comparison to other stationary iterative solvers, meaning the number of iterations required to solve a problem with n degrees of freedom is proportional to n. In other words, their total computational cost is $\mathcal{O}(n)$ making them cheaper than unpreconditioned GMRES. This makes multigrid methods faster in convergence than other iterative methods (of course this is true only if all the different components described earlier are chosen optimally). Multigrid methods are also scalable in parallel settings, which is a very important property in the modern world to take advantage of the hardware architecture in solving huge problems.

A crucial advantage in the optimality of the algorithm described is that the coarse-grid system is much cheaper to solve than the fine-grid system since it contains fewer grid points hence fewer unknowns to solve for. However, in almost all realistic models the number of coarse-grid points is still very large. So, we can further coarsen the coarse grid, building a hierarchy of meshes, until we get to a coarse enough grid that the coarsest system can be cheaply solved using a direct method, along the way, recursively applying the two-grid algorithm. This is why they are referred as the **multigrid** methods.

Vanka relaxation

The type of relaxation method chosen is usually based on the problem at hand and the discretization method used. The choice of an appropriate relaxation method is extremely important in increasing the efficiency of multigrid methods. As mentioned in the previous chapter, in our solver we use monolithic multigrid, as opposed to block preconditioning, when dealing with saddle-point problems. In these problems, it is common to use more intricate smoothing techniques than classical point-wise relaxation schemes such as Jacobi or Gauss-Seidel extensions that due to the structure of A can no longer be applied. The three relaxation methods most commonly used are either Uzawa [32], Braess-Sarazin [4], or Vanka [44]. We strictly use Vanka smoothing in this thesis for both the pre- and post-relaxation schemes in the monolithic multigrid cycle.

The Vanka scheme was famously first introduced as a relaxation method on the Navier-Stokes equation with a Marker-and-Cell (MAC) finite difference discretization in [44] during the 1980's. This scheme is essentially an overlapping Schwarz iterative method, which is a domain decomposition scheme. Originally introduced as multiplicative schemes, Vanka relaxation can also be applied additively. When parallel computation is used, additive Vanka relaxation is preferred [46].

Unlike in Uzawa and Braess-Sarazin where updates in each iteration happens to all grid points at once, the main concept of Vanka relaxation is the decomposition of all DoFs in the discretized mesh into blocks (also called *patches* in the literature [22]). In Vanka, updates to the global approximation are done by restricting to each individual Vanka block the residual, then performing a local solve, followed by adding the interpolated solutions globally. The goal is to set up the local Vanka block systems to have the same saddle-point structure as the global system. The size of the Vanka blocks highly depends on the discretization used (for example as seen in the blocks used in Chapter 4 compared to those used in Chapter 3). For incompressible fluid-flow problems, in order to enforce the incompressibility constraint it is necessary to include only one DoF of the constraint variable in each patch [31]. All other DoFs adjacent to this constraint DoF are included also in the Vanka block. No matter the type/shape of the Vanka blocks, each DoF in the mesh must be included in at least one block (but can appear in multiple blocks). The details of the Vanka iterations are discussed more deeply in the later chapters.

2.3 Finite-element method

To formulate the linear $A\mathbf{u} = \mathbf{f}$, a discretization method is first applied that transforms the continuous problem to a discrete one. For spatial variables, there are many discretization methods available to choose from, but one of the most popular and commonly used is the finite-element method (FEM). The key concept for this method is that it relies on integrals of functions that one can evaluate on arbitrarily shaped domains. FEM is usually favoured because it is more flexible in the choice of the discretized mesh structure which is crucial in problems with irregularly shaped domains. It is also generally easier to develop higher order approximations using FEM than finite difference methods. An additional advantage to using FEM is they are easier to implement in software due to the use of reference mapping in the variational formulation.

To introduce the concept of FEM, consider the d-dimensional Poisson equation, where $d \in \{2, 3\}$:

$$\begin{aligned} -\Delta u &= f, \quad \text{in } \Omega, \\ u &= 0, \quad \text{on } \partial\Omega, \end{aligned}$$

where u is the sufficiently smooth unknown solution in a certain space X, called the *solution space*. We note that, due to the Dirichlet condition above, all functions included in X vanish on the boundary. The above elliptic DE is sometimes called the *strong formulation*. The discrete form of this is based on the *weak formulation* (or the variational formulation).

To find the weak formulation, we start off by multiplying both sides of the above equation with a *test* function, $v \in X$, then integrating over the domain:

$$-\int_{\Omega} v\Delta u dV = \int_{\Omega} vf dV, \ \forall v \in X.$$

Next, we break down the integral on the left hand side by using integration by parts to get

$$\begin{aligned}
-\int_{\Omega} v \cdot \Delta u dV &= -\left(\int_{\partial\Omega} v\nabla u \cdot \hat{n} dA - \int_{\Omega} \nabla v \cdot \nabla u dV\right), \\
&= \int_{\Omega} \nabla v \cdot \nabla u dV - \int_{\partial\Omega} v\nabla u \cdot \hat{n} dA, \\
&= \int_{\Omega} \nabla v \cdot \nabla u dV,
\end{aligned}$$

since the boundary conditions imply $v = 0$ on $\partial\Omega$. This reduces the smoothness required of u. The weak formulation of the Poisson equation becomes: Find $u \in X$ such that

$$\int_{\Omega} \nabla v \cdot \nabla u dV = \int_{\Omega} fv dV, \ \forall v \in X.$$

Before we continue, we define two spaces to set up the remaining discussion. The space of squared integrable functions on the domain is defined as

$$L^2(\Omega) := \left\{ u \middle| \int_{\Omega} u^2 < \infty \right\},$$

with a norm of $||u||_2 := (\int_{\Omega} u^2)^{\frac{1}{2}}$ and an inner product of $\langle f, g \rangle = \int_{\Omega} fg dV$. The first degree Sobolev space for a two-dimensional Ω is denoted as

$$H^1(\Omega) := \left\{ u \middle| u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \in L^2(\Omega) \right\}.$$

We note that the solution of the strong formulation (with the Dirichlet boundary condition) must have continuous second derivatives that are continuous up to the boundary [3]. While solutions of the weak form are only required to be in the H^1 Sobolev space. A sufficiently smooth solution of the weak formulation is also a solution of the strong formulation (for proof see [7]).

Now, we define X_n, to be the finite n-dimensional *approximation space* over the set of basis functions $\{\phi_1, \phi_2, \ldots, \phi_n\}$ satisfying the boundary condition $\phi_i(x) = 0$ $\forall x \in \partial\Omega$. These functions make up a maximal set of linearly independent vectors that span the space X_n. If $u_n \in X_n$, then it can be written as a combination of the basis functions as $u_n = \sum\limits_{i=1}^{n} u_i\phi_i$, where the coefficients $\{u_i\}_{i=1}^{n}$ are the unique set of real unknown coefficient associated with u_n. The set of functions u_n are called *trial functions* (or shape functions) and are the approximations to the weak formulation solution u.

For simplicity, we assume that the basis set of the trial and test function spaces are the same. Hence, we assume the finite-dimensional version of the weak formulation is to find $u_n \in X_n$ such that

$$\int_\Omega \nabla v_n \cdot \nabla u_n dV = \int_\Omega f v_n dV, \ \forall v_n \in X_n. \tag{2.7}$$

If we plug in the expansions $u_n = \sum\limits_{j=1}^{n} u_j\phi_j$ and $v = \sum\limits_{i=1}^{n} v_i\phi_i$, where $\{v_i\}_{i=1}^{n}$ are the coefficient set for v, in (2.7) we get

$$\sum_{j=1}^{n} u_j \int_\Omega \nabla\phi_i \cdot \nabla\phi_j = \int_\Omega \phi_i f.$$

This then can be written as the *Galerkin system* of equations

$$Au_h = f,$$

where $A = [a_{ij}] = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j$ is the stiffness matrix and $f = [f_i] = \int_\Omega \phi_i f$. So, in order to find the solution u_n, we must solve the Galerkin system for u_h then plug it into the expansion of u_n. We note because of our assumption on the trial and test spaces, A is a symmetric and positive definite matrix [20].

What differentiates classes of finite-element methods is the choice of the approximation space X_n. If $X_n \subset X$, then we say X_n is X-*conforming* and we have a conforming FE approximation, otherwise it's called *non-conforming.* In our work, we use both classes of methods, depending on the problem at hand. Whatever the choice of X_n, it must be appropriately made in a way that allows the error in the approximation u_n to decrease as the dimensions of the space increase while maintaining reasonable cost for solving the produced Galerkin system.

We can minimize the cost of solving the Galerkin system by choosing the basis functions to have as small of a local support as possible. Local support refers to the narrowest region on the mesh that $\{\phi_i\}$ may have nonzero values. By taking a small local support basis set we can guarantee that the discretization matrix is sparse, thus cheaper to solve (under some assumptions). This local area is referred to as an element in the discretized mesh and most commonly are intervals in one-dimensional meshes, triangles or rectangles in two-dimensional meshes, or tetrahedra or cubes in three-dimensional meshes. These elements should be small enough that they approximate the domain and solution accurately, but not so small that they increase the cost of finding the approximation.

Generally, smooth functions can be approximated by piecewise polynomials. Commonly in FEM, the basis functions of X_n are *nodal functions*, specifically Lagrangian interpolants. This means that the bases produce a Kronecker property, $\phi_j(\mathbf{x}_i) = \delta_{ij}$, and so $u_n(\mathbf{x}_i) := \sum_{j=1}^{n} u_j\phi_j(\mathbf{x}_i) = u_i$. Let us consider the decomposition of a two-dimensional mesh into a finite number, N, of non-overlapping triangles for example to get a triangulation τ_N of Ω. This results in a total number of N triangular elements. If the solution space considered is $X = H^1(\Omega)$, then the simplest and smallest conforming FE space is the piecewise linear function space $\mathrm{P}_1(\tau_N)$. In this space, on any given element $T \in \tau_N$, there are 3 *nodes*, one on each vertex of T. In each element, the $\mathrm{P}_1(\tau_N)$ space will only have 3 basis functions that are not zero. These are ϕ_1, ϕ_2, ϕ_3 where the element vertices are numbered 1, 2 and 3 respectively in any order, making up the local DoFs related to that element T. Since these are Lagrangian interpolants, the value of ϕ_i is 1 on node i and zero on the other two nodes of T. If we want a more accurate approximation, we can increase the dimensions of the approximation space by adding more approximation nodes in the same triangulation. For example, for the piecewise quadratic function space, $\mathrm{P}_2(\tau_N)$, in addition to the three vertex nodes we also have a node on each edge of T for a total of 6 basis functions on T that are not zero. In general, the set of all nodes in the mesh make up the total set of DoFs. The total number of DoFs in a FE mesh is equal to the number of basis functions of the approximation space X_n. DoFs do not always come in the form of nodal values as described for spaces considered in the following chapters. The particular distribution of the DoFs in an element depends on the approximation space chosen. Since the value of the basis functions at each DoF contributes to finding u_n, the more DoFs used, the more accurate the approximation generally is. We must keep in mind that with the increase of the number of DoFs comes an increase in the size of the Galerkin system, thus increasing the computational cost of finding the approximation. Hence an important trade-off occurs between accuracy and cost of solution. See Figure 2.2 for an example of $\mathrm{P}_1(\tau_N)$ and $\mathrm{P}_2(\tau_N)$ elements on an arbitrary triangulation.

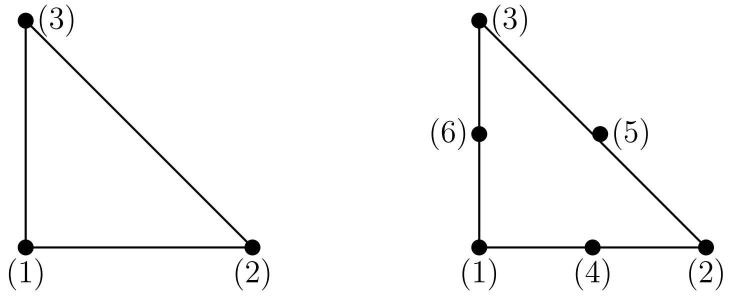


Figure 2.2: Left: P_1 triangular element. Right: P_2 triangular element.

In the Poisson model considered there was only one unknown, u. We note that in most realistic models, we have more unknown variables to solve for with either all or none belonging to the same solution space. In this case we use *mixed finite-elements.* For example, consider the time-independent Stokes equation on the two-dimensional domain Ω with viscosity set to 1 as

$$-\Delta \mathbf{u} + \nabla p = \mathbf{f} \in \Omega \tag{2.8a}$$

$$-\nabla \cdot \mathbf{u} = 0 \in \Omega, \tag{2.8b}$$

$$\mathbf{u} = 0 \in \partial\Omega, \tag{2.8c}$$

where $\mathbf{u}(\mathbf{x})$ is the velocity, and $p(\mathbf{x})$ is the pressure. Let the solution spaces be $\mathbf{u} \in \mathbf{H}_0^1$ and $p \in L_0^2(\Omega)$ where $L_0^2(\Omega) := \left\{u \in L^2(\Omega) \middle| \int_\Omega u dx = 0 \text{ on } \partial\Omega\right\}$. To apply mixed FEM discretization the same general concepts described earlier occur on a chosen decomposition of the mesh, taken to be a triangulation for example. For each of $\mathbf{u}$ and p, an approximation space is chosen defining the basis sets denoted as $V_h \in V$ and $Q_h \in Q$ respectively, where $V := \mathbf{H}_0^1(\Omega)$ and $Q := L_0^2(\Omega)$. Test functions $\mathbf{v}$ and q are defined from V_h and Q_h respectively and DoFs are distributed on the elements. The final Galerkin system is constructed and solved to then find all approximations of the unknowns. The difficulty in using mixed FEM is choosing carefully compatible approximation spaces. In order to avoid instability in the discretization scheme (i.e well-posedness of the saddle-point system), the approximation spaces chosen must satisfy the *inf-sup condition* [20]

$$\inf_{q\neq 0} \sup_{\mathbf{v}\neq 0} \frac{\left|\int q \ \mathrm{div}\mathbf{v}\right|}{||\mathbf{v}|| \ ||q||} \geq \gamma, \tag{2.9}$$

where γ is a strictly non-negative constant. Combinations that satisfy (2.9) are said to be *inf-sup stable*. Not all mixed finite-element spaces can be inf-sup, in fact using spaces of the same degree can never be inf-sup stable. One of the most popular conforming mixed finite-element spaces for the (Navier)-Stokes equations is the Taylor-Hood space where $(\mathbf{u}, p) \in V_h \times Q_h = \mathrm{P}_2 \times \mathrm{P}_1$. Many other conforming and nonconforming combinations with varying degrees exist on various types of meshes. All of the mixed finite-element spaces in our work were carefully chosen to be inf-sup stable.

2.4 Runge-Kutta methods

When spatial discretizaion is applied to a time-dependent PDE, we are left with a semi-discrete system of equations since the time variable is still continuous. This technique is known as the method of lines (MOL) [38, 39]. MOL can be applied to either linear or nonlinear PDEs where all variables are discretized except for one leading to a system of ordinary differential equations (ODEs). An ODE solver is then used to solve the produced system. In time-dependent PDEs specifically, MOL essentially separates the spatial and temporal discretizations since time integration techniques (also *temporal discretizations*) are applied to these system of ODEs. There are also a wide selection of temporal discretization schemes to choose from depending on the problem at hand and the stability requirements [29]. Although for a small number of problems, low-order schemes like the Euler method (which is first order) are used, generally all low-order time integration schemes have poor convergence. Hence, higher-order schemes are favoured. Two popular classes of higher-order schemes are multistep methods and one-step methods. Both types of schemes use information from previous time steps to find the approximation at the current time step, t^n. The difference is in the number of previous time steps needed (many or just one previous). A popular multistep method is the k^{th} order backward difference formula (BDF-k). On the other hand, a powerful family of one-step methods is the set of Runge-Kutta schemes (RK). In all our work, we use the Runge-Kutta methods.

Given the system of linear time-dependant ODEs $u'(t) = f(u(t), t)$, applying an r-stage Runge-Kutta method gives

$$\begin{aligned} k_i &= f\left(u^n + \Delta t \sum_{j=1}^{r} a_{ij} k_j, t^n + c_i \Delta t\right), \text{ for } i = 1, 2, \ldots, r, \\ u^{n+1} &= u^n + \Delta t \sum_{j=1}^{r} b_j k_j. \end{aligned} \tag{2.10}$$

In the scheme, the set $\{k_i\}_{i=1}^{r}$ represents the r stage values and the approximation at time $t^n = t^0 + n\Delta t$ is denoted by u^n. The real coefficients are the nodes c_i, the weights b_j, and the Runge-Kutta matrix $A = [a_{ij}]$. All coefficients form what is called the Butcher tableau [10, 11], where each RK scheme has a unique Butcher tableau structure.

RK methods can be classified based on the non-zero structure of the coefficient matrix A. The two main classifications are explicit Runge-Kutta methods (ERK), where $a_{ij} = 0\ \forall j \geq i$, and implicit Runge-Kutta methods (IRK), where $\exists j \geq i$ with $a_{ij} \neq 0$. IRK methods can further be categorized into diagonally implicit Runge-Kutta methods (DIRK), where $a_{ij} = 0\ \forall j > i$, and fully implicit Runge-Kutta (FIRK). Further, popular types of DIRK schemes are singly diagonally implicit Runge-Kutta methods (SDIRK), which have similar DIRK structure with the added property of $a_{ii} = a_{jj}$ for all i and j, and explicit singly diagonally implicit Runge-Kutta methods (ESDIRK), which in addition to an SDIRK structure have $a_{0j} = 0\ \forall j$. We note BDF-1, 1-stage IRK and Backward (implicit) Euler are all equivalent time integration schemes.

In general, the domain of stability, denoted by D, of a scheme is the set of all complex numbers z, such that $\lim_{n\to\infty} |u^n| \leq |u^0|$. This domain can be found by applying the time integration scheme to the Dahlquist test equation:

$$u' = \lambda u,$$

where $\lambda \in \mathbb{C}$ denoting $z = \Delta t\lambda$. The exact solution to this equation is $u = u_0 e^{\lambda t}$. To avoid the solution blowing up, the real part of the exponent must be negative. In other words if $Re(\lambda t) \leq 0$, then $\lim_{t\to\infty} |u| \leq |u(0)|$. Applying the general RK method Equation (2.10) to the test equation we get the stages

$$k_i = u^n + \Delta t\lambda \sum_{j=1}^{r} a_{ij}k_j.$$

If we denote the $r \times 1$ vectors $\mathbb{1} = [1, \dots, 1]^T$ and $\vec{\mathbf{k}} = [k_1, \dots, k_r]^T$, we have

$$\vec{\mathbf{k}} = (I - \Delta t\lambda A)^{-1}\mathbb{1}u^n,$$

which we can then plug into the second equation of Equation (2.10) to get

$$u^{n+1} = (1 + \Delta t\lambda b^T(I - \Delta t\lambda A)^{-1}\mathbb{1})u^n.$$

In the above, the function $R(z) = 1 + zb^T(I - zA)^{-1}\mathbb{1}$ is called the stability function of the RK scheme. In order for $\lim_{n\to\infty} |u^n| \leq |u^0|$ to be true, $|R(z)|$ must be less than 1. For RK methods, this implies that $D = \{z \in \mathbb{C}||R(z)| \leq 1\}$. Clearly, $R(z)$ is a rational function, however for ERK it is a polynomial of degree r.

The main advantage of one type of time integrator over another of equal order is the stability. There are many kinds of stability one might need for a given PDE (see [45] for details). In this thesis, we focus on two types, A-stability and L-stability. A method is called A-stable when the left-hand side of the complex plane lies completely within the domain of stability of the chosen scheme. If a scheme is A-stable, then the time step size can be chosen only based on accuracy while always guaranteeing stability. Thus, schemes with this stability are ideal for stiff problems. We note that, due to Dahlquist's second barrier, no linear multistep method with order higher than 2 can be A-stable [14], which is not the case for RK methods. This gives RK schemes a potential advantage over BDF methods. No ERK scheme is A-stable, but in fact many IRK schemes are. Since most of the problems considered in this thesis are stiff PDEs, we strictly use RK schemes that are at least A-stable.

For some problems (as clearly shown in Chapter 3), A-stability is not enough and a stronger stability criterion is needed, such as L-stability. If, in addition to A-stability, a scheme satisfies the condition $\lim_{z\to-\infty} |R(z)| = 0$, then it is called L-stable. Throughout this thesis, we focus on 3 main IRK schemes: Gauss-Legendre (commonly referred to as Gauss) [9], RadauIIA [8], and LobattoIIIC [8]. Both RadauIIA and LobattoIIIC are L-stable, while Gauss methods are only A-stable and not L-stable.

Clearly, for any differential equation that has had a discretization scheme applied to it (whether to space or time variables or both) some amount of error is introduced in the approximate solution obtained. In time integration schemes the local and global errors are found after either one time step is taken or accumulated after all time steps needed to reach a fixed final time, respectively. Any type of error in each time integration scheme can be bounded (from above) by a constant times $(\Delta t)^p$, where p denotes the global *order* of the scheme. Since in practice $\Delta t < 1$, the higher the global order of a scheme means the more accurate the approximation is (smaller error produced). In RK methods, the order is connected to the total number of stages, r. There exist ERK methods with $p \leq 4$ that have $p = r$. To achieve ERK schemes of order $p \geq 5$ we must have $r = p + 1$ stages [12]. This becomes a disadvantage in higher-order ERK methods since more stages means an increased computational cost. However, in IRK discretizations the maximum order of the global error achieved can be as twice as the number of stages, such as in the case of Gauss schemes with an order of $2r$. RadauIIA have an order of $2r - 1$ while LobattoIIIC have an order of $2r - 2$. In RK methods for stiff equations of differential-algebraic equations (DAEs), another type of order is of more importance, called the *stage order*, which is found by bounding the approximate solution of the i^{th} stage, i.e $u(t^n + c_i\Delta t)$, by some constant times $(\Delta t)^{q+1}$ then defining the stage order to be $\min\{p, q\}$ [12]. For many types of differential equations, the accuracy is limited by the stage order which limits the choice of higher-order schemes. IRK schemes have high stage orders that can be as large as the number of stages which makes them favourable choices for some problems, like those considered in this thesis, over DIRK methods which have a very limited low stage order.

Although IRK methods possess better convergence properties over other RK classes, especially when considering higher-order schemes, they easily are more computationally expensive. This is because at each timestep, they require solving a system of stage equations (usually large depending on the spatial discretization chosen). Since the stages depend on each other, this dense non-symmetric system does not have significant structural properties that one may take advantage of to reduce the cost of the applied numerical method. This disadvantage is intensified for higher-order IRK schemes where more stages are needed.

Despite this downside, growing interest has been seen for solver development specifically addressing IRK discretizations. Solvers that incorporate preconditioners are a very common technique used. For problems with IRK, several types of preconditioners have been built to handle the stage coupling in various ways. One way is to decouple the stages by using a block preconditioner and then applying an efficient method for each single stage system produced. In [13], expanding on work presented in [33], it was proven that the system with this preconditioner has a condition number with an upper bound not depending on either Δt nor Δx. A novel approach is the work done in [36] where the preconditioner is based on an LDU (lower-diagonal-upper) matrix-splitting of the Butcher coefficient matrix. Another novel example of a block solver is the work done in [41] where the preconditioner used is the same as in the case of backward Euler time-stepping. This solver has been proven to be optimal for both finite-element and finite-difference discretization, linear and nonlinear differential algebraic equations. In both preconditioners mentioned, a mutigrid cycle is used to solve each block.

One alternative to block structured preconditioners, is to solve for the coupled stages all at once. One commonly used monolithic preconditioner choice is multigrid where the relaxation schemes in the multigrid cycle are specifically designed so that the system solves for all components of $\vec{\mathbf{k}}$. For example in [43], waveform relaxation is used to achieve this. Additional examples are [21, 28] where Vanka relaxation is chosen. The solver we present adds to this already existing small body of work.

With the increased size of the IRK stage system, parallel computation is highly needed and used in almost all of the solvers mentioned. One strategy is specifically designing a preconditioner that allows the exploitation of the parallel architecture in a way that each stage is solved fully on an assigned group of processors, thus solving for all stages in parallel. This method is called stage-parallel IRK and is fairly new [35, 34, 17, 2]. Another novel strategy that takes advantage of parallel computing is proposed in [30] where time-stepping is computed in parallel. There, an SVD-based (of the the Runge-Kutta coefficient matrix) monolithic preconditioner is used to find all stage values at once, but parallelism is applied to the time variable.

2.4.1 Butcher tableaux of Runge-Kutta schemes used

For convenience, we include the Butcher Tableaux of all implicit Runge-Kutta schemes mentioned and/or used in our work below:

Two-stage Gauss-Legendre:
$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Three-stage Gauss-Legendre:
$$\begin{array}{c|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$$

Two-stage LobattoIIIC:
$$\begin{array}{c|cc} 0 & 1/2 & -1/2 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

Three-stage LobattoIIIC:
$$\begin{array}{c|ccc} 0 & 1/6 & -1/3 & 1/6 \\ 1/2 & 1/6 & 5/12 & -1/12 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

Two-stage RadauIIA:
$$\begin{array}{c|cc} 1/3 & 5/12 & -1/12 \\ 1 & 3/4 & 1/4 \\ \hline & 3/4 & 1/4 \end{array}$$

Three-stage RadauIIA:
$$\begin{array}{c|ccc} \frac{2}{5}-\frac{\sqrt{6}}{10} & \frac{11}{45}-\frac{7\sqrt{6}}{360} & \frac{37}{225}-\frac{169\sqrt{6}}{1800} & -\frac{2}{225}+\frac{\sqrt{6}}{75} \\ \frac{2}{5}+\frac{\sqrt{6}}{10} & \frac{37}{225}+\frac{169\sqrt{6}}{1800} & \frac{11}{45}+\frac{7\sqrt{6}}{360} & -\frac{2}{225}-\frac{\sqrt{6}}{75} \\ 1 & \frac{4}{9}-\frac{\sqrt{6}}{36} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{4}{9}-\frac{\sqrt{6}}{36} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

$$\textbf{Crank–Nicolson method:}\quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

$$\textbf{Two-stage Pareschi-Russo:}\quad \begin{array}{c|cc} x & x & 0 \\ 1-x & 1-2x & x \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

$$\textbf{Three-stage Alexander:}\quad \begin{array}{c|ccc} x & x & 0 & 0 \\ \frac{1+x}{2} & \frac{1-x}{2} & x & 0 \\ 1 & -1.5x^2+4x-0.25 & 1.5x^2-5x+1.25 & x \\ \hline & -1.5x^2+4x-0.25 & 1.5x^2-5x+1.25 & x \end{array}$$

Chapter 2 References

[1] J. H. Adler, H. Sterck, S. P. MacLachlan, and L. Olson. *Numerical Partial Differential Equations.* To appear, 2023.

[2] O. Axelsson, I. Dravins, and M. Neytcheva. *Stage-parallel Preconditioners for Implicit Runge-Kutta Methods of Arbitrary High Order: Linear Problems.* Department of Information Technology, Uppsala Universitet, 2022.

[3] D. Braess. *Finite elements: Theory, fast solvers, and applications in solid mechanics.* Cambridge University Press, 2007.

[4] D. Braess and R. Sarazin. An efficient smoother for the Stokes problem. *Applied Numerical Mathematics*, 23(1):3–19, 1997.

[5] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.

[6] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1-4):23–56, 1986.

[7] S. C. Brenner, L. R. Scott, and L. R. Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

[8] J. Butcher. Integration processes based on Radau quadrature formulas. *Mathematics of Computation*, 18(86):233–244, 1964.

[9] J. C. Butcher. Implicit Runge-Kutta processes. *Mathematics of Computation*, 18(85):50–64, 1964.

[10] J. C. Butcher. On the implementation of implicit Runge-Kutta methods. *BIT Numerical Mathematics*, 16(3):237–240, 1976.

[11] J. C. Butcher. General linear methods. *Acta Numerica*, 15:157–256, 2006.

[12] J. C. Butcher. *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2016.

[13] M. R. Clines, V. E. Howle, and K. R. Long. Efficient order-optimal preconditioners for implicit Runge-Kutta and Runge-Kutta-Nyström methods applicable to a large class of parabolic and hyperbolic PDEs. *arXiv preprint arXiv:2206.08991*, 2022.

[14] G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3(1):27–43, 1963.

[15] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[16] J. Dendy. Black box multigrid. *Journal of Computational Physics*, 48(3):366–386, 1982.

[17] I. Dravins, S. Serra-Capizzano, and M. Neytcheva. Fine spectral analysis of preconditioned matrices and matrix-sequences arising from stage-parallel implicit Runge-Kutta methods of arbitrarily high order. *arXiv preprint arXiv:2302.04657*, 2023.

[18] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.

[19] H. C. Elman, Y. Saad, and P. E. Saylor. A hybrid chebyshev krylov subspace algorithm for solving nonsymmetric systems of linear equations. *SIAM Journal on Scientific and Statistical Computing*, 7(3):840–855, 1986.

[20] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, USA, 2014.

[21] P. E. Farrell, R. C. Kirby, and J. Marchena-Menendez. Irksome: Automating Runge–Kutta time-stepping for finite element methods. *ACM Transactions on Mathematical Software (TOMS)*, 47(4):1–26, 2021.

[22] P. E. Farrell, M. G. Knepley, L. Mitchell, and F. Wechsung. PCPATCH: Software for the topological construction of multigrid relaxation methods. *ACM Trans. Math. Softw.*, 47(3), 2021.

[23] G. E. Forsythe and W. R. Wasow. Finite-difference methods for partial differential equations. *Applied Mathematics Series*, 1960.

[24] G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods. *Numerische Mathematik*, 3(1):157–168, 1961.

[25] M. H. Gutknecht and S. Röllin. The Chebyshev iteration revisited. *Parallel Computing*, 28(2):263–283, 2002.

[26] L. Hageman. The Chebyshev polynomial method of iteration. Technical report, BAPL (Bettis Atomic Power Laboratory (BAPL), West Mifflin, PA (United States)), 1967.

[27] P. W. Hemker. A note on defect correction processes with an approximate inverse of deficient rank. *Journal of Computational and Applied Mathematics*, 8(2):137–139, 1982.

[28] R. C. Kirby. On the convergence of monolithic multigrid for implicit Runge-Kutta time stepping of finite element problems. *arXiv preprint arXiv:2304.14879*, 2023.

[29] H.-O. Kreiss and O. E. Ortiz. *Introduction to numerical methods for time dependent differential equations*. John Wiley & Sons, 2014.

[30] S. Leveque, L. Bergamaschi, Á. Martínez, and J. W. Pearson. Fast iterative solver for the all-at-once Runge–Kutta discretization. *arXiv preprint arXiv:2303.02090*, 2023.

[31] S. P. MacLachlan and C. W. Oosterlee. A local Fourier analysis framework for finite-element discretizations of systems of pdes. *Numer. Linear Algebra Appl*, 18:751–774, 2011.

[32] J. Maitre, F. Musy, and P. Nigon. A fast solver for the Stokes equations using multigrid with a Uzawa smoother. In *Advances in Multi-Grid Methods: Proceedings of the conference held in Oberwolfach, December 8 to 13, 1984*, pages 77–83. Springer, 1985.

[33] K.-A. Mardal, T. K. Nilssen, and G. A. Staff. Order-optimal preconditioners for implicit Runge–Kutta schemes applied to parabolic PDEs. *SIAM Journal on Scientific Computing*, 29(1):361–375, 2007.

[34] P. Munch, I. Dravins, M. Kronbichler, and M. Neytcheva. Stage-parallel fully implicit runge–kutta implementations with optimal multilevel preconditioners at the scaling limit. *SIAM Journal on Scientific Computing*, pages S71–S96, 2023.

[35] W. Pazner and P.-O. Persson. Stage-parallel fully implicit Runge–Kutta solvers for discontinuous Galerkin fluid simulations. *Journal of Computational Physics*, 335:700–717, 2017.

[36] M. M. Rana, V. E. Howle, K. Long, A. Meek, and W. Milestone. A new block preconditioner for implicit Runge-Kutta methods for parabolic PDE. *SIAM J. Sci. Comp.*, 43(5):S475–S495, 2021.

[37] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[38] E. N. Sarmin and L. Chudov. On the stability of the numerical integration of systems of ordinary differential equations arising in the use of the straight line method. *USSR Computational Mathematics and Mathematical Physics*, 3(6):1537–1543, 1963.

[39] W. E. Schiesser. *The numerical method of lines: integration of partial differential equations*. Elsevier, 2012.

[40] A. H. Sherman. On Newton-iterative methods for the solution of systems of nonlinear equations. *SIAM Journal on Numerical Analysis*, 15(4):755–771, 1978.

[41] B. S. Southworth, O. Krzysik, W. Pazner, and H. D. Sterck. Fast solution of fully implicit Runge–Kutta and discontinuous Galerkin in time for numerical PDEs, part i: the linear setting. *SIAM Journal on Scientific Computing*, 44(1):A416–A443, 2022.

[42] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Elsevier, 2000.

[43] J. Van Lent and S. Vandewalle. Multigrid methods for implicit Runge–Kutta and boundary value method discretizations of parabolic PDEs. *SIAM Journal on Scientific Computing*, 27(1):67–92, 2005.

[44] S. P. Vanka. Block-implicit calculation of steady turbulent recirculating flows. *International Journal of Heat and Mass Transfer*, 28(11):2093–2103, 1985.

[45] G. Wanner and E. Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

[46] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM review*, 34(4):581–613, 1992.

[47] P.-b. Zhou. *Numerical analysis of electromagnetic fields*. Springer Science & Business Media, 2012.

Chapter 3

Monolithic multigrid for implicit Runge-Kutta discretizations of incompressible fluid flow

Abstract

[1]Most research on preconditioners for time-dependent PDEs has focused on implicit multi-step or diagonally-implicit multi-stage temporal discretizations. In this paper, we consider monolithic multigrid preconditioners for fully-implicit multi-stage Runge-Kutta (RK) time integration methods. These temporal discretizations have very attractive accuracy and stability properties, but they couple the spatial degrees of freedom across multiple time levels, requiring the solution of very large linear systems. We extend the classical Vanka relaxation scheme to implicit RK discretizations of saddle point problems. We present numerical results for the incompressible Stokes, Navier-Stokes, and resistive magnetohydrodynamics equations, in two and three dimensions, confirming that these relaxation schemes lead to robust and scalable monolithic multigrid methods for a challenging range of incompressible fluid-flow models.

[1]Authors are R. Abu-Labdeh, P.E. Farrell, and S.P. MacLachlan. This work is the adapted version of the published version as *Monolithic multigrid for implicit Runge-Kutta discretizations of incompressible fluid flow* in the Journal of Computational Physics, 2023.

Keywords: Implicit Runge-Kutta time integration, Monolithic multigrid, Newton-Krylov-multigrid Methods.

3.1 Introduction

Among the many applications of advanced computer simulation, models of fluid flow have been a persistent and common driving force in research and practice. The history of spatial discretization of fluid problems dates back at least to the 1960's (e.g., the MAC-scheme discretization of the Navier-Stokes equations [30, 19, 60]), but continues to this day with investigation of higher-order mixed finite-element discretizations for both Newtonian and complex fluids [56, 69, 37, 28, 55, 32, 34, 33]. Alongside this thrust to higher-order spatial discretizations comes a need for stable higher-order temporal discretizations, for which implicit Runge-Kutta methods are a natural choice. In this paper, we investigate the development of efficient Newton-Krylov-multigrid strategies for implicit Runge-Kutta discretizations of incompressible fluid-flow problems.

Effective solver strategies for both stationary problems and time-dependent flow models discretized via either multi-step schemes or diagonally implicit Runge-Kutta (DIRK) schemes have been studied for many years. For time-dependent Newtonian flows, both fully and semi-implicit pressure-correction schemes (e.g., [19, 20, 60, 61, 8]) have been proposed, based primarily on multigrid solution of the pressure-Poisson equation, but the construction and analysis of general high-order schemes is non-trivial [29]. Monolithic multigrid schemes (both linear and nonlinear) have also been broadly considered, first arising in the late 1970's and early 1980's [14, 13]. More approaches have been proposed since these early works, including techniques for Newtonian flows based on Vanka [64] and Braess-Sarazin [12] relaxation, and generalizations of these techniques to more complex flow settings and discretizations [3, 2, 1]. Simultaneously, block preconditioning strategies have also been developed, for a variety of discretizations and flow settings [38, 22, 66, 27, 26, 41]. Despite this substantial body of work on multi-step methods, there are (to our knowledge) few comparable publications on solution strategies for multi-stage implicit Runge-Kutta (IRK) discretizations of flow models [46, 58].

A small body of work exists on solvers for IRK discretizations for parabolic

PDEs [62, 51, 11, 43, 35, 18, 48, 58, 57]. Much of this work focuses on block-structured preconditioners for the tensor-product systems generated by IRK discretization [43, 35, 18, 48] where, for example, standard multigrid methods can be used to solve the diagonal blocks. The recent method of Southworth et al. [58, 57] appears to be very effective, again leveraging standard preconditioners for linear systems corresponding to BDF-type discretizations. On the other hand, the work of Vandewalle and others [62, 51, 11] applies monolithic multigrid methods to these discretizations, using block-Gauss-Seidel type relaxation for parabolic equations and a block-extension of the Hiptmair relaxation [31] for the eddy-current form of the curl-curl equation. Similar block-Jacobi relaxation was used for both the heat and Gross-Pitaevskii equations in [24]. Here, we investigate extensions of Vanka relaxation for IRK discretizations of fluid flow problems.

In this paper, we consider standard mixed finite-element (spatial) discretizations of Stokes, Navier-Stokes, and magnetohydrodynamic (MHD) flows, coupled with IRK discretizations in time. We focus on the development of monolithic geometric multigrid preconditioners for the coupled systems of equations to be solved at each timestep. For nonlinear problems, we use these preconditioners in a standard Newton-Krylov-multigrid setting, using Newton's method to linearize the coupled nonlinear systems at each timestep. We expect the same techniques would apply to the various simplifications of Newton's method that are applicable in the IRK context [15, 10]. Numerical results are presented for standard benchmarks in two and three spatial dimensions, showing that this solution approach is equally effective for IRK discretizations as it is for BDF discretizations.

The remainder of this paper is organized as follows. In Section 3.2, we review the Runge-Kutta discretization approach for systems of ODEs. For fluid-flow models, this is typically used in a method-of-lines approach with some spatial discretization, and Section 3.3 reviews mixed finite-element discretization of the Stokes, Navier-Stokes, and MHD models considered here. In Section 3.4, we present the constituent parts of the monolithic multigrid algorithm that we propose for solution of the resulting linear(ized) systems of equations. Numerical results that confirm the effectiveness of this approach are given in Section 3.5. Finally, conclusions and directions for future work are given in Section 3.6.

3.2 Runge-Kutta temporal discretizations

While BDF (and other linear multi-step) schemes can achieve higher-order convergence, they do so at a cost to their stability, with the widely known result that no linear multi-step scheme with order greater than two can be A-stable (the so-called Second Dahlquist Barrier) [65]. Because of this (and other reasons), Runge-Kutta integrators are widely used when we seek higher-order time integration methods. In contrast to multi-step schemes (where solutions at past time-steps are used in the approximation), Runge-Kutta methods are *multi-stage* schemes, where a number of intermediate stage values are used to achieve the approximation. In general, an r-stage Runge-Kutta method applied to the system of ordinary differential equations $u'(t) = f(u(t), t)$ is given by

$$\begin{aligned} k_i &= f\left(u^n + \Delta t \sum_{j=1}^{r} a_{ij} k_j, t^n + c_i \Delta t \right), \text{ for } i = 1, 2, \dots, r, \\ u^{n+1} &= u^n + \Delta t \sum_{j=1}^{r} b_j k_j. \end{aligned} \tag{3.1}$$

The coefficients in the scheme are the stage times (or nodes) c_i, the weights b_j, and the Runge-Kutta matrix $A = [a_{ij}]$. Taken together, these form the Butcher tableau for a given scheme [15, 16]. For consistency, we require that $\sum_{j=1}^{r} b_j = 1$ and $\sum_{j=1}^{r} a_{ij} = c_i$, for all $i = 1, 2, \dots, r$. The r stage values are represented by the set $\{k_i\}_{i=1}^{r}$ and the approximation at time $t^n = t^0 + n\Delta t$ is denoted by u^n.

Runge-Kutta methods are generally classified by the non-zero pattern of the matrix A. Methods can be explicit, with $a_{ij} = 0 \ \forall j \geq i$, or implicit, when $\exists j \geq i$ with $a_{ij} \neq 0$. The implicit methods can further be classified into diagonally implicit, with $a_{ij} = 0 \ \forall j > i$, or fully implicit, when $\exists j > i$ such that $a_{ij} \neq 0$. Further specialization is also possible, such as singly diagonally implicit Runge-Kutta (SDIRK) methods, which are diagonally implicit (DIRK) methods with the added property that $a_{ii} = a_{jj}$ for all i and j, and explicit singly diagonally implicit (ESDIRK) methods, which have an all-zero first row of A, followed by SDIRK structure on lower rows (of which the Crank-Nicolson scheme is a well-known example).

There are three main points to consider when choosing a Runge-Kutta method, regarding its stability, accuracy, and computational cost. For any scheme, we define the function $r(z)$ as the map produced when applying the scheme to the (scalar, linear) Dahlquist test problem, $u' = \lambda u$ for $\lambda \in \mathbb{C}$, with $u^{n+1} = r(\lambda \Delta t)u^n$. The domain of stability of the scheme is defined as the region in the complex plane where $|r(z)| < 1$. RK methods are said to be *A-stable* if their domain of stability includes the entire left-half of the complex plane. If, additionally, we have that $\lim_{z \to -\infty} |r(z)| = 0$, we say that the scheme is *L-stable*. For many applications, L-stability is the preferred property, since an L-stable scheme generally damps non-physical high-frequency oscillations that may pollute a numerical solution. As is typical, explicit Runge-Kutta (ERK) methods have finite regions of stability, and only implicit Runge-Kutta (IRK) schemes can be A- or L-stable.

The local truncation error of an RK scheme is defined as the error made in a single step of the scheme, starting with the analytical solution of the differential equation as u^n, compared to $u(t^{n+1})$, while the global error is the accumulated error in the approximate solution over the timesteps needed to reach a fixed time. We typically discuss such errors by their order, meaning that we bound the error by a constant (depending on $f(u,t)$ and the analytical solution, $u(t)$) times $(\Delta t)^p$ to establish that a scheme has order p. Typically (e.g., when $f(u,t)$ is continuous in t and Lipschitz continuous in u), the global error is one order less than the local truncation error. A well-known result is that the order of global error of an ERK method cannot be greater than its number of stages (and, to achieve order $p \geq 5$, an ERK scheme must have at least $p+1$ stages) [17, Section 324]. In contrast, the maximum order of global error for an IRK discretization can be as much as twice the number of stages in the scheme. While higher-order global error is attractive, for both stiff DEs and systems of differential-algebraic equations (DAEs), the so-called *stage order* of a Runge-Kutta method is more important [17, Section 362]. Here, the accuracy of a scheme is determined not just by its truncation error, but also by bounding the approximation of stage i to $u(t^n + c_i \Delta t)$ by some constant (depending on $f(u,t)$ and $u(t)$) times $(\Delta t)^{q+1}$, defining the stage order to be the smaller of q and the order of the scheme. For index-2 DAEs (as are considered here), the order of accuracy of a scheme is limited by its stage order, due to perturbation bounds on the solution of the constrained system [65, Section VII.4]. This greatly limits our choice of schemes that allow higher-order accuracy. While DIRK methods can have reasonable global order, their stage order is typically limited to 1. We note that ESDIRK methods are an exception to this, with stage order limited to 2, due to the structure of their

Butcher tableau. In contrast, the stage order of fully IRK schemes can be as large as the number of stages, making these the preferred schemes for integrating DAEs.

The downside of IRK schemes is their computational cost. ERK methods can be implemented at the cost of one evaluation of $f(u,t)$ for each stage in the method. In contrast, IRK methods require solution of a system of equations for each timestep (that may be large when u represents a spatially discretized approximation to the solution of a PDE). Herein lies the attraction of DIRK, SDIRK, and ESDIRK schemes. In these approaches, rather than having to solve for the stages in a coupled manner, each stage can be solved for sequentially, allowing the reuse of standard linear and nonlinear solvers from backward-Euler type schemes. SDIRK and ESDIRK afford even more of an advantage, particularly in the linear case, as the same solver architecture can be directly reused in the solution process for each stage. General IRK methods, in contrast, do not allow this simplification. While block-preconditioning strategies can be used again to leverage existing solver architectures from the multistep case [43, 35, 18, 48, 58, 57], these theoretical results tend to be limited to simple cases, excluding (for example) nonlinear systems of DAEs, as arise in standard models of computational fluid dynamics.

In this paper, we consider a standard Newton-Krylov-multigrid framework for the solution of the nonlinear systems of equations that arise from using general IRK discretizations for the Navier-Stokes equations and the equations of magnetohydrodynamics. Because the details of these solvers depend directly on the spatial discretization, we next discuss the mixed finite-element discretization of these models.

3.3 Discretization of fluid models

In this section, we consider the interplay of mixed finite-element spatial discretization for incompressible models of fluid flow with temporal discretization by IRK methods. We consider three models: the linear Stokes model, the (nonlinear) Navier-Stokes equations, and the equations of single-fluid visco-resistive incompressible magnetohydrodynamics (MHD). In Section 3.4, we will focus on the development of a monolithic multigrid methodology for the linearized systems that result from applying Newton's method to the nonlinear problems. Both here and in that exposition, we will focus on the details of the algorithm for the simplest case of the linear Stokes model.

3.3.1 Time-dependent Stokes equations

In the viscous limit of incompressible flow, inertial forces in the model can be neglected, leading to the time-steady Stokes equations. We consider here the time-dependent analogue of the Stokes equations on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$:

$$\begin{aligned} \rho \mathbf{u}_t - \mu \Delta \mathbf{u} + \nabla p &= \mathbf{f} && \text{in } \Omega \times (0, T_f) && \text{(3.2a)} \\ -\nabla \cdot \mathbf{u} &= 0 && \text{in } \Omega \times (0, T_f), && \text{(3.2b)} \\ \mathbf{u} &= 0 && \text{on } \partial\Omega \times (0, T_f), && \text{(3.2c)} \\ \mathbf{u}(\mathbf{x}, 0) &= \mathbf{g}(\mathbf{x}) && \text{on } \Omega \times \{t = 0\}, && \text{(3.2d)} \end{aligned}$$

where $\mathbf{u}(\mathbf{x}, t)$ is the velocity, $p(\mathbf{x}, t)$ is the pressure, and $\mathbf{f}(\mathbf{x}, t)$ is a suitably smooth forcing term. Here, ρ denotes the fluid density and μ denotes the fluid viscosity; we set both to 1 for simplicity. The final time is denoted by T_f. Since no time derivative of the pressure appears in the system, it is a DAE. The *index* of a DAE is defined as the number of analytical differentiations needed (along with algebraic manipulations) to convert the DAE into an explicit system of ODEs [65, Section VII.1]. Here, since the constraint equation is of the form $-\nabla \cdot \mathbf{u} = 0$, this is an index-two DAE, since one differentiation of the constraint (and applying the divergence to (3.2a)) allows us to explicitly solve for p in terms of $\mathbf{u}$, and a second gives an ODE for p. For index-two DAEs, the order of accuracy of a Runge–Kutta time-discretization is limited to the stage order of the scheme.

For the spatial discretization of (3.2), we use the mixed finite-element framework, considering the stable Taylor-Hood discretization on simplices [22]. Let $\mathcal{V} = \mathbf{H}_0^1(\Omega)$, where $\mathbf{H}_0^1(\Omega) = \{\mathbf{v} \in \mathbf{H}^1(\Omega) : \mathbf{u} = 0 \text{ on } \partial\Omega\}$, and $\mathcal{W} = L_0^2(\Omega)$ (the space of zero-mean functions in $L^2(\Omega)$), and consider a weak solution of (3.2) that is (at least) once continuously differentiable in time and such that for every $t \in (0, T_f)$, $\mathbf{u}(\cdot, t) \in \mathcal{V}$ and $p(\cdot, t) \in \mathcal{W}$. Multiplying the time-dependent equation by $\mathbf{v} \in \mathcal{V}$ and the divergence constraint by $q \in \mathcal{W}$ and integrating by parts, we get the weak form

$$\begin{aligned} \langle \mathbf{u}_t, \mathbf{v} \rangle + \langle \nabla \mathbf{u}, \nabla \mathbf{v} \rangle - \langle p, \nabla \cdot \mathbf{v} \rangle = \langle \mathbf{f}, \mathbf{v} \rangle, &\quad \forall \mathbf{v} \in \mathcal{V}, \\ -\langle q, \nabla \cdot \mathbf{u} \rangle = 0, &\quad \forall q \in \mathcal{W}, \end{aligned}$$

where the inner-product notation, $\langle \cdot, \cdot \rangle$, denotes integration in space but not time. The finite-element discretization is realized by constructing a triangulation, τ_h, of Ω, and approximating $\mathbf{u}$ and p in piecewise polynomial spaces defined over τ_h. Here, we use standard continuous Lagrange finite-element spaces, defining

$$P_k(\Omega, \tau_h) = \left\{ u \in C^0(\Omega) : \forall T \in \tau_h,\, u|_T(\mathbf{x}) \text{ is a polynomial of degree no more than } k \right\}.$$

We consider the standard stable Taylor-Hood discretization, with $\mathcal{V}_h = (P_2(\Omega, \tau_h))^d \cap \mathcal{V}$ and $\mathcal{W}_h = P_1(\Omega, \tau_h) \cap \mathcal{W}$ [22, 59]. This leads to the semi-discretized weak form of finding $(\mathbf{u}(\cdot, t), p(\cdot, t)) \in \mathcal{V}_h \times \mathcal{W}_h$ such that

$$\begin{aligned} \langle \mathbf{u}_t, \mathbf{v} \rangle + \langle \nabla \mathbf{u}, \nabla \mathbf{v} \rangle - \langle p, \nabla \cdot \mathbf{v} \rangle &= \langle \mathbf{f}, \mathbf{v} \rangle, \quad && \forall \mathbf{v} \in \mathcal{V}_h, \\ -\langle q, \nabla \cdot \mathbf{u} \rangle &= 0, && \forall q \in \mathcal{W}_h. \end{aligned}$$

Now writing $\vec{\mathbf{u}}(t)$ and $\vec{p}(t)$ for the (time-dependent) coefficients of $\mathbf{u}(\mathbf{x}, t)$ and $p(\mathbf{x}, t)$ in the finite-element basis, we can write this as a coupled linear system of DAEs, as

$$\begin{bmatrix} M\vec{\mathbf{u}}_t \\ 0 \end{bmatrix} + \begin{bmatrix} K & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \vec{\mathbf{u}} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} M\vec{\mathbf{f}} \\ 0 \end{bmatrix},$$

where $\vec{\mathbf{f}}$ is the vector of coefficients of the interpolant of $\mathbf{f}$ in $\mathcal{V}_h$. Here, M and K are the $(P_2(\Omega, \tau_h))^d$ mass and stiffness matrices, respectively, while B is the weak gradient operator mapping from $\mathcal{W}_h$ into $\mathcal{V}_h$.

It is this system of equations that we discretize using Runge-Kutta methods. As the system is a set of DAEs, and not ODEs, we cannot directly apply (3.1), but use its DAE analogue [65], writing

$$\begin{aligned} \vec{\mathbf{u}}_i^n &= \vec{\mathbf{u}}^n + \Delta t \sum_{j=1}^r a_{ij} \vec{k}_j^{(\mathbf{u})}, & \vec{p}_i^n &= \vec{p}^n + \Delta t \sum_{j=1}^r a_{ij} \vec{k}_j^{(p)}, \\ M\vec{k}_i^{(\mathbf{u})} + K\vec{\mathbf{u}}_i^n + B\vec{p}_i^n &= M\vec{\mathbf{f}}_i^n, & B^T \vec{\mathbf{u}}_i^n &= 0, \\ \vec{\mathbf{u}}^{n+1} &= \vec{\mathbf{u}}^n + \Delta t \sum_{j=1}^r b_j \vec{k}_j^{(\mathbf{u})}, & \vec{p}^{n+1} &= \vec{p}^n + \Delta t \sum_{j=1}^r b_j \vec{k}_j^{(p)}, \end{aligned}$$

where $\vec{\mathbf{f}}_i^n$ is the interpolant of $\mathbf{f}$ in $\mathcal{V}_h$ at time $t^n + c_i \Delta t$, $\vec{\mathbf{u}}_i^n$ and $\vec{p}_i^n$ are the approximations of $\vec{\mathbf{u}}$ and $\vec{p}$ at time $t^n + c_i \Delta t$, and $\vec{k}_i^{(\mathbf{u})}$ and $\vec{k}_i^{(p)}$ are the RK stages for which we solve. Rewriting the equations for $\vec{k}_i^{(\mathbf{u})}$ and $\vec{k}_i^{(p)}$, we have

$$\begin{aligned} M\vec{k}_i^{(\mathbf{u})} + K\left(\vec{\mathbf{u}}^n + \Delta t \sum_{j=1}^{r} a_{ij}\vec{k}_j^{(\mathbf{u})}\right) + B\left(\vec{p}^n + \Delta t \sum_{j=1}^{r} a_{ij}\vec{k}_j^{(p)}\right) &= M\vec{\mathbf{f}}_i^n \\ B^T\left(\vec{\mathbf{u}}^n + \Delta t \sum_{j=1}^{r} a_{ij}\vec{k}_j^{(\mathbf{u})}\right) &= 0 \end{aligned}$$

or

$$\begin{aligned} M\vec{k}_i^{(\mathbf{u})} + \Delta t \sum_{j=1}^{r} a_{ij}\left(K\vec{k}_j^{(\mathbf{u})} + B\vec{k}_j^{(p)}\right) &= M\vec{\mathbf{f}}_i^n - K\vec{\mathbf{u}}^n - B\vec{p}^n \\ \Delta t \sum_{j=1}^{r} a_{ij}B^T\vec{k}_j^{(\mathbf{u})} &= -B^T\vec{\mathbf{u}}^n \end{aligned}$$

for $1 \leq i \leq r$. The matrix on the left can easily be written in tensor-product form, leading to a concise description of the scheme as

$$\left(\mathbf{I}_r \otimes \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} + \Delta t A \otimes \begin{bmatrix} K & B \\ B^T & 0 \end{bmatrix}\right)\vec{\mathbf{k}} = \vec{\mathbf{F}}, \tag{3.3}$$

where $\vec{\mathbf{k}}$ is the vector of stages, ordered consecutively by stage index i, keeping the ordering of $(\vec{k}_i^{(\mathbf{u})}, \vec{k}_i^{(p)})$ pairs together, and $\vec{\mathbf{F}}$ is the corresponding vector of right-hand sides (including terms from timestep n).

3.3.2 Navier-Stokes equations

We next include the full inertial term, leading to the nonlinear incompressible Navier-Stokes equations,

$$\rho\left(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}\right) - \mu\Delta\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega \times (0, T_f), \tag{3.4a}$$

$$-\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega \times (0, T_f), \tag{3.4b}$$

$$\mathbf{u} = 0 \quad \text{on } \partial\Omega \times (0, T_f), \tag{3.4c}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{g}(\mathbf{x}) \text{ on } \Omega \times \{t = 0\}. \tag{3.4d}$$

We again take the density to be 1, but will allow the viscosity μ to be chosen differently, to consider problems at different Reynolds numbers. The additional term passes directly to the weak form, which we again discretize using a Taylor-Hood mixed finite-element discretization. The semi-discretized weak variational form of (3.4) is to find $(\mathbf{u}(\cdot,t), p(\cdot,t)) \in \mathcal{V}_h \times \mathcal{W}_h$ such that

$$\begin{aligned} \langle \mathbf{u}_t, \mathbf{v}\rangle + \langle \mathbf{u}\cdot\nabla\mathbf{u}, \mathbf{v}\rangle + \mu\langle \nabla\mathbf{u}, \nabla\mathbf{v}\rangle - \langle p, \nabla\cdot\mathbf{v}\rangle &= 0, \\ -\langle \nabla\cdot\mathbf{u}, q\rangle &= 0, \end{aligned} \tag{3.5}$$

for all test functions $(\mathbf{v}, q) \in \mathcal{V}_h \times \mathcal{W}_h$. Again writing $\vec{\mathbf{u}}(t)$ and $\vec{p}(t)$ for the coefficients of $\mathbf{u}$ and p in the finite-element basis, this leads to a nonlinear coupled system of DAEs, as

$$\begin{bmatrix} M\vec{\mathbf{u}}_t \\ 0 \end{bmatrix} + \begin{bmatrix} N(\vec{\mathbf{u}}) \\ 0 \end{bmatrix} + \begin{bmatrix} K & B \\ B^T & 0 \end{bmatrix}\begin{bmatrix} \vec{\mathbf{u}} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} M\vec{\mathbf{f}} \\ 0 \end{bmatrix},$$

where $N(\vec{\mathbf{u}})$ represents the discretization of $\langle \mathbf{u}\cdot\nabla\mathbf{u}, \mathbf{v}\rangle$. As above, accounting for this term in the RK stage equations leads to the nonlinear coupled system

$$\begin{aligned} M\vec{k}_i^{(\mathbf{u})} + N\left(\vec{\mathbf{u}}^n + \Delta t\sum_{j=1}^{r} a_{ij}\vec{k}_j^{(\mathbf{u})}\right) + \Delta t\sum_{j=1}^{r} a_{ij}\left(K\vec{k}_j^{(\mathbf{u})} + B\vec{k}_j^{(p)}\right) &= M\vec{\mathbf{f}}_i^n - K\vec{\mathbf{u}}^n - B\vec{p}^n, \\ \Delta t\sum_{j=1}^{r} a_{ij}B^T\vec{k}_j^{(\mathbf{u})} &= -B^T\vec{\mathbf{u}}^n, \end{aligned}$$

for $1 \le i \le r$. This system is solved using Newton's method.

Denoting the nonlinear system as $F\left(\vec{\mathbf{k}}^{n}\right) = 0$, a standard Newton approximation would be to solve

$$F\left(\vec{\mathbf{k}}^{n,\ell+1}\right) \approx F\left(\vec{\mathbf{k}}^{n,\ell}\right) + J\left(\vec{\mathbf{k}}^{n,\ell}\right)\delta\vec{\mathbf{k}}^{n,\ell} = 0,$$

where $J\left(\vec{\mathbf{k}}^{n,\ell}\right)$ is the Jacobian of the system at the current approximation, $\vec{\mathbf{k}}^{n,\ell}$ and $\delta\vec{\mathbf{k}}^{n,\ell} := \vec{\mathbf{k}}^{n,\ell+1} - \vec{\mathbf{k}}^{n,\ell}$ is the Newton search direction. Since we are timestepping, we use the computed solution at the previous time-step, $\vec{\mathbf{k}}^{n-1}$, for the initial guess for the stage values at step n, $\vec{\mathbf{k}}^{n,0}$. In this work, we use the Eisenstat-Walker stopping criterion for the Krylov iteration to solve for $\delta\vec{\mathbf{k}}^{n,\ell}$ [21], requiring that

$$\left\|F\left(\vec{\mathbf{k}}^{n,\ell}\right) + J\left(\vec{\mathbf{k}}^{n,\ell}\right)\delta\vec{\mathbf{k}}^{n,\ell}\right\| \le \eta_\ell \left\|F\left(\vec{\mathbf{k}}^{n,\ell}\right)\right\|,$$

for every step, ℓ, where $\eta_\ell \in [0,1)$ is updated for each nonlinear iteration based on convergence of the method.

3.3.3 Magnetohydrodynamics

Finally, we consider the equations of single-fluid viscoresistive magnetohydrodynamics (MHD). In general, MHD models the flow of conducting fluids in the presence of an electromagnetic field. These models are nonlinear and contain strong coupling between the fluid velocity and the electromagnetic variables. We follow the MHD formulation presented in [1],

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla\cdot(\frac{2}{\mathrm{Re}}\epsilon(\mathbf{u})) + \nabla p - (\nabla\times\mathbf{B})\times\mathbf{B} = \mathbf{f}_{\mathbf{u}} \text{ in } \Omega\times(0,T_f), \tag{3.6a}$$

$$\mathbf{B}_t + \frac{1}{\mathrm{Re}_m}\nabla\times\nabla\times\mathbf{B} - \nabla\times(\mathbf{u}\times\mathbf{B}) - \nabla\gamma = \mathbf{f}_{\mathbf{B}} \text{ in } \Omega\times(0,T_f), \tag{3.6b}$$

$$-\nabla\cdot\mathbf{u} = 0 \text{ in } \Omega\times(0,T_f), \tag{3.6c}$$

$$\nabla\cdot\mathbf{B} = 0 \text{ in } \Omega\times(0,T_f), \tag{3.6d}$$

$$\mathbf{u} = 0 \text{ on } \partial\Omega\times(0,T_f), \tag{3.6e}$$

$$\mathbf{B}\times\mathbf{n} = 0 \text{ on } \partial\Omega\times(0,T_f), \tag{3.6f}$$

$$\mathbf{u}(\mathbf{x},0) = \mathbf{g}_{\mathbf{u}}(\mathbf{x}) \text{ on } \Omega\times\{t=0\}, \tag{3.6g}$$

$$\mathbf{B}(\mathbf{x},0) = \mathbf{g}_{\mathbf{B}}(\mathbf{x}) \text{ on } \Omega\times\{t=0\}, \tag{3.6h}$$

where the four unknowns are the velocity vector, $\mathbf{u}$, the pressure, p, the magnetic field, $\mathbf{B}$, and the Lagrange multiplier, γ. The Lagrange multiplier is used to enforce the solenoidal condition (3.6d), while the pressure is used to enforce the incompressibility condition (3.6c). The strain-rate tensor is $\epsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$, and the two dimensionless constants, Re and Re_m, are the hydrodynamic Reynolds number and magnetic Reynolds number, respectively. We consider this equation in both two- and three-dimensional domains, Ω; in 2D, the curl and cross-product are defined by the natural extensions from three-dimensional vector fields to two-dimensional fields.

Here, for $\Omega \subset \mathbb{R}^d$, we take

$$(\mathbf{u}(\cdot,t), \mathbf{B}(\cdot,t), p(\cdot,t), \gamma(\cdot,t)) \in \mathbf{H}_0^1(\Omega)\times\mathbf{H}_0(\mathrm{curl},\Omega)\times L_0^2(\Omega)\times H_0^1(\Omega),$$

where

$$\begin{aligned}
\mathbf{H}_0^1(\Omega) &= \{\mathbf{v} \in \mathbf{H}^1(\Omega) : \mathbf{u} = 0 \text{ on } \partial\Omega\}, \\
\mathbf{H}_0(\text{curl}, \Omega) &= \{\mathbf{c} \in \mathbf{L}^2(\Omega) : \nabla \times \mathbf{c} \in \mathbf{L}^2(\Omega), \mathbf{n} \times \mathbf{c} = 0 \text{ on } \partial\Omega\}, \\
L_0^2(\Omega) &= \{q \in L^2(\Omega) : \int_\Omega q \, \mathrm{d}\mathbf{x} = 0\}, \\
H_0^1(\Omega) &= \{s \in H^1(\Omega) : s = 0 \text{ on } \partial\Omega\},
\end{aligned}$$

and $\mathbf{n}$ is the outward unit normal vector on $\partial\Omega$ [1, 54]. We discretize the fluid variables again with the Taylor-Hood discretization $\mathcal{V}_h \times \mathcal{W}_h \subset \mathbf{H}_0^1(\Omega) \times L_0^2(\Omega)$, and use lowest-order Nédélec elements for $\mathcal{C}_h \subset \mathbf{H}_0(\text{curl}, \Omega)$ and $\mathcal{S}_h = P_1(\Omega, \tau_h) \cap H_0^1(\Omega)$ for the Lagrange multiplier. Well-posedness (under small-data assumptions) of both the continuous and discrete formulations is shown in [54].

Multiplying (3.6) by the test functions $(\mathbf{v}, \mathbf{c}, q, s) \in \mathcal{V}_h \times \mathcal{C}_h \times \mathcal{W}_h \times \mathcal{S}_h$ and integrating by parts, we get the semi-discretized weak variational form of finding $(\mathbf{u}(\cdot,t), \mathbf{B}(\cdot,t), p(\cdot,t), \gamma(\cdot,t)) \in \mathcal{V}_h \times \mathcal{C}_h \times \mathcal{W}_h \times \mathcal{S}_h$ such that

$$\begin{aligned}
\int_\Omega \mathbf{u}_t \cdot \mathbf{v} + ((\mathbf{u} \cdot \nabla)\mathbf{u}) \cdot \mathbf{v} - \frac{2}{\text{Re}} (\epsilon(\mathbf{u}) : \epsilon(\mathbf{v})) - p \nabla \cdot \mathbf{v} - ((\nabla \times \mathbf{B}) \times \mathbf{B}) \cdot \mathbf{v} \, \mathrm{d}\mathbf{x}& \\
= \int_\Omega \mathbf{f}_\mathbf{u} \cdot \mathbf{v} \, \mathrm{d}\mathbf{x}&, \\
\int_\Omega \mathbf{B}_t \cdot \mathbf{c} + \frac{1}{\text{Re}_m} (\nabla \times \mathbf{B}) \cdot (\nabla \times \mathbf{c}) - (\mathbf{u} \times \mathbf{B}) \cdot (\nabla \times \mathbf{c}) - (\nabla\gamma) \cdot \mathbf{c} \, \mathrm{d}\mathbf{x} = \int_\Omega \mathbf{f}_\mathbf{B} \cdot \mathbf{c} \, \mathrm{d}\mathbf{x}&, \\
- \int_\Omega (\nabla \cdot \mathbf{u}) q \, \mathrm{d}\mathbf{x} = 0&, \\
- \int_\Omega \mathbf{B} \cdot (\nabla s) \, \mathrm{d}\mathbf{x} = 0&,
\end{aligned} \tag{3.7}$$

for all $(\mathbf{v}, \mathbf{c}, q, s) \in \mathcal{V}_h \times \mathcal{C}_h \times \mathcal{W}_h \times \mathcal{S}_h$. The corresponding IRK discretization is derived from this semi-discretized form as described above, and solved in the same Newton-Krylov-multigrid manner, using the Eisenstat-Walker stopping criterion for the Krylov iteration. We note that linear solvers for this spatial discretization using BDF2 in time was the subject of [1]; given the Dahlquist barrier, and the driving need for L-stability, multistage schemes, such as IRK, are needed to achieve higher-order time integration for this problem.

3.4 Monolithic multigrid for fluid problems

As described above, we use a Newton-Krylov-multigrid framework for solving the nonlinear systems of equations resulting from spatial and temporal discretization of the models in Section 3.3 (noting that the Newton linearization is trivial in the case of the linear Stokes equations). Since the systems are nonsymmetric, we use FGMRES [52] as the Krylov method, and seek to effectively precondition it. For IRK discretizations of scalar PDEs, such as the heat equation, block-diagonal preconditioning of the stage-coupled linear systems is known to be effective [43]. While block-diagonal preconditioning has also been developed for fluid models discretized using BDF-like methods [22, 66], we leave extension of this approach to IRK discretizations for future work. Instead, we follow the approach of [64, 62], and develop a monolithic multigrid preconditioner that makes use of an overlapping additive Schwarz relaxation that can be viewed as the extension of Vanka relaxation to the IRK case. Compared to the use of block-structured preconditioners, this offers the advantage of not needing to explicitly approximate Schur complements in the stage-coupled IRK linearizations (which may depend on properties of the specific IRK scheme chosen, for example). We note that FGMRES and classical right-preconditioned GMRES solve the same underlying optimization problem for the approximation in the same Krylov space, but that the underlying algorithms have important differences, with FGMRES requiring extra vector storage (to store both the Arnoldi vectors and their preconditioned counterparts) but right-preconditioned GMRES requiring an extra application of the preconditioner once the solution to the underlying Hessenberg system has been found. Thus, we choose to use FMGRES, instead of classical right-preconditioned GMRES, primarily because the cost of application of our preconditioners is non-trivial, but we are not memory-bound on the parallel machine used in the numerical results. Thus, the extra vector storage of FGMRES is an attractive trade-off over the extra preconditioner application required by standard right-preconditioned GMRES. An auxiliary advantage is that FGMRES allows the use of GMRES inside inner iterations (such as the relaxation).

3.4.1 Coarse-grid correction and transfer operators

In the numerical results that follow, we consider hierarchies of grids generated by taking uniform refinements of a given coarsest grid. To map functions from a coarse mesh to its refinement, we use canonical finite-element interpolation operators for each field in the discretization. For the single-stage case, for both the time-dependent Stokes and Navier-Stokes problems, interpolation takes the form

$$P = \begin{bmatrix} P_{\mathbf{u}} & \\ & P_p \end{bmatrix},$$

where $P_{\mathbf{u}}$ and P_p represent the interpolation operators for the P_2 and P_1 finite-element spaces, respectively. In the MHD case, we introduce finite-element interpolation operators $P_{\mathbf{B}}$ for the lowest-order Nédélec space and P_γ for the P_1 space (with suitable boundary conditions for γ), following [1], making the interpolation operator for the single-stage case

$$P = \begin{bmatrix} P_{\mathbf{u}} & & & \\ & P_{\mathbf{B}} & & \\ & & P_p & \\ & & & P_\gamma \end{bmatrix}.$$

For multistage IRK discretizations, the interpolation operator is defined as $I_r \otimes P$, creating a block-diagonal interpolation operator that applies the finite-element interpolation in P to each stage independently. We use the transpose of interpolation as the restriction operator.

We use rediscretization to define the coarse-grid operators, noting that this is equivalent to Galerkin coarsening in the finite-element case (if compatible quadrature rules are used to assemble on the fine and coarse grids). The coarsest-grid systems are solved directly, using MUMPS [6].

3.4.2 Vanka relaxation

Vanka relaxation was first introduced for the time-steady MAC-scheme discretization of the Navier-Stokes equations [63], but it has been extensively used in many more general settings in recent years [23, 3, 2, 1, 25]. Broadly defined, Vanka relaxation schemes are overlapping Schwarz (domain decomposition) methods used as relaxation for a multigrid algorithm. In order to achieve the expected cost of a multigrid relaxation scheme, the subdomain problems are generally quite small, on the order of 10s-100s of DoFs. Historically, the most common approaches were multiplicative in nature; however, we follow the recent trend towards additive schemes [23, 1, 25] that are naturally parallelizable.

To specify the details of relaxation, we now describe how the Schwarz subdomains (commonly referred to as the Vanka "blocks" or "patches") are constructed from the underlying mesh on any given level of the multigrid hierarchy. In this work, we follow the topological construction described in [25]. In particular, we form a Vanka patch for each vertex in the mesh, which consists of all degrees of freedom associated with the closure of the cells adjacent to the vertex. As is typical in Vanka relaxation, we exclude all degrees of freedom associated with P_1 constraints (the pressure and Lagrange multiplier in our systems) from the patch, except for those located at the vertex around which the patch is formed. For the models considered here, this results in patches like those shown in Figure 3.1 for regular two-dimensional grids, with a single pressure degree of freedom and all velocity DoFs on all elements adjacent to the node. When used in an IRK discretization, these patches include all stage degrees of freedom. For MHD, we note that the patch shown at right of Figure 3.1 coincides topologically with the *coupled Vanka* approach for the BDF2 discretization considered in [1], but the patches used here contain more degrees of freedom than those used in [1], due to inclusion of all stages in the IRK discretization.

Denoting the set of DoFs in the i^{th} Vanka patch by $\mathcal{S}_i$, we have (by construction) that every degree of freedom in the domain is contained in at least one patch: $\mathcal{S} = \bigcup_{i=1}^{N} \mathcal{S}_i$, where N is the total number of patches and $\mathcal{S}$ is the complete set of DoFs for the problem. Denoting R_i as a "restriction" operator that maps global DoFs to those in patch $\mathcal{S}_i$, we can write a single iteration of a weighted stationary iteration as

$$\vec{\mathbf{k}} \leftarrow \vec{\mathbf{k}} + \omega \sum_{i=1}^{N} R_i^T (R_i J R_i^T)^{-1} R_i (\vec{\mathbf{F}} - J\vec{\mathbf{k}}),$$

where $J\vec{\mathbf{k}} = \vec{\mathbf{F}}$ is the linear system to be solved, and $R_i J R_i^T$ is the restriction of J to the DoFs in patch $\mathcal{S}_i$. In practice, we use several steps of a Vanka-preconditioned Chebyshev or GMRES iteration as the relaxation scheme for our problems, with the endpoints of the interval defining the associated Chebyshev polynomials tuned by

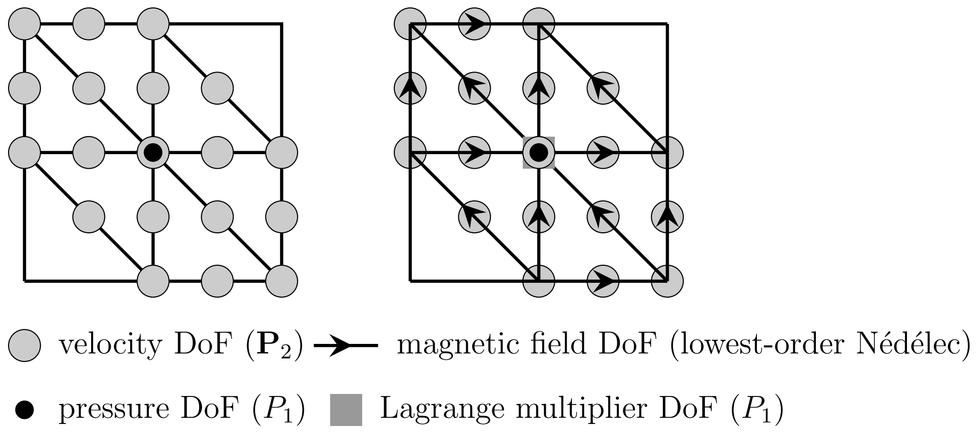

velocity DoF ($\mathbf{P}_2$)
magnetic field DoF (lowest-order Nédélec)
pressure DoF (P_1)
Lagrange multiplier DoF (P_1)

Figure 3.1: Left: Vanka patch for the Stokes and Navier-Stokes equations, consisting of P_2 velocity DoFs, and one P_1 pressure DoF. Right: Vanka patch for the MHD equations, consisting of P_2 velocity DoFs, lowest-order Nédélec DoFs for the magnetic field, one P_1 Lagrange multiplier DoF and one P_1 pressure DoF.

hand.

3.4.3 Implementation

The numerical results below are produced using Firedrake [49] for the spatial finite-element discretization and Irksome [24] for the temporal discretization. Linear and nonlinear solvers are implemented in PETSc [7], taking advantage of the close integration between discretizations and solvers provided by this combination [39]. The Vanka relaxation is implemented through PCPATCH [25]. For reproducibility, the codes used to generate the numerical results and the major components of Firedrake, Irksome, and PETSc needed, have been archived on Zenodo [68]. We emphasize that all aspects of the discretization and solver software are chosen to be naturally parallelizable. The coarsest mesh in each hierarchy is distributed across the available parallel cores, and then refined in parallel. For the two-dimensional problems below, after each refinement, the mesh is redistributed to better balance parallel work, but this was not done for the 3D example, due to software limitations. While not rebalancing the meshes leads to a small load imbalance on the finer grids in the hierarchies, this was not seen to lead to significant loss of performance in the weak scaling tests reported below. To account for the need to compute residuals for each DoF in each Vanka patch, a node-distance-2 halo is included in the parallel mesh distribution, to allow the relaxation scheme to be performed in parallel without additional communication [42].

3.5 Numerical Results

For the numerical results in this paper, we focus on 3 families of IRK methods: Gauss (also known as Gauss-Legendre), LobattoIIIC, and RadauIIA. We note that LobattoIIIC and RadauIIA are both L-stable and A-stable, while Gauss is A-stable, but not L-stable. All are fully implicit schemes, with stage order equal to the number of stages. For an r-stage method, Gauss schemes have order $2r$, while RadauIIA have order $2r-1$ and LobattoIIIC have order $2r-2$. We consider both 2- and 3-stage schemes here, with standard Butcher tableaux [67, 65].

We present results for four separate test cases: a simple two-dimensional time-dependent Stokes model, two-dimensional Navier-Stokes flow past a cylinder, and two MHD examples, a two-dimensional island-coalescence problem and a three-dimensional lid-driven cavity model. All results presented in this paper were computed on the Compute Canada cluster, *Niagara*, consisting of 2,024 nodes, each with 40 2.4 GHz Intel Skylake cores and 202GB of RAM, connected using a 100Gb/s EDR Dragonfly+ network.

3.5.1 Two-dimensional time-dependent Stokes

We consider a method of manufactured solutions test case, solving (3.2) on the two-dimensional unit square, $\Omega = (0,1)^2$. The forcing function $\mathbf{f}$ and boundary conditions are chosen so that the exact solution is

$$\mathbf{u} = \begin{bmatrix} \sin(\pi x)\cos(\pi y)e^{-2t\pi^2} \\ -\cos(\pi x)\sin(\pi y)e^{-2t\pi^2} \end{bmatrix}, \text{ and } p = 0.$$

For this example, we construct a coarsest grid by creating a uniform 8×8 quadrilateral mesh of the unit square, then cut each quadrilateral cell into 4 triangles, adding a vertex at the center of the quadrilateral. This mesh is then uniformly refined ℓ times; below, we present results for $\ell = 5, 6, 7$, where the Taylor-Hood discretization of the Stokes equations results in about 1.1 million DoFs per stage for $\ell = 5$ up to about 19 million DoFs per stage for $\ell = 7$. The initial condition is chosen by interpolating the exact solution into the finite-element space at $t = 0$, and we integrate up to time $T_f = 0.5$, with timestep $\Delta t = T_f/N$ for $N = 2^{\ell+3}$. To our knowledge, there are no rigorous stopping tolerances that guarantee discretization-error level accuracy for these systems; we use a hand-tuned stopping tolerance, where we require the absolute value of the ℓ_2 norm of the residual of the system to be reduced below $10^{-2} \times N^{-3}$ at each timestep, or a relative reduction in this norm by 10^{-8}. Based on preliminary experiments, we accelerate the relaxation process using Chebyshev polynomials of the first kind on the interval $[2, 8]$ and employ 2 pre- and post-relaxation sweeps. Proper choice of the Chebyshev interval is critical to achieving scalable performance. For two-dimensional problems with geometric coarsening by a factor of two (as used here), a reasonable strategy is to estimate the largest eigenvalue, λ, of the relaxation-preconditioned matrix (e.g., using Ritz values from preconditioned GMRES) and choose the interval to be $[\lambda/4, \lambda]$. Here, we started from similar estimates, but hand-tuned the intervals to optimize performance.

Table 3.1 presents a weak scaling study for this problem, for both two- and three-stage methods. For the two-stage methods, we use 10 cores on 1 node for $\ell = 5$, 40 cores on 1 node for $\ell = 6$ and 160 cores on 4 nodes for $\ell = 7$. For the three-stage methods, we increase core counts by 50%, to account for the increased number of degrees of freedom in the resulting linear systems, using 15 cores on 1 node for $\ell = 5$, 60 cores on 2 nodes for $\ell = 6$ and 240 cores on 6 nodes for $\ell = 7$. We report the relative L_2 error in the velocity and the absolute L_2 error in the pressure approximation at the final time, as well as the average number of linear iterations to achieve convergence over all timesteps and the total computational time needed in minutes. In the final column of Table 3.1, we report the average wall-clock time per Krylov iteration (t/K) in seconds.

Table 3.2 summarizes rates of convergence for the results shown in Table 3.1. We observe at least second-order convergence in the velocity error for all three IRK schemes; however, we notice much larger errors for the Gauss results than for the other two schemes. We note that the stopping tolerance decreases by a factor of 8 with each refinement, so that the slight increase in averaged iterations to convergence is not overly surprising, and seems to remain bounded at reasonable levels. Nonetheless, the factor four increase in the number of cores with each refinement is insufficient to

	ℓ	velocity error	pressure error	iterations	time	t/K
Gauss(2)	5	1.786×10^{-2}	2.328×10^{-2}	9.85	57.47	1.401
	6	3.155×10^{-3}	1.162×10^{-2}	13.39	202.98	1.779
	7	5.575×10^{-4}	6.271×10^{-3}	19.14	725.02	2.285
RadauIIA(2)	5	3.380×10^{-6}	6.327×10^{-9}	8.70	56.13	1.570
	6	4.297×10^{-7}	2.795×10^{-9}	10.99	171.08	1.837
	7	4.971×10^{-8}	1.028×10^{-9}	14.22	575.77	2.391
LobattoIIIC(2)	5	9.823×10^{-4}	4.594×10^{-7}	9.23	54.13	1.912
	6	2.495×10^{-4}	1.479×10^{-7}	11.71	181.88	1.833
	7	6.289×10^{-5}	4.823×10^{-8}	15.25	618.77	2.393
Gauss(3)	5	9.098×10^{-5}	1.481×10^{-4}	13.38	109.39	1.929
	6	1.839×10^{-5}	6.802×10^{-4}	24.51	483.75	2.323
	7	3.293×10^{-6}	3.235×10^{-4}	25.97	1332.40	3.249
RadauIIA(3)	5	6.151×10^{-7}	1.298×10^{-9}	9.32	82.29	2.083
	6	1.122×10^{-7}	1.310×10^{-9}	12.40	267.78	2.566
	7	1.300×10^{-8}	1.718×10^{-10}	13.91	877.99	3.786
LobattoIIIC(3)	5	6.019×10^{-7}	2.728×10^{-9}	9.61	85.25	2.120
	6	1.083×10^{-7}	1.966×10^{-9}	12.91	248.95	2.293
	7	1.271×10^{-8}	3.797×10^{-10}	14.91	938.06	3.770

Table 3.1: Numerical results for two-dimensional Stokes model problem with two- and three-stage IRK schemes. Relative L_2 errors in velocity and absolute L_2 errors for pressure are reported, along with average number of linear solver iterations per time-step, total wall-clock time-to-solution in minutes, and time per Krylov iteration in seconds, for refinement levels $\ell = 5, 6, 7$.

		Gauss		RadauIIA		LobattoIIIC	
		u	p	u	p	u	p
two-stage	$\log_2 e_5/e_6$	2.5	1.0	3.0	1.2	2.0	1.6
	$\log_2 e_6/e_7$	2.5	0.9	3.1	1.4	2.0	1.6
three-stage	$\log_2 e_5/e_6$	2.4	-2.3	2.5	0.0	2.5	0.5
	$\log_2 e_6/e_7$	2.5	1.1	3.1	2.9	3.1	2.4

Table 3.2: Rates of convergence in velocity and pressure for data in Table 3.1 with two- and three-stage IRK schemes for refinement levels $\ell = 5, 6, 7$. Here, e_ℓ denotes the error in a quantity on refinement level ℓ.

lead to ideal time scaling (which would be to double with each refinement, due to the doubling of the number of time-steps).

There are several contributing factors to the less-than-perfect scaling, beyond the simple increase in total number of Krylov iterations with refinement. When going from $\ell = 5$ to $\ell = 6$ with the two-stage schemes, we increase the number of cores used for the calculation, but those cores remain on one physical node, leading to a saturation of the memory bandwidth available. The same limitation occurs when going from $\ell = 6$ to $\ell = 7$ with the three-stage schemes, where we go from using 30 cores on each of 2 nodes to all 40 cores on 6 nodes. While this could be avoided by using the same number of cores on more nodes of the parallel machine, such usage is impractical when a single node has sufficient memory for the $\ell = 6$ problem with 2 IRK stages. Furthermore, when going from $\ell = 6$ to $\ell = 7$, the (direct) coarsest-grid solve goes from being dominated by its computation to being dominated by its communication. Here, we clearly see another increase in the cost per linear iteration, especially in the three-stage methods where the time required increases by about 50% for $\ell = 7$. Improved performance would almost certainly be seen by duplicating the coarse-grid solve on each node, as considered in [9, 44, 50]. We leave these performance enhancements for future work.

The experiment in Table 3.1 highlights convergence as we change both the spatial and temporal discretizations. Here, however, we note that the temporal discretizations are higher order than the spatial, particularly for the 3-stage discretizations. Thus, for comparison with this data, Table 3.3 presents results with the same setup as Table 3.1, but using a fixed timestep of $\Delta t = 0.5/2^8$, to match results with $\ell = 5$ (noting that

	ℓ	velocity error	pressure error	iterations	time	t/K
RadauIIA(2)	5	3.380×10^{-6}	6.327×10^{-9}	8.70	56.37	1.560
	6	3.119×10^{-6}	1.896×10^{-9}	12.18	96.77	0.940
	7	3.101×10^{-6}	2.793×10^{-9}	18.04	165.42	0.540
RadauIIA(3)	5	6.151×10^{-7}	1.298×10^{-9}	9.32	82.07	2.087
	6	3.677×10^{-8}	3.288×10^{-9}	14.22	116.27	0.964
	7	3.566×10^{-8}	8.063×10^{-10}	17.30	238.18	0.809
LobattoIIIC(3)	5	6.019×10^{-7}	2.728×10^{-9}	9.61	75.87	1.850
	6	6.066×10^{-8}	3.076×10^{-9}	14.60	107.87	0.866
	7	5.611×10^{-8}	1.136×10^{-9}	17.47	239.03	0.805

Table 3.3: Results analogous to Table 3.1, but with $\Delta t = 0.5/2^8$. Relative L_2 errors in velocity and absolute L_2 errors for pressure are reported, along with average number of linear solver iterations per time-step, total wall-clock time-to-solution in minutes, and time per Krylov iteration in seconds, for refinement levels $\ell = 5, 6, 7$.

these results were run independently, so small differences in timings for $\ell = 5$ naturally arise). We see that while quite reasonable convergence is observed in Table 3.1, we observe significant stagnation in convergence here. Thus, even though the temporal discretizations are higher order, we still see substantial benefits to varying the timestep simultaneously with refinement of the spatial mesh.

Finally, Table 3.4 presents comparison results for diagonal IRK schemes. Here, we consider the two-stage second-order Pareschi-Russo (with parameter $1 - \sqrt{2}/2$) [45] and three-stage third-order Alexander [5] integrators, which are both L-stable. While these results show some outperformance of the theoretical guarantees given by their stage order of one, they are also quite poor in comparison to the RadauIIA integrators of the same number of stages. In particular, comparing with the results in Table 3.1, we see that the errors achieved using DIRK(3) with $\ell = 6$ are comparable to those achieved when using RadauIIA(2) with $\ell = 5$, but that the latter calculation was achieved in about 60% of the wall-clock time and on one-eighth of the number of cores (10 for RadauIIA(2) with $\ell = 5$ vs. 80 for DIRK(3) with $\ell = 6$). Similarly, the errors for DIRK(3) with $\ell = 7$ are slightly better than those achieved with RadauIIA(2) and $\ell = 6$, and slightly worse than those achieved with RadauIIA(3) and $\ell = 6$. The two-stage Radau results, however, are achieved in just over 40% of the wall-clock time, and on one-sixth the cores, while the three-stage Radau results are achieved in about two-thirds the wall-clock time, on one-fourth the cores. These results highlight

	ℓ	velocity error	pressure error	iterations	time	t/K
DIRK(2)	5	2.480×10^{-5}	9.266×10^{-8}	6.42	35.20	1.312
	6	6.198×10^{-5}	3.254×10^{-8}	8.23	110.43	1.590
	7	1.546×10^{-5}	1.173×10^{-8}	10.07	213.76	1.835
DIRK(3)	5	1.106×10^{-5}	4.161×10^{-8}	6.99	35.28	1.240
	6	2.325×10^{-6}	1.857×10^{-9}	9.51	94.98*	1.12
	7	2.607×10^{-7}	4.578×10^{-9}	11.17	415.26	2.36

Table 3.4: Numerical results for two-dimensional Stokes model problem with two- and three-stage DIRK schemes. Relative L_2 errors in velocity and absolute L_2 errors for pressure are reported, along with average number of linear solver iterations per time-step, total wall-clock time-to-solution in minutes, and time per Krylov iteration in seconds, for refinement levels $\ell = 5, 6, 7$. Due to a change in the configuration of the machine, results for DIRK(3) at $\ell = 6$ were run on 80 cores, instead of 60; all other results were run with same parallelism as in Table 3.1.

the added accuracy that can be gained using fully implicit RK methods over DIRK methods, and the added efficiency possible when using state-of-the-art linear solvers to achieve that accuracy. For this reason, we focus on only the fully implicit RK schemes in the remainder of the paper.

3.5.2 Two-dimensional Navier-Stokes

We next consider two-dimensional Navier-Stokes flow past a cylinder, following the example given in [36, 24, 53]. Here, we consider the spatial domain $\Omega = (0, 2.2) \times (0, 0.41) \setminus B_r(0.2, 0.2)$, where $B_r(0.2, 0.2)$ is the disc of radius $r = 0.05$ centred at $(0.2, 0.2)$, shown in Figure 3.2. No-slip (zero-velocity) boundary conditions are imposed on the top and bottom boundaries of the rectangle and along the surface of the cylinder. Time-dependent inflow conditions are given on the left edge, prescribing

$$\mathbf{u}(0, y, t) = \begin{bmatrix} \frac{4U(t)y(0.41-y)}{0.41^2} \\ 0 \end{bmatrix},$$

where $U(t) = 1.5 \sin\left(\frac{\pi t}{8}\right)$ is the mean inflow velocity. No-stress outflow is prescribed on the right boundary. The viscosity is set as $\mu = 10^{-3}$, resulting in a Reynolds number of 100. The time step for these experiments is fixed as $\Delta t = \frac{1}{400}$, and we consider the final time $T_f = 8$. As above, we discretize using Taylor-Hood elements

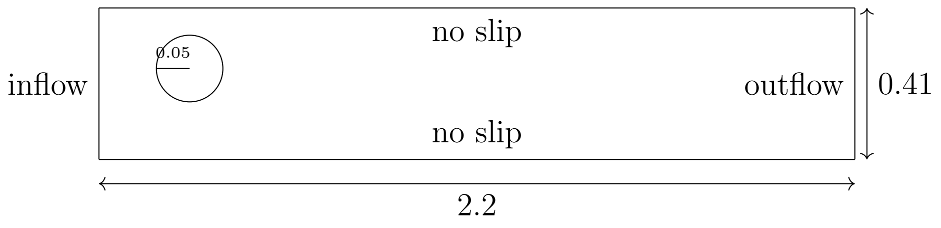


Figure 3.2: Domain for Navier-Stokes flow past a cylinder.

stages	ℓ	total DoFs	nodes	cores
2	3	489,656	1	6
	4	1,948,144	1	25
	5	7,771,616	4	100
	6	31,044,544	10	400
3	3	734,484	1	10
	4	2,922,216	1	40
	5	11,657,424	4	160
	6	46,566,816	16	640

Table 3.5: Total number of DoFs and number of nodes and cores used for the Navier-Stokes test problem with two- and three-stage IRK discretizations.

in space and IRK in time.

For this problem, an unstructured coarsest grid with 972 triangular elements is used, chosen to refine the representation around the included cylinder. Below, we report results for $3 \leq \ell \leq 6$, with discrete problem sizes for the Taylor-Hood discretization ranging from about 245 thousand DoFs per stage for $\ell = 3$ to about 15.5 million DoFs per stage for $\ell = 6$. Details of the parallelization are provided in Table 3.5, where we again note that we have increased the number of cores for the 3-stage IRK methods by about 60% over those for the 2-stage methods. For this problem, we use a nonlinear stopping tolerance requiring the absolute ℓ_2 norm of the nonlinear residual be below $1/N^3$ with $N = 2^{\ell+3}$, and use an Eisenstat-Walker inexact Newton scheme to determine the linear stopping tolerances for each nonlinear iteration. Here, again 2 pre- and post-relaxation sweeps are used, with Chebyshev polynomials for relaxation taken over the interval $[1.5, 8]$.

As no analytical solution is available in this case, we instead record the maximum drag and lift values computed over the simulations, for comparison with reference

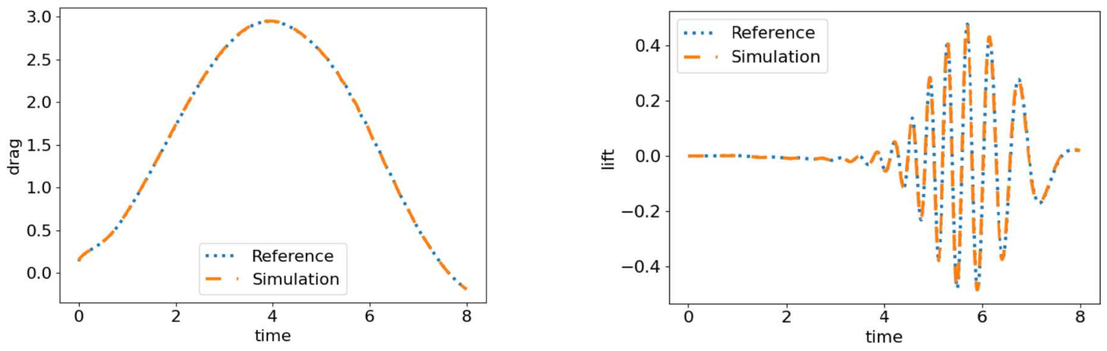


Figure 3.3: Comparison of reference and drag (left) and lift (right) computed using LobattoIIIC(2) and $\ell = 6$.

data [36, 24]. Figure 3.3 presents time-histories of these quantities for one simulation, showing excellent agreement with reference data. Results for other simulations with both RadauIIA and LobattoIIIC are visually similar. Table 3.6 presents these values for both of these integrators, along with the average wall-clock time in minutes per time-step, and average nonlinear and linear solver iterations per time-step. As before, we have decreasing solver tolerances as ℓ increases, so the small increase in iterations counts with refinement are expected.

Using both RadauIIA and LobattoIIIC IRK discretizations, with either two or three stages, results in computed lift and drag values that are consistent with those presented in [36, 24]. However, results computed with Gauss were not. Figure 3.4 shows results using the three-stage Gauss method with $\ell = 3$, computed with a stricter stopping tolerance (absolute nonlinear residual norm below $1/N^4$) than used above for RadauIIA and LobattoIIIC methods[2]. The appearance of "thick lines" in these plots reflects highly oscillatory numerical solutions. We hypothesize that this is due to the lack of L-stability of the integrator, where large negative eigenvalues of the linearized spatial operator are not quickly damped but, rather, slowly decay and oscillate in time due to a stability function value close to -1. Refinement in time for fixed spatial grids should ameliorate the issue, but leads to increased computational costs to achieve similar accuracy to that given by RadauIIA and LobattoIIIC with these timesteps.

[2]Using the same stopping tolerance led to even more inconsistent data.

	nref	drag max	lift max	time	nonlinear its	linear its
RadauIIA(2)	3	2.95	0.48	0.03	1.37	1.61
	4	2.95	0.48	0.04	1.76	2.39
	5	2.95	0.48	0.07	2.14	3.56
	6	2.95	0.48	0.13	2.52	5.09
LobattoIIIC(2)	3	2.95	0.48	0.03	1.35	1.80
	4	2.95	0.48	0.05	1.79	2.76
	5	2.95	0.48	0.07	2.15	4.30
	6	2.95	0.48	0.17	2.70	8.19
RadauIIA(3)	3	2.95	0.48	0.04	1.57	2.27
	4	2.95	0.48	0.07	1.90	2.78
	5	2.95	0.48	0.11	2.17	3.68
	6	2.95	0.48	0.17	2.56	5.01
LobattoIIIC(3)	3	2.95	0.48	0.04	1.38	1.74
	4	2.95	0.48	0.06	1.78	2.50
	5	2.95	0.48	0.10	2.14	3.60
	6	2.95	0.48	0.19	2.54	5.06

Table 3.6: Maximum drag and lift values, average wall-clock time per time-step (in minutes) and average numbers of nonlinear and linear iterations per time step for $3 \leq \ell \leq 6$ for Navier-Stokes flow past a cylinder.

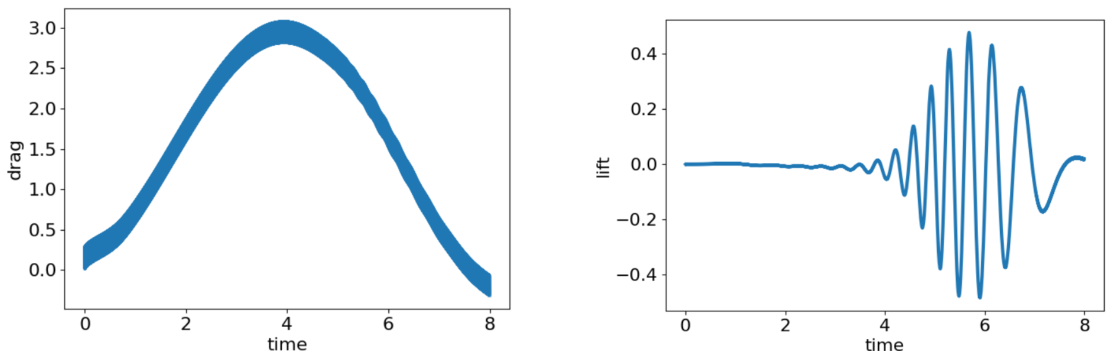


Figure 3.4: Drag and lift for $\ell = 4$ using Gauss(3). The “thick lines” indicate that the solutions are highly oscillatory in time, due to the lack of L-stability of the integrator.

3.5.3 Two-dimensional MHD island coalescence

We next consider a standard test model in MHD, of two-dimensional island coalescence. This model mimics flow in a large aspect ratio tokamak, considering a cross-section of flow of magnetically confined plasma. When a large external magnetic field is imposed in the "toroidal" direction of the tokamak, essentially two-dimensional dynamics result. This model geometry is then mapped and rescaled to a square domain, $\Omega = (-1,1)^2$, with periodic boundary conditions on the left and right edges (see [40, 1, 4] for more details). In this geometry, an equilibrium solution to the MHD equations is given by

$$\begin{aligned}
\mathbf{u}_0(x,y) &= \mathbf{0},\\
\mathbf{B}_0(x,y) &= \frac{1}{\cosh(2\pi y)+k\cos(2\pi x)}\begin{pmatrix}\sinh(2\pi y)\\ k\sin(2\pi x)\end{pmatrix},\\
p(x,y) &= \frac{1-k^2}{2}\left(1+\frac{1}{(\cosh(2\pi y)+k\cos(2\pi x))^2}\right),\\
\gamma(x,y) &= 0,
\end{aligned}$$

where $k = 0.2$, when forcing terms of

$$\begin{aligned}
\mathbf{f_u} &= \mathbf{0},\\
\mathbf{f_B} &= \frac{-8\pi^2(k^2-1)}{Re_m(\cosh(2\pi y)+k\cos(2\pi x))^3}\begin{pmatrix}\sinh(2\pi y)\\ k\sin(2\pi x)\end{pmatrix},
\end{aligned}$$

are imposed on the differential equation. To initialize a dynamic problem, these forcing terms are applied, but the initial condition is perturbed by adding

$$\delta\mathbf{B} = \frac{-0.01}{\pi}\begin{pmatrix}-\cos(\pi x)\sin(\frac{\pi y}{2})/2\\ \cos(\frac{\pi y}{2})\sin(\pi x)\end{pmatrix}.$$

to the equilibrium solution at $t = 0$. The expected effect of this perturbation is to create two initially separated "islands" of current density that break the magnetic field lines, which then reconnect. At the reconnection point (or $\mathcal{X}$-point), a sudden sharp spike should be seen in the magnetic current density. At higher Reynolds numbers, a "sloshing" effect should occur before the islands of current density merge.

As above, no analytical solution is known for this problem. A key measure of

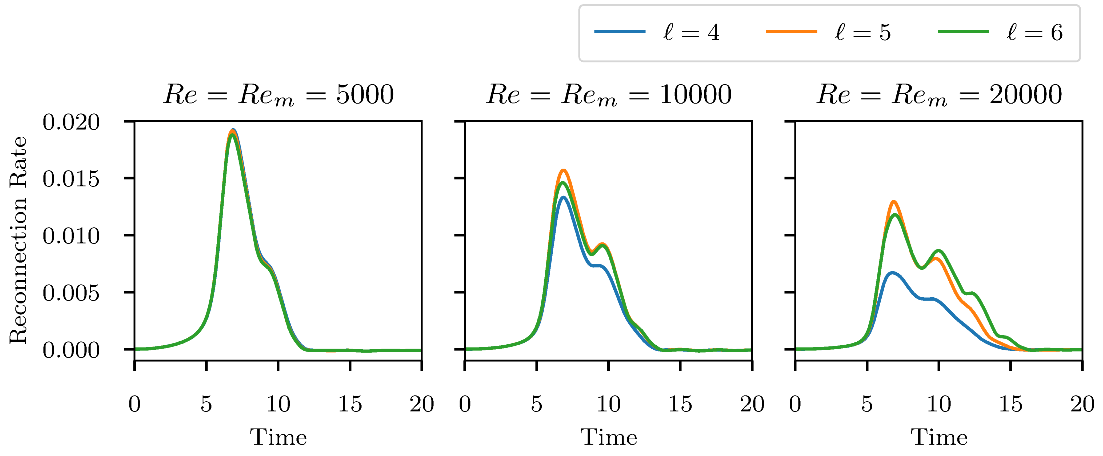


Figure 3.5: Reconnection rates recorded for the 2D MHD island coalescence model with varying Reynolds numbers of $\mathrm{Re} = \mathrm{Re}_m = 5000$, 10000 and 20000 on 3 different levels of refinement using the LobattoIIIC(2) temporal discretization.

the physical fidelity is the time-history of the *reconnection rate*, computed as the difference between the curl of $\mathbf{B}$ at the origin at the current time and its value at the origin, scaled by $1/\sqrt{\mathrm{Re}_m}$. We compute this using the same methodology as in [1]. As Re and Re_m increase, the peak value of the reconnection rate should decrease, and the length of time for which this value is nonzero should increase. In this section, we consider only the two-stage LobattoIIIC integrator, and integrate until $T_f = 20$. Following [1], we "substep" for the first time-step, taking 10 substeps to initialize the simulation and avoid problems with nonlinear convergence. We consider a coarsest spatial mesh of 20×20 quadrilateral elements, again each cut into 4 triangles, and present results for $\ell = 4, 5, 6$ refinements. For $\ell = 4$, the resulting discretization has about 2.7 million DoFs per stage, while it has about 42.7 million DoFs per stage for $\ell = 6$. For $\ell = 4$, we use $\Delta t = 0.025$, which is halved with each spatial refinement. We test for 3 different pairs of Reynolds numbers, $\mathrm{Re} = \mathrm{Re}_m$: 5000, 1000 and 20000. Figure 3.5 shows the computed reconnection rates for these problems with varying $\mathrm{Re} = \mathrm{Re}_m$ and ℓ, properly reflecting the expected behaviour.

For this problem, we adjust the nonlinear and linear solver parameters as follows. Taking $N = 20 \times 2^\ell$ as a representative number of elements in one dimension on refinement level ℓ, we set both nonlinear and linear stopping tolerances to demand an absolute reduction of the ℓ_2 norm of the corresponding residual below $1/N^2$. We now use 3 pre- and post-relaxation sweeps, with the Chebyshev polynomials defining

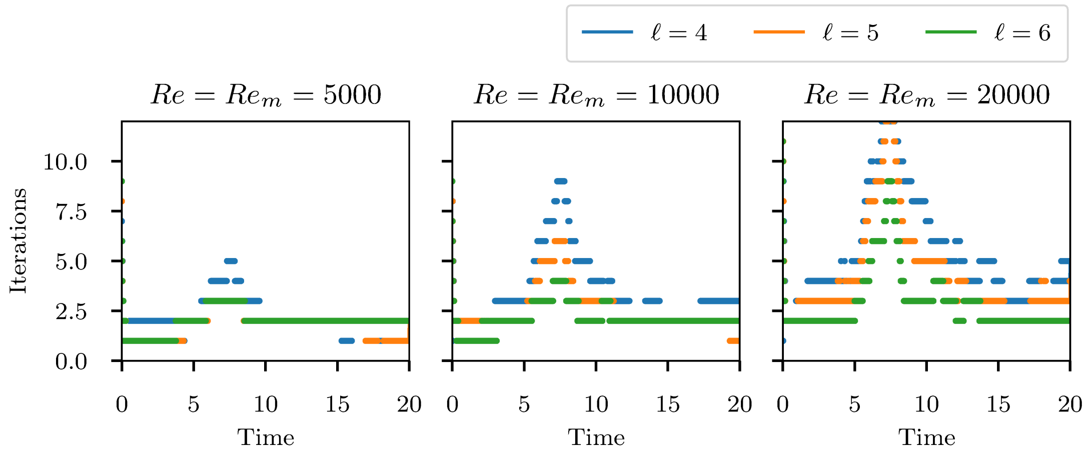


Figure 3.6: Number of linear iterations per timestep for the 2D MHD island coalescence model with varying Reynolds numbers of $\mathrm{Re} = \mathrm{Re}_m = 5000$, 10000 and 20000 on 3 different levels of refinements using the LobattoIIC(2) integrator.

the relaxation taken over the interval [2, 10]. Figures 3.6 and 3.7 present results from a weak scaling study, using 40 cores on 1 node for $\ell = 4$, 160 cores on 4 nodes for $\ell = 5$, and 640 cores on 16 nodes for $\ell = 6$. We note that these are larger core counts than those used for the same underlying meshes with a BDF2 discretization in [1]; however, this is due to the larger number of DoFs in the system using a 2-stage discretization. On average, our finest-grid problems have about 67 thousand DoFs per stage per core, which is a reasonable range for weak scaling. We note that the $\ell = 4$ problem takes about 2 hours of wall-clock time with these settings, with slightly better than doubling of wall-clock with each refinement (due to the halving of Δt with each refinement, but also improved solver performance).

Figure 3.6 shows the number of linear solver iterations recorded per timestep as we vary $\mathrm{Re} = \mathrm{Re}_m$ and ℓ. We note slight growth in iteration counts as $\mathrm{Re} = \mathrm{Re}_m$ increases (and the problem becomes less diffusive in nature), but also improving iteration counts at fixed values of $\mathrm{Re} = \mathrm{Re}_m$ as ℓ increases. Comparing with iteration counts from [1], we see slightly higher iteration counts here, with slightly worse dependence on $\mathrm{Re} = \mathrm{Re}_m$, but still reasonable performance overall. Wall-clock times per timestep, shown in Figure 3.7, generally reflect the linear iteration counts. In particular, we again see a general increase with $\mathrm{Re} = \mathrm{Re}_m$, and a general decrease with increasing ℓ. The most expensive solves in the test set are still achieved in under 1 minute per time-step, and the average time is about 0.2 minutes per time-step.

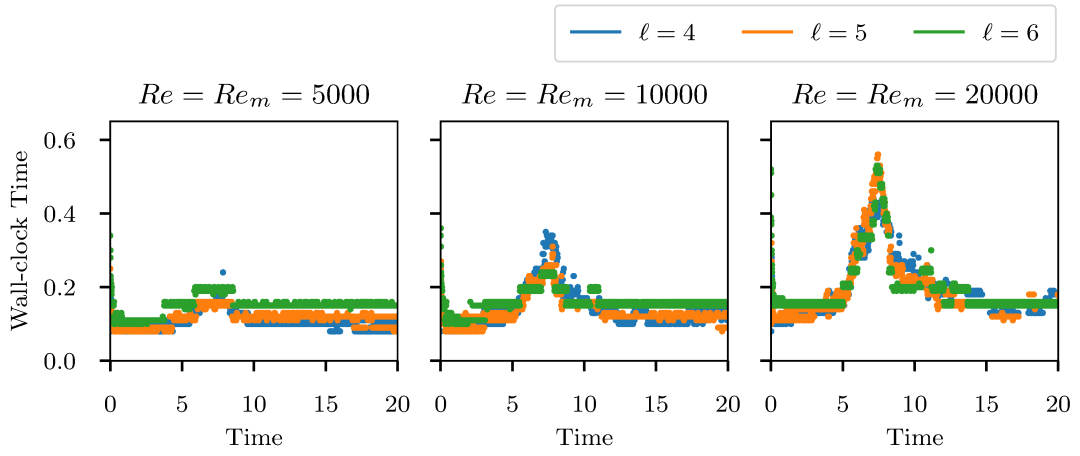


Figure 3.7: Wall-clock time (in minutes) for the nonlinear system solve at each timestep for the 2D MHD island coalescence model with varying Reynolds numbers of Re = Re_m = 5000, 10000 and 20000 on 3 different levels of refinements using the LobattoIIC(2) integrator.

To better understand the linear and nonlinear solver performance shown above, we compute both the fluid and magnetic (Alfvén) CFL numbers for the flow. At each timestep, for the given solutions for $\mathbf{u}$ and $\mathbf{B}$, we approximate the maximum magnitude of the vector fields (by projecting $\mathbf{u} \cdot \mathbf{u}$ and $\mathbf{B} \cdot \mathbf{B}$ into the discontinuous piecewise-constant finite-element space on the finest mesh and computing the maximum values of these projections), $u_{\max}$ and $B_{\max}$, and then computing the fluid CFL value, $u_{\max}\frac{\Delta t}{h}$, and the Alfvén CFL value, $B_{\max}\frac{\Delta t}{h}$, where h is a representative edge length for the spatial mesh. Figure 3.8 shows both CFL values calculated at each timestep for the simulations considered, showing identical results to those obtained in the BDF2 case in [1]. We note that, aside from the initial substeps, the Alfvén CFL is roughly constant at a value around 6, while the fluid CFL peaks at the same time as the reconnection rate, and is above 1 for the largest values of Re = Re_m considered.

3.5.4 Three-dimensional MHD lid-driven cavity

Finally, we present results for a three-dimensional lid-driven cavity MHD model on the unit cube, $\Omega = (0,1)^3$, following [47]. On the top face, $z = 1$, the flow is driven by imposed velocity $\mathbf{u} = (1,0,0)^T$, while $\mathbf{u} = (0,0,0)^T$ on all other faces. The tangential components of the magnetic field are set to match those of $\mathbf{B} = (-1,0,0)^T$ on all

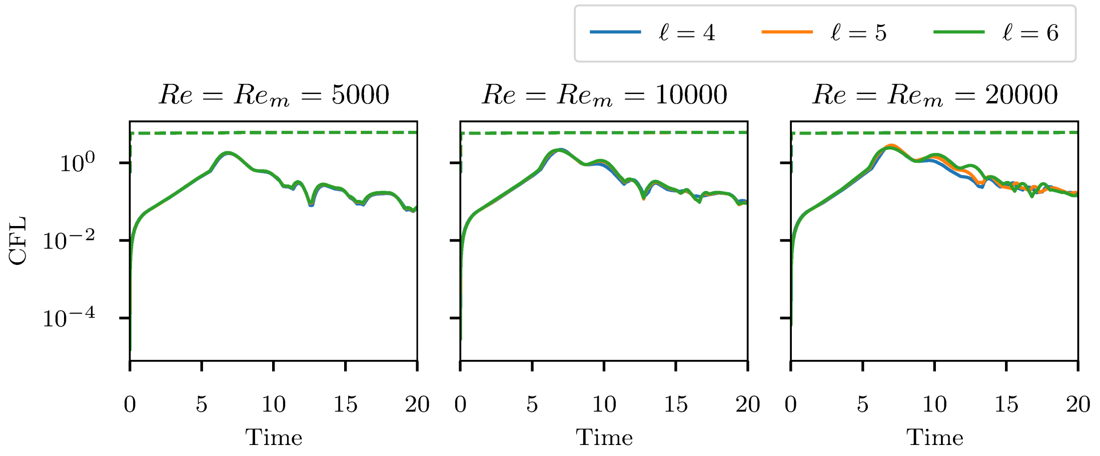


Figure 3.8: CFL values at each timestep for the 2D MHD island coalescence model with varying Reynolds numbers of $\mathrm{Re} = \mathrm{Re}_m = 5000$, 10000 and 20000 on 3 different levels of refinements using the LobattoIIC(2) integrator. Solid lines represent fluid CFL, while dashed lines represent Alfvén CFL.

faces of the cube. We set $\gamma = 0$ on all boundary faces and fix the pressure $p = 0$ at the origin. In this section, we consider 3-grid methods, refining a given coarsest grid twice for each test. This is driven by the consideration that, with increasing finest grid size, we require more cores over which to parallelize the computation; however, there is a software limitation within Firedrake that requires that the coarsest grid in the simulation must have at least 1 cell per core. While satisfying this requirement is not burdensome in 2D, it becomes problematic with increasing memory and computational requirements of 3D simulations. In all cases, we construct the coarsest grid by taking a uniform hexahedral mesh of the cube, then cutting each hexahedral element into 6 tetrahedra in the usual way. We still use ℓ to denote the levels of refinement, but now $\ell = 1$ denotes the smallest grid, created by refining a $2 \times 2 \times 2$ grid twice, while $\ell = 2$ denotes the grid created by refining a $4 \times 4 \times 4$ grid twice, and $\ell = 3$ denotes the grid created by refining a $8 \times 8 \times 8$ grid twice. With $\ell = 1$, our discretization has about 20 thousand DoFs per stage, increasing to about 1.1 million DoFs per stage for $\ell = 3$.

We employ the same spatial discretization and again use the LobattoIIIC(2) integrator. We integrate until $T_f = 2.5$. For $\ell = 1$, we take $\Delta t = 0.125$, and halve Δt with each refinement. An all-zero initial condition is used. Table 3.7 presents average linear and nonlinear iterations per timestep, along with average wall-clock time per nonlinear solve for the three grids above and $\mathrm{Re} = \mathrm{Re}_m = 10^p$ for $1 \leq p \leq 3$. For

ℓ		Re = 10	Re = 100	Re = 1000
1	linear its.	6.86	8.34	31.55
	nonlinear its.	3.10	2.76	3.52
	time	0.11	0.11	0.24
2	linear its	6.61	5.41	17.59
	nonlinear its	3.18	2.43	3.02
	time	0.21	0.17	0.30
3	linear its	5.76	4.20	6.08
	nonlinear its	2.57	2.33	2.16
	time	0.22	0.22	0.22

Table 3.7: Average number of linear and nonlinear iterations per time-step and wall-clock time per nonlinear iteration (in minutes) for the 3D MHD lid-driven cavity problem with various Reynolds numbers and grid refinements, using the LobattoIIIC(2) integrator.

$\ell = 1$, 10 cores on 1 node are used, increasing to 80 cores on 2 nodes for $\ell = 2$ and 640 cores on 16 nodes for $\ell = 3$. We use 3 pre- and post-relaxation sweeps, here accelerated using GMRES, as this was observed to result in better overall iteration counts and computation times than using Chebyshev acceleration, likely due to the convective nature of the problem at high Reynolds numbers. The nonlinear solve at each timestep requires the absolute value of the ℓ_2 norm of the residual to be reduced below 10^{-6}, and the same stopping criterion is used for the linear solves as well.

Several trends can be observed in these results. First, for fixed values of $\text{Re} = \text{Re}_m$, we generally observe improving solver performance as ℓ is increased, as expected. Similarly, we typically observe degrading solver performance as $\text{Re} = \text{Re}_m$ is increased for fixed ℓ. Overall, the iteration counts are quite reasonable, except for $\text{Re} = \text{Re}_m = 1000$ with $\ell = 1, 2$. Here, the problem is quite severely under-resolved, with a finest-grid mesh spacing of $h = 0.0625$ with $\ell = 2$, so it is not surprising that the solver suffers when the discretization is so poor. For smaller Reynolds numbers, $\text{Re} = \text{Re}_m = 1$ (not shown here), using Chebyshev acceleration gave significantly better results than using GMRES-accelerated relaxation, which failed to converge in some cases. Figure 3.9 presents representative solutions for $\ell = 3$ with $\text{Re} = \text{Re}_m = 10$ (where the solutions are well-resolved), showing streamlines of both the velocity field, $\mathbf{u}$, and the magnetic field, $\mathbf{B}$, at the final time at refinement $\ell = 3$.

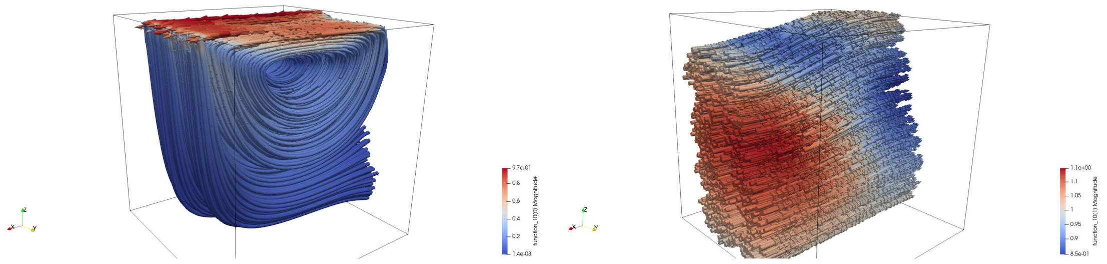


Figure 3.9: Streamlines of velocity (left) and magnetic field (right) for $\ell = 3$ with $\mathrm{Re} = \mathrm{Re}_m = 10$.

3.6 Conclusion

In this paper, we have developed monolithic Vanka relaxation schemes for fully-implicit Runge-Kutta discretizations of saddle point problems arising in models of fluid flow. Within a Newton-Krylov-multigrid setting, our method is shown to be effective for both Newtonian and magnetohydrodynamic flows, in both two and three spatial dimensions. The algorithm is chosen with parallel implementation in mind, and weak scaling results are shown up to 640 cores.

There are many possibilities for future work. We note primarily that the current study uses relatively low-order spatial discretizations, based on classical Taylor-Hood elements for velocity and pressure. A next step in this research is to extend these solvers to more sophisticated finite-element discretizations that preserve the incompressibility and solenoidality constraints exactly, as in [41, 32]. An important question for future work is the extension of these techniques to higher-order discretizations, where the cost of classical sparse direct solvers for the patch problems becomes prohibitive.

Acknowledgements

This research was enabled in part by support provided by ACENET (www.ace-net.ca), Scinet (www.scinethpc.ca), and Compute Canada (www.computecanada.ca). The work of SM and RA was partially supported by an NSERC Discovery Grant. The work of PEF was supported by the UK Engineering and Physical Sciences Research Council [grant numbers EP/R029423/1 and EP/W026163/1].

Chapter 3 References

[1] J. H. Adler, T. Benson, E. C. Cyr, P. E. Farrell, S. MacLachlan, and R. Tuminaro. Monolithic multigrid for magnetohydrodynamics. *SIAM J. Sci. Comput.*, 43(5):S70–S91, 2021.

[2] J. H. Adler, T. R. Benson, E. C. Cyr, S. P. MacLachlan, and R. S. Tuminaro. Monolithic multigrid methods for two-dimensional resistive magnetohydrodynamics. *SIAM Journal on Scientific Computing*, 38(1):B1–B24, 2016.

[3] J. H. Adler, T. R. Benson, and S. P. MacLachlan. Preconditioning a mass-conserving discontinuous Galerkin discretization of the Stokes equations. *Numerical Linear Algebra with Applications*, 24(3):e2047, 2017.

[4] J. H. Adler, M. Brezina, T. A. Manteuffel, S. F. McCormick, J. W. Ruge, and L. Tang. Island coalescence using parallel first-order system least squares on incompressible resistive magnetohydrodynamics. *SIAM Journal on Scientific Computing*, 35(5):S171–S191, 2013.

[5] R. Alexander. Diagonally implicit Runge-Kutta methods for stiff ODE's. *SIAM Journal on Numerical Analysis*, 14(6):1006–1021, 1977.

[6] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[7] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al. PETSc users manual: Revision 3.10. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2018.

[8] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85(2):257–283, 1989.

[9] J. D. Betteridge, P. E. Farrell, and D. A. Ham. Code generation for productive, portable, and scalable finite element simulation in firedrake. *Computing in Science and Engineering*, 23(4):8–17, 2021.

[10] T. A. Bickart. An efficient solution process for implicit Runge–Kutta methods. *SIAM Journal on Numerical Analysis*, 14(6):1022–1027, 1977.

[11] T. Boonen, J. Van lent, and S. Vandewalle. An algebraic multigrid method for high order time-discretizations of the div-grad and the curl-curl equations. *Applied Numerical Mathematics*, 59(3):507–521, 2009.

[12] D. Braess and R. Sarazin. An efficient smoother for the stokes problem. *Applied Numerical Mathematics*, 23(1):3–19, 1997.

[13] A. Brandt. *Multigrid techniques: 1984 guide with applications to fluid dynamics.* GMD–Studien Nr. 85. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

[14] A. Brandt and N. Dinar. Multigrid solutions to elliptic flow problems. In S. Parter, editor, *Numerical Methods for Partial Differential Equations*, pages 53–147. Academic Press, New York, 1979.

[15] J. C. Butcher. On the implementation of implicit Runge-Kutta methods. *BIT Numerical Mathematics*, 16(3):237–240, 1976.

[16] J. C. Butcher. General linear methods. *Acta Numerica*, 15:157–256, 2006.

[17] J. C. Butcher. *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2016.

[18] H. Chen. A splitting preconditioner for the iterative solution of implicit Runge-Kutta and boundary value methods. *BIT*, 54(3):607–621, 2014.

[19] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[20] A. J. Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comp.*, 23:341–353, 1969.

[21] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.

[22] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics.* Oxford University Press, USA, 2014.

[23] P. E. Farrell, Y. He, and S. P. MacLachlan. A local Fourier analysis of additive Vanka relaxation for the Stokes equations. *Numerical Linear Algebra with Applications*, page e2306, 2020.

[24] P. E. Farrell, R. C. Kirby, and J. Marchena-Menéndez. Irksome: Automating Runge–Kutta time-stepping for finite element methods. *ACM Trans. Math. Softw.*, 47(4), 2021.

[25] P. E. Farrell, M. G. Knepley, L. Mitchell, and F. Wechsung. PCPATCH: Software for the topological construction of multigrid relaxation methods. *ACM Trans. Math. Softw.*, 47(3), 2021.

[26] P. E. Farrell, L. Mitchell, L. R. Scott, and F. Wechsung. A Reynolds-robust preconditioner for the Scott-Vogelius discretization of the stationary incompressible Navier-Stokes equations. *The SMAI Journal of Computational Mathematics*, 7:75–96, 2021.

[27] P. E. Farrell, L. Mitchell, and F. Wechsung. An augmented Lagrangian preconditioner for the 3D stationary incompressible Navier–Stokes equations at high Reynolds number. *SIAM Journal on Scientific Computing*, 41(5):A3073–A3096, 2019.

[28] N. R. Gauger, A. Linke, and P. W. Schroeder. On high-order pressure-robust space discretisations, their advantages for incompressible high Reynolds number generalised Beltrami flows and beyond. *The SMAI Journal of Computational Mathematics*, 5:89–129, 2019.

[29] J. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44):6011–6045, 2006.

[30] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[31] R. Hiptmair. Multigrid method for Maxwell's equations. *SIAM J. Numer. Anal.*, 36(1):204–225, 1999.

[32] K. Hu, Y. Ma, and J. Xu. Stable finite element methods preserving $\nabla \cdot B = 0$ exactly for MHD models. *Numer. Math.*, 135(2):371–396, 2017.

[33] K. Hu, W. Qiu, and K. Shi. Convergence of a B-E based finite element method for MHD models on Lipschitz domains. *Journal of Computational and Applied Mathematics*, 368:112477, 2020.

[34] K. Hu and J. Xu. Structure-preserving finite element methods for stationary MHD models. *Math. Comp.*, 88(316):553–581, 2019.

[35] L. O. Jay. Inexact simplified Newton iterations for implicit Runge-Kutta methods. *SIAM Journal on Numerical Analysis*, 38(4):1369–1388, 2000.

[36] V. John. Reference values for drag and lift of a two-dimensional time-dependent flow around a cylinder. *International Journal for Numerical Methods in Fluids*, 44(7):777–788, 2004.

[37] V. John, A. Linke, C. Merdon, M. Neilan, and L. G. Rebholz. On the divergence constraint in mixed finite element methods for incompressible flows. *SIAM Review*, 59(3):492–544, 2017.

[38] D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256, 2002.

[39] R. C. Kirby and L. Mitchell. Solver composition across the PDE/linear algebra barrier. *SIAM J. Sci. Comput.*, 40(1):C76–C98, 2018.

[40] D. A. Knoll and L. Chacón. Coalescence of magnetic islands, sloshing, and the pressure problem. *Physics of Plasmas*, 13(3):032307, 2006.

[41] F. Laakmann, P. E. Farrell, and L. Mitchell. An augmented Lagrangian preconditioner for the magnetohydrodynamics equations at high Reynolds and coupling numbers, 2021.

[42] M. Lange, L. Mitchell, M. G. Knepley, and G. J. Gorman. Efficient mesh management in firedrake using PETSc DMPLEX. *SIAM Journal on Scientific Computing*, 38(5):S143–S155, 2016.

[43] K.-A. Mardal, T. K. Nilssen, and G. A. Staff. Order-optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDEs. *SIAM Journal on Scientific Computing*, 29(1):361–375, 2007.

[44] D. A. May, P. Sanan, K. Rupp, M. G. Knepley, and B. F. Smith. Extreme-scale multigrid components within PETSc. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '16, New York, NY, USA, 2016. Association for Computing Machinery.

[45] L. Pareschi and G. Russo. Implicit–explicit Runge–Kutta schemes and applications to hyperbolic systems with relaxation. *Journal of Scientific computing*, 25:129–155, 2005.

[46] W. Pazner and P.-O. Persson. Stage-parallel fully implicit Runge–Kutta solvers for discontinuous Galerkin fluid simulations. *Journal of Computational Physics*, 335:700–717, 2017.

[47] E. G. Phillips, J. N. Shadid, E. C. Cyr, H. C. Elman, and R. P. Pawlowski. Block preconditioners for stable mixed nodal and edge finite element representations of incompressible resistive MHD. *SIAM Journal on Scientific Computing*, 38(6):B1009–B1031, 2016.

[48] M. M. Rana, V. E. Howle, K. Long, A. Meek, and W. Milestone. A new block preconditioner for implicit Runge-Kutta methods for parabolic PDE. *SIAM J. Sci. Comp.*, 43(5):S475–S495, 2021.

[49] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. McRae, G.-T. Bercea, G. R. Markall, and P. H. J. Kelly. Firedrake: automating the finite element method by composing abstractions. *ACM Transactions on Mathematical Software (TOMS)*, 43(3):1–27, 2016.

[50] A. Reisner, L. N. Olson, and J. D. Moulton. Scaling structured multigrid to 500k+ cores through coarse-grid redistribution. *SIAM Journal on Scientific Computing*, 40(4):C581–C604, 2018.

[51] E. Rosseel, T. Boonen, and S. Vandewalle. Algebraic multigrid for stationary and time-dependent partial differential equations with stochastic coefficients. *Numer. Linear Algebra Appl.*, 15(2-3):141–163, 2008.

[52] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[53] M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher. Benchmark computations of laminar flow around a cylinder. In *Flow simulation with high-performance computers II*, pages 547–566. Springer, 1996.

[54] A. Schneebeli and D. Schötzau. Mixed finite elements for incompressible magneto-hydrodynamics. *Comptes Rendus Mathematique*, 337(1):71–74, 2003.

[55] D. Schötzau. Mixed finite element methods for stationary incompressible magneto-hydrodynamics. *Numer. Math.*, 96(4):771–800, 2004.

[56] L. R. Scott and M. Vogelius. Conforming finite element methods for incompressible and nearly incompressible continua. *Lectures in Applied Mathematics*, 22(2), 1985.

[57] B. S. Southworth, O. Krzysik, and W. Pazner. Fast parallel solution of fully implicit Runge-Kutta and discontinuous Galerkin in time for numerical PDEs, Part II: nonlinearities and DAEs, 2021.

[58] B. S. Southworth, O. Krzysik, W. Pazner, and H. D. Sterck. Fast parallel solution of fully implicit Runge-Kutta and discontinuous Galerkin in time for numerical PDEs, Part I: the linear setting, 2021.

[59] C. Taylor and P. Hood. A numerical solution of the Navier-Stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73–100, 1973.

[60] R. Témam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires. II. *Arch. Rational Mech. Anal.*, 33:377–385, 1969.

[61] J. van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM J. Sci. Stat. Comput.*, 7(3):870–891, 1986.

[62] J. Van Lent and S. Vandewalle. Multigrid methods for implicit Runge–Kutta and boundary value method discretizations of parabolic PDEs. *SIAM Journal on Scientific Computing*, 27(1):67–92, 2005.

[63] S. P. Vanka. Block-implicit calculation of steady turbulent recirculating flows. *International Journal of Heat and Mass Transfer*, 28(11):2093–2103, 1985.

[64] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65(1):138–158, 1986.

[65] G. Wanner and E. Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

[66] M. Wathen, C. Greif, and D. Schötzau. Preconditioners for mixed finite element discretizations of incompressible MHD equations. *SIAM Journal on Scientific Computing*, 39(6):A2993–A3013, 2017.

[67] Wikipedia. List of runge-kutta methods. `https://en.wikipedia.org/wiki/List_of_Runge-Kutta_methods`, 2021. [Online; accessed 7-January-2021].

[68] Software used in 'Monolithic multigrid for Implicit Runge–Kutta discretizations of incompressible fluid flow', feb 2022.

[69] S. Zhang. A new family of stable mixed finite elements for the 3D Stokes equations. *Math. Comp.*, 74(250):543–554, 2005.

Chapter 4

Monolithic Multigrid Preconditioners for High-Order Discretizations of the Navier-Stokes Equations

Abstract

[1] Recent years have seen substantial interest in the development of high-order spatial discretizations for the Navier-Stokes equations, using either Scott-Vogelius elements (on suitable meshes) or H(div)-conforming elements to achieve high-order discretizations that strongly enforce the incompressibility constraint. For time-dependent problems, a further complication comes from achieving similar higher-order accuracy for the time-stepping scheme while maintaining optimal cost per time-step, roughly proportional to the number of spatial degrees of freedom. Despite development of these higher-order discretizations, due to the large size and implicit structure of the resulting discrete systems, there has been a clear lack of efficient solvers for these discretizations in the literature. Monolithic multigrid preconditioning has proven to be among the more attractive solver choices for problems with high-order IRK discretizations, as shown in our earlier work [1], where extensions of Vanka-type relaxation schemes

[1]Authors are R. Abu-Labdeh, P.E. Farrell, R.C. Kirby, and S.P. MacLachlan.

are used to develop robust solvers. In this paper, we consider the extension of these monolithic multigrid methods to higher-order H(div)-conforming discretizations, coupled with suitably high-order implicit Runge-Kutta temporal discretizations. We show that, with a suitable definition of the Vanka "patches", we achieve a robust solution algorithm for two- and three-dimensional problems, showing true potential savings for such higher-order discretizations.

Keywords: High-order discretizations, Implicit Runge-Kutta time integration, Monolithic multigrid.

4.1 Introduction

The development of numerical solvers for the incompressible Navier-Stokes equations (NSE) has been of significant interest over the last several decades, due to their wide range of applications [19, 37]. These equations model the continuum flow of viscous fluids, modeling a range of flow phenomena including turbulent flows [26]. They are crucial in many fields of science and engineering, such as the study of flow around an airplane body, weather and climate modeling, modeling of ocean currents, and the flow of blood in the body. Despite their importance, several properties of the NSE make their solution notoriously difficult. First, as a saddle-point problem, we require either an inf-sup stable finite-element method or to use appropriate stabilizations terms to make the problem well-posed [26]. Secondly, when the viscosity is small, the NSE lead to advection-dominated flow problems for which it can be difficult to attain highly accurate velocity approximations [15]. Thirdly, if we seek accurate pressure solutions, we must also consider pressure-robust finite-element schemes, to avoid inaccurate pressure solutions from polluting the velocity solution [46, 32]. Finally, the system takes the form of differential-algebraic equations, making its time integration more difficult than for standard systems of PDEs [36]. There has been substantial and successful effort in the research community to address these discretization difficulties in recent years; however, the development of efficient non-linear and linear solver frameworks for these discretizations has lagged behind. The main goal of this paper is to show that existing monolithic multigrid solver methodologies can be extended to such discretizations in a robust and effective way.

There are several equivalent formulations of the NSE. Throughout this paper, we focus on the incompressible NSE on a bounded domain $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$, in the following form:

$$\begin{aligned}
\rho\left(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}\right) - 2\mu\nabla\cdot\epsilon(\mathbf{u}) + \nabla p &= \mathbf{f} && \text{in } \Omega\times(0,T_f), && (4.1a)\\
-\nabla\cdot\mathbf{u} &= 0 && \text{in } \Omega\times(0,T_f), && (4.1b)\\
\mathbf{u} &= \mathbf{g}_D && \text{on } \partial\Omega\times(0,T_f), && (4.1c)\\
\mathbf{u}(\mathbf{x},0) &= \mathbf{b}(\mathbf{x}) && \text{on } \Omega\times\{t=0\}, && (4.1d)
\end{aligned}$$

where $\mathbf{u}(\mathbf{x},t)$ is the fluid velocity, $p(\mathbf{x},t)$ is the fluid pressure and $\mathbf{f}(\mathbf{x},t)$ is a suitably smooth forcing term. Here, $\mathbf{g}_D$ denotes Dirichlet boundary conditions for the velocity, $\mathbf{b}$ denotes the initial velocity, $\epsilon(\mathbf{u}) = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)$ is the symmetric gradient of $\mathbf{u}$, T_f denotes the final time, while ρ and μ denote the fluid density and viscosity, respectively. For simplicity, we fix the fluid density, ρ, to be 1, while the viscosity, μ, varies in order to investigate problems at different Reynolds numbers, denoted by $\mathrm{Re} = \frac{\rho}{\mu}$ (assuming unit characteristic scales for length and velocity).

Historically, spatial discretization of incompressible NSE dates back at least to the 1960's, with the Marker-and-Cell (MAC) scheme [38], which is still used to this day as a finite-difference discretization method for the incompressible NSE on uniform meshes. Since realistic models of interest may be posed on arbitrarily shaped two- or three-dimensional domains, finite-element methods (FEM) are often a more natural tool, especially for higher-order discretizations. As is typical in fluid-flow models, due to the differing solution spaces that arise for the velocity and pressure, mixed FEM discretization must be used. Depending on the discrete solution spaces and the desired accuracy of the approximation, there are a wide variety of known mixed finite-element approximation spaces to choose from (see, for example, [8]). As usual, a major challenge in using mixed FEM discretizations for any problem is ensuring the chosen combination is inf-sup stable and satisfies appropriate accuracy properties [8, 26]. In order to obtain even more accurate approximations, higher-order discretizations can be used [20]. For NSE, various stable high-order mixed FE spatial discretization spaces, whether classical spaces or more uncommon ones [42], conforming or non-conforming, have been used and analysed (cf. [8, 59, 63, 39]). One commonly used class of these stable high-order mixed spaces are the $\mathbf{H}(\mathrm{div})$-DG non-conforming elements considered here.

An additional challenge in modeling high Reynolds number flows is stability of our approach for high velocity flows. As is well-known for advection-dominated advection-diffusion equations, standard Galerkin FEM approaches can lead to instabilities where non-physical oscillations appear in the numerical solution unless the mesh fully resolves any boundary or interior layers present in the flow. Thus, for simulations at high Reynolds numbers, it is typical to include additional stabilization terms to avoid such oscillations on reasonable meshes. Many such stabilization techniques have been developed for advection-diffusion and Navier-Stokes problems, such as the streamline upwind Petrov-Galerkin (SUPG) [14], (local) projection methods [33] and interior penalty methods. Here, we impose an extension of the interior penalty methods described in [15], commonly known as *Burman stabilization* or the *gradient jump penalty* approach.

Naturally, high-order spatial discretization shows most benefit when coupled with stable high-order temporal discretization [34]. Although there are many choices for temporal discretization of PDEs, we are somewhat limited in our options here. Since the NSE retain a dissipative term, we tend to consider only implicit methods, so that stability criteria do not unduly constrain the choice of time step. Furthermore, we have a preference for A-stable and L-stable integrators, so that any high-frequency errors introduced in timestepping are numerically dissipated. The well-known Dahlquist barrier says that no linear multi-step scheme with order greater than two can be A-stable, leading us to consider Runge-Kutta approaches. Of these, since the NSE (in the form given in (4.1)) are an index-two system of differential-algebraic equations, standard analysis tells us that the *stage order* of the scheme is more important than the usual global order [66]. Thus, even though diagonally implicit Runge-Kutta (DIRK) methods are computationally cheaper than fully implicit methods, they cannot yield high order when applied to DAEs, as their stage order can be no more than two. This leaves only two practical options: various IMEX schemes, where implicit and explicit time integrators are applied to the linear and nonlinear parts of the problem (cf. [4, 18]), and fully implicit Runge-Kutta schemes, that we consider here.

The combination of higher-order spatial finite-element discretizations and higher-order fully implicit Runge-Kutta discretizations leads to challenging nonlinear systems to solve at each time-step, as the solutions for each stage in the Runge-Kutta approximation are coupled with one-another in the discrete equations. There is, however, a long history of the development of effective solvers and preconditioners for similar linear systems, particularly in the case of lower-order discretizations. Block-factorization preconditioners are well-known for the time-stationary and time-dependent NSE [25]. These approaches have been extended to stage-coupled discretizations of parabolic PDEs [62, 48, 54]. One newly developed approach introduced in [61, 60] is constructing a preconditioner similar in structure to those used for backward Euler time integration schemes, but applied to 2×2 blocks corresponding to certain pairings of stages in the system. Another recent approach in block preconditioning of these systems is the distribution of each of the resulting blocks over multiple parallel processes to be solved all-at-once instead of solving them sequentially [50, 52]. Monolithic multigrid preconditioners, in contrast, operate on the fully coupled system. These were originally developed for stationary (or Euler-type temporal discretizations) Stokes and NSE [11, 65, 10, 41, 40, 21], but were extended to IRK discretizations of parabolic equations by Vandewalle and coauthors [64, 57, 9]. Work has also been done to further extend monolithic preconditioning with implicit temporal discretizations to flow problems as in [53], where a matrix-free SVD-based tensor product preconditioner is constructed for problems with the DG method applied.

In [1], we proposed an efficient Newton-Krylov-multigrid solver for the IRK discretizations of several incompressible fluid-flow problems, including Navier-Stokes. This work extends Vandewalle's earlier work, by combining Vanka-style relaxation [65, 41, 40, 2] for fluids with the stage-coupled relaxation ideas proposed in [64]. A notable limitation in the work of [1], however, is that the investigations therein were only for the lowest-order Taylor-Hood discretization of the flow variables. In the present work, we aim to advance the solver from [1] to apply it to high-order pressure-robust discretizations of NSE. To our knowledge, there is very little known work that addresses high-order pressure-robust discretizations of NSE, and much of this focuses on low-order temporal (or steady-state) discretizations (such as [29, 30]).

The outline of this paper is as follows: in section 4.2, we provide the details of the spatial and temporal discretizations considered in this paper. In section 4.3, we present details of the Newton-Krylov-multigrid numerical scheme that we propose here. Then, numerical results for two- and three- dimensional model problems are discussed in section 4.4. Finally, in section 4.5, conclusions and future research possibilities are discussed.

4.2 Higher-order discretization in space and time

In this section, we present the discretization techniques used for the time-dependent incompressible Navier-Stokes problem. In section 4.2.1, we discuss the mixed finite-element formulation to be used. In section 4.2.2, we detail the chosen IRK time-stepping scheme.

4.2.1 Spatial discretizations

Equation (4.1) gives the strong form of the Navier-Stokes equations, which we now convert to weak form in the usual way. We denote the solution spaces as

$$\begin{aligned}\mathbf{u} \in \mathcal{V} = \mathbf{H}_0^1(\Omega) &= \left\{ \mathbf{v} \in \mathbf{H}^1(\Omega) \,\middle|\, \mathbf{v} = \mathbf{0} \text{ on } \partial\Omega \right\}, \\ p \in \mathcal{W} = L_0^2(\Omega) &= \left\{ q \in L^2(\Omega) \,\middle|\, \int_\Omega q \,\mathrm{d}\mathbf{x} = 0 \right\}.\end{aligned}$$

We seek to find a weak formulation of (4.1) such that, for any time, $t \in [0, T_f]$, $(\mathbf{u}(\cdot, t), p(\cdot, t)) \in \mathcal{V} \times \mathcal{W}$. This is done by multiplying the momentum equation by $\mathbf{v} \in \mathcal{V}$ and the constraint equation by $q \in \mathcal{W}$, then integrating by parts to get

$$\begin{aligned}\langle \mathbf{u}_t, \mathbf{v} \rangle + \langle \mathbf{u} \cdot \nabla \mathbf{u}, \mathbf{v} \rangle + 2\mu \langle \epsilon(\mathbf{u}), \epsilon(\mathbf{v}) \rangle - \langle p, \nabla \cdot \mathbf{v} \rangle &= \langle \mathbf{f}, \mathbf{v} \rangle, \qquad \forall \mathbf{v} \in \mathcal{V}, \\ -\langle \nabla \cdot \mathbf{u}, q \rangle &= 0, \qquad\quad\;\; \forall q \in \mathcal{W},\end{aligned} \tag{4.2}$$

where the inner product $\langle \cdot, \cdot \rangle$ denotes spatial integration only.

To spatially discretize the weak formulation in (4.2), we use a mixed finite-element space. We consider a decomposition, Ω_h, of the domain, Ω, of non-overlapping elements, K. In this paper, we focus on triangular elements in two dimensions and tetrahedral elements in three dimensions.

A common choice for discretization of the Stokes equations is the Taylor-Hood mixed finite-element pairing, $\mathbf{P}_{k+1}(\Omega_h)$-$P_k(\Omega_h)$ [63], noted for their ease of implementation and relatively low cost. Although these elements satisfy the inf-sup stability condition for all degrees $k \geq 1$ [13], they are not *pressure-robust* elements since the divergence of vector functions in $\mathbf{P}_{k+1}(\Omega_h)$ are not guaranteed to be contained in the pressure space $P_k(\Omega_h)$. Pressure robustness can be an important property, as it allows us to achieve error estimates for the velocity approximation that are independent of the pressure error [45]. This ensures that, for problems with complicated or large pressures (or large viscosities), the error in the velocity approximation does not get polluted by the (necessarily) large pressure errors in these cases [46, 39].

Another higher-order polynomial pairing that we might consider are the Scott-Vogelius elements, where the velocity is in $\mathbf{P}_{k+1}(\Omega_h)$ and the pressure is in k^{th} degree *discontinuous* Lagrange space, $\mathrm{DG}_k(\Omega_h)$. With this choice of spaces, we have that $\mathrm{div}(\vec{v}_h) \in \mathrm{DG}_k(\Omega_h)$ for every function $\vec{v}_h \in \mathbf{P}_{k+1}(\Omega_h)$, so the space is pressure robust. However, achieving inf-sup stability for these spaces is known to be difficult. The pair is inf-sup stable for $k \geq 4$ on two-dimensional meshes, Ω_h, with no singular vertices [59, 35]. It is also known to be stable for $k \geq d-1$ on d-dimensional simplex meshes that result from a single step of barycentric refinement of any triangular/tetrahedral mesh [68]. These mesh requirements make construction of effective solvers for the resulting discretizations more complicated, as all meshes in a (geometric) multigrid hierarchy must satisfy such conditions. The multigrid method proposed in [29] is effective, but relies on a very expensive relaxation scheme that is forced upon it by the mesh construction needed for stability. Thus, these elements are seen, in practice, to have a very high computational cost and large memory requirements, so we consider an alternative pressure-robust discretization here.

Another family of pressure-robust mixed FEM spaces arise when using $\mathbf{H}(\mathrm{div})$-conforming velocity spaces, where

$$\mathbf{H}(\mathrm{div}, \Omega) = \{\mathbf{u} \in \mathbf{L}^2(\Omega) | \nabla \cdot \mathbf{u} \in L^2(\Omega)\}.$$

Here, on simplices, we use the Brezzi-Douglas-Marini (BDM) elements [12] for the velocity vector field, while the pressure space is chosen to always be $\mathrm{DG}_k(\Omega_h)$. The BDM space of degree k, for $k \geq 1$, is denoted by $\mathbf{BDM}_k(\Omega_h)$. On each element, K, of $\Omega_h \subset \mathbb{R}^d$, we consider vector-valued functions in $\mathbf{P}_k(K)$, where $\mathbf{P}_k(K) = (P_k(K))^d$ and

$$P_k(K) = \left\{u \in C^0(K) : u(\mathbf{x}) \text{ is a polynomial of degree no more than } k\right\}.$$

Since $\mathbf{BDM}_k$ is an $\mathbf{H}(\mathrm{div}, \Omega)$-conforming space, functions in this space have continuous normal components across neigbouring element facets (edges or facets). Following [51, 42], for $\mathbf{v} \in \mathbf{BDM}_k$, we can categorize the degrees of freedom for $\mathbf{v}$ as

1. $\int_f \mathbf{v} \cdot \mathbf{n} q \mathrm{d}s$, $\forall q \in P_k(f)$,

2. $\int_K \mathbf{v} \cdot \mathbf{q} \mathrm{d}x$, $\forall \mathbf{q} \in \mathcal{N}^1_{k-1}(K)$ for $k \geq 2$.

Here, $\mathbf{n}$ is the outward unit normal vector to a facet, f, in Ω_h, while $\mathcal{N}^1_{k-1}(K)$ is the Nédélec function space of the first kind on element K. We note that the first type of DoF leads directly to the normal continuity of vector fields in the BDM spaces.

Since we choose to use an $\mathbf{H}(\mathrm{div}, \Omega)$ conforming velocity space, we apply an interior penalty formulation to penalize jumps in the tangential derivatives of $\mathbf{u}$, which should be in $\mathbf{H}^1_0(\Omega)$, following [32, 56, 23, 43]. To do this, we denote the set of all facets in Ω_h as $\mathcal{F} = \mathcal{F}^i \cap \mathcal{F}^e$, where $\mathcal{F}^i$ are the interior facets of the mesh and $\mathcal{F}^e$ are the boundary facets, meaning $\mathcal{F}^e \subset \partial\Omega_h$. By definition, each $f \in \mathcal{F}^i$ is shared between two elements in the mesh. For such facets, we can define the jump operator for a piecewise smooth function, Φ, across f as $[[\Phi]] = \Phi^+ - \Phi^-$, where Φ^+ and Φ^- are the inward and outward traces of Φ on f, respectively. The average operator is defined as $\{\{\Phi\}\} = \frac{1}{2}(\Phi^+ + \Phi^-)$. Note that if $f \in \mathcal{F}^e$, then we reduce these definitions to $\{\{\Phi\}\} = [[\Phi]] = \Phi$. Following [23, Section 4.2], we use the symmetric interior penalty (SIP) form to stabilize our discretization of (4.1), choosing the jump stabilization parameter to be $\sigma = 10k^2$, as in [43]. This gives us the stabilized form (omitting the scaling by μ) of

$$2\int_\Omega \epsilon(\mathbf{u}) : \epsilon(\mathbf{v}) - \sum_{f\in\mathcal{F}} \int_f \left(\{\{2\epsilon(\mathbf{u})\}\} : \{\{\mathbf{v} \otimes \mathbf{n}\}\} + \{\{2\mathbf{u} \otimes \mathbf{n}\}\} : \{\{2\epsilon(\mathbf{v})\}\}\right)$$
$$+ \sum_{f\in\mathcal{F}} \frac{\sigma}{h_f} \int_f \{\{2\mathbf{u} \otimes \mathbf{n}\}\} : \{\{2\mathbf{v} \otimes \mathbf{n}\}\}.$$

To account for the non-homogeneous Dirichlet boundary condition, $\mathbf{u} = \mathbf{g}_D$ on $\partial\Omega$, the following two terms are added to the above formulation,

$$-\sum_{f\in\mathcal{F}^e} \frac{\sigma}{h_f} \int_f \mathbf{g}_D \cdot \mathbf{v} + \sum_{f\in\mathcal{F}^e} \int_f \left(\mathbf{g}_D \otimes \mathbf{n}\right) \cdot \left(2\epsilon(\mathbf{v})\right).$$

While this stabilization is sufficient for the Stokes equations, we must also deal with the second instance of $\nabla\mathbf{u}$ in (4.1), coming from the advective term in the Navier-Stokes equations. Here, we integrate by parts on $\langle \mathbf{u} \cdot \nabla\mathbf{u}, \mathbf{v}\rangle$ to get $-\langle \mathbf{u}, \mathrm{div}\,(\mathbf{v} \otimes \mathbf{u})\rangle$

plus boundary integrals on each element (since only the normal components of $\mathbf{u}$ and $\mathbf{v}$ are continuous across element edges). Making similar choices to those above, we arrive at the terms

$$\frac{1}{2}\sum_{f\in\mathcal{F}^i}\int_f [[(\mathbf{u}\cdot\mathbf{n}+|\mathbf{u}\cdot\mathbf{n}|)\mathbf{u}]]\cdot[[\mathbf{v}]] + \frac{1}{2}\sum_{f\in\mathcal{F}^e}\int_f (\mathbf{u}\cdot\mathbf{n}+|\mathbf{u}\cdot\mathbf{n}|)\mathbf{u}\cdot\mathbf{v} \\ + \frac{1}{2}\sum_{f\in\mathcal{F}^e}\int_f (\mathbf{u}\cdot\mathbf{n}-|\mathbf{u}\cdot\mathbf{n}|)\mathbf{g}_D\cdot\mathbf{v}.$$

Pairing the velocity space of $\mathbf{BDM}_{k+1}(\Omega_h)$ with the discontinuous Lagrange space, $\mathrm{DG}_k(\Omega_h)$ leads to an inf-sup stable pairing for the Navier-Stokes equations. Since divergences of vector fields in $\mathbf{BDM}_{k+1}(\Omega_h)$ are always in $\mathrm{DG}_k(\Omega_h)$, these spaces also lead to a pressure-robust discretization. Error analysis of this discretization is presented in [58].

Here, we also seek a discretization that is stable in the advection-dominated regime, when the Reynolds number is large relative to the mesh. In order to avoid additional non-physical oscillations in the velocity approximation, we follow [43, 15] in adding the following stabilization term in the discrete formulation,

$$\sum_{f\in\mathcal{F}^i}\int_{\partial\Omega}\frac{\sigma\beta_f h_f^2}{2}[[\nabla\mathbf{u}\cdot\mathbf{n}]]:[[\nabla\mathbf{v}\cdot\mathbf{n}]],$$

where σ is a mesh-independent parameter, chosen here to be 3×10^{-3} as in [15], and each facet, f, of K has a diameter h_f. Here, β_f is the average of the face-averaged velocity norms on f, which we simply lag in our time-stepping, using the computed velocity at the previous time step, to avoid introducing additional nonlinearities.

In all our numerical results, we seek an approximation of $(\mathbf{u}, p)$ over Ω_h in the mixed finite-element space $\mathcal{V}_h\times\mathcal{W}_h = \mathbf{BDM}_{k+1}(\Omega_h)\times\mathrm{DG}_k(\Omega_h)$ on simplices. Defining $\vec{\mathbf{u}}(t)$ and $\vec{p}(t)$ to be the (time-dependent) coefficients of $\mathbf{u}$ and p in the finite-element basis, this leads to a nonlinear coupled system written as

$$\begin{bmatrix} Z\vec{\mathbf{u}}_t \\ 0\end{bmatrix} + \begin{bmatrix} N(\vec{\mathbf{u}}) \\ 0\end{bmatrix} + \begin{bmatrix} C & B \\ B^T & 0\end{bmatrix}\begin{bmatrix}\vec{\mathbf{u}} \\ \vec{p}\end{bmatrix} = \begin{bmatrix} Z\vec{\mathbf{f}} \\ 0\end{bmatrix}, \tag{4.3}$$

where $N(\vec{\mathbf{u}})$ represents the discretization of the advection term, Z and C are the mass and stiffness matrices respectively, B is the weak gradient operator and $\vec{\mathbf{f}}$ is the vector of coefficients of the interpolant of $\mathbf{f}$ in $\mathcal{V}_h$.

4.2.2 Runge-Kutta discretizations

As stated above, among the many time integrators available, our main focus in this work is the use of higher-order temporal discretizations. While many high-order linear multi-step methods exist, these lack the desired stability properties, as seen in the second Dahlquist barrier theorem for BDF schemes [66]. Instead, we consider high-order Runge-Kutta methods. Unlike multi-step methods, Runge-Kutta schemes are *multi-stage*, where multiple stage values are used to find the approximate solution at each timestep. A general r-stage Runge-Kutta method applied to the system of ordinary differential equations: $u^{'}(t) = f(u(t), t)$ is given by

$$\begin{aligned} k_i &= f\left(u^n + \Delta t \sum_{j=1}^{r} a_{ij} k_j, t^n + c_i \Delta t\right), \text{ for } i = 1, 2, \ldots, r, \\ u^{n+1} &= u^n + \Delta t \sum_{j=1}^{r} b_j k_j. \end{aligned} \tag{4.4}$$

Here, the coefficients are the stage nodes, c_i, the weights, b_j, and the Runge-Kutta matrix, $A = [a_{ij}]$. These form the Butcher tableau for a given method [16, 17]. To guarantee consistency of the schemes, we require $\sum_{j=1}^{r} b_j = 1$ and $\sum_{j=1}^{r} a_{ij} = c_i$, for $i = 1, 2, \ldots, r$. The set $\{k_i\}_{i=1}^{r}$ represents the r stage values, which approximate $u'(t)$ at the stage times, $t^n + c_i \Delta t$, while u^n denotes the approximation at time $t^n = t^0 + n\Delta t$. Depending on the non-zero pattern of the matrix A, Runge-Kutta methods can be categorized into either explicit schemes, where $a_{ij} = 0 \; \forall j \geq i$, or implicit schemes, when $\exists j \geq i$ with $a_{ij} \neq 0$. Implicit schemes can also be sub-categorized into either diagonally implicit schemes, where $a_{ij} = 0 \; \forall j > i$, or fully implicit, when $\exists j > i$ such that $a_{ij} \neq 0$.

To discretize (4.1) fully, we apply the full IRK method to the semidiscrete system in (4.3). We note that (4.3) takes the form of a system of differential algebraic equations (DAE), for which the methodology remains well-defined. The resulting fully discretized system can be written as

$$Z\vec{k}_i^{(\mathbf{u})} + N\left(\vec{\mathbf{u}}^n + \Delta t\sum_{j=1}^{r} a_{ij}\vec{k}_j^{(\mathbf{u})}\right) + \Delta t\sum_{j=1}^{r} a_{ij}\left(C\vec{k}_j^{(\mathbf{u})} + B\vec{k}_j^{(p)}\right) = Z\vec{\mathbf{f}}_i^n - C\vec{\mathbf{u}}^n - B\vec{p}^n,$$

$$\Delta t\sum_{j=1}^{r} a_{ij}B^T\vec{k}_j^{(\mathbf{u})} = -B^T\vec{\mathbf{u}}^n,$$

for $1 \leq i \leq r$, as discussed, for example, in [1]. We note that when a fully implicit Runge-Kutta scheme is chosen, this represents a fully coupled system of nonlinear equations, coupling the stage values across all stages in the Runge-Kutta discretization.

The choice of which Runge-Kutta method to use depends heavily on the needs of the mathematical problem at hand, in terms of stability, accuracy, and computational cost. For a given method, we let $r(z)$ denote the *stability function* produced by applying the method to the Dahlquist test problem, $u' = \lambda u$ for $\lambda \in \mathbb{C}$, with $u^{n+1} = r(\lambda\Delta t)u^n$. Using this, we define the domain of stability of the method to be the region in the complex plane where $|r(z)| \leq 1$. When the entire left-half of the complex plane is included in the domain of stability of a scheme, it is said to be *A-stable*. If an A-stable scheme also satisfies $\lim_{z\to-\infty}|r(z)| = 0$, then it is also *L-stable*. Since $r(z)$ is a polynomial for explicit RK schemes, only implicit RK schemes (where $r(z)$ is a rational function) can be A- or L-stable.

When measuring the accuracy of an approximate solution, many tools rely on either the local truncation error, which is the error generated by a scheme in a single time step comparing the approximation at time t^{n+1} to that at t^n, or the global error, which is the accumulated error in the approximation over all timesteps taken. A scheme is said to have an error order of p if the error is bounded by a constant (that can depend on the analytic solution, $u(t)$, and on properties of $f(u,t)$) multiplied by $(\Delta t)^p$. For some IRK schemes, the maximum global error can be as much as twice the number of the scheme's stages, in contrast to explicit schemes where the order cannot exceed the number of stages. Although schemes with higher-order global error are generally desirable, for stiff differential equations or DAEs, the *stage order* of a Runge-Kutta method is more important in determining the accuracy of the scheme [66]. The stage order of a method is given by $\min\{q,p\}$, where q is determined from bounding the approximation to $u(t^n + c_i\Delta t)$ of stage i by a constant that depends on $f(u,t)$

and $u(t)$ times $(\Delta t)^{q+1}$.

Among IRK schemes, we have two main families to choose from. Diagonally implicit schemes offer decreased computational cost (since they can be solved by forward substitution for one stage at a time), but are known to possess stage order of at most two [66]. Thus, while they have attractive computational properties, they are unsuitable for high-order integration of DAEs, as we need here. Instead, we consider families of fully implicit schemes that are both A- and L-stable, following the failure of the only A-stable Gauss-Legendre scheme to effectively integrate a low-order spatial discretization of the Navier-Stokes equations in [1]. Here, we choose to consider only the RadauIIA IRK schemes, which have global order of $p = 2r - 1$ and stage order equalling the number of stages, r [36, 66]. One trade-off that we will consider in the numerical results is that between using a higher-order scheme (more stages) with larger timesteps and using a low-order scheme with smaller timesteps. Using RadauIIA, which gives the best-possible order for an r-stage scheme applied to a system of DAEs like we have, allows us to make the best-possible comparison of this trade-off.

Although fully implicit RK schemes are a powerful discretization tool, they come with two main disadvantages. The first is their high computational cost compared to multistep or diagonally implicit RK schemes. As stated earlier, this is because using fully implicit RK methods requires the solution of a very large non-linear system of equations at each timestep, including coupling between the RK stages. For diagonally implicit schemes, in contrast, we can solve for the stages sequentially, even reusing some parts of standard linear (and nonlinear) solvers for each stage, cutting down on the overall cost of solution. Generally, for fully implicit RK schemes, such simplifications cannot be used. Another downside to IRK methods is that the resulting systems of linearized equations produced at each timestep are usually nonsymmetric, due to the nonsymmetry of the Butcher matrix, A. This greatly limits the classes of solvers to choose from, although we note that this limitation is not as significant for the Navier-Stokes equations, which are already nonsymmetric.

4.3 Linear and Nonlinear solvers

In the previous section, after fully discretizing (4.1), we arrive at a nonlinear coupled system that must be solved at each time step. Thus, we use Newton's method to linearize the system for the stage values $\vec{\mathbf{k}}$. This produces a sequence of linear systems of equations that we solve using a Krylov method, preconditioned by monolithic multigrid. In this section, we discuss details of the resulting Newton-Krylov-multigrid solver.

4.3.1 Newton linearization

Denoting the nonlinear discretized system at time step n to be $F(\vec{\mathbf{k}}^{n}) = 0$, then the i^{th} Newton iteration can be written as

$$F(\vec{\mathbf{k}}^{n,\ell+1}) \approx F(\vec{\mathbf{k}}^{n,\ell}) + \mathbf{J}(\vec{\mathbf{k}}^{n,\ell})\delta\vec{\mathbf{k}}^{n,\ell} = \mathbf{0},$$

where $\delta\vec{\mathbf{k}}^{n,\ell} = \vec{\mathbf{k}}^{n,\ell+1} - \vec{\mathbf{k}}^{n,\ell}$ is the Newton direction and $\mathbf{J}(\vec{\mathbf{k}}^{n,\ell})$ is the Jacobian of the system at the current approximation. Here, since we apply Newton's method at each timestep, we will always use the stage approximation from the previous time step as the initial guess for Newton's method at the current time step, meaning $\vec{\mathbf{k}}_0^{n} = \vec{\mathbf{k}}^{(n-1)}$. We continue this iteration until the nonlinear residual norm, $\|F(\vec{\mathbf{k}}^{n,\ell}) + \mathbf{J}(\vec{\mathbf{k}}^{n,\ell})\vec{\mathbf{k}}^{n,\ell}\|$, is below a given tolerance, at which point we determine $\vec{\mathbf{k}}^{n} = \vec{\mathbf{k}}^{n,\ell} + \delta\vec{\mathbf{k}}^{n,\ell}$.

For each linear iteration, we use an inexact solver, based on multigrid-preconditioned FGMRES, that is run to a given stopping tolerance. For this, we use the Eisenstat-Walker stopping criteria [24]. For any given timestep and linear iteration, the step $\delta\vec{\mathbf{k}}^{n,\ell}$ is required to satisfy the following condition:

$$\|F(\vec{\mathbf{k}}^{n,\ell}) + \mathbf{J}(\vec{\mathbf{k}}^{n,\ell})\delta\vec{\mathbf{k}}^{n,\ell}\| \leq \eta_\ell \|F(\vec{\mathbf{k}}^{n,\ell})\|,$$

where $\eta_\ell \in [0, 1)$ is called the forcing term and is carefully chosen based on the convergence of the method.

4.3.2 Monolithic multigrid-preconditioned FGMRES

The now linearized system is nonsymmetric, so we use FGMRES for the outer Krylov method. We note that we primarily use FGMRES not for its flexibility (the ability to change the preconditioner between iterations), but for its efficiency in the setting considered here, where we have an expensive preconditioner to apply (as described below) but sufficient memory for the extra vector storage required by FGMRES over classical right-preconditioned GMRES.

While block-diagonal and block-triangular preconditioners are commonly chosen when solving either stationary fluid models or time-dependent models discretized with BDF-type schemes [26, 67], the tight coupling between stage values in the IRK schemes considered here suggests that an all-at-once preconditioner may be more successful (and, moreover, avoids the need to approximate Schur complements that couple the stages). Hence, we follow the monolithic approach as found in [65, 10] and many other papers, extending the monolithic multigrid preconditioner developed in [1] for Taylor-Hood style discretizations. This preconditioner relies on an overlapping Schwarz relaxation scheme that is viewed as an extension of the usual Vanka relaxation scheme first proposed in [65], that preserves the stage coupling of the system. As described below, we alter the Vanka patches to account for the change in discretization spaces, but otherwise follow the methodology of [1] for applying monolithic multigrid to IRK discretizations of systems of PDEs.

Since multigrid solvers rely on having several levels of refinements of a given coarsest grid, information must be passed between these grids using transfer operators. We define the interpolation operator for a single stage to be the following block matrix:

$$P = \begin{bmatrix} P_{\vec{\mathbf{u}}} & \\ & P_p \end{bmatrix},$$

where the blocks $P_{\vec{\mathbf{u}}}$ and P_p are the standard finite-element interpolation operators for the velocity and pressure spaces, respectively. When using this for IRK discretizations with more than one stage, the interpolation operator is then defined as $I_r \otimes P$, where I_r is an $r \times r$ identity matrix. This builds a block-diagonal interpolation operator that applies $P_{\vec{\mathbf{u}}}$ and P_p independently to the stage values of velocity and pressure at each stage, without introducing any coupling in the interpolation process. The restriction operator is defined to be the transpose of interpolation. To define the coarse-grid system operator, we rediscretize the differential equation on the coarser grids. The coarsest grid system in the hierarchy is then solved using the direct LU method.

4.3.3 Vanka relaxation

The only remaining component of a standard multigrid algorithm for the IRK coupled system needed is a relaxation scheme. Since the fully discrete system of (4.1) is a saddle-point problem, more advanced relaxation methods than Jacobi or Gauss-Seidel (as are commonly used for scalar diffusion equations) must be used. Here, we use Vanka-type relaxation [65]. In general, Vanka relaxation schemes are overlapping Schwarz methods. They can be applied either multiplicatively or additively. In our work, Vanka is used additively, making it more suitable for parallelization.

On any given mesh in the multigrid hierarchy, Vanka relaxation relies on dividing the mesh into small subdomains called *patches*. All degrees of freedom are decomposed into overlapping patches, where the l^{th} patch is denoted by S_l, and the set of all DoFs in the mesh is given by $S = \cup_{l=1}^{N}\{S_l\}$, where N is the total number of patches considered. For our problems, each patch is constructed to contain both pressure and velocity DoFs. We use P_l to denote the *extension operator* from the DoFs in S_l to all of S, where each column of P_l is a canonical unit vector, with a single unit entry in the row corresponding to the global numbering of the DoF. Using $J\vec{\mathbf{k}} = \vec{\mathbf{F}}$ to denote the linear system to be solved, one iteration of Vanka relaxation can be written as

$$\vec{\mathbf{k}} \leftarrow \vec{\mathbf{k}} + \omega \sum_{i=1}^{N} P_l (P_l^T J P_l)^{-1} P_l^T (\vec{\mathbf{F}} - J\vec{\mathbf{k}}).$$

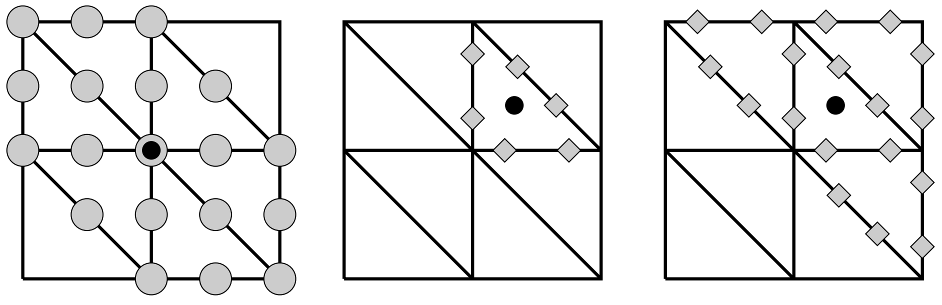
As in [1], good choices of the relaxation weight, ω, are critical to achieving good performance. Here, instead of choosing a single weight, we use several iterations of a Vanka-preconditioned Chebyshev iteration as the relaxation scheme on each level. To achieve best-possible performance, the Chebyshev polynomials are chosen with endpoints of the associated interval that are tuned by hand, based on preliminary experiments.

There are two common ways of forming these patches for discretizations of the Navier-Stokes equations, commonly termed "pressure-centric" or "element-centric" patches [40, 2]. Pressure-centric patches are generally built so that each patch contains a single pressure DoF and all velocity DoFs in the closure of the cells adjacent to that pressure. In each of these patches, the velocity DoFs generally have a nonzero connection to the pressure DoF in the divergence constraint matrix of the problem. This decomposition leads to the number of patches, N, being the same as the number of pressure DoFs in the discretization. The pressure-centric decomposition is very common for many discretization spaces, such as the Taylor-Hood setting [1, 7, 30, 3, 47]. On the other hand, element-centric patches contain all DoFs in each element, labeled by l [40, 6, 47, 31]. Such patches may contain more than one pressure DoF, and N will equal the number of elements in the discretized mesh. In [2], it was shown that using element-centric patches does not always yield scalable results for the $\mathbf{BDM}_1(\Omega_h) \times \mathrm{DG}_0(\Omega_h)$ discretization of the Stokes equations across varying mesh sizes. Thus, another choice of element-centric patches was proposed, termed "extended Vanka" patches. These are defined by extending the element-centric patch for each element, l, to include all velocity DoFs on elements that share a face with l. Although using these patches adds to the computational cost in comparison to standard element-centric patches, the scalability of the resulting solver is shown to be worth this added expense [2]. Figure 4.1 sketches the various patches discussed here. We note that one advantage of the extended patches is that they do not change with the order of the spatial discretization, since all $\mathrm{DG}_k(\Omega_h)$ pressures have the same topological relationship, internal to each element.

4.4 Numerical Results

For all results presented in this paper we use RadauIIA for the temporal integration, varying the number of stages from 1 to 4 stages. For details on RadauIIA methods and their Butcher Tables see [66]. For the mixed FEM spaces, we show results using $\mathbf{BDM}_{k+1}(\Omega_h) \times \mathrm{DG}_k(\Omega)$ for $k = \{2, \ldots, 5\}$, on two- or three-dimensional simplices. Similar results to those found in this paper can be seen using other L-stable IRK schemes, such as LobattoIIIC, and/or other $\mathbf{H}(\mathrm{div}, \Omega)$-conforming spaces, such as $\mathbf{RT}_{k+1}$ for the velocity.

For the implementation, all our numerical results were produced using Firedrake [55] for the spatial FE discretizations and Irksome [27] for the IRK discretization. For all



○ velocity DoF ($\mathbf{P}_2$) ● pressure DoF (P_1 or DG_0) ◆ velocity DoF (BDM_1)

Figure 4.1: Left: a pressure-centric Vanka patch for the Navier-Stokes equations, consisting of $\mathbf{P}_2(\Omega_h)$ velocity DoFs, and one $P_1(\Omega_h)$ pressure DoF. Center: an element-centric Vanka patch consisting of $\mathbf{BDM}_1(\Omega_h)$ velocity DoFs, and one $\mathrm{DG}_0(\Omega_h)$ pressure DoF. Right: an extended Vanka patch consisting of $\mathbf{BDM}_1(\Omega_h)$ velocity DoFs, and one $\mathrm{DG}_0(\Omega_h)$ pressure DoF.

linear and nonlinear solvers, we use PETSc [5], while the extended Vanka relaxation scheme is implemented using PCPATCH [28]. Since the fully discretized problems below can contain several million DoFs, we construct our solver in a way that is naturally parallelizable, and stress the need for the chosen discretization and solver software to be easily parallelizable. In this setting, the coarsest mesh is distributed almost evenly across the specified cores and then refined in parallel. We note that in order for the relaxation scheme to be efficient in a parallel distribution of the mesh, it is important to include a node-distance-2 halo at each processor boundary. This is done so that the Vanka patches on each node have all the necessary information to calculate the residual included in the relaxation iteration [44].

In the following, we present results for two problems: the two-dimensional Chorin vortex decay problem [19] and the three-dimensional lid driven cavity problem [22]. All tests included in this paper were done on a single machine with 2 Intel(R) Xeon(R) CPU E5-2650 v2 CPUs, each with 8 physical cores, but 16 virtual cores, at 2.60GHz. The total RAM size is 128GB.

4.4.1 Two-dimensional Chorin vortex decay

For the first model, we consider a Chorin vortex decay problem, first presented in [19] on a unit square domain $\Omega = (0,1)^2$. The manufactured exact solution considered is

$$\mathbf{u} = \begin{bmatrix} -\sin(\pi y)\cos(\pi x)e^{-2\pi^2 t} \\ \sin(\pi x)\cos(\pi y)e^{-2\pi^2 t} \end{bmatrix} \text{ and } p = -\frac{\mathrm{Re}}{4}e^{-4\pi^2 t}(\cos(2\pi x) + \cos(2\pi y)).$$

For this problem, we divide the unit square domain into a coarsest grid constructed of a uniform 4×4 quadrilateral grid, where each quadrilateral cell is cut into 2 triangles by drawing the edge from the top left vertex of each cell to the bottom right vertex.The coarsest mesh is then refined ℓ times, where $\ell \in \{1,2,3,4\}$. This results in a finest grid of $N \times N$ quadrilaterals cut in the same way into triangles, where $N = 2^{\ell+2}$, containing various counts of degrees of freedom depending on the degree of the finite-element space. When using 2 or 3 stages of RadauIIA, we use 1 core for $\ell = 2$, 4 cores for $\ell = 3$ and 16 cores for $\ell = 4$, while for 4 stage RadauIIA, we use 1 core for $\ell = 1$, 4 cores for $\ell = 2$, and 16 cores for $\ell = 3$. The DoFs counts **per stage** are summarized in Table 4.1.

		ℓ			
		1	2	3	4
k	1	-	5472	21696	86400
	2	-	10368	41216	164352
	3	4240	16800	66880	266880
	4	6240	24768	98688	-
	5	8624	34272	136640	-

Table 4.1: Number of DoFs per stage as a function of refinement level, ℓ, and degree, k, of the finite-element space used, $\mathbf{BDM}_{k+1}(\Omega_h) \times \mathrm{DG}_k(\Omega_h)$.

The initial condition chosen is simply the manufactured solution evaluated at $t = 0$, and we integrate to a final time of $T_f = 0.25$. We vary the timestep taken, investigating the impact that it has on the accuracy of the discrete solution at the final time. As we discuss later in this section, stopping tolerances for these systems are somewhat tricky, since (to our knowledge) there is no theory showing particular stopping tolerances to use. Through experimentation, we found the optimal nonlinear stopping criteria for these results is to require the L_2 norm of the nonlinear residual at each timestep be reduced below N^{-4}.

We found that accelerating the Vanka relaxation scheme with Chebyshev polynomials of the first kind on a chosen interval produces scalable results. Here, we use 2 pre- and post-relaxation sweeps. The default polynomial interval is set to be $[\frac{\lambda}{4}, \lambda]$ where λ is the estimated largest eigenvalue of the Vanka-Chebyshev-preconditioned matrix (estimated using PETSc's default estimation, based on GMRES). We record relative $L^2(\Omega)$-norm errors for velocity and absolute $L^2(\Omega)$-norm for pressure, the total number of linear and nonlinear iterations to converge for each timestep, averaged over all timesteps, and the total computational time to solution in minutes.

First, we study the effect of the spatial and temporal degrees, fixing the Reynolds number to be 1. For each time integrator (number of stages), we fix Δt based on N and T_f, but allow this to vary slightly between integrators. For 2 stages, we take $\Delta t = \frac{T_f}{4N}$, while we take $\Delta t = \frac{T_f}{2N}$ for 3 stages, and $\Delta t = \frac{T_f}{N}$ for 4 stages. Table 4.2 shows results for these three integrators and varying spatial discretization order, k. We clearly see that, when we fix the number of IRK stages, there are limited returns for increasing the spatial degree beyond the number of stages. This is expected because of the stage order accuracy of the time integrators. For example, using 2 stages, going from $k = 1$ to $k = 2$ for $\ell = 4$, requires significantly more CPU time (an increase of about 2.5×), but leads to a significant improvement in the velocity approximation. An even larger increase in CPU time is seen going to $k = 3$, but no notable change is seen in the velocity or pressure errors. Thus, for this problem with 2 stages, the pairing $\mathbf{BDM}_3(\Omega_h) \times \mathrm{DG}_2(\Omega_h)$ seems is optimal. We see similar behaviour in the accuracies for 3 and 4 stages. We note, however, that solver performance is largely independent of the number of stages or order of the spatial discretization. For all problems, we see between 4 and 5.3 nonlinear iterations per timestep (noting that we use inexact Newton, so expect to see slightly larger nonlinear iteration counts than may otherwise be typical), and between 10 and 26 linear iterations per timestep.

Next, we study the effects of the time step size on the solver performance, as well as the accuracy as we vary the number of IRK stages and spatial degree, in Table 4.3. For this, we fix the "optimal" pairings for stages and spatial degree found in Table 4.2, taking $k = r$. For a fixed number of stages and degree, we generally see the expected increase in accuracy as smaller time steps are taken. However, taking too small of a temporal step leads to a decrease in accuracy, likely due to accumulating errors from an increasing number of time steps. For example, using $\mathbf{BDM}_4(\Omega_h) \times \mathrm{DG}_3(\Omega_h)$ with 3-stage RadauIIA, $\Delta t = \frac{T_f}{N}$ gives the lowest velocity error at $\ell = 4$, with about a

	ℓ	L_2 error $\mathbf{u}$	L_2 error p	time	linear	nonlinear
Rad(2)			$k = 1$			
	2	2.664×10^{-4}	8.736×10^{-4}	6.2	10.0	4.0
	3	2.184×10^{-5}	2.351×10^{-4}	10.9	13.3	4.0
	4	2.279×10^{-6}	5.636×10^{-5}	44.4	18.0	4.9
			$k = 2$			
	2	2.385×10^{-4}	3.239×10^{-4}	7.2	13.9	4.0
	3	5.627×10^{-6}	2.702×10^{-5}	41.5	16.2	4.0
	4	4.350×10^{-7}	4.428×10^{-6}	102.4	21.3	4.4
			$k = 3$			
	2	1.553×10^{-5}	1.186×10^{-5}	21.3	14.4	4.0
	3	2.972×10^{-6}	3.797×10^{-6}	95.4	18.3	4.6
	4	6.382×10^{-7}	3.996×10^{-7}	357.3	21.4	4.7
Rad(3)			$k = 2$			
	2	3.258×10^{-5}	2.832×10^{-5}	12.2	15.7	4.0
	3	1.887×10^{-6}	3.400×10^{-6}	30.3	19.7	4.2
	4	1.083×10^{-7}	5.734×10^{-7}	160.5	25.8	4.9
			$k = 3$			
	2	6.894×10^{-6}	4.106×10^{-6}	13.0	16.6	4.2
	3	4.611×10^{-7}	3.285×10^{-7}	51.9	20.1	4.7
	4	2.165×10^{-8}	4.525×10^{-8}	302.2	25.0	5.3
			$k = 4$			
	2	3.333×10^{-6}	2.769×10^{-6}	58.6	16.5	4.5
	3	1.262×10^{-7}	1.744×10^{-7}	96.8	19.6	4.9
	4	3.108×10^{-8}	1.060×10^{-8}	492.5	24.1	5.1
Rad(4)			$k = 3$			
	1	4.334×10^{-5}	1.908×10^{-5}	3.9	14.2	4.1
	2	1.595×10^{-6}	1.114×10^{-6}	13.4	19.0	4.5
	3	1.117×10^{-7}	9.549×10^{-8}	70.9	22.4	5.0
			$k = 4$			
	1	1.272×10^{-5}	7.461×10^{-6}	5.1	15.1	4.3
	2	1.168×10^{-6}	1.012×10^{-6}	22.9	18.7	4.8
	3	3.096×10^{-8}	4.431×10^{-8}	143.4	21.7	5.0
			$k = 5$			
	1	4.776×10^{-6}	5.705×10^{-6}	8.1	14.0	4.1
	2	4.631×10^{-7}	4.601×10^{-7}	32.0	17.3	4.9
	3	2.325×10^{-8}	4.514×10^{-8}	312.0	19.8	5.0

Table 4.2: Numerical results for several refinement levels showing the effect of various degrees of spatial finite-element discretization with different numbers of stages of the temporal IRK discretization on the velocity and pressure L_2 errors, and average iteration counts and solver times.

10% increase when we halve Δt. Again, we see very little direct dependence on solver performance with Δt or degree of the discretization, still requiring between 4 and 5.3 nonlinear iterations per time step, and between 13 and 28 nonlinear iterations per time step. We note that the parallelism here is fixed based on the spatial refinement level, ℓ, so that the general expectation is for CPU times to double for a fixed discretization when we halve Δt, since we have twice as many time steps to solve for.

An expected observation that is clearly shown in these results is the fact that increasing the spatial and temporal degrees increases the accuracy of the resulting approximation, but at the expense of an increase in computational cost per nonlinear time step. This is because the size of the systems that are solved at each time step increase with more stages and the increase in the degree of the polynomial order within the FEM approximations. A clear question, then, is where is the trade-off point? A good example of this is noticing that the best error at $\ell = 4$ for the 2-stage RadauIIA using $\mathbf{BDM}_3(\Omega_h) \times \mathrm{DG}_2(\Omega_h)$ using $\Delta t = \frac{T_f}{2N}$ is surpassed at $\ell = 3$ for 3-stage RadauIIA using $\mathbf{BDM}_4(\Omega_h) \times \mathrm{DG}_3(\Omega_h)$ with $\Delta t = \frac{T_f}{N}$. Thus, we see a clear gain from increasing both temporal and spatial orders, but note that this represents an overall *decrease* in the computational cost, taking less CPU time on a quarter of the cores.

Finally, we study the effect increasing advection may have on the accuracy and solver performance by changing the Reynolds number. For this, we fix the discretization pairings and best time step sizes from the previous experiments and only vary Re to be 1, 10 or 100. In Table 4.4, we clearly see that the change of Reynolds number has almost no effect on the the accuracy of both pressure and velocity, which is more-or-less expected since the solution only depends weakly on Re in this case. Nonetheless, for similar examples in preliminary work, not including the stabilization terms led to notable degradation in accuracies, so these are encouraging results. Further, we see very little dependence of the solver performance on the Reynolds number in this setting.

4.4.2 Three-dimensional lid-driven cavity

Next, we consider a three-dimensional lid-driven cavity problem on a unit cube domain, $\Omega = (0,1)^3$. On the top face (where $z = 1$), the flow is driven from left to right by the imposed velocity $\mathbf{u} = (1,0,0)^T$, while fixing $\mathbf{u} = (0,0,0)^T$ on all other faces.

For this model, we consider three tests where, in each, a fixed coarsest grid is refined twice. The coarsest grid is always constructed by cutting the unit cube domain into a uniform hexahedral mesh, then cutting each hexahedral element into 6 tetrahedra. We now use ℓ to denote the dimension of the coarsest mesh, with $\ell = 1$ denoting the smallest mesh, built by refining a $1 \times 1 \times 1$ mesh twice, while $\ell = 2$ denotes a mesh built by refining a $2 \times 2 \times 2$ mesh twice, and $\ell = 3$ denotes a mesh built by refining a $3 \times 3 \times 3$ mesh twice. These choices (and the limitation on number of stages used and spatial degree considered) are made due to memory limitations for 3D simulations, as the increase in finest grid sizes would require more cores (and the same memory per core) for parallel computation. Table 4.5 summarizes the DoFs counts per stage for the tests considered.

We use the same spatial discretization here and RadauIIA integration with 1, 2 or 3 stages. We integrate until final time $T_f = 0.25$, fixing the timestep as $\Delta t = \frac{T_f}{4N}$ where $N = 4\ell$. Through experimentation, we found the optimal nonlinear stopping criteria for this test model is to require the L_2 norm of the nonlinear residual at each timestep be reduced below $0.01N^{-2}$. A zero pressure initial condition is used while $\mathbf{u} = (1, 0, 0)^T$ is the velocity initial condition. We also choose to accelerate the Vanka relaxation with Chebyshev polynomials of the first kind, using 3 pre- and post-relaxation sweeps on each level. The polynomial interval is set by estimating the largest eigenvalue of the Vanka-preconditioned matrix, λ, then choosing the Chebyshev interval to be $[0.25\lambda, 1.05\lambda]$. We record average linear and nonlinear iterations per timestep, and the total computational time to solution in minutes.

Fixing the Reynolds number to be 1 and spatial discretization order $k = 2$, Table 4.6 shows the effect of varying temporal degrees on solver computational cost. Note that $\star$ indicates an out-of-memory error for this test. In these results, we clearly see that the average iteration counts are independent of the number of stages chosen. We see a very reasonable number of nonlinear iterations per timestep for all tests, and between 10 and 15 total linear iterations per timestep. However, using the same number of parallel cores as we increase the mesh size and number of stages leads to a notable increase in CPU times.

Next, we study the effects of varying the spatial degree on solver performance, while continuing to fix the Reynolds number to be 1 and using 2 stages of RadauIIA.

These results are summarized in Table 4.7. Again, we see reasonable nonlinear iterations per timestep (less than 3). Although CPU times do increase as we increase the spatial order, we still maintain overall reasonable counts for the total number of linear iterations per timestep, ranging from about 6 iterations to 13 iterations.

Finally, we present results showing the effect of increasing the advection term on solver performance, by varying the Reynolds number to be 1, 10 or 100. Here, we fix the spatial discretization order to be $k = 2$ and only use 2 stages of the RadauIIA integrator. As in the Chorin vortex decay problem, we can clearly see in Table 4.8 that the change in Reynolds number has little effect on the solver performance, even though the solutions in this setting have stronger dependency on the Reynolds number than the Chorin problem. Figure 4.2 presents a streamline representation of the computed velocity field at the final timestep for $\ell = 2$ with $\mathrm{Re} = 10$ using Rad(2) and $k = 2$.

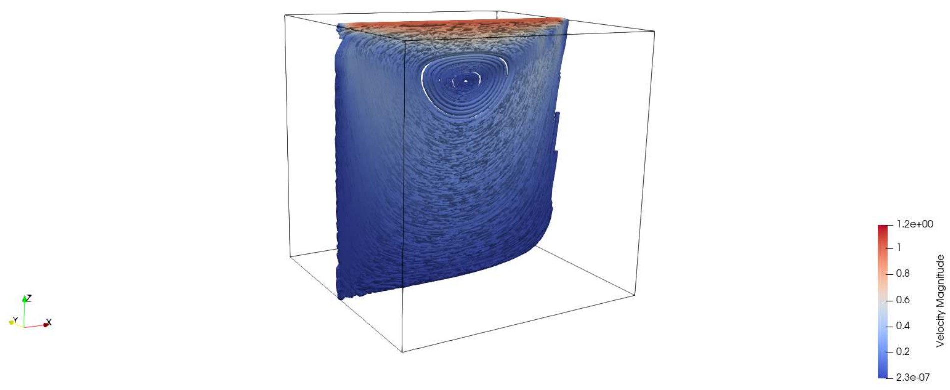


Figure 4.2: Streamlines of the velocity for the lid-driven cavity at final timestep for $\ell = 2$ and $\mathrm{Re} = 10$.

4.5 Conclusion

In this paper, we show that the monolithic Newton-Krylov-multigrid solver developed for incompressible fluid flow problems in [1] can be extended to higher-order pressure-robust discretizations, showing advantages in robustness and wall-clock time-to-solution over the low-order Taylor-Hood spatial discretization considered there. Here, we consider the combination of implicit L-stable Runge-Kutta temporal discretizations with the inf-sup stable and pressure-robust $\mathbf{BDM}_{k+1}(\Omega_h) \times \mathrm{DG}_k(\Omega_h)$

mixed finite-element space. Extending the Vanka relaxation patches from [2], we find order-robust multigrid convergence for standard Navier-Stokes test problems.

Although accurate results are seen with increasing numbers of IRK stages and finite-element order, a better understanding of the stopping criteria for the models considered here is needed to achieve expected convergence behaviour for even higher-order schemes than those used in this paper. Additionally, we note that the wide range of effective Implicit-Explicit (IMEX) Runge-Kutta time-integration methods (cf. [49, 18]) allows a possibly broader scope to studies such as this one, as the linear solver used here could be used for the implicit subsystems that require solution in such approaches.

	ℓ	L_2 error $\mathbf{u}$	L_2 error p	time	linear	nonlinear
Rad(2)+$k=2$		$\Delta t = T_f/N$				
	2	5.278×10^{-4}	2.547×10^{-4}	3.9	18.0	4.1
	3	4.617×10^{-5}	4.383×10^{-5}	7.2	22.3	4.6
	4	6.228×10^{-6}	8.127×10^{-6}	57.2	25.0	5.2
		$\Delta t = T_f/2N$				
	2	3.412×10^{-5}	4.327×10^{-4}	4.1	15.4	4.0
	3	2.888×10^{-6}	9.551×10^{-5}	10.9	19.3	4.2
	4	2.140×10^{-7}	2.017×10^{-5}	61.4	23.5	4.7
		$\Delta t = T_f/4N$				
	2	2.385×10^{-4}	3.239×10^{-4}	7.2	13.9	4.0
	3	5.627×10^{-6}	2.702×10^{-5}	41.5	16.2	4.0
	4	4.350×10^{-7}	4.428×10^{-6}	102.4	21.3	4.4
Rad(3)+$k=3$		$\Delta t = 2T_f/N$				
	2	3.311×10^{-5}	5.317×10^{-5}	7.6	21.3	4.9
	3	1.635×10^{-6}	5.623×10^{-6}	12.1	24.3	5.0
	4	6.224×10^{-8}	7.087×10^{-7}	97.5	27.6	5.0
		$\Delta t = T_f/N$				
	2	3.097×10^{-6}	5.456×10^{-6}	12.8	18.8	4.5
	3	1.662×10^{-7}	7.007×10^{-7}	41.0	22.2	5.0
	4	1.834×10^{-8}	8.506×10^{-8}	219.5	23.7	5.1
		$\Delta t = T_f/2N$				
	2	6.894×10^{-6}	4.106×10^{-6}	13.0	16.6	4.2
	3	4.611×10^{-7}	3.285×10^{-7}	51.9	20.1	4.7
	4	2.165×10^{-8}	4.525×10^{-8}	420.2	25.0	5.3
Rad(4)+$k=4$		$\Delta t = 4T_f/N$				
	1	1.036×10^{-3}	1.575×10^{-3}	3.8	19.0	5.0
	2	3.011×10^{-5}	5.341×10^{-5}	7.3	22.8	5.0
	3	6.500×10^{-7}	2.444×10^{-6}	40.6	25.6	5.0
		$\Delta t = 2T_f/N$				
	1	3.164×10^{-5}	5.346×10^{-5}	4.0	17.0	4.5
	2	7.886×10^{-7}	2.420×10^{-6}	13.7	20.8	5.0
	3	3.514×10^{-8}	3.961×10^{-7}	91.0	22.1	5.0
		$\Delta t = T_f/N$				
	1	1.272×10^{-5}	7.461×10^{-6}	5.1	15.1	4.3
	2	1.168×10^{-6}	1.012×10^{-6}	22.9	18.7	4.8
	3	3.096×10^{-8}	4.431×10^{-8}	143.4	21.7	5.0

Table 4.3: Results for several refinement levels showing the effect of varying time step size on discretization accuracy and solver performance.

	ℓ	L_2 error $\mathbf{u}$	L_2 error p	time	linear	nonlinear
Rad(2)+$k=2$			Re = 1			
	2	3.412×10^{-5}	4.327×10^{-4}	4.1	15.4	4.0
	3	2.888×10^{-6}	9.551×10^{-5}	10.9	19.3	4.2
	4	2.140×10^{-7}	2.017×10^{-5}	61.4	23.5	4.7
			Re = 10			
	2	3.413×10^{-5}	4.328×10^{-4}	3.4	15.5	4.0
	3	2.886×10^{-6}	9.531×10^{-5}	12.8	19.3	4.2
	4	3.097×10^{-7}	2.291×10^{-5}	65.1	23.7	4.7
			Re = 100			
	2	3.427×10^{-5}	4.342×10^{-4}	3.4	16.4	4.0
	3	2.879×10^{-6}	9.537×10^{-5}	12.7	22.2	4.2
	4	2.080×10^{-7}	2.234×10^{-5}	63.1	24.7	4.7
Rad(3)+$k=3$			Re = 1			
	2	3.097×10^{-6}	5.456×10^{-6}	12.8	18.8	4.5
	3	1.662×10^{-7}	7.007×10^{-7}	41.0	22.2	5.0
	4	1.834×10^{-8}	8.506×10^{-8}	219.5	23.7	5.1
			Re = 10			
	2	3.098×10^{-6}	5.458×10^{-6}	12.8	18.8	4.5
	3	1.663×10^{-7}	7.007×10^{-7}	44.0	22.2	5.0
	4	2.213×10^{-8}	9.003×10^{-8}	211.3	23.8	5.1
			Re = 100			
	2	5.115×10^{-6}	5.168×10^{-6}	13.2	19.4	4.4
	3	5.485×10^{-7}	7.555×10^{-7}	44.3	22.6	5.0
	4	1.864×10^{-8}	8.673×10^{-8}	222.0	23.6	5.0
Rad(4)+$k=4$			Re = 1			
	1	3.164×10^{-5}	5.346×10^{-5}	4.0	17.0	4.5
	2	7.886×10^{-7}	2.420×10^{-6}	13.7	20.8	5.0
	3	3.514×10^{-8}	3.961×10^{-7}	91.0	22.1	5.0
			Re = 10			
	1	3.164×10^{-5}	5.346×10^{-5}	5.1	17.0	4.5
	2	7.899×10^{-7}	2.461×10^{-6}	13.7	20.9	5.0
	3	3.572×10^{-8}	1.422×10^{-7}	93.4	23.6	5.0
			Re = 100			
	1	3.379×10^{-5}	5.314×10^{-5}	5.2	18.0	4.8
	2	3.014×10^{-7}	2.465×10^{-6}	28.9	21.8	5.0
	3	1.900×10^{-8}	1.480×10^{-7}	94.8	25.2	5.1

Table 4.4: Numerical results for several refinement levels showing the effect of varying Reynolds number on the accuracy of different temporal and spatial discretizations and the resulting solver performance.

		ℓ		
		1	2	3
k	1	9024	69888	233280
	2	20160	157440	527040
	3	37920	297600	997920

Table 4.5: Number of DoFs per stage as a function of refinement level, ℓ, and degree, k, of the finite-element space used, $\mathbf{BDM}_{k+1}(\Omega_h) \times \mathrm{DG}_k(\Omega_h)$ in the 3D lid-driven cavity model.

	ℓ	time	linear	nonlinear
	1	3.3	12.31	3.0
Rad(1)	2	13.8	12.28	2.88
	3	43.7	11.42	2.88
	1	14.9	10.50	2.69
Rad(2)	2	60.6	11.16	2.69
	3	149.2	10.67	2.73
	1	32.7	11.62	2.56
Rad(3)	2	156.3	11.94	2.66
	3	$\star$		

Table 4.6: Numerical results for several refinement levels for the 3D lid driven cavity model, showing the effect of using different numbers of stages of the temporal IRK discretization for $Re = 1$ and $k = 2$ on the average iteration counts and solver times. Note that $\star$ indicates an out-of-memory error for this test.

	ℓ	time	linear	nonlinear
	1	2.0	5.81	2.25
$k = 1$	2	10.4	9.03	2.50
	3	29.7	9.33	2.71
	1	14.9	10.5	2.69
$k = 2$	2	60.6	11.16	2.69
	3	149.2	10.67	2.73
	1	49.01	13.50	2.56
$k = 3$	2	255.7	12.59	2.69
	3	$\star$		

Table 4.7: Numerical results for several refinement levels for the 3D lid driven cavity model, with $Re = 1$ and 2 IRK stages, showing the effect of using different degrees of the spatial discretization on the average iteration counts and solver times. Note that $\star$ indicates an out-of-memory error for this test.

	ℓ	time	linear	nonlinear
Re = 1	1	14.9	10.5	2.69
	2	60.6	11.16	2.69
	3	149.2	10.67	2.73
Re = 10	1	15.8	10.31	2.50
	2	63.8	11.12	2.69
	3	151.9	10.73	2.77
Re = 100	1	12.4	13.4	2.75
	2	77.2	15.72	2.88
	3	167.5	14.92	2.90

Table 4.8: Numerical results for several refinement levels for the 3D lid driven cavity model, with $k = 2$ and 2 IRK stages, showing the effect of using varying Reynolds number on the average iteration counts and solver times for the 3D lid driven cavity model.

Chapter 4 References

[1] R. Abu-Labdeh, S. MacLachlan, and P. E. Farrell. Monolithic multigrid for implicit Runge–Kutta discretizations of incompressible fluid flow. *Journal of Computational Physics*, 478:111961, 2023.

[2] J. H. Adler, T. R. Benson, and S. P. MacLachlan. Preconditioning a mass-conserving discontinuous Galerkin discretization of the Stokes equations. *Numerical Linear Algebra with Applications*, 24(3):e2047, 2017.

[3] D. Arnold, R. Falk, and R. Winther. Preconditioning in H(div) and applications. *Mathematics of Computation*, 66(219):957–984, 1997.

[4] U. M. Ascher, S. J. Ruuth, and B. T. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

[5] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al. PETSc users manual: Revision 3.10. Technical report, Office of Scientific and Technical Information (OSTI), 2018.

[6] P. Bastian, M. Blatt, and R. Scheichl. Algebraic multigrid for discontinuous Galerkin discretizations of heterogeneous elliptic problems. *Numerical Linear Algebra with Applications*, 19(2):367–388, 2012.

[7] M. Benzi and M. A. Olshanskii. An augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing*, 28(6):2095–2113, 2006.

[8] D. Boffi, F. Brezzi, and M. Fortin. *Mixed finite element methods and applications*, volume 44 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2013.

[9] T. Boonen, J. Van lent, and S. Vandewalle. An algebraic multigrid method for high order time-discretizations of the div-grad and the curl-curl equations. *Applied Numerical Mathematics*, 59(3):507–521, 2009.

[10] D. Braess and R. Sarazin. An efficient smoother for the Stokes problem. *Applied Numerical Mathematics*, 23(1):3–19, 1997.

[11] A. Brandt and N. Dinar. Multigrid solutions to elliptic flow problems. In *Numerical Methods for Partial Differential Equations*, pages 53–147. Elsevier, 1979.

[12] F. Brezzi, J. Douglas, and L. D. Marini. Two families of mixed finite elements for second order elliptic problems. *Numerische Mathematik*, 47:217–235, 1985.

[13] F. Brezzi and R. S. Falk. Stability of higher-order Hood–Taylor methods. *SIAM Journal on Numerical Analysis*, 28(3):581–590, 1991.

[14] A. N. Brooks and T. J. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1-3):199–259, 1982.

[15] E. Burman and P. Hansbo. Edge stabilization for the generalized Stokes problem: a continuous interior penalty method. *Computer Methods in Applied Mechanics and Engineering*, 195(19-22):2393–2410, 2006.

[16] J. C. Butcher. On the implementation of implicit Runge-Kutta methods. *BIT Numerical Mathematics*, 16(3):237–240, 1976.

[17] J. C. Butcher. General linear methods. *Acta Numerica*, 15:157–256, 2006.

[18] T. Buvoli and B. S. Southworth. Additive polynomial time integrators, Part I: Framework and fully-implicit-explicit (FIMEX) collocation methods. *arXiv preprint arXiv:2109.03317*, 2021.

[19] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[20] C. H. Cooke and D. K. Blanchard. A higher order finite element algorithm for the unsteady Navier-Stokes equations. *Mathematics and Computers in Simulation*, 22(2):127–132, 1980.

[21] H. Damanik, J. Hron, A. Ouazzi, and S. Turek. Monolithic Newton-multigrid solution techniques for incompressible nonlinear flow models. *International Journal for Numerical Methods in Fluids*, 71(2):208–222, 2013.

[22] M. Deville, T.-H. Lê, and Y. Morchoisne. *Numerical simulation of 3-D incompressible unsteady viscous laminar flows: a GAMM-Workshop*, volume 48. Vieweg+ Teubner Verlag, 2013.

[23] D. A. Di Pietro and A. Ern. *Mathematical aspects of discontinuous Galerkin methods*, volume 69. Springer Science & Business Media, 2011.

[24] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.

[25] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 227(3):1790–1808, 2008.

[26] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics.* Oxford University Press, USA, 2014.

[27] P. E. Farrell, R. C. Kirby, and J. Marchena-Menendez. Irksome: Automating Runge–Kutta time-stepping for finite element methods. *ACM Transactions on Mathematical Software*, 47(4):1–26, 2021.

[28] P. E. Farrell, M. G. Knepley, L. Mitchell, and F. Wechsung. PCPATCH: software for the topological construction of multigrid relaxation methods. *ACM Transactions on Mathematical Software*, 47(3):1–22, 2021.

[29] P. E. Farrell, L. Mitchell, L. R. Scott, and F. Wechsung. A Reynolds-robust preconditioner for the Scott-Vogelius discretization of the stationary incompressible Navier-Stokes equations. *The SMAI Journal of Computational Mathematics*, 7:75–96, 2021.

[30] P. E. Farrell, L. Mitchell, and F. Wechsung. An augmented Lagrangian preconditioner for the 3D stationary incompressible Navier–Stokes equations at high Reynolds number. *SIAM Journal on Scientific Computing*, 41(5):A3073–A3096, 2019.

[31] P. F. Fischer. An overlapping Schwarz method for spectral element solution of the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 133(1):84–101, 1997.

[32] N. R. Gauger, A. Linke, and P. W. Schroeder. On high-order pressure-robust space discretisations, their advantages for incompressible high Reynolds number generalised Beltrami flows and beyond. *The SMAI Journal of Computational Mathematics*, 5:89–129, 2019.

[33] J.-L. Guermond. Stabilization of Galerkin approximations of transport equations by subgrid modeling. *ESAIM: Mathematical Modelling and Numerical Analysis*, 33(6):1293–1316, 1999.

[34] J.-L. Guermond and P. Minev. High-order time stepping for the incompressible Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 37(6):A2656–A2681, 2015.

[35] J. Guzmán and L. Scott. The Scott-Vogelius finite elements revisited. *Mathematics of Computation*, 88(316):515–529, 2019.

[36] E. Hairer and G. Wanner. Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics*, 111(1-2):93–111, 1999.

[37] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids*, 8(12):2182–2189, 1965.

[38] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[39] V. John, A. Linke, C. Merdon, M. Neilan, and L. G. Rebholz. On the divergence constraint in mixed finite element methods for incompressible flows. *SIAM Review*, 59(3):492–544, 2017.

[40] V. John and G. Matthies. Higher-order finite element discretizations in a benchmark problem for incompressible flows. *International Journal for Numerical Methods in Fluids*, 37(8):885–903, 2001.

[41] V. John and L. Tobiska. Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 33(4):453–473, 2000.

[42] R. C. Kirby, A. Logg, M. E. Rognes, and A. R. Terrel. Common and unusual finite elements. In *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*, pages 95–119. Springer, 2012.

[43] F. Laakmann, P. E. Farrell, and L. Mitchell. An augmented Lagrangian preconditioner for the magnetohydrodynamics equations at high Reynolds and coupling numbers. *SIAM Journal on Scientific Computing*, 44(4):B1018–B1044, 2022.

[44] M. Lange, L. Mitchell, M. G. Knepley, and G. J. Gorman. Efficient mesh management in Firedrake using PETSC DMPLEX. *SIAM Journal on Scientific Computing*, 38(5):S143–S155, 2016.

[45] P. L. Lederer, C. Merdon, and J. Schöberl. Refined a posteriori error estimation for classical and pressure-robust Stokes finite element methods. *Numerische Mathematik*, 142(3):713–748, 2019.

[46] A. Linke and C. Merdon. Pressure-robustness and discrete Helmholtz projectors in mixed finite element methods for the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 311:304–326, 2016.

[47] S. P. MacLachlan and C. W. Oosterlee. Local Fourier analysis for multigrid with overlapping smoothers applied to systems of PDEs. *Numerical Linear Algebra with Applications*, 18(4):751–774, 2011.

[48] K.-A. Mardal, T. K. Nilssen, and G. A. Staff. Order-optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDEs. *SIAM Journal on Scientific Computing*, 29(1):361–375, 2007.

[49] S. T. Miller, E. C. Cyr, J. N. Shadid, R. M. J. Kramer, E. G. Phillips, S. Conde, and R. P. Pawlowski. Imex and exact sequence discretization of the multi-fluid plasma model. *Journal of Computational Physics*, 397:108806, 2019.

[50] P. Munch, I. Dravins, M. Kronbichler, and M. Neytcheva. Stage-parallel fully implicit Runge–Kutta implementations with optimal multilevel preconditioners at the scaling limit. *SIAM Journal on Scientific Computing*, pages S71–S96, 2023.

[51] J.-C. Nédélec. A new family of mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 50:57–81, 1986.

[52] W. Pazner and P.-O. Persson. Stage-parallel fully implicit Runge–Kutta solvers for discontinuous Galerkin fluid simulations. *Journal of Computational Physics*, 335:700–717, 2017.

[53] W. Pazner and P.-O. Persson. Approximate tensor-product preconditioners for very high order discontinuous galerkin methods. *Journal of Computational Physics*, 354:344–369, 2018.

[54] M. M. Rana, V. E. Howle, K. Long, A. Meek, and W. Milestone. A new block preconditioner for implicit Runge-Kutta methods for parabolic PDE. *SIAM J. Sci. Comp.*, 43(5):S475–S495, 2021.

[55] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. McRae, G.-T. Bercea, G. R. Markall, and P. H. Kelly. Firedrake: automating the finite element method by composing abstractions. *ACM Transactions on Mathematical Software*, 43(3):1–27, 2016.

[56] B. Rivière. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[57] E. Rosseel, T. Boonen, and S. Vandewalle. Algebraic multigrid for stationary and time-dependent partial differential equations with stochastic coefficients. *Numer. Linear Algebra Appl.*, 15(2-3):141–163, 2008.

[58] P. W. Schroeder, C. Lehrenfeld, A. Linke, and G. Lube. Towards computable flows and robust estimates for inf-sup stable FEM applied to the time-dependent incompressible Navier–Stokes equations. *SeMA Journal*, 75:629–653, 2018.

[59] L. R. Scott and M. Vogelius. Norm estimates for a maximal right inverse of the divergence operator in spaces of piecewise polynomials. *ESAIM: Mathematical Modelling and Numerical Analysis*, 19(1):111–143, 1985.

[60] B. S. Southworth, O. Krzysik, and W. Pazner. Fast solution of fully implicit Runge–Kutta and discontinuous Galerkin in time for numerical PDEs, Part II: Nonlinearities and DAEs. *SIAM Journal on Scientific Computing*, 44(2):A636–A663, 2022.

[61] B. S. Southworth, O. Krzysik, W. Pazner, and H. D. Sterck. Fast solution of fully implicit Runge–Kutta and discontinuous Galerkin in time for numerical PDEs, Part I: the linear setting. *SIAM Journal on Scientific Computing*, 44(1):A416–A443, 2022.

[62] G. A. Staff, K.-A. Mardal, and T. K. Nilssen. Preconditioning of fully implicit Runge-Kutta schemes for parabolic PDEs. *Modeling, Identification and Control*, 27:109–123, 2006.

[63] C. Taylor and P. Hood. A numerical solution of the Navier-Stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73–100, 1973.

[64] J. Van Lent and S. Vandewalle. Multigrid methods for implicit Runge–Kutta and boundary value method discretizations of parabolic PDEs. *SIAM Journal on Scientific Computing*, 27(1):67–92, 2005.

[65] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65(1):138–158, 1986.

[66] G. Wanner and E. Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg, 1996.

[67] M. Wathen, C. Greif, and D. Schotzau. Preconditioners for mixed finite element discretizations of incompressible MHD equations. *SIAM Journal on Scientific Computing*, 39(6):A2993–A3013, 2017.

[68] S. Zhang. A new family of stable mixed finite elements for the 3D Stokes equations. *Math. Comp.*, 74(250):543–554, 2005.

Chapter 5

Monolithic Multigrid Methods for Higher-Order Discretizations of Time-Dependent Maxwell's Equations

Abstract

Maxwell's equations arise in many challenging applications in computational science. Several paths exist to achieve high-order discretizations, including the use of specialized finite-element spaces (such as the Raviart-Thomas, Brezzi-Douglas-Marini, and both first- and second-kind Nédélec elements) and high-order implicit temporal integration schemes. While significant effort has been invested in the past 25 years in developing efficient multigrid methods for various spatial discretizations, much of the work in the time-dependent case has been focused on multi-step (or diagonally implicit) temporal discretizations. In this paper, we present work on extending monolithic multigrid methods to fully implicit Runge-Kutta temporal discretizations of the $\mathbf{B}-\mathbf{E}$ form of Maxwell's equations. Particular focus is paid to extending the common overlapping Schwarz relaxation strategy to these discretizations, as well as their use on non-nested multigrid hierarchies, as needed to accurately model complex geometries.

Keywords: Maxwell's equations, Implicit Runge-Kutta time integration, Monolithic multigrid, non-nested multigrid

5.1 Introduction

Maxwell's equations are coupled partial differential equations that provide a mathematical model of the dynamic interaction between magnetic and electric fields [37, 54]. Among many multiphysics problems where they appear, they are used in describing the electromagnetic dynamics of (multi-)fluid plasma models. Plasma models in general, including single-fluid plasma models such as the magnetohydrodynamics (MHD), are coupled systems of differential equations that model the movement of charged plasma particles in the presence of electromagnetic (EM) fields. These models have several applications such as in the construction of rocket engines, in power generation using thermonuclear fusion reactors, and applications of magnetic confinement of plasma in tokamak reactors [13]. In these models, accurately resolving the coupling between the fluid flow and EM variables is critical, in both the continuum models and the corresponding discrete models and their solution algorithms. So, along with the many solver developments in the recent years for the governing fluid-flow models [10, 22, 20, 25, 38, 21], constructing good solvers for plasma models requires also developing solvers that efficiently handle the EM dynamics. The work proposed here is motivated by the modelling approach of [43], that proposes a split-stepping approach for an MHD system where the slower fluid-flow terms are solved explicitly and the fast EM terms are solved implicitly. In this setting, a high-order fluid model is considered along with the same EM form of Maxwell's equations considered here. The efficient solver developed here for the time-dependant Maxwell's equations with an implicit time integration method can be used in their split-step framework.

Many real-world applications of Maxwell's equations are posed on complicated domains containing curved edges or faces, adding to the difficulty of solving these problems. Although the finite-element method is well suited to handle the unstructured discrete meshes produced on these domains, poor convergence is often seen due to poor representation of the elements adjacent to the curved boundary. Here, we consider using isoparametric finite-element methods [19], wherein the problem geometry is represented by higher-order maps from reference elements to the problem geometry. One concern that arises when using these elements in a multigrid hierarchy is the possible mesh distortion that comes from improving resolution of a curved domain boundary, which can lead to both degraded finite-element approximation properties and degraded solver performance. Thus, on every mesh in the multigrid hiererchy, we make use of the technique proposed in [48], which uses an elasticity solve to improve the mesh quality, taking the boundary node locations as strongly enforced Dirichlet boundary conditions for the mesh DoF locations. In particular, we adopt this approach to yield improving representations of the problem domain with increasing refinement, without encountering problems with element quality. We note, however, that this leads us to non-nested multigrid methods, whose performance must be investigated.

An additional difficulty in solving Maxwell's equations on any domain is that of enforcing the constraints on $\nabla \cdot \mathbf{B}$ and $\nabla \cdot \mathbf{E}$. In the continuum, it is clear that if these constraints are satisfied by the initial conditions, they are always satisfied through the time evolution. However, applying finite-element methods to the continuum problem does not, in general, preserve such constraints [44]. In our context, the constraint of concern is Gauss's law, that the divergence of the magnetic field is zero, $\nabla \cdot \mathbf{B} = 0$. Using a Lagrange multiplier, we can enforce this constraint only weakly. On the other hand, using curl-conforming edge elements for the electric field and divergence-conforming face elements for the magnetic field may allow us to enforce this constraint [45]. In particular, [36] shows that Gauss's law is enforced pointwise (i.e strongly) with such elements, either through using a magnetic Lagrange multiplier as in [8, 18, 11, 56, 35, 5], or when using an augmented Lagrangian formulation, as in [40, 36]. Hence, careful choices of the finite-element spaces must be made along with including a (augmented) Lagrangian multiplier to enforce the constraint $\nabla \cdot \mathbf{B} = 0$, emphasizing the difficulty of spatial discretization of Maxwell's equations. In this paper, we follow the approach taken in [40] in defining the variational formulation to strongly enforce Gauss's Law. We note, of course, that even when using a suitable spatial discretization, we must also choose a constraint-preserving temporal integrator in order to preserve the constraint through numerical integration [30].

In recent years, there has been a growing need to solve these models more accurately, using higher-order discretization schemes to produce higher-fidelity approximations. For spatial discretizations, (isoparametric) FEM provides the ability to construct higher-order elements on complex domains, as discussed above. On the other hand, stability of temporal discretization methods is of great concern when used on stiff systems of differential equations, as we consider here. Implicit schemes are generally used to avoid having (greatly) limited time step size selection. Multi-step methods, such as BDF schemes, can be constructed, but higher than second-order linear multistep schemes lack the required stability properties, hitting the so-called second Dahlquist barrier. Alternatively, a consistently proven powerful high-order temporal discretization scheme is the multi-stage implicit Runge-Kutta method (IRK). Since the systems produced from the application of IRK schemes are very large, these schemes have been rarely used in large-scale computational PDEs. Recently, however, a small body of work has been done to investigate IRK schemes for plasma models [47, 2, 6]. To the authors knowledge, the only other work applying IRK to Maxwell's equations is [33]. Here, we are particularly interested in the coupling of Gauss-Legendre integrators to the constraint-preserving spatial discretizations, as these are known to preserve linear constraints (as of concern here) [30].

Since the systems of equations produced from fully discretizing Maxwell's equation at each time step (using an implicit temporal integration scheme) are very large, efficient solvers need to be constructed. The development of efficient solvers for some forms of Maxwell's equations has been a major topic of research since the pioneering work by Arnold, Falk, and Winther [7] and Hiptmair [31, 32]. Both families of preconditioners have been developed into powerful solvers for problems with divergence- or curl-conforming FEM. Despite the building blocks these provide and their wide use in accelerating the convergence of various numerical methods, they have primarily been applied to simpler formulations of Maxwell's equations than are relevant in the plasma modelling fomain, such as the Eddy-Current form of Maxwell's equation. This so-called "diffusive Maxwell" regime (cf. [12]) is naturally well-suited to efficient multigrid and domain decomposition solvers, as high-frequency modes can be readily damped by relaxation and low-frequency modes corrected from the coarse grid. In our setting, we consider Maxwell in the travelling-wave regime, where important physical constants such as the speed of light, electric permittivity, and magnetic permeability are not neglected. Nonetheless, we show that suitable modifications of the Arnold, Falk, and Winther preconditioner construction still lead to effective solvers for the stage-coupled IRK discretizations.

For problems involving more complicated forms of Maxwell's equations, block preconditioners can also be built in a way that decouples the problem variables, allowing known (and simpler) linear solvers to be applied to each block of the system. This approach is especially useful in more complicated plasma problems with multiple coupled fields that appear with drastically different discretization schemes [29, 5, 50, 51]. While such approaches can be very effective, their development depends on the approximation of the (possible nested) Schur complement(s) of the system, which can be complicated to approximate for complicated block structures. Alternatively, monolithic preconditioners are applied to the system as a whole, trading the need for constructing such Schur complements for that of dealing directly with the coupling between variables. Here, we consider the family of monolithic multigrid techniques, building on the success of multigrid methods as efficient preconditioners for many families of PDEs, and on recent work constructing effective monolithic multigrid preconditioners for MHD and similar systems [46, 2, 3, 4, 62]. To our knowledge, there has been no work done to apply these preconditioners directly to Maxwell's equations, particularly when coupled with IRK temporal discretizations.Here, we use an extension of the solver developed in [2] that was proven to be efficient on the MHD model, among other fluid flow problems, with IRK discretization.

The outline of this paper is organized as follows: in section 5.2, we review the details of the Runge-Kutta time-stepping scheme used for temporal discretization of a system of ODEs. In section 5.3, we explore multigrid methods for the heat equation on domains with curved boundaries, developing the non-nested multigrid approach needed later for Maxwell's equations. In section 5.4, we discuss the spatial discretization of Maxwell's equations, followed by the development of our multigrid methodology in section 5.4.1. Numerical results are presented in section 5.5. Conclusions and possible directions for future research are presented in section 5.6.

5.2 Runge-Kutta discretizations

Discretization of many time-dependent partial differential equations generally follows a method-of-lines process [55]. Usually, we first spatially discretize the time-dependent partial differential equation, leaving a semi-discrete system of ordinary differential equations that is then temporally discretized with a selected time-stepping scheme. There are a wide variety of temporal discretization methods to choose from depending on the problem at hand. While many high-order *multi-step* methods exist, these are generally expected to have stability limitations, for example as seen in the Second Dahlquist Barrier for linear multistep methods [60]. This makes them unsuitable for integrating stiff problems such as Maxwell's equations. Instead, the unconditional stability of some *multi-stage* time-stepping schemes can be taken advantage of. Among these schemes are some families of Runge-Kutta methods, where multiple stage values are computed to determine the approximate solution at each timestep. A general r-stage Runge-Kutta method applied to a system of ordinary differential equations, $u'(t) = f(u(t), t)$, can be written as

$$\begin{aligned} k_i &= f\left(u^n + \Delta t \sum_{j=1}^{r} a_{ij} k_j, t^n + c_i \Delta t\right), \text{ for } i = 1, 2, \ldots, r, \\ u^{n+1} &= u^n + \Delta t \sum_{j=1}^{r} b_j k_j. \end{aligned} \tag{5.1}$$

The set $\{k_j\}_{j=1}^{r}$ represents the r *stage values*, while u^n denotes the approximation to $u(t)$ at time $t^n = t^0 + n\Delta t$. In (5.1), the coefficients are the stage nodes, c_i, the weights, b_j, and the Runge-Kutta matrix, $A = [a_{ij}]$. Altogether, these form the scheme's Butcher tableau [16, 17]. To guarantee consistency of the schemes, we must have $\sum_{j=1}^{r} b_j = 1$ and $\sum_{j=1}^{r} a_{ij} = c_i$, for all $i = 1, 2, \ldots, r$.

Runge-Kutta schemes can be separated into several classifications depending on the non-zero pattern of the Butcher matrix, A. Notable classes of Runge-Kutta methods are explicit (ERK) methods, when $a_{ij} = 0 \ \forall j \geq i$, and implicit (IRK) schemes, when $\exists j \geq i$ with $a_{ij} \neq 0$. Implicit schemes can also be sub-categorized into either diagonally implicit (DIRK) schemes, where $a_{ij} = 0 \ \forall j > i$, or fully implicit (FIRK) methods, when $\exists j > i$ such that $a_{ij} \neq 0$. We naturally choose between such families of schemes based on their cost, accuracy, and stability properties. The cost is determined by both the number of stages and the classification of the scheme, with ERK schemes being cheapest (since they only require computing $f(u, t)$ for known values of u), DIRK schemes being in the middle (since they require an implicit solve for each stage in sequence), and FIRK schemes being most expensive (since they require a coupled implicit solve, typically for all stages at once).

Error analysis generally focuses on either the local truncation error, which is the amount of error introduced from the scheme in a single timestep, found by comparing the approximation at time t^{n+1} to the analytical solution starting from the previous time t^n, or the global error, which is the accumulated error over all timesteps. A time-integration method is said to have an error order of p if the error is bounded by a constant (that may depend on the analytic solution, $u(t)$, and on properties of $f(u,t)$) multiplied by $(\Delta t)^p$. While ERK schemes cannot have order higher than the number of stages, r, FIRK schemes are known that have orders as large as $2r$. Although schemes with higher-order global error are generally more accurate, the *stage order* of a Runge-Kutta method is generally more important when considering performance for stiff differential equations or systems of differential-algebraic equations. The stage order of a scheme is defined to be $\min\{q,p\}$, where q is determined by bounding the approximation to $u(t^n + c_i\Delta t)$ by stage i of the RK scheme by a constant (that can depend on $f(u,t)$ and $u(t)$) times $(\Delta t)^{q+1}$ and p is the global error. Again this shows an advantage of FIRK schemes, as they can have stage order equal to the number of the stages in the scheme, while DIRK schemes can have stage order of at most two.

For any given time-stepping method, we define $r(z)$ to be the *stability function* produced by applying the method to the Dahlquist test problem, $u' = \lambda u$ for $\lambda \in \mathbb{C}$, with $u^{n+1} = r(\lambda\Delta t)u^n$. In general, the domain of stability of the method is the region in the complex plane where $|r(z)| < 1$. The scheme is said to be *A-stable* if its domain of stability contains the entire left half of the complex plane, and such schemes are well-suited for stiff problems as they are unconditionally stable for many spatially discretized PDEs. However, for some stiff problems, A-stability is not enough and a stricter stability type is required. If an A-stable method also satisfies $\lim_{z\to-\infty} |r(z)| = 0$, then it is called an *L-stable* scheme. Here, we are primarily interested in A-stable schemes, as the **B**-**E** form of Maxwell's equations that are of interest are not diffusive in nature. Additionally, the A-stable (but not L-stable) Gauss-Legendre schemes are known to preserve linear and quadrative invariants of the solution of the PDE [30], which will be useful in our setting.

Despite the powerful performance and stability properties of FIRK schemes, they come with two main downsides. First, FIRK discretizations produce large and relatively dense linear or nonlinear systems of equations to solve for the stage approximations at each time step, leading to a high computational cost per time-step. In comparison, DIRK schemes can solve for the stages sequentially, while ERK schemes require only function evaluations of $f(u, t)$ and not solves, greatly reducing their computational cost. Generally, similar simplifications do not exist for FIRK schemes. These large systems are also nonsymmetric, even when the original system of PDEs is self-adjoint, limiting the classes of numerical methods that can be used for their solution and, thus, adding to the difficulty of their solution at each timestep. In this paper, we use a preconditioned flexible GMRES (FGMRES) iteration as the outer Krylov method in our solver. More detail on the preconditioner is given in later sections.

5.2.1 Implementation

All numerical results presented in this paper are produced using Firedrake [52] for the (mixed) finite-element discretization and Irksome [23] for the Runge-Kutta temporal discretization. The linear solvers are implemented through PETSc [9], while the Vanka iterations used as relaxation schemes (discussed later) are implemented using PCPATCH [24]. In the development of our solver, all components are carefully chosen to be naturally parallelizable. The coarsest meshes in our grid hierarchies are distributed (roughly) evenly over all cores, then refined in parallel. In order to perform the relaxation scheme in parallel and achieve accurate computation of the residuals for each DoF in the Vanka patches, we require a node-distance-2 halo in the parallel mesh distribution [42].

5.3 Multigrid on curved domains

Although the main focus in this paper is on solving Maxwell's equations, we first expose the difficulty of achieving high-order approximations for curved domains in the simpler setting of the heat equation, using standard Lagrange finite elements and FIRK temporal discretizations.

Consider, then, the heat equation on a domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$ with homogeneous boundary conditions, given by

$$\begin{aligned} u_t &= \Delta u \quad \text{in } \Omega \times (0, T_f) && (5.2a)\\ u &= 0 \quad \text{on } \partial\Omega \times (0, T_f), && (5.2b)\\ u(\mathbf{x}, 0) &= u_0(\mathbf{x}) \text{ on } \Omega \times \{t = 0\}, && (5.2c) \end{aligned}$$

where T_f denotes the final time. In order to find an approximation $u(\mathbf{x}, t)$, we first spatially discretize (5.2) using Lagrange finite elements, denoting the discretized mesh by Ω_h, which consists of simplex elements. Denoting the discretization space by V_h, the semi-discretized form of (5.2) is to find $u(\cdot, t) \in V_h$ such that

$$\langle u_t, v\rangle - \langle \nabla u, \nabla v\rangle = 0 \text{ for all } v \in V_h \tag{5.3}$$

and all $0 < t \leq T_f$, where the inner product denoted as $\langle w, v\rangle = \int_\Omega w(x)v(x)dV$. Since the weak solution, u, of (5.2) is in $V = \mathbf{H}_0^1(\Omega)$ at all times, we can choose V_h to be the space of continuous piecewise polynomial functions of degree k, referred to as $P_k(\Omega_h)$.

We then apply IRK temporal discretization to the semi-discretized weak form, leading to the fully discretized system of equations. We let $\mathbf{u}(t)$ denote the solution of (5.3), expressed as the vector of coefficients of the basis functions for $u(x, t) = \sum_{j=1}^{N} u_j(t)\phi_j(x) \in V_h$. The IRK temporal discretization then expresses the approximate solution at time t^{n+1},

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \sum_{i=1}^{r} b_i \mathbf{k}_i,$$

by finding the set of r-stage values, $k_i \in V_h$ for $1 \leq i \leq r$, (again expressed by $\mathbf{k}_i$ for their coefficients in the basis for V_h in the variational formula

$$\langle k_i, v_i\rangle + \left\langle \nabla \left(u^n + \Delta t \sum_{j=1}^{r} a_{ij} k_j \right), \nabla v_i \right\rangle = 0, \text{ for all } v_i \in V_h \tag{5.4}$$

for $1 \leq i \leq r$.

As is typical in the finite-element process, we can express (5.4) in terms of the finite-element stiffness and mass matrices, K and M, respectively. Because of the coupling between the spatial and temporal discretizations, the matrix form of (5.4) becomes a block matrix, with blocks defined by the FIRK stages. The resulting matrix system is given by

$$I \otimes M\vec{\mathbf{k}} + \Delta t A \otimes K\vec{\mathbf{k}} = \vec{\mathbf{F}},$$

where I is the $r \times r$ identity matrix, M is the $N \times N$ mass matrix, K is the $N \times N$ stiffness matrix and A is the $r \times r$ Butcher matrix. Both $\vec{\mathbf{k}}$ and $\vec{\mathbf{F}}$ are vectors of length rN, with the i^{th} block in $\vec{\mathbf{k}}$ being $\mathbf{k}_i$ and each block in $\vec{\mathbf{F}}$ coming from the assembled vector $\langle \nabla u^n, \nabla v_i \rangle$ for known function $u^n \in V_h$ and test functions $v_i \in V_h$. Defining $Q := I \otimes M + \Delta t A \otimes K$, then our goal is solve $Q\vec{\mathbf{k}} = \vec{\mathbf{F}}$ efficiently. We note that while Q is a block-lower triangular matrix if DIRK schemes are used, it is generally block dense when FIRK schemes are used.

5.3.1 Monolithic multigrid for the heat equation

The linear systems of equations at each time step produced from fully discretizing the heat equation can be effectively solved using stage-coupled multigrid methods, as first proposed by Vandewalle and co-authors [58, 53, 14], and later implemented in Irksome [23]. We use preconditioned FGMRES as an outer Krylov method, due to the nonsymmetry of these systems. A monolithic multigrid preconditioner is used in the solver. We note that we use FGMRES primarily because it is the "right" implementation of right-preconditioned GMRES for our setting, where we have enough memory available for the added vector storage required by FGMRES, but a relatively expensive preconditioner. Thus, we would rather store the extra vectors for FGMRES than pay for the added computational time needed for one additional preconditioner application, as would be needed by classical right-preconditioned GMRES.

As a preconditioner, we use monolithic multigrid. We define a hierarchy of spatial meshes and use rediscretization to define the system matrices on each mesh level. The coarsest mesh is chosen so that the resulting linear system is small enough to be solved efficiently using LU factorization. Classical finite-element interpolation operators are used for each field in the discretization. If the interpolation operator for scalar functions from $P_k(\Omega_{2h})$ to $P_k(\Omega_h)$ is denoted by $P_{\mathbf{u}}$, then the interpolation operator for the system $Q\mathbf{k} = \mathbf{f}$ is a block diagonal matrix $P = I_r \otimes P_{\mathbf{u}}$, simply using the same interpolation operator for each IRK stage. For the relaxation scheme, we follow [58, 23] and use a stage-coupled relaxation scheme. Here, we use vertex-centric patches for relaxation. For $P_1(\Omega_h)$, this amounts to just a (weighted) block-Jacobi iteration, coupling all stage values at each vertex in the mesh. For $P_2(\Omega_h)$, this gives an overlapping (weighted) block-Jacobi iteration, where all stage values at each vertex and on each edge incident on the vertex are relaxed together, so that all vertex-based DoFs are relaxed once per sweep, while all edge-based DoFs are relaxed twice per sweep. In this section, we use Chebyshev polynomials of the first kind to accelerate the relaxation, using the default PETSc parameters that estimate the largest eigenvalue, λ, of the Schwarz-preconditioned system and defining the minimax polynomial over the interval $[0.1\lambda, 1.1\lambda]$.

When Ω is a polytope, it is generally straightforward to define a hierarchy of nested meshes for the multigrid solution process. When Ω has curved boundaries, on the other hand, a nested multigrid hierarchy would lead to significant errors in resolving the domain boundaries, as the approximation of these on the coarsest mesh would need to be carried forward to *all* meshes in the hierarchy. A simple strategy to generate non-nested hierarchies that offer better representation of the domain boundaries is to take a given coarsest mesh, refine it once (uniformly, in the work considered here), then move all new nodes introduced on the boundary of the coarse mesh to the true boundary of the domain. As we will see in section 5.3.2, this strategy leads to problems in both finite-element and multigrid convergence, yielding linear systems that are difficult to solve in our multigrid framework and whose solutions do not yield the full accuracy promised by higher-order finite-element spaces. We note that the results below make use of isoparametric elements, where the map from the reference element (triangle or tetrahedron) to the elements of the mesh is of the same order as the finite-element approximation space, as needed to achieve the full approximation properties expected for higher-order finite-element methods (cf. [19]). Only a slight modification to the above procedure is needed to account for isoparametric elements, ensuring that all mesh locations associated with boundary edges are projected to the domain boundary with each refinement step. Figure 5.1 shows how the nodal degrees of freedom on the edges of a coarse discretization mesh, using a $P_2(\Omega_h)$ space that is then once refined, are moved to lie on the curved boundary of a unit circle domain.

At the left of Figure 5.2, we see a likely cause of the problematic convergence. The simple "refine-and-project" strategy proposed above leads to very different element quality in the interior of the mesh compared to elements near the mesh boundary. Those elements, in particular, get "stretched", as the boundary of the coarsest mesh

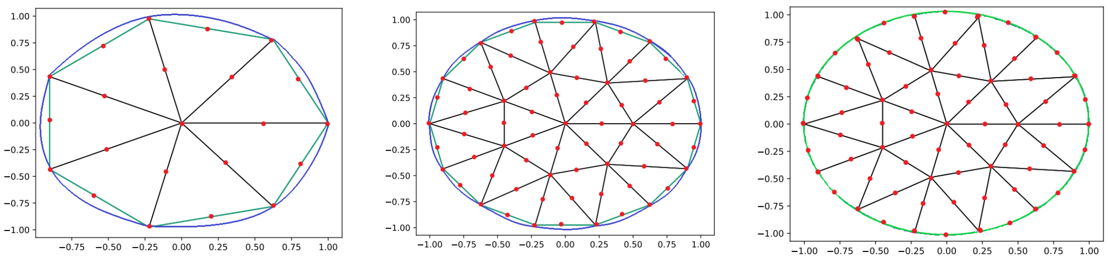


Figure 5.1: $P_2(\Omega_h)$ mesh triangulation of a circle domain with the nodal degrees of freedom indicated as red dots. At left, a coarse mesh discretization of the domain. In center, once refined mesh of the original discretization with vertices moved to the domain boundary. At right, the mesh that results from moving the edge-based nodes to the curved domain boundary.

does not move, so all introduced mesh nodes along each of the edges on the boundary of the coarsest mesh are forced to stretch to fill the gap between the original approximation of Ω and its true shape. To resolve this, we must somehow allow the interior mesh nodes to move closer to the domain boundary. To do so, we follow the approach introduced in [48], and introduce an additional step, solving for the mesh coordinates on the refined mesh using the equations of linear elasticity to assign new coordinates. Here, we treat the locations of the boundary mesh points (nodes of the mesh for linear elements, nodes and midpoints for quadratic elements) as Dirichlet data, and solve the linear elasticity equation, finding $\mathbf{x} \in (P_k(\Omega_h))^d$ that satisfies the Dirichlet data and minimizes $\frac{1}{2}\|\epsilon(\mathbf{x})\|^2 + \frac{1}{2}\|\nabla \cdot \mathbf{x}\|^2$, where $\epsilon(\mathbf{x}) = \frac{1}{2}\left(\nabla\mathbf{x} + (\nabla\mathbf{x})^T\right)$ is the symmetric part of the gradient of $\mathbf{x}$. While we could include more complicated Lamé coefficients in the energy, preliminary experiments showed that the quality of solution is not greatly dependent on these parameters. To solve this problem, we use the algebraic multigrid method in ML with default parameters [27], to a residual reduction tolerance of 10^{-6}, starting from the uniformly refined mesh node locations as the initial guess for $\mathbf{x}$. We note that the resulting multigrid hierarchy is now non-nested in both the interior and near the boundary, as the correspondence between fine-grid and coarse-grid nodes is broken by the elasticity solves. We choose to not account for this in our multigrid interpolation operators, simply interpolating from each coarse mesh to the next finest as if it were a uniform refinement, and will see that this does not lead to poor convergence on mesh hierarchies generated using the elasticity solves.

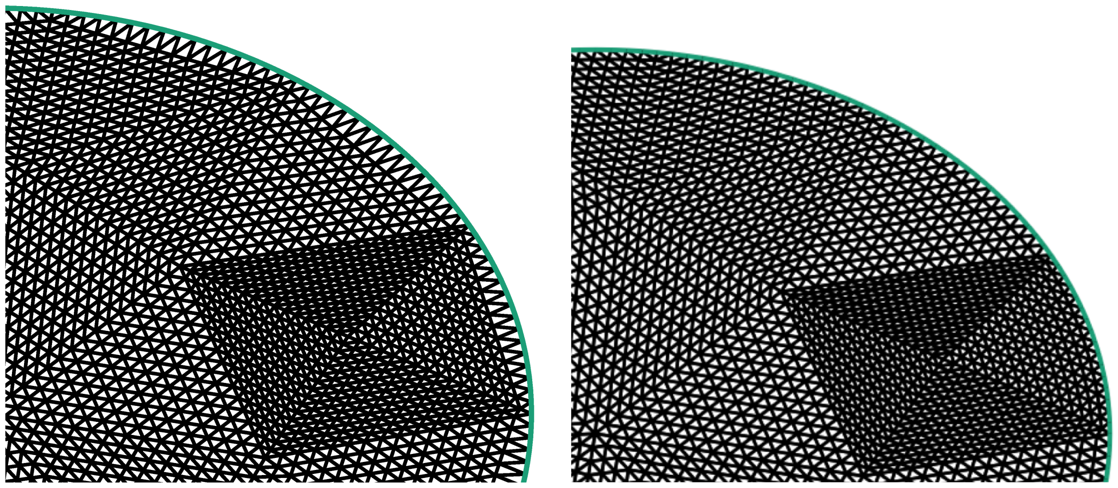


Figure 5.2: Sections of meshes of unit circle domain. At left, original mesh formed by the simple "refine-and-project" strategy of moving new mesh nodes on boundary of each mesh to domain boundary. On right, mesh that results from following this by an elasticity solve to move internal nodes.

5.3.2 Numerical Results

In this section, we consider the heat equation on the unit disk and unit sphere, and examine the convergence of the multigrid method proposed above, both with and without the elasticity solves for the mesh. For the temporal integrator, we use the Lobatto-IIIC method, as is appropriate for the the (diffusive) heat equation. We consider only the 2-stage method in this section. We use ℓ to denote the number of levels in our mesh hierarchies, starting from given coarsest grids that (poorly) resolve the domain geometries.

For the unit disk centered at $(0, 0)$, we consider a standard test case, setting initial and boundary conditions such that the solution to the heat equation, expressed in polar coordinates, is given by $u(r, \theta, t) = J_0(z_0 r)\mathrm{e}^{-z_0^2 t}$, where $J_0(z)$ is the zero-th order Bessel function, with first root z_0. The coarsest grid consists of 47 elements, and we consider $3 \leq \ell \leq 5$ refinement levels. Here, we consider integrating to $T_f = 0.25$ with $\Delta t = 2^{-(\ell+1)}T_f$ for the piecewise linear discretization and $\Delta t = 2^{-(\ell+3)}T_f$ for the piecewise quadratic discretization. We use FGMRES stopping tolerances requiring either a relative reduction in the residual norm by a factor of 10^{-5} or for the residual norm to be reduced below $0.1/N^3$ for $N = 2^{\ell+2}$ at each timestep. We use a V(2,2) monolithic multigrid cycle as preconditioner, with relaxation as described above. We

	ℓ	L_2 error	H_1 error	iterations	time
$P_1(\Omega_h)$	3	2.866×10^{-3}	3.807×10^{-2}	3.81	0.14
	4	7.512×10^{-4}	1.934×10^{-2}	4.03	0.34
	5	2.282×10^{-4}	1.011×10^{-2}	4.03	0.67
$P_2(\Omega_h)$	3	4.453×10^{-5}	1.832×10^{-3}	3.06	1.56
	4	$\star$			
	5	$\star$			

Table 5.1: Numerical results for heat equation on the disk with two-stage LobattoIIIC schemes, on multigrid hierarchy **without** elasticity solves. Relative L_2 and H_1 errors in u at the final time are recorded, along with the number of linear iterations per time step, averaged over all timesteps, and the total wall-clock time-to-solution in minutes, for refinement levels $3 \leq \ell \leq 5$.

consider a simple weak scaling study, using 1 core for $\ell = 3$, 4 cores for $\ell = 4$, and 16 cores for $\ell = 5$, on a machine with 2 Intel Xeon E5-2650 v2 CPUs, each with 8 physical cores, but 16 virtual cores, at 2.60GHz, with 128GB of RAM. We note that we expect the time-to-solution to approximately double with each refinement step for piecewise linears, and approximately quadruple at each time step for piecewise quadratics, as the increased parallelism accounts for the increase in number of spatial degrees of freedom, but not the increase in the number of time steps.

Table 5.1 presents results for the mesh hierarchy with no elasticity solves. We report the relative L_2 and H_1 errors in u at the final time, to assess the finite-element convergence, as well as the number of linear iterations per time step, averaged over all time steps, and the total computational time taken in minutes. The ($\star$) symbol indicates runs where the linear solver did not converge at some point in the time stepping. While the multigrid solver and finite-element convergence for the piecewise linear discretization looks acceptable, we see multigrid convergence failures for $\ell = 4$ and 5 for the piecewise quadratic discretization. For comparison, Table 5.2 shows results for the same experiment, now using linear elasticity solves for the meshes. We see slightly improved convergence for the piecewise linear case, but substantially improved convergence for the piecewise quadratic case. In particular, we see almost the expected finite-element convergence now using the piecewise quadratic approximation.

We next consider Ω to be the unit sphere, centered at $(0, 0, 0)$. The initial and

	ℓ	L_2 error	H_1 error	iterations	time
LobIIIC(2)+$\mathbf{P}_1$	3	3.040×10^{-3}	3.744×10^{-2}	3.06	1.25
	4	7.616×10^{-4}	1.871×10^{-2}	3.09	1.52
	5	1.906×10^{-4}	9.370×10^{-3}	3.05	2.16
LobIIIC(2)+$\mathbf{P}_2$	3	1.856×10^{-5}	3.279×10^{-4}	3.05	1.77
	4	4.866×10^{-6}	8.556×10^{-5}	4.02	6.32
	5	1.075×10^{-6}	3.075×10^{-5}	4.01	11.14

Table 5.2: Numerical results for heat equation on the disk with two-stage LobattoIIIC schemes, on multigrid hierarchy **with** elasticity solves. Relative L_2 and H_1 errors in u at the final time are recorded, along with the number of linear iterations per time step, averaged over all timesteps, and the total wall-clock time-to-solution in minutes, for refinement levels $3 \leq \ell \leq 5$.

boundary conditions are chosen so that the exact solution, expressed in spherical coordinates, is given by $u(r, \theta, \phi, t) = \frac{1}{r}\sin(r\pi)\mathrm{e}^{-\pi^2 t}$. A coarsest mesh of 151 tetrahedral elements is specified, that is then refined $1 \leq \ell \leq 4$ times to construct the hierarchy of meshes. We again set the final time to be $T_f = 0.25$, and take $\Delta t = 2^{-(2\ell+2)}T_f$ for the piecewise linear case and $\Delta t = 2^{-(\ell+5)}T_f$ for piecewise quadratics. We again use FGMRES stopping tolerances requiring either a relative reduction in the residual norm by a factor of 10^{-5} or for the residual norm to be reduced below $0.1/N^3$ for the piecewise linear case or $0.1/N^4$ for the piecewise quadratic case for $N = 2^{\ell+3}$ at each timestep. We again use a V(2,2) monolithic multigrid cycle as preconditioner. Here, for the weak scaling study, we use 1 core for $\ell = 2$ with piecewise linears or $\ell = 1$ with piecewise quadratics, and increase to 4 cores for the first refinement, and 10 cores for the second refinement. We note that we expect the time-to-solution to approximately quadruple with each refinement step for these runs, since we only account for part of the increase in the number of spatial DoFs with each refinement.

Table 5.3 shows the same data for this test problem as recorded before for the multigrid hierarchy without elasticity solves. Here, we see one multigrid convergence failure, for $\ell = 4$ with piecewise linears, but substantial issues with finite-element convergence, particularly for piecewise quadratics. Table 5.4 presents the corresponding results using the linear elasticity solves in the definition of the multigrid hierarchy. Here, we see the expected finite-element convergence for both spatial discretizations, as well as robust solver performance. Having established the necessity of elasticity solves in the mesh construction for both robust finite-element and linear solver

	ℓ	L_2 error	H_1 error	iterations	time
$P_1(\Omega_h)$	2	1.850×10^{-1}	2.171×10^{-1}	6.44	0.04
	3	6.146×10^{-2}	1.138×10^{-1}	6.28	0.22
	4	$\star$			
$P_2(\Omega_h)$	1	1.047×10^{-3}	3.235×10^{-2}	3.36	0.87
	2	2.136×10^{-4}	1.461×10^{-2}	4.32	2.68
	3	6.430×10^{-4}	9.682×10^{-3}	4.71	23.24

Table 5.3: Numerical results for heat equation on the sphere with two-stage LobattoIIIC schemes, on multigrid hierarchy **without** elasticity solves. Relative L_2 and H_1 errors in u at the final time are recorded, along with the number of linear iterations per time step, averaged over all timesteps, and the total wall-clock time-to-solution in minutes, for refinement levels $1 \leq \ell \leq 4$.

	ℓ	L_2 error	H_1 error	iterations	time
$P_1(\Omega_h)$	2	6.290×10^{-2}	1.129×10^{-1}	6.75	0.25
	3	1.726×10^{-2}	5.600×10^{-2}	6.07	1.52
	4	4.452×10^{-3}	2.803×10^{-2}	5.24	21.03
$P_2(\Omega_h)$	1	4.414×10^{-3}	2.730×10^{-2}	3.36	0.53
	2	5.083×10^{-4}	8.160×10^{-3}	4.30	2.79
	3	6.775×10^{-5}	2.333×10^{-3}	4.70	13.93

Table 5.4: Numerical results for heat equation on the sphere with two-stage LobattoIIIC schemes, on multigrid hierarchy **with** elasticity solves. Relative L_2 and H_1 errors in u at the final time are recorded, along with the number of linear iterations per time step, averaged over all timesteps, and the total wall-clock time-to-solution in minutes, for refinement levels $1 \leq \ell \leq 4$.

convergence, we now consider solution of Maxwell's equations, using such multigrid hierarchies.

5.4 Numerical approximation of solutions of Maxwell's equations

Maxwell's equations are well-known in many settings, including plasma physics [56, 3] and geophysical electromagnetics [28, 12]. We generally consider different regimes of the equations, depending on which physical effects are important, and whether we treat the problem in the time domain or in the frequency domain. Much of the work on multigrid solver development, in particular, has been done in the case where we can reduce the equations to consider just one of the electric, **E**, or magnetic, **B**, fields, taking the form of a Hodge Laplacian or the so-called "eddy current" approximation [7, 31, 32, 34, 39]. Here, we focus on constructing high-fidelity solvers specifically for the **B**-**E** form of Maxwell's equations in the time domain, noting that there is a general lack of research addressing this formulation. Similar models have been considered in [50], where backward Euler was used for discretization in time, and [5], where the temporal discretization chosen was Crank-Nicholson, developing block preconditioning approaches. The **B**-**E** model was considered in the context of MHD in [40, 41, 35], where lower-order implicit time stepping schemes are used such as BDF2. In this paper, we focus on the regime where Maxwell's equations can be written as

$$\begin{aligned} \mathbf{E}_t - c^2\nabla\times\mathbf{B} &= -\frac{1}{\epsilon}\mathbf{J}, \\ \mathbf{B}_t + \nabla\times\mathbf{E} &= 0, \end{aligned} \tag{5.5}$$

on domain $\Omega \subset \mathbb{R}^d$ with $d \in \{2,3\}$, where **E** is the electric field, **B** is the magnetic field, and **J** represents a source current. The constants c and ϵ represent the speed of light in a vacuum (299,792,458 m/s) and permittivity of free space, respectively. These equations result from taking the full Maxwell's equations and taking electrical conductivity $\sigma = 0$, magnetic permeability $\mu = \mu_0$, the permeability of free space, and the relation $c^2 = \frac{1}{\epsilon_0\mu_0}$. We note that the scalar constraints of (5.5) are weakly preserved by time stepping [11].

We consider a standard spatial semi-discretization of (5.5), taking $(\mathbf{B},\mathbf{E}) \in \mathbf{H}_0(\text{div},\Omega)\times \mathbf{H}_0(\text{curl},\Omega)$, where

$$\begin{aligned} \mathbf{H}(\text{div},\Omega) &= \{\mathbf{u}\in\mathbf{L}_2(\Omega) : \nabla\cdot\mathbf{u}\in L_2(\Omega)\}, \\ \mathbf{H}(\text{curl},\Omega) &= \{\mathbf{u}\in\mathbf{L}_2(\Omega) : \nabla\times\mathbf{u}\in \mathbf{L}_2(\Omega)\}, \\ \mathbf{H}_0(\text{div},\Omega) &= \{\mathbf{u}\in\mathbf{H}(\text{div},\Omega) : \mathbf{u}\cdot\mathbf{n} = \mathbf{0} \text{ on } \partial\Omega\}, \\ \mathbf{H}_0(\text{curl},\Omega) &= \{\mathbf{u}\in\mathbf{H}(\text{curl},\Omega) : \mathbf{u}\times\mathbf{n} = 0 \text{ on } \partial\Omega\}. \end{aligned}$$

Multiplying (5.5) by the test functions $(\mathbf{C},\mathbf{D}) \in \mathbf{H}_0(\text{div},\Omega)\times\mathbf{H}_0(\text{curl},\Omega)$, then integrating by parts gives the semi-discretized variational form: find $(\mathbf{B}(\cdot,t),\mathbf{E}(\cdot,t)) \in$

$\mathbf{H}_0(\mathrm{div},\Omega)\times\mathbf{H}_0(\mathrm{curl},\Omega)$ such that

$$\begin{aligned}\langle \mathbf{E}_t,\mathbf{D}\rangle - \langle c^2\mathbf{B},\nabla\times\mathbf{D}\rangle &= -\left\langle \frac{1}{\epsilon}\mathbf{J},\mathbf{D}\right\rangle,\\ \langle \mathbf{B}_t,\mathbf{C}\rangle + \langle\nabla\times\mathbf{E},\mathbf{C}\rangle &= \mathbf{0},\end{aligned} \tag{5.6}$$

$\forall(\mathbf{c},\mathbf{d})\in\mathbf{H}_0(\mathrm{div},\Omega)\times\mathbf{H}_0(\mathrm{curl},\Omega)$. Non-homogeneous (or more general) Dirichlet boundary conditions can be incorporated in the variational form in the usual way.

We consider mixed finite-element approximation for the variational form in (5.6), using both first-kind and second-kind elements. For the "first-kind" discretization, we use Raviart-Thomas (RT) elements for $\mathbf{B}$ and Nédléc elements of the first kind (N1Curl) for $\mathbf{E}$, of equal order. For the "second-kind" discretization, we use Brezzi-Douglas-Marini (BDM) elements for $\mathbf{B}$ and Nédléc elements of the second kind (N2Curl) for $\mathbf{E}$, again of equal order. For consistency, we follow the convention that the lowest-order spaces of either first or second kind are of order one, resulting in order k spaces that are subsets of the vector $P_k(\Omega_h)$ spaces. Hence, the semi-discretized system can be written in matrix form as

$$\begin{bmatrix} M_{\mathbf{E}} & 0\\ 0 & M_{\mathbf{B}}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{E}}_t\\ \hat{\mathbf{B}}_t\end{bmatrix} + \begin{bmatrix} 0 & -c^2L^T\\ L & 0\end{bmatrix}\begin{bmatrix}\hat{\mathbf{E}}\\ \hat{\mathbf{B}}\end{bmatrix} = \begin{bmatrix} -\frac{1}{\epsilon}\hat{\mathbf{J}}\\ \mathbf{0}\end{bmatrix}, \tag{5.7}$$

where $M_{\mathbf{B}}$ and $M_{\mathbf{E}}$ are mass matrices for $\mathbf{B}$ and $\mathbf{E}$ respectively, and L is the finite-element discretization of the curl operator, mapping from the Nédélec space into the RT or BDM space. Here, we use $\hat{\mathbf{E}}$ and $\hat{\mathbf{B}}$ to denote the coefficients of the finite-element basis functions for $\mathbf{E}$ and $\mathbf{B}$, respectively. To fully discretize the variational form, we then apply IRK temporal discretization to (5.7), which leads to the following block-structured system of equations,

$$\left(I_r\otimes\begin{bmatrix} M_{\mathbf{E}} & 0\\ 0 & M_{\mathbf{B}}\end{bmatrix} + \Delta tA\otimes\begin{bmatrix} 0 & -c^2L^T\\ L & 0\end{bmatrix}\right)\vec{\mathbf{k}} = \vec{\mathbf{F}}, \tag{5.8}$$

where $\vec{\mathbf{F}}$ is the corresponding right-hand side vector and $\vec{\mathbf{k}}$ is the vector of IRK stage values for the system.

5.4.1 Monolithic multigrid for Maxwell's equations

As above, we solve the linear system of equations in 5.8 using monolithic multigrid as a preconditioner for FGMRES. We construct mesh hierarchies using the refine-project-elasticity solve strategy proposed above, using finite-element interpolation to map approximations to $\mathbf{B}$ and $\mathbf{E}$ from one mesh to the next. If the multigrid interpolation operator for a single temporal stage is given by

$$\hat{P} = \begin{bmatrix} P_{\mathbf{E}} & \\ & P_{\mathbf{B}} \end{bmatrix},$$

then the multistage interpolation operator is given by $P = I_r \otimes \hat{P}$. The coarsest-grid system is solved by a direct solver.

As a relaxation scheme, we use an extension of the Arnold-Falk-Winther relaxation from [7], which is closely related to the Vanka relaxation [59] commonly used in monolithic multigrid solvers for the Stokes and Navier-Stokes equations. Both schemes can be viewed as overlapping Schwarz methods; here, we use the additive variants. For the Schwarz subdomains (also known as *patches* in the multigrid literature), we use the topological star around each vertex, following the terminology of [24]. For each vertex in the mesh, we form a patch that includes all DoFs defined at that vertex, as well as those DoFs on the interior of all edges, faces, and volumes adjacent to that vertex. The key difference between these patches and Vanka patches is that we do not include any DoFs on the closure of star. Sample patches for the first-order RT-N1Curl discretization are shown at left of Figure 5.3, while those for the first-order BDM-N2Curl discretization are shown at right of Figure 5.3. We note that the patches used include DoFs for all IRK stages at these mesh locations.

Let $\mathcal{S}_i$ denote the set of DoFs contained in the i^{th} vertex star patch, noting that $\mathcal{S} = \bigcup_{i=1}^{N} \mathcal{S}_i$, where N is the total number of patches (vertices in the mesh) and $\mathcal{S}$ is the set of all DoFs for the problem, noting that each DoF is contained in at least one patch, but that a DoF may appear in multiple patches. We use R_i to denote the "restriction" operator that maps global DoFs from $\mathcal{S}$ to those in $\mathcal{S}_i$, making R_i a matrix of size $|\mathcal{S}_i| \times |\mathcal{S}|$, with each row containing a single non-zero entry of one in the column corresponding to the global index associated with that local index of $\mathcal{S}_i$.

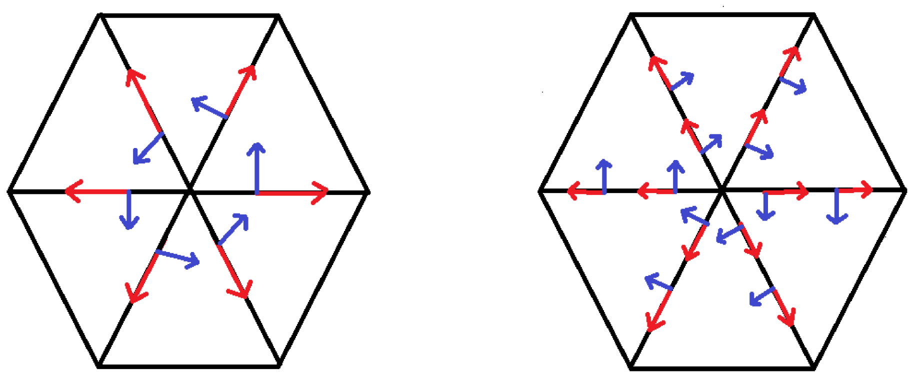


Figure 5.3: Vertex star patches in 2D for first degree RT-N1Curl (left) and BDM-N2Curl (right) discretizations. Red arrows denote Nédélec DoFs, while blue arrows denote either RT or BDM DoFs.

With this, we can write a single iteration of a weighted stationary iteration as

$$\vec{\mathbf{k}} \leftarrow \vec{\mathbf{k}} + \omega \sum_{i=1}^{N} R_i^T (R_i Q R_i^T)^{-1} R_i (\vec{\mathbf{F}} - Q\vec{\mathbf{k}}),$$

where $Q\vec{\mathbf{k}} = \vec{\mathbf{F}}$ is the linear system to be solved, and $R_i Q R_i^T$ is the restriction of Q to the DoFs in patch $\mathcal{S}_i$. In the numerical results that follow, we accelerate the convergence of the relaxation process by using GMRES with 2 or 3 pre- and post-relaxation sweeps on each level of the multigrid hierarchy, as preliminary experiments showed this to be slightly more efficient.

5.5 Numerical results

In this section, we present results for Maxwell's equations on a cube and sphere. For the temporal integration, we focus on 3 families of IRK methods, LobattoIIIC, RadauIIA, and Gauss-Legendre (Gauss) methods, using 2 or 3 stages. Both LobattoIIIC and RadauIIA are L-stable IRK methods, while Gauss is A-stable but symplectic. For an r-stage method, LobattoIIIC schemes have order $2r-2$, RadauIIA schemes have order $2r-1$, while Gauss schemes have order $2r$, although all have stage order r. Details of the Butcher tables of these methods can be found in [61, 60]. For the spatial discretization, we focus on using either first or second-degree spaces, denoted as RT(k)-N1Curl(k) or BDM(k)-N2Curl(k), where k is the chosen spatial degree.

We consider a time-periodic solution given by

$$\begin{bmatrix}\mathbf{E}(\mathbf{x},t)\\ \mathbf{B}(\mathbf{x},t)\end{bmatrix} = \begin{bmatrix}\mathbf{E}_0\cos(\mathbf{k}\cdot\mathbf{x}+\omega t)\\ \mathbf{B}_0\cos(\mathbf{k}\cdot\mathbf{x}+\omega t)\end{bmatrix}$$

with $\mathbf{E}_0 = [0,1,0]$, $\mathbf{k} = [1,0,0]$, and $\mathbf{B}_0 = \frac{-\mathbf{k}\times\mathbf{E}_0}{c}$. For this solution, $\mathbf{J} = \mathbf{0}$. The initial condition is given by interpolating the exact solution into the finite-element space at the initial time, $t = 0$, integrating to the final time, which is fixed to be $Tf = \frac{10\pi}{c}$. In both cases that follow, we fix the stopping criterion for FGMRES to be either a relative reduction in the residual norm by a a factor of 10^{-14} or for the absolute value of the residual norm to be reduced below $10^{-2}/N^2$, where N is (roughly) the number of elements in one dimension of the mesh, as described below.

5.5.1 Cube domain

For this model, we construct the coarsest grid by cutting the unit cube domain, $\Omega = [0,1]^3$, into $2\times2\times2$ cubic elements, each of which is cut into 6 tetrahedra. This coarsest mesh is then refined ℓ times for $1 \le \ell \le 4$. Table 5.5 shows the total number of DoFs **per stage** for each problem considered. We present results for a weak scaling study. For the first-order spatial discretization, we use 1 core for $\ell = 2$, 8 cores for $\ell = 3$, and 32 cores for $\ell = 4$ (the full machine). For the second-order spatial discretization, we use 1 core for $\ell = 1$, 8 cores for $\ell = 2$, and 32 cores for $\ell = 3$.

	1	2	3	4
RT(1)-N1Curl(1)	-	10712	81712	638048
BDM(1)-N2Curl(1)	-	27952	214112	1675456
RT(2)-N1Curl(2)	6680	50224	389216	-
BDM(2)-N2Curl(2)	11892	89736	696720	-

Table 5.5: Number of DoFs per stage for first- and second-degree spaces on a cube.

We first verify that we are achieving the expected discretization accuracy for all discretizations. For RT(1)-N1Curl(1), we use $\Delta t = \frac{Tf}{8N}$, while $\Delta t = \frac{Tf}{16N}$ is used for BDM(1)-N2Curl(1), RT(2)-N1Curl(2) and BDM(2)-N2Curl(2), where $N = 2^{\ell+1}$. Figures 5.4 and 5.5 show the L_2 errors for both $\mathbf{B}$ and $\mathbf{E}$ using the first- and second-degree spatial discretizations, respectively, with different 2- and 3-stage IRK schemes. We expect lowest accuracy with RT(1)-N1Curl(1), for which we expect first-order convergence in the L_2 errors, which is achieved in Figure 5.4. Second-order convergence is observed for some integrators with either BDM(1)-N2Curl(1) or RT(2)-N1Curl(2), but we note that the LobattoIIIC integrator clearly suffers. Finally, we observe nearly third-order convergence for the BDM(2)-N2Curl(2) discretization, again aside from the LobattoIIIC integrator.

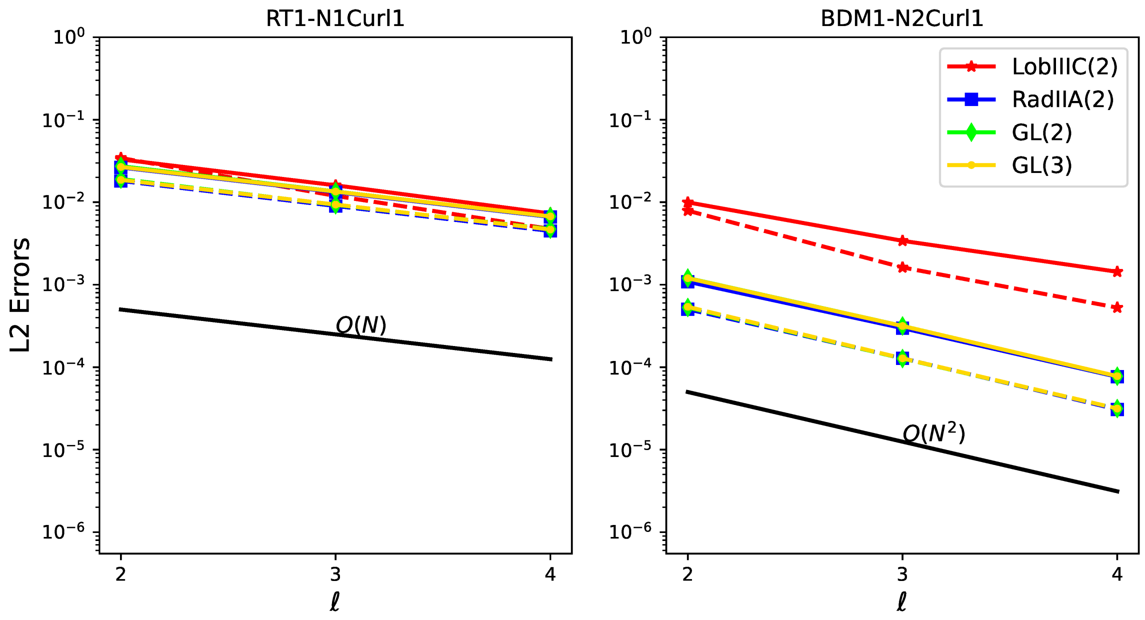


Figure 5.4: L_2 errors of Maxwell's equation on a unit cube with various refinement levels and time integrators for first-degree spatial discretizations. Dashed lines indicate errors in $\mathbf{B}$, solid lines indicate errors in $\mathbf{E}$.

We note two positive properties of the Gauss-Legendre integrator used here. While it is not L-stable, the natural solutions of this form of Maxwell's equations take the form of propagating waves, rather than diffusive solutions, so L-stability is not a natural requirement. However, Gauss-Legendre RK schemes are known for their geometric properties, including symplecticity and preservation of constraints that are linear or quadratic in the solution [30]. Thus, for a problem such as this one, with associated linear constraints on $\nabla\cdot\mathbf{E}$ and $\nabla\cdot\mathbf{B}$, Gauss-Legendre is a natural integrator. Furthermore, it also achieves a stage order equal to the number of stages, giving us the best-possible accuracy for problems such as these, where stiffness and DAE structure of boundary conditions leads to stage order determining the accuracy of the scheme.

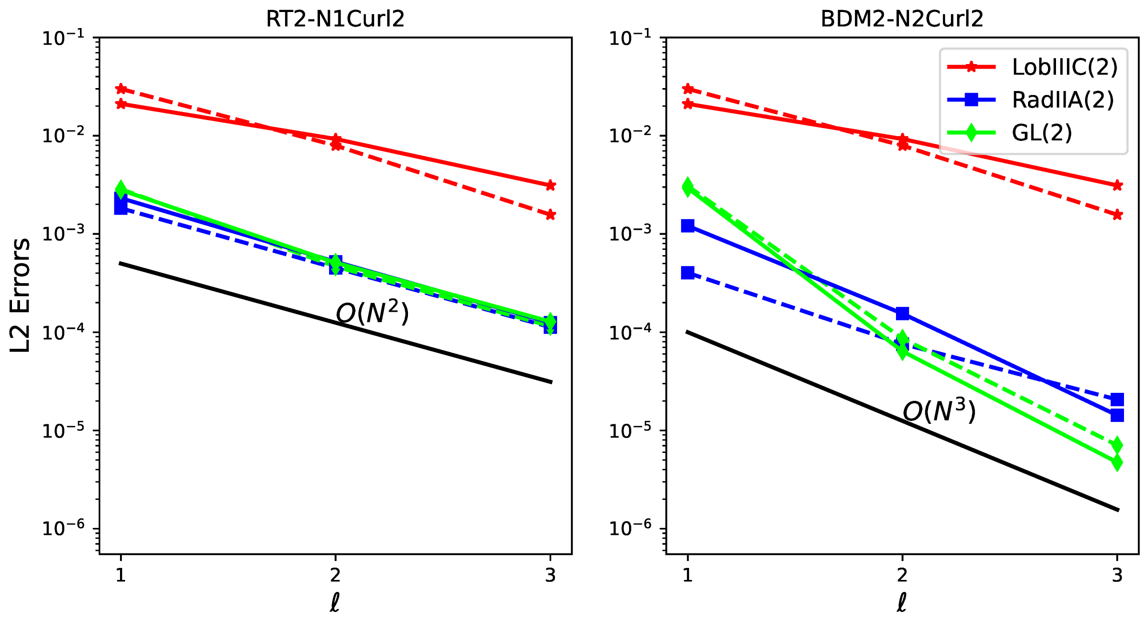


Figure 5.5: L_2 errors of Maxwell's equation on a unit cube with various refinement levels and time integrators for second-degree spatial discretizations. Dashed lines indicate errors in $\mathbf{B}$, solid lines indicate errors in $\mathbf{E}$.

Table 5.6 presents computational time and linear iteration counts per time step, both averaged over all time steps, for these numerical results. We note very steady iteration counts for all cases, with some increase for the lowest-order discretization, using RT(1)-N1Curl(1). We note that the somewhat larger than expected weak scaling for the final level of refinement is primarily due to the limitation of 32 cores on this machine, so the final scaling is not "true" weak scaling with the spatial problem size. Most importantly, with higher-order discretizations and integrators, we see no degradation in solver performance, and only modest increases in time-to-solution in comparison with the lowest-order integrator.

Table 5.7 explores the robustness of the solver with respect to Δt for the various spatial discretizations and 2-stage Gauss Runge-Kutta scheme. Here, we see very little variation in iteration counts as we change Δt. We note that the averaged time per time step changes only a little with Δt (primarily due to small changes in iteration counts), but that this hides the fact that as Δt decreases, more time steps are needed to reach the same final time, T_f.

5.5.2 Sphere domain

Finally, to solve (5.5) on a sphere, we construct a coarsest grid by approximating the unit sphere by a mesh of 47 tetrahedral elements that are then refined ℓ times for $1 \leq \ell \leq 4$. Table 5.8 presents the total number of DoFs **per stage** for these experiments. In the weak scaling study for this model, for the first-degree discretizations, we use 1 core for both $\ell = 1$ and 2, 8 cores for $\ell = 3$, and 32 cores for $\ell = 4$. For the second-degree discretizations, we use 1 core for $\ell = 1$, 4 cores for $\ell = 2$, and 16 cores for $\ell = 3$.

As even the coarsest mesh for this experiment is unstructured, we use $N = 2^{2+\ell}$ as an approximation of the number of elements in one dimension. With this, we use the time step $\Delta t = \frac{T_f}{4N}$ for RT(1)-N1Curl(1), BDM(1)-N2Curl(1) and RT(2)-N1Curl(2), while $\Delta t = \frac{T_f}{16N}$ for BDM(2)-N2Curl(2). Figures 5.6 and 5.7 show the L_2 errors for both $\mathbf{B}$ and $\mathbf{E}$ using the first and degree spatial discretizations, respectively, with different 2- and 3-stage IRK schemes. We clearly see the expected first- and second -order convergence for the RT(k)-N1Curl(k) spaces using $k = 1, 2$ respectively. For BDM(k)-N2Curl(k) spaces, we see that the 2-stage Gauss scheme shows slightly better than 3rd-order convergence. As before, the Lobatto integrator is least accurate, with comparable accuracy shown by both RadauIIA and Gauss-Legendre. For the low-order discretizations, we see little improvement going from 2 to 3 stages in the Gauss-Legendre results, suggesting that spatial discretization error is dominant here (noting that available memory precluded any experimenting with GL(3) and $\ell = 4$ for BDM(1)-N1Curl(1) for this test problem).

Iteration counts and time-to-solution per time step, averaged over all time steps, are presented in Table 5.9. As for the unit cube domain, these are generally robust. Here, we again see some poorer performance for the lowest-order discretization, RT(1)-N1Curl(1), particularly for the LobattoIIIC discretization.

A common concern with Runge-Kutta methods is the phenomenon of order reduction, which is known to occur for stiff ODEs or differential-algebraic equations. Order reduction describes cases where the actual convergence observed for a given scheme is worse than the theoretical convergence anticipated [60]. Here, stage order dominates convergence behaviour, not scheme order. For example, in Table 5.10, we observe a sharp drop in convergence using BDM(2)-DG(2) and 2-stage Gauss-Legendre when increasing the time step size. Although 2-stage Gauss-Legendre is a fourth-order

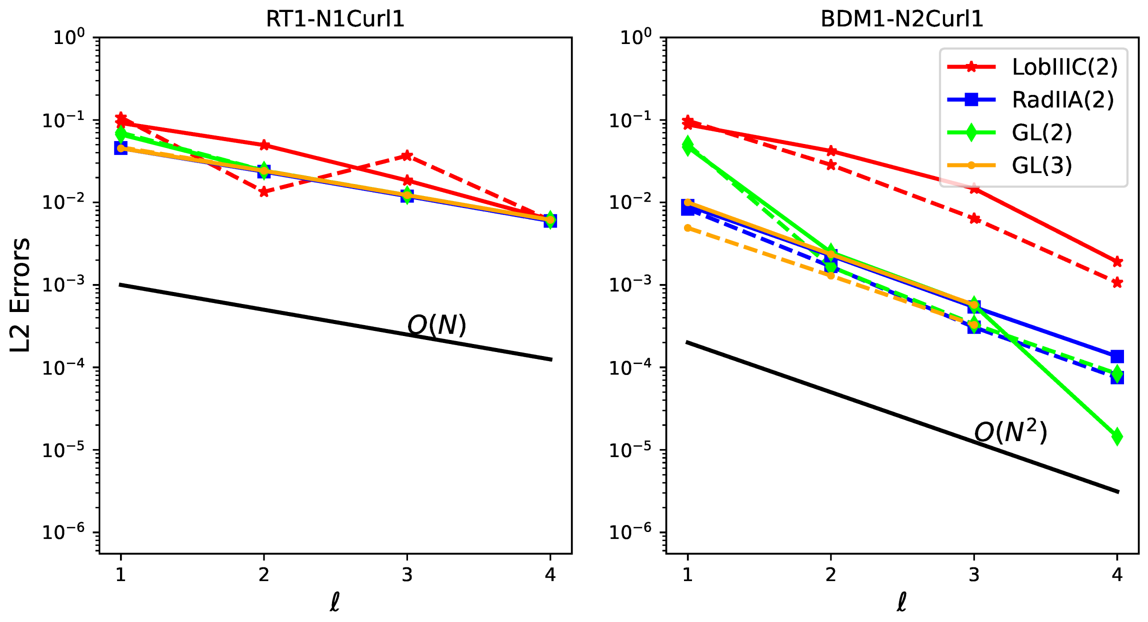


Figure 5.6: L_2 errors of Maxwell's equation on a unit sphere with various refinement levels and time integrators for first-degree spatial discretizations. Dashed lines indicate errors in $\mathbf{B}$, solid lines indicate errors in $\mathbf{E}$.

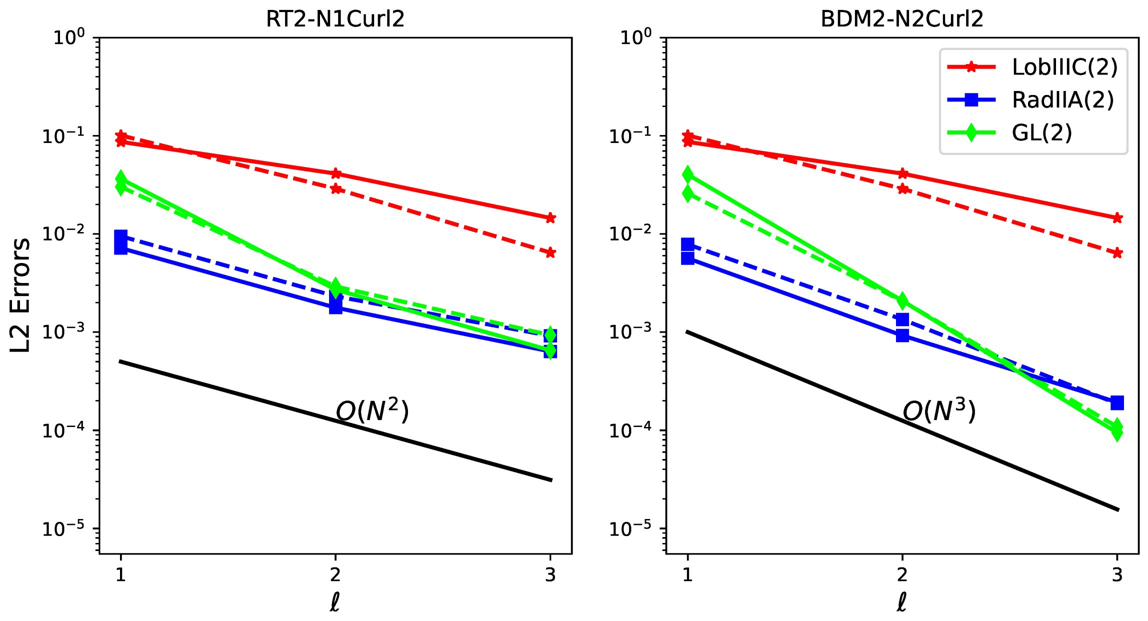


Figure 5.7: L_2 errors of Maxwell's equation on a unit sphere with various refinement levels and time integrators for second-degree spatial discretizations. Dashed lines indicate errors in $\mathbf{B}$, solid lines indicate errors in $\mathbf{E}$.

method and the expected spatial convergence is third order, the best convergence we observe is (roughly) second order, because 2-stage Gauss-Legendre has a stage order of 2. An obvious way to improve these results is to increase the number of stages in the IRK scheme used; however, this can be impractical either due to resource restrictions, as in our case, or the increase of computational time with more stages. Significant work has been done in developing techniques to decrease or eliminate the effects of order reduction [15, 57, 49, 26, 1]; however, we do not investigate this further.

5.6 Conclusion

In this paper, we extend recent work (cf. [2]) on monolithic multigrid preconditioners to the **B**-**E** form of Maxwell's equations, showing effective results for various mixed finite-element spatial discretizations and IRK time integrators. Robustness is seen across two types of **H**(div)-**H**(curl) conforming finite-element spaces of first and second degree, including on a unit sphere domain, showing capabilities on more realistic geometries. To achieve this convergence, care must be taken in mesh construction, using both isoparametric elements and adapting the mesh hierarchy using elasticity solves to improve mesh quality.

Future work includes extending this to even more realistic geometries, such as the tokamak geomtry used in fusion reactors. Furthermore, we look to couple the Maxwell solver developed here with a multi-fluid model for the plasma, to yield a high-fidelity simulation framework for plasmas in this regime. Finally, we note that extending three-dimensional simulations to high order (beyond the second-order elements considered here) requires significant resources, either in terms of memory-per-core on a parallel machine, or to develop low-memory intensity variants to make higher-order discretizations feasible.

	RT(1)-N1Curl(1)			BDM(1)-N2Curl(1)		
	ℓ	time	iteration	ℓ	time	iteration
GL(2)	2	0.04	11.9	2	0.05	7.0
	3	0.06	13.6	3	0.14	7.9
	4	0.27	14.4	4	0.56	8.0
Lob(2)	2	0.03	12.1	2	0.04	9.0
	3	0.06	14.6	3	0.09	9.0
	4	0.27	15.1	4	0.67	10.0
Rad(2)	2	0.04	12.4	2	0.04	7.9
	3	0.06	14.5	3	0.09	8.6
	4	0.30	15.7	4	0.66	9.0
GL(3)	2	0.07	13.7	2	0.03	8.0
	3	0.11	15.0	3	0.08	9.0
	4	0.54	16.0	4	1.42	9.0
	RT(2)-N1Curl(2)			BDM(2)-N2Curl(2)		
	ℓ	time	iteration	ℓ	time	iteration
GL(2)	1	0.06	6.0	1	0.14	5.8
	2	0.19	7.0	2	0.40	6.0
	3	0.51	8.0	3	1.06	6.6
Lob(2)	1	0.03	7.0	1	0.08	6.0
	2	0.12	8.4	2	0.18	6.7
	3	0.38	10.0	3	0.96	7.0
Rad(2)	1	0.05	6.9	1	0.08	5.9
	2	0.18	8.0	2	0.18	6.0
	3	0.36	9.0	3	0.97	7.0

Table 5.6: Linear iteration counts per time step and computational time (in minutes) needed per time step, averaged over all time steps, for first- and second-order spatial discretizations and various IRK temporal discretizations for Maxwell's equations on the unit cube.

	RT(1)-N1Curl(1)			BDM(1)-N2Curl(1)		
	ℓ	time	iteration	ℓ	time	iteration
$\Delta t = \frac{T_f}{2N}$	2	0.03	13.4	2	-	-
	3	0.09	15.4	3	-	-
$\Delta t = \frac{T_f}{4N}$	2	0.02	12.9	2	0.04	9.0
	3	0.08	14.3	3	0.13	9.5
$\Delta t = \frac{T_f}{8N}$	2	0.04	11.9	2	0.04	8.0
	3	0.06	13.6	3	0.10	8.2
$\Delta t = \frac{T_f}{16N}$	2	-	-	2	0.05	7.0
	3	-	-	3	0.14	7.9
	RT(2)-N1Curl(2)			BDM(2)-N2Curl(2)		
	ℓ	time	iteration	ℓ	time	iteration
$\Delta t = \frac{T_f}{4N}$	2	0.22	9.0	2	0.43	7.0
	3	0.61	10.0	3	1.21	7.9
$\Delta t = \frac{T_f}{8N}$	2	0.44	9.9	2	0.41	6.8
	3	0.55	11.0	3	1.43	8.8
$\Delta t = \frac{T_f}{16N}$	2	0.18	8.0	2	0.18	6.0
	3	0.36	9.0	3	0.97	7.0

Table 5.7: Linear iteration counts per time step and computational time (in minutes) needed per time step, averaged over all time steps, for first- and second-order spatial discretizations with 2-stage Gauss-Legendre RK and varying time steps for Maxwell's equations on the unit cube.

	1	2	3	4
RT(1)-N1Curl(1)	2200	16064	122560	957056
BDM(1)-N2Curl(1)	5696	41920	321152	2513152
RT(2)-N1Curl(2)	10016	75328	583808	-
BDM(2)-N2Curl(2)	17832	134592	1045056	-

Table 5.8: Number of DoFs per stage for first- and second-degree spaces on the sphere.

	RT(1)-N1Curl(1)			BDM(1)-N2Curl(1)		
	ℓ	time	iteration	ℓ	time	iteration
GL(2)	1	0.02	9.3	1	0.02	7.3
	2	0.11	12.3	2	0.13	8.9
	3	0.23	13.9	3	0.28	10.0
	4	0.41	15.7	4	1.15	11.7
Lob(2)	1	0.02	10.1	1	0.02	8.9
	2	0.08	14.9	2	0.18	10.0
	3	0.36	23.1	3	0.22	12.0
	4	1.97	69.7	4	1.35	14.6
Rad(2)	1	0.02	10.8	1	0.02	7.9
	2	0.10	15.7	2	0.18	9.0
	3	0.23	21.6	3	0.22	11.0
	4	0.56	23.1	4	1.35	12.7
GL(3)	1	0.03	10.0	1	0.06	8.9
	2	0.13	15.1	2	0.17	10.7
	3	0.55	17.4	3	0.42	12.9
	4	0.91	20.4	4	-	-
	RT(2)-N1Curl(2)			BDM(2)-N2Curl(2)		
	ℓ	time	iteration	ℓ	time	iteration
GL(2)	1	0.21	6.6	1	0.37	6.0
	2	0.60	8.0	2	1.24	7.0
	3	1.16	9.0	3	2.05	8.0
Lob(2)	1	0.19	7.0	1	0.26	6.0
	2	0.62	9.0	2	1.26	7.9
	3	1.03	9.6	3	2.38	9.0
Rad(2)	1	0.23	7.0	1	0.39	6.0
	2	0.63	9.01	2	1.14	7.0
	3	1.27	10.6	3	2.13	8.6

Table 5.9: Linear iteration counts per time step and computational time (in minutes) needed per time step, averaged over all time steps, for first- and second-order spatial discretizations and various IRK temporal discretizations for Maxwell's equations on the unit sphere.

	BDM(2)-N2Curl(2)		
	ℓ	L_2 errors $\mathbf{E}$	L_2 errors $\mathbf{B}$
$\Delta t = \frac{T_f}{2N}$	1	4.250×10^{-1}	5.547×10^{-2}
	2	2.318×10^{-2}	6.269×10^{-2}
	3	4.829×10^{-3}	4.652×10^{-3}
$\Delta t = \frac{T_f}{4N}$	1	4.026×10^{-2}	2.590×10^{-2}
	2	2.079×10^{-3}	2.071×10^{-3}
	3	9.461×10^{-5}	1.084×10^{-4}
$\Delta t = \frac{T_f}{8N}$	1	6.779×10^{-4}	7.262×10^{-4}
	2	8.031×10^{-5}	8.495×10^{-5}
	3	2.487×10^{-5}	3.151×10^{-5}
$\Delta t = \frac{T_f}{16N}$	1	2.964×10^{-4}	2.534×10^{-4}
	2	7.288×10^{-5}	7.027×10^{-5}
	3	2.466×10^{-5}	3.124×10^{-5}

Table 5.10: For Maxwell's equations on a unit sphere, the L_2 errors in $\mathbf{E}$ and $\mathbf{B}$ using different sized time steps with BDM(2)-DG(2) and the 2-stage Gauss for temporal discretization is recorded.

Chapter 5 References

[1] A. Abdulle and G. R. de Souza. Instabilities and order reduction phenomenon of an interpolation based multirate Runge-Kutta-Chebyshev method. *arXiv preprint arXiv:2003.03154*, 2020.

[2] R. Abu-Labdeh, S. MacLachlan, and P. E. Farrell. Monolithic multigrid for implicit Runge–Kutta discretizations of incompressible fluid flow. *Journal of Computational Physics*, 478:111961, 2023.

[3] J. H. Adler, T. R. Benson, E. C. Cyr, P. E. Farrell, S. P. MacLachlan, and R. S. Tuminaro. Monolithic multigrid methods for magnetohydrodynamics. *SIAM Journal on Scientific Computing*, 43(5):S70–S91, 2021.

[4] J. H. Adler, T. R. Benson, E. C. Cyr, S. P. MacLachlan, and R. S. Tuminaro. Monolithic multigrid methods for two-dimensional resistive magnetohydrodynamics. *SIAM Journal on Scientific Computing*, 38(1):B1–B24, 2016.

[5] J. H. Adler, X. Hu, and L. T. Zikatanov. Robust solvers for Maxwell's equations with dissipative boundary conditions. *SIAM Journal on Scientific Computing*, 39(5):S3–S23, 2017.

[6] M.-Á. Aloy and I. Cordero-Carrión. Minimally implicit Runge-Kutta methods for resistive relativistic MHD. In *Journal of Physics: Conference Series*, volume 719, page 012015. IOP Publishing, 2016.

[7] D. N. Arnold, R. S. Falk, and R. Winther. Multigrid in H(div) and H(curl). *Numerische Mathematik*, 85(2):197–217, 2000.

[8] F. Assous, P. Degond, E. Heintze, P.-A. Raviart, and J. Segré. On a finite-element method for solving the three-dimensional Maxwell equations. *Journal of Computational Physics*, 109(2):222–237, 1993.

[9] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al. PETSc users manual: Revision 3.10. Technical report, Office of Scientific and Technical Information (OSTI), 2018.

[10] M. Benzi, M. A. Olshanskii, and Z. Wang. Modified augmented Lagrangian preconditioners for the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 66(4):486–508, 2011.

[11] Y. Berchenko-Kogan and A. Stern. Constraint-preserving hybrid finite element methods for Maxwell's equations. *Foundations of Computational Mathematics*, 21(4):1075–1098, 2021.

[12] H. bin Zubair Syed, C. Farquharson, and S. MacLachlan. Block preconditioning techniques for geophysical electromagnetics. *SIAM J. Sci. Comp.*, 42(3):B696–B721, 2020.

[13] J. A. Bittencourt. *Fundamentals of plasma physics*. Springer Science & Business Media, 2004.

[14] T. Boonen, J. Van lent, and S. Vandewalle. An algebraic multigrid method for high order time-discretizations of the div-grad and the curl-curl equations. *Applied Numerical Mathematics*, 59(3):507–521, 2009.

[15] K. Burrage and L. Petzold. On order reduction for Runge–Kutta methods applied to differential/algebraic systems and to stiff systems of ODEs. *SIAM Journal on Numerical Analysis*, 27(2):447–456, 1990.

[16] J. C. Butcher. On the implementation of implicit Runge-Kutta methods. *BIT Numerical Mathematics*, 16(3):237–240, 1976.

[17] J. C. Butcher. General linear methods. *Acta Numerica*, 15:157–256, 2006.

[18] Z. Chen, Q. Du, and J. Zou. Finite element methods with matching and nonmatching meshes for Maxwell equations with discontinuous coefficients. *SIAM Journal on Numerical Analysis*, 37(5):1542–1570, 2000.

[19] P. G. Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[20] C. H. Cooke and D. K. Blanchard. A higher order finite element algorithm for the unsteady Navier-Stokes equations. *Mathematics and Computers in Simulation*, 22(2):127–132, 1980.

[21] H. Damanik, J. Hron, A. Ouazzi, and S. Turek. Monolithic Newton-multigrid solution techniques for incompressible nonlinear flow models. *International Journal for Numerical Methods in Fluids*, 71(2):208–222, 2013.

[22] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 227(3):1790–1808, 2008.

[23] P. E. Farrell, R. C. Kirby, and J. Marchena-Menendez. Irksome: Automating Runge–Kutta time-stepping for finite element methods. *ACM Transactions on Mathematical Software*, 47(4):1–26, 2021.

[24] P. E. Farrell, M. G. Knepley, L. Mitchell, and F. Wechsung. PCPATCH: software for the topological construction of multigrid relaxation methods. *ACM Transactions on Mathematical Software*, 47(3):1–22, 2021.

[25] P. E. Farrell, L. Mitchell, L. R. Scott, and F. Wechsung. A Reynolds-robust preconditioner for the Scott-Vogelius discretization of the stationary incompressible Navier-Stokes equations. *The SMAI Journal of Computational Mathematics*, 7:75–96, 2021.

[26] R. Frank, J. Schneid, and C. W. Ueberhuber. Order results for implicit Runge–Kutta methods applied to stiff systems. *SIAM journal on numerical analysis*, 22(3):515–534, 1985.

[27] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala. Ml 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[28] A. V. Grayver and T. V. Kolev. Large-scale 3D geoelectromagnetic modeling using parallel adaptive high-order finite element method. *Geophysics*, 80(6):E277–E291, 2015.

[29] C. Greif and D. Schötzau. Preconditioners for the discretized time-harmonic Maxwell equations in mixed form. *Numerical Linear Algebra with Applications*, 14(4):281–297, 2007.

[30] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration. Structure-preserving algorithms for ordinary differential equations. 2nd ed*, volume 31. 01 2006.

[31] R. Hiptmair. Multigrid method for Maxwell's equations. *SIAM Journal on Numerical Analysis*, 36(1):204–225, 1998.

[32] R. Hiptmair and J. Xu. Nodal auxiliary space preconditioning in H(curl) and H(div) spaces. *SIAM Journal on Numerical Analysis*, 45(6):2483–2509, 2007.

[33] M. Hochbruck and T. Pazur. Implicit Runge–Kutta methods and discontinuous Galerkin discretizations for linear Maxwell's equations. *SIAM Journal on Numerical Analysis*, 53(1):485–507, 2015.

[34] J. J. Hu, R. S. Tuminaro, P. B. Bochev, C. J. Garasi, and A. C. Robinson. Toward an h-independent algebraic multigrid method for Maxwell's equations. *SIAM Journal on Scientific Computing*, 27(5):1669–1688, 2006.

[35] K. Hu, Y. Ma, and J. Xu. Stable finite element methods preserving $\nabla \cdot B = 0$ exactly for MHD models. *Numerische Mathematik*, 135(2):371–396, 2017.

[36] K. Hu, W. Qiu, and K. Shi. Convergence of a $B-E$ based finite element method for MHD models on Lipschitz domains. *Journal of Computational and Applied Mathematics*, 368:112477, 2020.

[37] J. D. Jackson. Classical electrodynamics, 1999.

[38] V. John and L. Tobiska. Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 33(4):453–473, 2000.

[39] T. V. Kolev and P. S. Vassilevski. Parallel auxiliary space AMG for H(curl) problems. *Journal of Computational Mathematics*, pages 604–623, 2009.

[40] F. Laakmann, P. E. Farrell, and L. Mitchell. An augmented Lagrangian preconditioner for the magnetohydrodynamics equations at high Reynolds and coupling numbers. *SIAM Journal on Scientific Computing*, 44(4):B1018–B1044, 2022.

[41] F. Laakmann, K. Hu, and P. E. Farrell. Structure-preserving and helicity-conserving finite element approximations and preconditioning for the hall mhd equations. *Journal of Computational Physics*, page 112410, 2023.

[42] M. Lange, L. Mitchell, M. G. Knepley, and G. J. Gorman. Efficient mesh management in firedrake using PETSC DMPLEX. *SIAM Journal on Scientific Computing*, 38(5):S143–S155, 2016.

[43] S. T. Miller, E. C. Cyr, J. N. Shadid, R. M. J. Kramer, E. G. Phillips, S. Conde, and R. P. Pawlowski. IMEX and exact sequence discretization of the multi-fluid plasma model. *Journal of Computational Physics*, 397:108806, 2019.

[44] P. Monk. *Finite element methods for Maxwell's equations.* Oxford University Press, 2003.

[45] J.-C. Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35:315–341, 1980.

[46] P. Ohm, T. Wiesner, E. C. Cyr, J. J. Hu, J. N. Shadid, and R. S. Tuminaro. A monolithic algebraic multigrid framework for multiphysics applications with examples from resistive MHD. *arXiv preprint arXiv:2103.07537*, 2021.

[47] C. Pagliantini, G. Manzini, O. Koshkarov, G. L. Delzanno, and V. Roytershteyn. Energy-conserving explicit and implicit time integration methods for the multi-dimensional Hermite-DG discretization of the Vlasov-Maxwell equations. *Computer Physics Communications*, 284:108604, 2023.

[48] P.-O. Persson and J. Peraire. Curved mesh generation and mesh refinement using Lagrangian solid mechanics. In *47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 949, 2009.

[49] L. R. Petzold. Order results for implicit Runge–Kutta methods applied to differential/algebraic systems. *SIAM Journal on Numerical Analysis*, 23(4):837–852, 1986.

[50] E. G. Phillips, J. N. Shadid, and E. C. Cyr. Scalable preconditioners for structure preserving discretizations of Maxwell equations in first order form. *SIAM Journal on Scientific Computing*, 40(3):B723–B742, 2018.

[51] E. G. Phillips, J. N. Shadid, E. C. Cyr, and S. T. Miller. Enabling scalable multifluid plasma simulations through block preconditioning. *Numerical Methods for Flows: FEF 2017 Selected Contributions*, pages 231–244, 2020.

[52] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. McRae, G.-T. Bercea, G. R. Markall, and P. H. Kelly. Firedrake: automating the finite element method by composing abstractions. *ACM Transactions on Mathematical Software*, 43(3):1–27, 2016.

[53] E. Rosseel, T. Boonen, and S. Vandewalle. Algebraic multigrid for stationary and time-dependent partial differential equations with stochastic coefficients. *Numer. Linear Algebra Appl.*, 15(2-3):141–163, 2008.

[54] G. Rosser. *Interpretation of classical electromagnetism*, volume 78. Springer Science & Business Media, 2013.

[55] W. E. Schiesser. *The numerical method of lines: integration of partial differential equations*. Elsevier, 2012.

[56] D. Schötzau. Mixed finite element methods for stationary incompressible magneto–hydrodynamics. *Numerische Mathematik*, 96(4):771–800, 2004.

[57] L. M. Skvortsov. How to avoid accuracy and order reduction in Runge–Kutta methods as applied to stiff problems. *Computational Mathematics and Mathematical Physics*, 57:1124–1139, 2017.

[58] J. Van Lent and S. Vandewalle. Multigrid methods for implicit Runge–Kutta and boundary value method discretizations of parabolic PDEs. *SIAM Journal on Scientific Computing*, 27(1):67–92, 2005.

[59] S. Vanka. Block-implicit calculation of steady turbulent recirculating flows. *International Journal of Heat and Mass Transfer*, 28(11):2093–2103, 1985.

[60] G. Wanner and E. Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg, 1996.

[61] Wikipedia. List of Runge-Kutta methods. `https://en.wikipedia.org/wiki/List_of_Runge-Kutta_methods`, 2021. [Online; accessed 11-September-2023].

[62] X. Zhang and W. Zheng. Monolithic multigrid for reduced magnetohydrodynamic equations. *Journal of Computational Mathematics*, 39(3):453, 2021.

Chapter 6

Conclusion

In this thesis, our main interest has been the development of an efficient and robust solver framework for systems of saddle-point type that arise from fully discretizing fluid flow models. A particular interest has been the use of fully implicit Runge-Kutta methods for the temporal discretization, resulting in very large and difficult to solve systems of equations at each time step. Within this Newton-Krylov-Multigrid method we have implemented monolithic Vanka relaxation schemes, with varying patches appropriate to the mixed finite-element spaces applied, which allows us to preserve the coupling between the unknowns in problem (if any) when solving the system as a whole. We have successfully shown this solver to be effective for several incompressible Newtonian and magnetohydrodynamic flow problems in two and three dimensional domains.

We then implemented an extension of this solver to achieve high-fidelity approximations using higher-order discretizations in both space and time focusing on two well-known difficult problems, the Navier-Stokes equations and Maxwell's equations, again in two and three dimensions. For the Navier-Stokes models, an H(div)-DG finite-element space was used with increasingly higher orders. A Burmann stabilization penalty term was added in for testing accuracy of solutions with higher Reynolds numbers which was found to be the right strategy to achieve robustness. For Maxwell's equations, we specifically considered the lesser used **E**-**B** formulation, where we have shown by using an extended Vanka relaxation patch accounting for the H(div)-H(curl) conforming finite-element spaces used that very robust iteration counts and computational times are produced. Here, since our motivation was solving Maxwell's equations on more realistic domains that are often curved at the boundary, we investigated our work on cubical and spherical shaped meshes. We have found it necessary to include an elasticity solve on each level of the non-nested mesh hierarchy to attain correct convergence rates and properly represent the domain near the curved boundary in the discretized meshes.

There are several possibilities for future work based on the research presented in this thesis:

- We require a better understanding of the stopping tolerances in a way that will allow us to obtain highly-accurate solutions, specifically for the Navier-Stokes and Maxwell's equations models considered, using IRK temporal discretizations and FEM spaces as those used in this thesis with orders and degrees higher than 6 and 5 respectively.

- Implement our solver for more realistic geometries where Maxwell's equations are currently of interest, such as the torus domain. This is seen in many applications of plasma flow models on a tokamak that can be represented by the MHD problem.

- All finite-element discretizations in this thesis were defined on a triangulation of the domains, whether in two or three dimensions. Instead, we have the option of constructing the discrete meshes using quadrilaterals or hexaderas. Many inf-sup stable finite-element spaces similar to those discussed here, such as RTcf and BDMcf spaces, exist on these element shapes. Through preliminary experimentations, we have found that iteration counts can be decreased using these discretizatations. A more extensive study in this direction is a very valid next consideration.

- For fluid flow problems specifically, we can also further decrease the nonlinear iteration counts by using Impicit-Explict (IMEX) Runge-Kutta time stepping methods. This leaves room for a further study of extending our method to be implemented in this setting as the chosen implicit solver for the fast dynamic terms. Very preliminary results on a simple model of the Navier-Stokes equations show this can be successfully achieved. An interesting future study would be comparing this technique with the fully implicit Runge-Kutta schemes used in this thesis and determining which method outperforms the other.