FAST LATTICE GREEN'S FUNCTION METHODS FOR VISCOUS INCOMPRESSIBLE FLOWS ON UNBOUNDED DOMAINS

Thesis by
Sebastian Liska

In Partial Fulfillment of the Requirements for the
degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2016
Defended March 24$^{\text{th}}$, 2016

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my Ph.D. advisor Tim Colonius for the unwavering support and invaluable guidance throughout my graduate studies. It is thanks to his mentorship, patience, and knowledge that I have gained the perspective to approach fundamental scientific problems. Moreover, I expect the strong sense of perseverance that Tim has imbued in me will continue to help me overcome seemingly insurmountable difficulties and to give me the confidence to pursue increasingly challenging endeavors.

I would also like to thank my committee members, Guillaume Blanquart, Dan Meiron, and Anthony Leonard, as well as past and present members of my research group, including Andres Goza, Jeesson Choi, Hsieh-Chen Tsai, Benjamin Morley and Daniel Appelö, for helping me develop and strengthen the ideas ultimately used to accomplish my research goals. As a member of the Computational Flow Physics Groups, I have had the pleasure of being immersed in an intellectually stimulating environment filled with talented and insightful individuals that have contributed to personal and professional growth.

Last but not least, I would like to thank my mother and wife for the immense support, encouragement, and patience during the past seven years of graduate studies. I am particularly indebted to my wife, Gwen, for proofreading most of this thesis and related articles.

ABSTRACT

In this thesis, a collection of novel numerical techniques culminating in a fast, parallel method for the direct numerical simulation of incompressible viscous flows around surfaces immersed in unbounded fluid domains is presented. At the core of all these techniques is the use of the fundamental solutions, or lattice Green's functions, of *discrete* operators to solve inhomogeneous elliptic *difference* equations arising in the discretization of the three-dimensional incompressible Navier-Stokes equations on unbounded regular grids. In addition to automatically enforcing the natural free-space boundary conditions, these new lattice Green's function techniques facilitate the implementation of robust staggered-Cartesian-grid flow solvers with efficient nodal distributions and fast multipole methods. The provable conservation and stability properties of the appropriately combined discretization and solution techniques ensure robust numerical solutions. Numerical experiments on thin vortex rings, low-aspect-ratio flat plates, and spheres are used verify the accuracy, physical fidelity, and computational efficiency of the present formulations.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] S. Liska and T. Colonius. "A parallel fast multipole method for elliptic difference equations". *Journal of Computational Physics* 278 (2014), pp. 76–91. DOI: 10.1016/j.jcp.2014.07.048. ARXIV: 1402.6081 [physics.comp-ph].

[2] S. Liska and T. Colonius. "A fast lattice Green's function method for solving viscous incompressible flows on unbounded domains". *Accepted for publication in Journal of Computational Physics* (2015). ARXIV: 1601.00035 [physics.flu-dyn].

[3] S. Liska and T. Colonius. "A fast immersed boundary method for external incompressible viscous flows using lattice Green's functions". *Submitted to Journal of Computational Physics* (2016). ARXIV: 1604.01814 [physics.flu-dyn].

[4] A. Goza, S. Liska, B. Morley, and T. Colonius. "Accurate computation of surface stresses and forces with immersed boundary methods". *Revised and resubmitted to Journal of Computational Physics* (2015). ARXIV: 1603.02306 [physics.flu-dyn].

The author proposed the original idea, performed the work, analyzed and interpreted the results, and was the principal writer of the journal articles [1–3] that are included as chapters in this thesis. The author proposed the original idea, supervised the preliminary work, contributed to the analysis and interpretation of the results, and suggested revisions to the manuscript of the journal article [4]. The results reported in article [4] are used in parts of the discussions and examples included in article [3].

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Number *Page*

LIST OF TABLES

Number *Page*

Chapter 1

INTRODUCTION

The goal of computational fluid dynamics is to develop and use numerical techniques to understand and predict the mechanics of fluid flows. An inherent limitation on the wide-spread use of detailed and accurate numerical investigations of many fluid flows is the finite amount and associated cost of present day computational resources. Consequently, the development of accurate, robust, computationally efficient methods is of central importance to the field. This thesis is composed of three journal articles that detail the development of new computationally efficient techniques for numerically solving viscous incompressible flows on three-dimensional unbounded fluid domains.

Underlying all the numerical techniques discussed in this thesis are solution methods for partial *difference* equations resulting from the formal discretization of partial *differential* equations (PDEs) on unbounded regular grids. Though unbounded regular grids are often used in the analysis of numerical methods and for modeling discrete physical phenomena, they are rarely used in practice to solve discretized PDEs since the conventional grid-based approach of tracking values on all grid points renders unbounded grid methods impractical. On the other hand, there are several numerical methods, such as particle and vortex methods, that obtain practical solutions to PDEs on unbounded domains by numerically evaluating convolutions between the fundamental solution, or Green's function, of differential operators and compactly supported source terms. Motivated by the Green's function approach of these techniques, the methods discussed in subsequent chapters compute solutions to inhomogeneous difference equations by numerically evaluating discrete convolutions between the fundamental solution, or *lattice Green's function* (LGF), of discrete operators and compactly supported source terms. By blending some of the best features of grid-based methods (robustness, discrete conservation laws, etc.)

with those of particle-methods (efficient nodal distribution, fast free-space solvers, etc.), the present set of techniques offers an entirely new methodology for efficient incompressible flow simulations.

In Chapter 2, a fast solution method for elliptic constant-coefficient difference equations relevant to incompressible flows, e.g. discrete Poisson problems, on unbounded Cartesian grids is presented. This technique is developed as a kernel-independent, interpolation-based fast multipole method (FMM) that is accelerated by taking advantage of the regularity of the underlying grid and the efficiency of FFT-based discrete convolutions. Computational rates and parallel scaling for discrete (7-pt) free-space Poisson problems are shown to be comparable to those obtained for continuum free-space Poisson problems by other highly-optimized FMMs.

In Chapter 3, the LGF-FMM of Chapter 2 is used to develop a fast, robust incompressible flow solver. The incompressible Navier-Stokes equations are formally discretized on an unbounded staggered Cartesian grid using a second-order finite-volume scheme. An integrating factor (IF) technique is combined with a half-explicit Runge-Kutta (HERK) method in order to efficiently integrate the system of differential algebraic equations (of index 2) resulting from the spatial discretization of the momentum equations and the divergence-free constraint. A splitting-error-free projection method is used to cast the solution to the linear system of equations arising at each the stage of IF-HERK method as an equivalent discrete Poisson problem, which are in turn is computed using the LGF-FMM method. Solutions of unsteady flows are efficiently computed on small finite computational grids by combining a block-wise adaptive grid with a velocity refresh technique. The accuracy, physical fidelity, and computational rates of the LGF flow solver are demonstrated through numerical simulations of vortex rings at Reynolds numbers up to 20,000.

Lastly, in Chapter 4, the LGF flow solver of Chapter 3 is extended to include rigid immersed surfaces using an immersed boundary (IB) method. The method uses the discrete delta function approach of classical IB methods to regularize surface stresses

and the no-slip constraint. But, unlike classical IB methods, the present approach does not explicitly compute boundary forces from constitutive equations; instead the boundary forces are treated as Lagrange multipliers that are used to satisfy the no-slip constraint. This approach is shown to result in a system of semi-discrete equations that retains most of the structure of the semi-discrete equations of the flow solver of Chapter 2. Using appropriately modified IF-HERK schemes and an augmented (nested) projection technique, it is demonstrated that the IB-LGF method numerically solves practical flows with rigid surfaces at a computational rate that is typically one-and-a-half (at most two) times slower than the LGF flow solver. Numerical experiments on low-aspect-ratio flat plates and spheres at Reynolds numbers up to 3,700 are used to verify the accuracy and physical fidelity of the formulation.

Chapter 2

A PARALLEL FAST MULTIPOLE METHOD FOR ELLIPTIC DIFFERENCE EQUATIONS

Published in Journal of Computational Physics, December 2014

Chapter abstract

A new fast multipole formulation for solving elliptic difference equations on unbounded domains and its parallel implementation are presented. These difference equations can arise directly in the description of physical systems, e.g. crystal structures, or indirectly through the discretization of PDEs. In the analog to solving continuous inhomogeneous differential equations using Green's functions, the proposed method uses the fundamental solution of the discrete operator on an infinite grid, or lattice Green's function. Fast solutions $\mathcal{O}(N)$ are achieved by using a kernel-independent interpolation-based fast multipole method. Unlike other fast multipole algorithms, our approach exploits the regularity of the underlying Cartesian grid and the efficiency of FFTs to reduce the computation time. Our parallel implementation allows communications and computations to be overlapped and requires minimal global synchronization. The accuracy, efficiency, and parallel performance of the method are demonstrated through numerical experiments on the discrete 3D Poisson equation.

2.1 Introduction

Numerical simulations of physical phenomena often require fast solutions to linear, elliptic difference equations with constant coefficients on regular, unbounded domains. These difference equations naturally arise in the description of physical phenomena including random walks [5], crystal physics [6], and quantum mechanics [7]. Additionally, such difference equations can result from the discretization of PDEs on infinite regular grids or meshes [8–11]. Apart from the accuracy with which the underlying PDE is solved, an accurate solution of the difference equations themselves is relevant for compatible spatial discretization schemes that enforce discrete conservation laws [12, 13]. Examples of these techniques include finite-volume methods, mimetic schemes, covolume methods, and discrete calculus methods.

The present method considers difference equations formally defined on unbounded Cartesian grids. Solutions to the difference equations are obtained through the convolution of the fundamental solution of the discrete operator with the source terms of the difference equations. As a result, the formally infinite grid can be truncated to a finite computational grid by removing cells that contain negligible source strength. The ease with which this technique is able to adapt the computational domain makes it well-suited for applications involving the temporal evolution of irregular source distributions. For problems that are efficiently described by block-structured grids it is possible to adapt the computational domain by simply adding or removing blocks; an example of this technique applied to an incompressible flow is provided in Section 2.5.

The fundamental solution of discrete operators on regular grids, or lattices, are often referred to as lattice Green's functions (LGFs). Expressions for LGFs can be readily obtained in the form of Fourier integrals, but it is typically not possible to reduce the integral representations to expressions only involving a few elementary functions [14, 15]. The analytical treatment and the numerical evaluation of many LGFs is facilitated by the availability of asymptotic expansions [16–18]. Although LGFs have been extensively studied, they have rarely been used for solving large systems of elliptic difference equations (exceptions include 2D problems [10, 11, 19]). The present work extends the use of LGFs to large scale computations involving solutions to 3D elliptic difference equations.

Solving the system of difference equations using LGFs requires evaluating discrete convolutions of the form

$$u(\mathbf{x}_i) = [K * \mathbf{f}](\mathbf{x}_i) = \sum_{j=0}^{M-1} K(\mathbf{x}_i, \mathbf{y}_j) f(\mathbf{y}_j), \;\; i = 0, 1, \ldots, N-1, \tag{2.1}$$

where $K(\mathbf{x}_i, \mathbf{y}_j)$ is the kernel describing the influence of a source located at $\mathbf{y}_j$ with strength $f(\mathbf{y}_j)$ has on the field $u(\mathbf{x})$ at location $\mathbf{x}_i$. For the case of $M = N$, the straightforward approach to evaluate Eq. 2.1 requires $\mathcal{O}(N^2)$ operations. There are

several techniques for evaluating Eq. 2.1 in $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ operations. A few of these techniques are FMMs, FFT-based methods, particle-in-cell methods, particle-mesh methods, multigrid techniques, multilevel local-correction methods, and hierarchical matrix techniques. In the interest of brevity, a literature review of all the methods related to the fast evaluation of Eq. 2.1 is omitted; instead we focus our attention on FMMs.

The performance of FMMs relies on the existence of a compressed, or low-rank, representation of the far-field behavior of $K(\mathbf{x}, \mathbf{y})$ that can be used to evaluate Eq. 2.1 to a prescribed tolerance. Classical fast multipole methods [20, 21] require analytical expansions of the far-field behavior of kernels in order to derived low-rank approximations. Although classical FMMs can be developed for the asymptotic expansion of LGFs, alternative FMMs that are better suited for complicated kernel expressions have been developed. Kernel-independent FMMs [11, 22–26] do not require analytical expansions of the far-field; instead, for suitable kernels, these methods only require numerical evaluations of the kernel.

The present method is a kernel-independent interpolation-based FMM for non-oscillatory translation-invariant kernels [25, 27]. These FMMs achieve low-rank approximations of the kernel by projecting the kernel onto a finite basis of interpolation functions. Interpolation-based FMMs [25, 27] use Chebyshev interpolation and accelerate convolutions involving the compressed kernel using singular-value-decompositions (SVDs). In contrast, our method uses polynomial interpolation on equidistant nodes and accelerates convolutions involving the compressed kernel using FFTs. Intermediate regular grids and fast FFT-based convolutions have been used by other FMMs [23, 28–30], and have been shown to be particularly useful in accelerating the computations of 3D methods [23]. The use of intermediate regular grids in our method has the added advantage of simplifying the multilevel algorithm, since sources and evaluation points are defined on Cartesian grids at all levels of the multilevel scheme. The spatial regularity allows for the same fast convolution

techniques to be used in determining near-field and far-field contributions. In addition to the base algorithm, our method allows for pre-computations that further accelerate the solver.

The present FMM is similar to the recent 2D FMM [11] in that they both solve difference equations on unbounded domains. In contrast to our method, this method uses skeleton/proxy points and rank-revealing factorizations to obtain low-rank approximations of the kernel. Although we think it is possible to extend this method to 3D, we refrain from speculating on the performance of the algorithm since such extensions are not explored in current literature and their details are unclear to us.

Details regarding LGFs and their relation to solving difference equations on unbounded domains are presented in Section 2.2. This section also describes methods for performing fast convolutions based on kernel compression and FFT techniques, and presents a context in which these two techniques can be combined to yield an even faster convolution scheme. The resulting fast multipole algorithm and its parallel extension are then described in Section 2.3. Finally, serial and parallel numerical experiments are reported and analyzed in Section 2.4.

2.2 Lattice Green's functions and fast block-wise convolution techniques

2.2.1 Solving difference equations on infinite Cartesian grids

The method proposed in this paper is designed to solve inhomogeneous, linear, constant-coefficient difference equations on unbounded domains. As a representative problem, we consider in detail the difference equations resulting from the discretization of Poisson's equation in 3D. Consider the Poisson equation

$$[\Delta u](\mathbf{x}) = f(\mathbf{x}), \;\; supp(f) \in \Omega, \tag{2.2}$$

where $\mathbf{x} \in \mathbb{R}^3$, Ω is a bounded domain in $\mathbb{R}^3$, and $u(\mathbf{x})$ decays as $1/|\mathbf{x}|$ at infinity. Eq. 2.2 has the analytic solution

$$u(\mathbf{x}) = [G * f](\mathbf{x}) = \int_{\Omega} G(\mathbf{x} - \boldsymbol{\xi}) f(\boldsymbol{\xi})\, d\boldsymbol{\xi}, \tag{2.3}$$

where $G(\mathbf{x}) = -1/4\pi|\mathbf{x}|$ is the fundamental solution of the Laplace operator. Discretizing Eq. 2.2 on an infinite uniform Cartesian grid using a standard second-order finite-difference or finite-volume scheme produces a set of difference equations

$$[\mathsf{L}\mathsf{u}](\mathbf{n}) = \mathsf{f}(\mathbf{n}),\ \ supp(\mathsf{f}) \in \Omega_h, \tag{2.4}$$

where L is the standard 7-pt discrete Laplace operator, $\mathbf{n} \in \mathbb{Z}^3$, and Ω_h is a bounded domain in $\mathbb{Z}^3$. In practice, the constraint on $supp(\mathsf{f})$ can be relaxed by prescribing a finite tolerance and requiring that all non-negligible sources, i.e. sources with a magnitude greater than the prescribed tolerance, be located in a bounded region. The solution to Eq. 2.4 is given by

$$\mathsf{u}(\mathbf{n}) = [\mathsf{G} * \mathsf{f}](\mathbf{n}) = \sum_{\mathbf{m} \in \Omega_h} \mathsf{G}(\mathbf{n} - \mathbf{m}) \mathsf{f}(\mathbf{m}), \tag{2.5}$$

where $\mathsf{G}(\mathbf{n})$ is the fundamental solution of the discrete Laplace operator. An expression for $\mathsf{G}(\mathbf{n})$ in terms of Fourier integrals is provided by

$$\mathsf{G}(\mathbf{n}) = \frac{1}{8\pi^3} \int_{[-\pi,\pi]^3} \frac{\exp\left(-i\mathbf{n} \cdot \boldsymbol{\xi}\right)}{2\cos(\xi_1) + 2\cos(\xi_2) + 2\cos(\xi_3) - 6}\, d\boldsymbol{\xi}. \tag{2.6}$$

The expression in Eq. 2.6 is readily obtained by first using Discrete Fourier Transforms (DFTs) to diagonalize L. The diagonalized operator is then inverted and subsequently transformed back to the original space using inverse DFTs. Infinite sums in the resulting expression are converted to integrals using appropriate limiting procedures. Details regarding the construction of Eq. 2.6 and expressions for the fundamental solutions to other discrete operators are found in [5, 16, 18]. Additionally, Appendix 2.A provides an outline of the numerical procedures used to evaluate $\mathsf{G}(\mathbf{n})$ for small values of $|\mathbf{n}|$.

Although it is possible to compute $\mathsf{G}(\mathbf{n})$ by numerically evaluating Eq. 2.6, for large values of $|\mathbf{n}|$ it is more efficient to evaluate the LGF via its asymptotic expansion. Techniques for constructing asymptotic expansions of LGFs to arbitrary order are described in [16–18]. Let $A^q_{\mathsf{G}}(\mathbf{x})$ denote the q-term asymptotic expansion of $\mathsf{G}(\mathbf{n})$. For the 3D case, we define $A^q_{\mathsf{G}}(\mathbf{x})$ as the unique rational function such that

$$\mathsf{G}(\mathbf{n}) = A^q_{\mathsf{G}}(\mathbf{n}) + \mathcal{O}\left(|\mathbf{n}|^{-2q-1}\right) \tag{2.7}$$

as $|\mathbf{n}| \to \infty$. For $q = 2$, this asymptotic expansion is given by

$$A^2_{\mathsf{G}}(\mathbf{x}) = -\frac{1}{4\pi|\mathbf{x}|} - \frac{x_1^4 + x_2^4 + x_3^4 - 3x_1^2x_2^2 - 3x_1^2x_3^2 - 3x_2^2x_3^2}{16\pi|\mathbf{x}|^7}, \tag{2.8}$$

where $\mathbf{x} = (x_1, x_2, x_3)$. As expected, the first term in Eq. 2.8 corresponds to the fundamental solution of the Laplace operator. We note that, as is the case for many asymptotic expansions, it is not always possible to increase the accuracy of the expression for a fixed argument by increasing the number of terms.[1]

Despite the fact that $G(\mathbf{x})$ and $\mathsf{G}(\mathbf{n})$ share the same asymptotic behavior, there are significant differences in their behavior near the origin. Unlike $G(\mathbf{x})$, which is singular at the origin, $\mathsf{G}(\mathbf{n})$ remains finite for all values of $\mathbf{n}$. $G(\mathbf{x})$ is scale-invariant, i.e. there exists a k such that $G(\alpha\mathbf{x}) = \alpha^k G(\mathbf{x})$, whereas $\mathsf{G}(\mathbf{n})$ is not scale-invariant. Furthermore, $G(\mathbf{x})$ is spherically symmetric about the origin, as opposed to $\mathsf{G}(\mathbf{n})$, which has reflectional symmetry about the principal axes and is invariant under index permutations.

In addition to providing expressions for the fast evaluation of LGFs, asymptotic expansions of LGFs allow for the sum given in Eq. 2.5 to be decomposed into three parts:

$$\mathsf{u}(\mathbf{n}) = \mathsf{u}^{\text{direct}}(\mathbf{n}) + u^{\text{asympt},q}(\mathbf{n}) + \epsilon(\mathbf{n}), \tag{2.9}$$

[1] For example, for $|\mathbf{n}| = 10$ the minimum value of $|A^q_{\mathsf{G}}(\mathbf{n}) - \mathsf{G}(\mathbf{n})|/\mathsf{G}(\mathbf{n})$ is approximately 10^{-7} and is achieved by $n = 6$. Increasing or decreasing n, i.e. the number of terms in the asymptotic expansion, increases the relative difference between the $A^q_{\mathsf{G}}(\mathbf{n})$ and $\mathsf{G}(\mathbf{n})$ for $|\mathbf{n}| = 10$.

where

$$\mathsf{u}^{\text{direct}}(\mathbf{n}) = \sum_{\mathbf{m}\in\Omega_h^{\text{direct}}(\mathbf{n})} \mathsf{G}(\mathbf{n}-\mathbf{m})\mathsf{f}(\mathbf{m})\,, \tag{2.10}$$

$$u^{\text{asympt},q}(\mathbf{n}) = \sum_{\mathbf{m}\in\Omega_h\setminus\Omega_h^{\text{direct}}(\mathbf{n})} A_{\mathsf{G}}^{q}(\mathbf{n}-\mathbf{m})\mathsf{f}(\mathbf{m})\,, \tag{2.11}$$

and $\epsilon(\mathbf{n})$ is the error due to approximating $\mathsf{G}(\mathbf{n})$ with $A_{\mathsf{G}}^{q}(\mathbf{n})$ over the region $\Omega_h \setminus \Omega_h^{\text{direct}}$. The region $\Omega_h^{\text{direct}}(\mathbf{n})$ is a subset of Ω_h for which the LGF is evaluated directly, i.e. via numerical evaluation of Eq. 2.6, as opposed to being evaluated via its asymptotic expansion. Typically, the region $\Omega_h^{\text{direct}}(\mathbf{n})$ is defined by a small cubic box centered at the grid point $\mathbf{n}$.[2] The first term of Eq. 2.9, $\mathsf{u}^{\text{direct}}(\mathbf{n})$, is a grid function evaluated at the grid point $\mathbf{n}$, whereas the second term, $u^{\text{asympt},q}(\mathbf{n})$, is a continuous function evaluated at the location of the grid point $\mathbf{n}$. As will be discussed in subsequent sections, this decomposition allows for $u^{\text{asympt},q}(\mathbf{n})$ to be evaluated using fast techniques developed for continuous kernels.

2.2.2 Fast convolutions on regular grids via FFTs

Although discrete convolutions via FFTs is a well-known technique, a brief description is provided in order to introduce procedures and notation subsequently referenced in different steps of the overall algorithm. Consider the one-dimensional convolution given by

$$u(x_i) = \sum_{j=0}^{M-1} K(x_i, y_j) f(y_j)\,, i = 0, 1, \ldots, N-1. \tag{2.12}$$

where $x_i = x_0 + ih$ for $i = 0, 1, \ldots, N-1$, and $y_j = y_0 + jh$ for $j = 0, 1, \ldots, M-1$. If the kernel $K(x, y)$ is translation invariant, i.e. $K(x, y) = K(x-y)$, then Eq. 2.12

[2] For the case of the 3D discrete Laplace operator, choosing Ω_h^{direct} to be a cubic box with side lengths of 14, 41, and 134 grid points is sufficient to achieve relative errors less than 10^{-5}, 10^{-10}, and 10^{-15}, respectively, using the five term asymptotic expansion. The size of Ω_h^{direct} can be reduced by including more terms in the asymptotic expansion, for example, relative errors less than 10^{-15} are achieved by using the thirteen term asymptotic expansion and a box with 38 grid points on each side.

can be expressed as the discrete convolution between two vectors,

$$u_i = \sum_{j=0}^{M-1} k_{N-1+j-i} f_j \,, i = 0, 1, \ldots, N-1 \,, \tag{2.13}$$

where $u_i = u(x_i)$, $f_j = f(x_j)$, and $k_{N-1+j-i} = K(x_j - y_i)$. Discrete linear convolutions of this form can be cast into circular convolutions by appropriate padding of the vectors u and f. Performing these convolutions using DFTs leads to the fast FFT-based convolution technique given by

1. Pad sequence with zeros: append $N-1$ zeros to vector f.

$$\bar{f}_i = [\mathrm{Pad}(f)]_i = \begin{cases} f_i & i = 0, 1, \ldots, N-1 \\ 0 & i = N, N+1, \ldots, N+M-2 \end{cases} \tag{2.14}$$

2. Forward DFT: compute the DFT of sequences $\bar{f}$ and k via FFTs.

$$\hat{f} = \mathrm{FFT}(\bar{f}),\ \hat{k} = \mathrm{FFT}(k) \tag{2.15}$$

3. Convolution of DFTs: multiply complex coefficients of $\hat{f}$ and $\hat{k}$.

$$\hat{u}_i = [\mathrm{Prod}(g, k)]_i = \hat{f}_i \hat{k}_i \,, i = 0, 1, \ldots, N+M-2 \tag{2.16}$$

4. Backward DFT: compute the inverse DFT of sequence $\hat{u}$ via FFT.

$$\bar{u} = \mathrm{FFT}^{-1}(\hat{u}) \tag{2.17}$$

5. Truncate sequence: remove the first $M-1$ entries of $\bar{u}$ to obtain u.

$$u_i = [\mathrm{Trunc}(\bar{u})]_i = \bar{u}_{M+i} \,, i = 0, 1, \ldots, N-1 \tag{2.18}$$

This technique requires $\mathcal{O}\left((N+M)\log(N+M)\right)$ operations and is readily generalized to higher dimensions for the case of tensor-product grids by recursively applying the 1D version to each directions.

2.2.3 Adaptive block-structured grid

Fast convolutions via FFTs discussed in Section 2.2.2 can be used to accelerate the evaluation of Eq. 2.5. In order to use this technique, the support of f needs to be padded with zeros to form a box. Similarly, the region where u is evaluated needs to be extended to also form a box. For cases where the domain defined by the support of the source terms is not a box, the cost of the additional computational elements can outweigh the reduced operation count per grid point of the FFT-based convolution technique.

Computational domains defined by the union of blocks can, however, be used to avoid excessive padding and still retain sufficient regularity to benefit from the fast FFT-based convolution technique. Our formulation partitions the infinite grid into blocks defined on a logically Cartesian grid. Blocks can potentially have a different number of grid points in each direction, but all blocks are required to have the same dimensions. An active source block denotes a block containing non-zero sources. Similarly, an active evaluation block denotes a block containing grid points on which the induced field is evaluated. The union of active source (evaluation) blocks is referred to as the active source (evaluation) grid. We emphasize that grid adaptivity is achieved through the selective choice of active blocks in order to define efficient computational domains, the present method does not consider problems with multiple spatial resolutions.

Let B_s and B_e denote the sets of active source and evaluation blocks, respectively. The convolution given in Eq. 2.5 can be evaluated by

$$u^P = \sum_{Q \in B_s} \text{conv}(k^{Q-P}, f^Q), \ \forall P \in B_e, \tag{2.19}$$

where u^P and f^P denote vectors containing the values of $\mathsf{u}(\mathbf{n})$ and $\mathsf{f}(\mathbf{n})$, respectively, evaluated on the grid points belonging to block P. Similarly, k^{Q-P} denotes the vector containing the unique values of $\mathsf{G}(\mathbf{m}-\mathbf{n})$, as described in Section 2.2.2, for values of $\mathbf{n}$ and $\mathbf{m}$ corresponding to the indices of grid points belonging to block Q

and P, respectively. If blocks P and Q are sufficiently well-separated then $A^q_{\mathsf{G}}(\mathbf{m}-\mathbf{n})$ is used instead of $\mathsf{G}(\mathbf{m}-\mathbf{n})$ for constructing k^{Q-P}. The operator $\text{conv}(k^{Q-P}, f^Q)$ denotes the generalization of Eq. 2.12 to arbitrary dimensions. Details regarding the construction of vectors u^P, f^P, and k^{Q-P} are omitted, since it immediately follows the discussion regarding the tensor-product grid generalization of Eq. 2.12.

Computing each instance of $\text{conv}(k^{Q-P}, f^Q)$ in Eq. 2.19 using the fast FFT-based convolution technique leads to a scheme that evaluates Eq. 2.5, for the case of $B_s = B_e$, in $\mathcal{O}(N_B^2 N_b \log(N_b))$ operations, where N_B is the number of blocks belonging to B_s, and N_b is the number of grid points belonging to each block. The operation count can be further reduced to $\mathcal{O}(N_B N_b \log(N_b) + N_B^2 N_b)$ if the DFT of the kernel blocks, $\hat{k}^{Q-P}$, are pre-computed, and if the DFT of source and evaluation blocks are preformed as pre-processing and post-processing step, respectively. Details regarding pre-computations, and pre- and post-processing steps are discussed in subsequent sections.

2.2.4 Fast convolutions via interpolation-based kernel compression

Interpolation-based FMMs obtain a low-rank representation of the kernel, $K(\mathbf{x}, \mathbf{y})$, by projecting it onto a finite basis of interpolation functions. Consider a function $f(\mathbf{x})$ sampled at n points, $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$. An approximation for $f(\mathbf{x})$ is given by

$$\tilde{f}^n(\mathbf{x}) = \sum_{i=0}^{n-1} \phi_i(\mathbf{x}) f(\mathbf{x}_i), \tag{2.20}$$

where $\phi_i(\mathbf{x})$ is a interpolation function associated with the interpolation node $\mathbf{x}_i$. An approximation for $K(\mathbf{x}, \mathbf{y})$ is obtained by recursively applying Eq. 2.20 to each argument of $K(\mathbf{x}, \mathbf{y})$,

$$\tilde{K}^n(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \psi_i(\mathbf{x}) K(\mathbf{x}_i, \mathbf{y}_j) \phi_j(\mathbf{y}), \tag{2.21}$$

where $\{\phi\} = \{\phi_0, \phi_1, \ldots, \phi_{n-1}\}$ and $\{\psi\} = \{\psi_0, \psi_1, \ldots, \psi_{m-1}\}$ are potentially distinct bases of interpolation functions. In order to make the kernel-compression

technique symmetric, only schemes with $\{\phi\} = \{\psi\}$ are considered. Directly applying this kernel compression technique to discrete convolutions of the form of Eq. 2.1 leads to an approximation of $u(\mathbf{x}_i)$ given by

$$u(\mathbf{x}_i) \approx \sum_{j=0}^{M-1} \sum_{p=0}^{n-1} \sum_{q=0}^{n-1} \phi_p(\mathbf{x}_i) K(\mathbf{x}_p, \mathbf{y}_q) \phi_q(\mathbf{y}_j) f(\mathbf{y}_j), \; i = 0, 1, \ldots, N-1. \tag{2.22}$$

For cases involving multiple sets of either evaluation points, $\mathbf{x}_i$, or source points, $\mathbf{y}_j$, it is advantageous to decompose the evaluation of Eq. 2.22 into three steps:

1. Regularization: compute effective source terms using the adjoint of the interpolation procedure.

$$\tilde{f}(\mathbf{y}_q) = \sum_{j=0}^{M-1} \phi_q(\mathbf{y}_j) f(\mathbf{y}_j), \; q = 0, 1, \ldots, n-1 \tag{2.23}$$

2. Convolution: compute the field induced by effective source terms on interpolation nodes.

$$\tilde{u}(\mathbf{x}_p) = \sum_{q=0}^{n-1} K(\mathbf{x}_p, \mathbf{y}_q) \tilde{f}(\mathbf{y}_q), \; p = 0, 1, \ldots, n-1 \tag{2.24}$$

3. Interpolation: compute the field at evaluation points using the interpolation procedure.

$$u(\mathbf{y}_i) = \sum_{p=0}^{n-1} \phi_p(\mathbf{x}_i) \tilde{u}(\mathbf{x}_j), \; i = 0, 1, \ldots, N-1 \tag{2.25}$$

If the values of $\phi_q(\mathbf{x}_i)$ and $\phi_q(\mathbf{y}_j)$ are known, the number of operations required by this procedure, for the case of $M = N$, is $\mathcal{O}(2nN + n^2)$. For the case of $n \ll N$ this procedures represents a significant reduction in the number of operations compared to straightforward method of evaluating Eq. 2.1.

2.2.5 Fast convolution on regular grids using polynomial interpolation and FFTs

The fast convolution techniques presented in Sections 2.2.2 and 2.2.4 can be combined to yield a faster method for evaluating the block-wise convolutions involved in Eq. 2.19. This technique follows from the observation that Eq. 2.24 can be evaluated

using FFTs if the kernel is translation-invariant and the interpolation nodes on which the kernel is evaluated are restricted to be on a regular grid. The first requirement is assumed since the kernels corresponding to the fundamental solutions of the linear, constant-coefficient elliptic difference equations are translation-invariant. The second condition is achieved by using an interpolation scheme based on equidistant interpolation nodes on tensor-product grids. Although many interpolation schemes satisfy the latter condition, only polynomial interpolation schemes on tensor-product grids are presently considered since they are fast, simple to implement, and their behavior is well-understood.[3]

Polynomial interpolation on tensor-product grids is performed by recursively applying 1D polynomial interpolation along each direction. This generalization has the advantage of maintaining the number of operation per grid point independent of dimension, and allows the behavior of the interpolation process to be readily generalized from its 1D version.

In the absence of rounding errors, 1D polynomial interpolants converge geometrically if the function being interpolated is analytic in a region on the complex plane near the interpolation interval. The size and shape of this convergence region depends on the choice of interpolation nodes [31]. The kernels being considered correspond to the asymptotic expansion of the LGFs that are only discontinuous at the origin. Although convergence conditions should be verified for each kernel, the requirement that source and evaluation blocks be sufficiently well-separated to accurately evaluate the LGF using its asymptotic expansion is often sufficient to guarantee convergence.

Unlike Chebyshev interpolation, polynomial interpolation on equidistant nodes is

[3] Alternative interpolation procedures, with the exception of Fourier interpolation on non-periodic domains, have not been explored. Although Fourier interpolation is particularly appropriate given FFTs are used in our method to accelerate local computations, preliminary results have shown that this procedure is less efficient (in terms of points per unit accuracy) than the procedure described in this section.

ill-conditioned. Ill-conditioning can cause rounding errors due to finite numeric precision can be amplified. The Lebesgue constant, $\Lambda_n(X)$, for a set of n interpolation points $X = x_0, x_1, \ldots, x_n - 1$, can be used to bound the growth of perturbations in the data [32],

$$\max_{x \in I} |p(x) - \tilde{p}(x)| \leq \Lambda_n(X) \max_{0 \leq i \leq n-1} |f(x_i) - \tilde{f}(x_i)|, \tag{2.26}$$

where I is the interpolation interval, $p(x)$ and $\tilde{p}(x)$ are the polynomials interpolants resulting from the nodal values $f(x_i)$ and $\tilde{f}(x_i)$, respectively. The Lebesgue constant is given by

$$\Lambda(X) = \max_{x \in I} \sum_{i=0}^{n-1} |\phi_i(x)|, \tag{2.27}$$

where $\phi_i(x)$ is the Lagrange characteristic polynomial associate with x_i. Eq. 2.26 can be extended to polynomial interpolation on tensor product grids, with equal number of points and spacing in each direction, by replacing $\Lambda(X)$ with $(\Lambda_n(X))^d$, where d is the dimension of the problem.

In the limit of very large n, the Lebesgue constant of a set of equally spaced nodes is known to grow exponentially [32]. In order to avoid very large Lebesgue constants, the present scheme restricts the number of nodes used for polynomial interpolation to be at most n_{max}. If n_{max} nodes are insufficient to achieve a desired interpolation error, additional nodes are added to the interval, but only the closest n_{max} nodes to the evaluation point are used for interpolation. Thus, this hybrid scheme performs both p- and h-refinement to increase the accuracy of the interpolations procedure. As a result, geometric convergence rates are expected for $n \leq n_{max}$, and polynomial convergence rates of order $n_{max} - 1$ are expected for $n > n_{max}$. The values of n and n_{max} required to interpolate a function $f(x)$ over an interval I with an interpolation error less than ϵ are obtained in two steps:

1. Find the largest n_{max} such that $\Lambda_{n_{max}}(X)\epsilon_p$ is less than ϵ, where ϵ_p is the precision of the floating-point scheme.

2. Progressively increase the number of n until the difference between $f(x)$ and it approximation are less than ϵ.

The same procedure can be used in higher dimensions by having n and n_{max} correspond to the number of interpolation points along each direction, and replacing $\Lambda_{n_{max}}(X)$ with $(\Lambda_{n_{max}}(X))^d$. For example, approximating an analytic function in 3D using double-precision arithmetic to relative tolerance of $\epsilon = 1.25 \times 10^{-12}$ requires $n_{max} \leq 10$.

We omit a step-wise description of the combined fast algorithm for block-wise convolutions, since it readily follows from the discussion. Instead, we introduce the notation $f^P = \text{Interp}(f^Q)$ and $f^P = \text{Reg}(f^Q)$ to denote the interpolation and regularization (adjoint of interpolation) operations, respectively, Instead, we introduce use the notation $f^P = \text{Interp}(f^Q)$ and $f^P = \text{Reg}(f^Q)$, respectively, to denote the interpolation and regularization (adjoint of interpolation) of f from block/interval Q to block/interval P.

2.3 The Fast Lattice Green's Function method

2.3.1 Basic algorithm

Thus far we have discussed methods for accelerating the evaluation of Eq. 2.19 by performing fast block-wise convolutions involving interpolation-based kernel compression and/or FFT techniques. Asymptotically these schemes require $\mathcal{O}(N^2)$ operations, though the constant in front of the N^2 term can be significantly smaller compared to that of the straightforward method. For kernels that decay or exhibit progressively smoother behavior away from the origin, e.g. the fundamental solution of the discrete Laplace operator, it is possible to combine the fast block-wise convolution techniques discussed in Sections 2.2 with the multilevel scheme of the original FMM [20]. To facilitate the discussion, we will assume that the active evaluation grid is the same as the active source grid.

Our multilevel scheme follows a tree (octree in 3D) structure similar to that described in [20]. Tree nodes at all levels are said to correspond to intervals.[4] Each interval is defined by the tensor-product grid associated with the nodes of the interpolation scheme. The tree is constructed by first creating one tree leaf, i.e. tree node with no children, for each active grid block. The intervals of tree leaves are defined such that they occupy the same spatial region as their associated grid blocks. After defining all tree leaves, siblings are recursively merged to generate the multilevel structure. We use the convention that all tree leaves are located at level 1 and that the root of the tree is located at level L.

The set of intervals at level ℓ is denoted by B^ℓ, and N_B^ℓ denotes the size of B^ℓ. In order to facilitate the discussion, intervals at level ℓ are chosen to contain n_b^ℓ nodes in each directions. The total number of points in each interval at level ℓ is given by $N_b^\ell = (n_b^\ell)^d$, where d is the dimension of the problem. The set of blocks defining the underlying active grid is denoted by B^0. N_B^0, n_b^0, and N_b^0 have definitions analogous to N_B^ℓ, n_b^ℓ, and N_b^ℓ, respectively. We note that level zero, $\ell = 0$, is not part of the tree structure, but the slight abuse of notation facilitates the description of the algorithm.

By construction, all intervals are Cartesian grids. As a result the union of intervals belonging to the same level defines an analogous grid to that of the underlying adaptive block-structured grid. Therefore, the techniques for fast block-wise convolutions discussed in Section 2.2 are readily generalized to all levels of the tree structure.

Our overall algorithm for solving systems of difference equations of the form given by Eq. 2.4 on adaptive block-structured grids is referred to as the Fast Lattice Green's Function (FLGF) method. The FLGF method is described in the following steps:

[4] In the context of the hierarchical algorithm and structure, the term "interval" is equivalent to term "box" used in [20]. We reserve the term "box" for geometric descriptions, and do not associate any specific structure or information with the term.

0. *Pre-computation*: compute and store all unique $\hat{k}^{P-Q}$ used in Step 2.

$$\hat{k}^P = \mathrm{FFT}(k^{P-Q}) \tag{2.28}$$

1. *Upwards Pass*: $\forall P \in B^\ell$, for $\ell = 0, 1, \ldots, L$

   a) Regularize: compute effective source terms at interpolation nodes

$$\tilde{f}^P = \sum_{Q \in \mathrm{RegSupp}(P)} \mathrm{Reg}(\tilde{f}^Q), \tag{2.29}$$

   b) Padded forward DFT: prepare vectors for DFT convolutions

$$\hat{f}^P = \mathrm{FFT}(\mathrm{Pad}(\tilde{f}^P)), \tag{2.30}$$

2. *Level Interactions*: $\forall P \in B^\ell$, for $\ell = 0, 1, \ldots, L$

$$\hat{u}^P = \sum_{Q \in \mathrm{InflList}(P)} \mathrm{Prod}(\hat{f}^Q, \hat{k}^{P-Q}), \tag{2.31}$$

3. *Downwards Pass*: $\forall P \in B^\ell$, for $\ell = L, L-1, \ldots, 0$

   a) Truncated backwards DFT: extract relevant data from DFT convolution

$$\tilde{v}^P = \mathrm{Trunc}(\mathrm{FFT}^{-1}(\hat{u}^P)) \tag{2.32}$$

   b) Interpolate: compute and aggregate the induced field at interpolation nodes

$$\tilde{u}^P = \tilde{v}^P + \mathrm{Interp}(\tilde{u}^{\mathrm{IntrpSupp}(P)}) \tag{2.33}$$

We note that the operations performed in Steps 1, 2, and 3 are commonly referred to as the *multipole-to-multipole*, *multipole-to-local*, and *local-to-local* operations, respectively, in the FMM literature. The lists of blocks/intervals used by the algorithm

are given by

$$\text{RegSupp}(P) = \begin{cases} \text{block } P & \text{if } P \in B^0 \\ \text{block} \sim \text{interval } P & \text{if } P \in B^1 \\ \text{children of interval } P & \text{if } P \in B^\ell,\ 2 \leq \ell \leq L \end{cases} \tag{2.34}$$

$$\text{IntrpSupp}(P) = \begin{cases} \text{interval} \sim \text{block } P & \text{if } P \in B^0 \\ \text{parent of interval } P & \text{if } P \in B^\ell,\ 1 \leq \ell < L \\ \emptyset & \text{if } P \in B^L \end{cases} \tag{2.35}$$

$$\text{InflList}(P) = \begin{cases} \text{near-neighbors of block } P & \text{if } P \in B^0 \\ \text{interaction list of interval } P & \text{if } P \in B^\ell,\ 1 \leq \ell \leq L \end{cases} \tag{2.36}$$

where the symbol $\sim$ is taken here to mean *associated with*. Children, parents, near-neighbors, and interaction lists follow the same definitions those of the original FMM [20]. Based on these definitions, we note that Eq. 2.29 reduces to $\tilde{f}^P = f^P$ for $P \in B^0$, and Eq. 2.33 reduces to $\tilde{u}^P = \tilde{v}^P$ for $P \in B^L$.

2.3.2 Algorithmic complexity

The overall complexity of our algorithm is $\mathcal{O}(N)$, as is the case for the original FMM. For simplicity, the discussion concerning the cost of each step is limited to the 3D version of the algorithm. Details regarding to the cost of each block/interval are presented in Table 2.1. The factor of 8 in front of C^ℓ_{Interp} for the *Upwards Pass* ($\ell > 1$) is due to the fact that each interval has eight children. The constants, 27 and 189, associated with the cost of *Level Interactions* correspond to the number of near-neighbors and the number of members of each interaction list, respectively.

The specific values and a brief discussion of the constants presented in Table 2.1 are provided:

1. $C^\ell_{\text{EvalKernel}}$: Cost of kernel evaluation performed in Eq. 2.28. Constructing the vector k^{Q-P}, where Q and P are blocks/intervals at level ℓ, requires $8N_b^\ell$

Table 2.1: Operation counts per interval/block for each step of the FLGF method.

	cost	order
Pre-computations	$C^{\ell}_{\text{EvalKernel}} + C^{\ell}_{\text{PadFFT}}$	$N_b^{\ell} \log N_b^{\ell}$
Upwards pass ($\ell = 0$)	C^{0}_{PadFFT}	$N_b^{0} \log N_b^{0}$
Upwards pass ($\ell = 1$)	$C^{\ell}_{\text{Interp}} + C^{\ell}_{\text{PadFFT}}$	$N_b^{\ell} \log N_b^{\ell}$
Upwards pass ($\ell > 1$)	$8C^{\ell}_{\text{Interp}} + C^{\ell}_{\text{PadFFT}}$	$N_b^{\ell} \log N_b^{\ell}$
Level interactions ($\ell = 0$)	$27C^{\ell}_{\text{Prod}}$	$27N_b^{0}$
Level interactions ($\ell > 0$)	$189C^{\ell}_{\text{Prod}}$	$189N_b^{\ell}$
Downwards pass	$C^{\ell}_{\text{Interp}} + C^{\ell}_{\text{PadFFT}}$	$N_b^{\ell} \log N_b^{\ell}$

kernel evaluations. For small values of $|\mathbf{n} - \mathbf{m}|$ a look-up table is used to evaluate $\mathsf{G}(\mathbf{n} - \mathbf{m})$; otherwise the kernel is evaluated using $A^{n}_{\mathsf{G}}(\mathbf{n} - \mathbf{m})$.

2. C^{ℓ}_{Interp}: Cost of polynomial interpolation performed in Eq. 2.33. The coefficient mapping interpolation nodes to evaluation nodes are precomputed (only needed for 1D interpolation); therefore computing the values of a single block/interval at level ℓ requires $\min(n_b^{\ell+1}, n_{\max})N_b^{\ell}$ operations (1 real addition and 1 real multiplication per operation), where $n_{\max}$ is described in Section 2.2.5. In 3D, $n_{\max}$ is typically set to be no greater than 10. C^{ℓ}_{Interp} also describes the cost of performing the regularization, adjoint of interpolation, operation involved in each term of the sum in Eq. 2.29.

3. C^{ℓ}_{PadFFT}: Cost of performing a 3D FFT (real-to-complex) or inverse FFT (complex-to-real) on the padded vectors present in Eq. 2.28, 2.30, and 2.32. The operation count (total number of real additions and multiplications) for each FFT performed using the FFTW library is approximately $2(8N_b^{\ell}) \times \log_2(8N_b^{\ell})$ [33], where $8N_b^{\ell}$ is the size of the padded vectors. Since all FFTs are real-to-complex or complex-to-real approximately half of the coefficients are redundant and neither need be stored nor operated on.

4. C^{ℓ}_{Prod}: Cost of performing DFT convolutions, i.e. multiplication of complex coefficients, in Eq. 2.31. If blocks/intervals Q and P belong to level ℓ, then performing $\text{Prod}(\hat{f}^{P}, \hat{k}^{P-Q})$ requires approximately $4N_b^{\ell}$ operations (1 com-

plex multiplication per operation), where $4N_b^\ell$ is the number of non-redundant DFT coefficients per block/interval.

The cost per level of each operation, except for the *Pre-computation* step, can be obtained by multiplying the values of each row by the N_B^ℓ. In regards to the *Pre-computation* step, the cost per level for $\ell = 0$ and $\ell > 0$ is obtained by multiplying the cost per block/interval by 27 and 317, respectively, which correspond to the number of unique $\hat{k}^{P-Q}$ vectors that are used at each level. If the kernel shares the same symmetry as the LGF of the discrete Laplace operator, the number of unique $\hat{k}^{P-Q}$ vectors per level is reduced to 4 and 36 for $\ell = 0$ and $\ell > 0$, respectively. If symmetry is used to reduce number of pre-computed $\hat{k}^{P-Q}$ vectors, then C_{Prod}^ℓ is roughly doubled since a twiddle factor needs to be applied to the DFT coefficients for cases involving reflections.

2.3.3 Parallel implementation

A brief overview of our MPI-based algorithm is included to demonstrate that the present method allows for a simple parallel implementation suitable for practical large-scale scientific computing. In the present implementation the tree structure and load balancing estimates are redundantly computed (in serial) by all MPI-processes.[5] As a result, all the information necessary to evaluate RegSupp, IntrpSupp, and InflList for any block/interval is known by all processes. The tree structure is constructed following the bottom-up approach discussed in Section 2.3.1. Prior to partitioning the problem, the load balancing scheme first assigns a weight to every block and interval based on an estimate of its runtime cost. Next, parent tree nodes are recursively grouped with one of its child tree nodes, and tree leaves are grouped with their associated grid blocks. The set of groups is then partitioned into clusters

[5] Equivalently, the tree structure and load balancing estimates could be computed by a single MPI-process and then scattered to all processes, but this approach would result in additional communication costs.

in such a way that the weight of each cluster (aggregate weight of all interval/blocks belonging to all groups in the cluster) is roughly the same. Each cluster is then assigned to an MPI process. Given each group contains only one block, a Morton or Z-order curve [34] is used to preserve data locality during partitioning.

The parallel algorithm closely resembles the serial algorithm, since each process executes steps analogous to the *Upwards Pass*, *Level Interactions*, and *Downwards Pass*. Non-blocking routines are used for all communications, allowing for computations to be overlapped with communications. Furthermore, our algorithms use these routines to avoid any global synchronization within each steps and between steps.

The parallel execution of each step follows a similar event-driven paradigm, where processes perform particular "work units" based on the information that has been received or is locally available, and send information to other processes as soon as a set of "work units" have been completed. Send and receive buffers are used to avoid excessive memory requirements. The algorithm gives priority to "work units" yielding results that are sent to other processes. The time spent waiting to either receive information or to clear send buffers is used to perform local "work units", i.e. operations that only require data and yield results pertinent to the same process. During the *Upward* and *Downward Pass* a non-blocking send is posted after each interval-/block-wise regulation and interpolation operation is completed. In contrast, during *Level Interactions* step the influence of all intervals/blocks belonging to a process on all intervals/blocks belonging another process is aggregated and packaged before sending; thus each process sends at most one message to every other process during this step. For convenience, pseudo-codes for the communication patterns described in this section are given in Appendix 2.B.

2.4 Numerical results

In this section we present numerical results that demonstrate the accuracy, computational cost, and parallel performance of the FLGF method. The results reported are for the discrete Poisson equation in 3D. We clarify that the solution error is measured with respect to the exact solution of the *discrete* Poisson problem, as opposed to the exact solution of the *continuous* Poisson problem.

As in Section 2.3, the active evaluation grid is defined to be equal to the active source grid. For all test cases, blocks/intervals belonging to level ℓ are chosen to contain n_b^ℓ points in each directions. In the interest of brevity, we only consider schemes where the number of interpolation nodes per interval is the same for all levels, $n_b^\ell = n_I$ for $\ell = 1, 2, \ldots, L$, and require that $n_I = n_b^0 + 1$.[6] These considerations reduce the parameter space of possible schemes to cases where the spacing between nodes of parent intervals is twice that of child intervals. Furthermore, there is effectively no regularization/interpolation between level 0 and level 1, since the nodes on level 1 coincide with the underlying grid points. Following the discussion of Section 2.2.5 we set the maximum number of nodes used for interpolating a value at single point, $n_{\max}$, to be 10 for all test cases.

We note that the previous restrictions are only imposed for the purpose of a concise exposition. The non-scale-invariant behavior of most LGFs suggests that, in general, non-trivial performance gains can be achieved by tuning n_b^ℓ for each level. Given that the LGF of the discrete Laplace operator is approximately scale-invariant away from the origin, possible performance gains achieved by varying n_b^ℓ are not considered in the present discussion.

The present MPI-based implementation is written in Fortran and makes use of the

[6] Our implementation requires that intervals belonging to $\ell > 0$ have a one grid point overlap with their neighboring intervals along (n_1, n_2, n_3) directions, where $n_i = \{0, 1\}$ and at least one n_i is non-zero.

FFTW3 [33] library to compute FFTs. Numerical experiments are performed using our local computing facility which consists of 60 compute nodes connected by a QDR InfiniBand network. Each node contains 2 Intel Xeon X5650 processors (6-core, 12MB cache, 2.66GHz clock speed) and 48GB of RAM.

2.4.1 Error

The accuracy of the proposed methodology is investigated on cubic active grids containing different numbers of active grid points and partitioned into blocks of different sizes. A procedure based on random manufactured solutions is used to determine the error of each test case. In this procedure a solution, $\mathsf{u}_{\text{rand}}(\mathbf{n})$, is manufactured by assigning a random value between -1 and $+1$ to each grid points, except for grid points on the boundary of the active grid which are set to zero. The source distribution, $\mathsf{f}(\mathbf{n})$, which serves as the input for each test case, is computed by taking the discrete Laplacian of the prescribed solution, i.e. $\mathsf{f}(\mathbf{n}) = [\mathsf{L}\mathsf{u}_{\text{rand}}](\mathbf{n})$. The normalized error for each test case is computed by $\epsilon_p = ||\mathsf{u} - \mathsf{u}_{\text{rand}}||_p / ||\mathsf{u}_p||_p$, where $\mathsf{u}(\mathbf{n})$ is determined by solving $[\mathsf{L}\mathsf{u}_p](\mathbf{n}) = \mathsf{f}(\mathbf{n})$, and $||\mathsf{u}||_p$ is the L^p-norm of u computed over the active grid. The error for various problem sizes and schemes based on different blocks sizes is presented in Figure 2.1.

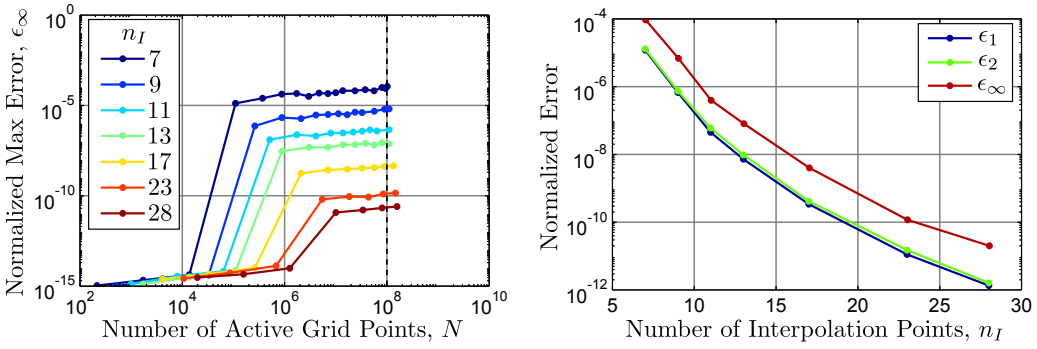

Figure 2.1: *Left*: max error, ϵ_∞, for cubic active grids containing N grid points partitioned into blocks with $n_b^0 = n_I - 1$ grid points along each direction. *Right*: error for test cases containing 10^8 grid points as function of n_I; the curve corresponding to ϵ_∞ is equivalent to the values on the *left* plot intersecting the vertical dashed line.

Test cases involving a small number blocks do not make use of the interpolation-based kernel compression technique; they only make use of the block-wise FFT-based convolution technique, which incurs an error close to machine precision. As a result, the three cases with the smallest values of N for each series have significantly smaller errors compared to their respective asymptotic (large N) error.

Given that we chose $n_{\max} = 10$ and constrained n_b^0 to be proportional to n_I, the error is expected to decay as n_I^{-10} for $n_I > n_{\max}$ (one order greater than interpolation order due to the $\sim |\mathbf{x}|^{-1}$ decay of the LGF). The data shown in Figure 2.1 are consistent with our estimates, exhibiting a behavior proportional to $n_I^{-10.7}$ for values of n_I between 11 and 28. The significant difference in the magnitude of ϵ_1 and ϵ_2 compared to ϵ_∞ suggests that the maximum error is concentrated in lower dimensional regions of the grid. Spatial plots of the error (not included) confirm that larger errors are always observed near or on the boundary points of blocks. These observations are characteristic of the interpolation scheme being used, which is known to exhibit larger errors near the boundaries of the interpolation interval (Runge phenomena).

Although the error for schemes with $n_I \neq n_b^0+1$ is not presented, it is readily deduced from our reported results that for any choice of n_b^0 the error can be controlled by changing n_I. Furthermore, different choices of $n_{\max}$ have not been explored since $n_{\max} = 10$ allows for schemes with errors as small as $\sim 10^{-12}$, which are sufficient for many practical applications. Errors smaller than 10^{-12} can be obtained by reducing $n_{\max}$ and increasing n_I.

2.4.2 Computation time

The test cases used to examine the computation time of the FLGF method follow the same setup as in Section 2.4.1, except that we now consider three types of active grid geometries: cubes, spheres, and spherical-shells (with a thickness of 0.1 diameters). For cases of spheres and spherical-shells, the set of blocks that defines the active grid

is constructed so as to approximate these geometries. Computation times for various schemes and problem sizes are presented in Figure 2.2, and asymptotic computation rates (number of active grid points per computation time) for a few schemes are included in Table 2.2.

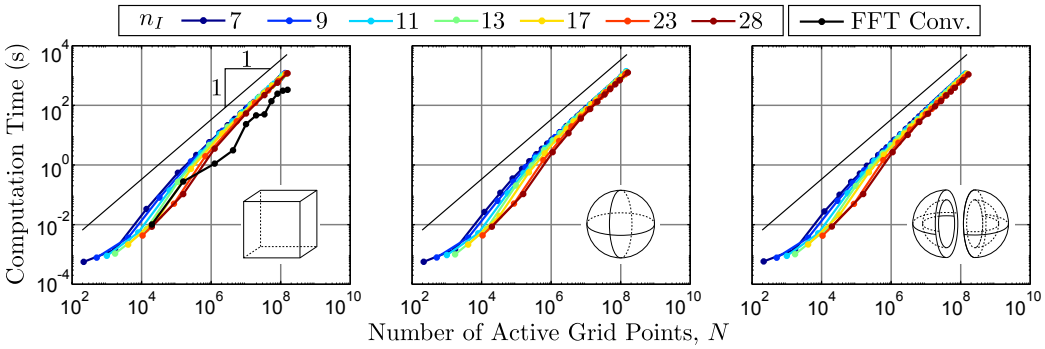

Figure 2.2: Computation times for active grids containing N grid points partitioned into blocks with $n_b^0 = n_I - 1$ point in each direction. The curve labeled "FFT Conv." corresponds to the special cases where the entire active grid is a single block. Results are presented for active grids with geometries approximating: cubes (*left*), spheres (*middle*), and spherical-shells (*right*).

Table 2.2: Approximate asymptotic computation rates for selected test cases presented in Figure 2.2. Values given in units of 10^5 pts/s. Rates are based on the test cases with 10^8 active grid points (values interpolated from nearest two data points). Asterisk ($*$) indicates rates that are not strictly asymptotic since they correspond to $\mathcal{O}(N \log N)$ schemes.

scheme	box	sphere	spherical-shell
$n_I = 7$	1.187	1.167	1.267
$n_I = 13$	1.189	1.169	1.315
$n_I = 28$	1.384	1.341	1.150
FFT Conv.	3.540*	n/a	n/a

As expected, the results for all schemes presented in Figure 2.2, with the exception of the one labeled "FFT Conv.", have an asymptotic computational complexity of $\mathcal{O}(N)$. FFT Conv. refers to the special case where the entire active grid is a single block for which the FLGF method reduces to a single FFT-based convolution. We note that in 3D the operation count of an FFT Conv. is roughly 8 times

the operation count of solving the systems of equations obtained from the spectral discretization of differential operators on periodic domains using FFTs (referred to as "FFT Periodic" in the subsequent discussion). FFT Conv. is a useful point of comparison since (1) many methods can readily consider the case of a single block with uniformly distributed sources, (2) it is well-established that for regular grids FFT-based elliptic solvers achieve very high, if not the highest, computation rates, and (3) the performance of an algorithm relative to FFT Conv. is approximately hardware independent.

Figure 2.2 demonstrates that the computation rates of our multi-block algorithm are within a factor of 10 of those corresponding to FFT Conv.. For large problems (our interest here), e.g. $N = \mathcal{O}(10^8)$, Table 2.2 indicates that our algorithm achieves computation rates that are roughly a third of the rate of FFT Conv. with up to 10 digits of accuracy. By comparison, the next closest method to ours, the 2D LGF FMM of Gillman and Martinsson [11], is claimed to be two orders of magnitude slower than an FFT Periodic. Even after accounting for the fact that in 2D an FFT Conv. is a factor of 4 slower than an FFT Periodic, we observe that our method has a significantly higher computation rate.[7] We also compare the performance of the present method to that of the black-box FMM of Fong and Darve [25]. In solving a 3D free-space Poisson problem with 10^6 uniformly distributed point sources over a cube with strengths ± 1, [25] reported a computation rate of about 1.8×10^3 points per second for 8 digits of accuracy. [8] Our method achieves a rate of about 1.2×10^5 points per second for 10 digits of accuracy, a speedup of about 65 times even with 2 more digits of accuracy. Finally, we observe that our computation

[7] For the numerical experiments reported, [11] stated that "the method was run at a requested relative precision of 10^{-10}". Given that [11] did not report the error of the experiments, we assume that computation rates quoted are for approximately 10 digits of accuracy, which is the same accuracy of the rates reported for the present method.

[8] We note that 8 digits is the maximum accuracy [25] reported for the 3D Laplace kernel. Additionally, we note that the hardware [25] used to perform the numerical experiments is not reported; therefore no attempt has been made to account for hardware differences.

rates are comparable with those of the adaptive (locally-refined) volume FMM of Langston et al. [35]. In solving a 3D free-space Poisson problem in a cube with sources corresponding to Gaussian bump solution, [35] reported computation rates between 2.6×10^4 and 1.3×10^5 points per second for cases with either 7 or 8 digits of accuracy (with number of unknowns between 2.8×10^6 and 2.6×10^7).[9]

A detail report of the computation time of the *Pre-computation* step is omitted, since this step is observed to require only a small fraction of the computation time of a single solve. We substantiate this claim by reporting that for test cases involving cubic active grids containing $\mathcal{O}(10^1)$, $\mathcal{O}(10^2)$, and $\mathcal{O}(10^4)$ blocks the *Pre-computation* step required less than 10%, 1%, and 0.1%, respectively, of the computation time of a single solve. These results are consistent with the fact that the operation count of the pre-computation step increases with the number of levels, which in turn increase logarithmically with the number of blocks for test cases involving cubic active grids.

2.4.3 Parallel performance

The parallel performance of the FLGF method is investigated by considering cubic active grids and using the scheme corresponding to $n_I = 13$ described in Sections 2.4.1 and 2.4.2. Computation rates and parallel efficiencies for various problem sizes with core counts between 12 and 660 are included in Figure 2.3. For all reported test cases the number of cores is a multiple of 12 (there are 12 cores per node in our test machine), and each MPI-process is mapped to a single core. The parallel

[9] The numerical experiments reported in [35] were performed using a shared-memory (OpenMP) implementation running on 16 cores of an Intel Xeon X7560 (2.27 GHz) based system. The rates included in the main text correspond to the rates we would expect to observe if the numerical experiments of [35] were performed on a single core of our local system; both the parallel efficiency (reported to be approximately 75% for 16 cores) and the difference in processor clock speed have been accounted for.

efficiency for each test series is defined by

$$e_N(p) = \frac{p_{\min}}{p} \frac{T_N(p_{\min})}{T_N(p)}, \tag{2.37}$$

where N is the total number of active grid points in the test series, p is the number of cores, $p_{\min}$ is the minimal number of cores considered in the test series, and $T_N(p)$ is the runtime of the test problem.

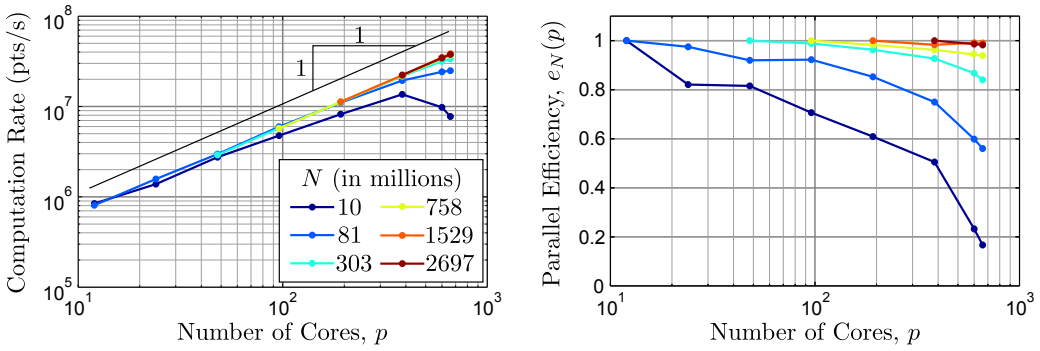

Figure 2.3: Computation rates (*left*) and parallel efficiencies (*right*) for cubic active grid of various sizes. The parameter n_I is set to 13 for all test cases. The listed values of N, in ascending order, correspond to problems containing 5.8×10^3, 4.6×10^4, 1.8×10^5, 4.4×10^5, 8.3×10^5, and 1.6×10^6 active blocks.

Both strong and weak scaling can be inferred from the left plot in Figure 2.3. Strong scaling, i.e. fixed N and increasing p, corresponds to the individual curves associated with each test series. Weak scaling, i.e. fixed N/p and increasing p, is achieved when the curves associated with different test series collapse. Figure 2.3 demonstrates that, over a reasonable range p, the curves for most of the test series collapse to a single line with a slope approximately equal to unity. This indicates that our implementation exhibits both good strong and weak scaling.

There are two main considerations that affect the performance of our parallel implementation. The first is the number of blocks per core. The FLGF method is broken into block-wise operations. If there are too few blocks per core our total work cannot be evenly distributed across all cores. Furthermore, given our communication scheme for the *Level Interactions* step, fewer blocks per core are likely to increase the

total number of MPI messages sent and received. The second consideration is the amount work per core. If the work per core is too small then communication cost can take an overwhelming fraction of the net run-time. Based on these considerations and the reported results, we conclude that if each core has, on average, more than 300,000 active grid points and 200 blocks, then the parallel efficiency, as defined in Eq. 2.37, is expected to be above 80%. This observation seems consistent with the results reported on other MPI-based implementations of kernel-independent FMMs, for example [36, 37].

In the interest of completeness, we note that computation rates reported in Figure 2.3, in particular those corresponding to 12 cores, are roughly half the rates expected based on the our serial results. This decrease in performance is due to an increase in cache-misses when more than one core per node is used. We expect that future higher-quality implementations of the FLGF method can readily mitigate this feature.

2.5 Conclusions

We have presented a new kernel-independent fast multipole method for elliptic difference equations on infinite Cartesian grids. The FLGF method exploits the regularity of the underlying grid to achieve small computation times by using a fast convolution technique that combines interpolation-based kernel compression and FFTs. Interpolation based on equidistant nodes, along with a p- and h-refinement technique, is shown to be an effective scheme for obtaining low-rank representations of kernels, while still preserving sufficient regularity to allow discrete convolutions to be performed quickly using FFTs. The adaptive block-structured grid strategy blends well with the overall algorithm, and the reported numerical experiments demonstrate that computation rates remain roughly invariant of source distributions.

The efficiency of the FLGF method is demonstrated through several numerical experiments solving the discrete 3D Poisson equation for cases involving up to 2 billion

grid points and 660 cores. Serial test cases confirm that the algorithm archives an asymptotic linear complexity. Computation rates of approximately 1.2×10^5 pts/s or, equivalently, grind times of 8.3μs are observed for problems containing 10^8 grid points. The computation rate is shown to be roughly invariant to different source distributions and block sizes. Furthermore, the time required to perform all pre-computations for typical problems is shown to negligible. Test cases investigating the parallel performance of our implementation demonstrate that parallel efficiencies higher than 80% are achieved under modest considerations (at least 200 blocks and $300,000$ grid points per core).

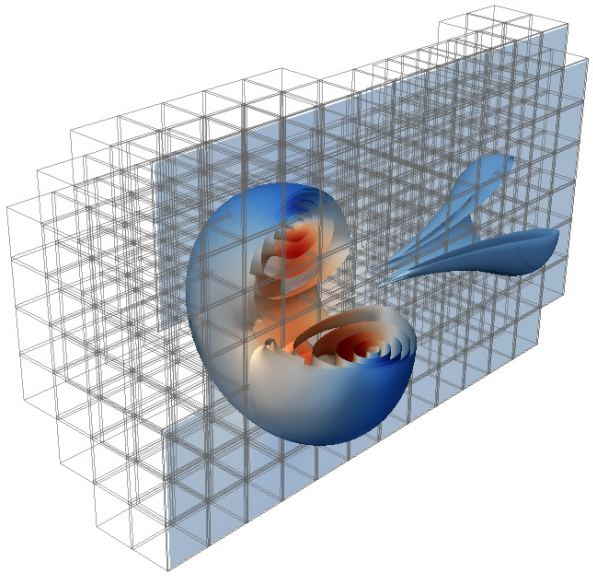

Figure 2.4: Vortex ring at a Reynolds number of 7,500. Isocontours correspond to the absolute value of vorticity (log scale), color corresponds to the streamwise velocity, and gray boxes correspond to the location of grid blocks used in the simulation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The FLGF method is particularly useful for solving PDEs that have been discretized using a numerical scheme that enforces discrete conservation laws. In such cases, accurate solutions to the difference equations, but not necessarily the original PDE, are necessary to preserve physical fidelity. We have applied the present method

to solve incompressible, viscous, external flows using a finite volume scheme and an infinite staggered Cartesian grid. Figure 2.4 includes a snapshot of thin vortex ring at a Reynolds number (based on ring circulation) of 7,500 simulated using this scheme. A detailed description and results pertaining to the application of the FLGF method to the incompressible Navier-Stokes are the subject of future publications.

In the interest of brevity, our discussion and reported results only pertain to the discrete Laplace kernel, yet the FLGF method can be applied to other non-oscillatory LGFs. In fact, our method can be readily generalized to any non-oscillatory kernel (including singular kernels); the only restriction is that sources and evaluation points be defined on a regular grid. Based on these observations, and the simple/standard routines and data-structures involved in the algorithm, it is expected that the FLGF method can be readily incorporated into a wide range of existing methods and codes that solve elliptic PDEs on unbounded domains.

APPENDICES

2.A Evaluating the LGF of the discrete Laplace operator numerically

Values of $\mathsf{G}(\mathbf{n})$ for small $|\mathbf{n}|$ are frequently used by the FLGF method. Therefore it is advantageous to program an accurate look-up table for values of $\mathbf{n}$ confined to a small cubic box centered at the origin. The symmetry of $\mathsf{G}(\mathbf{n})$, discussed in Section 2.2.1, suggests that only approximately 1/48-th of the total number of points in the box need to be numerically evaluated. It is possible to reduce the triple integral in Eq. 2.6 to a single semi-infinite integral [38] given by

$$\mathsf{G}(\mathbf{n}) = -\int_0^\infty e^{-6t} I_{n_1}(2t)\, I_{n_2}(2t)\, I_{n_3}(2t)\, dt, \tag{2.38}$$

where $I_k(x)$ is the modified Bessel function of the first kind of order k, and $\mathbf{n} = (n_1, n_2, n_3) \in \mathbb{Z}^3$. In our experience, it is easier and faster to numerically integrate Eq. 2.38 instead of Eq. 2.6. The integrand of Eq. 2.38 is non-oscillatory and smooth throughout the domain of integration. A simple adaptive Gauss-Kronrod scheme

can be used to perform the numerical integration and obtain error estimates. Furthermore, the semi-infinite integral can be partitioned into two intervals $[0, \alpha]$ and $[\alpha, \infty]$, where α is chosen such that the latter integral can be evaluated analytically using the asymptotic expansion (for large arguments) of $I_n(x)$ [39]. More efficient implementations might consider partitioning the integration interval into multiple subintervals, exploiting both the ascending series representation and the asymptotic expansion of each Bessel function.

2.B Communication patterns of MPI-based implementation

The pseudo-codes provided in this appendix complement the discussion regarding the parallel implementation of the present method included in Section 2.3.3. For convenience, in this appendix the term "node" is used to denote either grid blocks, grid intervals, or tree-nodes as defined in Section 2.3.1 and its precise meaning is deduced from the context. The algorithms discussed in this appendix are based on non-blocking MPI operations; we refer the reader to [40] for an introduction to these operations.

2.B.1 Level Interactions

The pseudo-code for the parallel implementation of *Level Interactions*, Step 2 of the FLGF algorithm of Section 2.3.1, is provided in Algorithm 1. A description of the terms and operations of Algorithm 1 is as follows:

- *Done sending (receiving)*: all non-blocking MPI messages being sent (received) have been posted and completed.
- *Done local-work*: any intra-MPI-process operations have been completed.
- *Send-/receive-messages*: As described in Section 2.3.3, each MPI-process sends, at most, one message to any other MPI-process. A message is composed of sub-messages, one for each target node. Each sub-message contains information iden-

Algorithm 1: Communication pattern of *Level Interactions*.

```
while not done sending or receiving or local-work do
    check status of all active messages;
    forall the receive-messages that have completed receive do
        process receive-message;
        mark receive-buffer unit associated with receive-message as available;
    forall the available receive-buffer units do
        if not done posting receive-messages then
            associate receive-message with receive-buffer unit;
            post non-blocking receive for receive-message;
    forall the send-messages that have completed send do
        mark send-buffer associated with send-message as available;
    if not done with all send-work units and send-buffer unit is available then
        if send-work unit corresponds to new send-message then
            associate send-message with send-buffer unit;
        perform M send-work units of send-message;
        if done building send-massage then
            post non-blocking send for send-message;
    else if not done with all local-work units then
        perform N local-work units;
```

tifying the target node, and the field induced on target node by all source nodes that belong to the sending MPI-process and interact with the target node.

- *Buffer and buffer units*: messages are buffered before being posted. The send-buffer and receive-buffer are composed of a fixed number of buffer-units. Each buffer-unit is allocated enough memory to handle any message that will be posted. The examples included in Section 2.4.3 use a total of four buffer units, two for sending and two for receiving.

- *Process receive-message*: read receive-message and add field contribution from non-local source nodes to local target nodes.

- *Send-work unit*: compute the field induced from a single local node to a target

node belonging to the current target MPI-process. The induced field of each target node is aggregated in send-buffer. Operations performed by a single send-work unit correspond to those of a single entry of the sum given in Eq. 2.31.

- *Local-work unit*: same as send-work unit, except that the induced field is added to the local storage of the target node (no need to buffer or perform any MPI communication).
- *Done building send-message*: the results from all send-work units required by send-message (or, equivalently, target MPI-process) have been packaged into a message.
- *Parameters M and N*: determine the number of work units performed at each iteration of the main loop. The examples included in Section 2.4.3 use $M = N = 20$.

2.B.2 Upwards and Downwards Pass

Algorithm 2: Communication pattern of *Upwards Pass*.

```
forall the local nodes do
    forall the non-local children of node do
        post non-blocking receive for message sent by child;
while not done sending or receiving or local-work do
    check status of all active messages and update receive-tracker;
    select M ready-for-processing nodes using receive-tracker;
    forall the selected nodes do
        if node has children then
            build node's weights by regularizing child nodes' weights;
        perform padded-FFT on node's weights;
        if node has a parent then
            if parent is non-local then
                post non-blocking send for message received by parent;
            else
                update local parent node's weights;
```

The pseudo-code for the parallel implementation of *Upwards Pass*, Step 1 of the FLGF algorithm of Section 2.3.1, is provided in Algorithm 2. A description of the terms and operations of Algorithm 2 is as follows:

- *Done sending (receiving) and done local-work*: same as in Appendix 2.B.1.
- *Message*: a messages contain a node's weights (sources or regularized sources from child nodes). Each node is provided enough auxiliary storage to receive the weights from all of its children.
- *Receive-tracker*: a tree-like data-structure that contains information regarding the progress of all communication and operations associated with each local node. It can be used to determine whether a node has finished receiving messages from all of its children (if any), whether a padded-FFT has been performed on its weights, and whether it has posted a non-blocking send to its parent (if any).
- *Ready-for-processing node*: a node that has not be processed, but has finished receiving messages from all of its children (if any). A node is said to be processed if its weights have been computed (or are known), i.e. Eq. 2.29, a padded-FFT has been performed on its weights, i.e. Eq. 2.30, and a non-blocking send to its parent-node has been posted (if parent exists).
- *Selecting ready-for-processing nodes*: the receive-tracker is transversed in leaf-to-root order and nodes that meet the ready-for-processing criteria are selected. Priority is given to nodes whose parent are non-local, i.e. belong to a different MPI-process.
- *Parameter M*: determine the number of nodes to be processed at each iteration of the main loop.

Pseudo-code for *Downwards Pass* is omitted since the communication pattern is very similar that of *Upwards Pass*. The only significant differences are that nodes send to their children, as opposed to their parent, and that when selecting ready-for-processing nodes the receive-tracker is transversed in root-to-leaf order, as opposed

to leaf-to-root order. The operations performed at each step and the ready-for-processing criteria are readily deduced from the discussion included in Sections 2.3.1 and 2.3.3.

Chapter 3

A FAST LATTICE GREEN'S FUNCTION METHOD FOR SOLVING VISCOUS INCOMPRESSIBLE FLOWS ON UNBOUNDED DOMAINS

Accepted for publication in Journal of Computational Physics, April 2016
(Appendices 3.E and 3.F have been added to the present manuscript)

Chapter abstract

A computationally efficient method for solving three-dimensional, viscous, incompressible flows on unbounded domains is presented. The method formally discretizes the incompressible Navier-Stokes equations on an unbounded staggered Cartesian grid. Operations are limited to a finite computational domain through a lattice Green's function technique. This technique obtains solutions to inhomogeneous *difference* equations through the discrete convolution of source terms with the fundamental solutions of the *discrete* operators. The differential algebraic equations describing the temporal evolution of the discrete momentum equation and incompressibility constraint are numerically solved by combining an integrating factor technique for the viscous term and a half-explicit Runge-Kutta scheme for the convective term. A projection method that exploits the mimetic and commutativity properties of the discrete operators is used to efficiently solve the system of equations that arises in each stage of the time integration scheme. Linear complexity, fast computation rates, and parallel scalability are achieved using recently developed fast multipole methods for difference equations. The accuracy and physical fidelity of solutions is verified through numerical simulations of vortex rings.

3.1 Introduction

Numerical simulations of viscous, incompressible flows on unbounded fluid domains require numerical techniques that can accurately approximate unbounded computational domains using only a finite number of operations. Spatial truncation and artificial boundary conditions have been developed for this purpose but they can adversely affect the accuracy of the solution and even change the dynamics of the flow [41–44]. Furthermore, minimizing the error due to artificial boundaries by employing large computational domains increases the number of computational elements

and often requires the use of solvers that are less efficient than those used on regular grids (e.g. FFT techniques, multigrid, etc.).

Recently, fast multipole methods (FMMs) for solving constant coefficient elliptic *difference* equations on unbounded regular grids have been developed for 2D [10, 11] and 3D [1] problems. These methods obtain solutions to inhomogeneous *difference* equations by using fast summation techniques to evaluate the discrete convolution of source terms with the fundamental solutions of the *discrete* operators. The fundamental solutions of discrete operators on unbounded regular grids, or lattices, are also referred to as lattice Green's functions (LGFs).

Similar to particle and vortex methods, e.g. [20, 23, 30, 45–54] and references therein, the LGF techniques discussed in [1, 10, 11] have efficient nodal distributions and automatically enforce free-space boundary conditions. As a result, needlessly large computational domains and artificial boundary conditions can be avoided when solving flows on unbounded regular grids by using LGF techniques to compute the action of solution operators. A significant advantage of recently developed particle and vortex methods is their ability to efficiently solve large scale problems relevant to 3D incompressible flows using fast, parallel methods based on techniques such as tree-codes, FMMs, dynamic error estimators, hybrid Eulerian-Lagrangian formulations, hierarchical grids, FFT methods, and domain decomposition techniques [23, 30, 47–49, 52–54]. It is demonstrated in [1] that LGF FMMs can achieve computational rates and parallel scaling for 3D discrete (7-pt Laplacian) Poisson problems comparable to existing fast 3D Poisson solvers.

The present formulation numerically solves the incompressible Navier-Stokes equations expressed in the non-dimensional form given by

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}, \tag{3.1a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{3.1b}$$

where $\mathbf{u}$, p, and Re correspond to the velocity, the pressure, and the Reynolds

number, respectively. The equations are defined on an unbounded domain in all directions, and are subject to the boundary conditions

$$\mathbf{u}(\mathbf{x},t) \to \mathbf{u}_\infty(t) \text{ as } |\mathbf{x}| \to \infty, \tag{3.2}$$

where $\mathbf{u}_\infty$ is a known time-dependent function. We limit our attention to flows in which the vorticity, $\boldsymbol{\omega} = \nabla \times \mathbf{u}$, decay exponentially fast as $|\mathbf{x}| \to \infty$.

The present formulation is simplified by considering the evolution of the velocity perturbation, $\mathbf{u}'(\mathbf{x},t) = \mathbf{u}(\mathbf{x},t) - \mathbf{u}_\infty(t)$, and pressure perturbation, $p'(\mathbf{x},t) = p(\mathbf{x},t) - p_\infty(\mathbf{x},t)$. The freestream pressure, p_∞, is given by

$$p_\infty(\mathbf{x},t) = \frac{d\mathbf{u}_\infty}{dt} \cdot \mathbf{x}, \tag{3.3}$$

where we have taken the arbitrary time-dependent constant to be zero. Subtracting the uniform freestream equations from Eq. (3.1) yields

$$\frac{\partial \mathbf{u}'}{\partial t} + (\mathbf{u}' + \mathbf{u}_\infty) \cdot \nabla \mathbf{u}' = -\nabla p' + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}', \quad \nabla \cdot \mathbf{u}' = 0, \tag{3.4}$$

subject to the boundary conditions $\mathbf{u}'(\mathbf{x},t) \to 0$ as $|\mathbf{x}| \to \infty$. The boundary conditions on $\mathbf{u}'$ and the irrotational nature of the flow at large distances imply that p' is subject to the compatibility condition[1]

$$p'(\mathbf{x},t) \to 0 \text{ as } |\mathbf{x}| \to \infty. \tag{3.5}$$

The remainder of the paper is organized as follows. In Section 3.2, we describe the spatial discretization of the governing equations on formally unbounded staggered Cartesian grids and discuss LGF techniques that can be used to obtain fast solutions to the associated discrete elliptic problems. Additionally, we present an integrating

[1] In the absence of sources and sinks, the velocity of an irrotational flow subject to zero boundary conditions at infinity is given by $\mathbf{v} = \nabla\phi$, where the leading order term of ϕ is $-\mathbf{M} \cdot \mathbf{x}/r^3$ [55]. Consequently, $p = -\left(\frac{\partial \phi}{\partial t} + \frac{1}{2}|\nabla\phi|^2\right) \to 0$ as $r \to \infty$, where we have taken the arbitrary time-dependent constant to be zero.

factor technique that facilitates the implementation of efficient, robust time integration schemes. In Section 3.3, the system of differential algebraic equations (DAEs) resulting from the spatial discretization and integrating factor techniques is numerically solved using a half-explicit Runge-Kutta method. We show that the linear systems of equations that arise at each stage of the time integration scheme can be efficiently solved, without splitting errors or additional stability constraints, by a fast projection method based on LGF techniques and the properties of the discrete operators. In Section 3.4, we demonstrate that an adaptive block-structured grid padded with appropriately sized buffer regions can be used to efficiently compute numerical solutions to a prescribed tolerance. In Section 3.5, we summarize the algorithm and discuss a few practical considerations including computational costs and performance optimization. Finally, in Section 3.6, we perform numerical experiments on vortex rings to verify the present formulation.

3.2 Spatial discretization

3.2.1 Unbounded staggered Cartesian grids

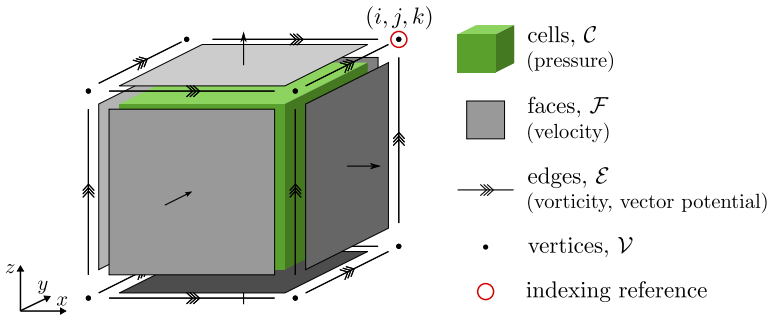

Figure 3.1: Unit cell of the staggered Cartesian grid. The vertex enclosed by the circle corresponds to the (i, j, k) vertex. The (i, j, k) cell, faces, and edges correspond to the depicted elements intersecting the (i, j, k) vertex. There are three faces and edges per vertex. The superscript "(q)" is used to denote faces (edges) normal (parallel) to x_q axis.

In this section we describe the discretization of Eq. (3.4) on a formally unbounded

staggered Cartesian grid. Figure 3.1 depicts our staggered grid, which consists of cells ($\mathcal{C}$) and vertices ($\mathcal{V}$) that house scalar quantities, and faces ($\mathcal{F}$) and edges ($\mathcal{E}$) that house vector quantities. The notation $\mathbb{R}^{\mathcal{Q}}$ denotes the set of real-valued grid functions with values defined on $\mathcal{Q} \in \{\mathcal{C}, \mathcal{F}, \mathcal{E}, \mathcal{V}\}$. The value of a grid function q evaluated at $\mathbf{n} = (i, j, k) \in \mathbb{Z}^3$ is given by $\mathsf{q}(\mathbf{n})$ and $\mathsf{q}_{i,j,k}$. For the case of a vector-valued grid function q, i.e. $\mathsf{q} \in \mathbb{R}^{\mathcal{F}}$ or $\mathsf{q} \in \mathbb{R}^{\mathcal{E}}$, $\mathsf{q}^{(k)}(\mathbf{n})$ denotes the component of $\mathsf{q}(\mathbf{n})$ in the k-th direction.

The spatial discretization of Eq. (3.4) is performed using the techniques of Nicolaides and Wu [56], and Zhang et al. [57]. The resulting discrete operators are similar or equivalent to those obtained from standard second-order finite-volume or finite-difference schemes, e.g. [58]. Yet we refer to the more general techniques of [56] and [57] since their discussions emphasize many of the algebraic properties of the discrete operators used by the present formulation. For convenience, point-operator representations of the discrete operators are included in Appendix 3.A.

The semi-discrete system of equations obtained from the spatial discretization of Eq. (3.4) is

$$\frac{d\mathsf{u}}{dt} + \mathsf{N}(\mathsf{u} + \mathsf{u}_\infty) = -\mathsf{G}\mathsf{p} + \frac{1}{\mathrm{Re}}\mathsf{L}_{\mathcal{F}}\mathsf{u}, \quad \overline{\mathsf{D}}\mathsf{u} = 0, \tag{3.6}$$

where $\mathsf{u} \in \mathbb{R}^{\mathcal{F}} \times \mathbb{R}$ and $\mathsf{p} \in \mathbb{R}^{\mathcal{C}} \times \mathbb{R}$ denote the time-dependent grid functions associated with the discrete velocity and pressure perturbation fields, respectively.[2] The time-dependent grid function $\mathsf{u}_\infty \in \mathbb{R}^{\mathcal{F}} \times \mathbb{R}$ is constant in space with values given by $\mathsf{u}_\infty(\mathbf{n}, t) = \mathbf{u}_\infty(t)$. Discrete operators $\mathsf{G} : \mathbb{R}^{\mathcal{C}} \mapsto \mathbb{R}^{\mathcal{F}}$, $\overline{\mathsf{D}} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{C}}$, and $\mathsf{L}_{\mathcal{F}} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{F}}$ correspond to the discretizations of the gradient, divergence, and vector Laplacian operators, respectively. Finally, $\mathsf{N} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{F}}$ denotes the discrete nonlinear operator approximating the convective term, i.e. $\mathsf{N}(\mathsf{u} + \mathsf{u}_\infty) \approx$

[2] In order to avoid a cumbersome notation, the prime symbols, $'$, are omitted from variables denoting grid functions associated with the perturbations of the discrete velocity and pressure fields.

$(\mathbf{u}' + \mathbf{u}_\infty) \cdot \nabla (\mathbf{u}' + \mathbf{u}_\infty) = (\mathbf{u}' + \mathbf{u}_\infty) \cdot \nabla \mathbf{u}'$.[3]

In addition to the aforementioned discrete operators, the subsequent discussion makes use of the discrete gradient operator $\overline{\mathsf{G}} : \mathbb{R}^{\mathcal{V}} \mapsto \mathbb{R}^{\mathcal{E}}$, the discrete curl operators $\mathsf{C} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{E}}$ and $\overline{\mathsf{C}} : \mathbb{R}^{\mathcal{E}} \mapsto \mathbb{R}^{\mathcal{F}}$, and the discrete Laplacian operators $\mathsf{L}_{\mathcal{Q}} : \mathbb{R}^{\mathcal{Q}} \mapsto \mathbb{R}^{\mathcal{Q}}$, where $\mathcal{Q} \in \{\mathcal{C}, \mathcal{E}, \mathcal{V}\}$. A summary of all the discrete vector operators and their definitions is also provided in Appendix 3.A.

The choice of discretization technique yields a numerical scheme with the following properties:

- *Second-order accuracy*: all discrete operators are second-order accurate in space.
- *Conservation properties*: using appropriate discretizations of the nonlinear convective term leads to a scheme that conserves momentum, kinetic energy, and circulation in the absence of time-differencing errors and viscosity [57, 59, 60]. The benefits of discrete conservation properties related to numerical stability and physical fidelity are discussed in the review by Perot [13] and references therein.
- *Mimetic properties*: discrete operators and their corresponding vector calculus operators satisfy similar symmetry and orthogonality properties in addition to similar integration by parts formulas [56, 57, 61, 62]. Specific properties pertinent to the discussion of the present method are:

$$\overline{\mathsf{D}} = -\mathsf{G}^\dagger, \quad \overline{\mathsf{C}} = \mathsf{C}^\dagger, \quad \overline{\mathsf{G}} = -\mathsf{D}^\dagger, \tag{3.7a}$$

$$\mathrm{Null}(\mathsf{C}) = \mathrm{Im}(\mathsf{G}),\ \mathrm{Null}(\mathsf{D}) = \mathrm{Im}(\mathsf{C}), \tag{3.7b}$$

$$\mathsf{L}_{\mathcal{C}} = -\mathsf{G}^\dagger\mathsf{G},\ \mathsf{L}_{\mathcal{F}} = -\mathsf{G}\mathsf{G}^\dagger - \mathsf{C}^\dagger\mathsf{C},\ \mathsf{L}_{\mathcal{E}} = -\mathsf{D}^\dagger\mathsf{D} - \mathsf{C}\mathsf{C}^\dagger,\ \mathsf{L}_{\mathcal{V}} = -\mathsf{D}\mathsf{D}^\dagger. \tag{3.7c}$$

  Many of the mimetic properties of discrete operators are closely related to the conservation properties [56, 57].

[3] No particular form (e.g. convection, rotational, divergence, skew-symmetric) or discretization scheme for the convection term is assumed by Eq. (3.6).

- *Commutativity properties*: on unbounded staggered grids, discrete Laplacians and integrating factors (to be introduced in Section 3.2.3) are able to commute with other operators in the sense $\mathsf{A}\mathsf{T}_{\mathcal{X}} = \mathsf{T}_{\mathcal{Y}}\mathsf{A}$, where $\mathsf{A} : \mathbb{R}^{\mathcal{X}} \mapsto \mathbb{R}^{\mathcal{Y}}$ is any of the previously mentioned linear operators, and $\mathsf{T}_{\mathcal{X}}$ ($\mathsf{T}_{\mathcal{Y}}$) is either the discrete Laplacian or integrating factor mapping $\mathbb{R}^{\mathcal{X}}$ to $\mathbb{R}^{\mathcal{X}}$ ($\mathbb{R}^{\mathcal{Y}}$ to $\mathbb{R}^{\mathcal{Y}}$). Similar commutativity properties result in discretizations of periodic domains using uniform staggered grids.

In subsequent sections we discuss how the mimetic and commutativity properties facilitate the construction of fast, stable methods for numerically solving Eq. (3.6).

It is convenient to define

$$\mathsf{d} = \mathsf{p} + \frac{1}{2}\mathsf{P}\left(\mathsf{u} + \mathsf{u}_\infty, \mathsf{u} + \mathsf{u}_\infty\right), \tag{3.8}$$

where $\mathsf{P} : \mathbb{R}^{\mathcal{F}} \times \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{C}}$ is an arbitrary discrete approximation of the vector dot-product, i.e. $\mathsf{P}(\mathsf{u}, \mathsf{v}) \approx \mathbf{u} \cdot \mathbf{v}$. The time-dependent grid function $\mathsf{d} \in \mathbb{R}^{\mathcal{C}} \times \mathbb{R}$ can be regarded as a discrete approximation of the total pressure perturbation, i.e. $\mathsf{d} \approx p' + \frac{1}{2}|\mathbf{u}' + \mathbf{u}_\infty|^2$. Using Eq. (3.8), we express Eq. (3.6) as

$$\frac{d\mathsf{u}}{dt} + \tilde{\mathsf{N}}(\mathsf{u} + \mathsf{u}_\infty) = -\mathsf{G}\mathsf{d} + \frac{1}{\mathrm{Re}}\mathsf{L}_{\mathcal{F}}\mathsf{u}, \quad \mathsf{G}^\dagger\mathsf{u} = 0, \tag{3.9}$$

where $\tilde{\mathsf{N}}(\mathsf{v}) = \mathsf{N}(\mathsf{v}) - \frac{1}{2}\mathsf{G}\mathsf{P}(\mathsf{v}, \mathsf{v})$. Consequently, $\tilde{\mathsf{N}}(\mathsf{u}+\mathsf{u}_\infty)$ is a discrete approximation of $\boldsymbol{\omega} \times (\mathbf{u} + \mathbf{u}_\infty)$.[4] As will be demonstrated in Section 3.4, an advantage of using $\tilde{\mathsf{N}}(\mathsf{u} + \mathsf{u}_\infty)$ instead of $\mathsf{N}(\mathsf{u} + \mathsf{u}_\infty)$ is that the former typically has a smaller support than that of the latter, which in turn reduces the number of operations and storage required to numerically solve the flow. We emphasize that Eq. (3.9) is equivalent to Eq. (3.6), and no additional discretization errors have been introduced.

[4] The discretization of Eq. (3.4) naturally assumes the form given by Eq. (3.9) if the convection term is discretized in its rotational form, $(\nabla \times \mathbf{v}) \times \mathbf{v} + \frac{1}{2}\nabla\mathbf{v}^2$, with the gradient term approximated by $\frac{1}{2}\mathsf{G}\mathsf{P}(\mathsf{v}, \mathsf{v})$.

3.2.2 Lattice Green's function techniques

The procedure for solving difference equations on unbounded regular grids using LGFs is analogous to the procedure for solving inhomogeneous PDEs on unbounded domains using the fundamental solution of continuum operators. As a representative example, we consider the (continuum) scalar Poisson equation

$$[\Delta u](\mathbf{x}) = f(\mathbf{x}), \quad supp(f) \subseteq \Omega, \tag{3.10}$$

where $\mathbf{x} \in \mathbb{R}$ and Ω is a bounded domain in $\mathbb{R}^3$. The solution to Eq. (3.10) is given by

$$u(\mathbf{x}) = [G * f](\mathbf{x}) = \int_{\Omega} G(\mathbf{x} - \mathbf{y}) f(\mathbf{y})\, d\mathbf{y}, \tag{3.11}$$

where $G(\mathbf{x}) = -1/(4\pi|\mathbf{x}|)$ is the fundamental solution of the Laplace operator. Similarly, we consider the discrete scalar Poisson equation

$$[\mathsf{L}_{\mathcal{Q}}\mathsf{u}](\mathbf{n}) = \mathsf{f}(\mathbf{n}), \quad supp(\mathsf{f}) \subseteq D, \tag{3.12}$$

where $\mathsf{u}, \mathsf{f} \in \mathbb{R}^{\mathcal{Q}}$, D is a bounded region in $\mathbb{Z}^3$, and $\mathcal{Q} \in \{\mathcal{C}, \mathcal{V}\}$. The solution to Eq. (3.12) is given by

$$\mathsf{u}(\mathbf{n}) = [\mathsf{G}_{\mathsf{L}} * \mathsf{f}](\mathbf{n}) = \sum_{\mathbf{m} \in D} \mathsf{G}_{\mathsf{L}}(\mathbf{n} - \mathbf{m}) \mathsf{f}(\mathbf{m}) \tag{3.13}$$

where $\mathsf{G}_{\mathsf{L}} : \mathbb{Z}^3 \mapsto \mathbb{R}$ is the fundamental solution, or LGF, of the discrete scalar Laplacian [1, 11]. Subsequently, we refer to the grid functions f and u as the source field and the induced field, respectively.

It is evident from the definitions of $\mathsf{L}_{\mathcal{F}}$ and $\mathsf{L}_{\mathcal{E}}$ that each component of a discrete vector Poisson problem corresponds to a discrete scalar Poisson problem. As a result, the q-th component of solutions to Eq. (3.12) for $\mathcal{Q} \in \{\mathcal{F}, \mathcal{E}\}$ are given by Eq. (3.13) with $\mathsf{u} \to \mathsf{u}^{(q)}$ and $\mathsf{f} \to \mathsf{f}^{(q)}$. Procedures for obtaining expressions for $\mathsf{G}_{\mathsf{L}}(\mathbf{n})$ are discussed in [5, 9, 16, 18]. For convenience, expressions for $\mathsf{G}_{\mathsf{L}}(\mathbf{n})$ are provided in Appendix 3.B.

Fast numerical methods for evaluating discrete convolutions involving LGFs have recently been proposed in 2D by Gillman and Martinsson [11] and in 3D by Liska and Colonius [1]. The 3D Fast Lattice Green's Function (FLGF) method of [1] is used to evaluate discrete convolutions involving G_L. The FLGF method is a kernel-independent interpolation based fast multipole method (FMM) specifically designed for solving difference equations on unbounded Cartesian grids. In addition to its asymptotic linear algorithmic complexity, it has been shown that the FLGF method achieves high computation rates and good parallel scaling for the case of G_L [1].

As final remark, the FLGF method is a direct solver that computes solutions to a prescribed tolerance ϵ, $\|\mathsf{y}_{\text{true}} - \mathsf{y}\|_\infty / \|\mathsf{y}_{\text{true}}\|_\infty \leq \epsilon$, where y is the numerical solution and y_{true} is the exact solution to the system of *difference* equations. In order to obtain accurate error bounds for the FLGF method it is necessary to profile the method once for each kernel and scheme used. Error estimates for the discrete 7-pt Laplace kernel and different schemes are provided in [1]. In the present formulation, all instances of $\mathsf{E}_\mathcal{Q}$ and $\mathsf{L}_\mathcal{Q}^{-1}$ are computed using values of ϵ that are less than or equal to prescribed value of ϵ_{FLGF}.

3.2.3 Integrating factor techniques

In this section we describe an integrating factor technique for integrating the stiff viscous term of Eq. (3.9) analytically. Analytical integration has the advantage of neither introducing discretization errors nor imposing stability constraints on the time marching scheme. Integrating factor techniques for the viscous term are widely used in Fourier pseudo-spectral methods. These methods typically compute the action of the integrating factor in Fourier-space. In contrast, the present method computes the action of the integrating factor in real-space, since the Fourier series of an arbitrary grid function on an unbounded domain is not computationally practical.

We consider integrating factors defined as the solution operators of the discrete

diffusion equation of the form

$$\frac{d\mathsf{h}}{dt} = \kappa \mathsf{L}_{\mathcal{Q}} \mathsf{h}, \quad \mathsf{h}(\mathbf{n}, t) \to \mathsf{h}_\infty(t) \text{ as } |\mathbf{n}| \to \infty, \tag{3.14}$$

where $\kappa \in \mathbb{R}_{\geq 0}$ and $\mathsf{h} \in \mathbb{R}^{\mathcal{Q}}$. As discussed in Appendix 3.A, the discrete Laplace operator $\mathsf{L}_{\mathcal{Q}}$ is diagonalized by the Fourier series operator $\mathfrak{F}_{\mathcal{Q}}$,

$$(\Delta x)^2 \mathsf{L}_{\mathcal{Q}} = \mathfrak{F}_{\mathcal{Q}}^{-1} \sigma_{\mathcal{Q}}^{\mathsf{L}} \mathfrak{F}_{\mathcal{Q}}, \tag{3.15}$$

where $\sigma_{\mathcal{Q}}^{\mathsf{L}}(\boldsymbol{\xi})$ for $\boldsymbol{\xi} \in (\pi, \pi)^3$ is the spectrum of $(\Delta x)^2 \mathsf{L}_{\mathcal{Q}}$. Next, we define the exponential of the $\mathsf{L}_{\mathcal{Q}}$ as

$$\mathsf{E}_{\mathcal{Q}}(\alpha) = \mathfrak{F}_{\mathcal{Q}}^{-1} \exp(\alpha \sigma_{\mathcal{Q}}^{\mathsf{L}}) \mathfrak{F}_{\mathcal{Q}}, \tag{3.16}$$

where $\alpha = \kappa(t - \tau)/(\Delta x)^2$. An immediate consequence of Eq. (3.16) is that

$$\frac{d}{d\alpha} \mathsf{E}_{\mathcal{Q}}(\alpha) = \mathfrak{F}_{\mathcal{Q}}^{-1} \sigma_{\mathcal{Q}}^{\mathsf{L}} \exp(\alpha \sigma_{\mathcal{Q}}^{\mathsf{L}}) \mathfrak{F}_{\mathcal{Q}}^{-1} = \mathsf{L}_{\mathcal{Q}} \mathsf{E}_{\mathcal{Q}}(\alpha) = \mathsf{E}_{\mathcal{Q}}(\alpha) \mathsf{L}_{\mathcal{Q}}, \tag{3.17}$$

which implies that the solution to Eq. (3.14) is given by

$$\mathsf{h}(\mathbf{n}, t) = \left[\mathsf{E}_{\mathcal{Q}} \left(\frac{\kappa(t - \tau)}{(\Delta x)^2} \right) \mathsf{h}_\tau \right] (\mathbf{n}, t), \quad t \geq \tau, \quad \forall \mathbf{n} \in \mathbb{Z}^3, \tag{3.18}$$

where $\mathsf{h}(\mathbf{n}, \tau) = \mathsf{h}_\tau(\mathbf{n})$.

We now consider using $\mathsf{E}_{\mathcal{Q}}(\alpha)$ as an integrating factor for Eq. (3.9). Operating from the left on the semi-discrete momentum equation of Eq. (3.9) with $\mathsf{E}_{\mathcal{F}} \left(\frac{t-\tau}{(\Delta x)^2 \mathrm{Re}} \right)$ and introducing the transformed variable $\mathsf{v} = \mathsf{E}_{\mathcal{F}} \left(\frac{t-\tau}{(\Delta x)^2 \mathrm{Re}} \right) \mathsf{u}$ yields the transformed system of semi-discrete equations

$$\frac{d\mathsf{v}}{dt} = -\mathsf{H}_{\mathcal{F}} \tilde{\mathsf{N}} \left(\mathsf{H}_{\mathcal{F}}^{-1} \mathsf{v} + \mathsf{u}_\infty \right) - \mathsf{H}_{\mathcal{F}} \mathsf{G} \mathsf{d}, \quad \mathsf{G}^\dagger \mathsf{H}_{\mathcal{C}}^{-1} \mathsf{v} = 0, \tag{3.19}$$

where $\mathsf{H}_{\mathcal{Q}} = \mathsf{E}_{\mathcal{Q}} \left(\frac{t-\tau}{(\Delta x)^2 \mathrm{Re}} \right)$. Using the commutativity properties of integrating factors, Eq. (3.19) simplifies to

$$\frac{d\mathsf{v}}{dt} = -\mathsf{H}_{\mathcal{F}} \tilde{\mathsf{N}} \left(\mathsf{H}_{\mathcal{F}}^{-1} \mathsf{v} + \mathsf{u}_\infty \right) - \mathsf{G} \mathsf{b}, \quad \mathsf{G}^\dagger \mathsf{v} = 0, \tag{3.20}$$

where $\mathsf{b} = \mathsf{H}_{\mathcal{F}}\mathsf{d}$. We emphasize that the transformed system of equations Eq. (3.20) is equivalent to the original system of equation Eq. (3.9). Furthermore, as is the case for Eq. (3.9), Eq. (3.20) represents a system of DAEs of index 2.

The procedures for obtaining expressions $\mathsf{G}_{\mathsf{L}}(\mathbf{n})$ can be readily extended to the case of $[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{n})$, where $\mathsf{G}_{\mathsf{E}}(\alpha)$ is the LGF of the integrating factor $\mathsf{E}_{\mathcal{Q}}(-\alpha)$. Expressions for $\mathsf{G}_{\mathsf{E}}(\mathbf{n})$ are also provided in Appendix 3.B. As for the case of $\mathsf{L}_{\mathcal{Q}}^{-1}$, fast solutions to expressions involving $\mathsf{G}_{\mathsf{E}}(\alpha)$ are computed using the FLGF method.

An important distinction between $\mathsf{G}_{\mathsf{L}}(\mathbf{n})$ and $[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{n})$ is found in their asymptotic behavior. Whereas the value of $|\mathsf{G}_{\mathsf{L}}(\mathbf{n})|$ decays as $1/|\mathbf{n}|$ as $|\mathbf{n}| \to \infty$, the value of $|[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{n})|$ decays faster than any exponential as $|\mathbf{n}| \to \infty$ for a fixed α.[5] The fast decay of G_{E} implies that, for typical computations, the application of $\mathsf{E}_{\mathcal{Q}}$ can be consider a local operation, i.e. values computed at a particular grid location only depend on the values of a few neighboring grid cells. Consequently, the FLGF method requires significantly fewer operations to evaluate the action of $\mathsf{E}_{\mathcal{Q}}$ compared to the action of $\mathsf{L}_{\mathcal{Q}}^{-1}$.[6]

3.3 Time integration

3.3.1 Half-explicit Runge-Kutta methods

Failing to properly identify the semi-discrete form of the governing equations, i.e. Eq. (3.9), as a system of differential algebraic equations (DAEs) of index 2 prior to choosing a time integration scheme can have undesirable consequences on the quality of the numerical solution [63, 64]. Half-explicit Runge-Kutta (HERK) methods are

[5] Consider $[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{n})$ for the case $\mathbf{n} = (n, 0, 0)$. As $n \to \infty$, $[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{n}) \sim \alpha^n/n!$. For $\alpha = 0.1$ and $\alpha = 1.0$, the value of $[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{n})/[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{0})$ is less than 10^{-10} at $n = 7$ and $n = 13$, respectively. The numerical simulations of Section 3.6 make use of integrating factors with $\alpha < 1$, but larger values of α are allowed.

[6] For the run parameters of the numerical experiments of Section 3.6, the action of $\mathsf{E}_{\mathcal{Q}}$ only requires approximately 10% of the total number of operations required to compute $\mathsf{L}_{\mathcal{Q}}^{-1}$.

a type of one-step time integration schemes developed for DAEs of index 2 [63, 65, 66]. Although there are multiple HERK methods [63], we limit our attention to the original HERK method proposed by Hairer et al. [65].

Consider DAE systems of index 2 of the form

$$\frac{dy}{dt} = f\left(y, z\right), \quad g\left(y\right) = 0, \tag{3.21}$$

where f and g are sufficiently differentiable, and z is an unknown that must be computed so as to have y satisfy $g(y) = 0$. Problems of this form are of index 2 if the product of partial derivatives $g_y(y)f_z(y, z)$ is non-singular in a neighborhood of the solution. The HERK method applied to Eq. (3.21) is given by an algorithm similar to that of explicit Runge-Kutta (ERK) methods except that the implicit constraint equation $g\left(y\right) = 0$ is solved at each stage of the ERK scheme.

Similarly to standard RK methods, HERK methods can be described by their Butcher tableau:

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^{\dagger} \end{array}, \tag{3.22}$$

where $\mathbf{A} = [a_{i,j}]$ is the Runge-Kutta matrix, $\mathbf{b} = [b_i]$ is the weight vector, and $\mathbf{c} = [c_i]$ is the node vector. In subsequent sections, it is often convenient to use the *shifted* tableau notation:

$$\tilde{a}_{i,j} = \begin{cases} a_{i+1,j} & \text{for } i = 1, 2, \ldots, s-1 \\ b_j & \text{for } i = s \end{cases}, \quad \tilde{c}_i = \begin{cases} c_{i+1} & \text{for } i = 1, 2, \ldots, s-1 \\ 1 & \text{for } i = s \end{cases}. \tag{3.23}$$

We refer the reader to the discussions of [65, 66] for a detailed algorithm and a list of order-conditions for the general case of Eq. (3.21).

We now turn our attention to the special case of the transformed semi-discrete governing equations given by Eq. (3.20). It is convenient to express the non-autonomous system of Eq. (3.20) in terms of the autonomous system of Eq. (3.21). This is achieved by letting $y = [\mathsf{v}, t]$ and $z = \mathsf{b}$, and by adding $t' = 1$ to Eq. (3.20).

For this case, $g_y = [\,\mathsf{G}^\dagger,\, 0]$ and $f_z = g_y^\dagger$, where $g_y = [g_\mathsf{u}, g_t]$ and $f_z = f_\mathsf{b}$. By construction, the operator G is a constant, which implies that f_z and g_y are also constants. As a result, order-conditions for the general system of Eq. (3.21) involving high-order derivatives of f_z and g_y are trivially satisfied for the case of Eq. (3.20). Fewer order-conditions permit a wider range of RK tableaus to be used for a given order of accuracy. This is particularly relevant for high-order HERK schemes, since the number of order-conditions is significantly larger than that of standard RK schemes [66].

The simplifications in the order-conditions obtained for the special case of constant f_z and g_y are well-described in the literature of HERK methods [63, 65–67]. Order-conditions up to order 4 for the y-component reduce to those of standard RK methods [67]. Similarly, order-conditions of order $r \leq 3$ for the z-component (up to fourth-order accurate z-component) reduce to having the shifted sub-tableau $[\tilde{a}_{i,j}]$ for $i, j = 1, 2, \ldots s-1$ satisfy the y-component order-conditions up to order r [66, 67]. It is beyond the scope of the present work to provide an extended discussion on the properties and implementation details of the HERK method for particular RK tableaus. Instead, the order of accuracy and linear stability of a few selected schemes used to perform the numerical experiments of Section 3.6 is discussed in Section 3.3.2 and Appendix 3.C, respectively.

3.3.2 Combined integrating factor and half-explicit Runge-Kutta method

In this section we present a method for obtaining numerical solutions for the (untransformed) discrete velocity and total pressure perturbation by combining the integrating factor technique of Section 3.2.3 with the HERK method of Section 3.3.1. The combined method, referred to as the IF-HERK method, integrates Eq. (3.6) over $t \in [0, T]$ subject to the initial condition $\mathsf{u}(\mathbf{n}, 0) = \mathsf{u}_0(\mathbf{n})$.

Formally, the IF-HERK method partitions the original problem into a sequences of n sub-problems, where the k-th sub-problem corresponds to numerical integration

of Eq. (3.6) from t_k to t_{k+1} subject to the initial condition $\mathsf{u}(\mathbf{n}, t_k) = \mathsf{u}_k(\mathbf{n})$. We restrict our discussion to the case of equispaced time-steps, i.e. $t_k = t_{k-1} + \Delta t$, since the more general case of variable time-step size is readily deduced.

The k-th sub-problem is solved by first introducing the transformed variables

$$\mathsf{v}(\mathbf{n}, t) = \left[\mathsf{E}_{\mathcal{F}}\left(\tfrac{\Delta t}{(\Delta x)^2 \mathrm{Re}}\right)\right] \mathsf{u}(\mathbf{n}, t), \;\; \mathsf{b}(\mathbf{n}, t) = \left[\mathsf{E}_{\mathcal{F}}\left(\tfrac{\Delta t}{(\Delta x)^2 \mathrm{Re}}\right)\right] \mathsf{q}(\mathbf{n}, t), \tag{3.24}$$

and using $\mathsf{E}_{\mathcal{F}}\left(\tfrac{\Delta t}{(\Delta x)^2 \mathrm{Re}}\right)$ as an integrating factor for Eq. (3.9). Next, the HERK method is used to integrate the transformed nonlinear equations from t_k to t_{k+1} in order to obtain $\mathsf{v}_{k+1}(\mathbf{n}) \approx \mathsf{v}(\mathbf{n}, t_{k+1})$ and $\mathsf{b}_{k+1} \approx \mathsf{b}(\mathbf{n}, t_{k+1})$. Finally, values for the discrete velocity and total pressure perturbation at t_{k+1}, i.e. $\mathsf{u}_{k+1}(\mathbf{n}) \approx \mathsf{u}(\mathbf{n}, t_{k+1})$ and $\mathsf{d}_{k+1}(\mathbf{n}) \approx \mathsf{d}(\mathbf{n}, t_{k+1})$, are obtained from v_{k+1} and b_{k+1} by using the integrating factor $\mathsf{E}_{\mathcal{F}}\left(\tfrac{-\Delta t}{(\Delta x)^2 \mathrm{Re}}\right)$.

A computationally convenient algorithm for the k-th time-step of the IF-HERK method, subsequently denoted by $(\mathsf{u}_{k+1}, t_{k+1}, \mathsf{p}_{k+1}) \leftarrow \text{IF-HERK}(\mathsf{u}_k, t_k)$, is given by:

1. *initialize*: copy solution values from the k-th time-step,

$$\mathsf{u}_k^0 = \mathsf{u}_k, \quad t_k^0 = t_k. \tag{3.25}$$

2. *multi-stage*: for $i = 1, 2, \ldots, s$, solve the linear system

$$\begin{bmatrix} \left(\mathsf{H}_{\mathcal{F}}^i\right)^{-1} & \mathsf{G} \\ \mathsf{G}^{\dagger} & 0 \end{bmatrix} \begin{bmatrix} \mathsf{u}_k^i \\ \hat{\mathsf{d}}_k^i \end{bmatrix} = \begin{bmatrix} \mathsf{r}_k^i \\ 0 \end{bmatrix}, \tag{3.26}$$

where

$$\mathsf{H}_{\mathcal{F}}^i = \mathsf{E}_{\mathcal{F}}\left(\tfrac{(\tilde{c}_i - \tilde{c}_{i-1})\Delta t}{(\Delta x)^2 \mathrm{Re}}\right), \quad \mathsf{r}_k^i = \mathsf{q}_k^i + \Delta t \sum_{j=1}^{i-1} \tilde{a}_{i,j} \mathsf{w}_k^{i,j} + \mathsf{g}_k^i, \tag{3.27}$$

$$\mathsf{g}_k^i = \; -\tilde{a}_{i,i} \Delta t \; \tilde{\mathsf{N}}\left(\mathsf{u}_k^{i-1} + \mathsf{u}_{\infty}(t_k^{i-1})\right), \quad t_k^i = t_k + \tilde{c}_i \Delta t. \tag{3.28}$$

For $i > 1$ and $j > i$, q_k^i and $\mathsf{w}_k^{i,j}$ are recursively computed using[7]

$$\mathsf{q}_k^i = \mathsf{H}_{\mathcal{F}}^{i-1}\mathsf{q}_k^{i-1}, \quad \mathsf{q}_k^1 = \mathsf{u}_k^0 \tag{3.29}$$

$$\mathsf{w}_k^{i,j} = \mathsf{H}_{\mathcal{F}}^{i-1}\mathsf{w}_k^{i-1,j}, \quad \mathsf{w}_k^{i,i} = (\tilde{a}_{i,i}\Delta t)^{-1}\left(\mathsf{g}_k^i - \mathsf{G}\hat{\mathsf{d}}_k^i\right). \tag{3.30}$$

3. *finalize*: define the solution and constraint values of the $(k+1)$-th time-step,

$$\mathsf{u}_{k+1} = \mathsf{u}_k^s, \quad \mathsf{d}_{k+1} = (\tilde{a}_{s,s}\Delta t)^{-1}\hat{\mathsf{d}}_k^s, \quad t_{k+1} = t_k^s. \tag{3.31}$$

The above algorithm is obtained by applying the HERK method to either Eq. (3.20) or, equivalently, Eq. (3.19) for the k-th sub-problem, and introducing the auxiliary variables

$$\mathsf{u}_k^i(\mathbf{n}) = \left[\mathsf{E}_{\mathcal{F}}\left(\tfrac{-\tilde{c}_i\Delta t}{(\Delta x)^2\mathrm{Re}}\right)\right]\mathsf{v}_k^i(\mathbf{n}), \quad \mathsf{d}_k^i(\mathbf{n}) = \left[\mathsf{E}_{\mathcal{F}}\left(\tfrac{-\tilde{c}_i\Delta t}{(\Delta x)^2\mathrm{Re}}\right)\right]\mathsf{b}_k^i(\mathbf{n}), \tag{3.32}$$

for $i = 1, 2, \ldots s$. Additionally, the intermediate steps used to obtained the final form IF-HERK algorithm make frequent use of the commutativity properties of $\mathsf{E}_{\mathcal{Q}}$ and the identity $\mathsf{E}_{\mathcal{Q}}(\alpha_1)\mathsf{E}_{\mathcal{Q}}(\alpha_2) = \mathsf{E}_{\mathcal{Q}}(\alpha_1 + \alpha_2)$.

The linear operator on the left-hand-side (LHS) of Eq. (3.26) is symmetric positive semi-definite and its null-space is spanned by the set of $[0, \mathsf{a}]^\dagger$, where $\mathsf{a} \in \mathbb{R}^{\mathcal{C}} \times \mathbb{R}$ is any discrete linear polynomial. Consequently, the compatibility condition on the pressure field given by Eq. (3.5) guarantees Eq. (3.26) has a unique solution. As presented, the IF-HERK algorithm is compatible with any HERK scheme since no assumptions have been made on the RK coefficients. Of course, more efficient versions of this algorithm can potentially be obtained for specific families of RK coefficients, but such details are beyond the scope of the present work.

---

[7] An efficient implementation of the IF-HERK algorithm recognizes that the application of $s-1$ integrating factors can be avoided during final, $i = s$, stage by computing $\mathsf{r}_k^s = \mathsf{H}_{\mathcal{F}}^{i-1}\left(\mathsf{q}_k^{s-1} + \Delta t\sum_{j=1}^{i-1}\tilde{a}_{i,j}\mathsf{w}_k^{i-1,j}\right) + \mathsf{g}_k^i$, as opposed to Eq. (3.28). This modification avoids having to explicitly compute q_k^s and $\mathsf{w}_k^{s,j}$ for $j = 1, 2, \ldots s-1$.

The IF-HERK schemes used to performed the numerical experiments of Section 3.6 are given by the following tableaus:

Scheme A

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & \frac{\sqrt{3}}{3} & \frac{3-\sqrt{3}}{3} & 0 \\ \hline & \frac{3+\sqrt{3}}{6} & -\frac{\sqrt{3}}{3} & \frac{3+\sqrt{3}}{6} \end{array},$$

Scheme B

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & 0 & \frac{3}{4} & \frac{1}{4} \end{array},$$

Scheme C

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{8}{15} & \frac{8}{15} & 0 & 0 \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} & 0 \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}. \tag{3.33}$$

The order of accuracy, based on the simplified order-conditions discussed in Section 3.3.1, for each scheme is provided in Table 3.1. As a point of comparison, Table 3.1 also provides the expected order of accuracy for general semi-explicit DAEs of index 2, i.e. Eq. (3.21).

Table 3.1: Expected order of accuracy of the solution y variable (velocity) and the constraint z variable (pressure) of HERK schemes. The superscript $*$ denotes values for general semi-explicit DAEs of index 2.

	y-Order	*z-Order*	*y-Order**	*z-Order**
Scheme A	2	2	2	2
Scheme B	3	2	3	2
Scheme C	3	1	2	1

The tableaus for Schemes B and C were obtained from [66] and [67]. As discussed in [67], the tableau for Scheme C corresponds to the RK coefficients of the popular three-stage fractional step method of [68]. Unlike Schemes B and C, the tableau for Scheme A was specifically defined for the IF-HERK method. An advantage of Scheme A over Schemes B and C is that the RK nodes, c_i's, are equally spaced. As a result, the IF-HERK method only requires a single non-trivial integrating factor.[8] This reduction in the number of distinct LGFs reduces the number of pre-processing operations and lowers the storage requirements of the FLGF method. Additionally, extensions of the present method including immersed surfaces, e.g.

[8] One additional integrating factor is required during the last stage of the IF-HERK algorithm, but for the case of $c_s = 1$ this additional integrating factor reduces to the identify operator.

via the treatment of immersed boundaries of [69], can potentially enjoy similar reductions in the computational costs of pre-processing operations by only having to consider a single non-trivial integrating factor. We will report on immersed boundary methods based on the present flow solver in subsequent publications. The linear stability analysis of the IF-HERK method is provided in Appendix 3.C.

3.3.3 Projection method

It is readily verified that the computationally expensive operation performed by the IF-HERK method corresponds to solving Eq. (3.26) for each stage. Systems of continuum or discrete equations similar to Eq. (3.26) often arise in the literature of numerical methods for simulating incompressible flows. Solutions to these system are frequently obtained through classical projection, fractional-step, or pressure Schur complement methods [70, 71]. These methods can be regarded as approximate block-wise LU decompositions of the original system [70, 71]. More recently, *exact* projection techniques that are free of any matrix/operator approximations have been proposed, e.g. [69, 72]. These techniques have the advantage of not introducing any "splitting errors" and do not require artificial pressure boundary conditions. The present formulation uses an exact projection method to solve Eq. (3.26), but differs from the methods of [69, 72] in that it does not use the null-space of the discrete operators to obtain solutions to the linear system.

The block-wise LU decomposition of the operator in Eq. (3.26) suggests a solution procedure, expressed in the standard correction form, given by:

$$\mathsf{u}^* = \mathsf{H}^i_{\mathcal{F}}\mathsf{r}^i_k \qquad \text{(compute intermediate velocity)} \tag{3.34a}$$

$$\mathsf{S}\hat{\mathsf{d}}^i_k = \mathsf{G}^\dagger\mathsf{u}^* \qquad \text{(solve for total pressure)} \tag{3.34b}$$

$$\mathsf{u}^i_k = \mathsf{u}^* - \mathsf{H}^i_{\mathcal{F}}\mathsf{G}\hat{\mathsf{d}}^i_k \qquad \text{(projection step)}, \tag{3.34c}$$

where $\mathsf{S} = \mathsf{G}^\dagger\mathsf{H}^i_{\mathcal{F}}\mathsf{G}$ is the Schur complement of the system.[9] By taking into account

[9] Without additional information the (scaled) total pressure perturbation, $\hat{\mathsf{d}}^i_k$, obtained from

the commutativity and mimetic properties of the spatial discretization scheme the procedure given by Eq. (3.34) simplifies to:

$$\hat{\mathsf{d}}_k^i = -\mathsf{L}_{\mathcal{C}}^{-1}\mathsf{G}^\dagger \mathsf{r}_k^i, \quad \mathsf{u}_k^i = \mathsf{H}_{\mathcal{F}}^i\left(\mathsf{r}_k^i - \mathsf{G}\hat{\mathsf{d}}_k^i\right), \tag{3.35}$$

where $\mathsf{x} = \mathsf{L}_{\mathcal{C}}^{-1}\mathsf{y}$ is equivalent to solving $\mathsf{L}_{\mathcal{C}}\mathsf{x} = \mathsf{y}$ subject to uniform boundary conditions at infinity. In this form, one of the two integrating factors has been eliminated and the original elliptic problem $\mathsf{G}^\dagger\mathsf{H}_{\mathcal{F}}^i\mathsf{G}\mathsf{x} = \mathsf{y}$ has been replaced by the Poisson problem $\mathsf{L}\mathsf{x} = \mathsf{y}$. Reducing the original discrete elliptic problem to a discrete Poisson problem is of significant practical importance since it permits the use of the FLGF method with known LGF expressions [1]. As will be discussed in Section 3.4, the operation count of our overall algorithm is dominated by the cost of solving for the discrete pressure perturbation; therefore, a projection method that is compatible with fast, robust discrete elliptic solvers greatly facilitates obtaining fast flow solutions.

3.4 Adaptive computational grid

3.4.1 Restricting operations to a finite computational grid

Thus far we have described algorithms for discretizing and computing the incompressible Navier-Stokes equations on unbounded grids. In this section, we present a method for computing solutions, to a prescribed tolerance, using only a finite number of operations. This approximation is accomplished by limiting all operations to a finite computational grid obtained by removing grid cells of the original unbounded grid containing field values that are sufficiently small so as not to significantly affect the evolution of the flow field. As will be demonstrated in the following discussion, the ability of the present method to only track a finite region of the unbounded

---

Eq. (3.34b) is unique up to a discrete linear polynomial. Yet, a unique $\hat{\mathsf{d}}_k^i$ is obtained by taking into account the compatibility condition $\mathsf{p}(\mathbf{n}, t) \to 0$ as $|\mathbf{n}| \to \infty$, i.e. $\hat{\mathsf{d}}_k^i(\mathbf{n}) \to \mathsf{c}_k^i$ as $|\mathbf{n}| \to \infty$ where $\mathsf{c}_k^i = \frac{1}{2}|\mathsf{u}_\infty(t_k^i)|^2$, discussed in Section 3.2.1.

domain is a consequence of the exponential decay of the vorticity at large distances, which is assumed for all flows under consideration.

We first consider the error resulting from neglecting field values outside a finite region when solving the elliptic problems of the IF-HERK method.[10] Using the notation of Section 3.2.2, the solution to the discrete Poisson problem of Eq. (3.35) is given by

$$\hat{\mathsf{d}}(\mathbf{n}) = [\mathsf{G}_{\mathcal{C}}^{\mathsf{L}} * \mathsf{f}](\mathbf{n}), \qquad \mathsf{f}(\mathbf{n}) = [-\mathsf{G}^{\dagger}\mathsf{r}_k^i](\mathbf{n}). \tag{3.36}$$

The source field $\mathsf{G}^{\dagger}\mathsf{r}_k^i$ is a discrete approximation of $\nabla\cdot\boldsymbol{\ell}$ at $t \approx k\Delta t$, where $\boldsymbol{\ell} = \boldsymbol{\omega}\times\mathbf{u}$ is the Lamb vector. It follows from the assumption that $\boldsymbol{\omega}$ is exponentially small at large distances that $\nabla\cdot\boldsymbol{\ell}$ and $\mathsf{G}^{\dagger}\mathsf{r}_k^i$ must also be exponentially small at large distances. As a result, the induced field of Eq. (3.36) is computed to a prescribed tolerance by defining the finite computational domain such that it includes the region where the magnitude of $\mathsf{G}^{\dagger}\mathsf{r}_k^i$ is greater than some positive value.

The action of all operators present in the IF-HERK and projection algorithms, with the exception of $\mathsf{L}_{\mathcal{C}}^{-1}$, are evaluated using only a few local operations. Many of these local operators act on fields that typically decay algebraically, e.g. u and d. As a result, the technique of only tracking regions with non-negligible source terms used for Eq. (3.35) is impractical for most other operations required by the IF-HERK method. Unlike the action of $\mathsf{L}_{\mathcal{C}}^{-1}$, the action of local operators only incurs an error limited to a few cells near the boundary of a finite region if field values of outside the region are ignored, i.e. taken to be zero. Furthermore, repeated applications of local operators only propagate the error into the interior of the region by a few grid cells per application. This type of error is prevented from significantly affecting the solution in the interior by padding the interior with buffer grid cells and by periodically computing ("refreshing") u from the discrete vorticity, $\mathsf{w} = \mathsf{C}\mathsf{u}$, which, like $\mathsf{G}^{\dagger}\mathsf{r}_k^i$, has bounded approximate support. As a result, the approximate support

[10] Field values outside the finite region being tracked are treated as zero.

of both $\mathsf{G}^\dagger \mathsf{r}_k^i$ and w must be contained in the finite computational domain. Bounds for the error resulting from approximating the support of these fields and estimates for the number of time steps that can elapse before the velocity needs to refreshed will be discussed in Sections 3.4.3 and 3.4.4, respectively.

We recall that the discrete velocity perturbation u is subject to the constraint $\mathsf{G}^\dagger \mathsf{u} = 0$ and that the null-space of $\mathsf{G}^\dagger$ is spanned by the image of $\mathsf{C}^\dagger$. As a result, it is possible to express u as

$$\mathsf{u} = \mathsf{C}^\dagger \mathsf{a}, \tag{3.37}$$

where $\mathsf{a} \in \mathbb{R}^{\mathcal{E}}$ can be regarded as the discrete vector potential or streamfunction. Additionally, we require $\mathsf{D}\mathsf{a} = 0$. The discrete vorticity, w, can now be expressed in terms of a as

$$\mathsf{w} = \mathsf{C}\mathsf{C}^\dagger \mathsf{a} = \left(\mathsf{C}\mathsf{C}^\dagger + \mathsf{D}^\dagger \mathsf{D}\right) \mathsf{a} = -\mathsf{L}_{\mathcal{E}} \mathsf{a}. \quad \mathsf{D}\mathsf{w} = 0 \tag{3.38}$$

Finally, Eq. (3.37) and (3.38) provide an expression for u in terms of w,

$$\mathsf{u} = -\mathsf{C}^\dagger \mathsf{L}_{\mathcal{E}}^{-1} \mathsf{w}, \tag{3.39}$$

where $\mathsf{L}_{\mathcal{E}}^{-1}$ imposes zero boundary conditions at infinity.[11] As expected, the expressions relating u, w, and a are analogous to the continuum expressions relating the velocity, vorticity, and streamfunction fields. We emphasize that Eq. (3.37), (3.38), and (3.39) were obtained through the algebraic properties of the discrete operators, as opposed to the discretization of continuum equations.

The present formulation can be cast into an equivalent vorticity formulation simply by taking the discrete curl of Eq. (3.9) and computing u, which is required to evaluate the non-linear term, using Eq. (3.39). This formulation is not pursued since each stage of the IF-HERK would require solving a discrete *vector* Poisson problem, as opposed to a discrete *scalar* Poisson problem, which would in turn roughly

[11] Without further considerations Eq. (3.38) implies that a is unique up to a discrete linear polynomial. Given that w is exponentially small at large distances and that u tends to zero at infinity, it follows that a is unique up to an arbitrary constant taken to be zero.

triple the cost of each stage.[12] The vorticity formulation has the advantage of not having to periodically evaluate Eq. (3.39) to refresh u, but, as will be discussed in Sections 3.4.4, this operation occurs, at most, once per time step. Based on the stability analysis of Appendix 3.C, RK schemes with a minimum of three stages are required to ensure stable solutions. As a result, the primitive variable formulation is approximately 1.5 to 3 times faster than the vorticity formulation. Differences in the errors between the two algebraically-equivalent formulations resulting from the finite tolerances used to compute the FLGF and the adaptive grid algorithms can be used to further distinguish each formulation, but such differences in errors are not considered here since they are expected to be on the order of the prescribed tolerances, which, as will be discussed in Section 3.5, are specified to be much smaller than the discretization errors for practical flows.

3.4.2 Block-structured active computational grid

We now turn our attention to the formal definition of the finite region of the unbounded computational domain tracked by our formulation, which we refer to as the *active* computational domain. Consider partitioning the unbounded staggered Cartesian grid described in Section 3.2 into an infinite set of equally sized blocks arranged on a logically Cartesian grid. The block corresponding to the $\mathbf{n} = (i, j, k)$ location is denoted by $B(\mathbf{n})$ or, equivalently, $B_{i,j,k}$, and the union of all blocks is denoted by D_∞. Each block is defined as a finite staggered Cartesian grid of $n_1^b \times n_2^b \times n_3^b$ cells. We limit our attention to the case in which each block contains the same number of cells in each direction, i.e. $n_i^b = n^b$, but note that the subsequent discussion readily extends to the general case. As a practical consideration, a layer of buffer or ghost grid cells surrounding each block is introduced to facilitate

[12] For the test case of the extremely thin $\delta/R = 0.0125$ vortex ring discussed in Section 6.3, the wall-time ratio of a vector to a scalar discrete Poisson solve is approximately 2.8, which is slightly less than the expected ratio of 3 based on operation count estimates of the FLGF method due to the larger parallel communication costs per problem unknown for the scalar case.

the implementation of the present algorithm.

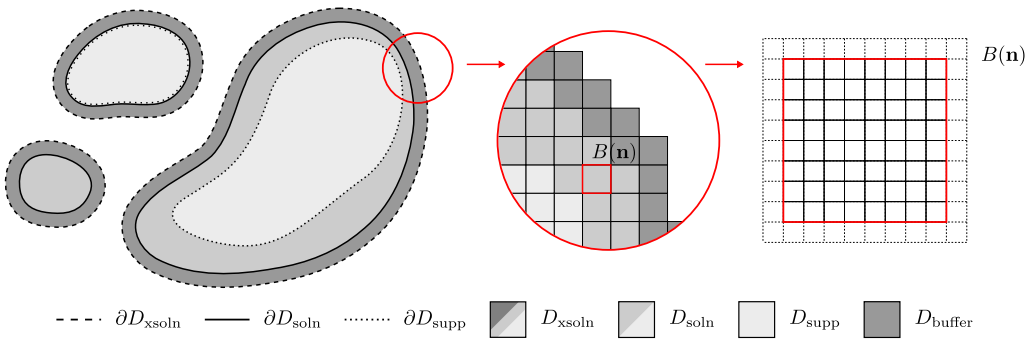

Figure 3.2: Depiction of the finite computational domain in two-dimensions. (*Left*) distant view of the three nested sub-domains $D_{\text{supp}} \subseteq D_{\text{soln}} \subset D_{\text{xsoln}}$ defined in the main text. (*Middle*) zoomed-in view illustrating the union of blocks used to define the domain. (*Right*) magnified view of an individual block. Each block is defined as a finite staggered Cartesian grid; dashed cells surrounding the interior grid correspond to buffer or ghost grid cells.

Figure 3.2 depicts the three nested sub-domains $D_{\text{supp}} \subseteq D_{\text{soln}} \subset D_{\text{xsoln}} \subset D_{\infty}$ that constitute the active computational domain. These sub-domains are defined as:

- *Support blocks* (D_{supp}): union of blocks that defines the support of the source field of the discrete Poisson problems of Eq. (3.35) and (3.38).

- *Solution blocks* (D_{soln}): union of blocks that tracks the solution fields u and d. All field values defined in the blocks belonging to D_{soln} are regarded as accurate approximations of the field values computed using an unbounded domain.

- *Expanded solution blocks* (D_{xsoln}): union of blocks given by a non-trivial neighborhood of D_{soln}. We limit our attention to neighborhoods defined by the union of blocks that are at most N_b blocks away from any block contained in D_{soln},

$$D_{\text{xsoln}} = \left\{ B(\mathbf{m}) : |\mathbf{n} - \mathbf{m}| \leq N_b,\ B(\mathbf{n}) \in D_{\text{soln}},\ \mathbf{m}, \mathbf{n} \in \mathbb{Z}^3 \right\}. \tag{3.40}$$

- *Buffer blocks* (D_{buffer}): union of blocks belonging to D_{xsoln}, but not belonging to D_{soln}, i.e. $D_{\text{buffer}} = D_{\text{xsoln}} \setminus D_{\text{soln}}$. (The domain D_{buffer} is not one of the three primary sub-domains, but it is introduced to facilitate the subsequent discussion.)

The criteria for selecting which blocks belong to D_{supp} and D_{soln} are discussed in Section 3.4.3, and the techniques for selecting values of N_n discussed in Section 3.4.4.

We now introduce the "mask operator" $\mathsf{M}^{\gamma}_{\mathcal{Q}} : \mathbb{R}^{\mathcal{Q}} \mapsto \mathbb{R}^{\mathcal{Q}}$ associated with the grid space $\mathcal{Q}$ and the domain γ, which is defined by

$$[\mathsf{M}^{\gamma}_{\mathcal{Q}}\mathsf{q}](\mathbf{n}) = \begin{cases} \mathsf{q}(\mathbf{n}) & \text{if } \mathbf{n} \in \mathit{ind}[B] \text{ and } B \in D_{\gamma} \\ 0 & \text{otherwise} \end{cases}, \tag{3.41}$$

where $\mathsf{q} \in \mathbb{R}^{\mathcal{Q}}$, and $\mathit{ind}[B]$ denotes the set of all indices of the unbounded staggered grid associated with block B. Mask operators are subsequently used to formally define operations performed on finite domains. For example, the operation Gd perform over D_{xsoln} is defined as $\mathsf{M}^{\text{xsoln}}_{\mathcal{F}}\mathsf{GM}^{\text{xsoln}}_{\mathcal{C}}\mathsf{d}$. For this particular operation, the values of $\mathsf{M}^{\text{xsoln}}_{\mathcal{C}}\mathsf{GM}^{\text{xsoln}}_{\mathcal{C}}\mathsf{d}$ and Gd are equivalent for grid cells in D_{xsoln}, except for a single layer of grid cells on the boundary of D_{xsoln}. Computationally efficient implementations of $\mathsf{M}^{\gamma'}_{\mathcal{Q}'}\mathsf{AM}^{\gamma}_{\mathcal{Q}}$ recognize that all non-trivial numerical operations are limited to grid cells contained in either D_{γ} and $D_{\gamma'}$.

3.4.3 Adaptivity

In this section we discuss the criteria used to select the blocks belonging to D_{supp} and D_{soln}. It follows from subsequent discussions that the field values on $D_{\text{soln}} \setminus D_{\text{supp}}$ can be computed as a post-processing step from the field values on D_{supp}; therefore, only the criteria used to define the D_{supp} affects the accuracy of the computed flow field. We allow for $D_{\text{soln}} \neq D_{\text{supp}}$ in order to emphasize that the present algorithm is able to track values of u and d over arbitrary regions of interest.

Consider a function W that maps an unbounded grid of blocks, i.e. D_∞, to an unbounded grid of positive real scalars. We define the support and solution regions as

$$D_{\text{supp}} = \left\{ B(\mathbf{n}) : [W_{\text{supp}}(D_\infty)](\mathbf{n}) > \epsilon_{\text{supp}}, \quad \mathbf{n} \in \mathbb{Z}^3 \right\}, \tag{3.42a}$$

$$D_{\text{soln}} = \left\{ B(\mathbf{n}) : [W_{\text{soln}}(D_\infty)](\mathbf{n}) > \epsilon_{\text{soln}}, \quad \mathbf{n} \in \mathbb{Z}^3 \right\}, \tag{3.42b}$$

respectively. The functions W_{supp} and W_{soln}, and the scalars ϵ_{supp} and ϵ_{soln} are referred to as weight functions and threshold levels, respectively.

Although the weight function W_{supp} can be defined to reflect any block selection criteria, we limit our attention to cases for which $[W_{\text{supp}}(D_\infty)](\mathbf{n})$ reflects the magnitude of the fields Cu and $\mathsf{G}^\dagger\tilde{\mathsf{N}}(\mathsf{u}+\mathsf{u}_\infty)$ over the block $B(\mathbf{n})$. This choice of W_{supp} facilitates establishing relationships between the threshold level ϵ_{supp} and the error incurred by neglecting source terms values outside D_{supp} when solving the discrete Poisson problems of Eq.(3.35) and (3.38). As a representative example, we consider the weight function W_{supp} given by

$$[W_{\text{supp}}(D_\infty)](\mathbf{n}) = \max\left(\mu(\mathbf{n})/\mu_{\text{global}},\, \nu(\mathbf{n})/\nu_{\text{global}}\right), \tag{3.43a}$$

$$\mu(\mathbf{n}) = \max_{\mathbf{m}\in ind[B(\mathbf{n})]}(|[\mathsf{Cu}](\mathbf{n})|), \qquad \mu_{\text{global}} = \max_{\mathbf{n}\in\mathbb{Z}^3}(\mu(\mathbf{n})), \tag{3.43b}$$

$$\nu(\mathbf{n}) = \max_{\mathbf{m}\in ind[B(\mathbf{n})]}(|[\mathsf{G}^\dagger\tilde{\mathsf{N}}(\mathsf{u}+\mathsf{u}_\infty)](\mathbf{n})|), \quad \nu_{\text{global}} = \max_{\mathbf{n}\in\mathbb{Z}^3}(\nu(\mathbf{n})). \tag{3.43c}$$

In the absence of any error associated with computing the action of $\mathsf{L}_\mathcal{Q}^{-1}$, this expression for W_{supp} results in an upper bound of ϵ_{supp} for the point-wise normalized residual of the active domain approximations of Eq. (3.35) and (3.38).[13] For these cases, the point-wise normalized residual is defined as $\|\mathsf{r}\|_\infty/\|\mathsf{x}\|_\infty$, where

$$\mathsf{r} = \mathsf{x} - \mathsf{M}^{\text{supp}}\mathsf{L}_\mathcal{Q}\mathsf{y}, \quad \mathsf{y} = \mathsf{M}_\mathcal{Q}^{\text{xsoln}}{\mathsf{L}_\mathcal{Q}}^{-1}\mathsf{M}_\mathcal{Q}^{\text{supp}}\mathsf{x}, \tag{3.44}$$

[13] Formally, ϵ_{supp} is only an approximate upper bound for the active domain case of Eq. (3.35) since the source field for this problem is not exactly equal to $-\mathsf{G}^\dagger\tilde{\mathsf{N}}(\mathsf{u}+\mathsf{u}_\infty)$. Yet, for the present error estimates, numerical experiments of representative flows indicate that $-\mathsf{G}^\dagger\tilde{\mathsf{N}}(\mathsf{u}+\mathsf{u}_\infty)$ at $t = t_k$ is a good approximation to $\mathsf{G}^\dagger\mathsf{r}_k^i$ of each stage of the k-th time-step.

and x is the source field of the corresponding discrete Poisson problem.

In general, as the solution changes over time the domain D_{supp}, as defined by Eq. (3.42) and Eq. (3.42a), will also change. Significant amounts of non-negligible source terms are prevented from being advected or diffused outside D_{supp} by recomputing and, if necessary, reinitializing the active domain at the beginning of a time-step. This operation is performed by first computing $\mathsf{w} \leftarrow \mathsf{C}\mathsf{u}$ and $\mathsf{q} \leftarrow -\mathsf{G}^{\dagger}\tilde{\mathsf{N}}(\mathsf{u}+\mathsf{u}_{\infty})$ on D_{xsoln}. Next, values of w and q of grid cells belonging to block in D_{buffer} that have been significantly contaminated by finite boundary errors are zeroed. Finally, $[W_{\text{supp}}(D_{\infty})](\mathbf{n})$ and $[W_{\text{soln}}(D_{\infty})](\mathbf{n})$ are computed using Eq. (3.42a) for all $\mathbf{n} \in \mathbb{Z}^3$ such that $B(\mathbf{n}) \in D_{\text{xsoln}}$ and are set to zero otherwise.

If either of the newly computed D_{supp} or D_{soln} differ from their respective previous values, then it is necessary to reinitialize the active grid and compute the discrete velocity perturbation, u, over the new D_{xsoln}. By construction, all non-negligible values of the discrete vorticity, w, are contained in D_{supp}; therefore, u over D_{xsoln} can be computed as

$$\mathsf{a} \leftarrow -\mathsf{M}_{\mathcal{E}}^{\text{xsoln}}\mathsf{L}_{\mathsf{Q}}{}^{-1}\mathsf{M}_{\mathcal{E}}^{\text{supp}}\mathsf{w}, \quad \mathsf{u} \leftarrow \mathsf{M}_{\mathcal{E}}^{\text{xsoln}}\mathsf{C}^{\dagger}\mathsf{M}_{\mathcal{E}}^{\text{xsoln}}\mathsf{a}. \tag{3.45}$$

Subsequently, we denote the procedure given by Eq. (3.45) as $\mathsf{u} \leftarrow \text{Vor2Vel}(\mathsf{w})$.

We emphasize that the present algorithm is also compatible with other choices of weight functions. Using weight functions that are well-suited for capturing the relevant flow physics of a particular application can potentially reduce the size of the active domain and the number of operations required to accurately simulate the flow. For example, if we are primarily interested in capturing the local physics of a flow over a particular region centered at $\mathbf{x}_0$, then a weight function $|\mathbf{n}-\mathbf{x}_0|^{-\alpha}[W(D_{\infty})](\mathbf{n})$ with $\alpha > 0$ and W given by Eq. (3.42a) might be an appropriate choice. Unless otherwise stated, subsequent discussions assume that W_{supp} is defined by Eq. (3.43a).

3.4.4 Velocity refresh

In this section we present a set of techniques for limiting the error introduced from truncating non-compact fields that decay algebraically, e.g. u and d, when computing the action of local operators. We limit the present discussion to issues that arise from evaluating expressions involving $\mathsf{E}^{\mathsf{L}}_{\mathcal{Q}}(\alpha)$ on the finite active domain since this operator has the largest stencil of all local operators involved in the IF-HERK and projection methods.

We recall that the action of $\mathsf{E}^{\mathsf{L}}_{\mathcal{Q}}(\alpha)$ on $\mathsf{q} \in \mathbb{R}^{\mathcal{Q}}$ is computed as $[\mathsf{G}_{\mathsf{E}}(\alpha) * \mathsf{q}](\mathbf{n})$. Formally, $\mathsf{G}_{\mathsf{E}}(\alpha)$ has an infinite support, but, as discussed in Section 3.2.2, $[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{n})$ decays rapidly as $|\mathbf{n}| \to \infty$; therefore, it is possible to approximate $\mathsf{G}_{\mathsf{E}}(\alpha)$ to prescribed tolerance using a finite support. Consequently, for a given α, there exists some $n_{\mathsf{E}} \in \mathbb{Z}$ such that the field induced from an arbitrary source field can be computed at a distance $n_{\mathsf{E}}\Delta x$ from $\partial D_{\text{xsoln}}$ to a prescribed accuracy ϵ_{E}. By choosing the parameter N_b, used to define D_{xsoln} in Eq. (3.40), to be equal or greater than $\lceil n_{\mathsf{E}}/n^b \rceil$ it is possible to evaluate the action of $\mathsf{E}^{\mathsf{L}}_{\mathcal{Q}}(\alpha)$ on D_{soln} to an accuracy ϵ_{E}. As a result, the flow inside D_{soln} remains an accurate approximation of the flow that would have been obtained using the entire unbounded grid.

As the solution is evolved using the IF-HERK method, the operator $\mathsf{E}^{\mathsf{L}}_{\mathcal{Q}}(\alpha)$ is repeatedly applied to various grid functions, causing the error associated with truncated non-compact source fields to progressively propagate into the interior of D_{xsoln}. The action of $\prod_{i=1}^{n} \mathsf{M}^{\text{xsoln}}_{\mathcal{Q}} \mathsf{E}^{\mathsf{L}}_{\mathcal{Q}}(\alpha_i) \mathsf{M}^{\text{xsoln}}_{\mathcal{Q}}$ is well-approximated by $\mathsf{M}^{\text{xsoln}}_{\mathcal{Q}} \mathsf{E}^{\mathsf{L}}_{\mathcal{Q}}(\beta) \mathsf{M}^{\text{xsoln}}_{\mathcal{Q}}$, where $\beta = \sum_{i=1}^{n} \alpha_i$. Given that the physical values of the nonlinear terms in the IF-HERK algorithm are approximately zero on D_{buffer}, the minimum buffer region required to integrate u over q time-steps is determined by the support of $\mathsf{G}_{\mathsf{E}}(q\beta)$, where $\beta = \sum_{i=1}^{s} \frac{\Delta \tilde{c}_i \Delta t}{(\Delta x)^2 \text{Re}} = \frac{\Delta t}{(\Delta x)^2 \text{Re}}$. A procedure for obtaining estimates for n_{E} from q and β is provided in Appendix 3.D. This procedure is extended to obtain an upper bound, $q_{\max}$, on the number of time-steps, q, before the error at prescribed distance $n_{\mathsf{E}}\Delta x$ away from $\partial D_{\text{xsoln}}$ exceeds a prescribed value of ϵ_{E}. At its minimum,

the depth of the buffer region is $n^b N_b \Delta x$; therefore, the present method takes n_{E} to be equal to $n^b N_b$.

Provided $q_{\max} \geq 1$, the solution is integrated over multiple time-steps before the error from truncating non-compact source field starts to significantly affect the accuracy of the solution on D_{soln}.[14] In order to maintain the prescribed accuracy, after $q_{\max}$ time-steps the discrete velocity perturbation on D_{xsoln} is recomputed or *refreshed* from the discrete vorticity on D_{supp} using the Vor2Vel procedure.

3.5 Algorithm summary

The present method for solving the incompressible Navier-Stokes on formally unbounded Cartesian grids using a finite number of operations and storage, referred to as the NSLGF method, is summarized in this section. Implementation details are omitted since they are beyond the scope of the present work. Instead, we refer the reader to the parallel implementation of the FLGF method [1], which can be readily extended to accommodate the additional operations required by the NSLGF method.

An outline of the steps performed by the NSLGF algorithm at k-th time-step is as follows:

1. *Preliminary*: compute the discrete vorticity, w_k, and divergence of the Lamb vector, q_k.

$$\mathsf{w}_k \leftarrow \mathsf{M}_{\mathcal{E}}^{\text{xsoln}} \mathsf{C} \mathsf{M}_{\mathcal{F}}^{\text{xsoln}} \mathsf{u}_k, \tag{3.46a}$$

$$\mathsf{q}_k \leftarrow -\mathsf{M}_{\mathcal{C}}^{\text{xsoln}} \mathsf{G}^{\dagger} \mathsf{M}_{\mathcal{F}}^{\text{xsoln}} \tilde{\mathsf{N}} (\mathsf{M}_{\mathcal{F}}^{\text{xsoln}} (\mathsf{u}_k + \mathsf{u}_\infty(t_k))). \tag{3.46b}$$

2. *Grid update*: update the computational grid based on prescribed criteria.

[14] Combinations of n^b, N_b, and β resulting in $q_{\max} = 0$ are not allow. For a given β, the value of $q_{\max} = 0$ can always be increased by using larger values of n^b or N_b.

a) *Query*: use weight functions W_{supp} and W_{soln}, threshold values ϵ_{supp} and ϵ_{soln}, and fields w_k and q_k to determine whether D_{supp} or D_{soln} need to be updated.

b) *Update*: (if necessary) update D_{supp}, D_{soln}, and D_{xsoln} by adding or removing blocks. Copy the values of the discrete vorticity from the old to the new computational grid for $\forall B \in D_{\text{supp}}^{\text{new}} \cap D_{\text{supp}}^{\text{old}}$, where $D_{\text{supp}}^{\text{new}}$ and $D_{\text{supp}}^{\text{old}}$ denote D_{supp} before and after the update, respectively.

3. *Velocity refresh*: compute the discrete velocity perturbation, u_k, from the discrete vorticity, w_k.

   a) *Query*: this operation is required if either the grid has been updated or if the number of time-steps since the last refresh is equal or greater than $q_{\max}$.

   b) *Refresh*: (if necessary) compute u_k using:

$$\mathsf{u}_k \leftarrow \text{Vor2Vel}(\mathsf{w}_k), \tag{3.47}$$

   where the Vor2Vel procedure given by Eq. (3.45).

4. *Time integration*: compute u_{k+1}, t_{k+1}, and p_{k+1} using:

$$(\mathsf{u}_{k+1}, t_{k+1}, \mathsf{p}_{k+1}) \leftarrow \text{xIF-HERK}(\mathsf{u}_k, t_k), \tag{3.48}$$

   where the xIF-HERK algorithm is the active computational domain version of the IF-HERK algorithm.

The xIF-HERK algorithm is identical to the IF-HERK algorithm, except for the presence of mask operators which are used to confine all operations to the finite active domain. With the exception of a few special cases, the xIF-HERK algorithm is obtained by operating from the left all operators and grid functions present in the IF-HERK algorithm by the appropriate $\mathsf{M}_{\mathcal{Q}}^{\text{xsoln}}$, e.g. $\mathsf{A} \to \mathsf{M}_{\mathcal{Q}}^{\text{xsoln}}\mathsf{A}$ and $\mathsf{y} \to \mathsf{M}_{\mathcal{Q}}^{\text{xsoln}}\mathsf{y}$.

The exceptions to this rule correspond to the expressions for g_k^i and $\hat{\mathsf{d}}_k^i$, which are given by

$$\mathsf{g}_k^i = \tilde{a}_{i,i}\Delta t \mathsf{M}_{\mathcal{F}}^{\text{soln}} \tilde{\mathsf{N}} \left(\mathsf{M}_{\mathcal{F}}^{\text{xsoln}} (\mathsf{u}_k^{i-1} + \mathsf{u}_\infty(t_k^{i-1})) \right), \tag{3.49a}$$

$$\hat{\mathsf{d}}_k^i = -\mathsf{M}_{\mathcal{C}}^{\text{xsoln}} \mathsf{L}_{\mathcal{C}}^{-1} \mathsf{M}_{\mathcal{C}}^{\text{supp}} \mathsf{G}^\dagger \mathsf{M}_{\mathcal{F}}^{\text{xsoln}} \mathsf{r}_k^i. \tag{3.49b}$$

Both Eq. (3.49a) and (3.49b) reflect the fact that, by construction, the non-negligible physical values of w_k and q_k are contained in W_{supp}.

The operation count for the k-th time-step of the NSLGF method, denoted by N_k^{NSLGF}, is dominated by the number of operations required to evaluate the actions of $\mathsf{L}_{\mathcal{Q}}^{-1}$ and $\mathsf{E}_{\mathcal{Q}}^{\mathsf{L}}$. As a result, an estimate for N_k^{NSLGF} is given by:

$$N_k^{\text{NSLGF}} \approx sN_k^{\mathsf{L}} + 3C(s)N_k^{\mathsf{E}} + \lceil 3N_k^{\mathsf{L}} \rfloor_k, \tag{3.50}$$

where s is the number of stages of the HERK scheme. N_k^{L} and N_k^{E} denote the number of operations required to compute the action of $\mathsf{M}_{\mathcal{Q}}^{\text{xsoln}} \mathsf{L}_{\mathcal{Q}}^{-1} \mathsf{M}_{\mathcal{Q}}^{\text{supp}}$ and $\mathsf{M}_{\mathcal{Q}}^{\text{xsoln}} \mathsf{L}_{\mathcal{Q}}^{-1} \mathsf{M}_{\mathcal{Q}}^{\text{xsoln}}$, respectively, using the FLGF method for scalar grid spaces.[15] Detailed estimates for the values of N_k^{L} and N_k^{E} can be obtained from the discussion of the FLGF method [1], but we note here that both N_k^{L} and N_k^{E} scale as $\mathcal{O}(N)$ for sufficiently large values of N, where N is the total number of grid cells of the active domain. The notation $\lceil \cdot \rfloor_k$ is used to clarify that cost associated with velocity update, i.e. $3N_k^{\mathsf{L}}$, should only be included if a velocity update is performed. Lastly, $C(s)$ specifies the number of integrating factors required by an s-stage IF-HERK scheme. In general, $C(s)$ is equal to $C_0(s)$, where

$$C_0(s) = s + \left\lfloor \frac{(s-1)s}{2} \right\rfloor. \tag{3.51}$$

For special case of second-order IF-HERK schemes, $C(s)$ reduces to $C_0(s) - 1$.[16]

---

[15] The factor of 3 that appears in the second and third terms of Eq. (3.50) accounts for the additional operations required to solve vector Poisson problems and vector integrating factors.

[16] The expression $c_s = 1$ is one of the HERK order-conditions associated with second-order accurate constraints. For the case of $c_s = \tilde{c}_{s-1} = 1$, the integrating factor $\mathsf{H}_{\mathcal{F}}^s$, defined by Eq. (3.27), simplifies to the identity operator.

For convenience, a summary of the parameters used in our treatment of the active computational domain is provided by Table 3.2. Of the parameters listed in Table 3.2, only ϵ_{FLGF}, ϵ_{E}, and ϵ_{supp} affect the accuracy of the numerical simulation. The *solution error* of the NSLGF method, i.e. the error associated with approximately solving the fully discretized unbounded grid equations, is approximately bounded above by the sum of these three parameters.

Table 3.2: Finite computational domain parameters used by the NSLGF method.

Symbol	*Description*	*Section*
N_b	Width of W_{buffer} (no. blocks)	3.4.1
n^b	Block size (no. cells)	3.4.1
ϵ_{FLGF}	FLGF method tolerance	3.2.2
ϵ_{supp}	Support region threshold	3.4.3
ϵ_{E}	Buffer region tolerance	3.4.4

The field values used to compute D_{supp} should represent field values that would be obtained using the unbounded grid in the absence of numerical errors associated with the evaluation of discrete operators. Spurious and unnecessary changes to the active domain are avoided by requiring

$$\max(\epsilon_{\text{FLGF}}, \epsilon_{\mathsf{E}}) < \alpha\epsilon_{\text{supp}}, \tag{3.52}$$

where $\alpha < 1$ is a safety parameter specifying the sensitivity of the adaptive scheme to the solution errors associated with ϵ_{FLGF} and ϵ_{E}.[17] Furthermore, using parameters that satisfy Eq. (3.52) eliminates the inclusion of blocks that only contain field values that are on the same order as the solution error.

The values for n^b and N_b can also significantly affect the number of numerical operations performed by the NSLGF method. Smaller values of n^b typically result in smaller active domains, but require more frequent velocity updates and often

[17] Numerical experiments of representative flows have shown that $\alpha \approx 0.1$ is sufficiently small as to avoid most spurious and unnecessary changes to the computational grid.

require the use of FLGF schemes with less than optimal computational rates. In practice, computationally efficient schemes are obtained by setting $N_b = 1$ and determining the lower bound for n^b, denoted by n_0^b, from the prescribed value of ϵ_{E}. Next, starting from n_0^b, progressively larger values of n^b are considered until an efficient FLGF scheme that achieves the prescribed ϵ_{FLGF} tolerance is obtained. The construction and computational performance of FLGF schemes are discussed in [1].

3.6 Verification examples

The behavior of the NSLGF method is verified through numerical simulations of thin vortex rings. We consider vortex rings of ring-radius R and core-radius δ, with circulation Γ and Reynolds number $\text{Re} = \frac{\Gamma}{\nu}$, where ν is the kinematic viscosity of the fluid. Unless otherwise stated, simulations are initiated with a vorticity distribution given by

$$\omega_\theta(r,z) = \frac{\Gamma}{\pi\delta^2}\exp\left(\frac{z^2+(r-R)^2}{\delta^2}\right), \quad \omega_z(r,z) = 0, \tag{3.53}$$

where $r = x^2 + y^2$ and $\theta = \tan^{-1}(y/x)$. As a result, the vortex ring initially translates in the positive z-direction due to its self-induced velocity [55].

The numerical experiments discussed in this section are initialized by first specifying an initial discrete vorticity, w_0, and then using Eq. (3.45) to obtain an initial discrete velocity perturbation, u_0. This procedure naturally leads to a u_0 that is compatible with the IF-HERK method, i.e. $\mathsf{G}^\dagger\mathsf{u}_0 = 0$. The initial active domain is chosen such that the $|\boldsymbol{\omega}| < 10^{-10}$ outside the D_{supp}. In order to avoid significant numerical artifacts due to the jump in the direction of the vorticity field at the ring origin, we limit our attention to vortex rings for which $|\boldsymbol{\omega}_{\text{center}}| < 10^{-10}\max|\boldsymbol{\omega}|$, where $\boldsymbol{\omega}_{\text{center}}$ is the value of $\boldsymbol{\omega}$ at the center of the ring. For the case of Eq. (3.53), this condition is satisfied for $\delta/R < 0.2$.

Provided a sufficiently large initial active domain, any sufficiently accurate process for computing w_0 from $\boldsymbol{\omega}_0$ can be used to initialize the numerical simulations. Yet

it is convenient to use a process that naturally leads to a w_0 such that $\mathsf{D}\mathsf{w}_0 \approx 0$. In the absence of any numerical errors, $\tilde{\mathsf{w}}_0 = \mathsf{C}\mathsf{u}_0$ is equal to w_0 if and only if $\mathsf{D}\mathsf{w}_0 = 0$. For the case of $\mathsf{D}\mathsf{w}_0 \neq 0$, the support of $\tilde{\mathsf{w}}_0$ is typically larger than the support of w_0, which in turn leads to larger active domains and complicates initial error estimates, i.e. $|\mathsf{w}_0| < \epsilon$ in D_{supp} does not imply $|\tilde{\mathsf{w}}_0| < \epsilon$ in D_{supp}. Provided $\nabla \cdot \boldsymbol{\omega} = 0$, it is possible to construct w_0 such that the magnitude of $\mathsf{D}\mathsf{w}_0$ is less than a prescribed tolerance by computing approximate values of the vorticity flux over the faces of the dual grid and applying the Divergence theorem to each dual cell.[18] For all test cases, a high-order quadrature scheme is used to integrate the initial vorticity distribution over the faces of the dual grid such that the resulting w_0 satisfies $\|\mathsf{D}\mathsf{w}_0\|_\infty \approx 10^{-10}$.

Test cases are performed using $n^b = 16$ and $N_b = 1$. This choice of parameters leads to $\epsilon_{\text{FLGF}} < 10^{-8}$ for all values of Δx, Δt, and Re considered. The values of ϵ_{supp} and ϵ_{E} are taken to be $\epsilon_{\text{supp}} = 0.1\epsilon^*$ and $\epsilon_{\mathsf{E}} = \epsilon^*$. The value of ϵ^* is varied across different sets of simulations, but is always such that $10^{-8} \leq \epsilon^* \leq 10^{-2}$. The support domain D_{supp} is computed using Eq. (3.42a) and Eq. (3.43a), and the solution domain D_{soln} is set to be equal to D_{supp}. It follows from our choice of parameters that the overall solution error is always bounded above by ϵ^*.[19]

With the exception of a few test cases discussed in Section 3.6.1, all numerical experiments are performed using the IF-HERK scheme denoted as "Scheme A" in Section 3.3.2. The time-step size, Δt, is held fixed during each simulation and chosen such that the CFL, based on the maximum point-wise velocity magnitude, does not exceed 0.75. Unless otherwise stated, the freestream velocity, u_∞, is set to be zero.

---

[18] The dual grid corresponds to a copy of the original staggered grid that has been shifted by half a grid cell in each direction. Cells, faces, edges, and vertices of the original grid can be regarded as vertices, edges, faces, and cells, respectively, of the dual grid.

[19] The *solution error*, as defined in Section 3.5, should not be confused with the error of the solution.

3.6.1 Discretization error

The order of accuracy of the discretization techniques is verified using spatial and temporal refinement studies on the early evolution of vortex rings at $\mathrm{Re}_0 = 1{,}000$ with initial vorticity distributions given by

$$\omega_\theta(r,z) = \begin{cases} \alpha \frac{\Gamma}{R^2} \exp\left(-4s^2/(R^2-s^2)\right) & \text{if } s \le R \\ 0 & \text{otherwise} \end{cases}, \quad \omega_z(r,z) = 0, \tag{3.54}$$

where $s^2 = z^2 + (r-R)^2$ and α is chosen such that ω_θ integrates to Γ, i.e. $\alpha \simeq 0.54857674$.[20] Test cases are performed using fixed grids that are sufficiently large such that at any time-step of the simulation the active domain corresponds to a value of ϵ^* less than 10^{-8}.

We use $\varepsilon_{\mathbf{u}} = \|\mathsf{u} - \mathsf{T}_{\mathcal{F}}\mathsf{u}^*\|_\infty / \|\mathsf{u}^*\|_\infty$ and $\varepsilon_{\mathbf{p}} = \|\mathsf{p} - \mathsf{T}_{\mathcal{C}}\mathsf{p}^*\|_\infty / \|\mathsf{p}^*\|_\infty$ to approximate the error at time T of the velocity field, u, and the pressure field, p, respectively. The superscript * is used to denote grid functions obtained from the test case with the highest resolution, i.e. smallest Δx or Δt, included in the corresponding refinement study. Point-wise comparisons between grid functions at different refinement levels are made possible through the use of the coarsening operators $\mathsf{T}_{\mathcal{F}}$ and $\mathsf{T}_{\mathcal{C}}$. Finally, we define $\|\mathsf{x}\|_\infty$ as the maximum value of $|\mathsf{x}(\mathbf{n})|$ for all $\mathbf{n}$ associated with grid locations in D_{soln}.

The spatial refinement study consists of seven test cases corresponding to $\Delta x/\Delta x_0 = 2^0, 2^{-1}, \ldots, 2^{-6}$. Test cases are performed using the same Δt, and $\varepsilon_{\mathbf{u}}$ and $\varepsilon_{\mathbf{p}}$ are evaluated at $T = 10\Delta t$. The computational grids are constructed such that the location of vertices of coarser grids always coincide with the location of vertices of finer grids. This enables the coarsened solution fields $T_{\mathcal{C}}\mathsf{p}^*$ and $T_{\mathcal{F}}\mathsf{u}^*$ to be computed by recursively averaging the values of the 8 (4) fine grid cells (faces) occupying the same physical region as the corresponding coarse grid cell (face). The slope of the

[20] The computational cost of the spatial convergence tests are reduced by using "fat" vortex rings such as those given by Eq. (3.54), which, unlike similar "fat" rings given by Eq. (3.53), are continuous and differentiable at the origin.

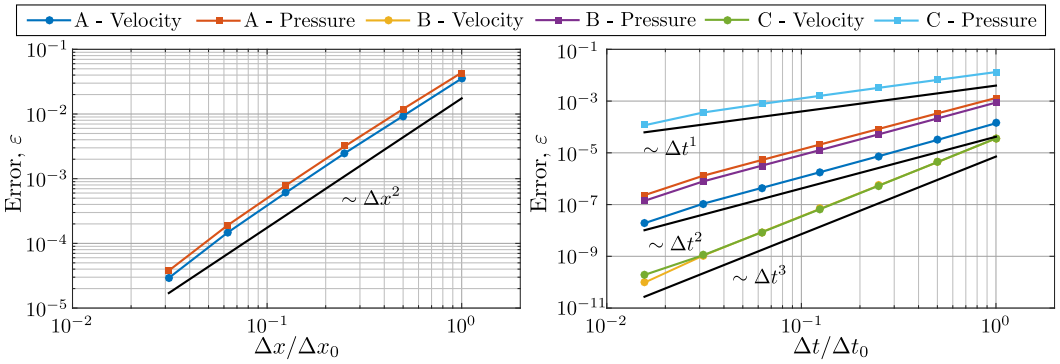

Figure 3.3: Velocity error, $\varepsilon_{\mathbf{u}}$, and pressure error, ε_p, for test cases. Spatial refinement study verifies second-order accuracy of the spatial discretization technique (*left*). Temporal refinement studies verify the expected order of accuracy of the three time integration schemes defined in Section 3.3.2 (*right*).

error curves depicted in the left plot of Figure 3.3 verifies that the solutions are second-order accurate in Δx.

Temporal refinement studies are performed for the three IF-HERK schemes, Scheme A–C, included in Section 3.3.2. For each scheme, a series of eight test cases is performed using $\Delta t/\Delta t_0 = 2^0, 2^{-1}, \ldots, 2^{-7}$. All test cases employ the same computational grid, and $\varepsilon_{\mathbf{u}}$ and $\varepsilon_{\mathbf{p}}$ are evaluated at $T = 10\Delta t_0$. Consequently, $\mathsf{T}_{\mathcal{F}}$ and $\mathsf{T}_{\mathcal{C}}$ are taken to be identity operators. The slopes of the error curves depicted in the right plot of Figure 3.3 verify that the accuracy with respect to Δt of each scheme is the same as the order of accuracy expected from the IF-HERK order-conditions.[21]

3.6.2 Quality metrics for thin vortex rings

In this section we consider the laminar evolution of a thin vortex ring at $\mathrm{Re}_0 = 7{,}500$ initiated with $\delta_0/R_0 = 0.2$. Six test cases for different values of Δx and Δt are performed. The ratio $\Delta t/\Delta x = 0.5734 R_0/\Gamma_0$ is held constant across all test cases.

[21] We note that the spatial discretization error associated with the computational grid is significantly larger than the temporal discretization error for some test cases. This does not affect the present refinement studies since the spatial discretization error is the same for all test cases and our error estimates are computed as the difference of two numerical solutions.

Unlike the numerical experiments of Section 3.6.1, the grid is allowed to freely adapt as the solution evolves. For all test cases, ϵ^* is taken to be 10^{-6}, which is significantly smaller than the discretization error inferred from the discussion of Section 3.6.1.

The evolution of isolated vortex rings is often characterized by the time-history of a few fundamental volume integrals. Quantities considered in the following numerical experiments include the hydrodynamic impulse $\boldsymbol{\mathcal{I}}$, the kinetic energy $\mathcal{K}$, enstrophy $\mathcal{E}$, the helicity $\mathcal{J}$, the Saffman-centroid $\boldsymbol{\mathcal{X}}$, and the ring-velocity $\boldsymbol{\mathcal{U}}$. Expressions for these quantities for unbounded fluid domains and exponentially decaying $\boldsymbol{\omega}$ fields are given by [55]:

$$\begin{aligned}
\boldsymbol{\mathcal{I}}(t) &= \frac{1}{2}\int_{\mathbb{R}^3} \mathbf{x}\times\boldsymbol{\omega}\, d\mathbf{x}, & \mathcal{J}(t) &= \int_{\mathbb{R}^3} \mathbf{u}\cdot\boldsymbol{\omega}\, d\mathbf{x},\\
\mathcal{K}(t) &= \int_{\mathbb{R}^3} \mathbf{u}\cdot(\mathbf{x}\times\boldsymbol{\omega})\, d\mathbf{x}, & \boldsymbol{\mathcal{X}}(t) &= \frac{1}{2}\int_{\mathbb{R}^3} \frac{(\mathbf{x}\times\boldsymbol{\omega})\cdot\boldsymbol{\mathcal{I}}}{|\boldsymbol{\mathcal{I}}|^2}\mathbf{x}\, d\mathbf{x} - \int_0^t \mathbf{u}_\infty(t')\, dt'\\
\mathcal{E}(t) &= \frac{1}{2}\int_{\mathbb{R}^3} |\boldsymbol{\omega}|^2\, d\mathbf{x}, & \boldsymbol{\mathcal{U}}(t) &= \frac{d\boldsymbol{\mathcal{X}}}{dt}.
\end{aligned} \tag{3.55}$$

The hydrodynamic impulse, $\boldsymbol{\mathcal{I}}$, is a conserved quantity in the absence of non-conservative forces [55]. As a result, $\boldsymbol{\mathcal{I}}$ provides a useful metric for assessing the accuracy and physical fidelity of numerical solutions. The time rate of change of $\mathcal{K}$ is related to $\mathcal{E}$ by the relationship $\frac{d}{dt}\mathcal{K} = -2\nu\mathcal{E}$. Differences in the time history of $\frac{d}{dt}\mathcal{K}$ between different numerical simulations of the same flow are commonly used to characterize the accuracy of solutions of unsteady flows [73–75]. In the absence of viscosity, the helicity, $\mathcal{J}$, is an invariant of the flow and provides a measure for the degree of linkage of the vortex lines of the flow [76]. Although the present simulations consider viscous flows, differences in $\mathcal{J}$ between test cases of the same flow are used as part of our quality metrics. Our definitions for the vortex ring centroid, $\boldsymbol{\mathcal{X}}$, and propagation velocity, $\boldsymbol{\mathcal{U}}$, are equivalent to those used by Saffman [55, 77]. Although all the integrals of Eq. (3.55) are formally over $\mathbb{R}^3$, they can be accurately computed for solutions obtained by the NSLGF method since the support of the integrands is

approximately contained in D_{soln}.[22]

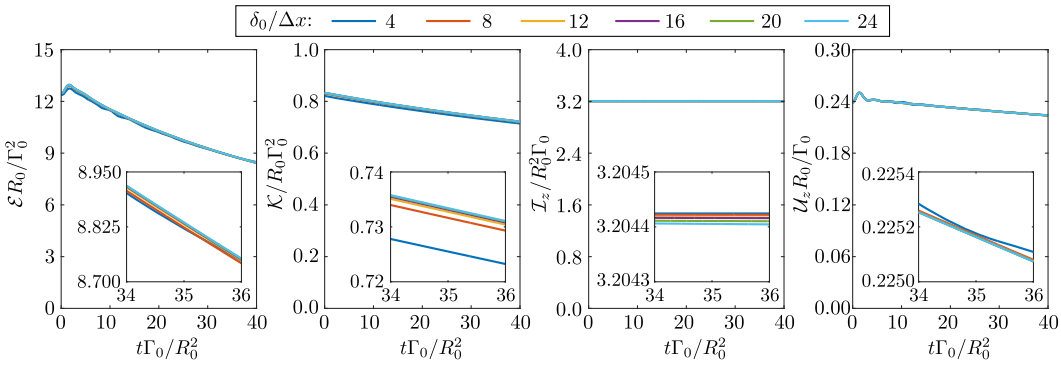

Figure 3.4: Time histories of $\mathcal{E}$, $\mathcal{K}$, $\mathcal{I}_z$, and $\mathcal{U}_z$ (*respectively, left to right*) for a vortex ring at $\text{Re}_0 = 7{,}500$ initiated with $\delta_0/R_0 = 0.2$. Numerical experiments are performed using different values of $\delta_0/\Delta x$ while holding $\Delta t/\Delta x$ constant.

Table 3.3: Maximum difference in $\mathcal{E}$, $\mathcal{K}$, $\mathcal{I}_z$, and $\mathcal{U}_z$ during $t\Gamma_0/R_0^2 \in [0, 40]$ between test cases with $\delta_0/\Delta x < 24$ and the test case with $\delta_0/\Delta x = 24$. Listed differences have been normalized by the maximum value of the respective quantity during $t\Gamma_0/R_0^2 \in [0, 40]$.

$\delta_0/\Delta x$	$\mathcal{E}$	$\mathcal{K}$	$\mathcal{I}_z$	$\mathcal{U}_z$
4	1.8×10^{-2}	1.5×10^{-2}	7.5×10^{-6}	4.9×10^{-3}
8	4.0×10^{-3}	3.5×10^{-3}	6.6×10^{-6}	4.8×10^{-4}
12	1.5×10^{-3}	1.3×10^{-3}	4.8×10^{-6}	1.7×10^{-4}
16	6.0×10^{-4}	5.3×10^{-4}	4.4×10^{-6}	7.3×10^{-5}
20	2.0×10^{-4}	2.2×10^{-4}	2.3×10^{-6}	2.7×10^{-5}

The time history for the values of $\mathcal{E}$, $\mathcal{K}$, $\mathcal{I}_z$, and $\mathcal{U}_z$, where subscripts "q" denotes the component of a vector quantity in q-th direction, are shown in Figure 3.4. The values for $\mathcal{J}$ and the components of $\boldsymbol{\mathcal{I}}$ and $\boldsymbol{\mathcal{U}}$ in the x- and y-directions were also computed, but are not depicted since the magnitude of these values remained less than 10^{-8}, which is significantly smaller than ϵ^*, for all test cases. Visual inspection of the curves included in Figure 3.4 suggests good agreement between all tests cases. This is quantified by Table 3.3, which lists the maximum difference between test cases

[22] Numerical solutions set the vorticity outside the computational to be zero. As a result, the only error involved in evaluating the integrals of Eq. (3.55) is the error resulting from their discretization.

with $\delta_0/\Delta x < 24$ and the test case with $\delta_0/\Delta x = 24$.

Figure 3.4 demonstrates that $\mathcal{E}$, $\mathcal{K}$, and $\mathcal{U}_z$ are most sensitive to changes in the resolution at early times, $t\Gamma_0/R_0^2 \in [0, 15]$. We attribute this to the rapid changes in the vorticity distribution observed shortly after the ring is initiated. For cases initiated with finite values of δ/R, it is well-known that vortex rings undergo an "equilibration" phase shortly after being initiated [73, 74, 78].[23] During this phase, vorticity starts to be shed into the wake and, over time, the core region of the ring assumes a more relaxed axisymmetric vorticity distribution in which ω_θ is no longer symmetric, but instead skewed so as to concentrate the vorticity away from the ring center. After the equilibration phase, i.e. approximately after $t\Gamma_0/R_0^2 > 15$ for test cases under consideration, the ring assumes a quasi-steady distribution that persists until the growth of linear instabilities causes the ring to transition into turbulence. This transition does not occur during the simulation time of the present study, but will be investigated in Section 3.6.4.

For each test case, the value of $\mathcal{I}$ remained nearly constant throughout the simulation time, only exhibiting deviations on the same order as ϵ^* (taken to be 10^{-6} for all test cases). Interestingly, the value $\mathcal{I}$ appears to be insensitive to changes in Δx, at least when maintaining $\Delta t/\Delta x$ constant, as demonstrated by Table 3.3. We refrain from speculating on whether the present method results in additional conservation properties beyond those mentioned in Section 3.2.1, since such investigations are beyond the scope of the present work. Instead, we simply note that $\mathcal{I}$ appears to be conserved approximately up to the solution error, i.e. ϵ^*, which further verifies the physical fidelity of solutions obtained using the NSLGF method.

The difference between the LHS and RHS of $\frac{d}{dt}\mathcal{K} = -2\nu\mathcal{E}$ is often used as a metric for the spatial discretization error. The maximum value of $\left|\frac{d}{dt}\mathcal{K} - (-2\nu\mathcal{E})\right| / (2\nu\mathcal{E})$

[23] Vortex rings initiated with vorticity distributions given by Eq. (3.53) are only solutions to the Navier-Stokes equations in the limit of $\delta/R \to 0$.

for $t\Gamma_0/R_0^2 \in [0, 40]$ is 6.8×10^{-2}, 2.1×10^{-2}, 9.6×10^{-3}, 5.3×10^{-3}, 3.4×10^{-3}, and 2.3×10^{-3} for the tests cases considered, sorted in ascending order of $\delta_0/\Delta x$. Values for $\frac{d\mathcal{K}}{dt}$ and $2\nu\mathcal{E}$ were computed at each half-time step using standard second-order differencing and averaging, respectively.

3.6.3 Propagation speed of thin vortex rings

The results of this section verify that the solutions obtained using the NSLGF method are indeed physical solutions to the incompressible Navier-Stokes equations. The translational speed of laminar vortex rings has been extensively studied through experimental, numerical, and theoretical investigations [55, 73, 79–81]. Saffman [77] showed that the propagation speed of viscous vortex rings with a vorticity distributions given by Eq. (3.53), in the limit of $\delta/R \to 0$, is

$$U_{\text{Saffman}} = \frac{\Gamma_0}{4\pi R_0}\left[\log\left(\frac{8}{\varepsilon}\right) - \beta_0 + \mathcal{O}\left(\varepsilon \log \varepsilon\right)\right], \tag{3.56}$$

where $\varepsilon = \delta/R$, $\beta_0 = \frac{1}{2}\left(1 - \gamma + \log 2\right) \simeq 0.557966$, and $\gamma \simeq 0.577216$ is Euler's constant. Subsequent numerical [73] and theoretical [82] investigations have shown that the error term is actually smaller, and is given by $\mathcal{O}\left(\varepsilon^2 \log \varepsilon\right)$.

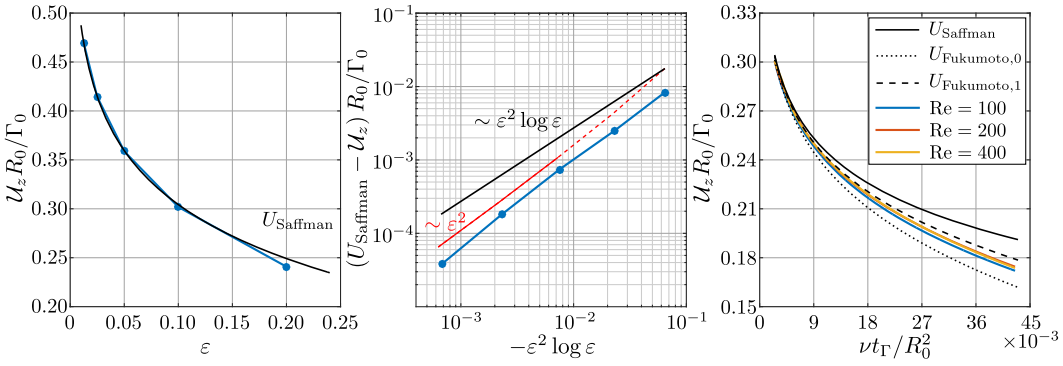

Figure 3.5: Propagation speed of thin vortex rings at $\text{Re}_0 = 7{,}500$ for the different values of $\varepsilon = \delta_0/R_0$ (*left*). Difference between the computed values at $\text{Re}_0 = 7{,}500$, $\mathcal{U}_z$, and the theoretical estimates, U_{Saffman}, for propagation speed of vortex rings (*middle*). Time history of the propagation speed of vortex rings initiated with $\delta_0/R_0 = 0.1$ at different Re (*right*).

The initial propagation speed of vortex rings, taken to be $\mathcal{U}_z$ as defined in Eq. (3.55), is computed for test cases at $\mathrm{Re}_0 = 7{,}500$ that have been initiated with $\varepsilon = 0.2$, 0.1, 0.05, 0.025, and 0.0125. For all test cases, $\delta_0/\Delta x = 20$, $\Delta t \Gamma_0 / R_0^2 = 10^{-6}$ and $\epsilon^* = 10^{-6}$. Values of $\mathcal{U}_z$ are computed via central differencing the values of $\boldsymbol{\mathcal{X}}$ between adjacent time-steps. The value $\mathcal{U}_z$ at $t^* = \Delta t/2$ for each test case is shown in the left plot of Figure 3.5. Visual inspection indicates good agreement between $\mathcal{U}_z$ and U_{Saffman}, which in turn verifies that numerical solutions obtained by the NSLGF method approximate actual physical solutions.

We further verify the present formulation by confirming the form of the error term of U_{Saffman}, i.e. $\mathcal{O}\left(\varepsilon^2 \log \varepsilon\right)$. Theoretical estimates for the effective ring and core radii for early times[24] indicate that, at time t^*, the ring and core size have not deviated enough from their initial values to significantly affect the value U_{Saffman} as to hinder the present comparison. The middle plot of Figure 3.5 shows the difference in the ring propagation speed between the numerical experiments, $\mathcal{U}_z$, and theoretical estimates, U_{Saffman}. For large values of ε, i.e. $\varepsilon > 0.05$, the rate of change of $\Delta\tilde{U}_z = \left(U_{\text{Saffman}} - \mathcal{U}_z\right) R_0/\Gamma_0$ with respect to ε is consistent with the theoretical $\mathcal{O}(\varepsilon^2 \log \varepsilon)$ error estimate. On the other hand, for $\varepsilon < 0.05$ the observed rate of change of $\Delta\tilde{U}_z$ with respect to ε suggests the error term in Eq. (3.56) is closer to $\mathcal{O}(\varepsilon^2)$ than $\mathcal{O}(\varepsilon^2 \log \varepsilon)$. We refrain from attributing any physical meaning to the difference in the behavior of the error at smaller values of ε since we have not thoroughly determined the numerical error for such test cases.[25]

We further verify the present implementation by comparing the time and Reynolds number dependence of $\mathcal{U}_z$ with previously reported theoretical [81] and numerical

[24] The radius of the core and the vorticity centroid in the radial direction are approximately $2\sqrt{\nu t}$ and $R_0 + 3\nu t/R_0$ at $\sqrt{\nu t} \ll R_0$ [81].

[25] Extrapolating from the results of Table 3.3 to the present tests cases, we estimate that the error of $\mathcal{U}_z$ to be between 10^{-5} and 10^{-4}. As a result, the assumption that $\mathcal{U}_z$ is more accurate than U_{Saffman} might need to be revisited for test cases resulting in values of $\Delta\tilde{U}_z < 10^{-4}$.

[73] results. To facilitate the comparisons, it is convenient to define

$$t_\Gamma = \frac{\delta_0^2}{4\nu} + t. \tag{3.57}$$

The discussion of [81] provides theoretical bounds on $\mathcal{U}_z$ of vortex rings initiated with $\delta/R \to 0$. In the low-Re limit $\mathcal{U}_z$ tends to

$$U_{\text{Fukumoto},0} = \frac{\Gamma_0}{4\pi R_0}\left[\log\left(\frac{8}{\eta}\right) - \beta_0 - \frac{9}{5}\left(\log\left(\frac{8}{\eta}\right) - \beta_1\right)\left(\frac{\eta}{2}\right)^2\right], \tag{3.58}$$

and in the high-Re limit $\mathcal{U}_z$ tends to

$$U_{\text{Fukumoto},1} = \frac{\Gamma_0}{4\pi R_0}\left[\log\left(\frac{8}{\eta}\right) - \beta_0 - \beta_2\left(\frac{\eta}{2}\right)^2\right], \tag{3.59}$$

where $\eta = 2\sqrt{\nu t_\Gamma}/R_0$, β_0 is the same as in Eq. (3.56), $\beta_1 \simeq 1.057967$, and $\beta_2 \simeq 3.671591$. For all test cases, $\delta_0/\Delta x = 15$ and Δt is determined by requiring the initial CFL to be 0.5. Test cases correspond to vortex rings at $\text{Re}_0 = 100$, 200, and 400 that are initiated with $\delta_0/R_0 = 0.1$. The right plot of Figure 3.5 demonstrates that, for all test cases, $\mathcal{U}_z$ remains bounded between $U_{\text{Fukumoto},0}$ and $U_{\text{Fukumoto},1}$, except at early times for the case of $\text{Re}_0 = 400$ where the numerical $\mathcal{U}_z$ slightly exceeds the $U_{\text{Fukumoto},1}$. This discrepancy is not surprising since the theory of Fukumoto [81] assumes that the vortex ring is initiated with $\delta/R \to 0$, and, as a result, does not properly account for the changes in the vorticity distribution that occur during the equilibration phase of vortex rings initiated with finite δ/R. Although not shown in Figure 3.5, the time history of $\mathcal{U}_z$ for all test cases has been compared to the numerical results of [73], and found to be in good agreement (overlaying the curves of both investigations reveal nearly identical results).

3.6.4 Finite active computational domain error

In this section, we investigate the effect that our adaptive grid technique has on the numerical solutions by considering the evolution of thin vortex rings computed using different values of ϵ^*. These test cases are used to verify that the solutions converge

as ϵ^* tends to zero and to verify, via comparisons with numerical investigations of other authors, the physical fidelity of the solutions.

For all test cases, the vortex ring is initiated with $\delta_0/R_0 = 0.2$ and a constant uniform flow, $\mathsf{u}_\infty = \left[0, 0, u_\infty^{(z)}\right]$, is superimposed to partially oppose the translational motion of the vortex ring. The value of $u_\infty^{(z)} R_0/\Gamma_0$ is taken to be -0.18686, which reduces the initial speed of the vortex ring by approximately 75%. Solutions are computed using $\delta_0/\Delta x = 10$ and $\Delta t \Gamma_0/R_0^2 \approx 0.01721$. The error estimates of Section 3.6.2 indicate that, for all test cases, the discretization error is on the order of 10^{-3}.

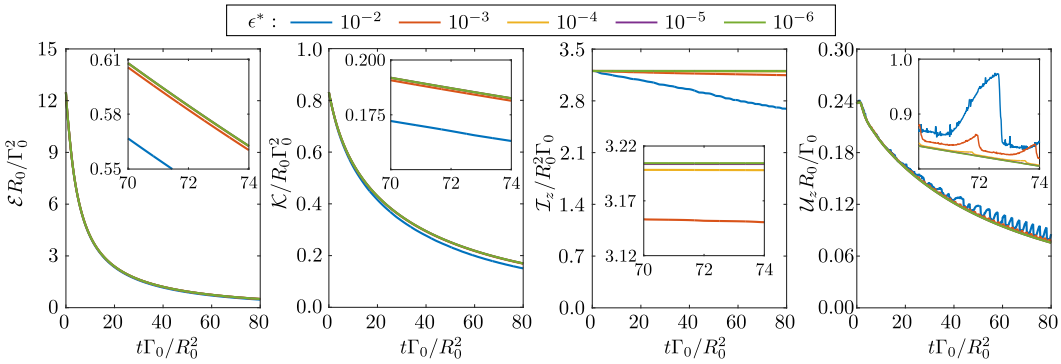

Figure 3.6: Time histories of $\mathcal{E}$, $\mathcal{K}$, $\mathcal{I}_z$, and $\mathcal{U}_z$ (*respectively, left to right*) for a vortex ring at $\mathrm{Re}_0 = 500$ initiated with $\delta_0/R_0 = 0.2$. All parameters, with the exception of ϵ^*, are held constant across all test cases.

Figure 3.6 depicts the time histories of $\mathcal{E}$, $\mathcal{K}$, $\mathcal{I}_z$, and $\mathcal{U}_z$ for a vortex ring at $\mathrm{Re} = 500$ computed using $\epsilon^* = 10^{-2}$, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}. The smooth decay of $\mathcal{E}$ and $\mathcal{K}$ indicates that the vortex ring remains laminar throughout the entire simulation time. This follows from the fact that a pronounced peak in $\mathcal{E}$ is observed during the transition to the early stages of turbulence resulting from a significant increase in the stretching of vortex filaments [74]. Figure 3.6 verifies that, for laminar flows, numerical solutions converge as ϵ^* tends to zero. For all test cases with values of $\epsilon^* > 10^{-2}$, the error[26] in the computed values $\mathcal{E}$, $\mathcal{K}$ and $\mathcal{I}_z$ is inversely proportional

[26] The error is estimated by assuming that the test case corresponding to $\epsilon^* = 10^{-6}$ is the true solution.

to ϵ^* for $t\Gamma_0/R_0^2 \in [10, 80]$. The large oscillations in $\mathcal{U}_z$ are due to shifts in $\boldsymbol{\mathcal{X}}$ resulting from the addition or removal of a single layer blocks in the z-direction. For times at which all test cases exhibit an approximate local minimum in $\mathcal{U}_z$, e.g. $t\Gamma_0/R_0^2 \approx 70.5$, the error in $\mathcal{U}_z$ is also inversely proportional to ϵ^*.

Next, we consider the effect ϵ^* has on solutions of unsteady flows that are sensitive to small perturbations. The numerical investigations of [74, 83] on thin vortex rings with Gaussian vorticity distributions at $\mathrm{Re}_0 = 7{,}500$ have shown that small sinusoidal perturbations to the vortex ring centerline result in the growth of azimuthal instabilities, which in turn facilitate the laminar to turbulent transition of the flow. Here, we consider the evolution of a vortex ring at $\mathrm{Re}_0 = 7{,}500$ computed using values of $\epsilon^* = 10^{-2}$, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}. Unlike the numerical experiments of [74, 83], the vortex ring is initiated without imposing any perturbations beyond those implied by the numerical scheme.

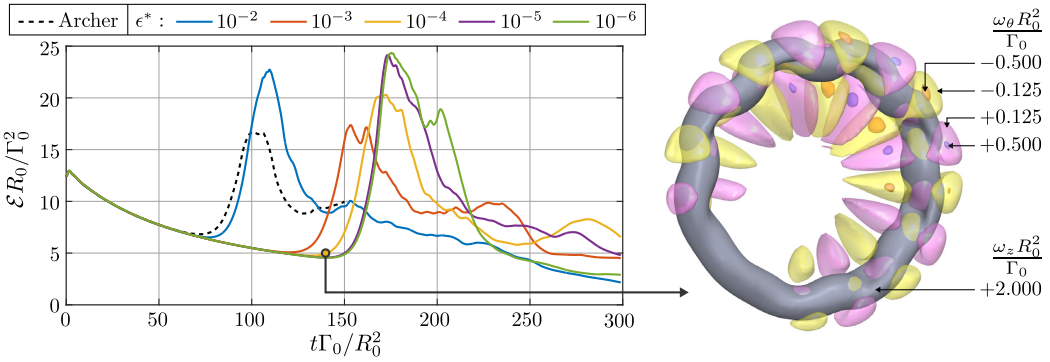

Figure 3.7: Time histories of $\mathcal{E}$ for a vortex ring at $\mathrm{Re}_0 = 7{,}500$ initiated with $\delta_0/R_0 = 0.2$ (*left*). Data point labeled as "Archer" where obtained from Archer et al. [74]. All parameters, with the exception of ϵ^*, are held constant across all test cases. Vorticity isosurfaces at $t\Gamma_0/R_0^2 = 137.6$ for test case $\epsilon^* = 10^{-4}$ (*right*).

The time history of $\mathcal{E}$ for all test cases is shown in the left plot of Figure 3.7. The transition into the early stages of turbulence, characterized by a peak in $\mathcal{E}$ resulting from an increase in the stretching of vortex filaments, is observed for all test cases. The growth of azimuthal instabilities and the development of secondary or "halo"

vortices occurring at beginning of the transition phase [74, 83] are depicted in the right plot of Figure 3.7.

As expected from the previous test cases for $\mathrm{Re}_0 = 500$, the values of $\mathcal{E}$ during the laminar regime for all test cases converge as ϵ^* tends zero. Also included in Figure 3.7 are the values of $\mathcal{E}$ reported in the numerical investigations of Archer et al. [74] for same vortex ring, which are nearly identical to values obtained from our test cases during the laminar regime.[27] Additionally, the vorticity isosurfaces shown in right plot of Figure 3.7 are qualitatively similar to the vorticity isosurfaces provided by Archer et al. [74] depicting the nonlinear growth of instabilities. In particular, the isosurfaces of both investigations demonstrate the noticeable presence of the $n = 1$ azimuthal Fourier mode and the presence of halo vortices (isosurfaces of ω_z in Figure 3.7) of similar magnitudes but alternating sign wedged between the approximately sinusoidally displaced inner-core (isosurfaces of ω_θ in Figure 3.7).

The time histories of $\mathcal{E}$ shown in Figure 3.7 indicate that the time at which $\mathcal{E}$ starts to increase prior to reaching its peak value, i.e. the time at which the flow starts to transition, increases as ϵ^* decreases, but converges as ϵ^* tend to zero. This trend is an expected consequence of the present adaptive grid technique since the flow field is slightly perturbed each time a block is removed, i.e. vorticity is implicitly set to zero outside D_{supp}. The magnitude of these perturbations is correlated to the value of ϵ^* used to compute the numerical solution. Over time, the perturbations introduced by the adaptive grid lead to changes in the flow field that break the axial symmetry of the solution, which in turn promotes the growth of instabilities. Figure 3.8 provides vorticity contours at different times that depict the breakdown of axial symmetry and the subsequent laminar to turbulent transition for a few test cases.

---

[27] In the discussion of Archer et al. [74], the test case corresponding to a vortex ring at $\mathrm{Re}_0 = 7{,}500$ initiated with $\delta_0/R_0 = 0.2$ is denoted as case "B3". Unlike the present test cases, the initial vorticity distribution for case B3 of Archer et al. [74] was slightly perturbed to promote an early transition.

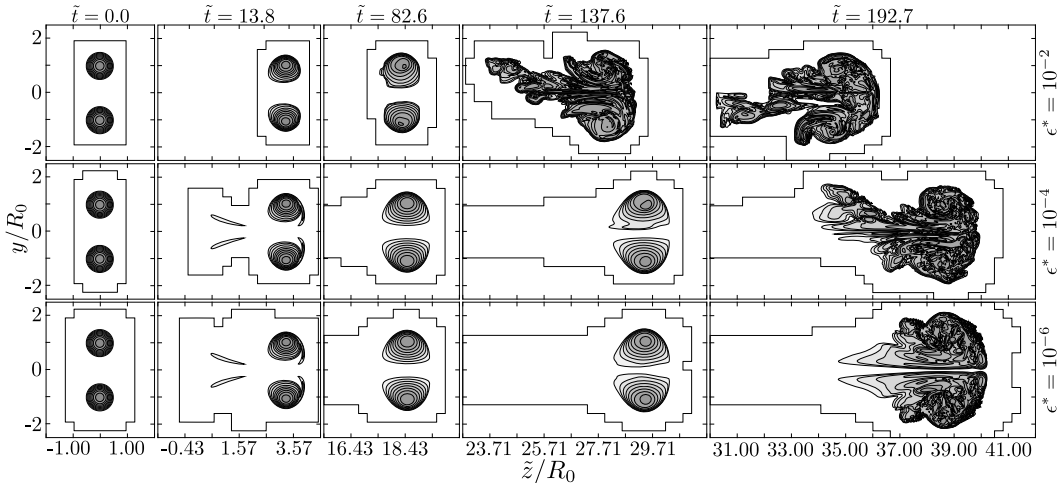

Figure 3.8: Vorticity magnitude on the y-z plane at $x = 0$ for test cases of $\epsilon^* = 10^{-2}$, 10^{-4}, and 10^{-6} at different times, $\tilde{t} = t\Gamma_0/R_0^2$. Contours correspond to values of $|\omega|R_0^2/\Gamma_0 = 4 \times \left(\frac{1}{2}\right)^i$ for $i = 8, 7, \ldots, 0$. Contours have been shifted the z-direction to account for the constant freestream velocity, $\tilde{z} = z - u_\infty^{(z)} t$. Thick lines depict the boundary of D_{xsoln}.

Figure 3.8 also depicts the computational domains that result from using different values of ϵ^*. As expected, higher values of ϵ^* result in tighter domains, but lead to some significant changes in the flow that are potentially relevant to specific applications. For example, Figure 3.8 indicates that using a value ϵ^* of 10^{-2} is sufficient to accurately track the laminar evolution of the vortex core, but does not adequately capture the large wake that develops behind the vortex ring.[28] We recall that the computational domain is determined by the particular choice of W_{supp} and ϵ_{supp}, both of which can be readily modified to accurately and efficiently capture the relevant physics of specific applications.

Figure 3.9 depicts vorticity isosurfaces during the transition phase ($t\Gamma_0/R_0^2 = 137.6$) and early turbulent regime ($t\Gamma_0/R_0^2 = 206.4$ and 275.2) for the test case of $\epsilon^* = 10^{-4}$. At $t\Gamma_0/R_0^2 = 206.4$ and 275.2, the presence of multiple thin vortex filaments and

[28]The maximum length, in terms of R_0, of the computational in the z-direction for is approximately 10, 26, 34, 46, 46 for test case with ϵ^* equal to 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}, respectively.

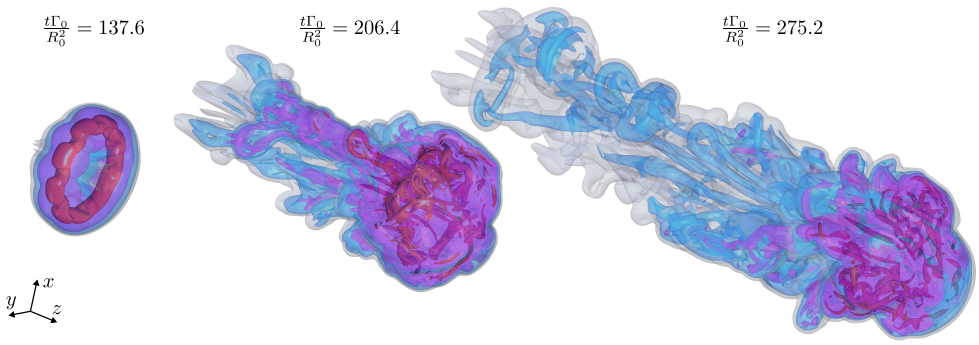

Figure 3.9: Translucent isosurfaces of the vorticity magnitude for the test case of $\epsilon^* = 10^{-4}$ at different times. Isosurfaces correspond to values of $|\omega|R_0^2/\Gamma_0 = 0.03125$, 0.125, 0.5, and 2.

the absence of a coherent core indicate that the vortex ring is in its early turbulent regime [74, 83]. A comparison of the vorticity isosurfaces at $t\Gamma_0/R_0^2 = 206.4$ and at $t\Gamma_0/R_0^2 = 275.2$ demonstrates that interwoven vorticity filaments near the core region are gradually pushed into the wake. As some of these structures are convected into the wake, they form hairpin vortices which persist for some time in the wake region. The periodic shedding of hairpin vortices into the wake is consistent with the numerical investigations of [74, 83], which in turn further verifies the physical fidelity of our solutions.

3.7 Conclusions

We have reported on a new fast, parallel solver for 3D, viscous, incompressible flows on unbounded domains based on LGFs. In this method, the incompressible Navier-Stokes equations are formally discretized on an unbounded staggered Cartesian grid using a second-order finite-volume scheme. This discretization technique has the advantage of enforcing discrete conservation laws and producing discrete operators with mimetic and commutativity properties that facilitate the implementation of fast, robust solvers. The system of DAEs resulting from the spatial-discretization of the momentum equation and the incompressibility constraint are integrated in

time by using an integrating factor technique for the viscous terms and a HERK scheme for the convective term and the incompressibility constraint. Computationally efficient expressions for the integrating factors are obtained via Fourier analysis on unbounded Cartesian grids. A projection method that takes advantage of the mimetic and commutativity properties of the discrete operators is used to efficiently solve the linear system of equations arising at each stage of the time integration scheme. This projection technique has the advantage of being equivalent to the LU decomposition of the system of equations, and, as a result, does not introduce any splitting-error and does not change the stability of the discretized equations.

In our formulation, solutions to the discrete Poisson problems and integration factor that are required to advance the flow are obtained through LGF techniques. These techniques express the solutions to inhomogeneous difference equations as the discrete convolution between source terms and the fundamental solutions of the discrete operators on unbounded regular grids. Fast, parallel solutions to the expressions resulting from the application of LGF techniques to discrete Poisson problems and integrating factors are obtained using the FMM for LGFs of [1].

As a result of our LGF formulation, the flow is solved using only information contained in the grid region where the vorticity and the divergence of the Lamb vector have non-negligible values. An adaptive block-structured grid and a velocity refresh technique are used to limit operations to a small finite computational domain. In order to efficiently compute solutions to a prescribed tolerance, weight functions and threshold values are used to determine the behavior of the adaptive grid. For the case of thin vortex rings, this approach results in computational domains that extend, at most, for approximately two ring radii in the radial direction. This is in contrast to previous grid-based methods which use a uniform grid to cover a box domain with lateral dimensions equal to seven [75] and eight [74] ring radii and impose periodic boundary conditions. In addition to the operation count reductions resulting from smaller computational grids, the automatically imposed natural free-

space boundary conditions of the present method circumvent the need to consider the physical implications of the non-negligible velocity field induced by the infinite array of vortex rings associated with periodic boundary conditions [74, 75].

The order of accuracy of the discretization and solution techniques is verified through refinement studies. The physical fidelity of the method is demonstrated in comparisons between computed and theoretical values for the propagation speed of thin vortex rings. Additionally, results for the evolution of a thin vortex ring at $\mathrm{Re}_0 = 7{,}500$ from the laminar to the early turbulent regime are shown to be in good agreement with investigations of other authors.

Acknowledgments

This work was partially supported by the United States Air Force Office of Scientific Research (FA950–09–1–0189) and the Caltech Field Laboratory for Optimized Wind Energy with Prof. John Dabiri as PI under the support of the Gordon and Betty Moore Foundation.

APPENDICES

3.A Discrete operators

In this appendix we provide point-operator and Fourier representations for the discrete operators of the present formulation. For operators that map onto $\mathbb{R}^{\mathcal{F}}$ or $\mathbb{R}^{\mathcal{E}}$, expressions for only one component of the resulting vector fields are provided since expressions for the other components are readily deduced. In the following discussion $\mathsf{c} \in \mathbb{R}^{\mathcal{C}}$, $\mathsf{f} \in \mathbb{R}^{\mathcal{F}}$, $\mathsf{e} \in \mathcal{E}$, and $\mathsf{v} \in \mathbb{R}^{\mathcal{V}}$ are arbitrary grid functions.

Point-operator representation based on the indexing convention depicted in Figure 3.1 are as follows:

Discrete gradient operators:

$$\mathsf{G} : \mathbb{R}^{\mathcal{C}} \mapsto \mathbb{R}^{\mathcal{F}},\ \Delta x[\mathsf{Gc}]^{(1)}_{i,j,k} = \mathsf{c}_{i+1,j,k} - \mathsf{c}_{i,j,k}, \tag{3.60}$$

$$\overline{\mathsf{G}} = -\mathsf{D}^\dagger : \mathbb{R}^{\mathcal{V}} \mapsto \mathbb{R}^{\mathcal{E}},\ -\Delta x[\mathsf{D}^\dagger \mathsf{v}]^{(1)}_{i,j,k} = \mathsf{v}^{(1)}_{i,j,k} - \mathsf{v}^{(1)}_{i-1,j,k}. \tag{3.61}$$

Discrete curl operators:

$$\mathsf{C} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{E}},\ \Delta x[\mathsf{Cf}]^{(1)}_{i,j,k} = \mathsf{f}^{(2)}_{i,j,k} - \mathsf{f}^{(2)}_{i,j,k+1} + \mathsf{f}^{(3)}_{i,j+1,k} - \mathsf{f}^{(3)}_{i,j,k}, \tag{3.62}$$

$$\overline{\mathsf{C}} = \mathsf{C}^\dagger : \mathbb{R}^{\mathcal{E}} \mapsto \mathbb{R}^{\mathcal{F}},\ \Delta x[\mathsf{C}^\dagger \mathsf{e}]^{(1)}_{i,j,k} = \mathsf{e}^{(2)}_{i,j,k-1} - \mathsf{e}^{(2)}_{i,j,k} + \mathsf{e}^{(3)}_{i,j,k} - \mathsf{e}^{(3)}_{i,j-1,k}. \tag{3.63}$$

Discrete divergence operators:

$$\mathsf{D} : \mathbb{R}^{\mathcal{E}} \mapsto \mathbb{R}^{\mathcal{V}},\ \Delta x[\mathsf{De}]_{i,j,k} = \mathsf{e}^{(1)}_{i+1,j,k} + \mathsf{e}^{(2)}_{i,j+1,k} + \mathsf{e}^{(3)}_{i,j,k+1} - \sum_{q=1}^{3} \mathsf{e}^{(q)}_{i,j,k}, \tag{3.64}$$

$$\overline{\mathsf{D}} = -\mathsf{G}^\dagger : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{C}},\ -\Delta x[\mathsf{G}^\dagger \mathsf{f}]_{i,j,k} = \sum_{q=1}^{3} \mathsf{f}^{(q)}_{i,j,k} - \mathsf{f}^{(1)}_{i-1,j,k} - \mathsf{f}^{(2)}_{i,j-1,k} - \mathsf{f}^{(3)}_{i,j,k-1}. \tag{3.65}$$

Discrete Laplace operators:

$$\mathsf{L}_{\mathcal{C}} : \mathbb{R}^{\mathcal{C}} \mapsto \mathbb{R}^{\mathcal{C}},\ \mathsf{L}_{\mathcal{C}} = -\mathsf{G}^\dagger \mathsf{G}, \qquad \mathsf{L}_{\mathcal{V}} : \mathbb{R}^{\mathcal{V}} \mapsto \mathbb{R}^{\mathcal{V}},\ \mathsf{L}_{\mathcal{V}} = -\mathsf{D}\mathsf{D}^\dagger, \tag{3.66}$$

$$\mathsf{L}_{\mathcal{F}} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{F}},\ \mathsf{L}_{\mathcal{F}} = -\mathsf{G}\mathsf{G}^\dagger - \mathsf{C}^\dagger \mathsf{C}, \qquad \mathsf{L}_{\mathcal{E}} : \mathbb{R}^{\mathcal{E}} \mapsto \mathbb{R}^{\mathcal{E}},\ \mathsf{L}_{\mathcal{E}} = -\mathsf{D}^\dagger \mathsf{D} - \mathsf{C}\mathsf{C}^\dagger. \tag{3.67}$$

Expressions for $[\mathsf{L}_{\mathcal{C}}\mathsf{c}]$, $[\mathsf{L}_{\mathcal{V}}\mathsf{v}]$, $[\mathsf{L}_{\mathcal{F}}\mathsf{f}]^{(\ell)}$, and $[\mathsf{L}_{\mathcal{E}}\mathsf{e}]^{(\ell)}$ are of the form:

$$(\Delta x)^2 [\mathsf{La}]_{i,j,k} = -6\mathsf{a}_{i,j,k} + \sum_{q \in \{-1,1\}} \left(\mathsf{a}_{i+q,j,k} + \mathsf{a}_{i,j+q,k} + \mathsf{a}_{i,j,k+q}\right). \tag{3.68}$$

Discrete nonlinear operator:[29]

$$\begin{aligned} &\tilde{\mathsf{N}} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{F}},\ [\tilde{\mathsf{N}}(\mathsf{f})]^{(1)}_{i,j,k} \\ &= \frac{1}{4} \sum_{q \in \{-1,0\}} \left[\mathsf{e}^{(2)}_{i,j,k+q} \left(\mathsf{f}^{(3)}_{i,j,k+q} + \mathsf{f}^{(3)}_{i+1,j,k+q}\right) - \mathsf{e}^{(3)}_{i,j+q,k} \left(\mathsf{f}^{(2)}_{i,j+q,k} + \mathsf{f}^{(2)}_{i+1,j+q,k}\right)\right], \end{aligned} \tag{3.69}$$

where $\mathsf{e} = \mathsf{Cf}$.

Linearized discrete nonlinear operator:

---

[29] The discrete nonlinear operator presented here is based on the discretization of the convective term in its rotational form, i.e. $\boldsymbol{\omega} \times \mathbf{u} - \frac{1}{2}\nabla(\mathbf{u} \cdot \mathbf{u})$, following the technique described in Zhang et al. [57]. As discussed in Section 3.2.1, $\tilde{\mathsf{N}}(\mathsf{f})$ is an approximation of $(\nabla \times \mathbf{f}) \times \mathbf{f}$.

The linearized form of $\tilde{\mathsf{N}}(\mathsf{f})$ about a constant uniform base flow, $\mathsf{f}_{\text{base}}(\mathbf{n},t)=\mathbf{f}_{\text{base}}$, is given by $\mathsf{M}\mathsf{f}'=[\mathsf{K}(\mathbf{f}_{\text{base}})]\mathsf{C}\mathsf{f}'$, where $\mathsf{f}'=\mathsf{f}-\mathsf{f}_{\text{base}}$ and

$$\mathsf{K}(\mathbf{f}_{\text{base}}):\mathbb{R}^{\mathcal{E}}\mapsto\mathbb{R}^{\mathcal{F}},\ [[\mathsf{K}(\mathbf{f}_{\text{base}})]\mathsf{e}']^{(1)}_{i,j,k}=\frac{1}{2}\sum_{q\in\{-1,0\}}\left(f^{(3)}_{\text{base}}\mathsf{e}^{(2)}_{i,j,k+q}-f^{(2)}_{\text{base}}\mathsf{e}^{(3)}_{i,j+q,k}\right). \tag{3.70}$$

Discussions regarding the properties of discrete operators are often facilitated by using a block vector/matrix notation to describe the grid functions and linear operators. Consider the grid spaces $\mathcal{X}$ and $\mathcal{Y}$ corresponding to either $\mathcal{F}$ or $\mathcal{E}$. Using block vector notation, a vector-valued grid function $\mathsf{x}\in\mathbb{R}^{\mathcal{X}}$ is expressed as

$$\mathsf{x}=\mathbb{S}_{\mathcal{X}}[\bar{\mathsf{x}}_1,\bar{\mathsf{x}}_2,\bar{\mathsf{x}}_3]^{\dagger}, \tag{3.71}$$

where the q-th block, $\bar{\mathsf{x}}_q$, corresponds to the values of the q-th component of x. Each x_q is a scalar real-valued grid function defined on an infinite Cartesian reference grid, which we denote by $\mathbb{R}^{\Lambda}$.[30] The shift operator $\mathbb{S}_{\mathcal{X}}:\mathbb{R}^{\Lambda}\mapsto\mathbb{R}^{\mathcal{X}}$ is used to transfer, or "shift", the values of grid functions defined on $\mathbb{R}^{\Lambda}$ to $\mathbb{R}^{\mathcal{X}}$ such that $[\mathsf{x}]^{(q)}(\mathbf{n})=\bar{\mathsf{x}}_q(\mathbf{n})$. Similarly, the transpose of $\mathbb{S}_{\mathcal{X}}$, denoted by $\mathbb{S}^{\dagger}_{\mathcal{X}}$, transfers values of grid functions defined on $\mathbb{R}^{\mathcal{X}}$ to $\mathbb{R}^{\Lambda}$. The block vector notation and shift operators readily extend to the case of linear operators. Using block matrix notation, a discrete linear operator $\mathsf{T}:\mathbb{R}^{\mathcal{X}}\mapsto\mathbb{R}^{\mathcal{Y}}$ is expressed as

$$\mathsf{T}=\mathbb{S}_{\mathcal{Y}}[\overline{\mathsf{T}}_{i,j}]\mathbb{S}^{\dagger}_{\mathcal{X}},\quad i,j=1,2,3, \tag{3.72}$$

where $\overline{\mathsf{T}}_{i,j}:\mathbb{R}^{\Lambda}\mapsto\mathbb{R}^{\Lambda}$.

We now turn our attention to the Fourier representations of grid functions and discrete linear operators. Consider the Fourier series, $\mathfrak{F}$, and the inverse Fourier transform, $\mathfrak{F}^{-1}$, given by:

$$[\mathfrak{F}\bar{\mathsf{u}}](\boldsymbol{\xi})=\sum_{\mathbf{m}\in\mathbb{Z}^3}e^{i\mathbf{m}\cdot\boldsymbol{\xi}}\bar{\mathsf{u}},\quad [\mathfrak{F}^{-1}\hat{\mathsf{u}}](\mathbf{m})=\frac{1}{(2\pi)^3}\int_{\boldsymbol{\xi}\in\Pi}e^{-i\boldsymbol{\xi}\cdot\mathbf{m}}\bar{\mathsf{u}}(\boldsymbol{\xi})\,d\boldsymbol{\xi}, \tag{3.73}$$

[30] Grid functions in $\mathbb{R}^{\Lambda}$ can also be regarded as functions mapping $\mathbb{Z}^3$ to R.

respectively, where $\Pi = (-\pi, \pi)^3$, $\mathsf{u} : \mathbb{Z}^3 \mapsto \mathbb{R}$, and $\hat{\mathsf{u}} : \Pi \mapsto \mathbb{C}$. Using block matrix notation, we extend $\mathfrak{F}$ and $\mathfrak{F}^{-1}$ to the case of grid functions in $\mathbb{R}^X$ by defining:

$$\mathfrak{F}_{\mathcal{X}} = \mathrm{diag}(\mathfrak{F}, \mathfrak{F}, \mathfrak{F})\mathbb{S}_{\mathcal{X}}, \quad \mathfrak{F}_{\mathcal{X}}^{-1} = \mathbb{S}_{\mathcal{X}}^{\dagger}\mathrm{diag}(\mathfrak{F}, \mathfrak{F}, \mathfrak{F}). \tag{3.74}$$

Next, let Ξ denote the set of all linear operators $\overline{\mathsf{Q}} : \mathbb{R}^{\Lambda} \mapsto \mathbb{R}^{\Lambda}$ such that the action of $\overline{\mathsf{Q}}$ on an arbitrary grid function $\bar{\mathsf{u}} \in \mathbb{R}^{\Lambda}$ is given by

$$[\overline{\mathsf{Q}}\bar{\mathsf{u}}](\mathbf{n}) = [\overline{\mathsf{K}}_{\mathsf{Q}} * \bar{\mathsf{u}}](\mathsf{n}) = \sum_{\mathbf{m} \in \mathbb{Z}^3} \overline{\mathsf{K}}_{\mathsf{Q}}(\mathbf{m} - \mathbf{n})\bar{\mathsf{u}}(\mathbf{m}), \tag{3.75}$$

where $\overline{\mathsf{K}}_{\mathsf{Q}} : \mathbb{Z}^3 \mapsto \mathbb{R}$ is a well-behaved discrete kernel function. Any operator belonging to Ξ is diagonalized using $\mathfrak{F}$ and $\mathfrak{F}^{-1}$,

$$[\overline{\mathsf{Q}}\bar{\mathsf{u}}](\mathbf{n}) = [\overline{\mathsf{K}}_{\mathsf{Q}} * \bar{\mathsf{u}}](\mathsf{n}) = [\mathfrak{F}^{-1}(\hat{\mathsf{K}}_{\mathsf{Q}}\hat{\mathsf{u}})](\mathsf{n}), \tag{3.76}$$

where $\hat{\mathsf{K}}_{\mathsf{Q}} = \mathfrak{F}\mathsf{K}_{\mathsf{Q}}$ and $\hat{\mathsf{u}} = \mathfrak{F}\mathsf{u}$. The block operators of all linear operators used in the present method belong to Ξ.

3.B Lattice Green's functions representations

The NSLGF method uses the LGFs G_{L} and $\mathsf{G}_{\mathsf{E}}(\alpha)$ to computed the action of $\mathsf{L}_{\mathcal{Q}}^{-1}$ and $\mathsf{E}_{\mathcal{Q}}(\alpha)$, respectively. Fourier and Bessel integrals for G_{L} and G_{E} are given by

$$(\Delta x)^2\mathsf{G}_{\mathsf{L}}(\mathbf{n}) = \frac{1}{8\pi^3}\int_{\Pi} \frac{\exp\left(-i\mathbf{n}\cdot\boldsymbol{\xi}\right)}{\sigma(\boldsymbol{\xi})}\, d\boldsymbol{\xi} = -\int_0^{\infty} e^{-6t} I_{n_1}(2t) I_{n_2}(2t) I_{n_3}(2t)\, dt \tag{3.77a}$$

$$[\mathsf{G}_{\mathsf{E}}(\alpha)](\mathbf{n}) = \frac{1}{8\pi^3}\int_{\Pi} \exp\left(-i\mathbf{n}\cdot\boldsymbol{\xi} - \sigma(\boldsymbol{\xi})\right)\, d\boldsymbol{\xi} = e^{-6\alpha} I_{n_1}(2\alpha) I_{n_2}(2\alpha) I_{n_3}(2\alpha) \tag{3.77b}$$

where $\sigma(\boldsymbol{\xi}) = 2\cos(\xi_1) + 2\cos(\xi_2) + 2\cos(\xi_3) - 6$, $\Pi = (-\pi, \pi)^3$, and $I_n(z)$ is the modified Bessel function of the first kind of order n.

Insights into the approximate behavior of $\mathsf{G}_{\mathsf{L}}(\mathbf{n})$ can be obtained by considering the case of $|\mathbf{n}| \to \infty$. Asymptotic expansions in terms of unique rational functions for

$\mathsf{G}_\mathsf{L}(\mathbf{n})$ are provided in [18]. For example,

$$(\Delta x)^2\mathsf{G}_\mathsf{L}(\mathbf{n}) = -\frac{1}{4\pi|\mathbf{n}|} - \frac{n_1^4+n_2^4+n_3^4-3n_1^2n_2^2-3n_1^2n_3^2-3n_2^2n_3^2}{16\pi|\mathbf{n}|^7} + \mathcal{O}\left(|\mathbf{n}|^{-5}\right), \tag{3.78}$$

as $|\mathbf{n}| \to \infty$. As expected, the leading order term corresponds to the fundamental solution of the Laplace operator.

Numerical procedures for efficiently evaluating $\mathsf{G}_\mathsf{L}(\mathbf{n})$ are provided in [1]. Values for $[\mathsf{G}_\mathsf{E}(\alpha)](\mathbf{n})$ can be readily computed using its Bessel form given by Eq. (3.77b). Although computing values of $\mathsf{G}_\mathsf{L}(\mathbf{n})$ and $[\mathsf{G}_\mathsf{E}(\alpha)](\mathbf{n})$ can potentially require a non-trivial number of operations, the FLGF method, used to compute the action of $\mathsf{L}_\mathcal{Q}^{-1}$ and $\mathsf{E}_\mathcal{Q}(\alpha)$, employs pre-processing techniques that limit the evaluation of point-wise values of LGFs to once per simulation.

3.C Stability analysis

Consider the linearization of Eq. (3.20) with respect to v about a uniform, constant base flow, $\mathsf{v}_\text{base}(\mathbf{n},t) = \tilde{\mathbf{u}}$, for the case of $\mathsf{u}_\infty = 0$,

$$\frac{d\mathsf{v}'}{dt} = [\mathsf{K}(\tilde{\mathbf{u}})]\mathsf{C}\mathsf{v}' + \mathsf{G}\mathsf{b}', \quad \mathsf{G}^\dagger\mathsf{v}' = 0, \tag{3.79}$$

where $\mathsf{v} = \mathsf{v}_\text{base} + \mathsf{v}'$ and $\mathsf{K}(\tilde{\mathbf{u}})$ is defined by Eq. (3.70).[31] The stability analysis of Eq. (3.79) is facilitated by using a null-space approach to transform the original DAE index 2 system to an equivalent ODE,

$$\frac{d\mathsf{q}}{dt} = \mathsf{C}^\dagger[\mathsf{K}(\mathsf{v}_\text{base})]\mathsf{q} \tag{3.80}$$

where $\mathsf{q} = \mathsf{C}\mathsf{v}'$, $\mathsf{v}' = \mathsf{C}^\dagger\mathsf{s}$, and $\mathsf{L}_\mathcal{E}\mathsf{s} = \mathsf{q}$ with $\mathsf{s} \to 0$ as $|\mathbf{n}| \to 0$. The details regarding the feasibility and equivalence of this transformation will be discussed in Section 3.4.1. It is readily verified that the discrete equations corresponding to the

[31] It is not necessary to linearize the integrating factors present in Eq. (3.20), since they can be commuted and made to cancel out after the linearization of $\tilde{\mathsf{N}}$.

HERK method for Eq. (3.79) and for Eq. (3.80) are also equivalent; hence, Eq. (3.79) and Eq. (3.80) have the same stability region.

The ODE given by Eq. (3.80) is diagonalized by the component-wise Fourier series $\mathfrak{F}_{\mathcal{E}}$, defined by Eq. (3.74),

$$\frac{d\hat{q}_k}{dt} = \frac{|\tilde{\mathbf{u}}|\Delta t}{\Delta x}\sigma(\boldsymbol{\xi})\hat{q}_k \;\; \forall i = 1, 2, 3, \tag{3.81a}$$

$$\sigma(\boldsymbol{\xi}) = -i\sum_{j=1}^{3}\frac{\tilde{u}_j}{|\tilde{\mathbf{u}}|}\sin\xi_i, \tag{3.81b}$$

where $\boldsymbol{\xi} \in \Pi = (-\pi, \pi)^3$.[32] It follows from Eq. (3.81b) that $\Re(\sigma(\boldsymbol{\xi})) = 0$ and $|\Im(\sigma(\boldsymbol{\xi}))| \leq \sqrt{3}$ for all $\boldsymbol{\xi} \in \Pi$. As a result, the linear stability Eq. (3.20) is determined by the stability of the scalar ODEs:

$$\frac{dy}{dt} = i\mu y \quad \forall \mu \in (-\gamma, \gamma), \quad \gamma = \sqrt{3}\frac{|\tilde{\mathbf{u}}|\Delta t}{\Delta x}. \tag{3.82}$$

Consider integrating the ODE given by Eq. (3.82) using the HERK method. In the absence of algebraic constraints, an HERK scheme reduces to a standard ERK scheme with the same tableau. Consequently, the region of absolute stability for the ODE of Eq. (3.82) is given by

$$\Omega = \{\mu \in \mathbb{R} : |R(i\mu)| < 1\}, \quad R(z) = 1 + z\mathbf{b}^{\dagger}\left(\mathbf{I} - z\mathbf{A}\right)^{-1}\mathbf{e}, \tag{3.83}$$

where $\mathbf{b}$ and $\mathbf{A}$ are defined by Eq. (3.22), and $\mathbf{e} = [\,1, 1, \ldots, 1\,]$ [63]. Eq. (3.83) implies that the IF-HERK method is linearly stable if the following CFL condition is satisfied:

$$\mathrm{CFL} = \frac{|\tilde{\mathbf{u}}|\Delta t}{\Delta x} < \mathrm{CFL}_{\max}, \quad \mathrm{CFL}_{\max} = \frac{\mu^*}{\gamma} \tag{3.84}$$

where $\mu^* = \sup(\Omega)$ depends on the RK coefficients of the scheme. For all the IF-HERK schemes defined in Eq. (3.33), the value of $\mathrm{CFL}_{\max}$ is unity.

---

[32] In order to simplify the expression for $\sigma(\boldsymbol{\xi})$ to the form given by Eq. (3.81a) it is necessary to account for $\mathsf{Dq} = 0$.

3.D Error estimates for integrating factors on finite domains

In this appendix we provide estimates for the difference between $\mathsf{E}_{\mathcal{Q}}(\alpha)$ and $\mathsf{M}_{\mathcal{Q}}^{\gamma}\mathsf{E}_{\mathcal{Q}}(\alpha)\mathsf{M}_{\mathcal{Q}}^{\gamma}$ inside D_γ, which are pertinent to the discussion of Section 3.4.4. Consider the constant uniform scalar field $\mathsf{u} \in \mathbb{R}^{\mathcal{Q}}$ and the domain D_γ, where D_γ is infinite in the x- and y-directions and semi-infinite in the z-direction. For this simplified case, it is sufficient to consider the 1D problem of computing

$$\mathsf{y} = \left[\mathsf{E}'(\alpha) - \mathsf{M}'\mathsf{E}'(\alpha)\mathsf{M}'\right]\mathsf{u} = \left[\mathsf{I} - \mathsf{M}'\mathsf{E}'(\alpha)\mathsf{M}'\right]\mathsf{u}, \tag{3.85}$$

where I is the identity operator,

$$\mathsf{E}'(\alpha)\mathsf{u} = \mathsf{G}'_{\mathsf{E}}(\alpha) * \mathsf{u}, \quad \mathsf{G}'_{\mathsf{E}}(n) = e^{-2\alpha}I_n(2\alpha), \tag{3.86}$$

and

$$[\mathsf{M}'\mathsf{u}](k) = \begin{cases} \mathsf{u}(k) & \text{if } k > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3.87}$$

As a result, the magnitude of the normalized difference, d, at $k > 0$ is given by

$$\mathsf{d}(k) = \frac{\mathsf{y}(k)}{|u|} = \sum_{j=0}^{\infty} e^{-2\alpha}I_{k-j+1}(2\alpha), \tag{3.88}$$

where $|u|$ is the magnitude of the uniform field u. Numerical approximations for $\mathsf{d}(k)$ are obtained by truncating the infinite sum of Eq. (3.88) to a finite number of terms, N, such that $I_{k-N+1}(2\alpha)/I_{k+1}(2\alpha)$ is less than a prescribed value.[33]

As discussed in Section 3.4.4, the current implementation of the NSLGF method uses Eq. (3.88) to estimate the error associated with approximating $\mathsf{E}_{\mathcal{Q}}(\alpha)\mathsf{u}$ by $\mathsf{M}_{\mathcal{Q}}^{\text{xsoln}}\mathsf{E}_{\mathcal{Q}}(\alpha)\mathsf{M}_{\mathcal{Q}}^{\text{xsoln}}\mathsf{u}$, where u is the velocity perturbation field. For this case, $|u|$ in Eq. (3.88) is set to be the maximum value of any component of u in D_{soln}. Numerical experiments of flows similar to those considered in Section 3.6 demonstrate that this technique leads to fairly conservative error estimates; in all experiments the actual

[33] For a fixed $z > 0$, $I_n(z)$ decreases as n increases. For a fixed $z > 0$, $I_n(z)$ decays faster than any exponential as $n \to \infty$.

error was less than 10% of the estimated error. Tighter error bounds that account for the domain shape and the distribution of u can potentially be obtained, but are not explored in the present work.

3.E Computation rates and parallel performance

The parallel performance of a MPI-based implementation of the present flow solver is investigated by computing a thin $\delta_0/R_0 = 0.2$ vortex ring at $\mathrm{Re}_0 = 7{,}500$ using different grid resolutions and core counts. Depicted in Figure 3.10 are the computation rates and parallel efficiencies of an average HERK stage computed on the US Army Research Laboratory's Cray XC40 (Excalibur) cluster. The parallel efficiency for each test series of constant problem size is reported as

$$e(p) = \frac{p_{\min}}{p}\frac{T(p_{\min})}{T(p)}, \tag{3.89}$$

where p is the number of cores, $p_{\min}$ is the minimal number of cores considered in the test series, and $T(p)$ is the wall-time. Values are averaged over the first 100 time-steps and the problem size of each test case held constant by disabling the grid adaptivity.

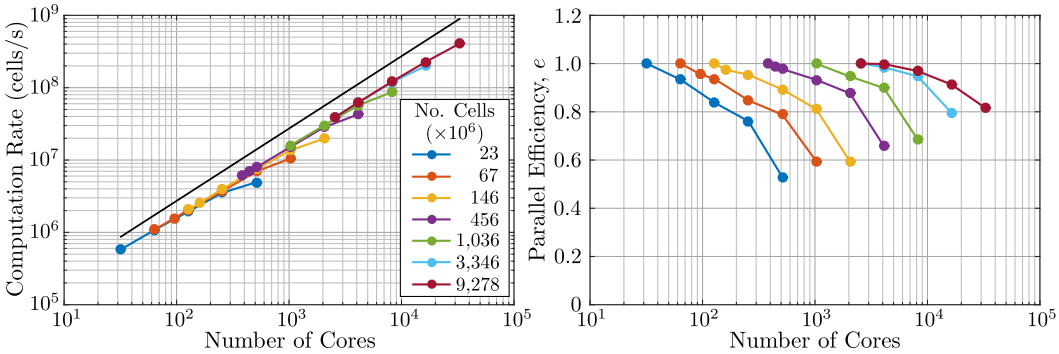

Figure 3.10: Average computation rates (*left*) and parallel efficiencies (*right*) for a single HERK stage of a thin $\delta_0/R_0 = 0.2$ vortex ring at $\mathrm{Re}_0 = 500$ computed using different grid resolutions and core counts. Each test series, depicted as points of the same color, corresponds to multiple test cases performed with same grid resolution, but with different core counts.

Figure 3.10 demonstrates that for appropriate problem size to core count ratios the present implementation achieves good parallel efficiency even for problems involving approximately 10^{10} grid cells and 3×10^4 cores. Based on the left plot of Figure 3.10 we expect parallel efficiency above 80% for flows computed using at least 2×10^5 grid cells per core. This constraint on the parallel efficiency of the flow solver is expected from the numerical experiments reported for FLGF method [1] and consistent with the parallel considerations of other FMM-based solvers [36, 37].

3.F Thin vortex ring at Re = 20,000

As a final demonstration of the capabilities of the LGF flow solver, we compute the evolution from the laminar to the early turbulent regime of a thin $\delta_0/R_0 = 0.1$ vortex ring at $\mathrm{Re}_0 = 20{,}000$. Recent direct numerical simulations [74, 83, 84] have considered the evolution of thin vortex rings at Reynolds numbers between 5,000 and 10,000. To our knowledge this is the first reported direct numerical simulation of a viscous, incompressible vortex ring at a Reynolds number above 10,000.

In contrast to the initially unperturbed vortex rings reported in Section 3.6, a small perturbation to the centreline path of the ring has been added, i.e. the ring radius is now $R'(\theta) = R + \epsilon(\theta)$, to break the initial axial symmetry and promote the laminar to turbulent transition of the flow. Here, we follow the perturbation procedure of Archer et al. [74] and consider perturbations of the form $\epsilon(\theta) = \zeta f_N(\theta)$, where $f_N(\theta)$ is the superposition of the first N Fourier modes (excluding the zeroth mode) each with unit amplitude and random phase. Perturbation amplitudes of $\zeta = 10^{-5}$ are imposed on the first $N = 32$ azimuthal modes of the initial vortex ring considered in this section.

Depicted in Figure 3.11 is the enstrophy time history of the vortex ring obtained using different grid resolutions. The curves for $\delta_0/R_0 = 16$ and $\delta_0/R_0 = 20$ are visually indistinguishable up to the point of transition, and result in enstrophy peaks with nearly identical shapes. This indicates that the fine scales resulting from the

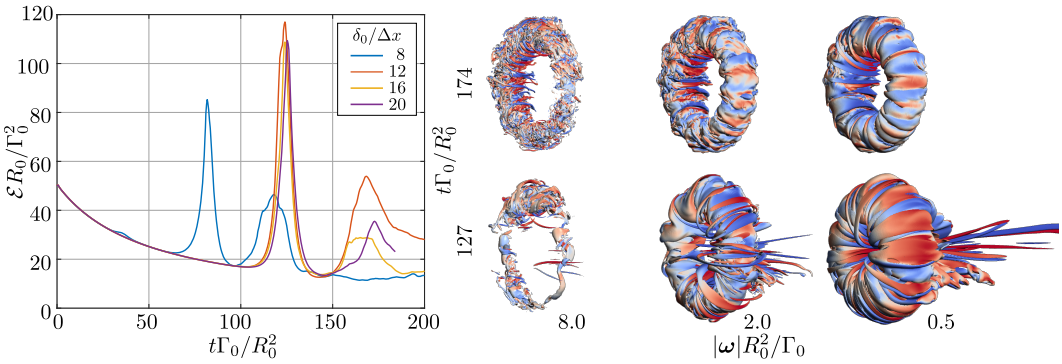

Figure 3.11: Time history of $\mathcal{E}$ for a vortex ring at $\mathrm{Re}_0 = 20{,}000$ initiated with $\delta_0/R_0 = 0.1$ computed using different grid resolutions (*left*). Vorticity iso-surfaces at $t\Gamma_0/R_0^2 = 127$ and $t\Gamma_0/R_0^2 = 174$ for the test case computed using $\delta_0/\Delta x = 20$ (*right*). Iso-surface color based on the stream-wise vorticity.

stretching vorticity filaments during the transition phase are well-resolved by the test case using $\delta_0/R_0 = 20$. Past the enstrophy peak, e.g. $t\Gamma_0/R_0^2 \geq 150$, the curves for all test cases diverge, which is expected from the sensitivity of instantaneous measurements of turbulent flows to small perturbations. The creation of fine vorticity filaments during the transition phase is confirmed by visual inspection of the vorticity strength isosurfaces shown in Figure 3.11 for $t\Gamma_0/R_0^2 = 127$. Also shown by the isosurfaces included in Figure 3.11 is the loss of a coherent ring core and the wrapping of thin vorticity filaments about the new nominal core region during the early turbulent regime, e.g. $t\Gamma_0/R_0^2 = 174$, which are expected from the $\mathrm{Re}_0 = 7{,}500$ test cases discussed in Section 3.6.4. As a point of reference, approximately 2.25×10^8 grid cells split across 2,048 cores were used to compute the $\delta_0/R_0 = 20$ test case at $t\Gamma_0/R_0^2 = 174$.

Chapter 4

A FAST IMMERSED BOUNDARY METHOD FOR EXTERNAL INCOMPRESSIBLE VISCOUS FLOWS USING LATTICE GREEN'S FUNCTIONS

Submitted to Journal of Computational Physics, April 2016

Chapter abstract

A new parallel, computationally efficient immersed boundary method for solving three-dimensional, viscous, incompressible flows on unbounded domains is presented. Immersed surfaces with prescribed motions are generated using the interpolation and regularization operators obtained from the discrete delta function approach of the original (Peskin's) immersed boundary method. Unlike Peskin's method, boundary forces are regarded as Lagrange multipliers that are used to satisfy the no-slip condition. The incompressible Navier-Stokes equations are discretized on an unbounded staggered Cartesian grid and are solved in a finite number of operations using lattice Green's function techniques. These techniques are used to automatically enforce the natural free-space boundary conditions and to implement a novel block-wise adaptive grid that significantly reduces the run-time cost of solutions by limiting operations to grid cells in the immediate vicinity and near-wake region of the immersed surface. These techniques also enable the construction of practical discrete viscous integrating factors that are used in combination with specialized half-explicit Runge-Kutta schemes to accurately and efficiently solve the differential algebraic equations describing the discrete momentum equation, incompressibility constraint, and no-slip constraint. Linear systems of equations resulting from the time integration scheme are efficiently solved using an approximation-free nested projection technique. The algebraic properties of the discrete operators are used to reduce projection steps to simple discrete elliptic problems, e.g. discrete Poisson problems, that are compatible with recent parallel fast multipole methods for difference equations. Numerical experiments on low-aspect-ratio flat plates and spheres at Reynolds numbers up to 3,700 are used to verify the accuracy and physical fidelity of the formulation.

4.1 Introduction

Immersed boundary (IB) methods are numerical techniques for solving PDEs on Eulerian grids with immersed surfaces that are described by Lagrangian structures

[85–87]. Immersed surfaces are emulated without modifying the underlying PDE discretization by the addition of forcing terms and constraint equations resulting from the regularization of Dirac delta convolutions linking Eulerian and Lagrangian quantities. In addition to circumventing computationally expensive body-fitted grid generation, this approach facilitates the extensions of robust and efficient solvers, e.g. Cartesian-grid methods, to problems involving immersed surfaces. The original IB method [88] was developed for flexible elastic structures, but has since been extended to handle more general fluid-structure interactions, including rigid bodies and bodies with prescribed motions [69, 89–95]. The numerous variants of the IB method and some of their higher-order extensions are reviewed in [85–87]. Here, we focus on *distributed Lagrange multiplier (DLM) methods* [69, 91, 93, 94, 96–98] since they are particularly robust IB methods for computing flows around bodies with prescribed motions [87].

DLM methods treat boundary forces as Lagrange multipliers used to enforce prescribed surface boundary conditions. For the case of fluid flows, these methods are typically expressible in forms analogous to traditional fractional-step and projection methods and can be distinguished in part by differences in splitting errors, underlying PDEs, discretization schemes, and numerical solvers [87, 93, 94]. The null-space (discrete streamfunction) projection approach [69] and the Rigid-IBAMR solver [94] are examples of robust incompressible Navier-Stokes DLM methods free of splitting errors. The absence of splitting errors ensures that solutions retain the accuracy, stability, and physical fidelity of the PDE discretization scheme [69, 70, 72, 93, 94].

IB methods for external flows typically employ spatially truncated fluid domains with approximate free-space boundary conditions, which in turn introduce *blockage* errors that adversely affect the accuracy and can even change the dynamics of the numerical solution [41–44]. Large computational domains in combination with stretched grids [93, 99, 100], local grid refinement [94, 101, 102], and far-field approximation techniques [69] are commonly used to reduce blockage errors. In ad-

dition to increasing the number of computational elements, these techniques often require the use of numerical solvers that are less efficient than regular-grid solvers (e.g. FFT techniques, multigrid, etc.) and typically result in discretization schemes that do not formally share the same conservation, commutativity, orthogonality, and symmetry properties of standard staggered Cartesian discretizations of *infinite* (periodic or unbounded) domains.

In order to eliminate the errors associated with artificial boundary conditions and to limit operations to small regions dictating the flow evolution (e.g. regions of significant vorticity), while preserving the efficiency and robustness inherent to Cartesian staggered grid methods, we proposed [2] a fast incompressible Navier-Stokes solver based on the fundamental solution, or lattice Green's function (LGF), of discrete operators. Similar to particle and vortex methods, LGF techniques have efficient nodal distributions, automatically enforce natural free-space boundary conditions, and can be evaluated using fast multipole methods (FMMs), e.g. the 2D serial method [11] and the 3D parallel method [1]. Using the LGF-FMM [1] in combination with an projection technique that is free of splitting errors, the LGF flow solver [2] computes fast, parallel solutions to the viscous integrating factor (IF) half-explicit Runge-Kutta (HERK) time integration scheme used to solve the velocity and pressure of the flow.

The present method numerically solves the IB formulation for the incompressible Navier-Stokes equations given, in its continuous form, by

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u} + \int_{\Gamma(t)} \mathbf{f}_\Gamma \left(\boldsymbol{\xi}, t \right) \delta \left(\mathbf{X} \left(\boldsymbol{\xi}, t \right) - \mathbf{x} \right) \, d\boldsymbol{\xi}, \tag{4.1a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{4.1b}$$

$$\int_{\mathbb{R}^3} \mathbf{u} \left(\mathbf{x}, t \right) \delta \left(\mathbf{x} - \mathbf{X} \left(\boldsymbol{\xi}, t \right) \right) \, d\mathbf{x} = \mathbf{u}_\Gamma \left(\boldsymbol{\xi}, t \right), \tag{4.1c}$$

where the immersed surface $\Gamma(t)$ is parametrized by $\boldsymbol{\xi}$, and $\mathbf{X}\left(\boldsymbol{\xi}, t\right) \in \Gamma(t)$. The velocity, pressure, and Reynolds number of the flow are denoted by $\mathbf{u}\left(\mathbf{x}, t\right)$, $p\left(\mathbf{x}, t\right)$, and Re. Here, Eq. (4.1c) is taken to be the no-slip condition on $\Gamma(t)$, where

$\mathbf{u}_\Gamma(\boldsymbol{\xi},t) = [\partial\mathbf{X}/\partial t](\boldsymbol{\xi},t)$.[1] The body force term in Eq. (4.1a), with unknown force density $\mathbf{f}_\Gamma(\boldsymbol{\xi},t)$, is computed so that $\mathbf{u}(\mathbf{x},t)$ satisfies Eq. (4.1c). The fluid variables $\mathbf{u}(\mathbf{x},t)$ and $p(\mathbf{x},t)$ are defined for all $\mathbf{x}\in\mathbb{R}^3$, and subject to the boundary condition $\mathbf{u}(\mathbf{x},t)\to 0$ as $|\mathbf{x}|\to\infty$.

Computationally efficient solutions for moving non-deformable (rigid) immersed surfaces are facilitated by writing Eq. (4.1) in an accelerating frame of reference (moving with the body), but with a change of dependent variable to the velocity in the inertial reference frame [103–105]. This change of variable is useful because the velocity in resulting equations tends to zero a large distances and source terms resulting from the accelerating reference frame can be absorbed into the non-linear and pressure gradient terms. The governing equations written in the accelerating frame of reference are give by

$$\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}_a\cdot\nabla)(\mathbf{u}_a + 2\boldsymbol{\Omega}\times\mathbf{x}_a) = -\nabla q + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} + \delta\left(\Gamma(t),\mathbf{f}_\Gamma,\mathbf{x}\right) \tag{4.2a}$$

$$\nabla\cdot\mathbf{u} = 0, \quad \delta\left(\Gamma(t),\mathbf{u},\boldsymbol{\xi}\right) = \mathbf{u}_{\Gamma,a}\left(\boldsymbol{\xi},t\right) + \mathbf{u}_r\left(\mathbf{X}\left(\boldsymbol{\xi},t\right),t\right). \tag{4.2b}$$

Here, $\mathbf{x}$ and $\mathbf{x}_a = \mathbf{x}-\mathbf{R}(t)$ denote the position vector of a point relative to the origin of the inertial- and accelerating-frame coordinates, respectively. The accelerating-frame coordinates are taken to be centered about $\mathbf{R}(t)$, to translate with a velocity $\mathbf{U}(t) = [\frac{d\mathbf{R}}{dt}](t)$, and to rotate about $\mathbf{R}(t)$ with an angular velocity $\boldsymbol{\Omega}(t)$ when viewed from the inertial frame. For ease of notation, we have re-used the same symbols for the differentials as in Eq. (4.1), but they now refer to the accelerating-frame coordinates, i.e. $\frac{\partial}{\partial t}$ means differentiation holding $\mathbf{x}_a$ fixed, ∇ refers to the gradient in the accelerating-frame coordinates, etc. The vectors $\mathbf{u}(\mathbf{x},t)$ and $\mathbf{u}_a(\mathbf{x},t) = \mathbf{u}(\mathbf{x},t) - \mathbf{u}_r(\mathbf{x}_a,t)$, respectively, correspond to the velocity in the inertial and accelerating reference frames, where $\mathbf{u}_r(\mathbf{x}_a,t) = \mathbf{U}(t) + \boldsymbol{\Omega}(t)\times\mathbf{x}_a$ is the velocity of

[1] For the case of closed immersed surfaces, we limit our attention to prescribed motions that are volume conserving or, equivalently, surface velocities that satisfy the incompressibility condition $\int_{\Gamma(t)}\mathbf{u}_\Gamma(\boldsymbol{\xi},t)\cdot\hat{\mathbf{n}}(\boldsymbol{\xi},t)\,d\boldsymbol{\xi} = 0$, where $\hat{\mathbf{n}}$ is the surface normal unit vector.

a point in the accelerating frame relative to the inertial frame. The scalar $q(\mathbf{x}, t)$ is a pressure-like quantity that can be related to the inertial-frame pressure $p(\mathbf{x}, t)$, up to an arbitrary time-dependent constant, using $q(\mathbf{x}, t) = p(\mathbf{x}, t) - \frac{1}{2}|\mathbf{u}_r(\mathbf{x}_a, t)|^2$. Operators $\delta\left(\Gamma(t), \mathbf{f}_\Gamma, \mathbf{x}\right)$ and $\delta\left(\Gamma(t), \mathbf{u}, \boldsymbol{\xi}\right)$ are shorthands for the δ-function convolutions of Eq. (4.1a) and Eq. (4.1c). The vectors $\mathbf{X}_a(\boldsymbol{\xi}, t) = \mathbf{X}(\boldsymbol{\xi}, t) - \mathbf{R}(t)$ and $\mathbf{u}_{\Gamma,a}(\boldsymbol{\xi}, t) = \mathbf{u}_\Gamma(\boldsymbol{\xi}, t) - \mathbf{u}_r(\mathbf{X}_a(\boldsymbol{\xi}, t), t)$, respectively, denote the position and corresponding of velocity of a point on $\Gamma(t)$ in the accelerating reference frame. Lastly, we clarify that the Eulerian grid and Lagrangian structure used to discretized Eq. (4.2) are constructed in the accelerating-frame coordinates, which implies that the Lagrangian structure of rigid surfaces can be made *stationary* with respect to the Eulerian grid by specifying appropriate values for $\mathbf{R}(t)$ and $\boldsymbol{\Omega}(t)$. This simplification is used to construct efficient solvers and pre-processing techniques that greatly reduce the run-time cost of practical flows around accelerating rigid surfaces.

In this paper, we extend the unbounded domain LGF flow solver [2] to include immersed surfaces with prescribed motions using a Lagrange multiplier approach. In Section 4.2, we discuss the discretization of Eq. (4.1) on unbounded fluid domains emphasizing the modifications to the LGF techniques and IF-HERK time integration schemes of [2] used to efficiently and accurately include immersed boundaries. Linear systems of equations arising at each Runge-Kutta stage are solved using the fast, LGF-based, *exact* nested projection technique described in Section 4.3. Computationally expensive projection steps are shown to be reducible to simple Poisson, Poisson-like, or viscous integrating factor problems that are compatible with the LGF-FMM [1] and make use of LGFs that are readily computed. Significant operation count reductions for the numerical solutions of discrete elliptic equations are demonstrated for problems involving the IB regularization and interpolation operators by limiting operations to small source and target regions near the immersed surface. Additionally, we discuss the computational considerations of some iterative and direct solution techniques for the boundary force Schur complement problem ar-

sing in the nested projection, and demonstrate that for many practical flows around rigid surfaces a dense linear algebra pre-processing technique results in boundary force solutions that contribute negligibly to the total run time. In Section 4.4, we modify the block-wise adaptive computational grid and specialize the adaptivity criteria [2] to efficiently accommodate immersed surfaces. Finally, in Section 4.5, we verify the formulation through numerical experiments on flows around flat plates and spheres at Reynolds numbers up to 3,700.

4.2 Discretization

4.2.1 Immersed boundary method on unbounded staggered Cartesian grids

In this section we highlight important features of the spatial discretization of Eq. 4.1. Additional details pertaining to the flow discretization and the IB regularization/interpolation operators are provided in the discussions of the LGF flow solver [2] and of the IB-DLM methods [69, 93, 94], respectively.

To begin, we formally discretize Eq. (4.2) on an unbounded staggered Cartesian grid using second-order finite-volume operators,

$$\frac{d\mathsf{u}}{dt} + \mathsf{N}(\mathsf{u},t) = -\mathsf{G}\mathsf{q} + \frac{1}{\mathrm{Re}}\mathsf{L}_{\mathcal{F}}\mathsf{u} + [\mathscr{R}(t)]\mathfrak{f}, \tag{4.3a}$$

$$\overline{\mathsf{D}}\mathsf{u} = 0, \quad [\mathscr{I}(t)]\mathsf{u} = \mathfrak{u}, \tag{4.3b}$$

where $\mathsf{u}(\mathbf{n},t)$ and $\mathsf{q}(\mathbf{n},t)$ are the discrete velocity and pressure-like variables, i.e. $\mathsf{u} \approx \mathbf{u}$ and $\mathsf{q} \approx q$, at time $t \in \mathbb{R}_{\geq 0}$ and grid location $\mathbf{n} \in \mathbb{Z}^3$. Operators G, $\overline{\mathsf{D}}$, and $\mathsf{L}_{\mathcal{F}}$ are discrete gradient, divergence, and vector-Laplace operators. The non-linear operator $\mathsf{N}(\mathsf{u},t)$ is a discrete approximation of $(\mathbf{u}_a \cdot \nabla)(\mathbf{u}_a - 2\boldsymbol{\Omega} \times \mathbf{x}_a)$.[2] The surface functions $\mathfrak{f}(i,t)$ and $\mathfrak{u}(i,t)$ correspond to the discrete body force and

[2]The present formulation does not assume a particular form or discretization scheme for the non-linear term $(\mathbf{u}_a \cdot \nabla)(\mathbf{u}_a + 2\boldsymbol{\Omega} \times \mathbf{x}_a)$. For example, standard inertial-frame techniques can be used to discretize the non-linear term in its divergence form $\nabla \cdot (\mathbf{u}_a(\mathbf{u}_a + 2\boldsymbol{\Omega} \times \mathbf{x}_a))$ [103] or in its rotational form $(\nabla \times \mathbf{u}) \times \mathbf{u}_a + \frac{1}{2}\nabla|\mathbf{u}_a|^2$ [105]. The numerical experiments of Section 4.5 are computed using the latter form and the operator stencils provided in [2].

surface velocity of the i-th Lagrangian marker located at $\mathbf{X}(\boldsymbol{\xi}_i, t) \in \Gamma(t)$, where $i \in [1, N_L]$. We clarify that $\mathfrak{u}(i,t) = \mathbf{u}_\Gamma(\boldsymbol{\xi}_i, t)$ includes the relative velocity of the Lagrangian structure with respect to the Eulerian grid $\mathbf{u}_{\Gamma,a}(\boldsymbol{\xi}_i, t)$ (equal to zero for rigid surfaces) and the relative velocity of the accelerating-frame with respect to the inertial-frame $\mathbf{u}_r(\mathbf{X}_a(\boldsymbol{\xi}_i, t), t)$. The time-dependent interpolation and regularization operators $\mathscr{I}(t)$ and $\mathscr{R}(t)$ are constructed by regularizing the δ-function convolutions of Eq. (4.1a) and Eq. (4.1c). We limit our attention to discretizations of the form

$$[[\mathscr{I}(t)]\mathsf{v}]^{(k)}(i,t) = (\Delta x)^3 \sum_{\mathbf{n}\in\mathbb{Z}^3} \mathsf{v}^{(k)}(\mathbf{n},t)\delta_{\Delta x}\left(\mathbf{x}_{\mathcal{F}}^{(k)}(\mathbf{n}) - \mathbf{X}(\boldsymbol{\xi}_i,t)\right), \tag{4.4a}$$

$$[[\mathscr{R}(t)]\mathfrak{g}]^{(k)}(\mathbf{n},t) = \sum_{i\in[1,N_L]} \mathfrak{g}^{(k)}(i,t)\delta_{\Delta x}\left(\mathbf{x}_{\mathcal{F}}^{(k)}(\mathbf{n}) - \mathbf{X}(\boldsymbol{\xi}_i,t)\right), \tag{4.4b}$$

where Δx is the grid cell size, $(\cdot)^{(k)}$ denotes the k-th vector component, $\mathbf{x}_{\mathcal{F}}^{(k)}(\mathbf{n})$ is the location of the k-th face of the $\mathbf{n}$-th grid cell, and $\delta_h(\mathbf{x}) = h^{-3}\prod_{k=1}^3 \phi(x_k/h)$ is a discrete delta function defined as the tensor product of the single-variable kernel function $\phi(x)$. The operators $\mathscr{I}(t)$ and $\mathscr{R}(t)$ are adjoints (up to a scalar factor) under the standard inner product, i.e. $\mathscr{I}(t) = (\Delta x)^3[\mathscr{R}(t)]^\dagger$.

The staggered grid consists of cells ($\mathcal{C}$) and vertices ($\mathcal{V}$) containing scalar flow quantities, and faces ($\mathcal{F}$) and edges ($\mathcal{E}$) containing vector flow quantities. We denote real-valued grid functions with values on $\mathcal{Q} \in \{\mathcal{C}, \mathcal{F}, \mathcal{E}, \mathcal{V}\}$ by $\mathbb{R}^{\mathcal{Q}}$, e.g. $[\mathsf{u}](t) \in \mathbb{R}^{\mathcal{F}}$ and $[\mathsf{q}](t) \in \mathbb{R}^{\mathcal{C}}$. Similarly, real-valued functions with vector values specified at each Lagrangian point are denoted by $\mathbb{R}^\Gamma$, e.g. $[\mathfrak{f}](t)$, $[\mathfrak{u}](t) \in \mathbb{R}^\Gamma$. The full set of discrete vector operators used in subsequent discussions is given by the discrete gradients $\mathsf{G} : \mathbb{R}^{\mathcal{C}} \mapsto \mathbb{R}^{\mathcal{F}}$ and $\overline{\mathsf{G}} : \mathbb{R}^{\mathcal{V}} \mapsto \mathbb{R}^{\mathcal{E}}$, the discrete curls $\mathsf{C} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{E}}$ and $\overline{\mathsf{C}} : \mathbb{R}^{\mathcal{E}} \mapsto \mathbb{R}^{\mathcal{F}}$, the discrete divergences $\mathsf{D} : \mathbb{R}^{\mathcal{E}} \mapsto \mathbb{R}^{\mathcal{V}}$ and $\overline{\mathsf{D}} : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\mathcal{C}}$, and the discrete Laplacians $\mathsf{L}_{\mathcal{Q}} : \mathbb{R}^{\mathcal{Q}} \mapsto \mathbb{R}^{\mathcal{Q}}$ for all $\mathcal{Q}$ in $\{\mathcal{C}, \mathcal{F}, \mathcal{E}, \mathcal{V}\}$. The present formulation extensively makes use of the symmetry (e.g. $\overline{\mathsf{D}} = -\mathsf{G}^\dagger$), orthogonality (e.g. $\mathrm{Im}(\mathsf{G}) = \mathrm{Null}(\mathsf{C})$), mimetic (e.g. $\mathsf{L}_{\mathcal{C}} = -\mathsf{G}^\dagger\mathsf{G}$, $\mathsf{L}_{\mathcal{F}} = -\mathsf{G}\mathsf{G}^\dagger - \mathsf{C}^\dagger\mathsf{C}$), and commutativity (e.g. $\mathsf{L}_{\mathcal{F}}\mathsf{G} = \mathsf{G}\mathsf{L}_{\mathcal{C}}$) properties of the discretization scheme. Related to these properties is the fact that the scheme conserves momentum, kinetic energy, and circulation in the absence of

time-differencing errors, viscosity, and immersed surfaces provided N is suitably discretized [57, 60]. Under similar provisions, the adjointness of $\mathscr{R}(t) : \mathbb{R}^{\Gamma} \mapsto \mathbb{R}^{\mathcal{F}}$ and $\mathscr{I}(t) : \mathbb{R}^{\mathcal{F}} \mapsto \mathbb{R}^{\Gamma}$ also ensures the conservation kinetic energy [85] for the case of stationary immersed surfaces.

The practical implementation of Eq. (4.3) is facilitated by subtracting $\frac{1}{2}\mathsf{GP}(\mathsf{u}-\mathsf{u}_r)$, where $\mathsf{P}(\mathsf{v})$ is a discrete approximation of $|\mathbf{v}|^2$, from both sides of Eq. (4.3a) and by writing f as $-(\Delta x)^3\tilde{\mathsf{f}}$. This yields

$$\frac{d\mathsf{u}}{dt} + \tilde{\mathsf{N}}(\mathsf{u},t) = -\mathsf{Gd} + \frac{1}{\mathrm{Re}}\mathsf{L}_{\mathcal{F}}\mathsf{u} + [\mathscr{I}(t)]^{\dagger}\,\tilde{\mathsf{f}}, \tag{4.5a}$$

$$\overline{\mathsf{D}}\mathsf{u} = 0, \tag{4.5b}$$

$$[\mathscr{I}(t)]\,\mathsf{u} = \mathfrak{u}, \tag{4.5c}$$

where $\tilde{\mathsf{N}}(\mathsf{u},t) = \mathsf{N}(\mathsf{u},t) - \frac{1}{2}\mathsf{GP}(\mathsf{u}-\mathsf{u}_r)$, $\mathsf{d} = \mathsf{q} + \frac{1}{2}\mathsf{P}(\mathsf{u}-\mathsf{u}_r)$, and $\mathsf{u}_r^{(k)}(\mathbf{n},t) = \mathbf{u}_r^{(k)}(\mathbf{x}_{\mathcal{F}}^{(k)}(\mathbf{n}),t)$. The non-linear term $\tilde{\mathsf{N}}(\mathsf{u},t)$ is a discrete approximation of $(\mathbf{u}_a \cdot \nabla)(\mathbf{u}_a + 2\boldsymbol{\Omega}\times\mathbf{x}_a) - \frac{1}{2}\nabla|\mathbf{u}_a|^2 = (\nabla\times\mathbf{u})\times\mathbf{u}_a$, and has the computational advantage having values that decay significantly faster at large distances compared to $\mathsf{N}(\mathsf{u},t)$; additional details for inertial-frame flows without immersed surfaces are discussed in [2]. We call attention to the fact that the discrete equations resulting from the temporal discretizations of Eq. (4.3) and Eq. (4.5) are, in general, different. But, as will be shown in Section 4.2.3, the present time integration scheme evaluates $\tilde{\mathsf{N}}(\mathsf{u},t)$ and Gd at the same times, and effectively computes the contributions from these terms as $\tilde{\mathsf{N}}(\mathsf{u},t) + \mathsf{Gd} = \mathsf{N}(\mathsf{u},t) + \mathsf{Gq}$. This implies that the numerically integrated solutions to Eq. (4.3) and Eq. (4.5) are *equivalent* in the absence of finite precision errors. Lastly, we note that the $\mathsf{q}(\mathbf{n},t)$ tends to an arbitrary time-dependent constant (taken to be zero) as $|\mathbf{n}| \to \infty$ and discrete pressure $\mathsf{p}(\mathbf{n},t)$ can be computed from $\mathsf{q}(\mathbf{n},t)$ using the expression $\mathsf{p} = \mathsf{q} + \frac{1}{2}\left(\mathsf{P}(\mathsf{u}-\mathsf{u}_r) - \mathsf{P}(\mathsf{u}_r)\right)$.

4.2.2 Lattice Green's function technique

In this section we provide an overview of the LGF techniques [1, 2] and some extensions used to solve inhomogeneous, elliptic *difference* equations relevant to incompressible flows on unbounded domains with immersed surfaces. We consider the representative problem of the *discrete* (7-pt) scalar Poisson equation

$$[\mathsf{L}\mathsf{x}](\mathbf{n}) = \mathsf{y}(\mathbf{n}), \quad supp(\mathsf{f}) \subseteq D, \tag{4.6}$$

where both x and y belong to either $\mathbb{R}^{\mathcal{C}}$ or $\mathbb{R}^{\mathcal{V}}$, and D is a bounded region in $\mathbb{Z}^3$. The procedure for solving Eq. (4.6) using the LGF of L is analogous to the procedure for solving free-space Poisson problems using the fundamental solution of ∇^2, i.e. $-1/(4\pi|\mathbf{x}|)$. The solution to Eq. (4.6) is given by the *discrete* convolution

$$\mathsf{u}(\mathbf{n}) = [\mathsf{G}_{\mathsf{L}} * \mathsf{f}](\mathbf{n}) = \sum_{\mathbf{m}\in D} \mathsf{G}_{\mathsf{L}}(\mathbf{n}-\mathbf{m})\mathsf{f}(\mathbf{m}), \tag{4.7}$$

where $\mathsf{G}_{\mathsf{L}} : \mathbb{Z}^3 \mapsto \mathbb{R}$ denotes the fundamental solution, or LGF, of L [1, 11].

The present formulation computes the actions of L^{-1}, $\mathsf{E}(\alpha)$, and $\mathsf{K}^{-1} = [\mathsf{E}(-\alpha)\mathsf{L}]^{-1}$ by evaluating expressions analogous to Eq. (4.7) for the LGFs G_{L}, $\mathsf{G}_{\mathsf{E}}(\alpha)$, and $\mathsf{G}_{\mathsf{K}}(\alpha)$, where $\mathsf{E}(\alpha)$ is the operator exponential of L that is used as a viscous integrating factor in the discussion of Section 4.2.3. Although the action of K^{-1} can be computed in two steps either as $[\mathsf{E}(\alpha)]\mathsf{L}^{-1}$ or $\mathsf{L}^{-1}[\mathsf{E}(\alpha)]$, significant operation count reductions are obtained by directly using $\mathsf{G}_{\mathsf{K}}(\alpha)$ to evaluate solutions to problems with source and target regions that are limited to a small neighborhood around the immersed surface, e.g. the support region of discrete delta functions. This follows from the fact that the target and source regions of the first and second operator of either two-step approach must be enlarged in each direction approximately by the size of the support of $\mathsf{G}_{\mathsf{E}}(\alpha)$.

The 3D LGF-FMM [1] is used by the present implementation to evaluate discrete LGF convolutions of the form given by Eq. (4.7). The LGF-FMM method is a kernel-independent, interpolation-based FMM for solving elliptic, constant-coefficient

difference equations on unbounded Cartesian grids to prescribed tolerances in linear algorithmic complexity. Computational rates and parallel scaling comparable to existing fast 3D free-space Poisson solvers have been demonstrated for the case of G_L [1]. LGF-specific computational considerations for problems involving G_L and $\mathsf{G}_\mathsf{E}(\alpha)$ are discussed in [1] and [2], respectively. Here, we note that the fast decay of $\mathsf{G}_\mathsf{E}(\alpha)$ allows for the nearly identical far-field treatment of $\mathsf{G}_\mathsf{K}(\alpha)$ compared to G_L, since for sufficiently large values $|\mathbf{n}|$ the asymptotic expansions of $\mathsf{G}_\mathsf{L}(\mathbf{n})$ [1, 18] are also accurate approximations to $[\mathsf{G}_\mathsf{K}(\alpha)](\mathbf{n})$.[3] Numerical procedures for computing $\mathsf{G}_\mathsf{K}(\alpha)$, and expressions in terms of Fourier and Bessel integrals for all the aforementioned LGFs are included in Appendix 4.A.

4.2.3 Time integration

Modifications to the IF-HERK time integration technique for incompressible flows [2] necessary to include immersed surfaces are discussed in this section. We begin by considering the discrete integrating factor $\mathsf{E}_\mathcal{Q}(\alpha)$ corresponding to the solution operator of the discrete heat equation $d\mathsf{h}/dt = \kappa\mathsf{L}_\mathcal{Q}\mathsf{h}$ with $\mathsf{h}(\mathbf{n},t) \to 0$ as $|\mathbf{n}| \to \infty$.[4] Taking u to be known at time τ and using the integrating factor $\mathsf{H}_\mathcal{Q}(t) = \mathsf{E}_\mathcal{Q}\left(\frac{t-\tau}{(\Delta x)^2\mathrm{Re}}\right)$, an equivalent expression for Eq. (4.5) for $t \geq \tau$ is given by

$$\frac{d\mathsf{v}}{dt} + \left[\mathsf{H}_\mathcal{F}(t)\right]\tilde{\mathsf{N}}\left(\left[\mathsf{H}_\mathcal{F}^{-1}(t)\right]\mathsf{v}, t\right) = -\mathsf{G}\mathsf{b} - \left[\mathsf{H}_\mathcal{F}(t)\right]\left[\mathscr{I}(t)\right]^\dagger\tilde{\mathsf{f}}, \tag{4.8a}$$

$$\mathsf{G}^\dagger\mathsf{v} = 0, \tag{4.8b}$$

$$\left[\mathscr{I}(t)\right]\left[\mathsf{H}_\mathcal{F}^{-1}(t)\right]\mathsf{v} = \mathsf{u}, \tag{4.8c}$$

---

[3] The value of $[\mathsf{G}_\mathsf{E}(\alpha)](\mathbf{n})$ decays faster than any exponential as $|\mathbf{n}| \to \infty$ for a given $\alpha \geq 0$ [2]. For typical flows, e.g. numerical experiments discussed in Section 4.5 and [2], $\alpha < \Delta t/\left((\Delta x)^2\mathrm{Re}\right) \lesssim 1$; for all $\alpha \in [0,1]$ and $|\mathbf{n}| > 10$, the values of $|[\mathsf{G}_\mathsf{E}(\alpha)](\mathbf{n})| \,/\, |[\mathsf{G}_\mathsf{E}(\alpha)](\mathbf{0})|$ and $|[\mathsf{G}_\mathsf{K}(\alpha)](\mathbf{n}) - \mathsf{G}_\mathsf{L}(\mathbf{n})| \,/\, |\mathsf{G}_\mathsf{L}(\mathbf{0})|$ are less than 10^{-7} and 10^{-9}, respectively.

[4] The solution to $d\mathsf{h}/dt = \kappa\mathsf{L}_\mathcal{Q}\mathsf{h}$ with $\mathsf{h}(\mathbf{n},t) \to 0$ as $|\mathbf{n}| \to \infty$ is given by $\mathsf{h}(\mathbf{n},t) = \left[\mathsf{E}_\mathcal{Q}\left(\kappa(t-\tau)/(\Delta x)^2\right)\mathsf{h}_\tau\right](\mathbf{n},t)$, where $\mathsf{h}_\tau(\mathbf{n}) = \mathsf{h}(\mathbf{n},\tau)$. An expression for $\mathsf{E}_\mathcal{Q}$ in terms of the Fourier series operator $\mathfrak{F}_\mathcal{Q}$ and the spectrum $\sigma_\mathcal{Q}^\mathsf{L}(\boldsymbol{\xi})$ of $(\Delta x)^2\mathsf{L}_\mathcal{Q}$ is given by $\mathsf{E}_\mathcal{Q}(\alpha) = \mathfrak{F}_\mathcal{Q}^{-1}\exp(\alpha\sigma_\mathcal{Q}^\mathsf{L})\mathfrak{F}_\mathcal{Q}$ [2].

where $\mathsf{v} = [\mathsf{H}_{\mathcal{F}}(t)]\mathsf{u}$ and $\mathsf{b} = [\mathsf{H}_{\mathcal{C}}(t)]\mathsf{d}$. The effect of the $\mathsf{H}_{\mathcal{F}}(t)$ and $\mathsf{H}_{\mathcal{F}}^{-1}(t)$ on the regularized forces and the no-slip constraint *cannot* be absorbed into $\tilde{\mathsf{f}}$ and u since, in general, there does *not* exist an operator $\mathfrak{M}(t) : \mathcal{R}^{\Gamma} \mapsto \mathcal{R}^{\Gamma}$ such that $[\mathsf{H}_{\mathcal{F}}(t)][\mathscr{I}(t)]^{\dagger} = [\mathscr{I}(t)]^{\dagger}[\mathfrak{M}(t)]$. This implies that, even for the case of stationary immersed surfaces, the constraint operators explicitly depend on t. The explicit temporal dependence of the constraint operators changes the character of the present system of differential algebraic equations (DAEs), i.e. Eq. (4.8), compared to the analogous system of DAEs formulated in [2], i.e. Eq. (4.8a) and (4.8b) with $\mathsf{f} = 0$. As a result, the simplifications to the HERK order-conditions for the case of trivial immersed surfaces [2] need to be modified in order to develop schemes for Eq. (4.8) that achieve a prescribed order of accuracy.

HERK methods [63, 66] are used to integrate DAE systems of index 2

$$\frac{dy}{dt} = f(y,z), \quad g(y) = 0, \tag{4.9}$$

where the product of partial derivatives $g_y(y)f_z(y,z)$ is non-singular in a neighborhood about the solution, and z is an unknown that must be computed so that y satisfies $g(y) = 0$. For the case of Eq. (4.8), or equivalently Eq. (4.5), the operator $g_y(y)f_z(y,z)$ is invertible if and only if $\overline{\mathsf{D}}\mathsf{G}$ and $[\mathscr{I}(t)](\mathsf{I} - \mathsf{G}(\overline{\mathsf{D}}\mathsf{G})^{-1}\overline{\mathsf{D}})[\mathscr{I}(t)]^{\dagger}$ are invertible. The invertibility of $\overline{\mathsf{D}}\mathsf{G} = \mathsf{L}_{\mathcal{C}}$ follows from taking u and d to tend to zero as $|\mathbf{n}| \to \infty$ [2], and the invertibility of $[\mathscr{I}(t)]\left(\mathsf{I} - \mathsf{G}\mathsf{L}_{\mathcal{C}}^{-1}\overline{\mathsf{D}}\right)[\mathscr{I}(t)]^{\dagger}$ is inferred, for practical flows, from the representative numerical experiments of Section 4.5 and from the discussions of similar operators arising in other IB-DLM methods [4, 69, 93, 94].[5] By considering Eq. (4.8) in its autonomous form[6] with $y = [\mathsf{v}, t]$ and

[5]The operators that arise in the discretizations [4, 69, 93, 94] are of the form $\mathsf{B} = [\mathscr{I}(t)]\mathsf{A}(\mathsf{I} - \mathsf{G}\mathsf{L}^{-1}\overline{\mathsf{D}})[\mathscr{I}(t)]^{\dagger}$, where A is an operator resulting from to implicit treatment of the viscous term. Previous numerical experiments of [4, 69, 94] have found B to be well-conditioned and solvable under grid refinement for sufficiently well-spaced IB markers.

[6]The non-autonomous system Eq. (4.8) can be written as an equivalent autonomous system by taking t to be part of the solution variables, e.g. $y = [\mathsf{v}, t]$, and by augmenting the system of DAEs by including the trivial ODE $dt/dt = 1$.

$z = [\mathsf{b}, \mathsf{f}]$ reveals that the corresponding partial derivatives f_z and g_y depend on t but do not depend either u or z. Simplifications to the general HERK order-conditions for the special case of solely time-dependent f_z and g_y are well-described in [65–67]. Tableaus and expected orders of accuracy for four representative schemes that are used to perform the numerical experiments in Section 4.5 are provided in Appendix 4.B.

Next, we consider the IF-HERK algorithm obtained using a s-stage HERK scheme with *shifted* coefficients $\tilde{a}_{i,j}$ and *shifted* nodes $\tilde{c}_i$ to integrate Eq. (4.8) from $t_k = k\Delta t$ to $t_{k+1} = (k+1)\Delta t$. The present IF-HERK algorithm is constructed by including the additional IB terms to the IF-HERK algorithm of [2]. Introducing the auxiliary variables

$$\mathsf{u}_k^i(\mathbf{n}) = \left[\mathsf{E}_{\mathcal{F}}\left(\tfrac{-\tilde{c}_i\Delta t}{(\Delta x)^2\mathrm{Re}}\right)\right]\mathsf{v}_k^i(\mathbf{n}),\ \mathsf{d}_k^i(\mathbf{n}) = \left[\mathsf{E}_{\mathcal{F}}\left(\tfrac{-\tilde{c}_i\Delta t}{(\Delta x)^2\mathrm{Re}}\right)\right]\mathsf{b}_k^i(\mathbf{n}),\ \forall i \in [1,s], \tag{4.10}$$

and grouping the constraint variables, RHSs, and operators

$$\lambda_k^i = \begin{bmatrix} \mathsf{d}_k^i \\ \tilde{\mathsf{f}}_k^i \end{bmatrix},\ \zeta_k^i = \begin{bmatrix} 0 \\ \mathfrak{u}(t_k^i) \end{bmatrix},\ Q_k^i = \begin{bmatrix} \mathsf{G} & [\mathscr{I}(t_k^i)]^\dagger \end{bmatrix},\ \forall i \in [1,s], \tag{4.11}$$

the k-th time-step of the time integration algorithm, IF-HERK(u_k, t_k), is as follows:

1. *initialize*: set $\mathsf{u}_k^0 = \mathsf{u}_k$ and $t_k^0 = t_k$.

2. *multi-stage*: for $i = 1, 2, \ldots, s$, solve the linear system

$$\begin{bmatrix} (\mathsf{H}_{\mathcal{F}}^i)^{-1} & Q_k^{i-1} \\ (Q_k^i)^\dagger & 0 \end{bmatrix} \begin{bmatrix} \mathsf{u}_k^i \\ \hat{\lambda}_k^i \end{bmatrix} = \begin{bmatrix} \mathsf{r}_k^i \\ \zeta_k^i \end{bmatrix}, \tag{4.12}$$

   where

$$\mathsf{H}_{\mathcal{F}}^i = \mathsf{E}_{\mathcal{F}}\left(\tfrac{(\tilde{c}_i - \tilde{c}_{i-1})\Delta t}{(\Delta x)^2\mathrm{Re}}\right), \quad \mathsf{r}_k^i = \mathsf{h}_k^i + \Delta t \sum_{j=1}^{i-1} \tilde{a}_{i,j}\mathsf{w}_k^{i,j} + \mathsf{g}_k^i, \tag{4.13}$$

$$\mathsf{g}_k^i = -\tilde{a}_{i,i}\Delta t\ \tilde{\mathsf{N}}\left(\mathsf{u}_k^{i-1}, t_k^{i-1}\right), \quad t_k^i = t_k + \tilde{c}_i\Delta t. \tag{4.14}$$

Variables h_k^i and $\mathsf{w}_k^{i,j}$ are recursively computed for $i > 1$ and $j > i$ using

$$\mathsf{h}_k^i = \mathsf{H}_{\mathcal{F}}^{i-1}\mathsf{h}_k^{i-1}, \quad \mathsf{h}_k^1 = \mathsf{u}_k^0 \tag{4.15}$$

$$\mathsf{w}_k^{i,j} = \mathsf{H}_{\mathcal{F}}^{i-1}\mathsf{w}_k^{i-1,j}, \quad \mathsf{w}_k^{i,i} = (\tilde{a}_{i,i}\Delta t)^{-1}\left(\mathsf{g}_k^i - Q_k^{i-1}\hat{\lambda}_k^i\right). \tag{4.16}$$

3. *finalize*: set $\mathsf{u}_{k+1} = \mathsf{u}_k^s$, $\lambda_{k+1} = (\tilde{a}_{s,s}\Delta t)^{-1}\hat{\lambda}_k^s$, and $t_{k+1} = t_k^s$.

Solving Eq. (4.12) is expected to dominate the overall run-time cost of each IF-HERK stage, and is discussed in the next section.

4.3 Fast linear system solver

4.3.1 Nested projection technique

In this section we demonstrate that Eq. (4.12) is efficiently solved using an operator-block decomposition that is analogous to standard matrix-block LU decompositions. Unlike traditional projection and fractional-step techniques [106, 107], which can be viewed as approximate LU decompositions [70], the present approach is an *exact*, i.e. free of operator approximations, projection technique [72]. As a result, the method is free of "splitting errors" and does not make use of artificial pressure boundary conditions [2, 69, 70, 72]. In contrast to 2D discrete null-space (discrete stream-function) methods [69, 72], we do not cast the discrete velocity-pressure equations into equivalent discrete streamfunction-vorticity equations since for 3D flows both formulation require solutions to an equal number of discrete Poisson problems but these are *scalar* problems in the former and *vector* problems in the latter. The issue of which formulation is computationally faster is less obvious in the finite computational domain algorithm, discussed in Section 4.4, since the discrete velocity in the velocity-pressure formulation is periodically "refreshed" from the discrete vorticity by solving a discrete *vector* Poisson problem. The arguments by the LGF flow solver [2] supporting the velocity-pressure formulation are readily extended to the present IB formulation.

We consider Eq. (4.12) written in terms of both Lagrange multipliers d_k^i and $\tilde{\mathsf{f}}_k^i$,

$$M_k^i \begin{bmatrix} \mathsf{u}_k^i \\ \hat{\mathsf{d}}_k^i \\ \hat{\mathsf{f}}_k^i \end{bmatrix} = \begin{bmatrix} (\mathsf{H}_{\mathcal{F}}^i)^{-1} & \mathsf{G} & (\mathscr{I}_k^{i-1})^\dagger \\ \mathsf{G}^\dagger & 0 & 0 \\ \mathscr{I}_k^i & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathsf{u}_k^i \\ \hat{\mathsf{d}}_k^i \\ \hat{\mathsf{f}}_k^i \end{bmatrix} = \begin{bmatrix} \mathsf{r}_k^i \\ 0 \\ \mathsf{u}_k^i \end{bmatrix}, \tag{4.17}$$

where $\hat{\mathsf{d}}_k^i/\mathsf{d}_k^i = \hat{\mathsf{f}}_k^i/\tilde{\mathsf{f}}_k^i = \tilde{a}_{s,s}$, $\mathsf{u}_k^i = \mathsf{u}(t_k^i)$, and $\mathscr{I}_k^i = \mathscr{I}(t_k^i)$. In general, M_k^i is *not* symmetric and cannot be symmetrized by rescaling $\hat{\mathsf{f}}_k^i$ since the image of $(\mathscr{I}_k^{i-1})^\dagger$ and of $(\mathscr{I}_k^i)^\dagger$ are different. This is in contrast to similar IB methods, e.g. [69, 93, 94], which solve symmetric systems of equations analogous to Eq. (4.12). The asymmetry of M_k^i is inherent to HERK integrations of DAE system of index 2 with time-dependent constraint operators [65, 66].[7] Special cases of interest for which M_k^i reduces to a symmetric operator include flows around rigid surfaces and the i-th stage of HERK schemes with $\tilde{c}_{i-1} = \tilde{c}_i$. Lastly, we call attention to the fact that the DAE index 2 conditions discussed in Section 4.2.3 ensure the solvability of Eq. (4.17) [66, 109], but emphasize that these conditions are satisfied only if $[\mathscr{I}(t)]\left(\mathsf{I} - \mathsf{G}\mathsf{L}_{\mathcal{C}}^{-1}\overline{\mathsf{D}}\right)[\mathscr{I}(t)]^\dagger$ is invertible. The invertibility of this operator is demonstrated for a few practical flows in Section 4.5. Additionally, the invertibility of similar operators arising in other IB-DLM formulations has been discussed and numerically demonstrated for several practical flows by previous IB methods [4, 69, 93, 94].

Solutions to Eq. (4.17) obtained from an operator-block LU decomposition of M_k^i

[7] Similar asymmetries in the $(1,3)$ and $(3,1)$ operators are expected for operators analogous to M_k^i arising from other standard semi-explicit single- or multi-step integration schemes for DAE systems of index 2, e.g. [63] and references therein. For example, the semi-explicit two-step Adams-Bashforth method [108] solves Eq. (4.9) as $y_{k+1} = y_k + \frac{\Delta t}{2}\left(3f(y_k, z_k) - f(y_{k-1}, z_{k-1})\right)$, where the unknown z_k is computed so that $0 = g(y_{k+1})$. Here, the regularization operator $[\mathscr{I}(t_k)]^\dagger$ acting on the unknown body forces included in $f(y_k, z_k)$ is evaluated at an earlier time (t is part of y) compared to the interpolation operator $\mathscr{I}(t_{k+1})$ included in $g(y_{k+1})$.

can be written in the projection-like form

$$\begin{aligned} \mathsf{A}^i\mathsf{u}^* &= \mathsf{r}_k^i & \hat{\mathsf{f}}_k^i &= \mathsf{f}^* \\ \mathsf{B}^i\mathsf{d}^* &= \mathsf{G}^\dagger\mathsf{u}^* & \hat{\mathsf{d}}_k^i &= \mathsf{d}^* - (\mathsf{B}^i)^{-1}\mathsf{G}^\dagger(\mathsf{A}^i)^{-1}(\mathscr{I}_k^{i-1})^\dagger\hat{\mathsf{f}}_k^i, \\ \mathfrak{C}_k^i\mathsf{f}^* &= \mathscr{I}_k^i[\mathsf{u}^* - (\mathsf{A}^i)^{-1}\mathsf{G}\mathsf{d}^*] - \mathsf{u}_k^i & \mathsf{u}_k^i &= \mathsf{u}^* - (\mathsf{A}^i)^{-1}[\mathsf{G}\hat{\mathsf{d}}_k^i + (\mathscr{I}_k^{i-1})^\dagger\hat{\mathsf{f}}_k^i] \end{aligned} \tag{4.18}$$

where

$$\mathsf{A}^i = (\mathsf{H}_{\mathcal{F}}^i)^{-1}, \quad \mathsf{B}^i = \mathsf{G}^\dagger(\mathsf{A}^i)^{-1}\mathsf{G}, \tag{4.19}$$

$$\mathfrak{C}_k^i = \mathscr{I}_k^i(\mathsf{A}^i)^{-1}[\mathsf{I}_{\mathcal{F}} - \mathsf{G}(\mathsf{B}^i)^{-1}\mathsf{G}^\dagger(\mathsf{A}^i)^{-1}](\mathscr{I}_k^{i-1})^\dagger, \tag{4.20}$$

and $\mathsf{I}_{\mathcal{F}}$ is the identity operator for $\mathbb{R}^{\mathcal{F}}$. Taking in account the mimetic, orthogonality, and commutativity properties of the discrete grid operators, the nested projection method Eq. (4.18) is reduced to the more computationally convenient form

$$\mathsf{L}_{\mathcal{C}}\mathsf{d}^* = -\mathsf{G}^\dagger\mathsf{r}_k^i \tag{4.21a}$$

$$\mathfrak{C}_k^i\hat{\mathsf{f}}_k^i = \mathscr{I}_k^i\mathsf{H}_{\mathcal{C}}^i[\mathsf{r}_k^i - \mathsf{G}^\dagger\mathsf{d}^*] - \mathsf{u}_k^i \tag{4.21b}$$

$$\hat{\mathsf{d}}_k^i = \mathsf{d}^* + \mathsf{L}_{\mathcal{C}}^{-1}\mathsf{G}^\dagger(\mathscr{I}_k^{i-1})^\dagger\hat{\mathsf{f}}_k^i \tag{4.21c}$$

$$\mathsf{u}_k^i = \mathsf{H}_{\mathcal{F}}^i[\mathsf{r}_k^i - \mathsf{G}\hat{\mathsf{d}}_k^i - (\mathscr{I}_k^{i-1})^\dagger\hat{\mathsf{f}}_k^i]. \tag{4.21d}$$

Similar considerations are used to reduce the *force Schur complement* operator $\mathfrak{C}_k^i$ to the more computationally efficient form

$$\mathfrak{C}_k^i = \mathscr{I}_k^i[\mathsf{H}_{\mathcal{F}}^i + \mathsf{G}(\mathsf{K}_{\mathcal{C}}^i)^{-1}\mathsf{G}^\dagger](\mathscr{I}_k^{i-1})^\dagger, \tag{4.22}$$

where $\mathsf{K}_{\mathcal{C}} = (\mathsf{H}_{\mathcal{C}}^i)^{-1}\mathsf{L}_{\mathcal{C}}$. As an aside, the physical interpretation of $\mathfrak{C}_k^i$ and its similarity to analogous operators arising in other IB methods, e.g. [69, 94], are facilitated by writing the operator as $\mathscr{I}_k^i\mathsf{H}_{\mathcal{F}}^i\mathsf{S}(\mathscr{I}_k^{i-1})^\dagger$, where $\mathsf{S} = \mathsf{I}_{\mathcal{F}} - \mathsf{G}\mathsf{L}_{\mathcal{F}}^{-1}\overline{\mathsf{D}} = -\overline{\mathsf{C}}\mathsf{L}_{\mathcal{E}}^{-1}\mathsf{C}$ is the orthogonal discrete divergence-free projection operator.

Efficient computations of Eq. (4.21) make use of the flexible source and target regions of the LGF-FMM. This is particularly relevant to computations of $\mathfrak{C}_k^i$, $\mathscr{I}_k^i\mathsf{H}_{\mathcal{C}}^i$, and $\mathsf{L}_{\mathcal{C}}^{-1}\mathsf{G}^\dagger(\mathscr{I}_k^{i-1})^\dagger$ since the target and source regions can be limited to a small neighborhoods about the support of the discrete delta functions. Since the IB markers

are confined to a lower dimensional sub-region of the overall computational grid, significant operation count reductions are expected when compared to schemes that do not limit the source and target regions of elliptic problems.[8] Formal definitions for the various sub-regions of the adaptive block-wise grid used in the present implementation are discussed in Section 4.4.1.

With the exception of $\hat{\mathfrak{f}}_k^i$, every term in Eq. (4.21) is efficiently computed either using the point-operator representation of discrete operators or using the LGF-FMM. The remaining problem of efficient techniques for solving equations of the form $\mathfrak{C}_k^i \hat{\mathfrak{f}} = \mathfrak{r}$ is discussed in the following section.

4.3.2 Force Schur complement solvers

In this section we consider solutions to $\mathfrak{C}_k^i \hat{\mathfrak{f}} = \mathfrak{r}$ obtained using either iterative methods or dense linear algebra techniques. We clarify that although the discussion of this section describes techniques for solving flows around immersed surfaces with general prescribed motions, the present implementation only considers the case of rigid surfaces. Here, it is assumed that for asymptotically large problems the number of Lagrangian points, N_L, scales like $N_E^{\frac{2}{3}}$, where N_E is the total number of Eulerian grid cells used in the finite computational domain. Additionally, the action of $\mathfrak{C}_k^i$ is taken to be evaluated in $\mathcal{O}(N_L)$ by limiting operations of the LGF-FMM solver to a few grid cells near the immersed surface.

We begin by considering the cases resulting in $\mathscr{I}_k^{i-1} = \mathscr{I}_k^i$, which include flows around rigid immersed surfaces and RK stages with $\tilde{c}_{i-1} = \tilde{c}_i$. For these cases $\mathfrak{C}_k^i$ is symmetric positive-definite (SPD), which makes the conjugate gradient (CG) method the natural iterative solver for $\mathfrak{C}_k^i \hat{\mathfrak{f}} = \mathfrak{r}$. This iterative method is used in

[8] For test cases included in Section 4.5, the typical computational time for Eq. (4.21c) is less than 50% of that for Eq. (4.21a). Although Eq. (4.21c) typically requires significantly fewer than 50% of the number of operations required by Eq. (4.21a), parallel load balancing aiming to optimize Eq. (4.21a) (most computationally expensive step) results in a parallel work imbalance when computing Eq. (4.21c).

[69] to solve for the body forces from similar systems of equations, but, in contrast to the present technique, each iteration requires $\mathcal{O}(N_E \log N_E)$ instead of $\mathcal{O}(N_L)$. Estimating the number of iterations to scale like $N_L^{\frac{1}{2}}$ [69], we expect the body forces for each RK stage to be computed in $\mathcal{O}(N_L^{\frac{3}{2}})$ operations, that is to say, $\mathcal{O}(N_E)$ operations. As a result, the overall *operation count* of the nested projection technique, i.e. Eq. 4.21, is $\mathcal{O}(N_E)$.

In order to estimate the *computation time* of parallel algorithms it is necessary to account for the parallel scaling of the technique. Similar to most parallel FMMs, the LGF-FMM requires a minimum number of grid cells per processor, γ_{eff}, in order to sustain reasonable parallel efficiencies, e.g. greater than 80%, as the number of processes, N_p, increases.[9] In practice, we expect an approximately constant $N_E/N_p \approx \gamma > \gamma_{\text{eff}}$, which implies that the action of $\mathfrak{C}_k^i$ is computed with $N_L/N_p \approx \alpha N_L/N_E \sim \mathcal{O}(N^{-\frac{1}{3}})$ grid cells per processor. For sufficiently large problems, N_L/N_p will be less than γ_{eff} and continue to decrease as the problem size increases; thus, the CG body force solver is not expected to scale well.[10] Provided γ is held constant, we expect the *computation time* of evaluating $\mathfrak{C}_k^i$ to be $\mathcal{O}(N_E)$ and of the CG method be to $\mathcal{O}(N_L^{\frac{1}{2}} N_E)$ or, equivalently, $\mathcal{O}(N_L^2) \sim \mathcal{O}(N_E^{\frac{4}{3}})$.

An alternative approach is to use dense linear algebra to build and factor the matrix corresponding to $\mathfrak{C}_k^i$, and use its factored form to solve for $\hat{\mathfrak{f}}$. For the case of rigid surfaces, the construction ($\mathcal{O}(N_L^2)$ operations) and factorization ($\mathcal{O}(N_L^3)$ operations) of the matrix $\mathbf{C} = [\mathfrak{C}_k^i]$ only needs to be performed once per simulation as a pre-processing step. In fact, the factored form of $\mathbf{C}$ can be reused to compute flows that share the same geometry, RK tableau, and value of $\frac{\Delta t}{(\Delta x)^2 \text{Re}}$. Here, the Cholesky decomposition of the $\mathbf{C}$, i.e. $\mathbf{C} = \mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ is a lower-triangular matrix, is

[9] Here, we define the parallel efficiency as $T_{p=1} / \left(T_{p=N_p} N_p\right)$, where $T_{p=n}$ is the wall-time of the algorithm.

[10] Most simulations performed in Section 4.5 were performed with $\gamma \approx \gamma_{\text{eff}}$ and with $N_L/N_E < 10^{-3}$ (for fully developed wakes). For these test cases, the parallel efficiency of evaluating $\mathfrak{C}_k^i$ can be estimated to be less than 10%.

computed in parallel using the ScaLAPACK library [110]. Backward and forward substitutions can be used to evaluate $[\hat{\mathfrak{f}}] = \mathbf{f} = \mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{r}$ in $\mathcal{O}(N_L^2)$ operations, but the inherent sequential nature of backward substitution limits the parallel speed-up of this approach. We circumvent this potential bottleneck by explicitly computing $\mathbf{W} = \mathbf{L}^{-1}$ ($\mathcal{O}(N_L^3)$ operations) as part of the pre-processing step, and solve for $\mathbf{f}$ by evaluating $\mathbf{y} = \mathbf{W}\mathbf{r}$ and $\mathbf{f} = \mathbf{W}^T\mathbf{y}$ using parallel matrix-vector multiplications ($\mathcal{O}(N_L^2)$ operations).[11] By distributing the columns or rows of $\mathbf{W}$ and maintaining a local copy of $\mathbf{r}$, the parallel matrix-vector multiplication is expected to achieve nearly perfect parallel efficiency for $N_p \ll N_L$, which is typical for most practical simulations. As a point of reference, the largest problem considered in Section 4.5, a sphere defined by approximately 8×10^4 IB markers, the average fraction of time spent evaluating $\hat{\mathfrak{f}}$ compared to the rest of Eq. (4.21) was less than 3%.

Asymptotically, the computation time factoring $\mathbf{C}$ and inverting $\mathbf{L}$, and the memory requirements associated with storing $\mathbf{W}$ are expected to render the pre-processing technique less efficient than the CG solver and potentially unfeasible on some computational resources. Yet, for many practical problems, such as the test cases of Section 4.5, the pre-processing technique is expected to take a small fraction of the overall computation time and memory, and to yield significantly faster body forces solutions compared to the CG method.[12]

---

[11] Although possible reductions, up to a factor of two, in the computation time of $\mathbf{f}$ are achieved by pre-computing $\mathbf{W}^T\mathbf{W}$ and evaluating $\mathbf{f}$ using a single parallel matrix-vector multiplication, this approach is expected to lead to a greater amplification of numerical errors compared to the two step approach described in the main text.

[12] For the largest simulation in Section 4.5, i.e. sphere at Re = 3700, pre-processing only required approximately 10% of the total computation time, with less than 10% of the pre-processing time dedicated to factoring $\mathbf{C}$ and inverting $\mathbf{L}$. For this test case, the time spent evaluating $\mathfrak{C}_k^i$ (roughly equal to the time a single CG iteration) is approximately 60% of the time spent computing $\mathbf{f} = \mathbf{W}^T(\mathbf{W}\mathbf{r})$. Estimating the number of CG iterations to reduce the initial residual by ϵ to be $\frac{1}{2}\sqrt{\kappa}\ln\frac{2}{\epsilon}$ [111], and taking $\epsilon = 0.1$ (assumes a good initial guess) and $\kappa \simeq 1.1 \times 10^3$ (computed condition number of $\mathbf{C}$), we expect the CG solver to require 50 iterations and, as a result, to be 30 times slower than $\mathbf{f} = \mathbf{W}^T(\mathbf{W}\mathbf{y})$.

The general case of immersed surfaces with prescribed motion requires additional solution techniques since, for at least one RK stage, $\mathfrak{C}_k^i$ is only approximately symmetric, i.e. $\mathscr{I}_k^{i-1} = \mathscr{I}_k^i + \mathcal{O}(\Delta t)$. Efficient parallel implementations of Krylov solvers such as GMRES and BiCGSTAB, and their "flexible" extensions [112, 113], can be used to solve for $\hat{\mathfrak{f}}$ for the case of non-symmetric $\mathfrak{C}_k^i$.[13] Another approach that takes advantage of the efficiency of the CG method is to symmetrize M_k^i by introducing an $\mathcal{O}(\Delta t)$ error so that $\mathscr{I}_k^{i-1}$ is replaced by $\mathscr{I}_k^i$. Although this $\mathcal{O}(\Delta t)$ approximation results in solutions that still satisfy the discrete divergence-free and non-slip constraints, further analysis is required to determine its effect on the global (entire integration period) accuracy and stability of the solution.

4.4 Adaptive computational domain

4.4.1 Block-wise adaptive grid

The present incompressible flow solver is implemented using the block-wise adaptive grid of LGF flow solver [2]. When coupled with the LGF techniques discussed in Section 4.2.2, this approach has the advantage of limiting the computational domain to a small, finite region of the unbounded domain that efficiently accommodates temporally evolving solutions by dynamically adding and removing blocks. Errors concentrated near the finite boundaries that result from neglecting values outside the finite computation domain are prevented from significantly contaminating the solution by padding the interior domain with buffer grid cells and periodically computing ("refreshing") the algebraically-decaying velocity perturbation from the exponentially-decaying vorticity [2],

$$\mathsf{w} \leftarrow \mathsf{C}\mathsf{u}, \quad \mathsf{s} \leftarrow \mathsf{L}^{-1}\mathsf{w}, \quad \mathsf{u} \leftarrow \mathsf{C}^{\dagger}\mathsf{s}. \tag{4.23}$$

[13] Flexible Krylov methods are often used to iteratively solve preconditioned linear systems that require additional ("inner") iterations to approximate the action of the preconditioner. For the present case, one possible preconditioner is the symmetrized version of $\mathfrak{C}_k^i$ obtained by approximating $\mathscr{I}_k^{i-1}$ as $\mathscr{I}_k^i$, which in turn allows for efficient "inner" CG iterations.

Estimates for the number of time-steps, $Q_{\max}$, before u needs to be refreshed from w are provided in [2], but we note here that for typical schemes multiple time-steps, e.g. $Q_{\max} \gtrsim 10$, can elapse before this refresh operation is required. In the following discussion, we highlight additional key features of this approach and extend the base method [2] to efficiently incorporate the immersed surfaces.

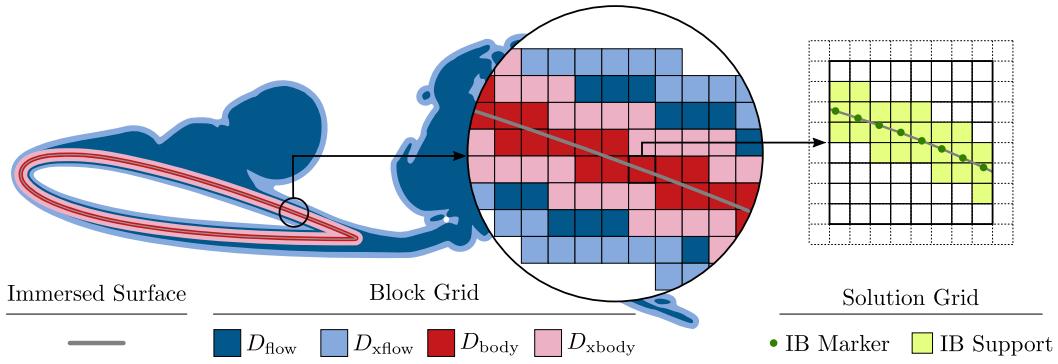

Figure 4.1: Depiction of a 2D finite computational grid, *solution grid*, comprised of blocks arranged on a Cartesian grid, *block grid*. Distant view of nested sub-domains $D_{\text{flow}} \subset D_{\text{xflow}}$ and $D_{\text{body}} \subset D_{\text{xbody}}$, where $D_{\text{body}} \subseteq D_{\text{flow}}$ and $D_{\text{xbody}} \subseteq D_{\text{xbody}}$ (*left*). Zoomed-in view illustrating the union of blocks used to define the domain (*middle*). Magnified view of the finite staggered Cartesian grid and IB markers associated with a single block (*right*). Dashed cells surrounding the interior cells of the isolated block correspond to ghost cells used to facilitate the parallel implementation.

We consider partitioning the unbounded *solution grid* into blocks arranged on a Cartesian *block grid*. Approximate values for each term of the IF-HERK and the velocity refresh algorithms are computed by limiting the source (domain) and target (co-domain) regions of discrete operators to the sub-domains depicted in Figure 4.1. These sub-domains are defined as follows:

- *Flow domain* D_{flow}: union of blocks containing non-negligible source terms for any discrete Poisson problem solved in the present formulation, i.e. Eq. (4.21a)–(4.21c) and velocity refresh. Flow quantities on D_{flow} are taken to be accurate approximations of the corresponding flow quantities that would have been obtained by numerically operating on the entire unbounded do-

main.[14]

- *Expanded flow domain* D_{xflow}: union of blocks that are at most $N_{\text{buf}}^{\text{flow}}$ blocks in any direction from any block in D_{flow}. The value of $N_{\text{buf}}^{\text{flow}}$ is determined using the procedure discussed in [2] so that the error in D_{flow} remains below a prescribed threshold for the time-steps between velocity refreshes.

- *Body blocks* D_{body}: union of blocks containing grid cells that are at most one grid cell away from the support of any discrete delta function during a single time-step. This implies, for example, that all non-zero values of $\mathsf{G}^{\dagger}[\mathscr{I}^{\dagger}(t)]$ are contained within D_{body}.

- *Expanded body domain* D_{xbody}: union of blocks that are at most $N_{\text{buf}}^{\text{body}} \leq N_{\text{buf}}^{\text{flow}}$ blocks in any direction from any block in D_{body}. We limit our attention to the case of $N_{\text{buf}}^{\text{body}} = N_{\text{buf}}^{\text{flow}} = N_{\text{buf}}$, and note that the subsequent discussions readily extend to the general case of $N_{\text{buf}}^{\text{body}} \neq N_{\text{buf}}^{\text{flow}}$.

Unlike domains D_{flow} and D_{xflow} which are recomputed only when the grid adapts [2], domains D_{body} and D_{xbody} are recomputed at every time-step during which the immersed surface moves.

Summarized in Table 4.1 are the source and target regions used in the present formulation to evaluate the action of discrete operators with wide, or potentially wide, stencils (discrete kernels), i.e. $\mathsf{L}_{\mathcal{Q}}^{-1}$, $\mathsf{E}_{\mathcal{Q}}(\alpha)$, and $\mathsf{K}_{\mathcal{Q}}^{-1}(\alpha)$. Similar source and target region considerations are readily deduced for all other operators, but are not discussed here since the operation count and propagation of finite boundary errors associated with these compact-stencil operators are significantly smaller than those of the operators listed in Table 4.1. Also highlighted by Table 4.1 are the significant

[14] In general, the solution can be tracked over arbitrary regions that include D_{flow}; such generalizations are discussed in [2].

Eq.	(4.21a)*	(4.21b)†	(4.21c)†	(4.21d)	(4.22)†	(4.22)†	(4.23)
Operator	$\mathsf{L}_{\mathcal{C}}^{-1}$	$\mathsf{H}_{\mathcal{F}}^{i}$	$\mathsf{L}_{\mathcal{C}}^{-1}$	$\mathsf{H}_{\mathcal{F}}^{i}$	$(\mathsf{K}_{\mathcal{F}}^{i})^{-1}$	$\mathsf{H}_{\mathcal{F}}^{i}$	$\mathsf{L}_{\mathcal{F}}^{-1}$
Source	D_{flow}	D_{xbody}	D_{body}	D_{xflow}	D_{body}	D_{body}	D_{flow}
Target	D_{xflow}	D_{body}	D_{xflow}	D_{xflow}	D_{body}	D_{body}	D_{xflow}
Op. Count	$M_{\text{f}\to\text{xf}}^{\mathsf{L}}$	$3M_{\text{xb}\to\text{b}}^{\mathsf{E}}$	$M_{\text{b}\to\text{xf}}^{\mathsf{L}}$	$3M_{\text{xf}\to\text{xf}}^{\mathsf{E}}$	$M_{\text{b}\to\text{b}}^{\mathsf{K}}$	$3M_{\text{b}\to\text{b}}^{\mathsf{E}}$	$3M_{\text{f}\to\text{xf}}^{\mathsf{L}}$
Op. Scaling	$\mathcal{O}(N_E)$	$\mathcal{O}(N_L)$	$\mathcal{O}(N_E)$	$\mathcal{O}(N_E)$	$\mathcal{O}(N_L)$	$\mathcal{O}(N_L)$	$\mathcal{O}(N_E)$

Table 4.1: Source and target regions used to approximate the action of non-compact discrete operators. The number of operations required to compute the action of each operator is denoted in the second to last row; operation counts for vector operations are approximated as three corresponding scalar operations. Superscript $*$ indicates equations originally given in the form $\mathsf{Lx} = \mathsf{y}$ that are written here as $\mathsf{y} = \mathsf{L}^{-1}\mathsf{x}$. Superscript † indicate equations that are *not* computed for cases without immersed surfaces.

operation count reductions achieved by taking advantage of the flexible source and target regions of LGF-FMM for cases involving immersed surfaces with $N_L \ll N_E$.

Temporal variations in the non-negligible support regions of discrete operators are facilitated by adding and removing blocks to D_{flow} and D_{body} (D_{xflow} and D_{xbody} are updated accordingly). At the end of every time-step, flow quantities on a region that is a few grid cells greater than D_{flow} are used to compute the block-wise weight function W_{flow} for each block in D_{xflow}, which in turn is used to define D_{flow} of the next time-step,

$$D_{\text{flow}}^{k+1} = \left\{ B : [W_{\text{flow}}(B')] > \epsilon_{\text{supp}},\ B' \in D_{\text{xflow}}^{k} \right\}. \tag{4.24}$$

Here, we use the block-wise weight function proposed in [2],

$$W_{\text{flow}}(B) = W(\text{pos}(B)) \max_{B \in D_{\text{xflow}}} \left(\mu(B)/\mu_{\text{global}},\, \nu(B)/\nu_{\text{global}} \right), \tag{4.25}$$

where $W(\text{pos}(B))$ is a function of the position of block B,

$$\mu(B) = \max_{\mathbf{m} \in ind(B)} |\mathsf{w}(\mathbf{n})|, \quad \mu_{\text{global}} = \max_{B \in D_{\text{xflow}}} \mu(B), \tag{4.26a}$$

$$\nu(B) = \max_{\mathbf{m} \in ind(B)} |\mathsf{h}(\mathbf{n})|, \quad \nu_{\text{global}} = \max_{B \in D_{\text{xflow}}} \nu(B). \tag{4.26b}$$

Solution grid variables $\mathsf{w} = \mathsf{Cu}$ and $\mathsf{h} = \overline{\mathsf{D}}\tilde{\mathsf{N}}(\mathsf{u}, t)$ correspond to the discrete vorticity and divergence of the Lamb vector.

For the case of $W(\mathbf{r}) = 1$, Eq. 4.25 approximates the maximum normalized residual of the discrete Poisson problems Eq. (4.21a) and (4.23) resulting from excluding source terms outside of D_{flow} [2].[15] Accurate solutions to the laminar-to-turbulence transition of thin vortex rings at Reynolds numbers up to 7,500 resulting in small solution grids near the ring core are reported in [2] using $W(\mathbf{r}) = 1$. In contrast, small solution grids are not expected to result from flows around immersed surfaces computed with $W(\mathbf{r}) = 1$ since vorticity is constantly generated at the surface and convected downstream. In practice, we are often interested in accurately reproducing the flow physics in the vicinity of the immersed surface, and are willing to reduce computational costs by neglecting flow features in far-downstream wake regions that do not significantly affect the near-surface flow. Point-wise estimates for the residuals of Eq. (4.21a) and (4.23) based on the asymptotic $\mathcal{O}(|\mathbf{n}|^{-1})$ decay of G_{L} [1] indicate that errors near the immersed surface resulting from

$$W(\mathbf{r}) = \frac{1}{\max\left(\eta, \text{dist}(\mathbf{r})\right)}, \tag{4.27}$$

where $\eta \geq 1$ is a prescribed parameter and $\text{dist}(\mathbf{r})$ is the non-dimensionalized distance from $\Gamma(t)$, are comparable in magnitude to those resulting from using $W(\mathbf{r}) = 1$. Unless otherwise stated, subsequent discussions and numerical experiments take W_{flow} to be given by Eq. (4.25) and (4.27).

4.4.2 Algorithm summary

In this section we summarize the present IB-LGF method for incompressible flows on unbounded staggered Cartesian grids. The sequence of steps performed in the IB-LGF algorithm for an s-stage IF-HERK scheme is outlined as follows:

1. *Pre-processing* [Sec. 4.3.2]: (rigid surface only, optional) for each unique force

[15] The residuals resulting from neglecting source terms outside D_{body} when solving the discrete Poisson problems in Eq. (4.21c) and (4.22) are zero since, by construction, D_{body} contains all non-zero source terms for these problems.

Schur complement operator $\hat{\mathfrak{C}} \in \{\mathfrak{C}^i, \forall\, i = [1, s]\}$, build its dense SPD matrix representation $\mathbf{C}$ by applying the operator to each standard basis vector, compute the Cholesky decomposition of $\mathbf{C}$ and invert the Cholesky factor $\mathbf{L}$ using ScaLAPACK, and store $\mathbf{W} = \mathbf{L}^{-1}$.

2. *Time integration*: for the k-th time-step:

   a) *Grid body update* [Sec. 4.4.1]: use the prescribed motion of $\Gamma(t)$ and the support region of discrete delta functions to compute D_{body} and D_{xbody} for $t \in [t_k, t_{k+1}]$.

   b) *Grid flow update* [Sec. 4.4.1]: use $D_{\text{body}} \subseteq D_{\text{flow}}$, weight function W_{flow}, threshold values ϵ_{supp}, and $\mathsf{w}_k \leftarrow \mathsf{C}\mathsf{u}_k$ and $\mathsf{h}_k \leftarrow \overline{\mathsf{D}}\tilde{\mathsf{N}}(\mathsf{u}_k, t_k)$ to construct new D_{flow}. If necessary, update old D_{flow} and D_{xflow} by adding or removing blocks. Copy w_k from the old to the new solution grid and zero values on D_{buffer}.

   c) *Velocity refresh* [Sec. 4.4.1]: if either D_{body} has been updated or the number of time-steps since last refresh exceeds $Q_{\max}$, compute u_k from w_k using Eq. (4.23).

   d) *IF-HERK* [Sec. 4.2.3]: use xIF-HERK(u_k, t_k) to compute u_{k+1}, t_{k+1}, q_{k+1}, and f_{k+1}, where xIF-HERK is the version of the IF-HERK algorithm that restricts the source and target regions of discrete operators to finite sub-regions of the solution grid, e.g. operations defined in Table 4.1. Linear systems arising at each RK stage are solved using the nested projection technique Eq. (4.21) with an appropriate body forces solver (Section 4.3.2).

An operation count estimate for a single time-step based on the action of all non-compact operators, i.e. operators listed in Table (4.1) and $\mathfrak{C}^{-1}$, is given by $M = M_{\text{flow}} + M_{\text{ib}}$, where M_{flow} is the number of operations used to solve the flow without

immersed surfaces [2],

$$M_{\text{flow}} = sM^{\mathsf{L}}_{\text{f}\to\text{xf}} + 3C(s)M^{\mathsf{E}}_{\text{xf}\to\text{xf}} + \lceil 3M^{\mathsf{L}}_{\text{f}\to\text{xf}} \rfloor, \tag{4.28}$$

M_{ib} is the number of operations required to compute the additional IB terms,

$$M_{\text{ib}} = sM^{\mathsf{L}}_{\text{b}\to\text{xf}} + sM^{\mathfrak{C}}_{\text{f}\to\text{f}} + 3sM^{\mathsf{E}}_{\text{xb}\to\text{b}}, \tag{4.29}$$

and $M^{\mathfrak{C}}$ is the number of operations used to solve for the body forces. For a general s-stage HERK scheme with second-order accurate constraint variables $C(s)$ is equal to $s + ((s-1)s)/2 - 1$ [2]. The last term in Eq. (4.28) is associated with the vector Poisson solve (roughly equal to three scalar Poisson solves) required by the velocity refresh procedure, which is not necessarily performed at every time-step as indicated by the notation $\lceil \cdot \rfloor$. All terms in Eq. (4.28) and (4.29), except for $M^{\mathfrak{C}}_{\text{f}\to\text{f}}$ and $M^{\mathsf{E}}_{\text{xb}\to\text{b}}$, scale as $\mathcal{O}(N_E)$. The term $M^{\mathsf{E}}_{\text{xb}\to\text{b}}$ scales as $\mathcal{O}(N_L)$ and, for typical flows, is negligible compared to any of the M^{L}_x terms. Estimates and scaling for $M^{\mathfrak{C}}_{\text{f}\to\text{f}}$ are discussed in Section 4.3.2, but we note here that computation time spent solving for body forces is less than 3% of the total run time for all test cases included in Section 4.5.

The present MPI-based parallel implementation partitions and distributes the support of the discrete delta functions according to the block-wise partition and distribution of the solution grid. Values for all IB markers are taken to be known by all processors, which is accomplished by having values broadcast before and aggregated after the application of $\mathscr{I}^\dagger$ and $\mathscr{I}$, respectively. Details regarding the parallel implementation of the flow solver, i.e. the IB-LGF method without an immersed surface, are discussed in [1, 2]. Load balancing is performed every time D_{flow} or D_{xflow} are updated. This operation consists of optimizing the most computationally expensive operations, i.e. Eq. (4.21a) and (4.23), following the procedure described in [1] with the additional requirement of having all blocks belonging to D_{body} be distributed as equally as possible across all processors.[16] Each RK stage of the

[16] The assumption that solving Eq. (4.21a) and (4.23) are the most computationally expensive

representative problems of Section 4.5 is evaluated at a typical computation rate (normalized by the total number of MPI processes) of approximately 80 micro-seconds per active grid cell or, equivalently, 20 micro-seconds per active flow variable (3 velocity components and 1 pressure per grid cell).

Lastly, we clarify that the LGF-FMM [1] and the LGF flow solver [2] are direct solvers that compute solutions to prescribed tolerances based on a set of parameters. Aside from convergence criteria required for the case of iterative boundary force solutions, the IB-LGF method does not depend on any additional parameters beyond those of the LGF flow solver [2]. Furthermore, having limited our attention to cases with $D_{\text{body}} \subseteq D_{\text{flow}}$, $D_{\text{xbody}} \subseteq D_{\text{xflow}}$, and $N_{\text{buf}}^{\text{body}} = N_{\text{buf}}^{\text{flow}}$ ensures that the procedures of LGF flow solver used to determine appropriate values for all parameters based on single threshold ϵ^* extend to the IB-LGF method. Subsequent discussions take ϵ^* to be equal to the grid adaptivity parameter ϵ_{supp}.

4.5 Verification examples

Numerical experiments on flows around infinitely thin rectangular flat plates and spherical shells are used to verify the IB-LGF method. Rectangular flat plates are generated by a set of IB markers arranged in a 2D uniform Cartesian grid with a prescribed aspect-ratio AR and angle-of-attack α. Spherical shells, subsequently referred to as spheres, are generated by placing IB markers at the centroids of the faces of an *icosphere*. Icospheres are triangulated surfaces constructed by recursively subdividing the faces and radially projecting newly created vertices onto the unit sphere of an initial icosahedron [114]. The ratio of the minimum to maximum distances between any two IB markers tends to approximately 0.28 after a large number of subdivisions.

For all test cases, the minimum spacing between any two IB markers, $\Delta s^* =$

---

operations is based on the numerical experiments of Section 4.5 for rigid surfaces solved using the pre-processing technique.

$\min_{\forall(i,j)} |\mathbf{X}(\boldsymbol{\xi}_i) - \mathbf{X}(\boldsymbol{\xi}_j)|$, is taken to be approximately equal to the grid spacing ($1.0 < \Delta s^*/\Delta x < 1.1$), and the smoothed version [115] of the 3-pt δ_h [101] is used to construct $\mathscr{I}$. Boundary forces are computed using the parameter-free Cholesky pre-processing technique discussed in Section 4.3.2. Unless otherwise stated, the adaptivity threshold parameter ϵ^* is taken to be 5×10^{-4}, a choice that will be justified in Section 4.5.3. All test cases, except those for the temporal refinement studies, are performed using the HERK coefficients of Scheme A included in Appendix 4.B and are subject to the CFL condition $|\mathbf{u}|\Delta t/\Delta x < 1$. The time-step size is specified so that the CFL based on the maximum point-wise velocity remains below 0.5 and 0.9 for flows at $\mathrm{Re} \leq 500$ and $\mathrm{Re} = 3{,}700$, respectively, except for the first few time-steps of impulsively started flows during which the CFL is allowed to be approximately equal to unity. We clarify that the average CFL for spheres at $\mathrm{Re} = 3{,}700$ is approximately 0.6, which is significantly lower than the large-time maximum of 0.9.

4.5.1 Discretization error

The convergence rates of the discretization technique is examined through spatial and temporal refinement studies of flows around spheres of radius D with a prescribed velocity $\mathbf{U}(t) = (U_x(t), 0, 0)$,

$$U_x(t) = \begin{cases} 4\beta U \displaystyle\int_0^t e^{\frac{-1}{1-(4t'-1)^2}}\, dt' & \text{for } 0 \leq \frac{tU}{D} \leq \frac{1}{2} \\ U & \text{for } \frac{1}{2} < \frac{tU}{D} \end{cases}, \tag{4.30}$$

where $\beta \simeq 2.25228$ is taken so that $U_x(t)$ is infinitely differentiable for all t (assuming $U_x(t) = 0$ for $t < 0$). The instantaneous Reynolds number $\mathrm{Re} = U_x(t)D/\nu$ levels to a constant equal to 100 for $t > \frac{1}{2}D/U_\infty$. All numerical experiments discussed in this section are performed using $\epsilon^* = 10^{-8}$.

The spatial convergence study is performed using a total of seven test cases, S.I–S.VII, corresponding to spheres generated by $20 \times 4^{i-1}$ for $i = 1, 2, \ldots, 8$ IB markers. The time-step size, Δt, is held fixed across all test cases, and chosen so that the

maximum CFL for S.VII is less than 0.25. Estimates for the errors obtained by taking S.VII to be the reference, or *true*, solution are reported in Figure 4.2. As expected from analysis [116, 117] and numerical experiments [69, 93, 94] of similar IB methods, the velocity $\mathbf{u} \approx \mathsf{u}$ is verified to be first-order accurate in the L_∞ norm. Less intuitive is the fact that the pressure $p \approx \mathsf{p}$ and the net body force $\mathbf{F} \approx \sum_i \mathsf{f}_i$ also exhibit first-order convergence rates under the L_∞ norm.

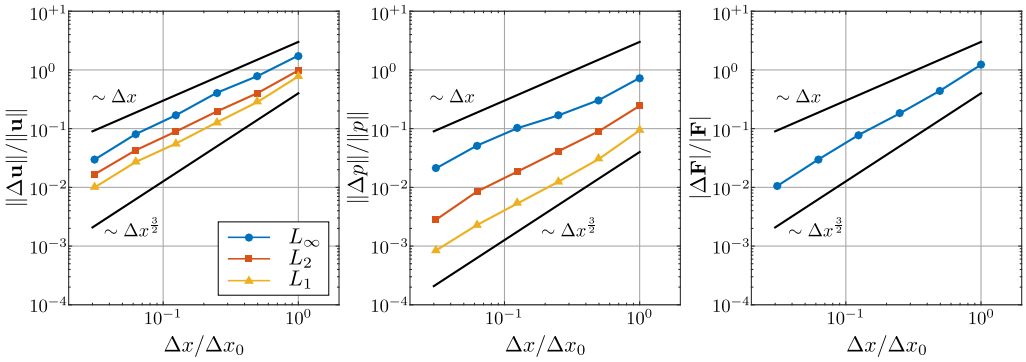

Figure 4.2: Differences in the velocity (*left*), pressure (*middle*), and net body force (*right*) for different values of Δx while holding Δt fixed. The value of Δx_0 is equal to Δx of coarsest test case (S.I).

It is well-known that the spatial smoothing inherent to the regularized delta function treatment of the immersed surfaces is unable to correctly capture the discontinuous pressure across interfaces. The spatial regularization resulting from δ_h has been shown to lead to $\mathcal{O}(1)$ errors in the pressure near the immersed surface [116, 118], which in turn prevents the L_∞ convergence of the discrete pressure to the actual, continuum pressure. The first-order L_∞ convergence rate of the pressure shown in Figure 4.2 is a consequence of taking the reference solution to be that of S.VII as opposed to the actual solution to Eq. (4.1). Convergence rates equal or greater to first-order follow from the fact that the continuum pressure across surfaces regularized by $\delta_h(\mathbf{x})$ is continuous and differentiable (provided a sufficiently smooth δ_h) [119]. However, for the present refinement study the convergence rate is at most first order since the regularization length-scale h is taken to be equal to Δx of the

reference (finest) solution and, as a result, will change as progressively finer test cases are considered.

Next, we consider the slightly greater than first-order convergence rate of the body forces. Surface stresses obtained from most IB methods are known to exhibit less than first-order point-wise convergence rates [4, 94, 115] and can even grow as the immersed surface is refined [4]. Yet, it is also known that the low-order moments for the surface stresses, such as the net force on the immersed body, are physically accurate. This is verified by the right plot of Figure 4.2, which shows that the net body force, $\mathbf{F}$, convergence at a rate that is slightly greater than first-order. Approximate first-order convergence rates for the net force on rigid surfaces also have been demonstrated for other IB-DLM methods [4, 94].

Lastly, we report the L_2 condition number, κ, of the force Schur complement of the last RK stage, $\mathfrak{C}^s$, for each test case in Table 4.2. As points of comparison, Table 4.2 also includes values for κ resulting from using values $\Delta s^*/\Delta x$ that are smaller (1.00 and 0.95) than those used for S.I–S.VII (1.05). Table 4.2 verifies the intuitive fact, based on the finite spatial resolution of the fluid solver, that the matrix corresponding to $\mathfrak{C}^s$ rapidly becomes ill-conditioned for values $\Delta s/\Delta x$ below certain threshold, which is approximately unity for the present test cases. The heuristic constraint of requiring $\Delta s^*/\Delta x \geq 1$ is not universal across IB-DLM methods; [69] states that $\Delta s/\Delta x \approx 1$ results in "reasonable" condition numbers for $\mathfrak{C}$-like matrices computed using the 3-pt δ_h of [101], and [94] numerically demonstrates that $\Delta s/\Delta x \approx 2$ results in condition numbers for $\mathfrak{C}$-like matrices computed using the 6-pt δ_h of [120] comparable to those listed in Table 4.2 for the case of steady Stokes flow around a sphere.[17] We clarify that, in the continuum limit, the boundary integral operator associated with $\mathfrak{C}$ has a zero-eigenvalue mode corresponding to the uniform compression of the sphere. Typically, we expect that the small geometric irregularities, strict symme-

[17] Condition numbers of $\mathcal{O}(10^6 - 10^7)$ are reported in [94] for the case of steady Stokes flow around a sphere with $\Delta s/\Delta x \approx 1$.

try-breaking, and slight porosity[18] of numerical immersed surfaces generated using standard discrete delta functions and well-spaced IB-markers, i.e. $\Delta s/\Delta x \approx 1$, to result in non-singular discrete $\mathfrak{C}$ operators, even if the continuous version of $\mathfrak{C}$ is singular. Small-eigenvalue discrete modes associated with zero-eigenvalue continuous modes that do not satisfy the continuous divergence-free constraint are expected to be only a small part of the solution of $\mathfrak{C}\mathfrak{f} = \mathfrak{r}$, i.e. the dot product of these modes with $\mathfrak{r}$ is small, since $\mathfrak{r}$ is interpolated from a discrete divergence-free field. Additional discussions and numerical examples regarding the null-space, or lack thereof, and conditioning of $\mathfrak{C}$-like operators arising in other IB-DLM discretizations are provided in [4, 94]. Here, the results of Table 4.2 are used to verify the well-posedness of Eq. (4.8) as a (solvable) DAE system of index 2 and to motivate the nominal value of $\Delta s^* \approx 1.05\Delta x$ used in subsequent numerical experiments.

No. IB markers	20	80	320	1,280	5,120	20,480	81,920
$\Delta s^*/\Delta x \simeq 1.05$	0.19	0.20	0.18	0.17	0.19	0.44	1.31
$\Delta s^*/\Delta x \simeq 1.00$	0.19	0.52	0.36	0.42	1.61	1.41	–
$\Delta s^*/\Delta x \simeq 0.90$	2.06	3.13	1.72	2.69	8.51	27.94	–

Table 4.2: L_2 condition number of $\mathfrak{C}^s$ for different ratios of $\Delta s^*/\Delta x$. Values of $\Delta s^*/\Delta x \simeq 1.05$ are used in the numerical experiments S.I–S.VII.

We now turn our attention to the temporal discretization error. The temporal convergence studies are performed for each of the four IF-HERK schemes included in Appendix 4.B, Scheme A – D, for a sphere generated by 1280 IB markers. A total of nine test cases, T.I – T.IX, of varying time-step size, $\Delta t/\Delta t_0 = 2^{-i+1}$ for $i = 1, 2, \ldots, 9$, are considered for each scheme. Here, Δt_0 is chosen such that the maximum CFL of test case T.I is less than 0.5. Error estimates for each test case are obtained by taking T.IX of the corresponding IF-HERK scheme to be the reference solution.[19] The L_∞ norm of the errors, depicted in Figure 4.3, verifies that the

[18]The no-slip constraint is only enforced at a finite number of points. The velocity at points located between IB markers is not required to satisfy the no-slip constraint.

[19]For some cases, the spatial discretization error is significantly larger than the temporal dis-

computed convergence rates of each IF-HERK scheme with respect to Δt is equal to the expected order of accuracy based on the HERK order conditions discussed in Section 4.2.3 and Appendix 4.B.

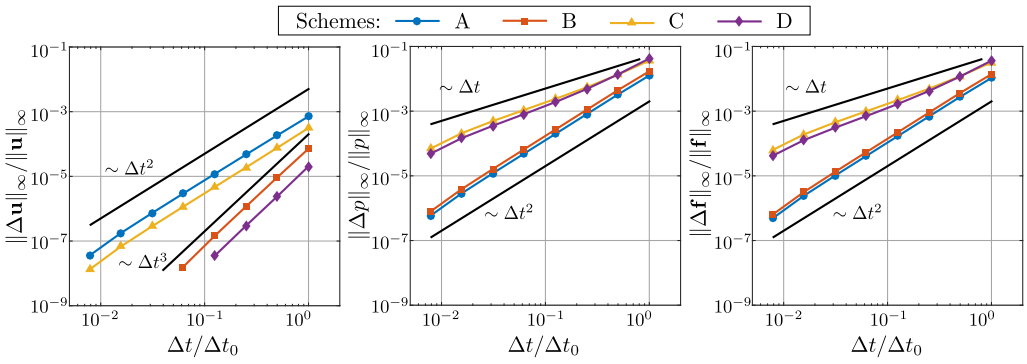

Figure 4.3: Differences in the velocity (*left*), pressure (*middle*), and net body force (*right*) for different values of Δt while holding Δx fixed. The value of Δt_0 is equal to Δt of coarsest test case (T.I). Entries for the velocity error for Scheme B and D that are below 10^{-8} are excluded from the left plot since the error for these cases saturates between 10^{-9} and 10^{-8} due to prescribed adaptivity threshold $\epsilon^* = 10^{-8}$.

We emphasize that the refinement studies of this section have only verified that numerical solutions converge at the expected rate under Δx and Δt refinements. The tests cases discussed in the following sections will demonstrate that the computed solutions are in fact accurate physical approximation to Eq. (4.1).

4.5.2 Flow around low-aspect-ratio rectangular plates

The physical fidelity of the IB-LGF method is verified in this section by comparing solutions for impulsively-started rectangular flat plates of chord-length c and area A to previously published results. We begin by considering the experimentally-validated test cases [121] of flows around plates of $AR = 2$ at $\mathrm{Re} = 100$ and $0° \leq \alpha \leq 90°$. Here, test cases are performed taking $\Delta x = 0.020c$, which is comparable

cretization error. This does not affect the present study since the spatial discretization error is approximately the same for all test cases and our error estimates are computed as the difference of two test case solutions.

to the near-surface grid spacing of $0.025c$ used to compute these flows by other IB methods [100, 121]. Previous refinement studies and comparisons with experimental data [121] indicate that the flow is sufficiently well-resolved using the present value of Δx.

The left plot of Figure 4.4 demonstrates that the computed drag and lift coefficients, $C_{\mathrm{D}} = -F_x/(\frac{1}{2}\rho U^2 A)$ and $C_{\mathrm{L}} = F_y/(\frac{1}{2}\rho U^2 A)$, at $tU/c = 13$ are in good agreement with previously reported values [100, 121]. The force coefficients from the three methods are nearly indistinguishable for $0^\circ \leq \alpha \leq 50^\circ$ and by less than 5% for $60^\circ \leq \alpha \leq 90^\circ$. The large-time ($50 \leq tU/c \leq 75$) behavior of the mean and fluctuating components of C_{D} and C_{L} are summarized in the right plot of Figure 4.4. This plot demonstrates that for $60^\circ \leq \alpha < 90^\circ$ the flow is unsteady and that the large-time mean forces are significantly different from the instantaneous forces at $tU/c = 13$.[20] For $\alpha = 60^\circ$ and $\alpha = 70^\circ$ the flow is periodic with Strouhal numbers $\mathrm{St} = F_y c \sin\alpha/U$ equal to 0.13 and 0.11, respectively. In contrast, for $\alpha = 80^\circ$ and $\alpha = 90^\circ$ the flow is a aperiodic (at least during $50 \leq tU/c \leq 75$) since the force coefficients neither have a constant mean value nor a clear dominant frequency.[21] We suspect that the sensitivity of instantaneous measurements of unsteady flows to small perturbations is responsible for the larger differences across the three numerical investigations presently considered for test cases with $60^\circ \leq \alpha \leq 90^\circ$ when compared to the same differences for test cases with $0^\circ \leq \alpha \leq 50^\circ$.

Next, we consider impulsively started plates of different ARs at $\alpha = 30^\circ$ and $\mathrm{Re} = 300$. Previous numerical experiments on these flows [100, 121, 122] have used grid spacings of approximately $0.025c$ in the immediate vicinity of the plate (same as for

[20] The discussion of [100] regarding the present test cases states that the flow has reached a steady state at $tU/c = 13$. A comparison between the force coefficients shown in the left and right plots of Figure 4.4 indicates that only flows with $0^\circ \leq \alpha \leq 50^\circ$ have reach a steady state at $tU/c = 13$.

[21] The value of C_{L} for $\alpha = 90^\circ$ is approximately zero for $tU/c < 10$ and decreases non-uniformly (oscillates about local mean) to approximately -3×10^{-3} at $tU/c = 75$. As a point of reference, the value of $|C_{\mathrm{L}}|$ for $\alpha = 0^\circ$ is less than 5×10^{-4} throughout the entire simulation period.

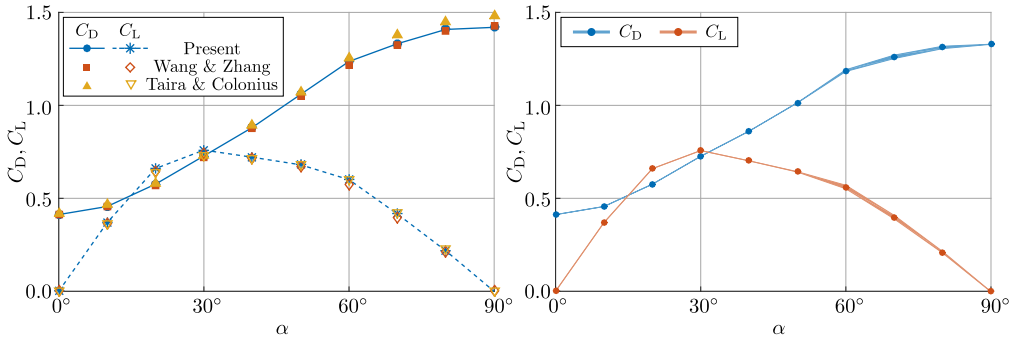

Figure 4.4: Drag and lift coefficients for rectangular flat plates of AR = 2 at Re = 100 and different values of α. Instantaneous values at $tU/c = 13$ (*left*). Range (shaded regions) and mean value (solid circles) of force coefficients during $50 \leq tU/c \leq 75$ (*right*).

the previously referenced Re = 100 test cases). Here, each flow is computed using (A) $\Delta x = 0.025c$ and (B) $\Delta x = 0.015c$.

Vortical structures in the wake of plates of AR = 1, 2, and 4 are illustrated as iso-surfaces of constant vorticity strength in Figure 4.5. The depicted structures are in good visual agreement with the structures reported in previous numerical experiments for AR $= 1, 2, 4$ [121] and AR = 4 [100, 122]. Also shown in Figure 4.5 are snapshots of cross-sectional cuts of the finite computational domains resulting from the block-wise adaptive computational grid algorithm. As expected from the adaptivity criteria discussed in Section 4.4.1, strong vortical regions near the immersed surface are efficiently tracked by adding and removing computational blocks. For the test case of AR = 4, the stream-wise length of the union of blocks is approximately $[-1c, 9c]$ (about the leading edge), which can be compared to the stream-wise length $[-4c, 6.1c]$ of the finite domain (with approximate boundary conditions) used in [100, 121].

The large-time ($50 \leq tU/c \leq 80$) temporal statistics for the force coefficients of the plates depicted in Figure 4.5 are included in Table 4.3. Strouhal numbers of 0.12 for AR = 4 obtained for (A) and (B) are also reported in [100, 121]. For all

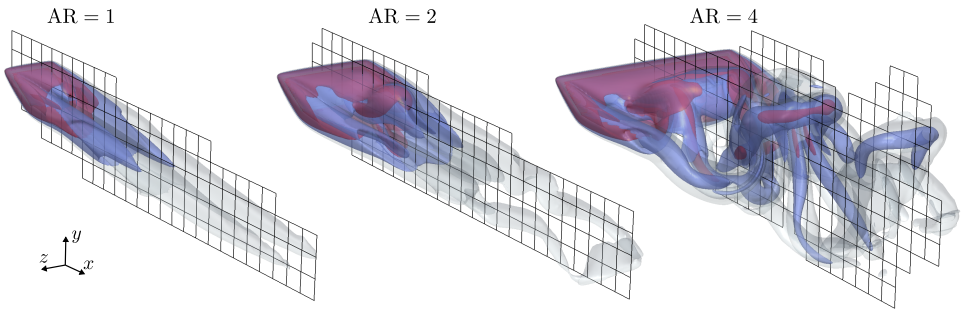

Figure 4.5: Vortices in the wakes of rectangular flat plates of different ARs at $\alpha = 30^\circ$ and $\mathrm{Re} = 300$. Shown above are iso-surfaces of $|\boldsymbol{\omega}|c/U = 2, 4, 8$ at $tU/c \simeq 46.2$ computed using $\Delta x = 0.015c$. The union of boxes shown on the $x - y$ plane centered about the plate center depict the cross-sectional cut of the block-wise adaptive computational domain. Depicted blocks have been coarsened by a factor of two in each direction for visualization purposes.

ARs, differences in mean force coefficients between (B) and [100, 121, 122] are less than 12%. The effect of the grid resolution on the accuracy of the solution can be approximated by comparing the results of the low-resolution test cases (A) with the results of the high-resolution test cases (B). The values of Table 4.3 indicate that the differences in mean force coefficients between (A) and (B) are less than 4% for all ARs. Since the grid resolutions of (A) and [100, 121, 122] are approximately the same, we suspect that modeling errors resulting from the small computational domains and approximate boundary conditions used in [100, 121, 122] account for most of the differences in the mean force values.[22]

4.5.3 Flow around spheres

In this section we further verify the IB-LGF method by computing flows around impulsively started spheres. A small perturbation $(0, \hat{u}(t), 0)$ is introduced to the nominal velocity of the sphere $(U, 0, 0)$ in order to break axial symmetry. We take $\hat{u}(t)$ to be non-zero for $1 < tU/D < \frac{4}{3}$ with values equal to the bump function

[22] The numerical methods of [121] and [100, 122] use stretched and locally refined grids, respectively, to discretize computational domains of size $[-4c, 6.1c] \times [-5c, 5c] \times [-5c, 5c]$.

	AR = 1		AR = 2		AR = 4				
	C_{D}	C_{L}	$\overline{C_{\mathrm{D}}}$	$\overline{C_{\mathrm{L}}}$	$\overline{C_{\mathrm{D}}}$	ΔC_{D}	$\overline{C_{\mathrm{L}}}$	ΔC_{L}	St
Present (A)	0.543	0.627	0.504	0.571	0.620	0.018	0.791	0.074	0.118
Present (B)	0.526	0.644	0.488	0.587	0.593	0.016	0.798	0.053	0.124
TC09 [121]	0.56	0.60	0.53	0.57	0.66	–	0.80	–	0.12
WZ11 [100]	–	–	–	–	–	–	–	–	0.12
WZ13 [122]	–	–	–	–	–	–	0.79	–	–

Table 4.3: Drag and lift coefficients, and Strouhal numbers for rectangular flat plates of different ARs at $\alpha = 30°$ and Re = 300. Present (A) and (B) correspond to test cases computed using $\Delta x/c = 2.5\times10^{-2}$ and $\Delta x/c = 1.5\times10^{-2}$, respectively. Results from TC09 – Taira and Colonius [121], WZ11 – Wang and Zhang [100], and WZ13 – Wang et al. [122] are also provided.

$\frac{1}{10}Ue^{1-1/(1-\tau^2)}$ with $\tau = 8t - 9$. Net body forces are reported as non-dimensional force coefficients $C_q = F_q/(\frac{1}{2}\rho U^2\pi(\frac{D}{2})^2)$ for $q \in \{x, y, z\}$, and correspond to the drag ($C_{\mathrm{D}} = C_x$), lateral ($C_{\mathrm{L}} = C_y$), and side ($C_{\mathrm{S}} = C_z$) coefficients.

First, we demonstrate that the grid adaptivity criteria and the nominal threshold value $\epsilon^* = 5\times10^{-4}$ accurately capture the unbounded domain flow by only tracking the solution on a small, finite region near the surface and immediate wake of a sphere at Re = 300. Periodic flows exhibiting planar symmetry are limited to a narrow range of Reynolds numbers that has been numerically estimated to be $280 < \mathrm{Re} < 375$ [123, 124]. The temporal periodicity and spatial symmetry about the x-y plane of the flow [123, 125, 126] makes this test case a challenging problem that still permits meaningful force coefficient comparisons.

The time histories for C_{D} and C_{L}, and snapshots of the vorticity field for spheres generated by 20,480 IB markers ($\Delta x \simeq 9.33\times10^{-3}$) computed with values of $\epsilon^* = 5\times10^{-i}$ for $i = 2$, 3, 4, and 5 are shown in Figure 4.6. The maximum value of $|C_{\mathrm{S}}|$ for $0 \le tU/D \le 90$ is less than 2×10^{-2}, 1×10^{-2}, 5×10^{-3}, and 2×10^{-3}, respectively, for the present test cases sorted in decreasing order of ϵ^*, which in turn confirm the expect planar symmetry of the flow. Periodic oscillations in the time history of force coefficients are clearly observed for $\epsilon^* \le 5\times10^{-4}$, but not for $\epsilon^* \ge 5\times10^{-3}$. The apparent stabilization of the flow for $\epsilon^* \ge 5\times10^{-3}$ is expected from

the fact that the finite computational domains of these test cases do not support a complete wavelength of the wake instability. In contrast, the computational domain of $\epsilon^* = 5 \times 10^{-4}$ supports at least one full wavelength of the instability and is able to accurately reproduce the large-time mean and oscillatory components of C_{D} and C_{L} obtained by the most conservative test case, i.e. $\epsilon^* = 5 \times 10^{-5}$.

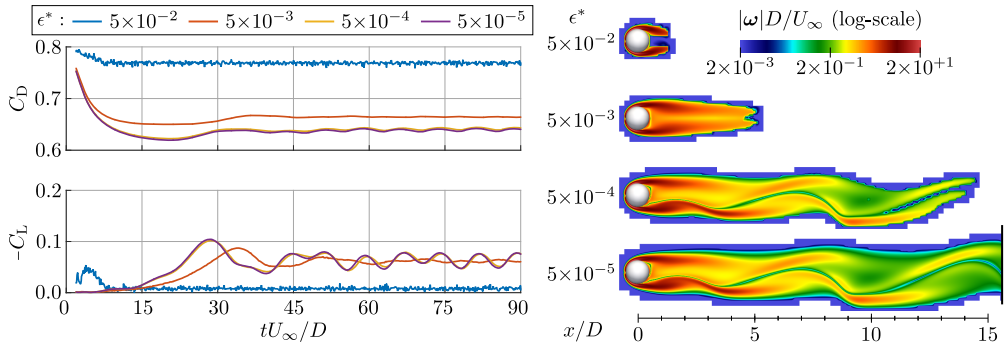

Figure 4.6: Flow around a sphere at Re = 300 computed using different adaptive threshold, ϵ^*, values. Time history of drag and lateral force coefficients (*left*). Side-view of finite computational domains at $tU/D = 80.3$ (*right*). For each case, the finite computational domain corresponds to the non-white region near the body and its wake. For the case of $\epsilon^* = 5 \times 10^{-5}$ the finite computational domain continues for an additional $15D$ beyond solid black line.

Quantitative estimates for the errors resulting from non-zero values of ϵ^* are computed here as the differences in the force coefficient C between two test cases:

$$\varepsilon^I(C) = \max_{t \in T_A} |C(t) - C'(t)|, \ \varepsilon^{II}_{\mathrm{EXT}}(C) = \left| \mathrm{EXT}_{t \in T_B} C(t) - \mathrm{EXT}_{t \in T_B} C'(t) \right|, \quad (4.31)$$

where $T_A U/D = [2, 30]$, $T_B U/D = [75, 90]$, and EXT is either the minimum (min) or maximum (max) extremum. Error estimates obtained by taking the force coefficients of $\epsilon^* = 5 \times 10^{-5}$ to be the reference values are provided in Table 4.4. As expected, the error associated with neglecting values outside D_{flow} based on the criteria of Section 4.4.1 is approximately proportional to ϵ^*. The results of Table 4.4 indicate that, in the absence of discretization errors, the forces computed using the nominal

threshold, $\epsilon^* = 5 \times 10^{-4}$, are accurate up to 0.6% of the actual physical forces.[23]

ϵ^*	$\varepsilon^I(C_\mathrm{D})$	$\varepsilon^{II}_{\min}(C_\mathrm{D})$	$\varepsilon^{II}_{\max}(C_\mathrm{D})$	$\varepsilon^I(C_\mathrm{L})$	$\varepsilon^{II}_{\min}(C_\mathrm{L})$	$\varepsilon^{II}_{\max}(C_\mathrm{L})$
5×10^{-2}	0.1600	0.1117	0.1386	0.1038	0.0470	0.0522
5×10^{-3}	0.0311	0.0269	0.0235	0.0509	0.0108	0.0116
5×10^{-4}	0.0030	0.0027	0.0022	0.0041	0.0003	0.0002

Table 4.4: Estimates for the error resulting from non-zero values of ϵ^* for a sphere at Re = 300. Reference values, i.e. C' in Eq. (4.31), are taken from test case with $\epsilon^* = 5 \times 10^{-5}$.

Having verified the error of the adaptive grid, we now turn our attention to confirming the physical fidelity of the IB-LGF method by comparing with previous investigations of flow around spheres. Large-time ($90 \leq tU/D \leq 150$) force statistics of spheres computed at Re = 100, 200, and 300 are compared in Table 4.5. The flow is steady and axi-symmetric at Re = 100, and steady and planar x-y symmetric at Re = 250. At Re = 250 and Re = 300 the absence of axial symmetry results in a non-zero C_L. Values computed using a coarse (5,120 IB markers and $\Delta x \simeq 1.8 \times 10^{-2}$) and a fine (20,480 IB markers and $\Delta x \simeq 8.8 \times 10^{-3}$) grid are shown by Table 4.5 to be consistent with the range of previously reported values.

The spread of values shown in Table 4.5 for spheres at Re = 300 indicates that this test case is difficult to compute accurately. The large spread of values for ΔC_L and ΔC_D (largest spread based on relative differences) has been attributed to differences in the domain size and boundary conditions of different numerical methods [126]. Consistent with this argument are the small differences in ΔC_L and ΔC_D between the present (B) results and those of the unbounded domain vortex method [126].

The numerical experiments on spheres discussed thus far have considered the steady axis-symmetric (Re = 100), the steady planar-symmetric (Re = 250), and the periodic planar-symmetric (Re = 300) flow regimes. Next we verify that the flow is

[23] Error percentage is computed as the maximum value obtained after normalizing the errors in C_D and C_L provided in Table 4.4 by the large-time total force coefficient $C_\mathrm{T} = |\mathbf{F}|/(\frac{1}{2}\rho U^2 \pi(\frac{D}{2})^2) \simeq 0.66$.

	Re = 100	Re = 250		Re = 300				
	C_D	C_D	C_L	$\overline{C_D}$	ΔC_D	$\overline{C_L}$	ΔC_L	St
Present (A)	1.084	0.709	0.060	0.665	0.0024	0.067	0.013	0.133
Present (B)	1.086	0.694	0.059	0.656	0.0024	0.065	0.014	0.134
JP99 [123]	1.10	–	0.062	0.656	0.0035	0.069	0.016	0.137
TO00 [125]	–	–	–	0.671	0.0028	–	–	0.136
KK01 [127]	1.087	0.701	0.059	0.657	–	0.067	–	0.134
PW02 [126]	–	–	–	0.683	0.0025	0.061	0.014	0.135
CS03 [128]	–	0.70	0.062	0.655	–	0.065	–	0.136
WZ11 [100]	1.13	–	–	0.68	–	0.071	–	0.135

Table 4.5: Drag and lift coefficients, and Strouhal numbers for a sphere at different Reynolds numbers. Present (A) and (B) correspond to test cases computed using $\Delta x/D \simeq 1.8 \times 10^{-2}$ and $\Delta x/D \simeq 8.8 \times 10^{-2}$, respectively. Results from JP99 – Johnson and Patel [123], TO00 – Tomboulides and Orszag [125], KK01 – Kim et al. [127], PW02 – Ploumhans et al. [126], CS03 – Constantinescu and Squires [128], and WZ11 – Wang and Zhang [100] are also provided.

unsteady with no fixed planar symmetry at Re = 500 [125, 126]. Figure 4.7 provides snapshots of stream-wise vorticity iso-surfaces for the aforementioned flow regimes (case of Re = 100 is not shown since stream-wise vorticity is of negligible magnitude). The flow at Re = 500 is approximately symmetric about the x-y plane at early times (similar to the Re = 300 test case), but such symmetry is lost at later times as shown in Figure 4.7 by the axial rotation of the stream-wise vortices.

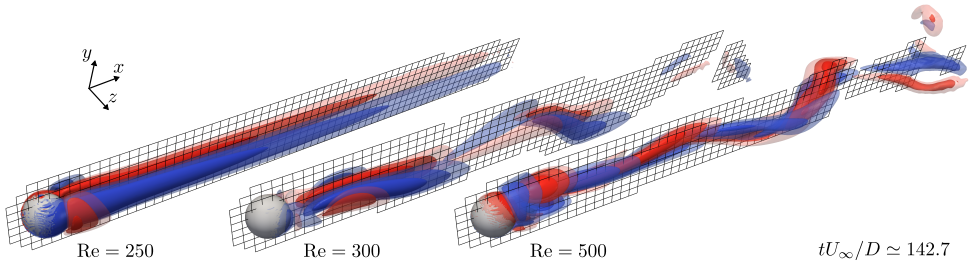

Figure 4.7: Stream-wise (x-direction) vortices in the wake of spheres at different Reynolds numbers depicted as iso-surfaces of constant ω_x. Iso-surfaces are for values of $\omega_x = \pm 0.20$, 0.10, and 0.05 at Re = 250, and of $\omega_x = \pm 1.0$, 0.50, and 0.25 at Re = 300 and 500. Depicted boxes are described in the caption of Figure 4.5.

As a final demonstration of the IB-LGF method, we consider the turbulent flow

around a sphere at Re = 3,700. This flow has been characterized in previous numerical [99, 129, 130] and experimental [131] investigations.[24] The flow is computed for $0 \leq t^*/U \leq 60$ using 81,920 markers and $\Delta x \simeq 4.3 \times 10^{-3}$, where t^* is used to indicate that flow was initialized from the large-time solution of a sphere at Re = 1,000. Subsequently reported time-averaged values are computed over the last five large-scale vortex shedding cycles (St = 0.215 [130]).

The thin boundary layer on the surface of the sphere is expected to be sufficiently well-resolved since the present value of $(\Delta x)\mathrm{Re}^{\frac{1}{2}}$ (scaling based on the expected $\mathcal{O}(\mathrm{Re}^{-\frac{1}{2}})$ laminar boundary layer thickness [132]) is between the values of $(\Delta x)\mathrm{Re}^{\frac{1}{2}}$ used to compute test cases (A) and (B) at Re = 300. As a point of reference, spheres at Re = 3,700 have been previously computed using a IB/LES method combined with a stretched grid with a near-surface minimum spacings of $9 \times 10^{-3}D$ [99] and using a unstructured mesh with a near-surface minimum element side lengths of $1.5 \times 10^{-3}D$ [130]. The *a posteriori* grid analysis of [130] demonstrates that the turbulent flow, with a minimum Kolmogorov length scale of $\eta/D = 1.34 \times 10^{-2}$ occurring in the $x/D < 3$ wake region, is well-resolved by a second-order unstructured mesh with an average element side length of $\overline{h}/D = 8 \times 10^{-3}$ over the $x/D < 3$ wake region. Based on these grid considerations, we assume that the present set of grid parameters are adequate to capture both the thin laminar boundary layer on the surface and the turbulent wake of the flow.

The core of vortical structures in the wake are depicted as iso-surfaces of constant Q-value in Figure 4.8. The Q-criterion [133] defines coherent vorticies as connected regions where Q, the second invariant of $\nabla \mathbf{u}$, is positive. Positive Q-values indicate a local excess of the rotation rate compared to the strain rate. Figure 4.8 confirms the previously reported [99, 130] pronounced helical-like pattern of large-scale vortical structures in wake. A visual analysis of multiple snapshots verifies that

[24] Previous investigations were conducted at Re = 3,700 [99, 130], Re = 4,200 [131], and Re = 5,000 [129].

the dominant vorticies forming the helical-like pattern are convected downstream without significant axial rotations and that the pattern is the result of the apparent random azimuthal position of growing shear layer instabilities [99, 130]. We clarify that the strong small-scale vortical filament- and horseshoe-like structures in the downstream wake regions shown in Figure 4.8 are not readily seen in comparable plots of previous numerical experiments [99, 130], but this is expected from the fact that these previous experiments aggressively coarsen downstream grid regions.

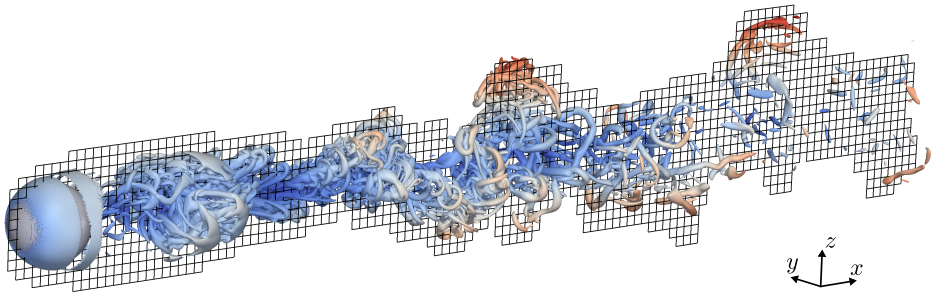

Figure 4.8: Vortex cores in the wake of a sphere at Re = 3,700 are illustrated by iso-surfaces of constant Q-value. Depicted are iso-surface of $QD^2/U^2 = 2$ colored according to the radial distance from the center-line of the sphere in the stream-wise direction. Depicted boxes are described in the caption of Figure 4.5.

We further characterize the flow by reporting on the large-time mean surface stresses and net body forces. The i-th component of the *stress vector*, $\sigma_i = [\mathbf{t}]_i$, at the location of the q-th IB marker can be approximated as $\sigma_i(\boldsymbol{\xi}_q) \approx [\mathbf{f}_q]_i / A_q$, where A_q is the area associated with the q-th IB marker. Given that our IB markers are located at the centroids of the faces (triangles) of an icosphere, we take the A_q to be equal to the area of the corresponding face. The non-dimensional surface stress coefficients in spherical coordinates are taken to be

$$C_{\sigma,\mathbf{r}} = \frac{\sigma_r - p_0}{\rho U}, \quad C_{\sigma,\boldsymbol{\theta}} = \frac{\sigma_\theta}{\rho U}, \quad C_{\sigma,\boldsymbol{\phi}} = \frac{\sigma_\phi}{\rho U}, \tag{4.32}$$

where $\mathbf{t} = \sigma_r \hat{\mathbf{r}} + \sigma_\theta \hat{\boldsymbol{\theta}} + \sigma_\phi \hat{\boldsymbol{\phi}}$, and θ and ϕ correspond to the polar and azimuthal angles, respectively (stagnation point located at $\theta = 0$). Here, the reference pressure

p_0 is taken to be $p_\infty - p_{\text{sphere}}$, where p_{sphere} is the value of the approximately uniform pressure distribution inside the sphere.[25] In the continuum limit, the surface normal stress coefficient $C_{\sigma,r}$ is equal to half of the pressure coefficient $C_p = (p - p_\infty)/(\frac{1}{2}\rho U)$. As discussed in Section 4.5.1, raw point-wise values of $\boldsymbol{\sigma}$ contain unphysical large high-frequency oscillations. These oscillations are partially filtered out and the point-wise accuracy of $\boldsymbol{\sigma}$ is significantly improved using the boundary force post-processing technique [4], which can be interpreted as a spatial weighted moving average smoothing technique that uses $\delta_{\Delta x}$ as the smoothing kernel. This technique constructs smoothed boundary forces $\hat{\mathfrak{f}}$ by evaluating the expression $\hat{\mathfrak{f}} = \mathscr{I}\mathsf{W}\mathscr{I}^\dagger\mathfrak{f}$, where $\mathsf{W}(\mathbf{n})$ is equal to $1/\gamma(\mathbf{n})$ for the case of non-zero $\gamma(\mathbf{n}) = [\mathscr{I}^\dagger\mathbf{1}](\mathbf{n})$ and equal to zero otherwise.

Time-averaged values of C_p and $C_{\sigma,\boldsymbol{\theta}}$ as functions of the polar angle, θ, are depicted in Figure 4.9.[26] The present values are in good visual agreement with the body-fitted mesh DNS values reported in [130]. We clarify that the curves shown in Figure 4.9 include a small $\mathcal{O}(\Delta s)$ post-processing error resulting from interpolating values of $\hat{\mathfrak{f}}$, which is defined on the faces of a six-times subdivided icosphere, onto geodesic lines between $\theta = 0^\circ$ and $\theta = 180^\circ$. Small remnants of the unphysical high-frequency oscillations in $\mathfrak{f}$ are visually noticeable in the values of $C_{\sigma,\boldsymbol{\theta}}$ over the region of $0^\circ \leq \theta \lesssim 50^\circ$.[27] Although these are undesirable features of the present

[25] The slight porosity of the numerical immersed surface results in an approximately uniform time-dependent pressure distribution inside the sphere. At $tU/D \approx 50$ the difference between the minimum and maximum pressure inside the sphere, but not in the support of $\mathscr{I}$, is approximately 0.3% of $\frac{1}{2}\rho U^2$. The maximum difference in p_{sphere} between any two instantaneous measurements during $30 \leq tU/D \leq 60$ is approximately 1% of $\frac{1}{2}\rho U^2$.

[26] The normalized skin-friction coefficient, $C_{\sigma,\boldsymbol{\theta}}\text{Re}^{\frac{1}{2}}$, depicted in [130] for the computations of [129] are approximately 16% larger than to those shown in the left plot of Figure 4.9. The curve shown in Figure 4.9 was computed by scaling the values of $\sigma_\theta\text{Re}/\rho U$ reported in [129] by $\text{Re}^{-\frac{1}{2}}$, where Re is taken to be the Reynolds number at which the numerical simulation was performed, i.e $\text{Re} = 5{,}000$.

[27] Visual inspections three-dimensional plots of the distribution of $\boldsymbol{\sigma}$ indicate that the oscillations $C_{\sigma,\boldsymbol{\theta}}$ for $0^\circ \leq \theta \lesssim 50^\circ$ are in fact small oscillations in $\boldsymbol{\sigma}$ as opposed to oscillatory errors resulting from the interpolating values onto geodesic lines.

non-body-conforming discretization, we find the magnitude of this error, $\Delta C_{\sigma,\boldsymbol{\theta}} \approx 0.1\mathrm{Re}^{-\frac{1}{2}} \approx 0.006$, to be acceptable.

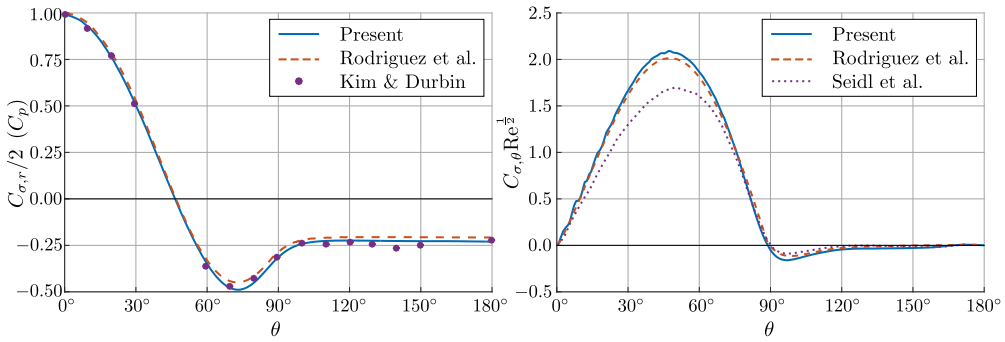

Figure 4.9: Time averaged pressure (*left*) and skin-friction (*right*) coefficients as functions of the polar angle, θ, for a sphere at Re = 3,700. Results compared to values reported by Rodriguez et al. [130] (DNS at Re = 3,700), Kim and Durbin [131] (exp. at Re = 4,200), and Seidl et al. [129] (DNS at Re = 5,000).

Lastly, we report in Table 4.6 mean values for the drag coefficient ($\overline{C_{\mathrm{D}}}$), base pressure coefficient ($\overline{C_{p,\mathrm{b}}}$), separation angle ($\overline{\theta_s}$), and polar locations of the minimum surface pressure ($\overline{\theta_{p,\mathrm{min}}}$) and of the maximum skin friction ($\overline{\theta_{\tau,\mathrm{max}}}$). With the exception of $\overline{C_{p,\mathrm{b}}}$, the present values are within 2.1% of those reported in [130]. The difference in $\overline{C_{p,\mathrm{b}}}$ is also seen to be small, i.e. approximately 2.3%, when compared to the maximum $\overline{C_p}$ shown in Figure 4.9. The value of $\overline{C_{\mathrm{D}}}$ reported in [99] (LES) is approximately 12% lower than the $\overline{C_{\mathrm{D}}}$ values reported here and in [130] (DNS). The grid refinement and mean turbulent statistics studies of [130] attribute this discrepancy in $\overline{C_{\mathrm{D}}}$ value to the sub-grid model used in the numerical experiments of [99].

4.6 Conclusions

A computationally efficient IB method for external, viscous, incompressible flows around immersed surfaces with prescribed kinematics has been presented. The IB-LGF method is a significant extension of the LGF flow solver [2], which retains the efficiency and robustness of the flow solver by coupling a Lagrange multiplier

		Re	$\overline{C_{\mathrm{D}}}$	$\overline{C_{p,\mathrm{b}}}$	$\overline{\theta_s}$	$\overline{\theta_{p,\min}}$	$\overline{\theta_{\tau,\max}}$
Present	DNS	3,700	0.389	−0.230	88.9°	73°	47°
YK06 [99]	LES	3,700	0.355	−0.194	90°	–	–
RB11 [130]	DNS	3,700	0.393	−0.207	89.3°	72°	48°
KD88 [131]	exp.	4,200	–	−0.224	–	–	–
SM98 [129]	DNS	5,000	0.38	–	89.5°	71°	50°

Table 4.6: Mean flow features of a sphere at Reynolds numbers between 3,700 and 5,000. Results from YK06 – Yun et al. [99], RB11 – Rodriguez et al. [130], KD88 – Kim and Durbin [131], and SM98 – Seidl et al. [129] are provided.

treatment of the discrete boundary forces and the discretized no-slip constraint with existing and new LGF techniques. The semi-discrete equations resulting from the formal spatial discretization of the incompressible Navier-Stokes equations on unbounded staggered Cartesian grids and the discrete delta function treatment of the IB regularization and interpolation operators is shown to constitute a DAE system of index 2. Using appropriately specialized order conditions for HERK integrators we proposed a few time integration schemes, which, when coupled with a viscous integrating factor technique, efficiently solve the discrete momentum ODE and the discrete divergence-free and no-slip constraints under a well-understood theoretical framework.

Fast flow solutions are facilitated by using a projection-like solver for the linear systems of equations arising from the implicit coupling the velocity, pressure, and boundary forces of the IF-HERK scheme. Unlike classical projection techniques, the present nested projection method is free of operator approximations, which in turn preserves the formal properties of the DAE time integration technique. This method is equivalent to a LU decomposition of the linear system and is formulated as two sequential intermediate velocity and pressure computation steps, followed by a single boundary force solution step, and finalized by two sequential pressure and velocity correction steps. Computational considerations for efficient iterative and direct boundary force solution techniques are discussed, and it is demonstrated that

for many practical flows involving rigid surfaces a Cholesky-based pre-processing technique results in force solutions that require negligible computation times. The pre-processing technique results in a flow solver that depends on the solution of one additional discrete elliptic problem, i.e. force correction on the pressure, which is shown, by virtue of the flexible source and target regions of the LGF solver, to require significantly less computation time than the discrete pressure Poisson problem inherent to the flow solver (less than 50% for the numerical experiments considered).

We implemented a parallel version of the IB-LGF method for the case of rigid surfaces following the block-wise adaptive grid of the LGF flow solver. Modifications to the adaptivity criteria, grid sub-domains, and parallel load balancing procedures were performed in order to efficiently and accurately capture the flow near immersed surfaces. Detailed spatial and temporal refinement studies on flows around spheres were used to verify the expected convergence rates of the formulation. Comparisons with previous numerical investigations on flows around rectangular flat plates and spheres at Reynolds numbers up to 3,700 were used to confirm the physical fidelity of computed solutions. We also showed that accurate surface stresses can be obtained from the computed boundary forces using the post-processing technique of [4]. All together, the present numerical experiments have demonstrated that the IB-LGF method can overcome many of the limitations of previous IB methods including robust rigid-surface solutions, accurate and efficient unbounded domain flow solutions, physical surface stress solutions, and the feasibility of fast, accurate numerical solutions to high (based on present day DNS capabilities) Reynolds numbers flows.

Acknowledgments

This work was partially supported by the United States Air Force Office of Scientific Research (FA950–09–1–0189) and the Caltech Field Laboratory for Optimized Wind Energy with Prof. John Dabiri as PI under the support of the Gordon and Betty

Moore Foundation.

APPENDICES

4.A Lattice Green's functions representations

The present formulation computes the action of L^{-1}, $\mathsf{E}(\alpha)$, and $\mathsf{K}(\alpha)$ as discrete convolutions, e.g. Eq (4.7), of G_L, $\mathsf{G}_\mathsf{E}(\alpha)$, and $\mathsf{G}_\mathsf{K}(\alpha)$. Expressions for these LGFs in terms of Fourier and Bessel integrals are given by

$$[\mathsf{G}_\mathsf{E}(\alpha)](\mathbf{n}) = \frac{1}{8\pi^3}\int_\Pi e^{-i\mathbf{n}\cdot\boldsymbol{\xi}-\sigma(\boldsymbol{\xi})}\,d\boldsymbol{\xi} = \prod_{q\in\mathbf{n}}\left[e^{-2\alpha}I_q(2\alpha)\right], \tag{4.33a}$$

$$(\Delta x)^2\mathsf{G}_\mathsf{L}(\mathbf{n}) = \frac{1}{8\pi^3}\int_\Pi \frac{e^{-i\mathbf{n}\cdot\boldsymbol{\xi}}}{\sigma(\boldsymbol{\xi})}\,d\boldsymbol{\xi} = -\int_0^\infty [\mathsf{G}_\mathsf{E}(t)](\mathbf{n})\,dt, \tag{4.33b}$$

$$(\Delta x)^2[\mathsf{G}_\mathsf{K}(\alpha)](\mathbf{n}) = \frac{1}{8\pi^3}\int_\Pi \frac{e^{-i\mathbf{n}\cdot\boldsymbol{\xi}-\sigma(\boldsymbol{\xi})}}{\sigma(\boldsymbol{\xi})}\,d\boldsymbol{\xi} = -\int_\alpha^\infty [\mathsf{G}_\mathsf{E}(t)](\mathbf{n})\,dt, \tag{4.33c}$$

where $\sigma(\boldsymbol{\xi}) = 2\cos(\xi_1) + 2\cos(\xi_2) + 2\cos(\xi_3) - 6$, $\Pi = (-\pi,\pi)^3$, and $I_n(z)$ is the modified Bessel function of the first kind of order n.

Here, we introduce a simple procedure for efficiently computing $[\mathsf{G}_\mathsf{K}(\alpha)]\,(\mathbf{n})$ and refer the reader to the discussions of [1] and [2] for examples of numerical techniques used to evaluate $\mathsf{G}_\mathsf{L}(\mathbf{n})$ and $[\mathsf{G}_\mathsf{E}(t)](\mathbf{n})$. We consider the partition of $[\mathsf{G}_\mathsf{K}(\alpha)]\,(\mathbf{n})$ given by

$$[\mathsf{G}_\mathsf{K}(\alpha)]\,(\mathbf{n}) = \mathsf{G}_\mathsf{L}(\mathbf{n}) + [\mathsf{R}(\alpha)]\,(\mathbf{n}), \tag{4.34}$$

where $[\mathsf{R}(\alpha)]\,(\mathbf{n}) = (\Delta x)^{-2}\int_0^\alpha [\mathsf{G}_\mathsf{E}(t)](\mathbf{n})\,dt$. The combined look-up table and asymptotic expansion approach of [1] is used to compute the first term, $\mathsf{G}_\mathsf{L}(\mathbf{n})$, and an adaptive Gauss-Kronrod (GK) integration scheme is used to evaluate the second term, $[\mathsf{R}(\alpha)]\,(\mathbf{n})$. For large values of $\mathbf{n}$ few, if any, subdivisions are required by the GK scheme since the value of $[\mathsf{R}(\alpha)]\,(\mathbf{n})$ is significantly smaller than the value of $\mathsf{G}_\mathsf{L}(\mathbf{n})$.[28] Lastly, we note that evaluating discrete convolution of LGFs using the

[28] The leading order term in the asymptotic expansion of $\mathsf{G}_\mathsf{L}(\mathbf{n})$ is $1/(4\pi|\mathbf{n}|)$ [1]. For a fixed α, the integrand $[\mathsf{R}(\alpha)]\,(\mathbf{n})$, i.e. $[\mathsf{G}_\mathsf{E}(t)](\mathbf{n})$, decays faster than any exponential [2].

LGF-FMM [1] only requires the point-wise values of LGFs to be computed once, as a pre-processing step, per simulation.

4.B Half-explicit Runge-Kutta schemes

The IF-HERK schemes used to perform the numerical experiments of Section 4.5 are:

Scheme A

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ 1 & \frac{\sqrt{3}}{3} & \frac{3-\sqrt{3}}{3} \\ \hline & \frac{3+\sqrt{3}}{6} & \frac{-\sqrt{3}}{3} \quad \frac{3+\sqrt{3}}{6} \end{array}$$

Scheme B

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ 1 & -1 & 2 & \\ \hline & 0 & \frac{3}{4} & \frac{1}{4} \end{array}$$

Scheme C

$$\begin{array}{c|ccc} 0 & & & \\ \frac{8}{15} & \frac{8}{15} & & \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} & \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$$

Scheme D

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & \frac{1}{4} & 0 & 0 & 1 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

(4.35)

The expected order of accuracy for Schemes A–D based on the simplified HERK order-conditions discussed in Section 4.2.3 are included in Table 4.7. As a point of comparison, Table 4.7 also includes the expected order of accuracy for problems with no immersed surfaces (i.e. Eq. (4.8a) and (4.8b) with $\mathfrak{f} = 0$) and for general semi-explicit DAEs of index 2 (i.e. Eq. (4.9)).

Scheme	y	z	y^+	z^+	y^*	z^*
A	2	2	2	2	2	2
B	3	2	3	2	3	2
C	2	1	3	1	2	1
D	3	1	4	1	2	1

Table 4.7: Expected order of accuracy of HERK schemes for the solution variable y (velocity) and for the constraint variable z (pressure and body forces). The superscripts $+$ and $*$ denotes values for problems with no immersed surface and for general semi-explicit DAEs of index 2, respectively.

Scheme B is the only three-stage scheme with a third-order accurate solution variable for general semi-explicit DAEs of index 2 [66]. The RK coefficients of Scheme C and Scheme D correspond to the popular three-stage fractional step method of [68] and

the four-stage "original" RK method, respectively. As discussed in [2], Scheme A has the advantage of having equispaced RK nodes, i.e. c_i's, which reduce the number of distinct integrating factors. Fewer distinct integrating factors reduces the number of pre-processing operations and lowers the storage requirements of the LGF-FMM [1] and of the Cholesky-based force Schur complement technique discussed in Section 4.3.2. Scheme A and D only require two distinct integrating factors (with one of them being the identity operator), as opposed to the three distinct integrating factors required by Scheme B and C.

In the absence of an immersed surface, a linear stability analysis about a uniform base flow $\mathbf{U}$ of the IF-HERK method [2] indicates that solutions are subject to the CFL condition

$$\mathrm{CFL} = \frac{|\mathbf{U}|\Delta t}{\Delta x} < \mathrm{CFL}_{\max}, \tag{4.36}$$

where $\mathrm{CFL}_{\max}$ depends on the RK coefficients of the scheme. The value of $\mathrm{CFL}_{\max}$ is unity for Schemes A–C and $\frac{2\sqrt{2}}{\sqrt{3}}$ for Scheme D. In practice, we expect solutions to the non-linear governing equations to remain stable as long as the CFL conditions resulting from linearizing the flow at each grid point are satisfied, i.e. as long as $\max(|\mathbf{u}|)\Delta t/\Delta x < \mathrm{CFL}_{\max}$.

The CFL condition $\Delta x \sim \Delta t$ and the second-order accuracy (in the absence of immersed surfaces) of the present solver imply that the potential reduction in the operation count resulting from higher than second-order HERK schemes is limited. As a result, the lower pre-processing cost of Scheme A compared to Scheme C makes Scheme A the preferred HERK scheme for the present formulation. Here, we did not consider Schemes C and D to be potential "preferred" schemes since they are only first-order accurate in the constraint variables.

Chapter 5

CONCLUSIONS

This thesis presents a fundamentally new approach to numerically solving viscous, incompressible flows on unbounded fluid domains. The novelty of the approach stems from the use of lattice Green's function techniques to obtain practical solutions to difference equations resulting from the discretization of the Navier-Stokes equations on unbounded regular grids.

In Chapter 2 solutions to difference equations on unbounded Cartesian grids with compactly supported source terms are shown to be expressible as discrete convolutions between the lattice Green's function of difference operators and the discrete source terms. This approach enables the computation of practical solutions to elliptic difference equations relevant to incompressible flows by limiting operations to a finite region of non-negligible source terms. The $\mathcal{O}(N^2)$ operations required to evaluate the resulting discrete convolutions with the straightforward approach is reduced to $\mathcal{O}(N)$ operations by a new FMM specifically designed to solve difference equations. The LGF-FMM is a kernel-independent method that combines the hierarchical structure of traditional FMMs with piece-wise polynomial interpolation kernel-compression techniques and fast FFT-based discrete convolution methods to solve elliptic, constant-coefficient, difference equations on unbounded Cartesian grids. In addition to its asymptotic linear algorithmic complexity, it is demonstrated for the case of discrete 7-pt Poisson problems that the LGF-FMM achieves computation rates and parallel scalings comparable to those obtained for Poisson problems by other state of the art FMMs.

In Chapter 3, the LGF-FMM is used as the basic building block of a fast, robust parallel incompressible flow solver. It is shown that the finite-volume discretization of the incompressible Navier-Stokes equations on an unbounded staggered Cartesian grid is efficiently integrated in time by combining a discrete viscous integration factor

and a half-explicit Runge-Kutta technique. The resulting equations are shown to be solved in a finite number of operations and in linear algorithmic complexity by using the LGF-FMM to evaluate the discrete pressure Poisson equation and integrating factors arising in an approximation-free projection method. A block-wise structured adaptive grid and velocity refresh technique are implemented so as to efficiently compute solutions of unsteady flows by limiting operations to small computational grids that track relevant flow regions by adding and removing grid blocks. An extensive set of numerical experiments on the evolution of thin vortex rings at Reynolds numbers up to 20,000 are used to verify the accuracy and computational efficiency of the formulation.

In Chapter 4, a fast, robust immersed boundary method is constructed using the LGF flow solver. Following a Lagrange multiplier treatment of the regularized boundary forces, the IB-LGF method extends the IF-HERK time integration scheme and the projection method of the LGF flow solver to efficiently simulate flows around surfaces with prescribed motions. It is shown that significant operation count reductions are obtained by taking advantage of the flexible source and target regions of the LGF-FMM when evaluating terms involving the compactly supported IB force regularization and velocity interpolation operators. The base algorithm is further accelerated for a wide-range of practical flows through the implementation of a dense linear algebra pre-processing technique for computing boundary forces. The expected convergence rates and the physical fidelity of computed solutions are verified by performing grid refinement studies and comparisons to previous investigations on flows around low-aspect-ratio flat plates and spheres at Reynolds numbers up to 3,700. It is also shown that post-processing the computed boundary forces with the kernel smoothing technique [4] produces accurate point-wise surface stresses for the test case of a sphere at $\mathrm{Re} = 3{,}700$.

There are several extensions to the LGF techniques discussed in this thesis that can be readily implemented in order to significantly enlarge the range of fluid flow

problems that can be practically investigated. Some of the extensions discussed below are being actively developed by other members of the Computational Flow Physics Group.

The present methods can be readily extended to handle 2D-unbounded and 2D-unbounded / 1D-periodic problems. Aside from a few implementation details, a 2D-unbounded flow solver can be obtained using the 2D version of the standard difference operators presently employed. Expressions for the relevant 2D LGFs a readily deduced from the discussions of previous chapters and are explicitly provided for the 4-pt Laplacian in [5, 16, 18]. Similar considerations are necessary for a 2D-unbounded / 1D-periodic flow solver, but this version is likely to be implemented using a Fourier treatment of the 1D periodic direction, which in turn would require solutions to discrete Helmholtz equations with imaginary wavenumbers (modified Helmholtz equations). A Fourier integral representation for the LGF of the discrete modified Helmholtz operator is readily obtained following the procedures discussed in Section 2.2.1. Higher order discretization schemes on staggered Cartesian grids can also be incorporated into the present framework through considerations similar to those mentioned above. It is worth emphasizing that the pre-processing technique of the LGF-FMM prevents the potentially large computational cost of numerically evaluating LGFs from affecting the run-time cost of the flows solver.

The Immersed Boundary method is only one of the many embedded boundary methods that compute solutions to PDEs over non-trivial domains using regular-grid discretizations. Higher order Immersed Interface [134], Ghost Fluid [135], Volume Penalty [136], and Smooth Extension [95] methods can be used within the LGF framework to remove some of the practical limitations imposed by the first-order accuracy of the Immersed Boundary method. Although some of these methods have the advantage of being higher-order, they often require more sophisticated implementations, impose constraints on the allowable geometries, and result in discretized equations that are not readily compatible with the fast solution techniques of the

IB-LGF method. An alternative approach to reducing the error (without increasing the order) of the IB method is to use "smooth" discrete delta functions [115, 120, 137], but the efficacy of this approach remains an active area of research.

The inefficiencies inherent to use of uniform grids to accurately resolve flows with localized small-scale features can be reduced by using standard local grid refinement techniques [138, 139]. These techniques have been previously used in combination with IB methods to efficiently resolve thin laminar boundary layers on immersed surfaces [101, 140, 141], and are expected to also be compatible with the present IB method. Block-wise adaptive mesh refinement techniques, e.g. [142, 143], are particularly compatible with the block-wise grid partitioning employed by the present algorithms. Furthermore, the octree and grid hierarchy already implemented for the LGF-FMM are expected to expedite the implementation of block-wise locally-refined flow solvers.

The prohibitive grid resolution requirements of direct numerical simulations dictates the use of turbulence models for computing practical solutions to high Reynolds number flows. Large Eddy Simulation (LES) techniques, such as those reviewed in [144–147], are suitable candidates for reducing the range of time- and length-scales that need to be resolved. The conservation and stability properties of the LGF flow solver are expected to facilitate the robust implementation of these techniques within the LGF framework. Furthermore, for flows around immersed surfaces with turbulent boundary layers, a LES implementation could be extended to include wall-models, such those discussed in [148–152], in an attempt to model the effect of near-wall eddies at sub-grid scales.

The LGF techniques described in this thesis have efficient nodal distributions, automatically impose natural boundary conditions, are compatible with fast free-space solvers, and have provable conservation and stability properties. Altogether, the present collection of LGF techniques provides a new framework for efficient and robust numerical simulations of incompressible flows, and is expected to serve as

a solid base for future numerical methods capable of investigating the increasingly complex flows of emerging scientific and engineering applications.

REFERENCES

[1] S. Liska and T. Colonius. "A parallel fast multipole method for elliptic difference equations". *Journal of Computational Physics* 278 (2014), pp. 76–91. arXiv: 1402.6081 [physics.comp-ph].

[2] S. Liska and T. Colonius. "A fast lattice Green's function method for solving viscous incompressible flows on unbounded domains". *Accepted for publication in Journal of Computational Physics* (2015). arXiv: 1601.00035 [physics.flu-dyn].

[3] S. Liska and T. Colonius. "A fast immersed boundary method for external incompressible viscous flows using lattice Green's functions". *Submitted to Journal of Computational Physics* (2016). arXiv: 1604.01814 [physics.flu-dyn].

[4] A. Goza, S. Liska, B. Morley, and T. Colonius. "Accurate computation of surface stresses and forces with immersed boundary methods". *Revised and resubmitted to Journal of Computational Physics* (2015). arXiv: 1603.02306 [physics.flu-dyn].

[5] W. H. McCrea and F. J. W. Whipple. "Random paths in two and three dimensions". *Proceedings of the Royal Society of Edinburgh* 60 (1940), pp. 281–298.

[6] E. W. Montroll and R. B. Potts. "Effect of defects on lattice vibrations". *Physical Review* 100.2 (1955), pp. 525–543.

[7] E. N. Economou. *Green's functions in quantum physics*. Berlin: Springer-Verlag, 1984.

[8] J. H. Bramble and B. E. Hubbard. "On the formulation of finite difference analogues of the Dirichlet problem for Poisson's equation". *Numerische Mathematik* 4.1 (1962), pp. 313–327.

[9] O. Buneman. "Analytic inversion of the five-point Poisson operator". *Journal of Computational Physics* 8.3 (1971), pp. 500–505.

[10] A. Gillman and P. G. Martinsson. "Fast and accurate numerical methods for solving elliptic difference equations defined on lattices". *Journal of Computational Physics* 229.24 (2010), pp. 9026–9041.

[11] A. Gillman and P. G. Martinsson. "A fast solver for Poisson problems on infinite regular lattices". *Journal of Computational and Applied Mathematics* 258 (2014), pp. 42–56.

[12] D. N. Arnold, P. B. Bochev, R. B. Lehoucq, R. A. Nicolaides, and M. Shashkov. *Compatible spatial discretizations*. Vol. 142. The IMA Volumes in Mathematics and its Applications. New York: Springer New York, 2006.

[13] B. Perot. "Discrete conservation properties of unstructured mesh schemes". *Annual Review of Fluid Mechanics* 43.1 (2011), pp. 299–318.

[14] M. L. Glasser and I. J. Zucker. "Extended Watson integrals for the cubic lattices". *Proceedings of the National Academy of Sciences* 74.5 (1977), pp. 1800–1801.

[15] R. T. Delves and G. S. Joyce. "On the Green function for the anisotropic simple cubic lattice". *Annals of Physics* 291.1 (2001), pp. 71–133.

[16] R. J. Duffin and E. P. Shelly. "Difference equations of polyharmonic type". *Duke Mathematical Journal* 25.2 (1958), pp. 209–238.

[17] M. Mangad. "Asymptotic expansions of Fourier transforms and discrete polyharmonic Green's functions". *Pacific Journal of Mathematics* 20 (1967), pp. 85–98.

[18] P. G. Martinsson and G. J. Rodin. "Asymptotic expansions of lattice Green's functions". *Proceedings of the Royal Society of London Series A* 458.2027 (2002), pp. 2609–2622.

[19] P. G. Martinsson and G. J. Rodin. "Boundary algebraic equations for lattice problems". *Proceedings of the Royal Society of London Series A* (2009).

[20] L. Greengard and V. Rokhlin. "A fast algorithm for particle simulations". *Journal of Computational Physics* 73.2 (1987), pp. 325–348.

[21] L. Greengard and V. Rokhlin. "A new version of the fast multipole method for the Laplace equation in three dimensions". *Acta Numerica* 6.1 (1997), pp. 229–269.

[22] Z. Gimbutas and V. Rokhlin. "A generalized fast multipole method for nonoscillatory kernels". *SIAM Journal on Scientific Computing* 24.3 (2003), pp. 796–817.

[23] L. Ying, G. Biros, and D. Zorin. "A kernel-independent adaptive fast multipole algorithm in two and three dimensions". *Journal of Computational Physics* 196.2 (2004), pp. 591–626.

[24] P. G. Martinsson and V. Rokhlin. "An accelerated kernel-independent fast multipole method in one dimension". *SIAM Journal on Scientific Computing* 29.3 (2007), pp. 1160–1178.

[25] W. Fong and E. Darve. "The black-box fast multipole method". *Journal of Computational Physics* 228.23 (2009), pp. 8712–8725.

[26] B. Zhang, J. Huang, N. P. Pitsianis, and X. Sun. "A Fourier-series-based kernel-independent fast multipole method". *Journal of Computational Physics* 230.15 (2011), pp. 5807–5821.

[27] A. Dutt, M. Gu, and V. Rokhlin. "Fast algorithms for polynomial interpolation, integration, and differentiation". *SIAM Journal on Numerical Analysis* 33.5 (1996), pp. 1689–1711.

[28] C. Berman. "Grid-multipole calculations". *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1082–1091.

[29] J. R. Phillips and J. K. White. "A precorrected-FFT method for electrostatic analysis of complicated 3D structures". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 16.10 (1997), pp. 1059–1072.

[30] P. Chatelain and P. Koumoutsakos. "A Fourier-based elliptic solver for vortical flows with periodic and unbounded directions". *Journal of Computational Physics* 229.7 (2010), pp. 2425–2431.

[31] L. N. Trefethen. *Spectral methods in MATLAB*. Vol. 10. Philadelphia, PA: SIAM, 2000.

[32] A. Quarteroni, F. Saleri, and P. Gervasio. *Scientific computing with MATLAB and Octave*. Berlin: Springer, 2010, pp. 75–106.

[33] S. G. Johnson and M. Frigo. "A modified split-radix FFT with fewer arithmetic operations". *IEEE Transactions on Signal Processing* 55.1 (2007), pp. 111–119.

[34] G. M. Morton. "A computer oriented geodetic data base and a new technique in file sequencing". *IBM* (1966).

[35] H. Langston, L. Greengard, and D. Zorin. "A free-space adaptive FMM–based PDE solver in three dimensions". *Communications in Applied Mathematics and Computational Science* 6.1 (2011), pp. 79–122.

[36] L. Ying, G. Biros, D. Zorin, and H. Langston. "A new parallel kernel-independent fast multipole method". *Proceedings of the 2003 ACM / IEEE Conference on Supercomputing*. 2003, pp. 14–14.

[37] A. Gholaminejad, D. Malhotra, H. Sundar, and G. Biros. "FFT, FMM, or Multigrid? A comparative study of state-of-the-art Poisson solvers". *Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium 2014*. 2014.

[38] J. Cserti. "Application of the lattice Green's function for calculating the resistance of an infinite network of resistors". *American Journal of Physics* 68 (2000), p. 896.

[39] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. New York: Dover, 1972.

[40] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message-passing interface*. Vol. 1. MIT press, 1999.

[41] S. V. Tsynkov. "Numerical solution of problems on unbounded domains. A review". *Applied Numerical Mathematics* 27.4 (1998), pp. 465–532.

[42] T. Colonius. "Modeling artificial boundary conditions for compressible flow". *Annual Review of Fluid Mechanics* 36 (2004), pp. 315–345.

[43] D. S. Pradeep and F. Hussain. "Effects of boundary condition in numerical simulations of vortex dynamics". *Journal of Fluid Mechanics* 516 (2004), pp. 115–124.

[44] S. Dong, G. E. Karniadakis, and C. Chryssostomidis. "A robust and accurate outflow boundary condition for incompressible flow simulations on severely-truncated unbounded domains". *Journal of Computational Physics* 261 (2014), pp. 83–105.

[45] A. Leonard. "Vortex methods for flow simulation". *Journal of Computational Physics* 37.3 (1980), pp. 289–335.

[46] G. S. Winckelmans and A. Leonard. "Contributions to Vortex Particle Methods for the Computation of Three-Dimensional Incompressible Unsteady Flows". *Journal of Computational Physics* 109.2 (1993), pp. 247–273.

[47] M. S. Warren and J. K. Salmon. "A parallel hashed oct-tree n-body algorithm". *Proceedings of the 1993 ACM / IEEE conference on Supercomputing*. ACM. 1993, pp. 12–21.

[48] H. Cheng, L. Greengard, and V. Rokhlin. "A fast adaptive multipole algorithm in three dimensions". *Journal of Computational Physics* 155.2 (1999), pp. 468–498.

[49] P. Ploumhans and G. S. Winckelmans. "Vortex Methods for High-Resolution Simulations of Viscous Flow Past Bluff Bodies of General Geometry". *Journal of Computational Physics* 165.2 (2000), pp. 354–406.

[50] G.-H. Cottet and P. D. Koumoutsakos. *Vortex methods: theory and practice*. Cambridge University Press, 2000.

[51] G. S. Winckelmans. "Vortex Methods". *Encyclopedia of Computational Mechanics*. John Wiley & Sons, Ltd, 2004.

[52] R. Cocle, G. Winckelmans, and G. Daeninck. "Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations". *Journal of Computational Physics* 227.21 (2008), pp. 9091–9120.

[53] J. T. Rasmussen, G.-H. Cottet, and J. H. Walther. "A multiresolution remeshed Vortex-In-Cell algorithm using patches". *Journal of Computational Physics* 230.17 (2011), pp. 6742–6755.

[54] M. M. Hejlesen, J. T. Rasmussen, P. Chatelain, and J. H. Walther. "A high order solver for the unbounded Poisson equation". *Journal of Computational Physics* 252 (2013), pp. 458–467.

[55] P. G. Saffman. *Vortex dynamics*. Cambridge University Press, 1992.

[56] R. A. Nicolaides and X. Wu. "Covolume solutions of three-dimensional div-curl equations". *SIAM Journal on Numerical Analysis* 34.6 (1997), pp. 2195–2203.

[57] X. Zhang, D. Schmidt, and B. Perot. "Accuracy and conservation properties of a three-dimensional unstructured staggered mesh scheme for fluid dynamics". *Journal of Computational Physics* 175.2 (2002), pp. 764–791.

[58] 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 (1965), pp. 2182–2189.

[59] D. K. Lilly. "On the computational stability of numerical solutions of time-dependent non-linear geophysical fluid dynamics problems". *Monthly Weather Review* 93.11 (1965).

[60] Y. Morinishi, T. S. Lund, O. V. Vasilyev, and P. Moin. "Fully conservative higher order finite difference schemes for incompressible flow". *Journal of Computational Physics* 143.1 (1998), pp. 90–124.

[61] R. A. Nicolaides. "Direct discretization of planar div-curl problems". *SIAM Journal on Numerical Analysis* 29.1 (1992), pp. 32–56.

[62] B. Perot. “Conservation properties of unstructured staggered mesh schemes”. *Journal of Computational Physics* 159.1 (2000), pp. 58–89.

[63] E. Hairer and G. Wanner. *Solving ordinary differential equations II: stiff and differential-algebraic problems*. Berlin Springer, 1996.

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

[65] E. Hairer, C. Lubich, and M. Roche. *The numerical solution of differential-algebraic systems by Runge-Kutta methods*. Lecture Notes in Mathematics. Springer-Verlag, 1989.

[66] V. Brasey and E. Hairer. “Half-explicit Runge-Kutta methods for differential-algebraic systems of index 2”. *SIAM Journal on Numerical Analysis* 30.2 (1993), pp. 538–552.

[67] B. Sanderse and B. Koren. “Accuracy analysis of explicit Runge-Kutta methods applied to the incompressible Navier-Stokes equations”. *Journal of Computational Physics* 231.8 (2012), pp. 3041–3063.

[68] H. Le and P. Moin. “An improvement of fractional step methods for the incompressible Navier-Stokes equations”. *Journal of Computational Physics* 92.2 (1991), pp. 369–379.

[69] T. Colonius and K. Taira. “A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions”. *Computer Methods in Applied Mechanics and Engineering* 197.25–28 (2008), pp. 2131–2146.

[70] B. Perot. “An analysis of the fractional step method”. *Journal of Computational Physics* 108.1 (1993), pp. 51–58.

[71] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*. Vol. 6. Springer Science & Business Media, 1999.

[72] W. Chang, F. Giraldo, and B. Perot. “Analysis of an exact fractional step method”. *Journal of Computational Physics* 180.1 (2002), pp. 183–199.

[73] S. K. Stanaway, B. J. Cantwell, and P. R. Spalart. “A numerical study of viscous vortex rings using a spectral method”. *NASA Technical Memorandum* (1988), p. 101041.

[74] P. J. Archer, T. G. Thomas, and G. N. Coleman. "Direct numerical simulation of vortex ring evolution from the laminar to the early turbulent regime". *Journal of Fluid Mechanics* 598 (2008), pp. 201–226.

[75] M. Cheng, J. Lou, and T. T. Lim. "Leapfrogging of multiple coaxial viscous vortex rings". *Physics of Fluids* 27.3 (2015), p. 031702.

[76] H. K. Moffatt and A. Tsinober. "Helicity in laminar and turbulent flow". *Annual Review of Fluid Mechanics* 24.1 (1992), pp. 281–312.

[77] P. G. Saffman. "The velocity of viscous vortex rings". *Studies in Applied Mathematics* 49 (1970), pp. 371–380.

[78] K. Shariff, R. Verzicco, and P. Orlandi. "A numerical study of three-dimensional vortex ring instabilities: viscous corrections and early nonlinear stage". *Journal of Fluid Mechanics* 279 (1994), pp. 351–375.

[79] D. G. Akhmetov. *Vortex rings*. Springer Science & Business Media, 2009.

[80] I. S. Sullivan, J. J. Niemela, R. E. Hershberger, D. Bolster, and R. J. Donnelly. "Dynamics of thin vortex rings". *Journal of Fluid Mechanics* 609 (2008), pp. 319–347.

[81] Y. Fukumoto. "Global time evolution of viscous vortex rings". *Theoretical and Computational Fluid Dynamics* 24.1-4 (2010), pp. 335–347.

[82] Y. Fukumoto and H. K. Moffatt. "Motion and expansion of a viscous vortex ring. Part 1. A higher-order asymptotic formula for the velocity". *Journal of Fluid Mechanics* 417 (2000), pp. 1–45.

[83] M. Bergdorf, P. Koumoutsakos, and A. Leonard. "Direct numerical simulations of vortex rings at Re = 7500". *Journal of Fluid Mechanics* 581 (2007), pp. 495–505.

[84] M. M. Hejlesen, H. J. Spietz, and J. H. Walther. "Simulations of a single vortex ring using an unbounded, regularized particle-mesh based vortex method". *6th International Conference on Vortex Flows and Vortex Models (ICVFM 2014)*. 2014.

[85] C. S. Peskin. "The immersed boundary method". *Acta numerica* 11 (2002), pp. 479–517.

[86] R. Mittal and G. Iaccarino. "Immersed boundary methods". *Annual Review of Fluid Mechanics* 37 (2005), pp. 239–261.

[87] G. Hou, J. Wang, and A. Layton. "Numerical methods for fluid-structure interaction: a review". *Communications in Computational Physics* 12.2 (2012), pp. 337–377.

[88] C. S. Peskin. "Flow patterns around heart valves: a numerical method". *Journal of Computational Physics* 10.2 (1972), pp. 252–271.

[89] R. P. Beyer and R. J. LeVeque. "Analysis of a One-Dimensional Model for the Immersed Boundary Method". *SIAM Journal on Numerical Analysis* 29.2 (1992), pp. 332–364.

[90] D. Goldstein, R. Handler, and L. Sirovich. "Modeling a No-Slip Flow Boundary with an External Force Field". *Journal of Computational Physics* 105.2 (1993), pp. 354–366.

[91] R. Glowinski, T. W. Pan, and J. Périaux. "Distributed Lagrange multiplier methods for incompressible viscous flow around moving rigid bodies". *Computer Methods in Applied Mechanics and Engineering* 151.1–2 (1998), pp. 181–194.

[92] M.-C. Lai and C. S. Peskin. "An Immersed Boundary Method with Formal Second-Order Accuracy and Reduced Numerical Viscosity". *Journal of Computational Physics* 160.2 (2000), pp. 705–719.

[93] K. Taira and T. Colonius. "The immersed boundary method: a projection approach". *Journal of Computational Physics* 225.2 (2007), pp. 2118–2137.

[94] B. Kallemov, A. Bhalla, B. E. Griffith, and A. Donev. "An immersed boundary method for rigid bodies". *Communications in Applied Mathematics and Computational Science* 11.1 (2016), pp. 79–141.

[95] D. B. Stein, R. D. Guy, and B. Thomases. "Immersed boundary smooth extension: A high-order method for solving PDE on arbitrary smooth domains using Fourier spectral methods". *Journal of Computational Physics* 304 (2016), pp. 252–274.

[96] N. A. Patankar, P. Singh, D. D. Joseph, R. Glowinski, and T. W. Pan. "A new formulation of the distributed Lagrange multiplier / fictitious domain method for particulate flows". *International Journal of Multiphase Flow* 26.9 (2000), pp. 1509–1524.

[97] A. Wachs. "Numerical simulation of steady Bingham flow through an eccentric annular cross-section by distributed Lagrange multiplier / fictitious domain and augmented Lagrangian methods". *Journal of Non-Newtonian Fluid Mechanics* 142.1–3 (2007), pp. 183–198.

[98] A. P. S. Bhalla, R. Bale, B. E. Griffith, and N. A. Patankar. "A unified mathematical framework and an adaptive numerical method for fluid-structure interaction with rigid, deforming, and elastic bodies". *Journal of Computational Physics* 250 (2013), pp. 446–476.

[99] G. Yun, D. Kim, and H. Choi. "Vortical structures behind a sphere at subcritical Reynolds numbers". *Physics of Fluids* 18.1 (2006).

[100] S. Wang and X. Zhang. "An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows". *Journal of Computational Physics* 230.9 (2011), pp. 3479–3499.

[101] A. M. Roma, C. S. Peskin, and M. J. Berger. "An Adaptive Version of the Immersed Boundary Method". *Journal of Computational Physics* 153.2 (1999), pp. 509–534.

[102] B. E. Griffith, R. D. Hornung, D. M. McQueen, and C. S. Peskin. "An adaptive, formally second order accurate version of the immersed boundary method". *Journal of Computational Physics* 223.1 (2007), pp. 10–49.

[103] M. Beddhu, L. K. Taylor, and D. L. Whitfield. "Strong conservative form of the incompressible Navier–Stokes equations in a rotating frame with a solution procedure". *Journal of Computational Physics* 128.2 (1996), pp. 427–437.

[104] D. Kim and H. Choi. "Immersed boundary method for flow around an arbitrarily moving body". *Journal of Computational Physics* 212.2 (2006), pp. 662–680.

[105] H.-C. Tsai and T. Colonius. "Coriolis Effect on Dynamic Stall in a Vertical Axis Wind Turbine at Moderate Reynolds Number". *32nd AIAA applied aerodynamics conference*. AIAA. Atlanta, GA, 2014.

[106] A. J. Chori. "A numerical method for solving incompressible viscous flow problems". *Journal of Computational Physics* 2.1 (1967), pp. 12–26.

[107] R. Temam. "Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (I)". *Archive for Rational Mechanics and Analysis* 32.2 (1969), pp. 135–153.

[108] Y. Cao and Q. Li. "Highest order multistep formula for solving index-2 differential-algebraic equations". *BIT Numerical Mathematics* 38.4 (1998), pp. 663–673.

[109] C. Fuhrer. *Numerical Methods in Multibody Dynamics*. Springer Science & Business Media, 2013.

[110] L. S. Blackford et al. *ScaLAPACK users' guide*. Vol. 4. SIAM, 1997.

[111] J. R. Shewchuk. *An introduction to the conjugate gradient method without the agonizing pain*. 1994.

[112] Y. Saad. "A flexible inner-outer preconditioned GMRES algorithm". *SIAM Journal on Scientific Computing* 14.2 (1993), pp. 461–469.

[113] J. Chen, L. C. McInnes, and H. Zhang. "Analysis and practical use of flexible BiCGSTAB". *Journal of Scientific Computing* (2012), pp. 1–23.

[114] E. B. Saff and A. B. J. Kuijlaars. "Distributing many points on a sphere". *The Mathematical Intelligencer* 19.1 (1997), pp. 5–11.

[115] X. Yang, X. Zhang, Z. Li, and G. W. He. "A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations". *Journal of Computational Physics* 228.20 (2009), pp. 7821–7836.

[116] A.-K. Tornberg and B. Engquist. "Numerical approximations of singular source terms in differential equations". *Journal of Computational Physics* 200.2 (2004), pp. 462–488.

[117] Y. Mori. "Convergence proof of the velocity field for a Stokes flow immersed boundary method". *Communications on Pure and Applied Mathematics* 61.9 (2008), pp. 1213–1263.

[118] K.-Y. Chen, K.-A. Feng, Y. Kim, and M.-C. Lai. "A note on pressure accuracy in immersed boundary method for Stokes flow". *Journal of Computational Physics* 230.12 (2011), pp. 4377–4383.

[119] A.-K. Tornberg and B. Engquist. "Regularization techniques for numerical approximation of PDEs with singularities". *Journal of Scientific Computing* 19.1-3 (2003), pp. 527–552.

[120] Y.-x. Bao, J. Kaye, and C. S. Peskin. "A Gaussian-like immersed boundary kernel with improved translational invariance and smoothness" (2015). arXiv: 1505.07529 [math.NA].

[121] K. Taira and T. Colonius. "Three-dimensional flows around low-aspect-ratio flat-plate wings at low Reynolds numbers". *Journal of Fluid Mechanics* 623 (2009), pp. 187–207.

[122] S. Wang, X. Zhang, G. He, and T. Liu. "A lift formula applied to low-Reynolds-number unsteady flows". *Physics of Fluids* 25.9 (2013).

[123] T. A. Johnson and V. C. Patel. "Flow past a sphere up to a Reynolds number of 300". *Journal of Fluid Mechanics* 378 (1999), pp. 19–70.

[124] R. Mittal. "Planar symmetry in the unsteady wake of a sphere". *AIAA Journal* 37.3 (1999), pp. 388–390.

[125] A. G. Tomboulides and S. A. Orszag. "Numerical investigation of transitional and weak turbulent flow past a sphere". *Journal of Fluid Mechanics* 416 (2000), pp. 45–73.

[126] P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. "Vortex Methods for Direct Numerical Simulation of Three-Dimensional Bluff Body Flows: Application to the Sphere at Re = 300, 500, and 1000". *Journal of Computational Physics* 178.2 (2002), pp. 427–463.

[127] J. Kim, D. Kim, and H. Choi. "An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries". *Journal of Computational Physics* 171.1 (2001), pp. 132–150.

[128] G. S. Constantinescu and K. D. Squires. "LES and DES Investigations of Turbulent Flow over a Sphere at Re = 10,000". English. *Flow, Turbulence and Combustion* 70.1-4 (2003), pp. 267–298.

[129] V. Seidl, S. Muzaferija, and M. Perić. "Parallel DNS with Local Grid Refinement". *Applied Scientific Research* 59.4 (1989), pp. 379–394.

[130] I. Rodriguez, R. Borrel, O. Lehmkuhl, C. D. Perez Segarra, and A. Oliva. "Direct numerical simulation of the flow over a sphere at Re = 3700". *Journal of Fluid Mechanics* 679 (2011), pp. 263–287.

[131] H. J. Kim and P. A. Durbin. "Observations of the frequencies in a sphere wake and of drag increase by acoustic excitation". *Physics of Fluids* 31.11 (1988), pp. 3260–3265.

[132] H. Schlichting and K. Gersten. *Boundary-layer theory*. Springer Science & Business Media, 2003.

[133] J. C. R. Hunt, A. A. Wray, and P. Moin. "Eddies, Stream, and Convergence Zones in Turbulent Flows". *Center for Turbulence Research CTR-S88* (1988).

[134] Z. Li and K. Ito. *The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains*. Vol. 33. SIAM, 2006.

[135] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. "A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)". *Journal of computational physics* 152.2 (1999), pp. 457–492.

[136] P. Angot, C.-H. Bruneau, and P. Fabrie. "A penalization method to take into account obstacles in incompressible viscous flows". *Numerische Mathematik* 81.4 (1999), pp. 497–520.

[137] Y. Liu and Y. Mori. "Properties of discrete delta functions and local convergence of the immersed boundary method". *SIAM Journal on Numerical Analysis* 50.6 (2012), pp. 2986–3015.

[138] M. J. Berger and J. Oliger. "Adaptive mesh refinement for hyperbolic partial differential equations". *Journal of computational Physics* 53.3 (1984), pp. 484–512.

[139] M. J. Berger and P. Colella. "Local adaptive mesh refinement for shock hydrodynamics". *Journal of computational Physics* 82.1 (1989), pp. 64–84.

[140] B. E. Griffith and C. S. Peskin. "On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems". *Journal of Computational Physics* 208.1 (2005), pp. 75–105.

[141] M. Vanella, A. Posa, and E. Balaras. "Adaptive mesh refinement for immersed boundary methods". *Journal of Fluids Engineering* 136.4 (2014), p. 040909.

[142] A. Nissen, G. Kreiss, and M. Gerritsen. "High order stable finite difference methods for the Schrödinger equation". *Journal of Scientific Computing* 55.1 (2013), pp. 173–199.

[143] A. Nissen, K. Kormann, M. Grandin, and K. Virta. "Stable difference methods for block-oriented adaptive grids". *Journal of Scientific Computing* (), pp. 1–26.

[144] M. Lesieur and O. Metais. "New trends in large-eddy simulations of turbulence". *Annual Review of Fluid Mechanics* 28.1 (1996), pp. 45–82.

[145] C. Meneveau and J. Katz. "Scale-invariance and turbulence models for large-eddy simulation". *Annual Review of Fluid Mechanics* 32.1 (2000), pp. 1–32.

[146] R. Verzicco, J. Mohd-Yusof, P. Orlandi, and D. Haworth. "Large eddy simulation in complex geometric configurations using boundary body forces". *AIAA journal* 38.3 (2000), pp. 427–433.

[147] U. Piomelli. “Large eddy simulations in 2030 and beyond”. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 372.2022 (2014), p. 20130320.

[148] W. Cabot and P. Moin. “Approximate wall boundary conditions in the large-eddy simulation of high Reynolds number flow”. *Flow, Turbulence and Combustion* 63.1-4 (2000), pp. 269–291.

[149] M. Wang and P. Moin. “Dynamic wall modeling for large-eddy simulation of complex turbulent flows”. *Physics of Fluids (1994-present)* 14.7 (2002), pp. 2043–2051.

[150] U. Piomelli and E. Balaras. “Wall-layer models for large-eddy simulations”. *Annual review of fluid mechanics* 34.1 (2002), pp. 349–374.

[151] J. A. Templeton, G. Medic, and G. Kalitzin. “An eddy-viscosity based near-wall treatment for coarse grid large-eddy simulation”. *Physics of Fluids (1994-present)* 17.10 (2005), p. 105101.

[152] D. Chung and D. I. Pullin. “Large-eddy simulation and wall modelling of turbulent channel flow”. *Journal of fluid mechanics* 631 (2009), pp. 281–309.