Fall 2012

A massively parallel adaptive sharp interface solver with application to mechanical heart valve simulations

John Arnold Mousel
University of Iowa

A MASSIVELY PARALLEL ADAPTIVE SHARP INTERFACE SOLVER WITH APPLICATION TO MECHANICAL HEART VALVE SIMULATIONS

by

John Arnold Mousel

An Abstract

Of a thesis submitted in partial fulfillment
of the requirements for the Doctor of
Philosophy degree in Mechanical Engineering
in the Graduate College of
The University of Iowa

December 2012

Thesis Supervisor: Professor H. S. Udaykumar

ABSTRACT

This thesis presents a framework for simulating the fluid dynamical behavior of complex moving boundary problems in a high-performance computing environment. The framework is implemented in the pELAFINT3D software package. Moving boundaries are evolved in a seamless fashion through the use of distributed narrow band level set methods and the effect of moving boundaries is incorporated into the flow solution by a novel Cartesian grid method. The proposed Cartesian grid approach builds on the concept of a ghost fluid method where boundary conditions are applied through least-squares polynomial extrapolations. The method is hybridized such that computational cells adjacent to moving boundaries change discretization schemes smoothly in time to avoid the introduction of strong oscillations in the pressure field. The hybridization is shown to have minimal effect on accuracy while significantly suppressing pressure oscillations.

The computational capability of the Cartesian grid approach is enhanced with a massively parallel adaptive meshing algorithm. Local mesh enrichment is effected through the use of octree refinement, and a scalable mesh pruning algorithm is used to reduce the memory footprint of the Cartesian grid for geometries which are not well bounded by a rectangular cuboid. The computational work is kept in a well-balanced state through the use of an adaptive repartitioning strategy. The numerical scheme is validated against many benchmark problems and the composite approach is demonstrated to work well on tens of thousands of computational cores. A simulation of the closure phase of a mechanical heart valve was carried out to demonstrate the ability of the pELAFINT3D package to compute high Reynolds number flows with complex moving boundaries and a wide disparity in length scales. Finally, a novel image-to-computation algorithm was implemented to demonstrate the flexibility the current method allows in designing new applications.

Abstract Approved: ______________________________
Thesis Supervisor

____
Title and Department

____
Date

A MASSIVELY PARALLEL ADAPTIVE SHARP INTERFACE SOLVER WITH APPLICATION TO MECHANICAL HEART VALVE SIMULATIONS

by

John Arnold Mousel

A thesis submitted in partial fulfillment
of the requirements for the Doctor of
Philosophy degree in Mechanical Engineering
in the Graduate College of
The University of Iowa

December 2012

Thesis Supervisor: Professor H. S. Udaykumar

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

John Arnold Mousel

has been approved by the Examining Committee
for the thesis requirement for the Doctor of Philosophy
degree in Mechanical Engineering at the December 2012 graduation.

Thesis Committee:

H. S. Udaykumar, Thesis Supervisor

K. B. Chandran

Jia Lu

Pablo Carrica

James Buchholz

Sarah Vigmostad

ACKNOWLEDGMENTS

I would like to thank all those that helped me on this long journey. First, I must thank my advisor, H. S. Udaykumar. Uday supported me in every way possible through what has been the most trying time of my life in many ways. His role in this stage of my life will never be forgotten.

Next, I must thank the members of my committee. I have learned a lot from all the members. Their input both guided and pushed me throughout this process. Having gotten to know most of my committee members over several years, I am confident that it would be a task of the utmost difficulty to find a group of scholars with a better blend of intelligence, work-ethic, and genuine care for the students under their leadership.

To my friends and family, I am deeply indebted to you for everything you have done for me throughout this journey. I must give special credit to my mom and dad who only cared that their son was doing what made him happy. To my current and former lab mates: Seth, Anil, Shiv, Mehrdad, Liza, Keshav, Nirmal, Oishek, Vishwa, Neal, Ehsan, and Scott. My experiences with you are those that I will remember for the rest of my life.

Finally, to Jacquelyn. You endured with me through the most stressful period of my life without fail. You will never know how much your support means to me. I can only attempt to repay your kindness, generosity, and understanding.

ABSTRACT

This thesis presents a framework for simulating the fluid dynamical behavior of complex moving boundary problems in a high-performance computing environment. The framework is implemented in the pELAFINT3D software package. Moving boundaries are evolved in a seamless fashion through the use of distributed narrow band level set methods and the effect of moving boundaries is incorporated into the flow solution by a novel Cartesian grid method. The proposed Cartesian grid approach builds on the concept of a ghost fluid method where boundary conditions are applied through least-squares polynomial extrapolations. The method is hybridized such that computational cells adjacent to moving boundaries change discretization schemes smoothly in time to avoid the introduction of strong oscillations in the pressure field. The hybridization is shown to have minimal effect on accuracy while significantly suppressing pressure oscillations.

The computational capability of the Cartesian grid approach is enhanced with a massively parallel adaptive meshing algorithm. Local mesh enrichment is effected through the use of octree refinement, and a scalable mesh pruning algorithm is used to reduce the memory footprint of the Cartesian grid for geometries which are not well bounded by a rectangular cuboid. The computational work is kept in a well-balanced state through the use of an adaptive repartitioning strategy. The numerical scheme is validated against many benchmark problems and the composite approach is demonstrated to work well on tens of thousands of computational cores. A simulation of the closure phase of a mechanical heart valve was carried out to demonstrate the ability of the pELAFINT3D package to compute high Reynolds number flows with complex moving boundaries and a wide disparity in length scales. Finally, a novel image-to-computation algorithm was implemented to demonstrate the flexibility the current method allows in designing new applications.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

A common treatment for compromised heart valves is surgical replacement with some form of prosthetic valve, with approximately 100,000 replacement procedures performed in the United States annually [1]. Valve prostheses are generalized into two broad categories, mechanical heart valves (MHV) and bio-prosthetic heart valves (BHV). The most commonly used mechanical valves, accounting for 80% of mechanical prosthesis replacements, are based on the bileaflet design introduced by St. Jude Medical, Inc. in 1978 [2]. The St. Jude Medical valve design, which is shown in Figure 1, has two pyrolytic carbon leaflets which rotate within a hinge recess in a surrounding valve housing. In the fully open position, the leaflets are nearly parallel with the flow direction in an attempt to minimize flow disturbance. In the fully closed position (Figure 2), there exist small gaps between the adjacent leaflet edges as well as between the leaflets and the housing which are designed to clear out critical regions and prevent blood element buildup to reduce complications from the device and minimize the amount of coagulation therapy the patient must receive. These gaps are two orders of magnitude smaller than the housing diameter. In Figure 3(a), the butterfly shape of the housing recesses about which the leaflets rotate is shown. Figure 3(b) shows the manner in which the leaflets are inserted in butterfly shaped recesses in the housing is shown, while Figure 3(c) identifies a third gap between the outer edge of the leaflet and the housing recess.

Even after half a century of use, mechanical heart valves still require further research to reduce the non-physiologic nature of the flow field, which is the source of potential medical complications [3], of which the most serious complications include thrombus formation due to blood element trauma and subsequent device failure [2]. In the systolic phase of the flow, excessive fluid stresses are generated by the non-physiologic flow patterns. The increased stress as well as contact of blood elements with the device surface may lead to blood element damage [4, 5]. In the closed valve position, a large pressure gradient is imposed across the device which leads to the generation of strong and damaging small-scale leakage flows that entrain platelets such that they are exposed to elevated stresses for a extended durations [6-8]. Simon et al. [8] performed numerical simulations of leakage flow in the hinge region of a St. Jude mechanical valve and found that platelet activation and thrombus formation was likely to occur near the hinge region during systole, while thrombosis is more likely to occur during systole.

While thrombus formation can be managed with anti-coagulant therapy, this approach has its own set of complications, including increased risk of hemorrhage and infection [2]. Additionally, the high shear stresses induced by the device may lead to the stripping of endothelial cells from vessel walls, which further promotes platelet activation. Other complications of mechanical heart valves include leakages, infections, and tissue overgrowth [2].

There is still a lack of consensus in the literature about which flow scales are most damaging to cells. In [9] and [10] it is argued that it is the energetic middle-range flow structures that are most damaging, while [11, 12] propose that it is in fact the smallest structures that are responsible for blood element damage. Furthermore, it is still unclear what the smallest scale is that exists in the MHV flow field. During the systolic phase of the cardiac cycle, the forward flow has a peak Reynolds number of approximately 6,000 and a mean Reynolds number of nearly 4000, which results in an estimate of 0.04 and 0.06 mm as the size of the smallest flow structures based on Kolmogorov theory [13]. While the assumptions of Kolmogorov theory are not applicable due to the short transient nature of the MHV flow field, Bellofiore and Quinlan [14] used PIV to establish that there exists structures blood damaging smaller than 0.120 mm in the forward flow phase. During diastole, the jets occurring due to the leakage flow are of the scale of the gap dimensions (O(0.1 mm)).

While a simulation resolved at the inertial length scale will obtain correct leaflet dynamics [13], the damage of blood elements by flow structures cannot be examined at such a coarse scale. To probe the required range of scales computationally requires a dense grid in order to resolve the smallest vortical structures. Considering a tube one valve diameter across and one valve diameter long, a grid spacing of the Kolmogorov length scale would require 200 million grid points. Therefore, such a simulation will require a combination of high performance parallel computing, as well as adaptive meshing to make accurately solving the problem tractable.

A comprehensive numerical investigation of the process of thrombus formation in valve prostheses would require a combination of many computational techniques to account for all the physical mechanisms at play. First, a flow solver that can handle moving boundaries and moderate Reynolds number flows is required. In order to examine the response of blood elements to fluid-induced stress, an accurate computational model that can handle large numbers of deformable particles is needed.

Computing the deformation of the vascular structures in the vicinity of the prosthesis would require a fluid-structure interaction (FSI) algorithm which can handle flexible solids. Modeling the effects of the chemical factors in the blood which play an important role in the activation and aggregation of platelets introduces another level of complexity. Additionally, if patient-specific simulations are desired, a technique to convert medical images to simulation geometries is required. In this work, we focus on constructing a flow solver which promotes a unified algorithm to perform direct image to computation simulations.

A considerable amount of work has been conducted with regards to heart valve modeling. The simulation of a single phase Newtonian fluid in MHV devices has been carried out by many researchers with varying levels of approximation. Currently, the most sophisticated three-dimensional calculations of the MHV problem have computed the forward flow at the physiologic Reynolds numbers, but no simulation has completely resolved the flow field and the prostheses gaps have never been completely resolved in a simulation of the full system [13, 15-17]. In order to attempt to elucidate the physics in the gap region, two-dimensional and three-dimensional calculations of the leakage flow have been carried out with simplifying assumptions. In [6, 7, 18, 19], the closing phase of valve motion and the associated leakage jet through the gap around the leaflet edge was assumed to start from a quiescent condition. However, starting from a quiescent condition neglects the presence of the rich vortical structures generated during the opening phase and subsequently swept to the ventricular side of the valve during closure, which may limit the physical insight into the cell damage process that can be obtained from such simulations. In [8], the leakage flow in the hinge region was modeled by performing a simulation on one quarter of the geometry in the vicinity of the hinge with boundary conditions obtained from a separate simulation

Much work has also been carried out on obtaining realistic cardiovascular geometries from medical images and then solving for the flow field on an unstructured mesh [20, 21] or structured curvilinear grids [22]. While this approach was successful in these applications, extending the methodology to boundaries with large scale boundary motion may be difficult to manage due to the complexity of generating high quality unstructured meshes [23]. In [24], patient specific cardiovascular geometries were obtained from medical images and the flow fields were subsequently computed using an unstructured Cartesian approach with a reduced memory footprint. However, the manner in which the geometries were obtained was independent of the nature of the Cartesian method.

Due to the complexity of unstructured mesh generation, as well as the high memory requirements and reduced accuracy of unstructured grid methods [23, 25], and the intimate connection with image analysis techniques, we propose that an enhanced Cartesian approach is a natural route to assemble the computational components required to examine the process of thrombus formation in mechanical heart valves. An efficient moving boundary flow solver can be based on a Cartesian grid interface treatment approach [26] combined with level set representation of geometries [27]. There exist mesh refinement algorithms for Cartesian grids which scale on the largest supercomputers currently available and which can be used to increase the efficiency of the method [28, 29]. This thesis develops the beginning of a massively parallel framework for solving complex moving boundary problems on Cartesian grids, enabling the solution of the MHV problem along with a variety of other problems in a relatively simple, unified fashion.

1.2 Contributions of the Current Thesis

In the current thesis, a framework, pELAFINT3D, is established as a starting point for assembling the multi-physics components required in examining the process of thrombus formation in mechanical heart valves. The motivating goal of pELAFINT3D is to allow the simulation of computationally challenging moving boundary problems with no requirements of the user to interact with mesh construction or maintenance. The user is only required to provide a description of the initial state of any complex boundaries. The boundary configuration may be supplied by a mesh description of the surface, but this mesh must only smoothly represent the boundary. There is no restriction on mesh quality issues such as element size or skewness. This greatly reduces the burden a user must endure to setup a challenging simulation.

The heart of pELAFINT3D is a completely distributed Cartesian grid which is enhanced at run time to ensure solution quality. Geometries are represented using a distributed level set method which allows for unified treatment of complex boundary motions. The flow solver is constructed using a novel ghost fluid Cartesian grid method. Specifically, the features of pELAFINT3D are:

a) A massively parallel distributed Cartesian grid with octree refinement and mesh pruning capabilities. These capabilities enhance the applicability of the package to problems with small localized features as well as tortuous geometries.

b) A hybridized least-squares ghost fluid Cartesian grid method for accurately simulating the flow dynamics in an incompressible fluid with moving boundaries.

The least-squares construction serves two purposes. First, it is a starting point for a higher-order interface treatment. Second, it enhances the robustness of the interface treatment in the presence of strong boundary curvature and contacting surfaces. The hybridization alleviates the strong numerical pressure oscillations that plague sharp-interface Cartesian grid methods.

c) An efficient distributed narrow band level set method.

d) A direct image-to-computation algorithm founded in the methods of image segmentation. The method is applied to examine the flow in an intracranial aneurysm.

The remainder of the thesis is a presentation of the components of pELAFINT3D and its application. In Chapter 2, the novel Cartesian grid method is presented and examined in detail. Emphasis is placed on the properties of the scheme for moving boundary problems, and in particular, the ability of the scheme to suppress the strong numerical oscillations in pressure common to most Cartesian grid methods is demonstrated. In Chapter 3, the parallel mesh structure and algorithms are presented. The framework is demonstrated to operate efficiently on tens of thousands of cores on Kraken, a state-of-the-art supercomputer. In Chapter 4, the package is validated against several standard benchmark problems. Chapter 5 presents an application of the pELAFINT3D package to compute the flow through a three-dimensional mechanical heart valve undergoing closure. In Chapter 6, a novel image-to-computation algorithm is presented which is a natural extension of the current methodology. The algorithm is then applied to simulate the flow through an intracranial aneurysm which is extracted from CT medical images. Chapter 7 provides a summary of the work accomplished in this thesis, and provides several avenues to extend the current research.

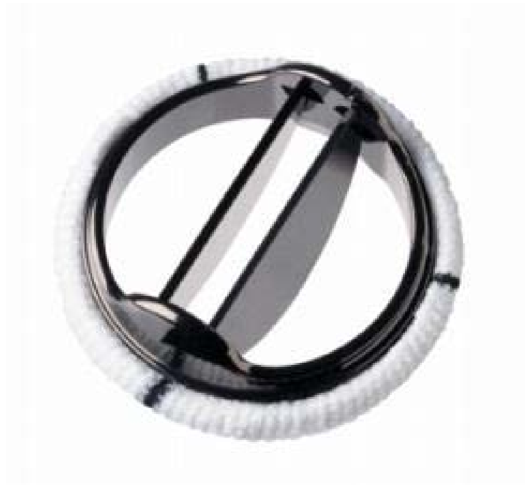


Figure 1 An example of a bileaflet mechanical heart valve.

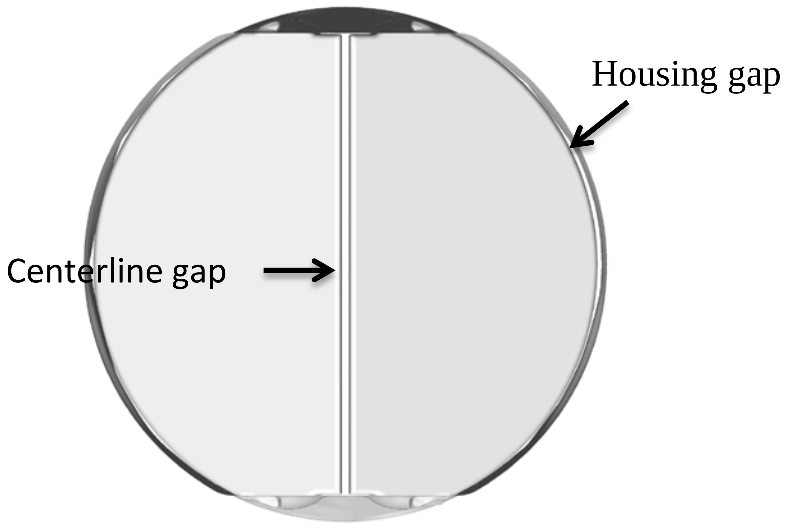


Figure 2 An axial view of the valve in the fully closed position.

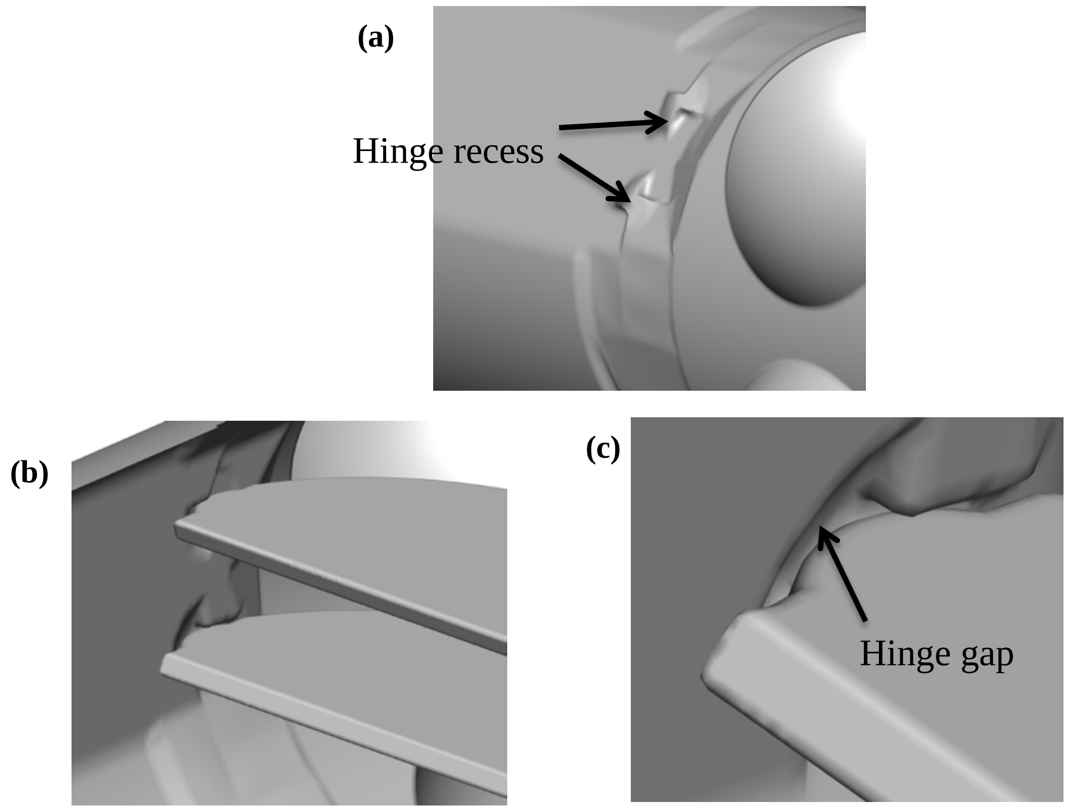


Figure 3 A set of figures showing the assembly of the valve leaflets in the housing. The butterfly-shaped hinge recesses are show in (a). In (b), the tabs at the ends of the leaflets are inserted into the hinge recesses. In (c), a zoomed-in view shows the gap between the hinge recess wall and the leaflet tab.

CHAPTER 2

THE CARTESIAN GRID METHOD

2.1 Introduction

The current chapter describes the novel Cartesian grid approach developed to facilitate the solution of complex moving boundary problems which contain strong curvatures and nearly contacting interfaces. The governing equations are solved on a Cartesian mesh which is not required to align with the geometry of the problem. Level set representations of embedded boundaries are used to construct a sharp interface treatment which includes the effects of boundary conditions directly in the discrete operators.

To this end, boundary conditions are utilized to construct an extrapolation of the flow variables to a set of ghost grid points inside solid objects such that all computational stencils for grid nodes in the fluid are complete. A robust high-order extrapolation is created using a least-squares technique. To accurately incorporate the shape of the embedded boundary, the boundary conditions at multiple points on the interface are used to construct the extrapolation. The required locations of the interface where boundary conditions are required are easily computed with high accuracy using the level set representation.

Following an approach similar to Luo et al. [30], a hybridization of the method is developed such that grid points smoothly transition between solving the governing equations and being populated by extrapolation techniques as the surface evolves. This approach stabilizes the temporal character of the pressure field by smoothing the strong variation in the truncation error as a grid point transitions from one type to another.

The current method is accurate, efficient, and simple to implement. The use of the least-squares procedure with the level set representation allows the current method to be extended to higher orders of accuracy. The current scheme is shown to obtain locally 2nd-order results with and without hybridization for Poisson problems with both Dirichlet and Neumann conditions. Additionally, the method handles difficult interface conditions, such as strongly concave curvatures and contacting surfaces, naturally through the least-squares procedure without the need to consider special case logic. The hybridization technique is shown to significantly reduce the magnitude of oscillations in the pressure field associated with moving boundary problems.

2.2 Review of Cartesian Grid Methods

In order to handle moving boundary problems, traditional body-fitted grid methods require re-meshing as the object moves. Re-meshing is typically an expensive operation, and obtaining high quality meshes is a difficult task, so avoiding this operation is desirable. Cartesian grid methods dispense with the need to continuously regenerate the computational grid by solving the governing equations on a fixed Cartesian mesh for which the embedded boundary is allowed to cut through without any restrictions on topology [26]. The challenge then becomes one of how to apply boundary conditions specified on embedded surfaces.

Cartesian grid methods can be separated into two classes which are determined by whether the boundary conditions are imposed by modifying the continuous governing equations or modifying the numerical discretization scheme [26]. The first class of methods, known as continuous forcing methods, introduce the effect of the boundary through the addition of a forcing function to the momentum equations, which now takes the form:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\nabla p + \frac{1}{Re} \nabla^2 \vec{u} + \vec{f}_b \ . \qquad (2.1)$$

The governing equations are then solved for one continuous field that covers both the interior and exterior of the object. Any particular continuous forcing method is defined by the form of the forcing term, $\vec{f}_b$. The most popular continuous forcing method has been the Immersed Boundary Method (IBM) [31, 32] which was developed to facilitate the solution of the flow of blood in the human heart [33]. In this approach, singular surface forces defined on a Lagrangian mesh are converted to volumetric forces defined on Cartesian flow mesh using numerical delta functions. This approach has found extensive use in the area of bio-fluid dynamics [31, 33, 34], especially in the study of suspensions of flowing cells [35-37]. However, IBM has several drawbacks. First, the numerical delta function spreads the force over a finite support which smears the effect of the interface, thereby degrading accuracy [38]. Additionally, the above approach requires special treatment to handle rigid boundaries [39]. Improvements have been made to the IBM in recent years [39, 40], and the idea of transmitting the effect of boundaries to the fluid through the use of source has been extended to the finite element method by the Fictitious Domain Method [41] and Immersed Finite-Element Method [42].

Recently, many methods have moved away from using delta functions to distribute the force between grids. Instead, a forcing is derived in terms of the numerical discretization scheme such that velocity boundary conditions are discretely satisfied after the imposition of the forcing [43-45]. The forcing is applied at a set of points adjacent to the interface, where the points can be inside the interface, outside the interface, or both.

This type of approach does not suffer from the accuracy or time-step restrictions of the original IBM. Because the forcing is derived from the numerical approximation to the governing equations, this type of method is not strictly a member of the continuous forcing method, and is termed a discrete forcing approach. It is instead similar in spirit to the methods which will be discussed next in that the treatment of boundary conditions is intimately connected with the numerical scheme. However, in contrast to the methods to be discussed next, discrete forcing methods do not directly modify numerical discretization operators, but instead determine a numerically consistent forcing term to be supplied to the continuous governing equations.

In a second class of methods, boundary conditions are imposed directly by modification of the discretization of operators for grid points adjacent to the boundary. The governing equations are solved in the form

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\nabla p + \frac{1}{Re} \nabla^2 \vec{u} \tag{2.2}$$

The spatially discrete operators are constructed in a way such that boundary conditions are directly incorporated into non-uniform discretization stencils. Examples of this type of approach are the Immersed Interface Method (IIM) [38, 46, 47], the Ghost Fluid Method (GFM) [25, 48-50], the Sharp Interface Method (SIM) [51-53], and the Extended Finite-Element Method (XFEM) [54-56]. The Immersed Interface Method converts to normal and tangential coordinates and then solves a set of linear equations for coefficients that incorporate the imposed jump conditions into the discretization matrix directly for all cells adjacent to the interface. The Ghost Fluid Method uses a boundary-condition satisfying extrapolation scheme to assign values to the interface cells interior to the object, so that standard discretization stencils can be used at all cells in the fluid phase. The Sharp Interface Method solves a control-volume formulation of the governing equations, such that control volumes that are intersected by the interface are re-shaped to conform to the boundary. XFEM subdivides elements that are cut by the interface such that local degrees of freedom always align with the interface.

The approaches discussed above all have relative advantages and disadvantages. The Sharp Interface Method and XFEM are the most accurate. The Sharp Interface method conserves mass and momentum, but formulating the scheme in three dimensions is challenging due to the number of special cases that must be considered when re-shaping control volumes. Additionally, small control volumes produced by the re-shaping process must be handled through special techniques such as cell merging or cell linking to avoid introducing a stringent time-step restriction due to the CFL condition [57-59]. The Immersed Interface Method is accurate, but it is not directly applicable to cases where the object is a rigid solid. Ghost fluid methods use extrapolation stencils that require no matrix inversions to formulate, are simple to implement, and can handle rigid objects. However, ghost fluid methods are not conservative and are also generally slightly less accurate than the other approaches. In this work, we adopt the ghost fluid approach due to its simplicity.

The concept of a ghost fluid, first popularized by Fedkiw et al. [48], was originally used to simulate shocks and discontinuities. The approach was extended to the solution of Poisson and heat equations by Gibou et al. in [49] and later a fourth-order GFM scheme based on dimension-by-dimension Lagrange interpolation was proposed in [60]. The concept of a ghost fluid was extended to incompressible flows by Tseng and

Ferziger [25]. Mittal et al. [50] put forth a slight variant of the scheme proposed by Tseng and Ferziger.

It is well known that many Cartesian grid methods produce strong numerical oscillations in the pressure field in the case of moving boundaries due to a lack of geometric volume conservation and a temporal discontinuity in the velocity field as a grid point changes from fluid to solid phase and vice versa [30, 61, 62]. While the pressure oscillation does not significantly affect the velocity field, it can be problematic if the motion of a solid object is determined by the force induced by the pressure field. Kim et al. [44] significantly reduced the pressure oscillation associated with their discrete forcing approach by utilizing mass sources/sinks to satisfy mass conservation at grid cells that contained the immersed interface. Seo and Mittal [62] used a hybrid ghost cell method-cut cell technique where only the source term of the Poisson equation was discretized using a cut-cell technique. This approach ensures that the geometric volume of the embedded boundary is discretely conserved. Luo et al. [30] introduced a scheme where the flow variables are computed as weighted averages of the solutions to the governing equations and values extrapolated from neighboring grid nodes.

2.3 Level Set Fields

Objects embedded in the computational mesh are represented implicitly by level set fields. A level set field, $\phi(\vec{x}, t)$, is an n+1 dimensional scalar field defined as a signed distance from a grid point to the nearest n-dimensional surface [27, 63]. The surface itself is embedded as the zero iso-contour of the scalar field. In this work, the level set field is more specifically taken to be a signed normal distance function, which gives the normal distance from the location to the closest point on the embedded surface. It is defined to be positive if is exterior to the embedded object and negative otherwise.

The level set field is governed by a hyperbolic partial differential equation subject to a given propagation velocity [63]. The level set advection equation is

$$\frac{\partial \phi}{\partial t} + \vec{V} \cdot \nabla \phi = 0. \tag{2.3}$$

Here, $\vec{V}$ is the propagation velocity of the level set field which is required in the entire domain for which the level set is defined. Typically, the physics of the problem only defines the propagation velocity of the zero level set contours. In this case, the interface velocity is extended along the normal direction to define propagation velocities everywhere else.

The level set advection equation (2.3) is solved numerically using a 3rd-order TVD Runge-Kutta time-stepping scheme in conjunction with 5th-order WENO upwind spatial discretizations [64, 65]. In order to save computational effort and memory, the level set is only calculated and stored in a narrow band of grid points around the interface [66], or more specifically a dynamic tubular grid [67]. As the level set moves, the narrow band requires rebuilding so the zero level set contour does not pass out of the domain over which it is computed. During this rebuilding step, points that are newly introduced to the band are initialized with the level set re-initialization equation [68]:

$$\frac{\partial \phi}{\partial \tau} + \mathrm{sign}(\phi)\vec{n} \cdot \nabla \phi = 0 \tag{2.4}$$

Here, τ is a pseudo-time, and the solution of equation (2.4) is marched to a steady state. After convergence, the level set field satisfies the normal distance property, $|\nabla \phi| = 1$, and the points newly introduced into the narrow band have a valid value. The implementation details of the narrow band structure substantially increase the difficulty of parallelizing the level set scheme. The current parallel narrow band construction algorithm is discussed in Chapter 3.

2.4 The Current Cartesian Grid Method

2.4.1 Governing Equations

The incompressible Navier-Stokes equations are solved in the current work. In non-dimensional form, this set of equations is:

$$\nabla \cdot \vec{u} = 0 \tag{2.5}$$

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\nabla p + \frac{1}{Re} \nabla^2 \vec{u} \ . \tag{2.6}$$

Here $\vec{u}$ is the non-dimensional velocity vector, p is non-dimensional pressure and $Re = \frac{UL}{\nu}$ is the Reynolds number based on a characteristic velocity, U, a characteristic length, L, and the fluid kinematic viscosity, ν, which is assumed to be constant. The governing equations must be supplemented by a set of Dirichlet boundary conditions for the velocity field on all domain and immersed boundaries and a divergence free initial condition, $\vec{u}_o$.

2.4.2 Classification of Grid Points

The current ghost fluid method is based on a disjoint partitioning of the computational domain into four types of grid nodes as shown in Figure 4, where a node is classified according to the technique for which its flow variables are updated. Away from the interface in the fluid phase, there is no difficulty applying standard finite difference techniques to solve the Navier-Stokes equations. The grid nodes for which standard treatment can be applied will be called fluid nodes. Immediately inside the surface, a set of nodes called ghost nodes are populated with flow variable values using an extrapolation technique such that a prescribed boundary condition is satisfied on the surface. These nodes have a level set value such that at least one of the neighbors in the cardinal directions has a positive level set value. By assigning a flow variable value to this set, central difference stencils can be formed for all cells on the fluid side of the interface.

The partitioning of the computational nodes into ghost and fluid nodes is sufficient to construct a ghost fluid method for stationary surfaces. In the case of moving boundary problems, the changing of grid nodes from ghost nodes to fluid nodes has been shown to contribute to a strong temporal oscillation in the pressure field. Seo and Mittal [62] show that the pressure oscillation can be attributed to the introduction of a spurious geometric volume during grid node transition. Their analysis results in an estimate of the volume conservation error as $\frac{\Delta V}{\Delta t}|1 - \mathrm{CFL}_s|$. This shows that in two-dimensions the error is proportional to $\Delta x^2/\Delta t$. In order to alleviate this issue, the authors implemented a hybridized scheme where the momentum equations are solved using a ghost fluid method while the Poisson equation is solved using a mass conserving cut-cell approach, which results in a 6-fold reduction in the magnitude of pressure oscillations.

Taking a different approach, Luo et al. [30] attributed the source of the pressure oscillation to the change in character of the discretization error as a grid node transition between partition sets. While both the extrapolation method and the solution of the governing equations may have $O(\Delta x^2)$ truncation error, the inconsistency in the form of the errors leads to a temporal error which is also $O(\Delta x^2)$. When this error is introduced in the source term of the Poisson equation, it becomes an $O(\Delta x^2/\Delta t)$ error, which exhibits the same dependence on the grid spacing and time step sizes as the analysis of [62].

The analysis of [30] demonstrates that two steps can be taken to control pressure oscillations. First, the CFL number a simulation is performed at can be made as close to unity as possible while refining the grid to make $\Delta x^2/\Delta t$ as small as possible. Second, one can attempt to minimize the temporal discrepancy in the spatial error by smoothly transitioning between numerical treatments. This can be effected by a hybridized treatment where a third subset of grid nodes is introduced. In this set the flow variables are computed according to a weighted blending of the solution of the governing equations and a solution by an interpolation from surrounding nodes. In this work, a point is specified to have hybridized treatment if it has a positive level set value less than $\sqrt{2}\Delta x$. If a hybridized treatment is not desired, the blending coefficient is simply set to be purely biased towards the solution of the governing equations.

During the transition from a ghost cell to a hybrid cell, the freshly created hybrid cell does not have a valid flow history. Additionally, the cell has newly formed ghost cell neighbors for which there is also no history. Fresh cells are characterized in terms of the level set field by a change from negative to positive level set value between time steps. These cells are given a pure interpolation from the surrounding cells during the solution process. This obviates the need for a time-history of flow variables. After one time step, a fresh cell is transitioned to a hybrid cell.

This closes the definition of the types of computational points required for the construction of the current ghost fluid method. Table 1 lists the types of computational nodes and their characterization in terms of the level set field. The partitioning of the computational domain into the current classification scheme is very natural using the current level set formulation. Furthermore, the level set field will be shown to lend itself to the simplicity of the construction algorithm for the interface discretization operators.

2.4.3 Fractional Step Algorithm

In the current framework, the governing equations are solved at both fluid and hybrid nodes. The flow variables are evolved on a hybrid collocated cell-centered grid, where the velocity and pressure are stored at cell centers and a secondary set of mass-conserving velocities are stored at cell faces. A four-step fractional step method [69, 70] is implemented to decouple the velocity and pressure variables. In the first step, a provisional intermediate velocity is computed for which the incompressibility constraint is not explicitly enforced. A 2nd-order explicit Adams-Bashforth scheme is used for the time-discretization of the non-linear term, while the 2nd-order semi-implicit Crank-Nicholson method is applied to the viscous term. The resulting semi-discrete form of the equation for the intermediate velocity, $\vec{u}^*$, is

$$\frac{\vec{u}^* - \vec{u}^n}{\Delta t} = -\frac{3}{2} H(\vec{u}^n) + \frac{1}{2} H(\vec{u}^{n-1}) + \frac{1}{2Re}\left(L(\vec{u}^*) + L(\vec{u}^n)\right) - G(q) \,. \tag{2.7}$$

Here, H is a discrete spatial approximation of the non-linear term, L is a spatial approximation to the Laplacian, and $G(q)$ is some suitable estimate of the pressure gradient.

The second step consists of removing the estimate of the pressure gradient used in the first step:

$$\vec{u}^{**} = \vec{u}^* + G(q)\Delta t. \tag{2.8}$$

This has the effect of stabilizing the calculation by addressing the pressure-velocity decoupling issues encountered with the use of non-staggered grids.

The effect of incompressibility is enforced by decomposing $\vec{u}^{**}$ into the sum of divergence-free and irrotational vector fields,

$$\vec{u}^{n+1} = \vec{u}^{**} - \Delta t \nabla \varphi. \qquad (2.9)$$

A Poisson equation for the scalar potential φ is derived by taking the divergence of Equation (2.9) and requiring that $\vec{u}^{n+1}$ be divergence-free, giving

$$\nabla^2 \varphi = \frac{\nabla \cdot \vec{u}^{**}}{\Delta t} \qquad (2.10)$$

The remaining issues to be addressed are boundary conditions for each of the steps in the algorithm, how to define q, and how to relate φ to pressure. It should be emphasized that φ is not pressure, but is instead a purely mathematical variable that can be related to pressure only when the discretization scheme has been chosen [71]. From Equation (2.10), it is clear that the boundary condition for equation (2.10) must be:

$$\frac{\partial \varphi}{\partial n}\Big|_{\partial\Omega} = \frac{(\vec{u}^{**} - \vec{u}^{n+1}) \cdot \vec{n}}{\Delta t}\Big|_{\partial\Omega} \qquad (2.11)$$

If one sets the boundary condition for the intermediate velocity equal to that of the final velocity, a homogeneous Neumann condition is obtained. To maintain second order time accuracy of the velocity field, one can select to set q to the previous time step pressure and φ can then be considered an O(Δt) approximation to pressure.

2.4.4 Discretization of Operators

In the current method the solution of the Navier-Stokes equations is required at both fluid nodes and hybrid nodes. Operators required for the solution of the Navier-Stokes equations are discretized using standard centered finite differences. There is no difficulty in utilizing centered differences for operator construction at hybrid nodes because the fluid variables are defined at the neighboring interface ghost nodes which ensure that the stencil is always properly posed. Accordingly, the Laplacian operator has the standard 5-point stencil. For a fluid node P with neighboring grid nodes E, W, N, and S in the east, west, north, and south directions, and grid spacing $(\Delta x_P, \Delta y_P)$, the Laplacian operator at P is approximated as

$$\nabla^2\psi|_P = \frac{\psi_E - 2\psi_P + \psi_W}{\Delta {x_P}^2} + \frac{\psi_N - 2\psi_P + \psi_S}{\Delta {y_P}^2}. \qquad (2.12)$$

The convection term has the form

$$\nabla \cdot (\vec{u}\psi)|_P = \frac{u_e\psi_e - u_w\psi_w}{\Delta x_P} + \frac{v_n\psi_n - u_s\psi_s}{\Delta y_P}. \qquad (2.13)$$

In Equation (2.13), ψ_i is the average of ψ to the ith face, and ui and vi are mass conserving velocities which are stored at the faces.

2.4.5 Ghost Node Treatment

The over-arching premise of the Ghost Fluid Method (GFM) is to extend the fluid variables across an interface into a set of ghost nodes where they were not originally defined such that the extension takes the desired boundary condition value at the interface [48]. Standard discretization stencils can then be used for every fluid grid node. In the present method, ghost nodes are used to complete discretization stencils required for hybrid nodes. The current section describes in detail the technique applied to populate flow variables in the ghost nodes.

To make the concept of ghost node population concrete, consider the one-dimensional case in Figure 5 where the interface is located between grid nodes i and i+1, where node i is in the fluid and i+1 is in the solid, such that i+1 is a ghost node in a centered finite difference of the one-dimensional Laplace operator at node i. An arbitrary function, $\widetilde{\psi}(x)$, such as a cubic polynomial can be fit to the values of a variable ψ at grid points i, i-1, and i-2, along with the boundary value at the interface. The extension of the field, ψ_{i+1}^{G}, would occur by assigning the node the value of the polynomial at x_{i+1},

$\widetilde{\psi}(x_{i+1})$. The extending polynomial may be re-written in terms of the field values at the points i, i-1, and i-2 using Lagrange polynomials. This facilitates the use of the extrapolating polynomials for implicit operators.

If the extrapolating function is selected to be a polynomial for the one-dimensional case, then the only parameter requiring specification is the degree of the polynomial fit. Once this has been established the neighboring points to be included in the extrapolation are completely determined by the location of the fluid phase and the problem is closed. In higher dimensions the situation is not as obvious as many extrapolation possibilities exist. Consider the two-dimensional case presented in Figure 6 where the standard five-point discretization of the Laplacian operator at grid node (i,j) is required, but the neighboring grid point (i-1,j) is inside a solid object. The stencil branch in the west direction is illegitimate unless a suitable GFM extrapolation is chosen to assign a value at node (i-1,j).

Two possible extrapolation choices are shown in Figure 7 and Figure 8. In Figure 7, the extrapolation is performed in a dimension-by-dimension manner. If a neighboring node in a cardinal direction is found to be inside the solid, a one-dimensional extrapolating polynomial is formed using neighboring nodes in the opposite direction as well as the boundary value. The one-dimensional approach is extremely simple to implement even for high orders of accuracy. Gibou et. al [49, 60] have developed numerical schemes based on direction-by-direction extrapolations with up to fourth-order spatial truncation error for the heat and Poisson equations with Dirichlet conditions, and Marella et al. [72] have proposed a globally second-order solver for the solution of the incompressible Navier-Stokes equations. However, stability issues often accompany attempts to use polynomials smoother than quadratic.

The use of a one-dimensional extrapolation leads to the introduction of a non-uniqueness into the definition of the ghost value at nodes similar to (i-1,j) in Figure 6. The stencils of both (i,j) and (i-1,j+1) use one-dimensional extrapolations to fill in necessary ghost information at (i,j), and it is not generally possible for the two extrapolations to be consistent. Furthermore, due to the lack of strong coupling between grid nodes, even explicit source terms must be constructed using extrapolations, which can require complicated code if the interface is moving during the simulation.

Presently, an alternative to the dimension-by-dimension approach is pursued whereby a single unique extrapolation stencil is assigned to each interfacial ghost node. A popular choice for a unique extension is to construct an extrapolating polynomial which is fit to the value of the closest point on the interface to the interface ghost node (the interfacial intersection, ψ_{int} in Figure 8), and the values at a set of probe points located along the ray emanating from the boundary point along the normal direction to the interface into the fluid (e.g. ψ_{P1} and ψ_{P2}) [25, 50]. Because the probe points are located at arbitrary locations, their values must then be interpolated from surrounding nodes. Tseng and Ferziger [25] and Mittal et al. [50] implement this type of approach by placing a single probe point at a position located symmetrically about the interface along the line connecting the ghost node and the closest interface point (Figure 8). This is the approach adopted in the current work.

Once the scheme for assigning probe values is defined, it is simple to apply various types of boundary conditions. In the case of Dirichlet boundary conditions, the value of the ghost point is found in terms of the probe value, ψ_P, and boundary condition, ψ_{int}, according to:

$$\psi_G = 2\psi_{int} - \psi_P, \tag{2.14}$$

Neumann conditions are applied using a second order central difference about the boundary intersection point:

$$\psi_G = \psi_P - \Delta l \frac{\partial\psi}{\partial n}|_{int}, \tag{2.15}$$

where $\frac{\partial\psi}{\partial n}|_{int}$ is the value of the normal derivative of ψ at the interfacial intersection, and Δl is the distance between the probe and ghost points.

The remaining part of the numerical scheme requiring specification is how to assign values to probe points. Tseng and Ferziger [25] utilize the two closest grid points to the ghost node that are in the fluid as well the value at the boundary intersection to establish an interpolating polynomial of the form

$$\tilde{\psi}(x, y) = a + bx + cy. \tag{2.16}$$

The probe value is then taken to be the value of the interpolating polynomial at the probe location, (x_P,y_P). Mittal et al. [50] instead find the four closest nodes to the probe location, regardless of whether they are in the solid or not. By construction, the nodes found under such a criteria must be either fluid nodes or other ghost nodes. A polynomial of the form

$$\tilde{\psi}(x, y) = a + bx + cy + dxy \tag{2.17}$$

is then constructed and $\psi_P = \tilde{\psi}(x_P, y_P)$.

In the current work, required probe values are interpolated using a weighted least-squares approach, which allows for seamless incorporation of both Dirichlet as well as Neumann conditions into a functional fit at the probe point. A multi-dimensional polynomial of M basis functions is constructed from a set of N cloud points surrounding the probe point. For clarity, the following discussion will focus on a two-dimensional quadratic polynomial in the relative coordinates,

$$\tilde{\psi}(x, y) = a + b\hat{x} + c\hat{y} + d\hat{x}\hat{y} + e\hat{x}^2 + f\hat{y}^2, \tag{2.18}$$

$$\hat{x} = (x - x_P)/h \tag{2.19}$$

$$\hat{y} = (y - y_P)/h \tag{2.20}$$

$$h = 0.7\sqrt{\sum_{i=1}^{N}(x^i - x_P)^2 + (y^i - y_P)^2} \tag{2.21}$$

Here, (x^i, y^i) are the coordinates of the ith cloud point, and h is a measure of the cloud width. The coordinate differences are scaled with h to ensure numerical stability in the situation where points are closely clustered about the probe point. In this case M = 6.

Applying a least-squares procedure leads to a matrix equation of the form:

$$A\vec{a} = \vec{\psi}. \tag{2.22}$$

Here, A is an NxM coefficient matrix dependent on the coordinates of the cloud points, $\vec{a}$ is the vector of polynomial coefficients of length M, and $\vec{\psi}$ is a vector of field variable or derivative values on the set of cloud points. Both sides of Equation (2.22) can be modified by a weighting function which ensures that the points closets to the probe point are given the largest contribution to determining the expansion coefficients. The weighting function, Wi, is taken to be a Gaussian function of coordinate differences:

$$W_i = e^{-\left[\left(x^i - x_P\right)^2 + \left(y^i - y_P)\right)^2\right]/h^2}. \tag{2.23}$$

To carry out a general discussion, consider a set of cloud points where the value of ψ is specified at all cloud points except one, where the value of the normal derivative, $\frac{\partial \psi}{\partial n}$, is specified instead. Let this cloud point be labeled as the first cloud point, with all other cloud points being labeled 2 through N. A set of N equations can be generated by applying the right-hand side of Equation (2.18) at the locations of the cloud points and setting the result equal to the supplied cloud variable value. This procedure gives rise to the matrix system

$$\begin{bmatrix} 0 & W_1 n_x & W_1 n_y & W_1(n_x\hat{y}_1 + n_y\hat{x}_1) & 2W_1\hat{x}_1 & 2W_1\hat{y}_1 \\ W_2 & W_2\hat{x}_2 & W_2\hat{y}_2 & W_2\hat{x}_2\hat{y}_2 & W_2\hat{x}_2^2 & W_2\hat{y}_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W_N & W_N\hat{x}_N & W_N\hat{y}_N & W_N\hat{x}_N\hat{y}_N & W_N\hat{y}_N^2 & W_N\hat{y}_N^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} =$$

$$\begin{bmatrix} W_1\, \partial\psi/\partial n_1 \\ W_2\psi_2 \\ \vdots \\ W_N\psi_N \end{bmatrix}, \tag{2.24}$$

or more briefly,

$$A_{\text{cloud}}\vec{a} = \vec{\psi}. \tag{2.25}$$

In general, this system is rectangular as the number of cloud points can be larger than the number of expansion coefficients that need to be determined to specify the polynomial fit. In the case of a rectangular matrix, one must solve the resultant linear system by computing the pseudo-inverse of A, A^+. For example, any mxn matrix, A, has a singular value decomposition (SVD),

$$A = U\Sigma V^T, \tag{2.26}$$

where U and V are mxm and nxn orthogonal matrices respectively, and Σ is an mxn diagonal matrix with m non-negative diagonal values. Due to the structure of this decomposition, the pseudo-inverse, A^+, can be found quite easily:

$$A^+ = V\Sigma^+U^T. \tag{2.27}$$

Being diagonal, the pseudo-inverse of Σ is computed by taking the reciprocal of the diagonal entries. If any diagonal term is zero, the reciprocal of the term is set to zero. Alternatively, a QR factorization can be used, where A is decomposed as

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}. \tag{2.28}$$

Here, Q is an mxm orthogonal matrix constructed from Householder reflectors and R is an nxn upper triangular matrix R. Exploiting the structure of this decomposition results in a pseudo-inverse of the form

$$A^{+} = R^{-1}Q^{T}. \tag{2.29}$$

As the inverse of an upper triangular matrix is upper triangular, specialized algorithms for the product of the matrices on the right-hand side of Equation (2.29) can be used to increase efficiency. Finally, a third approach to invert Equation (2.25) is based on the normal equation form of the rectangular matrix problem. Here, both sides of the equation are pre-multiplied by A_{cloud}^{T}, which results in a square system. In this case,

$$A^{+} = (A^{T}A)^{-1}A^{T}. \tag{2.30}$$

Because the pseudo-inverse is identical to the inverse for a square matrix, we can generalize the solution for the polynomial expansion coefficients in terms of the pseudo-inverse. The expansion coefficients can be determined as

$$\vec{a} = A_{\text{cloud}}^{+}\vec{\psi}. \tag{2.31}$$

For the current scheme, only the value of ψ at the probe point is required. This corresponds to the first element of $\vec{a}$ in Equation (2.31). Therefore, the value of ψ at the probe point is computed as

$$\psi_P = \sum_{i=1}^{N} A_{1i}^{+}\psi_i. \tag{2.32}$$

The set of coefficients $(A^+_{11}, A^+_{12}, \dots, A^+_{1N})$ are pre-computed and stored for each interface ghost node at the start of each time-step. The gradient of ψ can also be recovered from the same pseudo-inverse as

$$\nabla\psi_P = \sum_{i=1}^{N} A^+_{2i}\psi_i\,\vec{e}_x + \sum_{i=1}^{N} A^+_{3i}\psi_i\,\vec{e}_y + \sum_{i=1}^{N} A^+_{4i}\psi_i\,\vec{e}_z \tag{2.33}$$

What remains is how to specify the set of cloud points from which the least-squares fit is constructed. For a probe point associated with a ghost cell the procedure is as follows. First, the grid points within a specified radius of the probe location are determined using a fast search algorithm. If the potential cloud point is a ghost node it is replaced in the cloud by the projection of the ghost node onto the interface Figure 9, and the boundary value at the projected interfacial location is used in the right hand side of Equation (2.25). Finding the interfacial projection is easily accomplished by using the normal distance and normal vectors generated by the level set field.

In the case of thin structures this paring procedure may not be restrictive enough, for if the cloud radius is large it is possible that points from opposite sides of the object may be specified as a single cloud. To ensure that the cloud is composed of points from a consistent side of the interface, normal vectors constructed from the level set field are used as criteria to discard illegitimate cloud points. In practice, this is accomplished by discarding cloud points with normal vectors that have a negative dot product with the normal vector of the ghost point.

2.4.6 Hybrid Node Treatment

In order to update the fluid variables at hybrid nodes, we follow a procedure similar to Luo et al [30]. During the solution of the momentum and Poisson equations of the fractional step method, hybrid cells are updated as a weighted average of the solution of the governing equations and the solution obtained by interpolating from a set of nearby grid nodes in the fluid. The correction of the velocity is then carried out in the standard manner. The current approach is different from that of [30] in two ways. First, in [30] hybridization was applied to a discrete forcing method, whereas this work focuses on a sharp interface approach, where the numerical operators are modified. Second, the current work uses least-squares fitting to modify the numerical operators instead of standard bi-linear interpolants.

The construction of the blending weight is determined by the level set value of the hybrid grid point and a generalized definition for any grid point with positive level set value is

$$\alpha_P = MAX(1 - \frac{\phi_P}{\sqrt{2}\Delta x}, 0). \tag{2.34}$$

This definition has the property that $\alpha_P \rightarrow 0$ as the grid point moves away from the interface and into the fluid and $\alpha_P \rightarrow 1$ as the grid point approaches the interface.

Denoting $(\vec{u}_P^*)_{NS}$ and $(\vec{u}_P^*)_{int}$ as approximations of the intermediate velocity obtained by solving the governing equations or applying an interpolation from surrounding nodes, the hybridized algorithm computes the variables at a hybrid node P as

$$\vec{u}_P^* = (1 - \alpha_P)(\vec{u}_P^*)_{NS} + \alpha_P(\vec{u}_P^*)_{int} \tag{2.35}$$

The discrete form of the momentum equation at a node P can be written as

$$\vec{u}_P^* + \sum_{i \neq P} A_i^{\vec{u}^*} \vec{u}_i^* = \vec{R}_P^{\vec{u}^*}. \tag{2.36}$$

Here, $A_i^{\vec{u}^*}$ are the off-diagonal discretization coefficients of the momentum equation which are only non-zero in a compact region surrounding P depending on the discretization scheme which have been re-scaled such that $A_P^{\vec{u}^*}$ is unity. $\vec{R}_P^{\vec{u}^*}$ contains the explicit contributions to the momentum and equations which has been re-scaled in a manner consistent with the left-hand side of the equation. Similarly, an interpolation for the intermediate velocity at P can be written as

$$\vec{u}_P^* = \sum_{j \neq P} \beta_j^{\vec{u}^*} \vec{u}_j^*, \tag{2.37}$$

where the $\beta_j^{\vec{u}^*}$ are interpolation weights which are non-zero over a compact set of neighboring grid points. Combining the two approximations for the intermediate velocity according to Equation (2.36) results in a hybrid equation for the intermediate velocity,

$$\vec{u}_P^* + (1 - \alpha_P) \sum_{i \neq P} A_i^{\vec{u}^*} \vec{u}_i^* - \alpha_P \sum_{j \neq P} \beta_j^{\vec{u}^*} \vec{u}_j^* = (1 - \alpha_P) \vec{R}_P^{\vec{u}^*}. \tag{2.38}$$

Similarly, the blending equation for updating pressure in hybrid cells is

$$p_P + (1 - \alpha_P) \sum_{i \neq P} A_i^p p_i - \alpha_P \sum_{j \neq P} \beta_j^p p_j = (1 - \alpha_P) \vec{R}_P^p. \tag{2.39}$$

The stencil for the interpolation component of the hybridized treatment is formed in a very similar manner to the formation of extrapolations for ghost cells. The least-squares fit is computed at the hybrid point directly instead of at a probe location. The structure of the least-squares cloud for hybrid cells is shown as the gray shaded area in Figure 10. First, all grid points within a specified radius are found. Then, any potential cloud cell located in the solid is replaced with the nearest point on the boundary and the boundary condition at that location.

During the transition from ghost to hybrid status, the newly formed hybrid node will not have a valid flow history. Therefore, the explicit source term in Equation (2.36) cannot be properly formulated. The treatment of freshly-cleared cells is handled very naturally in the current hybridized formulation. The values at fresh cells are computed by simply setting the blending coefficient α_P to 1. Due to the purely implicit nature of the Poisson equation, no special treatment is required beyond the solution of the momentum equations.

2.4.7 Surface Vorticity and Force Calculation

The calculation of surface quantities in sharp interface methods is not a trivial task. The stress and vorticity fields are derivatives of the velocity field, which tend to suffer from the amplification of numerical noise. The computation of surface derivatives at arbitrary locations on the boundary is complicated by the lack of regular computational stencils, which adversely affects accuracy and introduces noise into the discrete field, especially in cases where derivatives have large magnitudes, such as in the boundary layers found in high Reynolds number flows. While this may be acceptable for tasks such as post-processing and visualization, there are numerical formulations that require the specification of higher-order derivatives of the velocity field on the immersed surface. A relevant example is the consistent treatment of pressure boundary conditions in velocity-correction methods, where the correct pressure boundary condition to be applied at a stationary immersed surface is [73]

$$\frac{\partial p}{\partial n} = -\nu(\nabla \times \vec{\omega}) \cdot \vec{n}. \tag{2.40}$$

In order for the numerically stability to be ensured, the computation of the right-hand side of Equation (2.41) should be lead to smooth results.

To address these challenges in a unified framework, surface quantities are evaluated using a least-squares procedure similar to that utilized in the construction of the current GFM method. A multidimensional polynomial of arbitrary order is fit to a set of nearest fluid, hybrid, and ghost grid points surrounding the interfacial location where the surface quantity is required. By specifying a greater number of cloud points than the number of points required to compute a unique polynomial fit, the least-squares approach provides high accuracy as well as an implicit filtering effect. Because the least-squares approach operates on data in cloud form with no structure requirement, it is very robust to the configuration of the embedded boundary with respect to the underlying computational grid. Additionally, multiple quantities can be obtained at the same point from one solution of the least-squares problem. These considerations in concert make the current least-squares approach very attractive in obtaining values of variables at surfaces as well as computing hydrodynamic forces on embedded objects.

The process of computing surface quantities starts from a quadratic least-squares polynomial fit about the interfacial location (x_{int},y_{int},z_{int}) of the form:

$$\psi = a_\psi + b_\psi x_r + c_\psi y_r + d_\psi z_r + e_\psi x_r y_r + f_\psi x_r z_r + g_\psi y_r z_r + h_\psi x_r^2 + i_\psi y_r^2 + j_\psi z_r^2 \tag{2.41}$$

$$x_r = x - x_{int} \tag{2.42}$$

$$y_r = y - y_{int} \tag{2.43}$$

$$z_r = z - z_{int} \tag{2.44}$$

The least-squares problem is set up and solved using the same procedure applied during the population of ghost and hybrid cells. A unique set of expansion coefficients can be determined from a set of 10 cloud points. However, to obtain a smooth field and avoid numerical stability issues a larger set of points is generally used. Once the expansion coefficients are available, the gradient of a field at the interfacial location is found by taking the analytical gradient of the expansion (2.40), and subsequently setting all relative coordinates to zero, which leads to the simple expression

$$\nabla\psi = b_\psi \vec{e}_x + c_\psi \vec{e}_y + d_\psi \vec{e}_z. \tag{2.45}$$

Applying a polynomial fit to each component of the velocity and denoting the leading coefficients of the expansions as (a_u,b_u,c_u,d_u), (a_v,b_v,c_v,d_v), (a_w,b_w,c_w,d_w), respectively, an expression for the vorticity vector at the surface in terms of expansion coefficients is

$$\vec{\omega}|_S = (b_w - c_v)\vec{e}_x + (c_u - a_w)\vec{e}_y + (b_v - c_u)\vec{e}_z. \tag{2.46}$$

Similarly, upon determining expansion coefficients of the pressure, (a_p,b_p,c_p,d_p), the stress tensor at the interfacial location can be formed as:

$$\underline{T}|_S = -p|_S\underline{I} + 2\mu\underline{D}|_S = -a_p\underline{I} + \begin{bmatrix} 2\mu b_u & \mu(c_u + b_v) & \mu(d_u + b_w) \\ \mu(c_u + b_v) & 2\mu b_v & \mu(c_w + d_v) \\ \mu(d_u + b_w) & \mu(c_w + d_v) & 2\mu b_w \end{bmatrix}, \tag{2.47}$$

where $p|_S$ is the surface pressure, $\underline{D} = \frac{1}{2}(\nabla\vec{u} + \nabla\vec{u}^T)$ is the rate of deformation tensor, and $\underline{I}$ is the 3x3 identity tensor.

The lift and drag forces exerted on an object are computed by numerically integrating the stress tensor over a discretization of the object surface, which is approximated as a set of planar elements of arbitrary shape. The surface elements are constructed from the level set representation of the object using the Marching Cubes method [74]. In this method, each grid cell for which a neighboring grid cell has an opposite level set sign is checked to see if the zero level set contour cuts through the grid cell volume in any orientation by comparing the sign of the level set field at the corners of the grid cell volume. 15 of the 16 basic configurations of the planar cut inside the cell are shown in Figure 11, with the remaining one being the case where all cell vertices have negative level set value. In order to efficiently determine which of the 256 overall possible configurations is at hand, a hexadecimal look up table is used to identify the case based on which corners of the grid cell volume have negative level set values.

Using this information, a triangulation of the planar surface is obtained from a second look-up table. Once a triangulation is established, it is easy to compute the geometric quantities required to evaluate a midpoint integration of the force on the object, which takes the form:

$$\vec{F} = -\oiint \underline{T} \cdot \vec{n} dA \cong -\sum_{i=1}^{Np} \underline{T} \cdot \vec{n}|_i A_i \, . \tag{2.48}$$

Here, $\vec{F}$ is the hydrodynamic force vector, and $\underline{T} \cdot \vec{n}|_i$ is the projection of the stress tensor along the normal at the centroid of a patch of the discretized surface, and $A_i$is the patch area.

2.5 Validation of the Current Interface Treatment

2.5.1 Poisson Equation with Dirichlet Conditions

In this section, we use the method of manufactured solutions to test the accuracy of the current ghost fluid approach. This procedure can be used to assess the accuracy of the method for a range of immersed boundary configurations. To that end, consider the periodic function

$$\phi(x, y) = \sin 3\pi x \sin 3\pi y. \tag{2.49}$$

The method of manufactured solutions implies that this function can be regarded as the solution of the Poisson equation,

$$\nabla^2 \phi = -18\pi^2 \sin 3\pi x \sin 3\pi y, \tag{2.50}$$

supplemented by Dirichlet boundary conditions derived from the desired solution. We apply the current ghost fluid method to solve the manufactured Poisson equation for two model problems that demonstrate accuracy of the approach for two embedded shapes: a circle of unit diameter and a nine-pointed star for which the surface is parametrized according to:

$$x(\theta) = 1 + (0.5 + 0.2\sin 9\theta)\cos 9\theta \tag{2.51}$$

$$y(\theta) = 1 + (0.5 + 0.2\sin 9\theta)\sin 9\theta. \tag{2.52}$$

Both geometries are embedded at the center of a 2x2 domain and the solution to the Poisson equation is computed in the region of the domain exterior to the embedded curve. The boundary conditions on the embedded boundary and domain boundaries are specified according to Equation (2.49).

First, the standard ghost fluid method is applied to solve the Poisson equation. This is accomplished by setting α to zero in the hybrid cells. The probe points of the ghost fluid treatment are interpolated using a quadratic polynomial fit, where the least-squares cloud has a radius of 2Δx. Numerical experimentation showed that the solution is insensitive to the cloud radius for this test case. The contours of the solution for the two boundary configurations are shown in Figure 12. The solution contours are smooth for both cases despite the strong concave corner features of the star boundary. To assess the order of accuracy of the ghost fluid method, the solution is obtained on five grids where the grids were each successive mesh has twice the resolution. In Figure 13, the maximum error versus grid spacing is displayed for both the circular and star-shaped boundaries. The convergence results are shown in Table 2. The convergence behavior for the circular boundary is very smooth, showing a consistent order of accuracy of 2.00. The convergence for the star-shaped boundary has a less regular pattern, but the case demonstrates an average order of accuracy of 1.95. For both geometries, the obtained order of accuracy very closely matches the nominal order of 2.

We examine the effect of the hybridized treatment on the accuracy of the ghost fluid method by again computing the solution to Equation (2.50), but this time we specify the blending weight using the level set-based criteria (Equation (2.34)). The results of the accuracy analysis are compared with the results obtained from the ghost fluid treatment without hybridization in Table 3. It is clear that for the circular embedded boundary, in the case of Dirichlet conditions, the hybridization has no significant effect on the maximum error. The L_2-error in the solution is increased, but only by a maximum of 7 percent. For the solution on the star-shaped domain, the situation is much different. The hybridization actually makes the maximum error decrease by nearly a factor of 2. Therefore, the effect of hybridization cannot be predicted to simply decrease accuracy, as it has just been shown that in a complicated geometry with strongly concave sections, the hybridization increases the accuracy of the computation.

2.5.2 Poisson Equation with Neumann Conditions

We now move forward to consider Neumann conditions prescribed on the immersed boundary. Consider the Poisson problem

$$\nabla^2\phi = -r^2 \sin 2\theta, \tag{2.53}$$

which is to be solved on a domain defined by the region between two concentric circles of radius $R_1 = 1.0$ and $R_2 = 1.5$. The equation is supplemented with Neumann conditions prescribed on the concentric circles,

$$\frac{\partial\phi}{\partial r}(R_1,\theta) = 0, \frac{\partial\phi}{\partial r}(R_2,\theta) = 0. \tag{2.54}$$

The analytical solution to this problem is

$$\phi = c - \frac{r^4}{12}\sin 2\theta + \frac{1}{6}\left(\frac{R_1^6 - R_2^6}{R_1^4 - R_2^4}\right) r^2 \sin 2\theta + \frac{1}{6r^2}\left(\frac{R_1^2 - R_2^2}{R_1^4 - R_2^4}\right) R_1^4 R_2^4 \sin 2\theta. \tag{2.55}$$

The solution is unique up to a constant, c, as the null space of the Poisson operator contains the constant vector. In this work, a unique solution is specified by requiring the solution to have an average of zero.

The solution to the Neumann problem is obtained with and without hybridized treatment. Again, a quadratic polynomial fit is used for the least-squares treatment, and the cloud radius is set to $2\Delta x$. The solution contours are shown in Figure 14, and the maximum errors for the two solutions are compared in Figure 15. In Figure 14, it is seen that the contours smoothly capture the zero normal derivative condition on both the convex inner boundary as well as the concave outer boundary. The error analysis of Figure 15 demonstrates second-order accuracy when the current ghost fluid method is applied to Poisson problems with Neumann conditions. The standard ghost fluid method obtains a solution with error that is approximately 4 times smaller than that of the hybridized ghost fluid method. However, the hybridized treatment is still second-order accurate.

2.5.3 In-line Oscillating Cylinder

In this section, the properties of the current numerical scheme in application to moving boundary problems are assessed for the standard and hybrid formulations. The main focus of this investigation is a characterization of the spurious oscillation of the pressure field for both the standard and hybridized schemes. In formulating the numerical scheme, there are several choices in how to apply hybridization to the fractional step algorithm. One could apply hybridization to both the computation of the intermediate velocity and the pressure, or alternatively, one could hybridize a single step in the algorithm. We seek to establish the combination which has the maximum pressure oscillation suppression ability and to determine the magnitude of the divergence which is introduced by such a treatment. We then demonstrate the behavior of the standard and optimal hybridized schemes with variation in grid spacing and time-step size.

To that end, we consider the flow past a cylinder of unit diameter which is undergoing inline oscillatory motion. The cylinder is initially located at the center of a 4x4 domain. A uniform velocity of 1.0 is specified at the west, north, and south boundaries, and a convective outflow condition is specified at the east boundary. The cylinder's center of mass and velocity are determined according to,

$$x_c(t) = x_c(0) + X_o \sin(2\pi f t). \tag{2.56}$$

$$u_c(t) = 2\pi f X_o \sin(2\pi f t) \tag{2.57}$$

The oscillation amplitude, X_o, and the frequency of oscillation, *f*, are set to 0.15 and 0.2, respectively. The Reynolds number, based on the cylinder diameter, *D*, and the inlet velocity, *U*, is 100.

In order to determine which hybridization strategy is the most effective at suppressing pressure oscillations, we perform a set of simulations with a fixed grid spacing and CFL number while applying the set of combinations of hybridization to the fractional step algorithm in turn. The grid spacing is set to $\Delta x = 1/16$, and the CFL number is fixed at 0.1. The combination of large grid-spacing and small time-step sizes should lead to severe pressure fluctuations. The combinations of hybridization tested are: 1) no hybridization, 2) hybridization of the momentum equation, 3) hybridization of the Poisson equation, and 4) hybridization of both the momentum and Poisson equations.

Figure 16 shows the time history of the drag force as well as the time history of the average and maximum divergence in the solution domain for each hybridization strategy. As can be seen from Figure 16.a, the standard ghost fluid formulation leads to the formation of strong pressure oscillations with the largest fluctuations biased towards the convex side of the drag curve. The bias of the fluctuations is due to the spurious volume introduced by freshly-cleared cells. When a cell transitions from a ghost to hybrid point, the volume associated with the source term in the Poisson equation is that of the whole cell, which is larger than the true portion of the hybrid cell located in the fluid. This causes a sudden increase in pressure in the freshly-cleared cells, which leads to a sudden pressure force in a non-physical direction. The biased nature of the pressure fluctuations, as well as their irregular pattern means that the true force curve cannot be extracted by smoothing the noisy results. The maximum and average divergence are of $O(10^{-2}\text{-}10^{-3})$ and $O(10^{-4}\text{-}10^{-5})$. The average divergence history is very smooth when the cylinder is moving against the fluid (t = 2-5), but becomes extremely noisy when the cylinder moves with the fluid (t = 5-7). This is most likely due to the fact that the velocity on the downstream side of the cylinder is of much lower magnitude than that of the upstream side, such that fresh cell population occurs over a smoother field.

Next, hybridization is applied solely to the momentum and Poisson equations in turn. From Figure 16.b, it is observed that hybridizing the momentum equation redistributes the force fluctuations to be more symmetric during the times when the cylinder is moving against the uniform far-field condition, while decreasing the magnitude of fluctuations while the cylinder is moving with the fluid. The maximum divergence is slightly lowered, but also becomes more irregular in time, as does the average divergence. The hybridization of the Poisson equation (Figure 16.c) reduces the magnitude of the force fluctuations by nearly a factor of 4 as compared to the standard ghost fluid method. However, the directional bias of the fluctuations remains, and in stark contrast to applying hybridization to the momentum equation, in the current case, the maximum divergence is now O(1).

Finally, we investigate performing hybridization in both the momentum and Poisson equations. The results are shown in Figure 16.d. It is observed that the fluctuations are further reduced and less biased than the results of hybridizing the Poisson equation alone. The maximum divergence is still markedly larger than the standard ghost fluid approach, but is nearly half that obtained by hybridizing the Poisson equation alone. The magnitudes of the average divergence are fairly insensitive to the hybridization scheme.

Having established that the hybridization of both the momentum and Poisson equations leads to the strongest suppression of force fluctuations, we now analyze the dependence of the standard approach and optimal hybridized approach on grid spacing and time-step sizes. First, the problem is simulated with a fixed CFL number of 0.1, and grid spacings of Δx = 1/16, 1/32, and 1/64 for both the standard ghost fluid approach and the optimal hybridized method. The drag force curves are plotted in Figure 17, where the results of the standard method are shown on the left, and the hybridized method results are given on the right. Both approaches experience a significant suppression of the pressure oscillations as the grid spacing is decreased, and both approaches are converging to the same drag curve. The hybridized approach has significantly smaller oscillations at all grid spacings. The oscillation amplitude of the standard approach on the finest grid spacing is comparable to that of the hybridized method results on the coarsest grid spacing.

The procedure is repeated to examine the dependence of the force fluctuations on time-step size. The problem is simulated with a fixed grid spacing of dx = 1/16, while the CFL = 0.1, 0.2, and 0.4. The results are shown in Figure 18. Again, the numerical oscillations of the hybridized approach are far smaller than those of the standard approach. The current numerical experimentation indicates that the maximum divergence produced by the hybridized approach is insensitive to grid spacing or time-step size.

2.6 Conclusion

In the current chapter, the details of the numerical scheme have been described. A novel ghost fluid method has been constructed where least-squares fitting is utilized to extend flow variable values into a set of ghost grid nodes inside solid objects. The extrapolating fields satisfy the boundary condition on the solid surface in multiple locations thereby mores strongly enforcing the boundary condition in a discrete fashion. The interface treatment was implemented in the context of a four-step fractional step method. Additionally, hybridization was incorporated into the interface treatment in order to alleviate the strong artificial oscillations in the pressure field associated with moving objects. This was accomplished by constructing a set of grid nodes adjacent to the solid surface and inside the fluid where the flow variable values are updated by a weighted average of solutions from the governing equations and interpolation. The application of this hybridized approach to the various steps of the fractional step algorithm was investigated. It was determined that applying hybridization to both the momentum and Poisson equations lead to the strongest suppression of artificial force fluctuations. However, this approach also leads to a large divergence around the interface. It has been established that this divergence is localized to the interface, and that the average divergence in the domain is of an acceptable magnitude.

Table 1 The level set criteria for the four types of computational points in the hybridized ghost fluid method.

Grid Point Type	Level Set Condition
Fluid Node	$\phi(\vec{x}_P) > \sqrt{2\Delta x}$
Ghost Node	$\exists\, \vec{x}_{nbr}: \phi(\vec{x}_P)\phi(\vec{x}_{nbr}) < 0$
Hybrid Node	$0 < \phi(\vec{x}_P) \leq \sqrt{2}\Delta x$
Fresh Node	$\phi(\vec{x}_P^{n+1}) > 0;\;\; \phi(\vec{x}_P^{n})\phi(\vec{x}_P^{n+1}) < 0$

Table 2 Accuracy analysis of the solution of Equation (2.49) for a domain with an embedded circle and an embedded nine-pointed star.

	circle		nine-pointed star	
Δx	ε_∞	order	ε_∞	order
0.04	$4.80 \text{x} 10^{-4}$	----	$8.12 \text{X} 10^{-4}$	----
0.02	$1.23 \text{x} 10^{-4}$	1.96	$2.23 \text{X} 10^{-4}$	1.86
0.01	$3.08 \text{x} 10^{-5}$	2.00	$4.74 \text{X} 10^{-5}$	2.23
0.005	$7.71 \text{x} 10^{-6}$	2.00	$1.38 \text{X} 10^{-5}$	1.78
0.0025	$1.93 \text{x} 10^{-6}$	2.00	$3.61 \text{X} 10^{-6}$	1.93

Table 3 A comparison of the accuracy of the current ghost fluid method with and without hybridization for the embedded boundary shapes detailed in Table 2.

	circle		nine-pointed star	
Δx	ε_∞ (α=0)	ε_∞ ($\alpha \neq 0$)	ε_∞ (α=0)	($\alpha \neq 0$)
0.04	$4.80 \text{x} 10^{-4}$	$4.84 \text{x} 10^{-4}$	$8.12 \text{X} 10^{-4}$	$4.84 \text{x} 10^{-4}$
0.02	$1.23 \text{x} 10^{-4}$	$1.23 \text{x} 10^{-4}$	$2.23 \text{X} 10^{-4}$	$1.23 \text{x} 10^{-4}$
0.01	$3.08 \text{x} 10^{-5}$	$3.08 \text{x} 10^{-5}$	$4.74 \text{X} 10^{-5}$	$3.08 \text{x} 10^{-5}$
0.005	$7.71 \text{x} 10^{-6}$	$7.71 \text{x} 10^{-6}$	$1.38 \text{X} 10^{-5}$	$7.79 \text{x} 10^{-6}$
0.0025	$1.93 \text{x} 10^{-6}$	$1.93 \text{x} 10^{-6}$	$3.61 \text{X} 10^{-6}$	$1.93 \text{x} 10^{-6}$

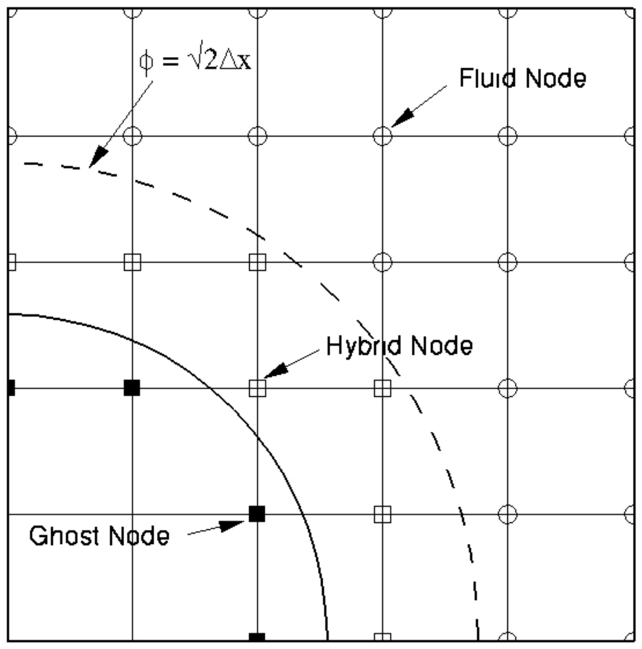


Figure 4 A depiction of the character of each of the types of grid nodes in the current ghost fluid method. Fluid nodes are updated by solving the Navier-Stokes equations, while ghost nodes are given values by boundary condition preserving extrapolations. Values at hybrid nodes are computed by a weighted average of interpolations and solutions of the Navier-Stokes equations.

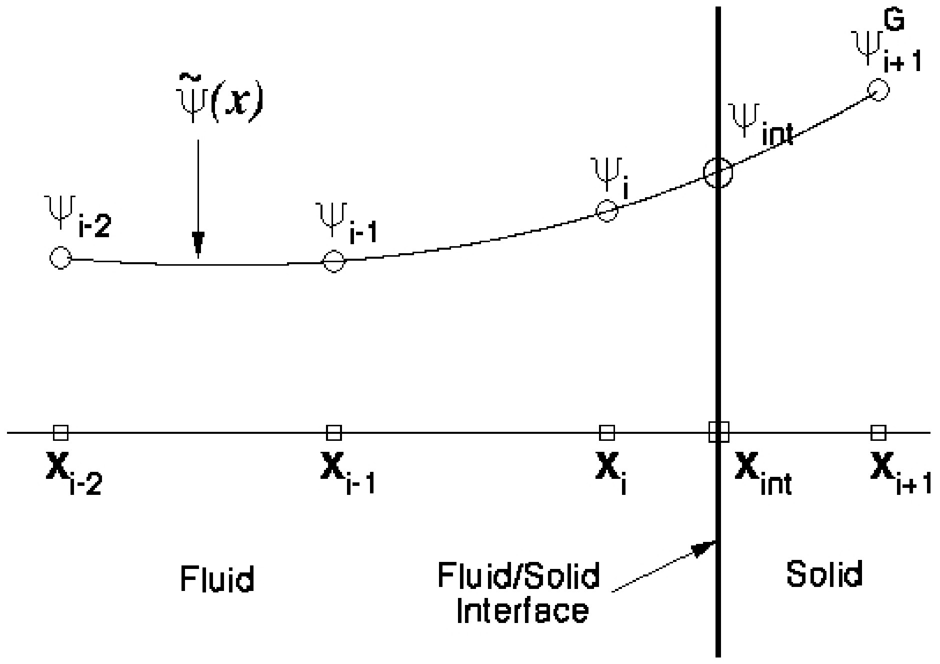


Figure 5 A one-dimensional example of the concept of a ghost fluid method. An extrapolation fit to values at the interface and neighboring grid nodes in the fluid is used to populate the required value in the solid.

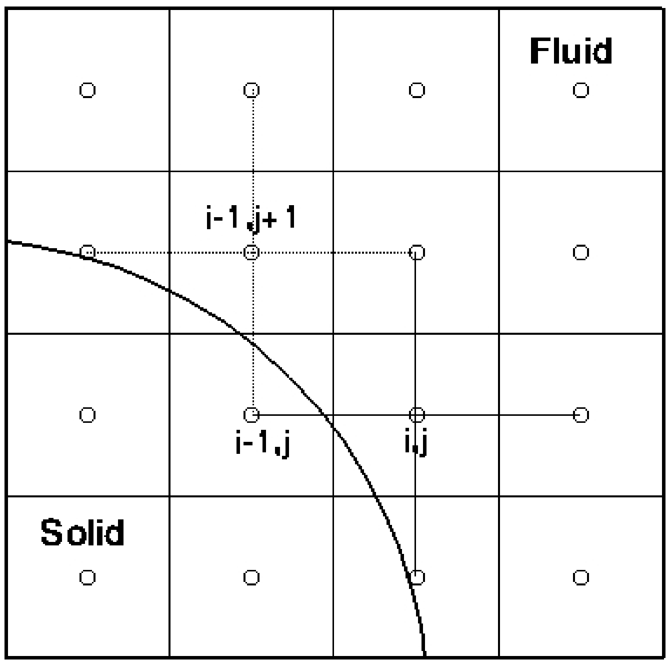


Figure 6 A depiction of the discretization stencil for a grid node adjacent to a solid object. In this case, the west neighbor of (i,j) is in the solid phase, making the Laplacian operator ill-posed. The concept of a ghost fluid method is to extend the flow field to nodes similar to (i,j) such that the boundary condition on the solid surface is satisfied.

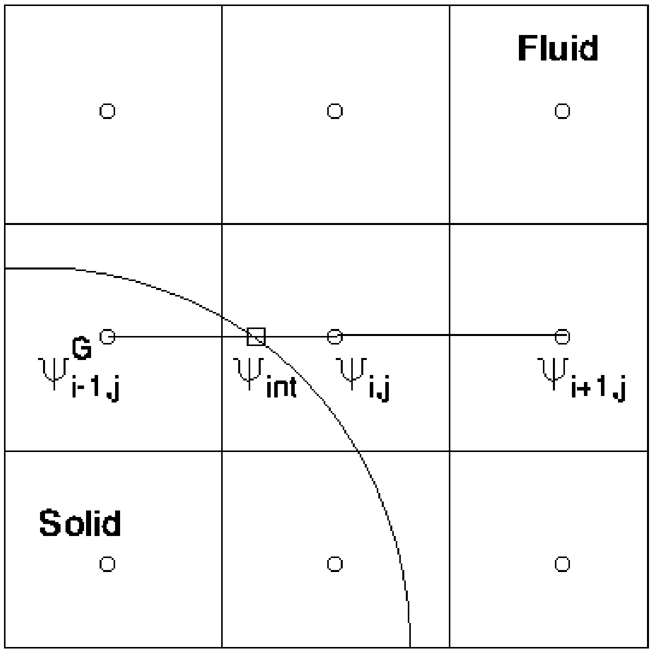


Figure 7 A depiction of a one-dimensional extrapolation ghost population approach. In this approach, ghosts are assigned values by fitting an extrapolating polynomial to the values at the interface and neighbor grid nodes in a Cartesian direction.

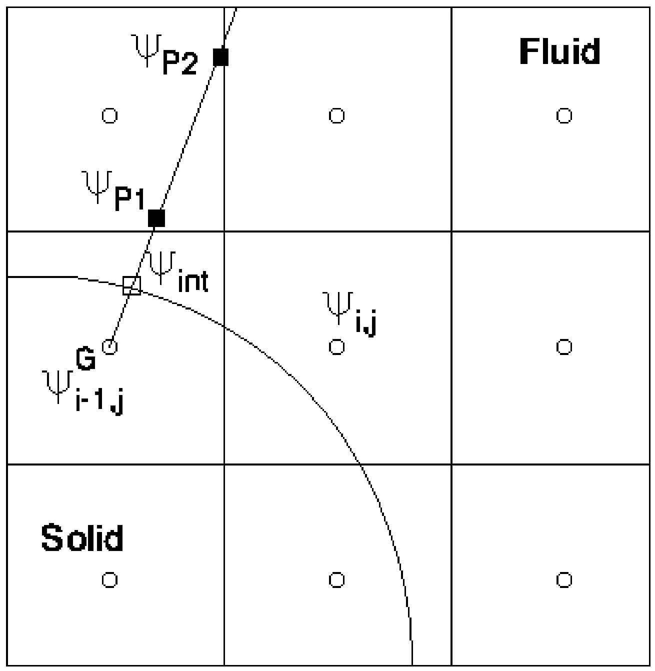


Figure 8 A schematic of a normal probe ghost fluid method. Ghosts are assigned values by fitting an extrapolating polynomial along the direction normal to the interface. The interface value at the normal intersection of the solid surface and the values at a set of probe points are used to construct the extrapolating polynomial. Probe points are located at arbitrary locations and therefore must be interpolated from surrounding grid nodes.

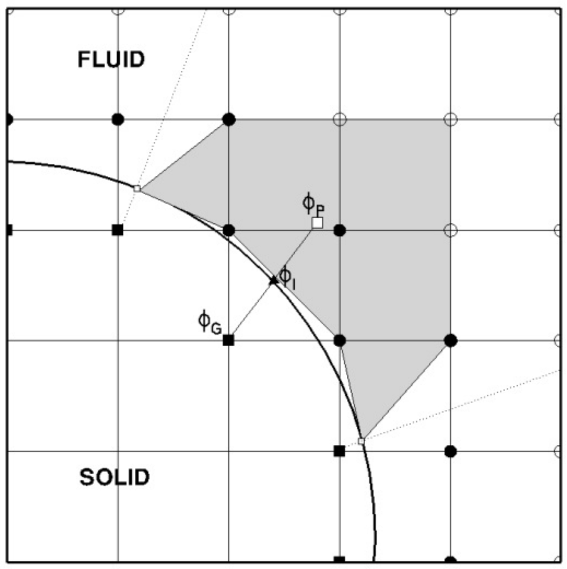


Figure 9 A schematic of the current construction of the least-squares cloud used to interpolate the value at a probe point for ghost cell population. A fast search algorithm is used to identify grid points in the neighborhood of the probe location. Then, any grid node which has been found which is in the solid is replaced by the boundary condition at the closest point on the interface (open white circles). The ghost node is excluded from the cloud.

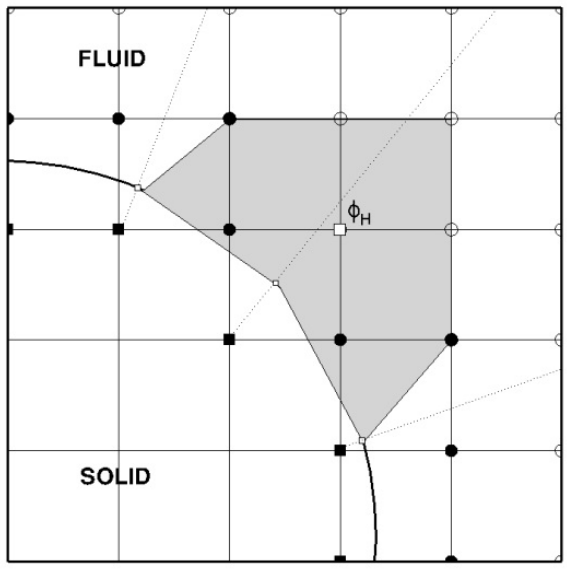


Figure 10 A schematic of the current construction of the least-squares cloud used to interpolate the value at a probe point for hybrid cell population. A fast search algorithm is used to identify grid points in the neighborhood of the probe location. Any grid node which has been found which is in the solid is replaced by the boundary condition at the closest point on the interface (open white circles). The hybrid node is excluded from the cloud.

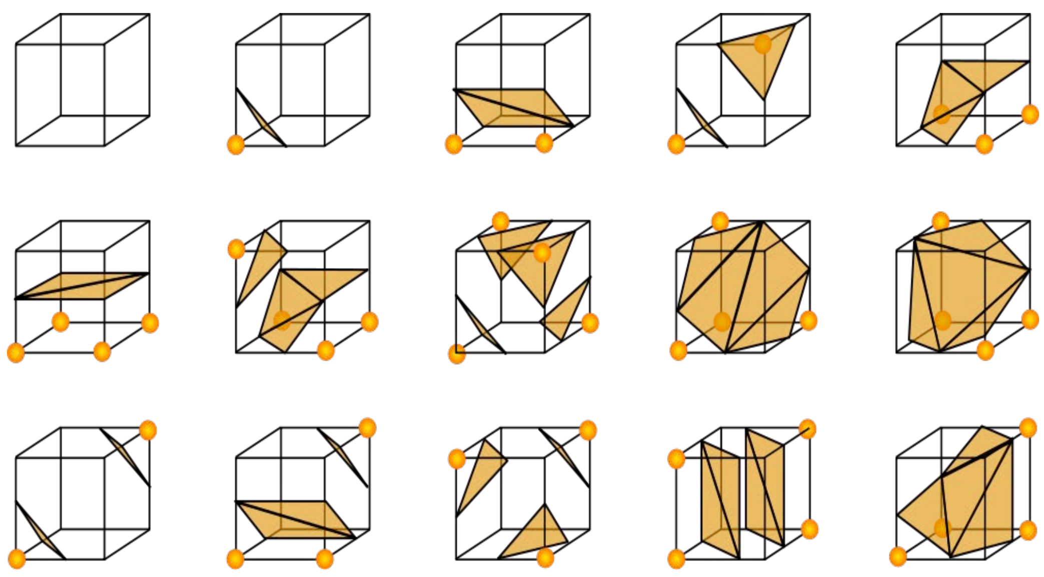


Figure 11 The 16 standard configurations of the Marching Cubes algorithm [74]. The intersection of an isocontour with the edges of a computational cell leads to arbitrary polygonal shapes. Marching cubes efficiently determines the shape from a hexadecimal lookup table using the set of cut edges.

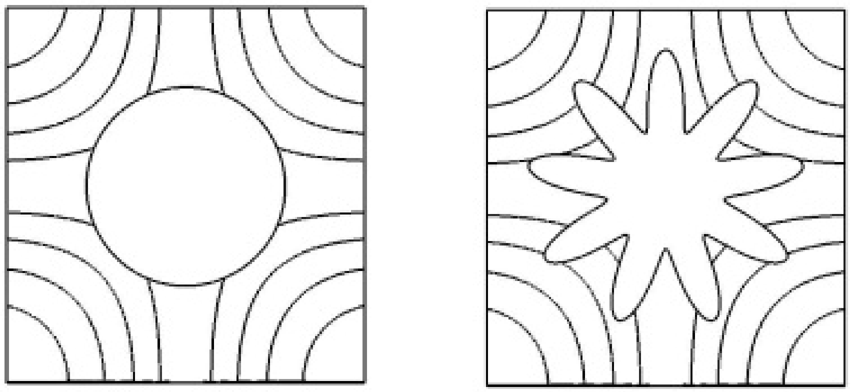


Figure 12 Contours of the solution of Equation (2.49) for a domain with an embedded circle (left) and a domain with an embedded nine-pointed star (right). The current ghost fluid method produces smooth accurate contours even for the highly concave region of the star case.

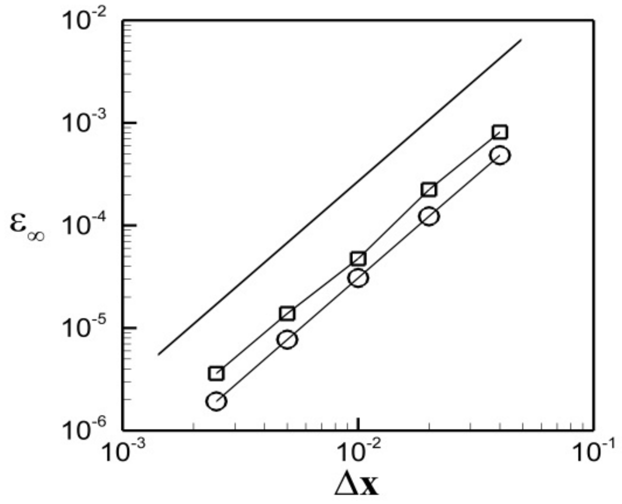


Figure 13 Error analysis of the solution of Equation (2.49) in the L_∞-norm for a domain with an embedded circle (circles) and an embedded nine-pointed star (squares). The maximum error is computed on 5 grids with a mesh refinement ratio of two. The solid black line shows a slope of 2 for comparison.

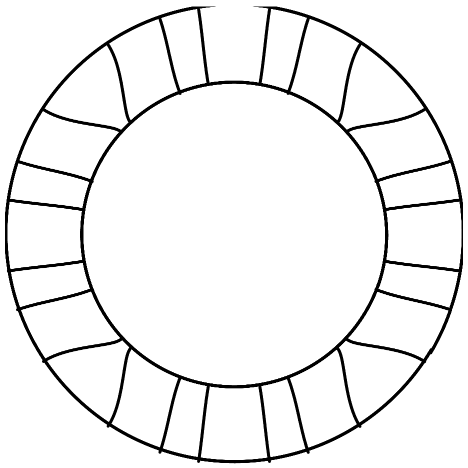


Figure 14 Solution contours for a Neumann problem on concentric circles. The solution smoothly captures the zero normal derivative condition on the concave and convex boundaries.

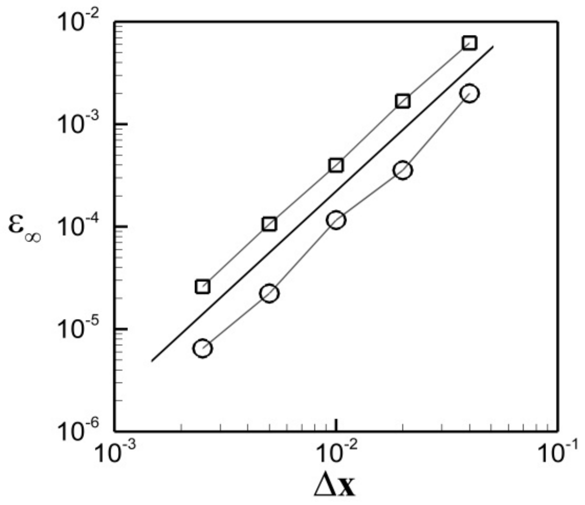


Figure 15 Error analysis of the solution of Equation (2.52) in the L_∞-norm for the standard (circles) and hybridized (squares) ghost fluid method. The solid black line shows a slope of 2 for comparison.

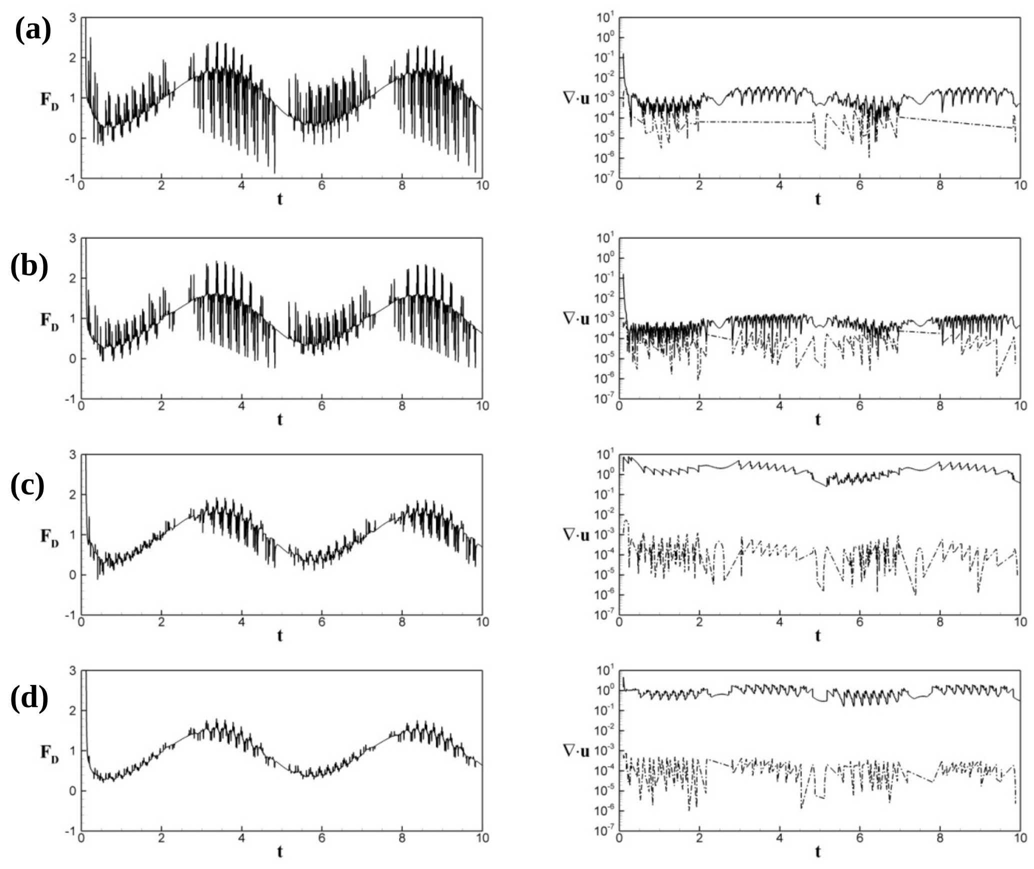


Figure 16 Time histories of drag force (left) and divergence (right) for an oscillating inline cylinder where the ghost fluid method has (a) no hybridization, (b) hybridization of the momentum equation, (c) hybridization of the Poisson equation, and (d) hybridization of both the momentum and Poisson equations. The maximum divergence is denoted by the solid black line, while the average divergence is represented by the dashed line.

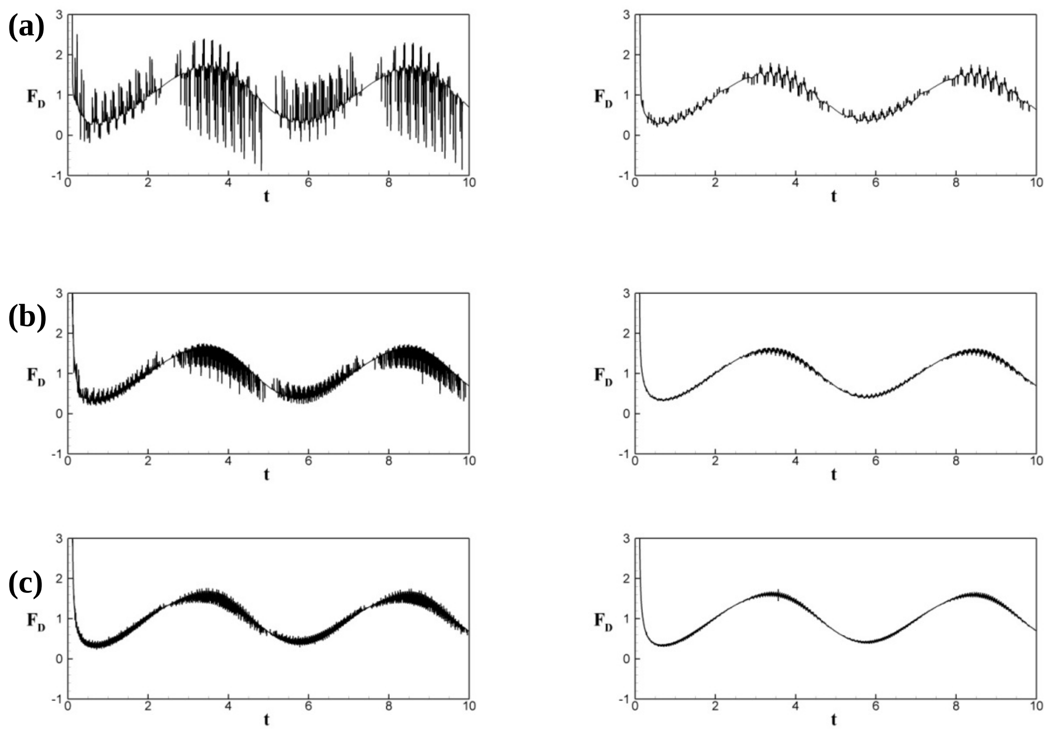


Figure 17 Time histories of drag force for an oscillating inline cylinder solved with the standard ghost fluid method (left) and the optimal hybridized method (right) for a CFL number of 0.1 and a grid spacing of (a) dx = 1/16, (b) dx = 1/32, and (c) dx = 1/64.

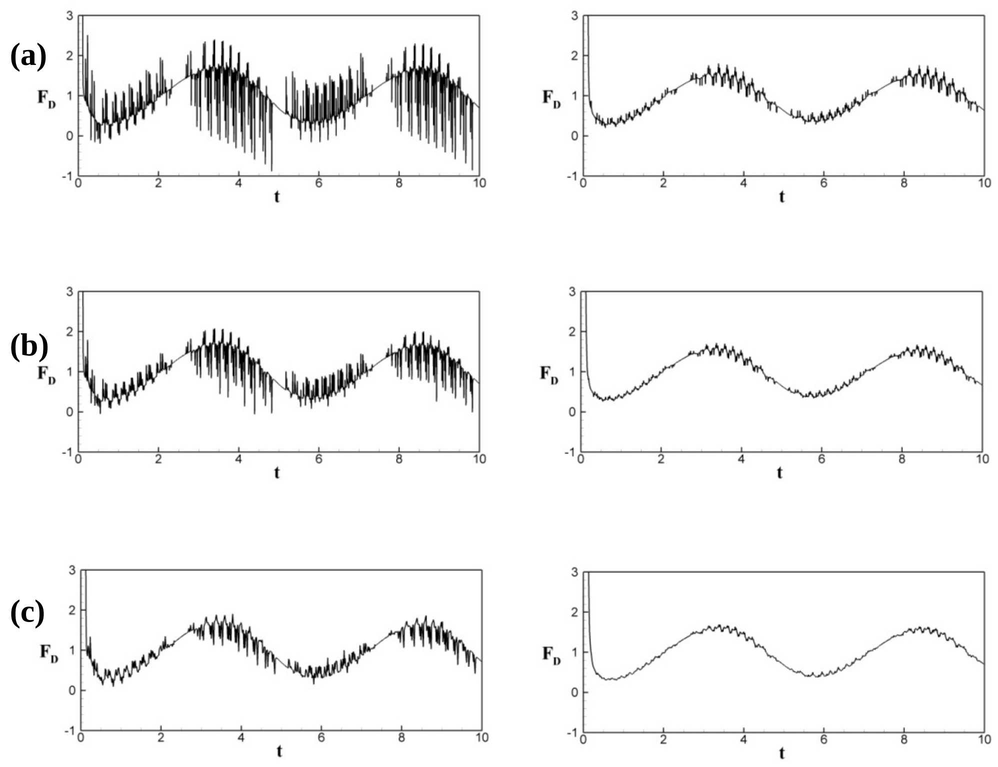


Figure 18 Time histories of drag force for an oscillating inline cylinder solved with the standard ghost fluid method (left) and the optimal hybridized method (right) for a grid spacing of dx = 1/16 and (a) CFL = 0.1 (b) CFL = 0.2, and (c) CFL = 0.4.

CHAPTER 3

PARALLEL ADAPTIVE FRAMEWORK

3.1 Overview of the Current Framework

The focus of Chapter 3 is the development of an efficient and user-friendly framework which enhances the applicability of the sharp-interface Cartesian grid method formulated in Chapter 2 to computationally intensive moving boundary problems. The framework developed in this thesis is manifested in the pELAFINT3D simulation package. pELAFINT3D is a Fortran-based software package which efficiently solves moving boundary problems by utilizing the current sharp-interface method in conjunction with a partitioned and pruned Cartesian mesh which is enriched with adaptive octree-based grid refinement. The mesh adapts to the flow physics as well as to the motion of any embedded geometry. Mesh pruning occurs in regions that have been identified as extraneous to the computation, such as the interior of solid objects. This significantly reduces the memory foot print of pELAFINT3D in the case of tortuous tube-like geometries. This greatly enhances the applicability of the pELAFINT3D to certain cardiovascular systems.

Significant flexibility is gained in the pELAFINT3D package by the choice of representing geometries as level set fields. The implicit nature of level sets allows pELAFINT3D to handle strong topological changes in boundary configurations such as merging and pinching off in a unified fashion. The level set representation also simplifies several operations such as the construction of sharp-interface discretization operators. The level set representation is a useful indicator function in mesh refinement and trimming operations. Additionally, as will be seen in Chapter 6, level set representations provide a powerful connection to the methods of image analysis, which allows for the formulation of a novel image to computation technique.

The basic operation of the current algorithm is demonstrated in Figure 19. The complex geometry for which the flow is to be computed is shown in Figure 19(a). In this case, the geometry is an aortic valve which is described by a surface mesh. A Cartesian grid is constructed such that the surface mesh is completely contained in the grid (Figure 19(b)). The regions of the Cartesian grid which are not required to complete the simulation are identified and pruned away. The result of the pruning operation is shown in Figure 19(c). Finally, a refined mesh is constructed (Figure 19(d)), and the simulation is carried out using the Cartesian grid method described in Chapter 2.

The level set representation of geometries utilized in the pELAFINT3D framework allows for considerable flexibility in how geometric information is provided to the pELAFINT3D package. Currently level set fields may be constructed by three methods. The most general technique for constructing a level set representation is from a user-provided surface mesh. Level set fields may also be constructed from data that may be obtained from imaging modalities such as MRI by appealing to methods from the field of image analysis. This level set construction method forms the basis of a direct image-to-computation algorithm which is the subject of Chapter 6. Finally, level set fields may be constructed from analytical expressions.

The algorithmic details of the pELAFINT3D package are discussed in the remainder of the current chapter. First, the parallel mesh structure which is at the heart of the package is presented, and a highly scalable adaptive meshing strategy is developed.

The adaptive meshing strategy is used to construct a scalable mesh initialization procedure which is robust and capable of producing high quality initial mesh refinement even in the case of geometries that are not well represented by the Cartesian base mesh. A discussion of a parallel level set construction technique is given next, followed by a discussion of the parallel linear solvers used in the current work is given. Finally, the pELAFINT3D package is demonstrated to work efficiently on tens of thousands of cores.

3.2 Parallel Mesh Structure

In the current implementation, the computational grid is a completely distributed Cartesian base mesh which is enriched with local refinement and mesh pruning. The mesh is partitioned using the MPI library [75] such that no MPI process contains a copy of any globally sized mesh entity. Local refinement is achieved using a forest of octrees approach [28]. Each base mesh cell holds the root of a fully-threaded octree. A fully-threaded octree is a pointer-based data structure which originates at a root node. The root node may be sub-divided into 8 child nodes, each of which is stored in a child pointer at the root. Each of the child nodes may subsequently be divided using pointers. Each node in the octree stores pointers to all neighboring nodes in all cardinal directions. The octree satisfies a 2:1 balancing constraint, where no node in the tree has a neighboring node that is more than one level coarser or finer. The set of octree nodes that have no children will be called leaf cells. The leaf cells define a grid cell with the size specified by the level the leaf is located at in the octree. Flow variables are only stored on leaf cells.

The use of adaptive meshing of Cartesian grids is an established concept. The Adaptive Mesh Refinement (AMR) approach refine a grid in a nested set of regular rectangular patches [76, 77]. AMR libraries such as PARAMESH [78] and SAMRAI [29] have been developed to remove the burden of mesh management from application programmers. The use of octree-based refinement has also been used by many researchers. This approach forms the foundation of the GERRIS flow solver [79] and p4est libraries [28]. The concept of pruning of a Cartesian grid is similar to the Dynamic Tubular Grid (DTG) approach which was proposed by Nielsen and Museth [67] as a more memory efficient alternative to local mesh refinement to store fields for which values are not required over the entire Cartesian grid. In the DTG approach, data structures required to compute the evolution of a quantity are stored on a sparse tube of rectangular grid cells only where necessary. This concept was extended to a Cartesian grid solver by de Zelicourt et al. [80]. However, to our knowledge, the DTG approach has not been enhanced with local mesh refinement and massive parallelization in the context of a Cartesian grid solver.

An example of a mesh generated by the pELAFINT3D package is shown in Figure 20. In this case, the mesh is partitioned between four MPI processes which are demarcated in the figure by the four colored regions. The middle of the mesh has been pruned away, as represented by the open white area. Two levels of refinement have been applied in a circular pattern.

In the parallel context, the mesh is completely distributed with partitioning occurring at the base mesh level. Due to the partitioning along base cells, all leaf cells contained in a base mesh cell are owned by the same partition as can be seen by the partitioning pattern in Figure 20. Each partition is supplemented by a duplication of a layer of cells owned by neighboring partitions such that complete discretization stencils can be formed on every partition. The concept of a ghost layer is made clear in Figure 21.

The figure on the right shows the partitioning of the base mesh between three MPI processes. In the left figure, the partitions are separated to clearly show the halo of ghost cells that are attached at each partition edge. In the figure, the large chunks of the same color are the cells owned by each partition. The cells with different coloring around the edges of the chunks represent ghost cells. As is clear from the figure, the ghost cells are an exact duplicate of the cells at the edge of neighboring partitions.

The local form of the Cartesian base mesh is in actuality an unstructured mesh. With consideration of implementation, the base mesh on each partition is stored as an array of pointers, each of which points to the root of the octree contained in a base cell. This implementation is particularly useful in the case of adaptive repartitioning of the mesh and mesh pruning, as it allows temporary pointers to hold the octree structures that will not be moved during the operation, and simple deletion of pointers for base cells that are to be trimmed. This not only simplifies the implementation of the repartitioning operation, but also avoids expensive duplication of complete octree structures that will not be moved off-process.

In a distributed framework there exists a need for separate local partition and global numbering systems for grid cells. The global indices of a grid node are often required to properly complete operations across partition boundaries, whereas partition numbering allows for variables to be stored in appropriately sized data structures. Examples of operations for which the global numbering is required include the establishment of ghost layers and the assembly of global coefficient matrices. A memory efficient mapping between global and local numbering is accomplished using hash table data structures where the global index is used as a key to a hashing function which returns the correct entry in a mapping array to obtain the associated local index.

In the case of the base mesh, two sets of numberings are used. The natural numbering of the Cartesian grid (Figure 22(a)) is used to assign coordinates to each base mesh cell, as well as in the construction of the base mesh ghost layer. This numbering is continuous along rows of the base mesh, but is not continuous on the partition. As is depicted in in Figure 22(a), the natural base mesh numbering is such that the numbering is established as if no mesh has been pruned. This ensures that the process of locating the closest grid point to an arbitrary location can be performed in a straight-forward and efficient manner.

Continuity of numbering on partitions is required by the PETSc [81] and [82] ParMETIS libraries, which are used by pELAFINT3D to solve linear systems and compute load-balanced partitions. Consequently, a second global numbering is assigned which will be called the application ordering. In this numbering scheme, base cells are numbered from 1 to NB0 on partition 0, NB0+1 to NB0+NB1 on partition 1, and so forth. Here, NB0 and NB1 are the number of base cells owned by partition 0 and partition 1. An example of the base cell application ordering is shown in Figure 22(b), where partition 0 is shown in dark blue, partition 1 is colored light blue, partition 2 is green, and partition 3 is yellow. Note that the application ordering is constructed such that pruned cells are accounted for in the numbering scheme.

With regards to leaf cell numbering, no natural ordering is necessary and only an application ordering is constructed. All child cells of a parent are numbered consecutively as shown in Figure 22(d). This is performed recursively, which induces a partial z-ordering of leaf cells inside a base cell. This improves memory locality and cache reuse when performing operations involving neighbor grid cells.

3.3 Parallel Mesh Construction and Refinement

The local mesh refinement technology utilized to enhance the Cartesian mesh is built around a set of key operations. These operations include refinement and coarsening of an octree structure inside each base cell, as well as ensuring that the refined mesh satisfies the 2:1 balancing constraint. The implementation of the refinement and coarsening procedure is a purely communication free task, and so while this procedure may affect the serial performance of the current algorithm, it does not influence the algorithm’s parallel scalability. The non-local nature of the 2:1 balance constraint requires that balance information propagate across partition boundaries, and therefore this operation has a significant effect on the scalability of the mesh refinement procedure. In this section, the mesh refinement algorithm is outlined, and a detailed discussion of the 2:1 balancing procedure is given. A discussion of the mesh repartitioning operation concludes this section.

3.3.1 Parallel Local Mesh Refinement Algorithm

The parallel LMR algorithm can be summarized according to the following set of steps:

1. Compute the quantities to be used as indicator fields for determining whether refinement or coarsening is required for a mesh cell using available data.
2. Set refinement/coarsening flags for nodes in the octree structure.
3. Perform refinement and coarsening.

4. Perform 2:1 octree smoothing to ensure that no cell has a neighbor that is more than one level coarser or finer.

5. Update level set tube to flow mesh numbering.

6. Interpolate or restrict variable values for locally owned cells created by refinement or coarsening.

7. Repartition the mesh.

In this work, the vorticity and velocity gradient fields are used to determine whether refinement or coarsening is required for a particular cell. If h is a characteristic grid spacing, and $\varepsilon_{|\vec{\omega}|}$ and $\varepsilon_{|\nabla\vec{u}|}$ are user-specified tolerances for the vorticity and velocity gradient magnitudes, then a grid cell is determined to require refinement if $|\vec{\omega}|h > \varepsilon_{|\vec{\omega}|}$ or $|\nabla\vec{u}|h > \varepsilon_{|\nabla\vec{u}|}$. If on the other hand, $|\vec{\omega}|h < 0.25\varepsilon_{|\vec{\omega}|}$ and $|\nabla\vec{u}|h < 0.25\varepsilon_{|\nabla\vec{u}|}$, the cell is determined to require coarsening.

Assuming that the values of all variables are current in the partition ghost layers, all steps except 4 and 7 of the above algorithm can be performed without communication. Step 7 is not strictly a part of the LMR algorithm. Therefore, Step 4 is the determining factor of the scalability of the LMR procedure. Truly scalable algorithms for octree mesh smoothing have only been introduced recently. The scalable implementation of this operation is the subject of the next section.

3.3.2 Parallel Octree Smoothing

In order to simplify the construction of discretization operators on the locally refined grids, the octree structure is required to satisfy a 2:1 balance constraint. A mesh satisfying the balance constraint has no cell for which a neighboring grid cell across cell faces, edges, or corners is more than one level finer or coarser. An example of an unbalanced two-dimensional version of an octree, a quadtree, and its balanced completion are shown in Figure 23. In this example, the quadtree mesh is distributed across to partitions, P0 and P1, which are distinguished by the grey and white shading. Note that the balanced completion of the octree mesh requires propagation of information across the boundary between partitions.

Enforcement of the 2:1 octree balance constraint is non-local operation as the resolution of imbalances has a ripple effect which is not confined to direct neighbors of the cell for which an imbalance has been resolved. A naïve algorithm for mesh smoothing may proceed by attempting to fix balances as they are detected when traversing the octree in its natural numbering. Unfortunately, resolution of an imbalance between a cell and one of its neighbors may introduce a new imbalance between the neighbor and its respective neighbors. This strategy requires a recursive algorithm which checks the balance status of neighbor cells as imbalances are resolved.

A depiction of the recursive behavior of this naive 2:1 balancing algorithm is shown in Figure 24. The balancing is applied to a cell represented by a closed black circle, and the mesh is distributed over two partitions, where the white region belongs to partition 1 and the grey is owned by partition 2. In Figure 24(a), an imbalance originally exists between the cell and its east neighbor. In Figure 24(b), the imbalance of Figure 24(a) is resolved. Unfortunately, the resolution leads to imbalances with the west, north, and north-west neighbors of the original cell. The newly introduced imbalances are resolved in Figure 24(c), but this again leads to the introduction of new imbalances. In Figure 24(d), the resolution of the imbalance formed in Figure 24(c) introduces an imbalance across the partition boundary. In Figure 24(e), the imbalance of the neighboring partition cell is resolved.

As is evident from Figure 24, this approach may lead to an explosion of recursive function calls. More importantly, establishing an efficient parallel communication pattern is impossible as imbalances may be introduced or resolved across partition boundaries at any time during the balancing operation. In order to address this computational challenge, Tu et al. [83] proposed the Prioritized Ripple Propagation algorithm which builds on the realization that if cells at the finest level are balanced before cells at any coarser level, no future balancing operations can alter their balanced status. Moving on to the next finest level and balancing leads to the same result that no future operation can modify the balanced status of the two finest levels of refinement. Applying this strategy iteratively all the way to the coarsest mesh leads to an efficient algorithm with an optimal number of balancing operations as well as a completely deterministic parallel communication pattern. A parallel formulation of this algorithm is very easily accomplished by decomposing the operation into a balancing of local cells at the current level against local neighbors, and a balancing of ghost cells at the current level against local neighbors [28, 84, 85]. The cells at the edge of a partition that are at each specified level can be communicated in one message per level. The communicated cells can be reconstructed in the ghost layer of neighboring partitions, and these ghost cells can be used to apply balancing across partition boundaries.

The main components of this algorithm are: 1) the construction of a table of octree leaf nodes that are at the current level for which balancing is applied, 2) communication and reconstruction of partition boundary leaf nodes at the current level on neighboring partitions, and 3) a balancing operation. If new leaf nodes are introduced during the balancing operation at a specified level, they naturally enter the leaf node table at the next coarser level. With a general description of the method in place, the parallel Prioritized Ripple Propagation algorithm can be summarized according to the following set of steps:

1. Set the level to the most refined level.
2. Form a table of all locally owned leaf cells at the current level.
3. Find all leaf cells in the table that are at a partition boundary. Start communicating these cells to neighboring partitions.
4. For all cells in the table, find any neighbors that are more than one level coarser than the table cell and refine the neighbors until they satisfy the balance constraint.
5. Finish the communication started in Step 3.
6. Construct communicated leaf cells on neighboring partitions.
7. Resolve balance violations between communicated cells and locally owned cells.
8. Destroy the table at the current level.
9. If level > 3, level = level – 1, otherwise exit the loop.
10. Return to Step 2.

As the resolution of imbalances at a lower level can never introduce imbalances at a more refined level, no recursive algorithm is required. Therefore, this algorithm has a definite communication pattern with one communication required for each mesh level which is smoothed. Furthermore, no collective communication is required as the only information that needs to be communicated is the state of cells at a partition edge to the partition that adjoins that edge. This approach is extremely efficient and has been demonstrated to scale to over 220,000 cores [28]. Additionally, the mesh obtained at the end of the algorithm is unique and identical with the mesh that would be obtained in the serial case.

The current algorithm is applied to the most refined mesh level of an octree mesh which is distributed between two partitions, P0 and P1, in Figure 25(a)-(d). The left column shows the state for the cells owned by P0 and its ghost layer during the balancing procedure, and the right column shows the state of the mesh on P1. At the start of the balancing procedure, the ghost cells are devoid of any refinement. In this example, all cells at the current level are owned by P0. These cells which compose the level table formed in step 2 of the current algorithm are marked with symbols in Figure 25(a). The open symbols represent cells at the current level that are at the edge of P0 and must be communicated to P1 in step 3. In Figure 25(b), the table cells are balanced against their locally owned neighbors. Because P1 has no cells in its level table, no balancing occurs in the right panel at this stage. Next, in Figure 25(c), the cells at the edge of P0 are communicated to P1, where they are reconstructed. Finally, in Figure 25(d), the cells just constructed in the ghost layer of P1 are balanced against their neighbors that are owned by P1.

3.3.3 Adaptive Repartitioning

During the course of refining the mesh, significant load imbalances may be introduced if a set of partitions refine or coarsen significantly while other partitions remain mostly unchanged. In this case, it is prudent to re-assign the mesh distribution such that the work distribution is again balanced amongst processes. In this work, a set of weights that estimate the amount of computational work in each base cell are computed and provided to partitioning software package ParMETIS [82]. ParMETIS attempts to compute a new partitioning that results in a balanced distribution of work where the new partitioning has a minimum amount of communication. Once a new partitioning is established, variables are transferred to their new owning partition and data structures are rebuilt.

The repartitioning is implemented in a manner such that computation and communication are overlapped as much as possible. This is accomplished using the non-blocking asynchronous MPI_ISend and MPI_IRecv operations [75]. This asynchronous point-to-point communication routines return immediately after a buffer is off-loaded to the communication hardware instead of blocking until the communication is completed. This allows the algorithm to continue computing while the communication is occurring in the background. The MPI standard requires that the memory of the communication buffer must not be accessed until the communication has been completed. The status of the communication can be checked using the MPI_Waitall or MPI_Waitany operations. MPI_Waitall is a blocking routine that only returns when all of the messages indicated for testing have completed. The MPI_Waitany operation returns when any of the messages has completed. This allows each message to be received as it is ready and the received data to be processed. As a significant amount of computational work such as octree construction can be accomplished with each of the received messages, the current work waits for messages to complete using the MPI_Waitany operation. Additionally, many MPI implementations utilize message size-based delivery protocols such as the eager and rendezvous protocols. The eager protocol assumes that a receiving process has the capability to receive a small message whenever it receives it, and therefore the message is sent without checking if the receiving process is ready. The rendezvous protocol explicitly checks that the receiving process is ready for the message, and thus has additional overhead. To utilize the knowledge that small messages will be most likely arrive faster than long messages, the current repartitioning algorithm sends large messages as soon as possible so that they are completed by the time the algorithm requires the data contained in any particular message.

The current algorithm for accomplishing the re-partitioning task can be summarized as:

Given the initial distribution of a refined mesh:

1. Estimate the amount of work associated with each base mesh cell that is locally owned in the old partitioning. Currently, this is taken to be the total number of multiplications for all leaf cells contained in the base mesh cell for the latest Poisson solve.
2. Compute a new partitioning of the base mesh using ParMETIS, where the work estimate computed in Step 1 is used as the balancing parameter.
3. Based on the new partitioning returned in Step 2, construct a communication pattern with processes for which data needs to be sent or received. Currently repartitioning partners are constructed using MPI_ALLTOALL.

4. For each base cell in the old partition, determine whether it is to be kept on the new partition or moved to a repartitioning partner. Store these in lists by partitioning partner.
5. For each base cell to be sent to a new process, store the global base index and the minimum set of leaf cells contained in the base cell for which the total refined mesh in the base mesh cell can be reconstructed. Communicate these values to the partitioning partners.
6. For each kept cell, store its new ghost base cells.
7. For each base cell to be sent, store its new ghost layer and communicate to the new owning process.
8. Pack and start communicating flow variables for all leaf cells which are to be moved.
9. Delete the mesh in all base cells that are being moved off-process.
10. Create two temporary base mesh data structures. The first holds the base mesh cells kept from the previous partition, while the second holds base mesh cells that are newly assigned to the partition.
11. Reconstruct the refined mesh in newly received base mesh cells.
12. Reconstruct the hash-table based global indexing for the base mesh such that the global indexing is again continuous on each partition.
13. Finish communication started in Step 8, and unpack variables on their destination partitions.
14. Pack and start communicating variables stored in array format.

15. Receive the communication started in 7 and reconstruct the base cell ghost layer.
16. Duplicate the refined mesh in the base cell ghost layer.
17. Create the permanent base mesh data structure which is composed of the base mesh kept from the previous partition, newly assigned base mesh, and the base ghost layer.
18. Finish the communications started in Step 12 and unpack variables into the newly sized arrays.
19. Create a new pointer array to the leaf cells on the new partition for efficient access in the flow computations.
20. Reconstruct new global leaf cell indices.
21. Repartition the level set fields.
22. Clean up all partitioning data structures.

3.3.4 Parallel Mesh Trimming

In section 3.3.3, the adaptive repartitioning algorithm was presented. As part of that algorithm, the base mesh of the original partition was broken in to two disjoint pieces: the cells that are kept on the process and those to be moved. The cells are to be kept are temporarily stored in a secondary data structure, and those destined for another partition are packed and then their octree structures are destroyed. The form of the data structures and operations performed in the adaptive repartitioning algorithm make it a simple matter to permanently delete a base cell from the computation. To accomplish the trimming operation, a cell that is to be pruned is simply not included in either of the sets of cells to keep or move, but it is added to the list of cells to destroy.

In the current implementation, only cells located greater than a specified distance interior to a stationary solid are eligible for trimming. The trimming operation occurs during the initialization of the octree mesh in two steps. First, during the initial construction of the base mesh, base cells further than a certain distance inside a stationary solid object are flagged for removal. Second, during the iterative initial mesh refinement phase to be presented in section 3.5.4, base cells that are interior to the stationary solid which contain no leaf cells of the currently most refined level are deleted.

The trimming procedure is depicted in a simple case in Figure 26. Here, a Cartesian mesh which is distributed amongst four partitions is iteratively refined such that the refinement occurs in a circular pattern. The starting base mesh and initial partitioning is shown in Figure 26(a). The dashed circle indicates the region of the original Cartesian mesh that is eligible for pruning. Figure 26(b) shows the Cartesian grid after pruning. The mesh is then iteratively refined in Figure 26(c)-(e). In Figure 26(c)-(d), all remaining base cells which are inside the circle contain mesh at the current most refined level so no additional pruning is possible in these steps. In Figure 26(e), the mesh has been updated to a state such that a set of base cells interior to the circle do not contain mesh at the finest level. These cells are subsequently removed from the mesh.

3.4 Simulation Initialization

The current initialization strategy is carefully implemented in a manner that introduces no restrictions on the size of the computational grid or on the number of processes that can be used during the simulation. The strategy is formulated such that no difficulties are encountered in the initial construction of mesh refinement patterns for geometries that change rapidly with respect to the base mesh size but smoothly with respect to the refined mesh grid spacing. In the following subsections, each major aspect of the initialization of the simulation is discussed.

3.4.1 Input of Parameters

The input of parameters is performed using a read-broadcast approach. A root process opens and reads parameters from a set of input files. The parameters are then packed into a long character buffer and broadcast to all other processes using a small number of global broadcasts. This approach is utilized for two reasons. Most importantly, parallel input/output (IO) is a complicated and expensive operation. A read-broadcast approach is much more efficient, especially on large distributed machines where the ratio of compute cores to hard disks is large. Less importantly, the current approach precludes the requirement of storing a different set of input files for each process to read, in order to avoid IO contention between processes. This leads to a simpler procedure for setting up and running simulations on high-end machines.

3.4.2 Base Mesh Initialization

If the initial base mesh distribution is not constructed in a careful manner, the computational work load of the initial mesh refinement may not be well balanced between processors which may induce unacceptable set-up times in the best case and more severely, memory bottlenecks and simulation termination. The initial mesh refinement pattern is determined solely by the level set fields of the embedded geometries. However, initial level set representations cannot be constructed until a base mesh has been established. The current strategy uses a four-step setup approach to circumvent this difficulty. The result of this procedure is an acceptable partitioning for the initial mesh refinement phase.

In a first step, the natural ordering of the NB_{global} global base mesh grid points is partitioned between the N_P MPI processes by assigning the first NB_{global}/N_P grid points to process 0, the next NB_{global}/N_P to process 1, and so forth. Due to the structure of the Cartesian mesh, it is a simple matter to compute which section of the base mesh each partition initially owns as well as the number of grid points that every other partition contains.

Next, a level set field is constructed over the entire base mesh on the initial partitioning. This level set field is used as an indicator which is used to specify the base cells that are to be removed during the first application of the mesh trimming operation. Grid cells that are deeper than a specified distance inside a stationary boundary are flagged for deletion. The flagged cells are deleted and the mesh structure and ghost layer are rebuilt.

The level set field is regenerated on the trimmed domain in order to compute an estimate of the initial refinement location and load. This information is used to specify a set of partitioning weights that are supplied to the domain partitioning algorithm, ParMETIS [82]. The graph partitioning algorithm returns a new partitioning of the base mesh that is both load-balanced and minimizes the amount of communication between the new partitions. The mesh is reassigned according to the updated partitioning, and the level set field is reconstructed. The simulation then proceeds to carry out the initial mesh refinement.

3.4.3 Initial Mesh Refinement

The initial mesh refinement occurs only adjacent to the interface where the level set value is smaller than some prescribed value. The goal of this step is to provide a minimal mesh refinement that accurately captures the shape of the geometry. Because the geometry may have regions that change rapidly compared to the base mesh, refining directly from the base mesh to the finest level is not optimal because this introduces large amounts of extraneous mesh to ensure that the geometry is accurately captured within the refinement region. To avoid this complication, the current work utilizes an iterative scheme to construct the initial refinement pattern.

The refinement procedure starts from a level set field constructed on the pruned base mesh. Given this level set field, a set of base mesh cells that have a level set value less than some desired cut-off can be probed to determine if they require refinement. The probing procedure is implemented using a Taylor series expansion of the base mesh level set field to compute an estimate of what the level set value would be at each of the base cell's potential children. If any potential child of the base mesh cell is found to have an estimated level set value less than the desired cut-off at the first level of refinement the base cell is subdivided. Once all base cells have been probed, the newly refined mesh is re-partitioned using the algorithm described in Section 3.3.3. After re-partitioning, the level set field is reconstructed on the set of cells that were constructed during the probing operation. If the maximum number of levels of refinement is greater than one, the procedure is repeated until the maximum number of refinement levels is reached.

3.5 Ghost Layer Construction

In the current approach, a partition may have an arbitrary number of irregularly shaped neighboring partitions with which it must exchange data. This makes efficiently establishing ghost communication patterns a challenging task if no information is available to narrow the set of partitions that may potentially contain future ghost cells.

One may attempt to bound each partition with a bounding shape, and detect collisions between partition bounding boxes. While guaranteed to work, there are three drawbacks to this naive algorithm. First, in order to check collisions, each partition must have access to the bounding coordinates of all other partitions. This can only be effected using an MPI_Alltoall communication which inhibits parallel scalability at large core counts. Second, the irregular nature of the partition shapes makes it difficult to tightly bound each partition, especially when the partition is composed of two or more disjoint pieces located at arbitrary locations in the solution domain. Using a simple bounding shape such as a single sphere or cube generates a large number of false positive collisions, which in the worst case leads to a partition checking every other partition for ghost cells. Finally, the bounding shape algorithm only provides information on which neighboring partitions a partition must communicate with. It does not specify which ghost cells those partitions contain. To establish this information, the ghost layer must be communicated to all potential owners who must return which of the cells they actually own.

To overcome the difficulty in determining ghost communication partners, the current algorithm builds ghost layers from an assumed initial condition. Recall that the base mesh is initially partitioned by assigning equal sized chunks to each partition, where each chunk contained a set of cells with sequential natural numbering. This information is used to construct the initial ghost layer. To do this, each cell in the initial partitioning is queried for any ghosts that are required to complete discretization stencils. These ghosts are added to a hash table which allows the same ghost to be entered multiple times without consuming additional memory or artificially increasing the total number of ghost cells required. A list of ghost cells is generated from the hash table and the partition where the cells are located is easily obtained from the initial cell distribution counts, which every partition can generate without communication.

During the repartitioning process ghost layers must be reconstructed. If the new partition mapping computed by ParMETIS is available in the ghost layer before partitioning, then any cell to be moved to another partition can be checked for any potential future ghost cells by scanning the partition mapping of its neighbors. If a set of ghost cells are identified, they are communicated to the future owning process along with the partition number that contains the ghost cell. After a partition receives the lists of its future ghost cells from each process that will be sending cells, the partition can form a concatenated list of ghost cells which surround the portion of the domain recently moved on process. A similar list of ghost cells surrounding cells retained on the partition can be formed, and the complete ghost layer is easily formed. Hash tables are again used to simplify the operations of assembling the ghost layer.

3.6 Parallel Level Set Construction

In its most basic form, construction of level sets in parallel is a straightforward task. A partial differential equation is solved on a Cartesian grid which requires a simple static communication pattern. However, in order to increase efficiency of level set computations, the current work utilizes a narrow band approach where the level set field is only stored in a tube of grid points which moves with the interface, and which is wide enough to hold the level set contours of a prescribed range instead of over the entire flow mesh. For example, consider the canonical test case of a cylinder with a notch cut in it, for which the geometry is shown in Figure 27(a). The level set field for this geometry is shown in Figure 27(b). It is clear from this figure that the level set field is only computed and stored on a small subset of the Cartesian grid in the immediate vicinity of the interface. This is shown even more clearly in Figure 27(c), which depicts the Cartesian grid nodes that are contained in the level set tube.

The tube variables include a set of mappings between the tube ordering and the flow mesh numbering, which will be called the tube indices, as well as the level set value, the level set normal vector, the level set curvature, etc. These are referred to as the tube field variables. Each level set is generally located on a subset of the total number of processes, but level sets may enter new partitions during either the movement of the level set or the mesh repartitioning process. The distributed nature of the level set complicates the implementation of the narrowband method tremendously. Consider the Zalesak problem. Figure 28(a)-(d) show the evolution of the Zalesak level set distributed across 4 partitions. The figures on the left side depict the evolution of the level set field, while the figures on the right show the distribution of the level set tube across the 4 processes. This case demonstrates all the possible parallel scenarios. For example, the tube smoothly enters multiple new processes simultaneously as well as leaves and re-enters partitions.

3.6.1 Parallel Level Set Tube Reconstruction

In the current work, the level set tube is implemented in a three-tiered structure. The first tier is the set of all level set tube indices such that a neighboring cell has a level set value of opposite sign. The second tier is the set of grid nodes which are within a distance of $6\Delta x$ of the interface, where Δx is a representative grid spacing in the level set tube. The third tier is the union of the second tier and one halo layer of adjacent cells. The level set advection equation is solved on the set of points in the second tier. After the advection of the level set field, it may be required to reconstruct the tube. To this end, the level set re-initialization equation is solved to extend the level set field into the edge of the third tier points of the original tube. The tube is then reconstructed by identifying which of the points in the original tube satisfy the properties of the first two tiers. The additional set of halo cells are then found by scanning the edge of the new set of points constituting the second tier. This implementation strategy is possible due to the CFL restriction on the allowable distance the level set field may move during one step.

The halo cells are the key to smoothly introducing a level set tube onto partitions where it did not previously exist but must enter due to advection of the interface. If the tube is reconstructed on a CFL-based timescale, the only way that a new tube can be introduced into a partition is if a tube halo cell moves into the ghost layer of another partition. This information is communicated between neighboring partitions during tube construction. The neighboring partition then simply checks if it has already identified the point as a tube point. If it has not, the point is added to the neighbor partition's tube.

3.6.2 Parallel Level Set Repartitioning

During the following presentation it is assumed that the level set tube will be repartitioned simultaneously with the rest of the flow variables, such that a repartitioning pattern of the flow mesh can be used to extract repartitioning patterns for the level set tubes. The communication of variables is set up using asynchronous non-blocking operations which make it possible to overlap communication and computation if the system hardware makes such support available. The algorithm for repartitioning the level set tubes can now be summarized as:

1. Given the flow mesh repartition list of grid cells that are to be moved to another process, create counts for each level set tube of the number of cells in the tube

that are to be moved off-process. Communicate these counts between the repartitioning partner pairs in the flow mesh repartitioning communication pattern.

2. Based on the counts in step 1, construct both send and receive level set repartitioning data structures that contain:

    a. The number of repartitioning partners for which a non-zero number of tubes will be exchanged.

    b. A list of MPI communication status indicators.

    c. A list of data structures to hold information about data to be moved to each repartitioning partner. This data structure holds:

        i. The partner's rank in MPI_COMM_WORLD.

        ii. Buffers for sending aggregated messages for the union of all tube points to be moved to the partner for communicating the level set field variables such as the level set value, normal vector, and curvature in one message for real variables and one message for integer variables.

        iii. The number of tubes being moved to or from the partner

        iv. A list of data structures to hold information about what indices to move for each tube to the partner.

3. Pack and start asynchronous non-blocking sends of indices required to reconstruct the tube on the new set of partitions that will own it.

4. Pack and start asynchronous non-blocking sends of the field variables stored in the buffers mentioned in step 2.c.ii.

5. Receive the messages from step 3 as they arrive and reconstruct the index mappings for the tubes received.

6. Re-allocate the tube field variable arrays to the correct size and store the values that are kept from the previous partition.

7. Receive the messages from step 4 as they arrive and store the received field variable values in the newly sized arrays.

8. Destroy the level set repartitioning data structures.

Because the level set repartitioning pattern is extracted from the mesh repartitioning pattern, determining communication partners only requires the use of point to point communication patterns. As a level set tube is a small subset of the overall flow mesh, the amount of data that is moved during this procedure is also small, so message aggregation is used to minimize latency penalties.

3.7 Time-Advancement Algorithm

Now that the components of the current algorithm have been established, the overall time-advancement scheme can be developed. Recalling that the governing equations are computed using a four-step fractional step method, the operations that compose the main time loop are the update of moving geometries, solving the momentum equation, removing the predicted pressure gradient, solving a Poisson equation for pressure, and adaptive refinement of the computational mesh. In order to prevent the introduction of a divergence into the velocity field when the mesh is refined, mesh refinement is completed before the pressure is computed. Because the intermediate velocity is an $O(\Delta t)$ approximation to the final velocity, the intermediate velocity is an adequate indicator of where refinement is required.

For convenience, the time advancement of the computation is summarized in the following set of steps:

1. Move geometries to their positions at time tn+1 and compute the new level set fields.

2. Compute the first provisional velocity field from the first step of the four-step fractional step algorithm.

3. Use the velocity field computed in Step 1 to compute the provisional vorticity and velocity gradient fields. Use the fields as indicator functions for refinement and coarsening in the LMR algorithm.

4. Apply the LMR algorithm with dynamic repartitioning if requested.

5. Remove the pressure gradient used in Step 2.

6. Communicate the ghost values of the second provisional velocity.

7. Compute intermediate face velocities.

8. Compute the pressure from the second step of the fractional step algorithm.

9. Project the velocity field onto its final value.

10. Communicate the ghost values of the final velocity field.

11. Return to step 1.

3.8 Discretization of Operators on Octree Meshes

The utilization of the octree mesh structure introduces the need for more complicated discretization schemes for cells that are located at mesh interfaces. In the current work a finite-volume scheme is used for discretization of cells located at mesh interfaces in order to ensure mass conservation. Due to the enforcement of the 2:1 balancing constraint, it is only possible for a neighbor of a cell located across a mesh interface to fall into two possible categories: i) the neighbor has children one level finer than the cell, or ii) the neighbor is one level coarser than the present cell. In order to maintain a compact and symmetric discretization of the Poisson equation, the convection and Laplacian terms are treated with different approaches at mesh interfaces. The details of the particular implementation for each term are now presented.

3.8.1 Treatment of the Laplacian Term at Mesh Interfaces

At a mesh interface, the Laplacian term is discretized in finite-volume form using a second-order integration scheme:

$$\int \nabla^2 \psi dV \cong (f_e - f_w)\Delta y_P + (f_n - f_s)\Delta x_P, \tag{3.1}$$

where f_e, f_w, f_n, and f_s are the fluxes at the east, west, north, and south faces of the cell, and Δx_P and Δy_P are the grid spacings. For faces where both the cell and the neighbor are at the same level, the flux is computed according to a central difference approximation. For example, in Figure 29(a), grid nodes N and S are at the same level as P. The fluxes at these faces can be approximated using central differencing:

$$f_N = \frac{\psi_N - \psi_P}{\Delta y_P}, \tag{3.2}$$

$$f_S = \frac{\psi_P - \psi_S}{\Delta y_P}. \tag{3.3}$$

In directions where the neighbor is coarser, a special form of the flux is required. In the current work, a symmetric 2nd-order discretization proposed by Lossaso [65] has been implemented. In Figure 29(a), the east neighbor of P is one level coarser. In this case, the flux at the east face is approximated using the value at E and the arithmetic average of the values at the cells which compose E’s west face, which in this case are P and S. Therefore, the east flux of cell P is

$$f_e = \frac{\psi_E - 0.5(\psi_P + \psi_S)}{1.5\Delta x_P}. \tag{3.4}$$

Now consider the west neighbor, W, in Figure 29(a) which is divided. In this case, the flux at the west face of cell P is computed as

$$f_w = 0.5(f_{w1} + f_{w2}), \tag{3.5}$$

where $f_{w1}$and f_{w2} are computed as the east fluxes of W_1 and W_2 according to equations similar to (3.4). However, in the present approach, f_{w1} and f_{w2} will have the exact same stencil, and so the west flux is

$$f_w = \frac{\psi_P - 0.5(\psi_{W1} + \psi_{W2})}{1.5\Delta x_{W1}} = \frac{\psi_P - 0.5(\psi_{W1} + \psi_{W2})}{3/4\Delta x_P}. \tag{3.6}$$

The overall discretization has the final form

$$\int \nabla^2 \psi dV \cong a_S\psi_S + a_{W1}\psi_{W1} + a_{W2}\psi_{W2} + a_P\psi_P + a_E\psi_E + a_N\psi_N, \tag{3.7}$$

with discretization coefficients

$$a_S = \frac{\Delta x}{\Delta y} - \frac{\Delta y}{3\Delta x}, a_{W1} = \frac{2\Delta y}{3\Delta x}, a_{W2} = \frac{2\Delta y}{3\Delta x}, a_E = \frac{2\Delta y}{3\Delta x}, a_N = \frac{\Delta x}{\Delta y}, a_P = -\left(\frac{2\Delta x}{\Delta y} + \frac{5\Delta y}{3\Delta x}\right). \tag{3.8}$$

This discretization of the Laplacian operator is much more compact than the discretization applied to the convection term as will be seen in Section 3.9.2. This is important for two reasons. First, the compact stencil leads to better conditioning of the coefficient matrix, which enhances the convergence rate of the solution of the Poisson equation in the fractional step algorithm. Second, the amount of data that is required to be communicated during the application of iterative solvers is much lower than would be required for extended discretization schemes.

3.8.2 Treatment of the Convection Term at Mesh Interfaces

The explicit time discretization of the convection term provides more freedom in choosing a numerical scheme because the consequence of the discretization on the convergence rate of matrix problems does not need to be considered. In the current work, convective fluxes are computed using a conservative bilinear interpolation scheme [86]. Again, the starting point for the numerical scheme is the finite-volume form of the convective term:

$$\int \nabla \cdot (\vec{u}\psi) dV \cong (f_e - f_w)\Delta y_P + (f_n - f_s)\Delta x_P. \tag{3.9}$$

In this equation, f_e, f_w, f_n, and f_s are the convective fluxes at the faces, which have the form $f_i = \psi_i(\vec{u} \cdot \vec{n})_i$ at the ith face. Here, ψ_i is an interpolation of ψ to the face, and $(\vec{u} \cdot \vec{n})_i$is a mass-conserving face-velocity.

Consider the case in Figure 29(b). In this case, the neighbors N and S are at the same level as P and they are not divided. The convective fluxes are therefore computed with standard central differences:

$$f_n = \frac{1}{2}(\psi_N + \psi_P)v_n, \tag{3.10}$$

$$f_s = \frac{1}{2}(\psi_S + \psi_P)v_s. \tag{3.11}$$

Here, v_n and v_s are the normal velocities at the north and south faces. Now, consider the east face, where the neighbor E is at a coarser level. In this case, direct application of central differencing is not possible. To handle this, a fictitious point, E*, is placed where the location of the neighbor's cell center would be located if both cells were at the same level. The value of ψ at E*, ψ_{E^*}, is then interpolated from the surrounding cells using a conservative bilinear interpolation. In the case of Figure 29(b), the bilinear interpolation has the form

$$\psi_{E^*} = 0.5625\psi_E + 0.1875\psi_{P^*} + 0.1875\psi_{NE} + 0.0625\psi_{N^*}, \tag{3.12}$$

Because ψ does not exist at P* and N*, bilinear interpolations are required to assign values to these locations as well,

$$\psi_{P^*} = \frac{1}{4}(\psi_P + \psi_W + \psi_S + \psi_{SE}), \tag{3.13}$$

$$\psi_{N^*} = \frac{1}{4}(\psi_N + \psi_{NW} + \psi_{NN} + \psi_{NNW}). \tag{3.14}$$

Substituting (3.13) and (3.14) into (3.12) results in

$$\psi_{E^*} = 0.046875\psi_P + 0.5625\psi_E + 0.046875\psi_W + 0.015625\psi_N + 0.046875\psi_S + 0.1875\psi_{NE} + 0.015625\psi_{NW} + 0.046875\psi_{SE} + 0.015625\psi_{NN} + 0.015625\psi_{NNW}. \tag{3.15}$$

Once the value at ψ_{E^*}has been computed, the flux at the east face is now computed using a central difference according to

$$f_e = \frac{1}{2}(\psi_{E^*} + \psi_P)u_e \tag{3.16}$$

In order to evaluate the west flux of cell E in Figure 29(b), the east fluxes of P and S, $\mathrm{f_e^P}$ and $\mathrm{f_e^S}$, are constructed in the manner of equation (3.16). Then, for grid node E the west flux is

$$f_w^E \Delta y_P = f_e^P \Delta y_P + f_e^S \Delta y_S. \tag{3.17}$$

This definition conserves mass explicitly at mesh interfaces.

3.8.3 Velocity Correction at Mesh Interfaces

In the current work, a hybrid variable arrangement is used where the velocity vector and pressure are stored at the cell center while a set of divergence-free normal velocities are stored at the cell faces. After the solution of the momentum equation in the fractional step method, an intermediate face velocity field, u_f^*, is constructed by linear interpolation from the intermediate velocity at cell centers. In the case of computing face velocities at mesh interfaces, this procedure is applied in manner similar to that described in section 3.8.2. For directions where no mesh interface exists, the value of the intermediate velocity is linearly interpolated from the velocity at the cell and its neighbor. In the case where a coarser neighbor exists, the interpolation scheme of Section 3.8.2 is used to evaluate the intermediate velocity at a starred point, and then a linear interpolation between the cell center value and the starred value to the face is computed. In the case where a neighbor is divided, the face velocity is set to be the area average of the face velocities of the two children that share the face.

The intermediate face velocity field is used to construct the source term for the Poisson equation according to

$$\int \nabla \cdot \vec{u}^* dV \cong \left(u_{f,e}^* - u_{f,w}^*\right)\Delta y_P + \left(v_{f,e}^* - v_{f,w}^*\right)\Delta x_P. \tag{3.18}$$

After the solution of the Poisson equation, the face velocities are corrected using the pressure gradients at the face while the cell center velocities are corrected using the average of the face pressure gradients. In the case where the neighbor is at the same level as the cell, standard central differencing about the face is used to compute the face pressure gradient. When the neighbor is coarser, the pressure gradient at the face is computed according to equation (3.6). In the case where a neighbor is divided, the face velocity is taken to be the area weighted average of the children’s corrected face velocities.

3.9 Parallel Linear Solvers

The efficiency of the current solver is highly dependent on the effectiveness of the methods used to compute the solutions of the linear systems associated with the first and second steps of the fractional step algorithm. Effective preconditioning can dramatically increase convergence by shifting and clustering the eigenvalues of the linear problem away from the zero eigenvalue [87]. Of particular importance is the construction of an efficient preconditioning technique for the Poisson equation contained in the fractional step algorithm, which accounts for almost 90% of the flow solver time.

In the current framework, the solutions of matrix problems are computed using iterative solvers contained in the Portable Extensible Toolkit for Scientific Computation (PETSc) [81]. PETSc is a numerical suite of Krylov-space solvers and preconditioning techniques that is built on the concept of command line driven composability of Krylov solvers and preconditioners. This makes it easy to test the effectiveness of a range of linear solvers for a particular problem without requiring any modification to source code. PETSc also provides access to many third-party preconditioner and sparse direct solver packages through the same command line approach. In this work, the most important preconditioning techniques are incomplete LU (ILU) factorizations and algebraic multigrid. The PETSc library contains ILU preconditioning methods, as well as an algebraic multigrid solver, GAMG. An important third-party package which has strong relevance to the current work is the HYPRE BoomerAMG algebraic multigrid solver [88, 89].

The non-symmetric character of the matrices that result from the current sharp-interface scheme considerably restricts the available options for Krylov solvers. Krylov methods that are applicable to non-symmetric matrices include: the minimal residual method (MINRES), the generalized minimal residual (GMRES) method, various formulations of the bi-conjugate gradient algorithm (BiCG, BiCGStab(*l*)), and the transpose-free quasi-minimal residual (TFQMR) method [87]. Numerical experimentation shows that the stabilized version of the bi-conjugate gradient method, BiCGStab(*l*), produces the most efficient convergence behavior across a range of problems.

The selection of an appropriate preconditioner is much more problem dependent. Due to the diagonal dominance of the linear system associated with the momentum equation, ILU preconditioning is sufficient. The preconditioner used in conjunction with the Poisson equation is a more critical choice. For unstructured meshes, algebraic multigrid methods are considered the state-of-the-art in preconditioning elliptic problems. However, the expensive setup phase of a multigrid algorithm may overwhelm the reduced solution time if the preconditioner must be reconstructed frequently. In the case of moving boundaries, the preconditioner must be rebuilt each time step. In this case, a less effective preconditioner may be more efficient due to less investment in setup costs. Also, in the case of geometric regions of small scale, convergence of multigrid methods may deteriorate drastically due to the inability of the multigrid algorithm to form acceptable aggregates.

In order to assess the behavior of each preconditioner when applied to matrices arising from the current discretization scheme, we solve the test Poisson problem given in Equation (2.49) on a set of grids of increasing size. The linear system associated with Equation (2.49) is solved with the BiCGStab(*l*) Krylov solver, where the matrix is preconditioned with ILU, GAMG, and HYPRE BoomerAMG in turn. To ensure that the order of the least-squares treatment utilized in the sharp-interface approach does not significantly affect solve times, Equation (2.49) is solved using both linear and quadratic least-squares fits.

The times required to solve the system for each of the preconditioners are shown in Figure 30. It is evident that both GAMG and HYPRE solve times grow linearly with problem size, whereas ILU preconditioning becomes less efficient as the system size increases. The HYPRE preconditioner is approximately twice as fast as the GAMG multigrid solver. The increased cost of GAMG is due to a more expensive setup phase. The preconditioner setup cost can be amortized in circumstances where the matrix does not change between solves. This is the case when no boundaries are in motion and mesh refinement has not occurred since the last solve. Additionally, this optimization can be utilized during a sub-iteration algorithm where the matrix does not change between sub-iterations. This can significantly decrease the total amount of time consumed by linear solves.

3.10 Scaling Study on Kraken

In this section, we assess the parallel scalability of the current numerical scheme by carrying out a set of computations on Kraken. Kraken is a Cray XT5 cluster operated through the National Institute of Computational Sciences at the University of Tennessee which is located at Oak Ridge National Laboratory. Kraken is composed of 9,408 compute nodes, where each node has two sockets. Each socket has a 2.6 GHz six-core AMD Opteron processor and 8 GB of memory. In total, Kraken has 112,896 compute cores, 147 TB of compute memory, and a peak performance of 1.17 PetaFLOP [90]. Each computational node is connected to a Cray SeaStar2+ router and the SeaStar2+ routers are connected in a 3-D torus topology.

The vendor tuned MPI library, MPT 5.3.5, was used in the current scalability study. MPT is an optimized Cray implementation of the MPICH MPI library. Several environment variables are available to tune the performance of the MPI functionality. The only environment variable altered in the current study was MPICH_FAST_MEMCPY, which enabled the use of an optimized memcpy MPI point-to-point and collective operations.

The fully adaptive algorithm was tested at large core counts by solving 50 time steps of flow past a sphere, where the mesh was adapted and re-partitioned every 5 time steps. The domain size was set to 15x15x50. The base mesh grid spacing was 0.1, and 3 levels of refinement were applied to the mesh. The Reynolds number was set to 1000, and a refinement criteria threshold of 0.01 was used to ensure that maximum refinement occurred where vorticity was located.

The simulation was performed on 1200, 2400, 4800, 9600, and 16512 computational cores. Because unstructured algorithms and sparse matrix solvers are bandwidth limited, using all the cores on a node will not lead to improved performance, as the memory bandwidth will saturate which starves the CPU of data. To mitigate this effect, only three threads were bound to each node socket such that each node supported 6 MPI tasks instead of 12. Allocating MPI tasks in this manner also doubled the amount of memory available to each process.

The wall clock times versus time step are shown for each of the job core count in Figure 31. For all cases, the wall clock time at time steps when mesh refinement and repartitioning are performed is approximately 2.5 times larger than the walk clock times of the time steps when refinement does not occur. Therefore, the refinement and repartitioning operation is only slightly more expensive than the flow solver. It is observed that the wall clock time of the refinement steps is consistently decreased up to a core count of 9600. There exists more variability in the reduction of run time with core count for the intermediate time steps, with some steps having a significant reduction in time between 4800 and 9600 cores, while other time steps do not. In Figure 32, the global grid size is shown for the 50 time steps of the simulation. The grid initially has approximately 20 million cells, and after subsequent refinements, has a final size of over 300 million cells. On 9600 cores, the flow solver requires approximately 9 seconds to solver for 300 million cells, while on 1200 cores, it requires between 30 and 60 seconds.

The total run time versus number of cores is shown in Figure 33. The straight solid line represents perfect scaling, where a doubling of cores reduces wall clock time by a factor of 2. It is observed that below 4800 cores, the scalability of the current application is quite good. Beyond 4800 cores parallel scalability starts to deviate significantly from the ideal case. However, even between 9600 and 16,512 cores, there is a decrease in walk clock time.

In Figure 34, the average number of grid cells per partition is shown for each core count. The average partition size at time step 1 is marked with triangles, while the average partition size at time step 50 is shown by squares. At startup, there are nearly 10,000 grid cells per process on 1200 cores, while on 16,512 cores, there are approximately 700 grid cells per process. At the termination of the simulation, the 1200 core job had an average number of cores of 250,000 cells, while the 16,512 core job had just over 18,000 cells. Therefore, the parallel scalability at the largest core counts may be suffering due to a lack of enough computational work to compensation for communication costs.

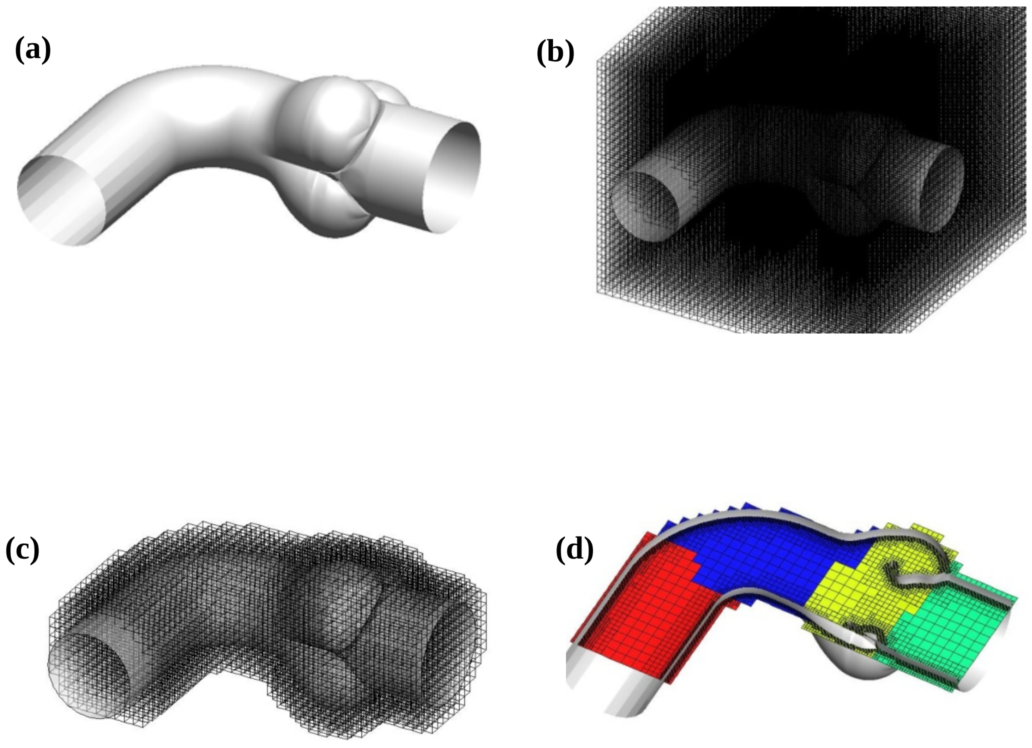


Figure 19 An example of the pELAFINT3D framework applied to an aortic valve geometry. The geometry is shown in panel (a). In (b), the geometry is enclosed in a bounding Cartesian grid. In (c), the Cartesian grid is trimmed to conform to the geometry. In (d), local mesh refinement is applied in the desired locations. The regions of (d) are colored by partition.

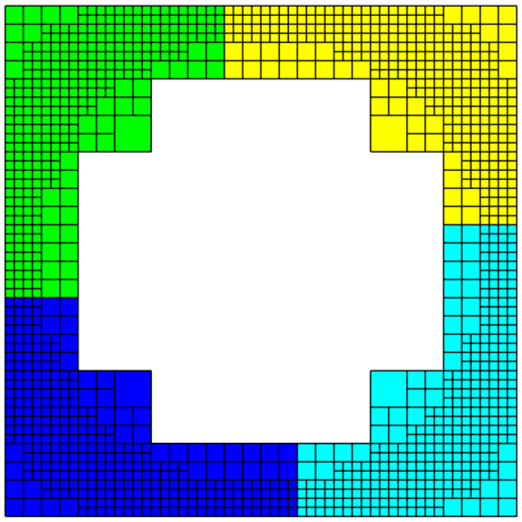


Figure 20 An example of a mesh generated by pELAFINT3D. The mesh is a Cartesian base grid partitioned between four MPI processes with two levels of quadtree refinement applied in a circular pattern.

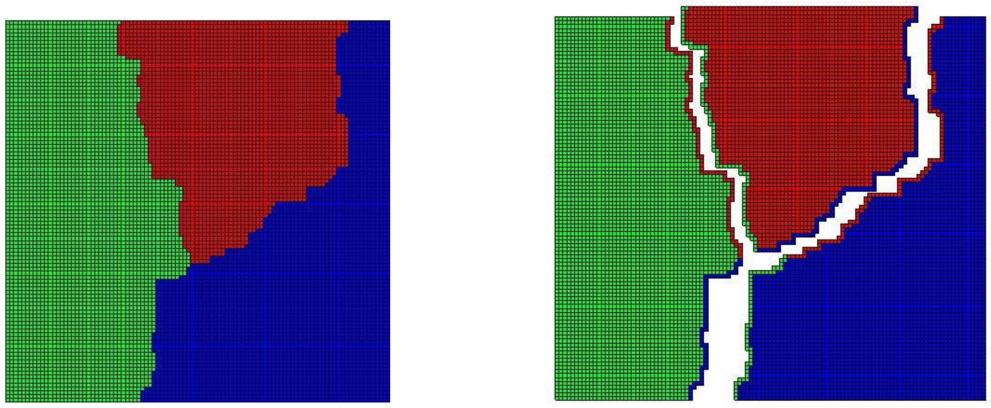
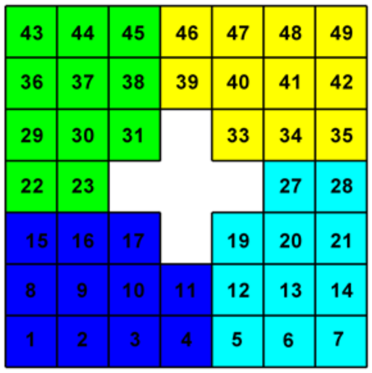
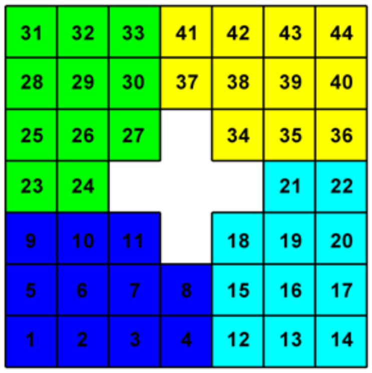


Figure 21 A depiction of a distributed base mesh to make the concept of a ghost layer clear. On the left, the base mesh is distributed between three partitions. On the right, the partitions are separated to show the layer of ghost cells around each partition edge. The ghost cells are duplicates of the set of cells that are adjacent to the cells at the edge of a partition. In this case, the blue cells at the edge of the green piece of the mesh are the neighbors on the blue partition that have information required by cells on the green partition.

(a)



(b)



(c)

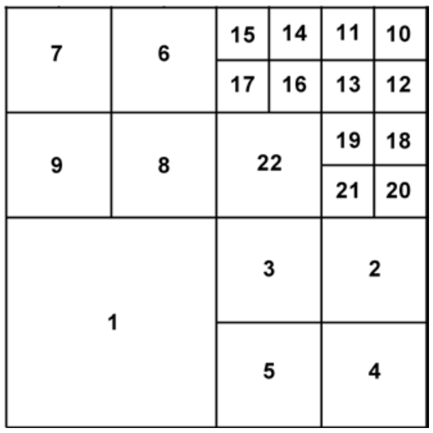


Figure 22 Schematics of the three numbering systems required in the current algorithm. In (a), the natural base mesh numbering is shown. This numbering is the standard Cartesian numbering system. It is not continuous on a partition. The application numbering of the base mesh is shown in (b). This numbering is continuous on a partition, and is such that the corresponding natural numbering of the cells is in sorted order. In (c), the application ordering of the refined mesh is shown. This scheme induces a partial z-ordering.

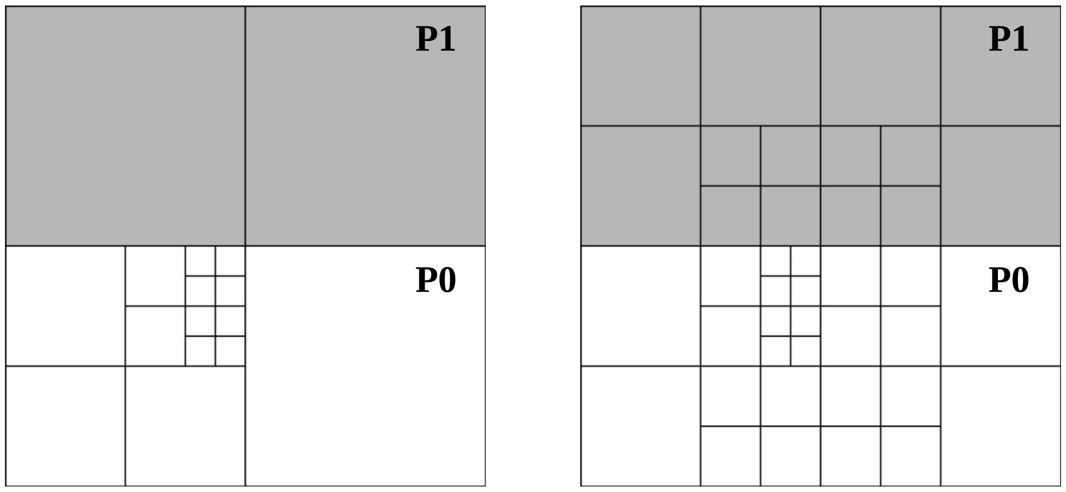


Figure 23 A quadtree which does not satisfy the 2:1 balance constraint (left) and the quadtree after balancing. The quadtree is distributed over two partitions, P0 and P1, and the imbalance must be resolved across the partition boundary.

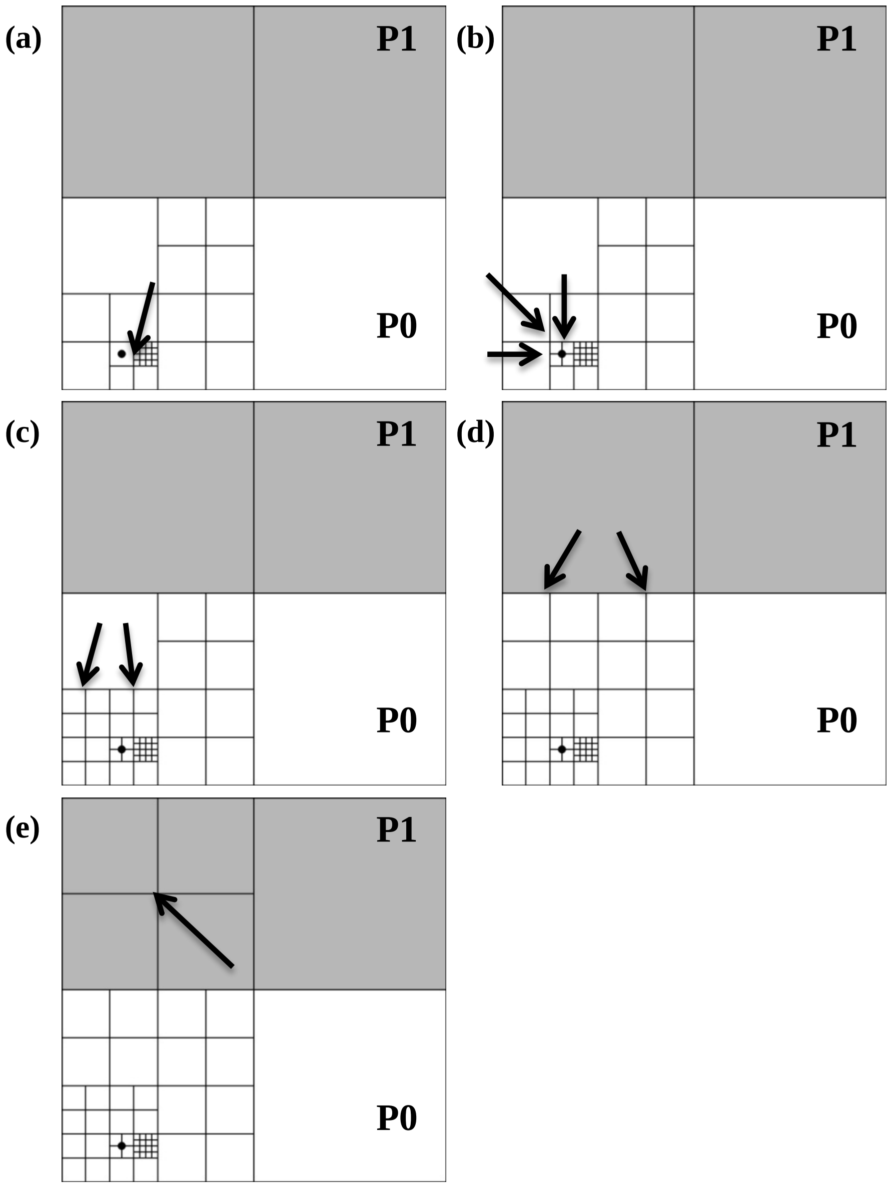


Figure 24 A depiction of the recursive behavior of a naïve 2:1 quadtree balancing algorithm in a two partition grid where the white region belongs to partition 1 and the grey is owned by partition 2. The black circle represents a cell for which imbalances are to be detected and resolved. In (a), an original imbalance exists between the cell and its east neighbor. In (b) and (c), the resolution of an imbalance leads to new imbalances for neighbor cells. In (d), the resolution of the imbalance formed in (c) introduces an imbalance across the partition boundary. In (e), the imbalance of the neighboring partition cell is resolved.

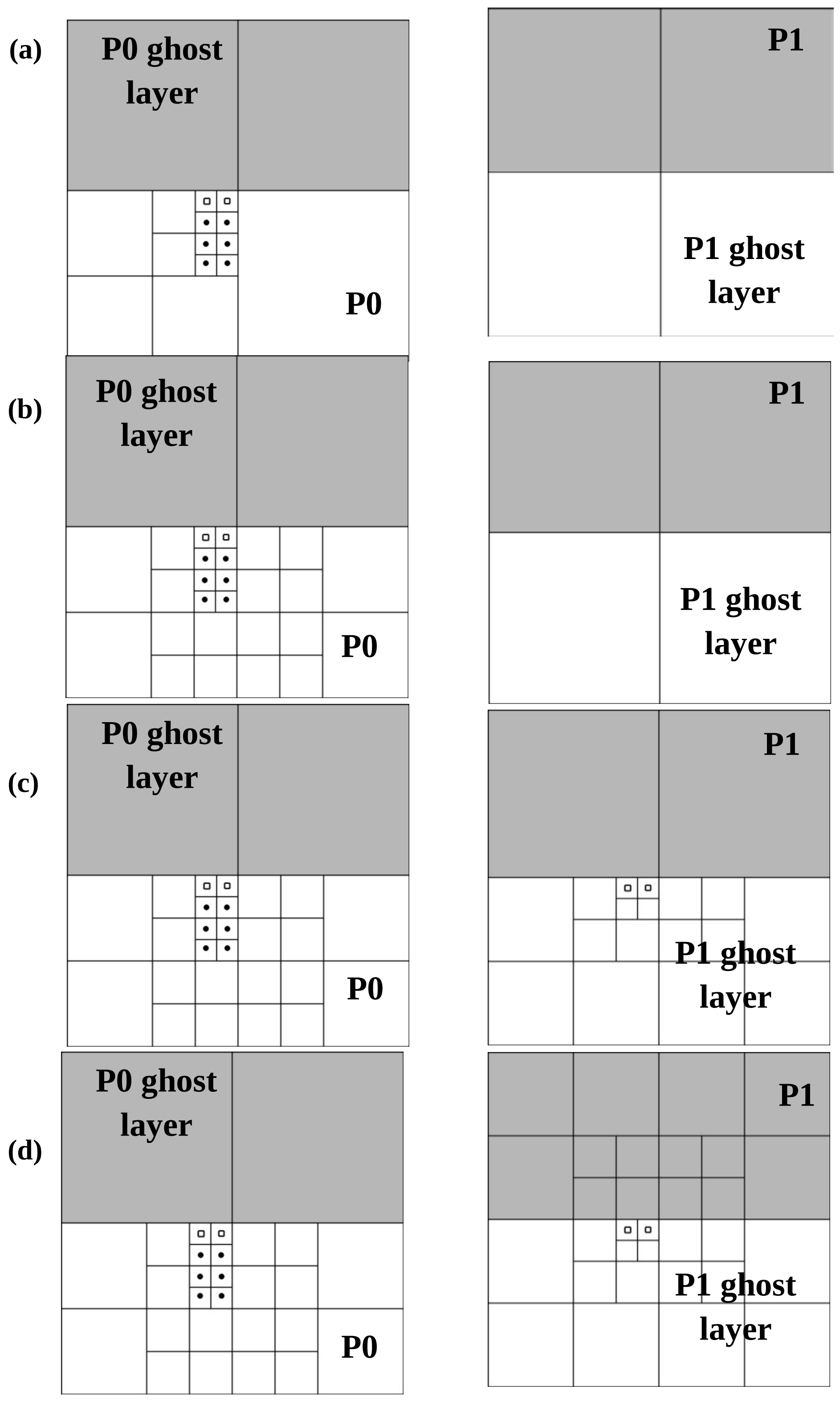


Figure 25 A depiction of the parallel Prioritized Ripple Propagation.

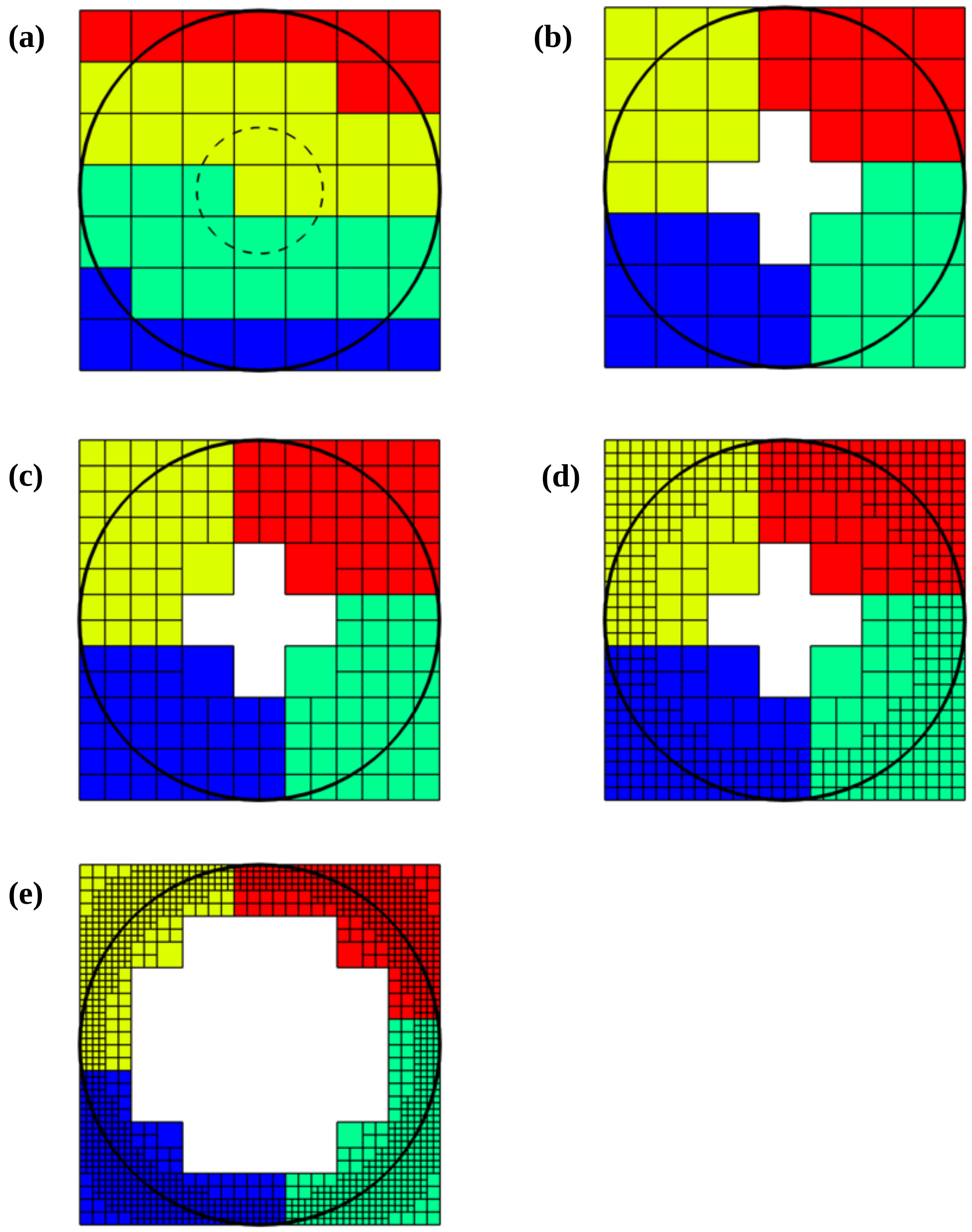


Figure 26 A schematic of the mesh initialization procedure. In (a), an initial base mesh is partitioned in vertical strips. In (b), a portion of the base mesh is pruned and the remaining mesh is re-partitioned. In (c)-(e), additional refinements occurs, and base cells that have no mesh at the finest level inside the circle are pruned away.

(a)

(b)

(c)

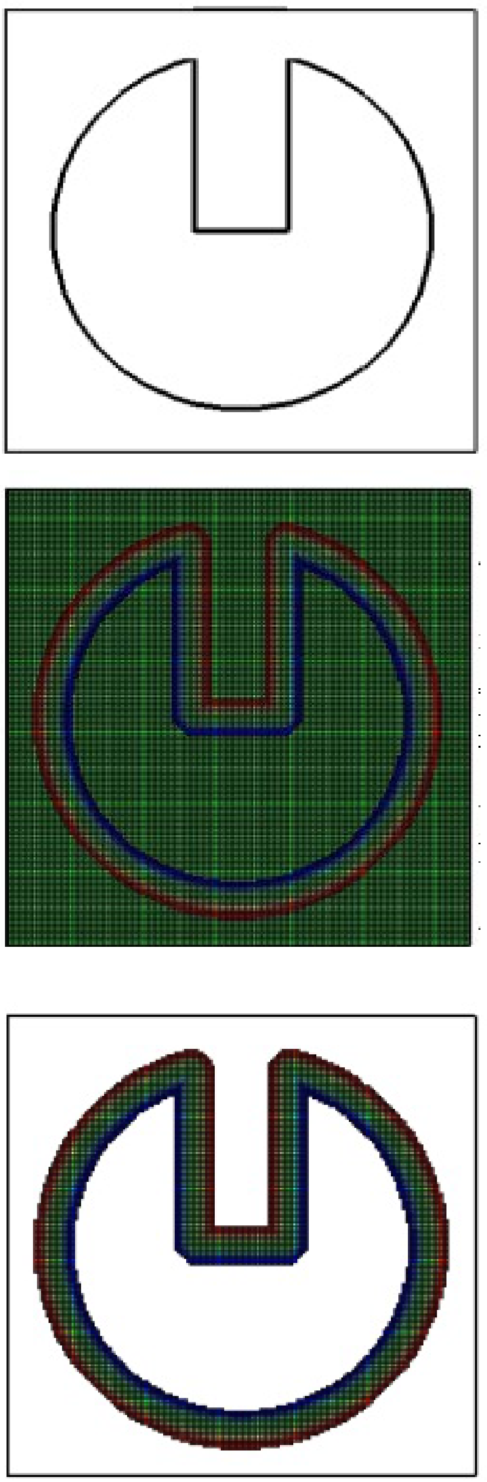


Figure 27 The initial level set field of the Zalesak disk problem. The interface shape is shown in (a). In (b), the associated level set field is shown, and (c) shows the region of the mesh that corresponds to the level set narrow band.

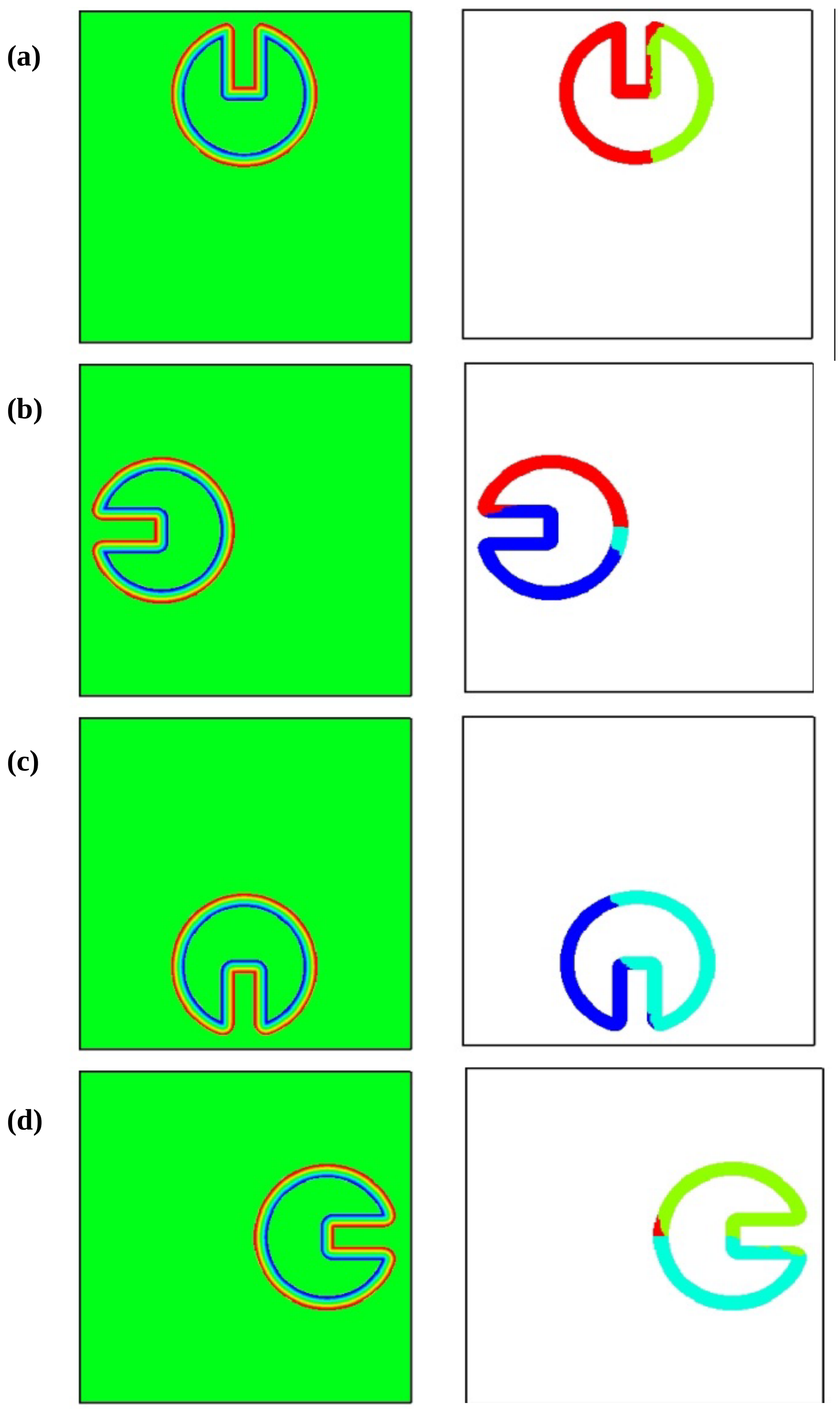
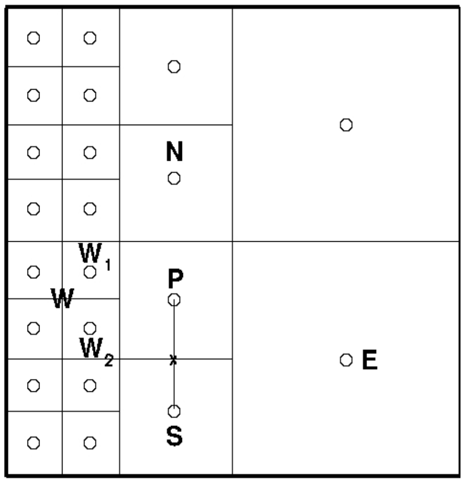


Figure 28 The evolution of the Zalesak disk level set field over one complete revolution. The figure shows the level set contours (left), and the narrow band colored by partition (right) at rotation angles of (a) 0 degrees, (b) 90 degrees, (c) 180 degrees, and (d) 270 degrees.

(a)



(b)

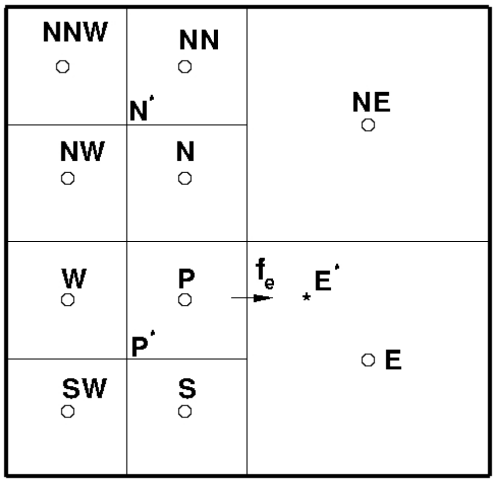


Figure 29 Configurations of the mesh used in the discussion of (a) the discretization of the Laplacian operator at mesh interfaces, and (b) the discretization of the convection operator at mesh interfaces.

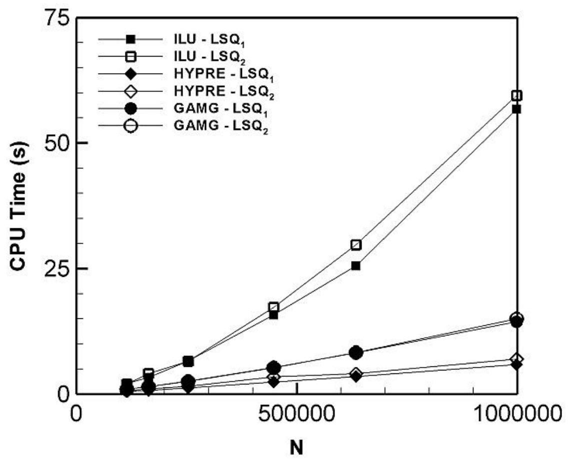


Figure 30 Time required in solving the Poisson problem specified by Equation (2.49) on grids of increasing size. The BiCGStab(*l*) Krylov solver is preconditioned by ILU, GAMG, and HYPRE. The sensitivity of solve time to the least-squares treatment applied in the sharp-interface method is tested by computing the time required for both linear and quadratic least-squares fits.

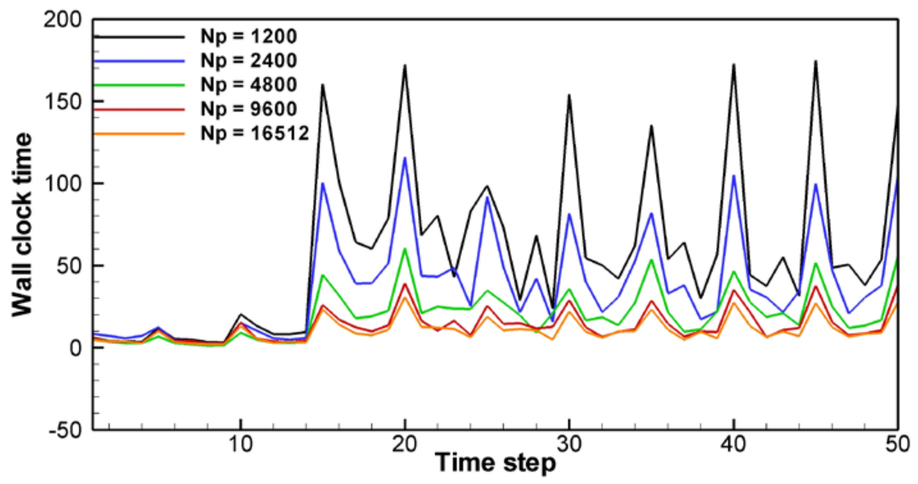


Figure 31 Wall clock time for the first 50 time steps of the computation of flow past a sphere with adaptive meshing and repartitioning every 5 time steps for cores counts of 1200, 2400, 4800, 9600, and 16512. The mesh refinement procedure is approximately the same computational cost as applying the flow solver. All parts of the algorithm reduce in time as the core count increases.

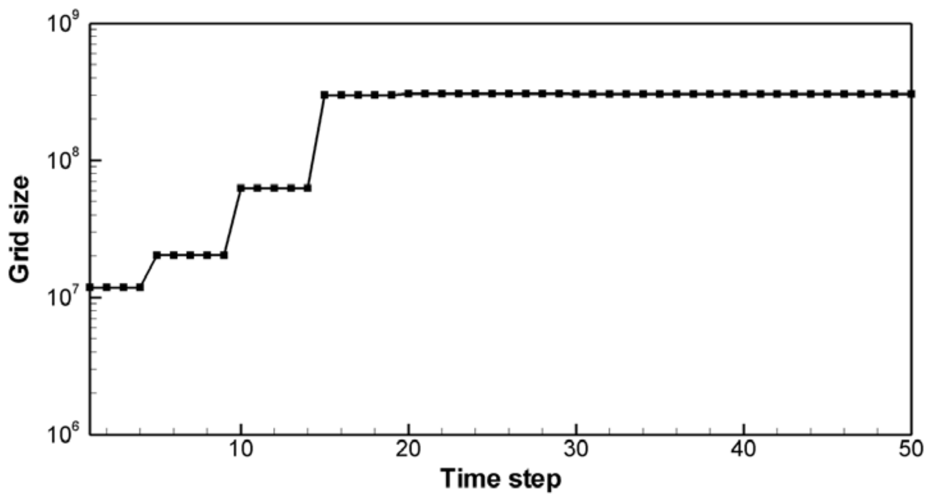


Figure 32 The grid size for the first 50 time steps of flow past a sphere. The grid starts at over 20 million grid points, and is refined to approximately 300 million grid points.

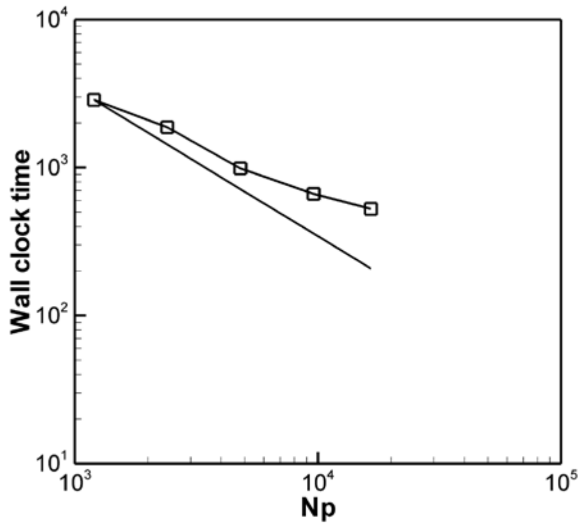


Figure 33 Wall clock time versus processor count for the solution of flow past a sphere on 1200, 2400, 4800, 9600, and 16512 cores. Ideal scaling is shown by the solid black line. The scalability up to 4800 cores shows very good agreement with perfect scaling. Core counts larger than 4800 show significant deviation from the ideal scaling.

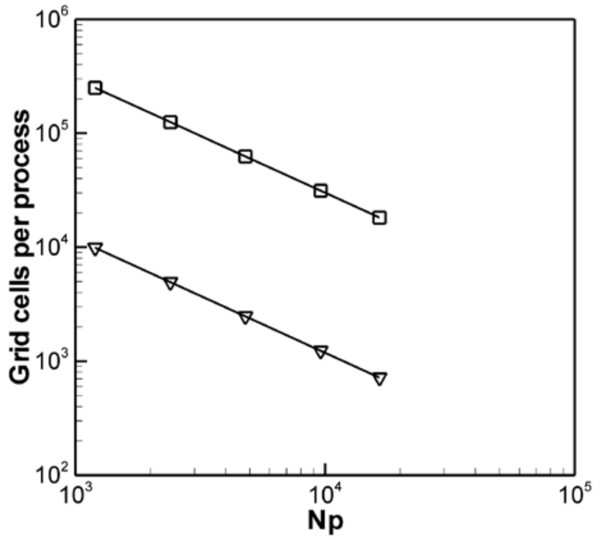


Figure 34 The average number of grid cells per processor for flow past a sphere on 1200, 2400, 4800, 9600, and 16512 cores. The average partition size at time step 1 is showne by triangles, and the average partition size at time step 50 is shown by squares.

CHAPTER 4

VALIDATING THE FLOW SOLVER

4.1 Introduction

In the current chapter, the computational capabilities of the solver are examined through simulations of problems for which benchmark solutions are available. The level set capabilities are shown to correctly capture the solution of two test problems from the literature; the straining of a circular level set in a vortex velocity field, and the rigid body rotation of a disk that contains a sharp notch. The ability of the flow solver to accurately capture the vorticity dynamics at solid surfaces at high Reynolds number values is confirmed by comparison with surface vorticity results for the impulsively started flow past a stationary cylinder at Reynolds number values of 1000 and 9500. To demonstrate that the current approach can handle a complex geometry with both concave and convex curvatures, the flow in a two-dimensional channel with two subsequent semi-circular constrictions is simulated and compared with experimental and computational results from the literature. In three-dimensions, the flow past a sphere at a set of Reynolds numbers is simulated, and the drag coefficient is shown to be in good agreement with previous studies. To highlight the ability of the mesh pruning strategy to make simulations in curved three-dimensional geometries feasible, the flow in a 90 degree bend is computed and compared with experimental results.

Moving boundary capabilities are validated in two-dimensions by re-producing the surface vorticity results of the impulsively started cylinder problem in a reference frame where the cylinder is moving while the velocity is quiescent in the far field for a

Reynolds number of 1000. To validate the three-dimensional moving boundary capabilities, the sedimentation of a sphere is simulated and the terminal drag is shown to match the drag on a stationary sphere in a flow of equal Reynolds number.

4.2 Deformation of a Circular Level Set by a Vortex Flow

In the current section, the evolution of an initially circular level set field by a vortex induced velocity field is examined. A circular level set field for which the zero contour had a radius of 0.15 was initialized with its origin located at (0.5,0.75) in a 1x1 domain. The level set was then subjected to the velocity field:

$$u(x,y) = -2\cos\pi y \sin\pi x \sin^2\pi x \tag{4.1}$$

$$v(x,y) = -2\cos\pi x \sin\pi y \sin^2\pi y \tag{4.2}$$

The solution is computed on a Cartesian mesh of with a grid spacing of 0.04. The Cartesian mesh is enriched with 3, 4, and 5 levels of refinement for fine mesh grid spacings of 0.005, 0.0025, and 0.00125, respectively, and is distributed across 4 MPI processes.

The level set field for the case of 5 levels of refinement is shown at T = 0, 1, 2, and 3 in Figure 35(a)-(d). The circular level set is observed to deform into a long filament with a thick head and thin tail. The partitioning of the level set tube across the MPI processes is shown in Figure 36. The level set field is computed smoothly across the time-varying partitioning of the level set tube, which smoothly adapts such that approximately the same numbers of tube points are owned by each of the 4 processes for each of the times shown in Figure 36. Generally, the level set tube will not be distributed across all processors, but instead only a small subset of the total number of the process pool. However, this does not introduce significant scalability issues, because the level set calculations are a small portion of the overall simulation time.

The representation of geometries by level set fields is known to suffer from mass conservation issues. The dissipative nature of the numerical scheme near regions of high curvature induces a monotonic decay of the object volume as the simulation proceeds. In the current work, the high-order HJ-WENO spatial discretization of the level set advection equation is augmented with octree mesh refinement in the vicinity of the zero level set contour. This enhancement significantly reduces the mass loss that is experienced when compared to the nominal WENO scheme. To demonstrate this, the level set solution for the three different levels of refinement is shown in Figure 37(a)-(b). In these figures, the red coloring represents the location of the level tube for the least refined case, and blue shows the level set tube for the most refined case. Figure 37(a) shows that mass loss is much more appreciable in the region of the filament tail where the zero level set starts to merge, while the object volume is more robust near the head as the grid becomes coarser (Figure 37(b)). As the mesh is refined, the level set field is capable of representing the thin tail through a longer period of deformation.

4.3 The Zalesak Disk Problem

In the current section, the behavior of the current level set technique near sharp edges is investigated by solving for the Zalesak disk problem [91]. The disk of radius 0.2 is originally centered at (0.5,0.75) in a 1x1 domain. The disk is then rotated once about the point (0.5,0.5) with a rigid body velocity of

$$u(x,y) = -2\pi(y-0.5) \tag{4.3}$$

$$v(x,y) = 2\pi(x-0.5) \tag{4.4}$$

The problem is solved on a Cartesian base mesh with a grid spacing of 0.04 with 4 levels of refinement for a grid spacing of 0.0025 in the level set tube. The WENO solution of this problem is already presented in Chapter 3. Here, the 5th-order WENO [64] solution is compared with the 4th-order ENO construction used in [72]. Both of the solutions are moved forward in time with 3rd-order TVD Runge-Kutta time-stepping. A comparison of the location of the zero level set contours for both discretization schemes is presented in Figure 38. From the figure, it is observed that the WENO construction, while being simpler and cheaper than ENO, maintains a more accurate solution near sharp corners than the more dissipative ENO scheme.

4.4 Impulsive Flow Past a Stationary Cylinder

To investigate the ability of the current scheme to accurately capture vorticity evolution at embedded surfaces, the solver is used to simulate the impulsive flow around a stationary two-dimensional circular cylinder. Koumousatkos and Leonard [92, 93] have studied the impulsive cylinder problem in the context of Lagrangian vortex methods and in doing so have accurately computed the drag and lift coefficient histories and temporal evolution of vorticity at the cylinder surface for Reynolds number values from 40 to 9500. Because the vorticity field of the impulsive problem displays a strong interaction between localized wake features and boundary vorticity, it provides a good bench mark case to demonstrate the ability of the current code to capture all relevant physics in an efficient manner.

To that end, simulations are performed at Reynolds numbers of 1000 and 9500. The simulation domain is set to a size of 30x30, and the cylinder has a diameter of 1 and is centered at (7.5,15). The potential flow solution is used to prescribe the initial and boundary conditions. In order to resolve the boundary layer as it forms along the cylinder surface, a minimum of 10 mesh points across the boundary layer thickness is generally necessary. An estimate for the average boundary layer thickness over the cylinder surface can be obtained from $\sim 1/\sqrt{Re}$. Therefore, the grid spacing is selected such that the most refined mesh spacing is $\Delta x = 1/10\sqrt{Re}$. Subsequently, the estimated required fine grid spacing for the Re = 1000 and Re = 9500 simulations are 0.00316 and 0.001025.

The Re = 1000 simulation was performed using a base mesh spacing of 0.2 with 6 levels of refinement to obtain a resultant finest mesh spacing of 0.003125. The simulation was run for 30,000 time steps at a constant time step size of 0.0001. The initial and final mesh sizes were approximately 46,000 and 200,000 cells, respectively. For the simulation with Re = 9500, a base mesh spacing of 0.08 is used where the mesh is enriched with 6 levels of refinement which results in a fine mesh spacing of 0.00125. Both simulations were computed on 16 processors.

Consider the Re = 1000 case. A qualitative comparison of the resultant vorticity fields of the current work and of is made in Figure 39. In Figure 40, the current surface vorticity results are compared with the published results for Re = 1000 at time instants T = 0.4, 1.0, 3.0, and 5.0, and the time history of the drag coefficient is compared in Figure 41. It is observed that there is excellent agreement between the two sets of results. Similarly to the Re = 1000 case, the vorticity field, surface vorticity, and drag coefficient for a Reynolds number of 9500 are compared in Figure 42, Figure 43, and Figure 44. Again, there is excellent agreement between the two simulations. The vorticity and computational mesh near the cylinder surface are shown at times of 2.0, 3.5, and 6.0 in

Figure 45. It is observed that the mesh follows the resulting vorticity distribution, while away from the dynamical action, the mesh remains coarse.

The results of this section demonstrate that the current approach is capable of long-time resolution of complicated interplay between surface vorticity generation and coherent wake structures. This interplay is seamlessly handled in the current framework by the introduction of physics-adaptive mesh distribution which is supplemented by mesh-aware adaptive partitioning.

4.5 Flow Past a Stationary Sphere

The flow past a stationary sphere is a classic benchmark case for validating three-dimensional solvers. Despite the symmetric nature of the sphere, the wake is unsteady and fully three-dimensional in the most general case. A steady axisymmetric wake is developed for Reynolds number values less than approximately 210 [93]. For Reynolds number values between 210 and 270, the wake settles into a non-symmetric steady pattern, while Reynolds numbers above 270 display unsteady flow patterns. The rich structures of the wake make it a challenging test problem for which there are a large set of results in the literature with which to compare [72, 93, 94].

In this section, the current results for flow past a sphere at Reynolds numbers of 50, 100, and 300 are compared with results from the literature. The sphere is positioned at coordinates of (7.5,7.5,7.5) in a domain of dimensions 15x15x30. A convective outflow condition is specified on the top domain boundary, and a velocity of 1 in the z-direction is specified at all other boundaries. The equations are solved on an adaptive mesh where the finest mesh spacing is 0.02. ry problems.

Table 4 presents a comparison of the current results with the results reported in [72, 93]. For the Reynolds number of 50 and 100 cases, the flow is steady and there exists a stationary vortex bubble behind the sphere. The drag coefficient and the ratio of the bubble length, LB, to the sphere diameter, D, are computed and compared with the results of [72, 93]. Due to the unsteady nature of the flow for the Reynolds number of 300, the average drag coefficient and the Strouhal number, which represents the frequency with which vortical structures are shed, are reported in the table. The current values of 0.65 and 0.134 are in excellent agreement with the results of Johnson and Patel (0.65, 0.137), and those of Marella et al. (0.65, 0.133). In Figure 46, the vortical structures in the wake of the Re = 300 case are identified using the λ-2 criteria. The interconnected hairpin structures have been accurately captured for about 10 diameters downstream. In Figure 47, the time history of the lift and drag coefficient for the Re = 300 case are displayed. The average values of the drag and lift coefficients (0.65, -0.068) are again in good agreement with the results reported by Johnson and Patel (0.65, -0.069).

4.6 Flow In a Bent Tube

Biological flows often contain flows in tube-like structures with strong secondary fluid motion. In this section, the steady flow in a tube bent through a ninety degree angle is used to validate the ability of the current approach to capture secondary flows in tortuous tubes. Van de Vosse et al. [95] performed laser Doppler velocimetry experiments to obtain the center-plane axial velocities at a set of angles around the tube bend. The Reynolds number based on the tube diameter and the mean inlet velocity is 300. The internal diameter of the tube is 1, and the radius of curvature of the pipe bend is 6. Extensions are added to the start and end of the bend, each 1 diameter in length, to match the experimental conditions. A fully developed parabolic profile is prescribed at the south domain boundary, and a convective outflow condition is prescribed at the west domain boundary.

The shape of the background mesh is shown in Figure 48(a). This demonstrates the efficacy of the mesh pruning algorithm. Streamlines are plotted over contours of velocity magnitude on the center-plane for the current solution in Figure 48(b). The streamlines indicate the presence of a secondary flow in the tube. The axial velocity on the center-plane at the start of the outlet extension is plotted in Figure 48(c) with the experimental values obtained by Van de Vosse et al. [95] It is observed that there is excellent agreement between the current computational results and the experimental results.

4.7 Flow in a Vocal Fold Geometry at Re = 900

A characteristic feature of the flow in the gap regions of the mechanical heart valve is that the flow field exhibits a strong jetting which in turn leads to the production of coherent vortex structures. This is also a characteristic displayed in the flow through the human vocal folds. As can be seen in Figure 49, the human larynx is a complex geometry that has a strong region of constriction and expansion which is formed by structures known as the true and false vocal folds. As air passes through the vocal folds, a strong jet is formed which becomes unstable and breaks down into a set of vortex structures. These vertical structures induce an oscillation of the vocal folds thereby producing sound. Chisari et al. [96] numerically and experimentally examined the start-up flow through a channel which contains two consecutive constrictions which approximates the vocal fold structure. Each constriction leads to the formation of a jet which rolls up into a dipolar vortex structure. The structure remains attached at the head of the jet which experiences secondary instabilities. The vortex head of the first jet then interacts with the second constriction, leading to the formation of rebound vortices. Qualitative flow visualizations and quantitative data on the location and size of the dipolar vortex heads, as well as pressure distributions across the channel width at the location of the vortex head are reported. The current framework is validated against these results.

The simulation configuration is shown in Figure 50. The inlet channel has a non-dimensional diameter of 1. The constrictions are spaced 0.8 units apart in the stream-wise direction and 0.04 and 0.08 units in the cross-stream direction. The hemispherical caps of the constrictions have a radius of 0.2. A time-varying uniform inlet boundary condition is specified at the left domain boundary such that the dimensional experimental velocity is ramped from 0 m/s to 0.54 m/s over a period of 5 ms. A convective outflow condition is used at the right domain boundary. The Reynolds number is set at 900. The Cartesian mesh has a spacing of 0.05 with 5 levels of refinement to give a fine mesh spacing of 0.0015625. The time step size is set to 0.0001. The grid spacing gives about 30 points across the first constriction. The location of the vortex head emanating from the first constriction is tracked by storing the location of the minimum pressure between the first and second constrictions.

In Figure 51, the vorticity contours for the current results are compared with the results of [96] at dimensional times of 3.53 ms, 5.8 ms, 7.2 ms, 7.8 ms, and 9.5 ms. It is observed that qualitatively, there is excellent agreement between the two sets of results in Figure 51(a)-(c). While qualitatively similar, the current vorticity contours predict a stronger set of secondary instabilities on the portion of the jet trailing the vortex head which originates at the first constriction. As the current simulation is performed at twice the effective resolution as the results presented in [96], this is probably due to the better resolution of the thin jet in this work. A comparison of pressure across the channel at the axial location of the first vortex head is shown in Figure 52 for dimensional times of 3.73 and 4.43 ms. It is observed that at these time steps, there is quite good agreement.

4.8 Flow Past an Impulsively Started Moving Cylinder

To demonstrate the capability to accurately simulate moving boundary problems, the impulsive cylinder problem studied in Section 4.4 is repeated for the case where the cylinder is given an impulsive motion in a quiescent fluid at a Reynolds number of 1000. The velocity is set to zero at all domain boundaries. At T = 0, the cylinder is given an impulsive velocity of 1.0 in the negative x-direction. The numerical parameters such as grid spacing and number of processors is the same as the Re = 1000 case in Section 4.3. The surface vorticity is compared with the results of [92] in Figure 53 for T = 0.4, 1.0, and 3.0. It is observed that while the curves are slightly less smooth than the stationary computation, there is still excellent agreement with the published results. The excellent agreement of the stringent surface vorticity test shows that the current approach produces high quality results in the presence of moving boundaries.

4.9 Sedimentation of a Sphere

To demonstrate the three dimensional moving boundary capabilities of the current work, we use pELAFINT3D to compute the sedimentation of a sphere in an otherwise quiescent fluid. As the motion of the sphere is not forced, but instead evolves as a response to the flow solution, this problem provides a thorough assessment of the accuracy of the moving boundary solver. To that end, a sphere is placed at a location of (7.5,7.5,45) in a domain of 15x15x50 units. The ratio of the sphere density to the fluid density is 10, and a force balance at steady state gives the Froude number to be $\mathrm{Fr} = \mathrm{U}^2/\mathrm{gD} = 11$, where U and D are the sphere's steady settling velocity and diameter, and g is the magnitude of gravitational acceleration. Here, U and D are taken to be 1. The Reynolds number, based on the steady settling velocity and sphere diameter is 100. The Cartesian base mesh is set to a spacing of 0.16, and the mesh is enriched with 3 levels of refinement for a fine mesh spacing of 0.02. The time step is computed such that a CFL criteria of 0.1 is maintained.

In order to evolve the motion of the sphere, a weakly coupled FSI model is used. The sphere's centroid is advected using Newton's equations of motion, where the force is the due to the pressure and viscous stress integrated over the sphere's surface, and the weight and buoyancy effects due to gravity. No restriction is placed on the sphere to settle along any specified direction. This means any spurious force may significantly alter the sphere's trajectory.

Because the drag force in the steady settling mode should be equal to the drag force of a stationary sphere at the same Reynolds number, the current simulation should have a drag coefficient that converges to a value of 1.09 [93]. The settling velocity should converge to a value of 1 as was specified in the problem setup. The current results for vorticity about the x-axis are shown in Figure 54 for the steady settling of the sphere. The evolution of the drag coefficient and the settling velocity are shown in Figure 55. It is observed that the drag coefficient evolves in a smooth manner and that it asymptotes to approximately a value of 1.10. The settling velocity attains a steady value of 1. This set of results lends confidence to the capabilities of pELAFINT3D to handle three-dimensional moving boundary problems.

Table 4 Comparisons of drag coefficient, vortex bubble length, and Strouhal number for flow behind a sphere at Reynolds numbers of 50, 100, and 300.

Re	50		100		300	
Study	C_D	L_B/D	C_D	L_B/D	C_D	St
Johnson & Patel [93]	1.57	0.40	1.08	0.88	0.65	0.137
Mittal [94]	1.57	0.44	1.09	0.84	0.00	0.00
Marella [72]	1.56	0.39	1.06	0.88	0.65	0.133
Current	1.57	0.41	1.07	0.87	0.65	0.134

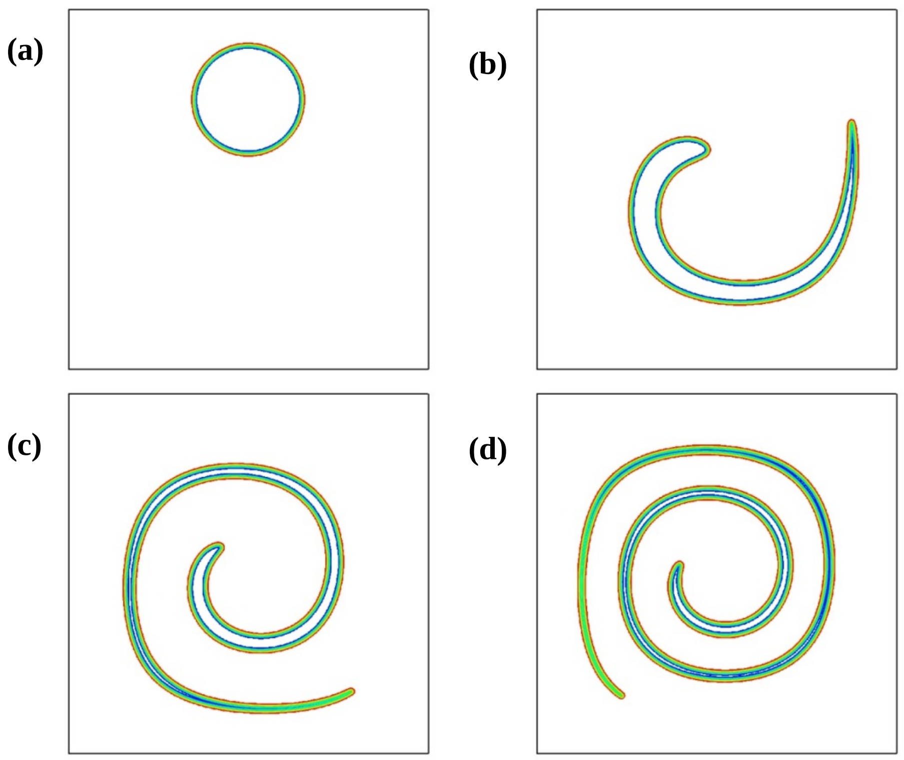


Figure 35 Evolution of an initially circular level set field in a straining vortex flow. The field is shown at times (a) 0, (b) 1, (c) 2, and (d) 3. The level set field is capable of handling large deformations in a natural manner.

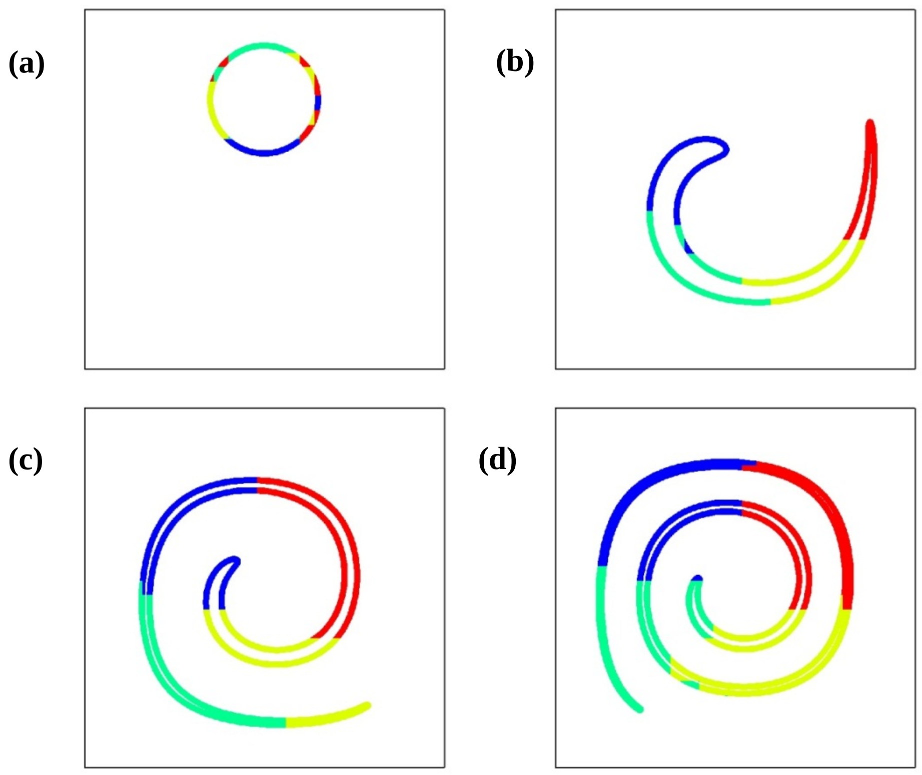
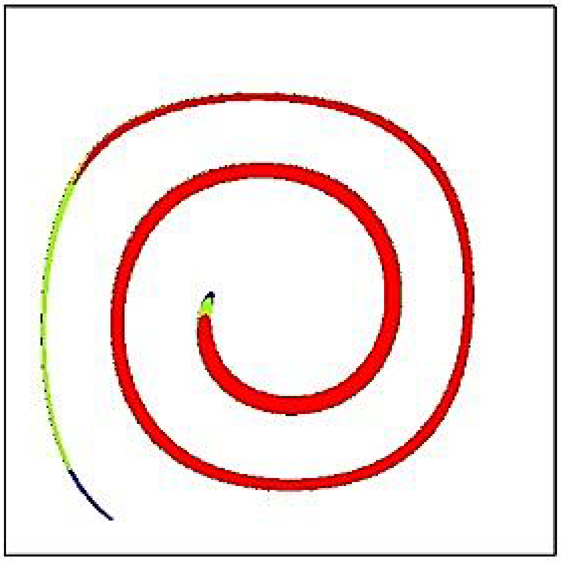


Figure 36 Partitioning of the level set tube for the circle in a vortex field across 4 MPI processes. The partitioning adapts as the level set field evolves such that each partition has approximately the same number of level set tube points.

(a)



(b)

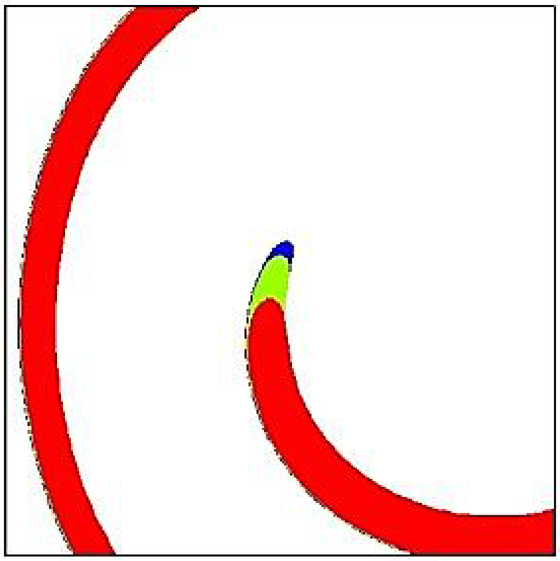


Figure 37 Comparison of the results for the level set field of a circle in a vortex flow for 3 subsequent levels of refinement. The coarsest mesh is shown in red, while the finest mesh is blue. The entire level set is compared in (a) while the deviations in the position of the leading tip are shown in (b). The thicker head region of the level set field suffers less mass loss than the thin tail. Even though the level set is not resolved at the tail, the computation remains stable.

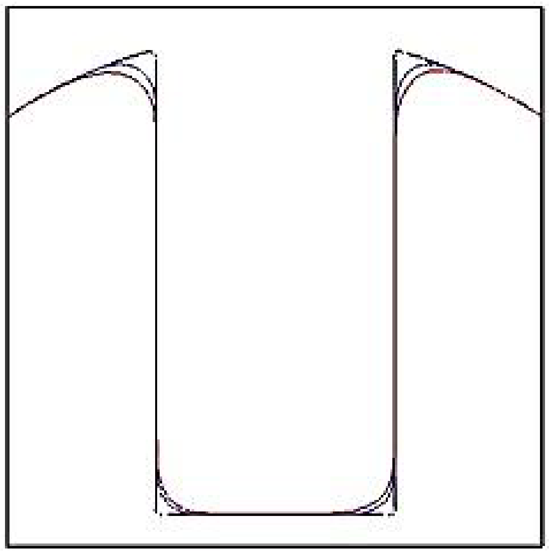


Figure 38 Comparison of the final zero contour positions of level set field for the Zalesak disk using ENO (red) and WENO (blue) spatial discretization for the solution of the level set advection equation.

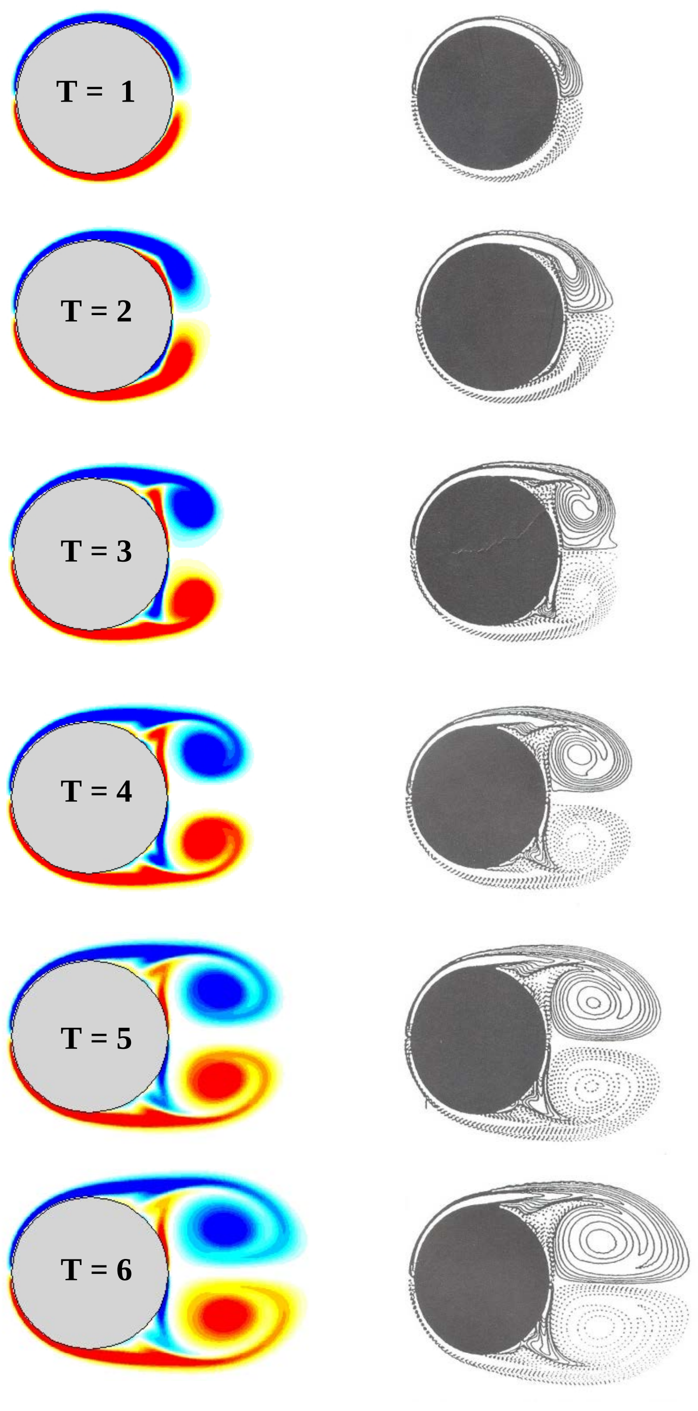


Figure 39 A comparison the current results for vorticity contours with the results of [92] for an impulsively started flow around a stationary cylinder at Re = 1000 at a time of 1, 2, 3, 4, 5, and 6.

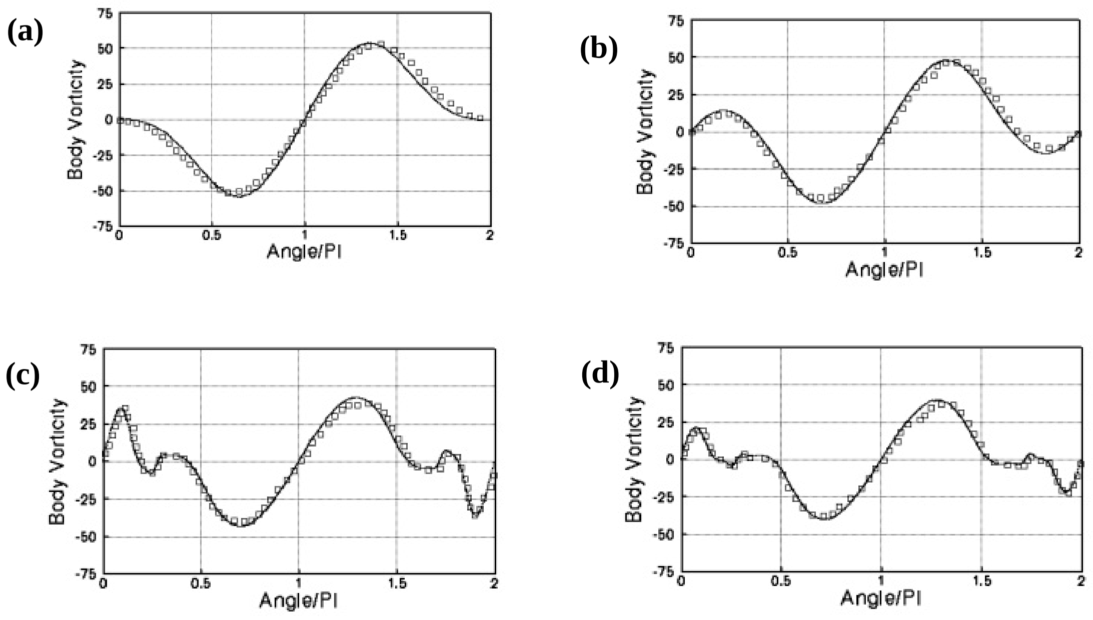


Figure 40 A comparison of the current surface vorticity results (solid line) with the results of [92] (squares) for an impulsively started flow around a stationary cylinder at Re = 1000 at time (a) T = 0.4, (b) T = 1.0, (c) T = 3.0, and (d) T = 5.0.

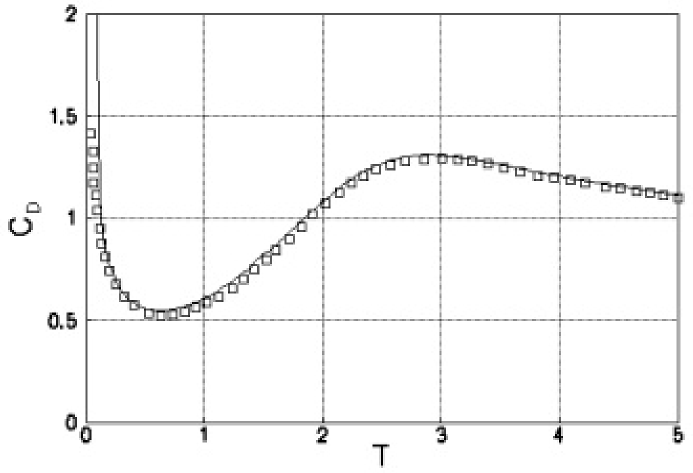


Figure 41 A comparison of the current drag coefficient results (solid line) with the results of [92] for an impulsively started flow around a stationary cylinder at Re = 1000.

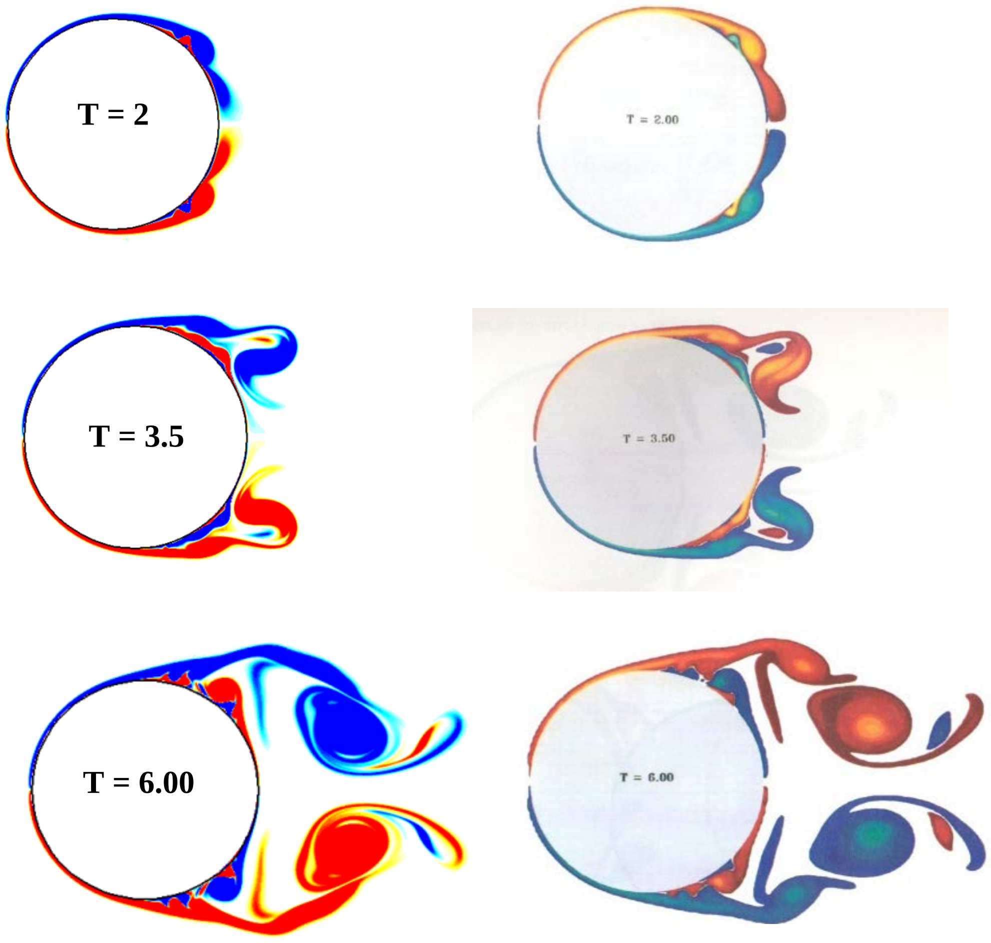


Figure 42 A comparison the current results for vorticity contours with the results of [92] for an impulsively started flow around a stationary cylinder at Re = 9500 at times of 2, 3.5, and 6.

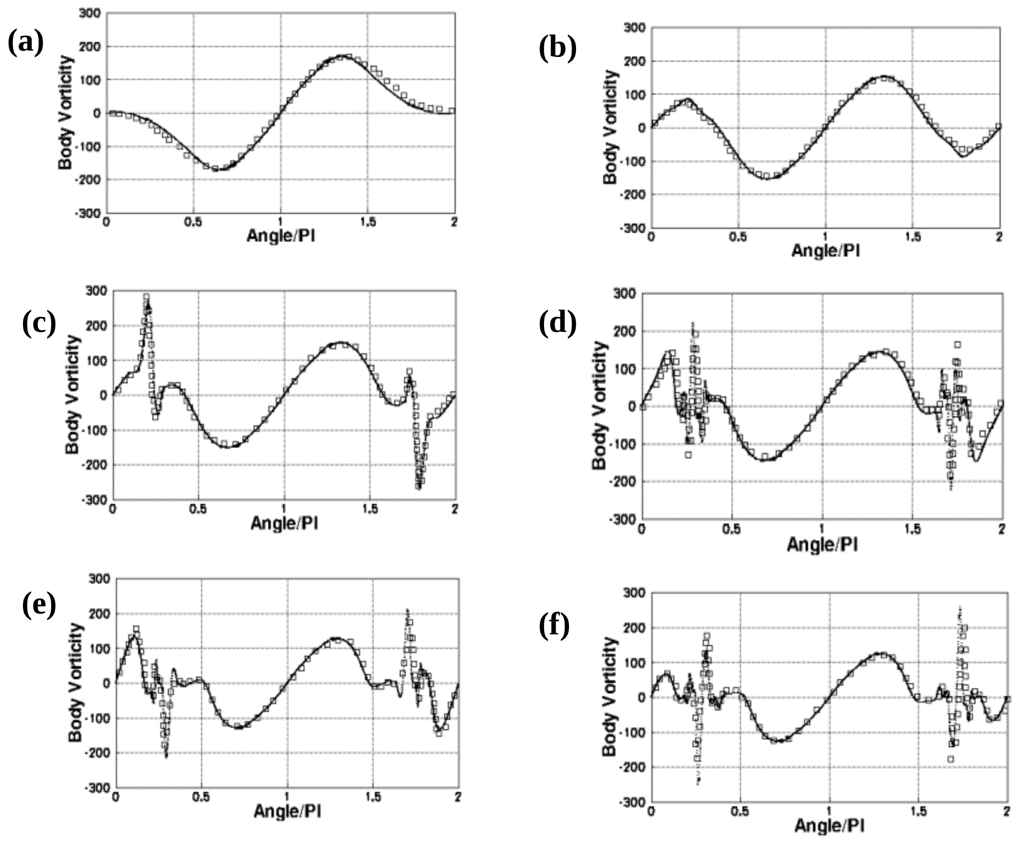


Figure 43 A comparison of the current surface vorticity results (solid line) with the results of [92] (squares) for an impulsively started flow around a stationary cylinder at Re = 9500 at time (a) T = 0.4, (b) T = 1.4, (c) T = 1.8, (d) T = 2.6, (e) 4.2, and (f) 5.0.

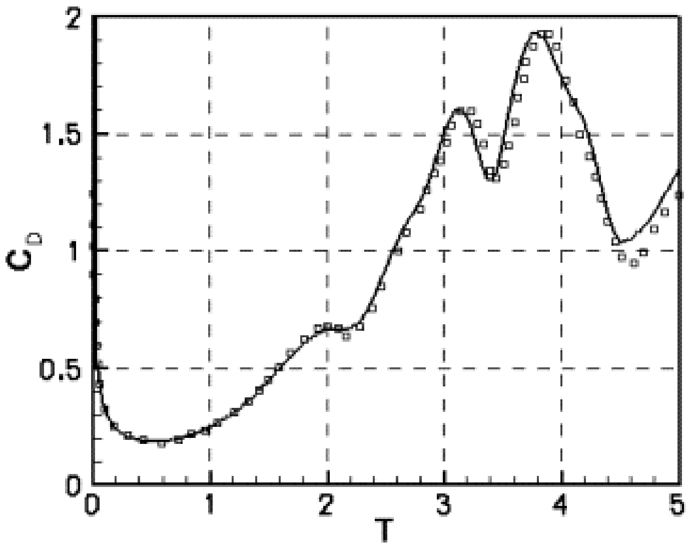
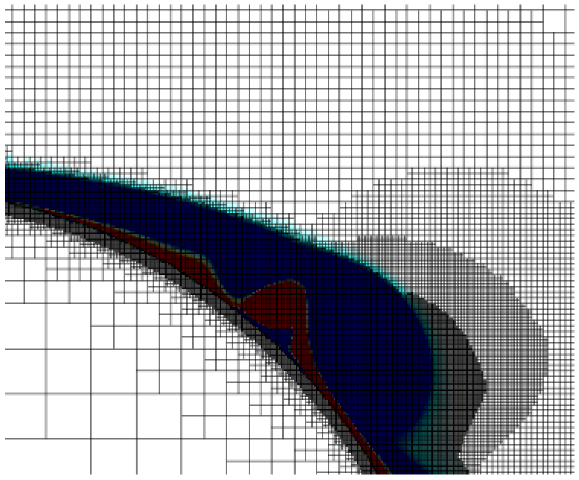
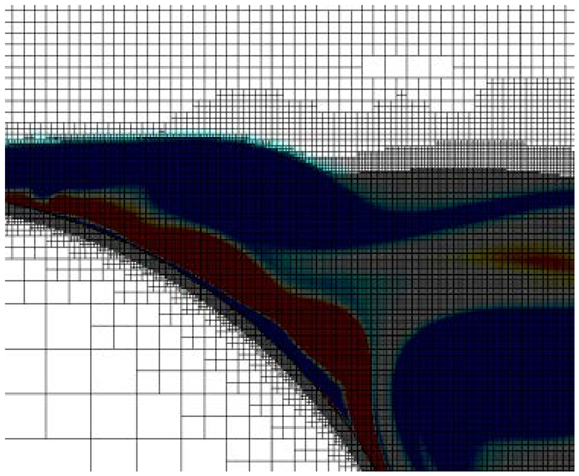


Figure 44 A comparison of the current drag coefficient results (solid line) with the results of [92] an impulsively started flow around a stationary cylinder at Re = 9500.

(a)



(b)



(c)

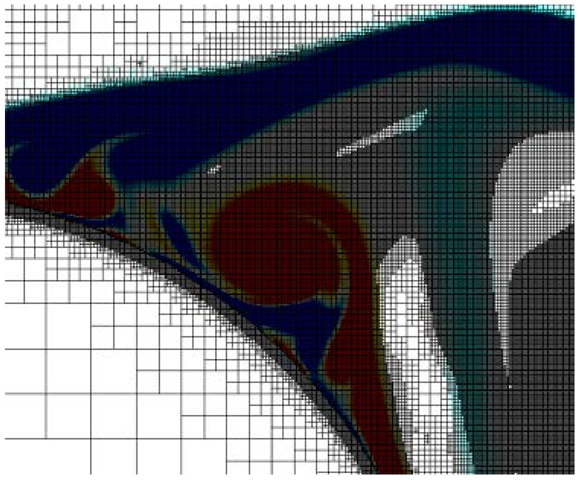


Figure 45 The evolution of the vorticity field and computational mesh for the impulsively started flow around a stationary cylinder at Re = 9500 for times of (a) 2.0, (b) 3.5, and (c) 6.0.

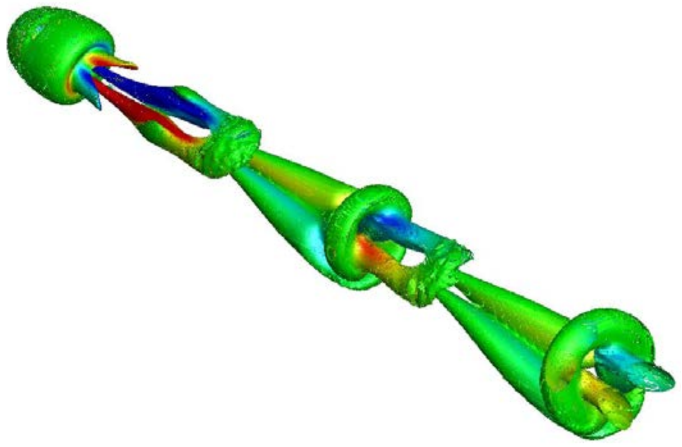


Figure 46 Vortical structures behind a sphere at a Reynolds number of 300 as indicated by the λ-2 criterion. The current approach is capable of the long time integration which is required to capture the wake in great detail far downstream.

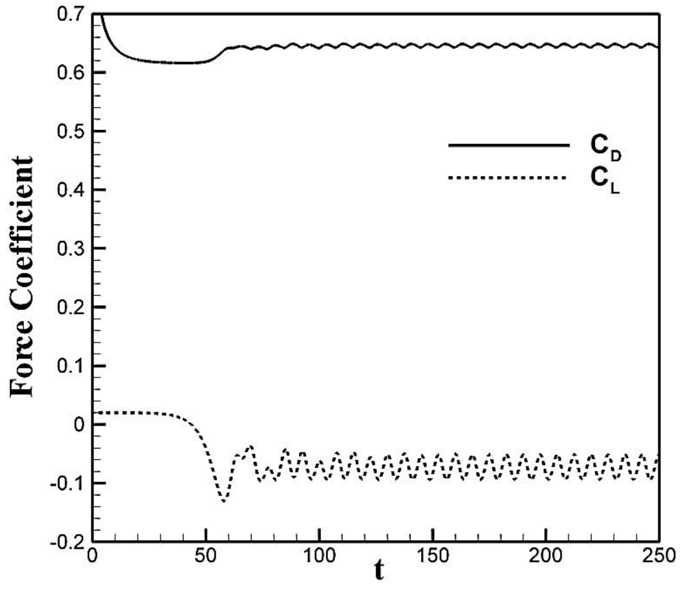


Figure 47 The time histories of lift and drag coefficient for flow past a sphere at a Reynolds number of 300. The average values of both coefficients are in good agreement with the results of [93].

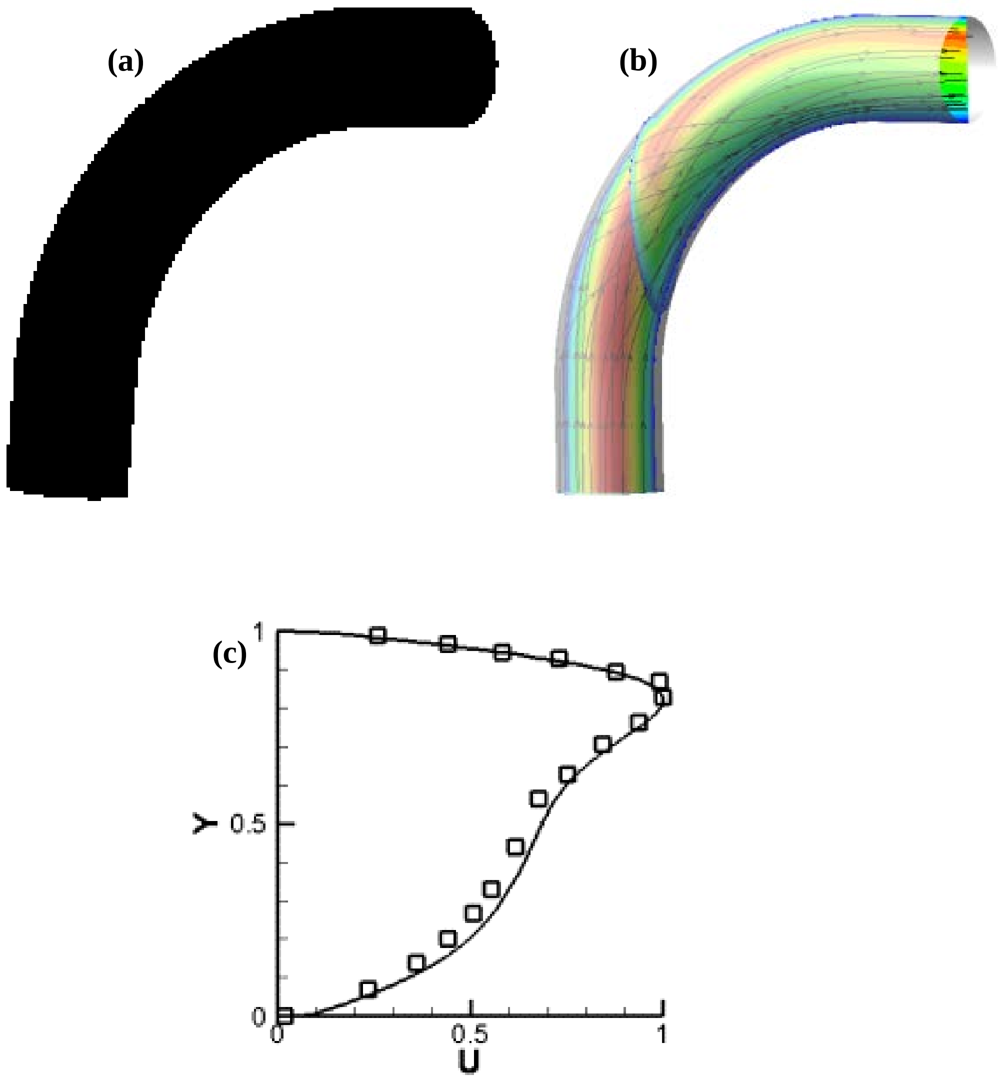


Figure 48 The simulation of secondary flow in a 90 degree bent tube at a Reynolds number of 300. In (a), the ability of the current pruning algorithm is shown by the resulting Cartesian background grid. In (b) the velocity magnitude is plotted on the center-plane and streamlines are plotted to indicate the presence of a secondary flow. The axial velocity at the bend outlet is compared with the experimental results of [95] in (c).

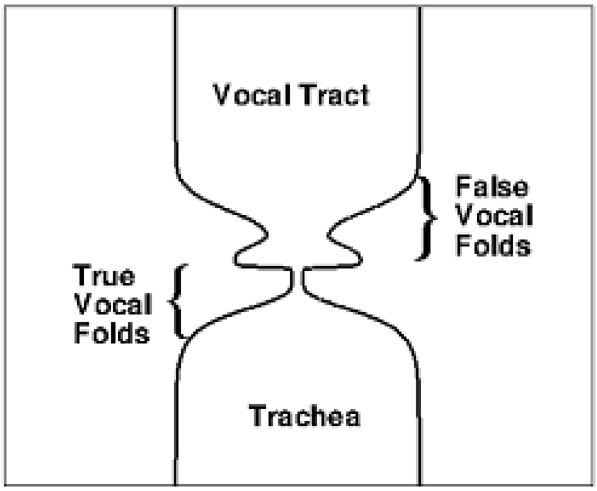


Figure 49 A schematic of the human larynx. The upstream and downstream channels are separated by a complex flap system which induces the jet flow required for sound production.

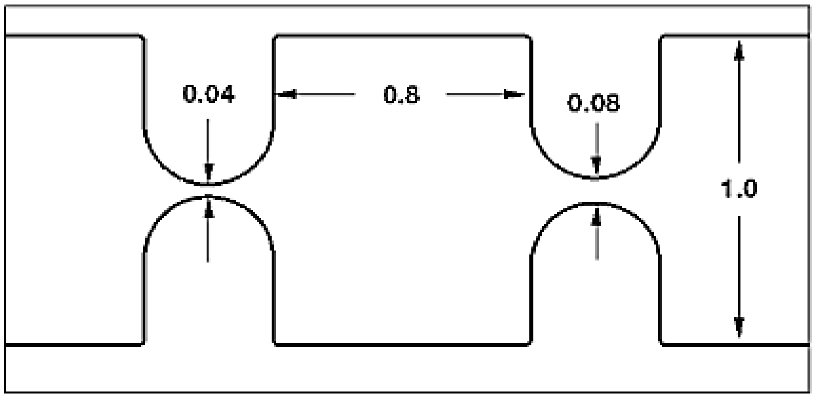


Figure 50 The representative vocal fold geometry replicated from [96]. The second constriction gap is twice that of the first constriction.

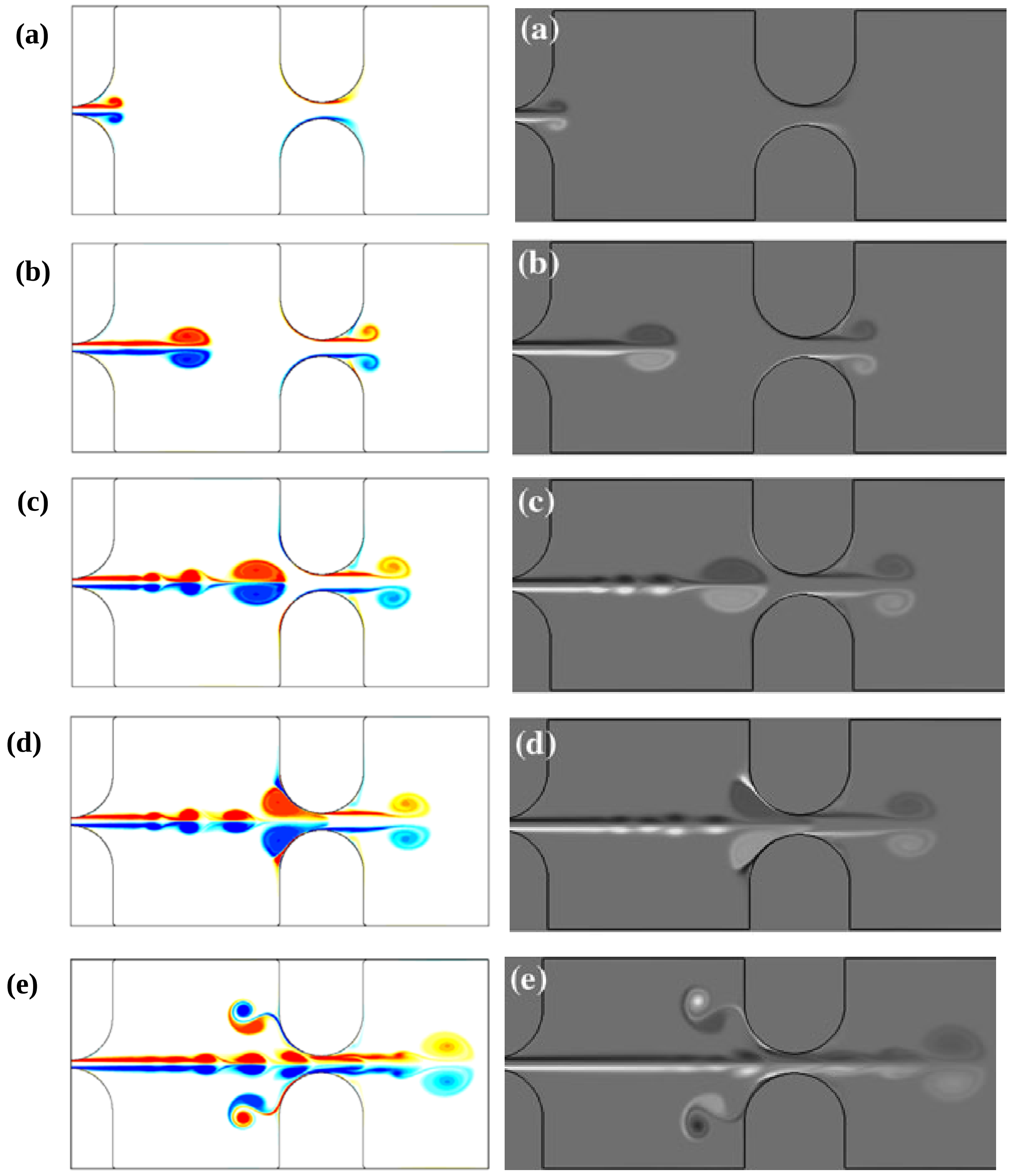


Figure 51 A comparison of vorticity contours for the current results (left) with the results of [96] (right) at times of (a) 3.53 ms, (b) 5.8 ms, (c) 7.2 ms, (d) 7.8 ms, and (e) 9.5 ms. There is very good qualitative agreement, but the current results resolve the smaller-scale features with higher fidelity.

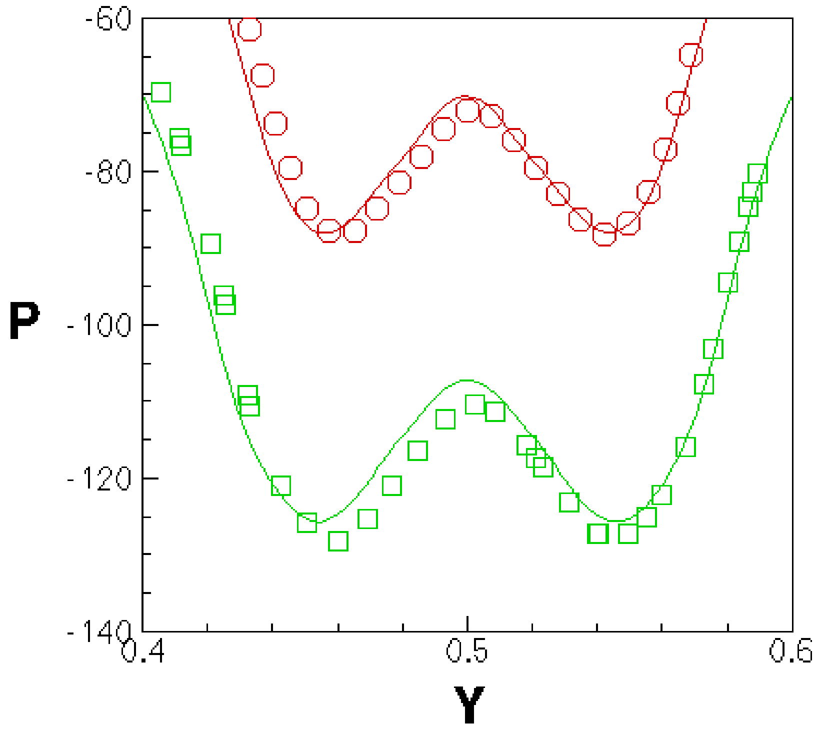


Figure 52 A comparison of the current results (solid lines) for pressure with the results of [96] (symbols) at times of 3.73 ms (circles) and 4.43 ms (squares).

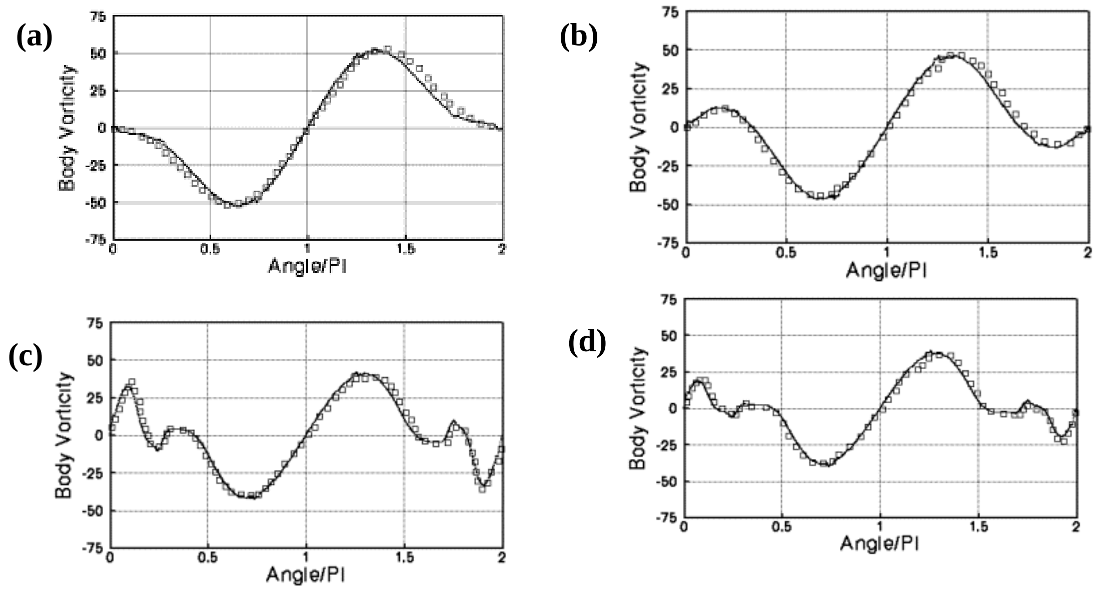


Figure 53 The vorticity on the surface of an impulsively moving cylinder. The cylinder is set in motion with unit velocity in the negative x-direction. The current results (solid lines) compare well with the results of Koumousatkos and Leonard [92] (squares).

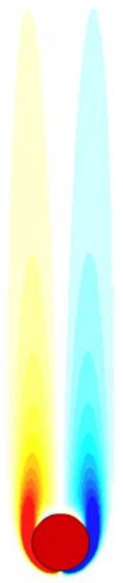


Figure 54 The x-vorticity about a sphere settling under the action of gravity. The Reynolds number is 100 and the Froude number is 10.

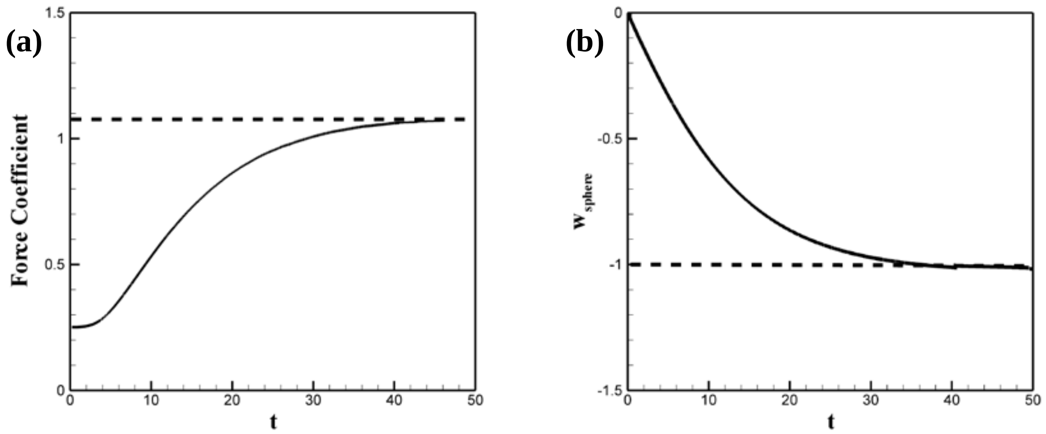


Figure 55 The time histories of (a) drag force and (b) settling velocity for a sphere settling under the action of gravity. The Reynolds number is 100 and the Froude number is 11. The force history is smooth in time, showing the efficacy of the current hybridized interface treatment. The drag force at steady state matches the value of 1.09 predicted by drag correlation very well.

CHAPTER 5

CLOSURE OF A BILEAFLET MECHANICAL HEART VALVE

5.1 Introduction

As demonstrated in Chapters 3 and 4, the current moving boundary solver is capable of accurately capturing the intricate vorticity dynamics of moving boundaries in the Reynolds number range of the MHV problem, where flow features of widely disparate length scales were resolved in an efficient manner. Additionally it was concluded that the approach is highly scalable and capable of handling intensive computational workloads. In this chapter, we show that the current package is a good approach to modeling the MHV problem by computing the closing phase of the leaflet motion through the Medtronic ADVANTAGE MHV. In the rest of the chapter, previous simulations from the literature are reviewed, and the problem setup and results are discussed.

5.2 Previous Mechanical Valve Simulations

The first simulations of MHVs did not appear until the middle of the 1990's due to the computational intensiveness of the problem. In preliminary attempts to study the MHV problem, the literature consists of simulations in which the valve is held fixed and the inlet condition lacks pulsatility. Additionally, sometimes the simulations were performed on reduced portions of the full 3D geometry on the basis of the symmetry of the prosthesis. Shim and Chang [97] examined static tilting-disk and Bjork-Shiley [98] valves in the half-open position using the finite-element method. Kiris et al. [99] computed a RANS solution for the flow through a Bjork-Shiley valve fixed at an angle of

30 degrees and the Reynolds numbers was varied from 2,000 to 6,000. King et al. [100] appealed to the symmetry of the valve geometry to simulate one-quarter of a valve in the fixed position through the first half of systole up to a Reynolds number of 1500.

Ge et al. [101] performed the first full 3D DNS simulation of a fixed valve in the fully open position for a steady inlet condition and obtained grid-insensitive results (the large-scale structures are independent of grid-size). In the static configuration, two pairs of trailing vortices emanate from the leaflets which surround a core jet. When the Reynolds number exceeds 1200, the trailing vortices experience oscillations that in turn induce oscillatory motion of the core jet, eventually leading the jet to become unstable. Additionally, it was found that when the grid is under-resolved, the mechanism leading to instability is artificially exhibited at a reduced Reynolds number. These results indicate that future simulations should be performed for the full heart valve geometry on sufficiently fine grids.

As an attempt to increase the complexity of MHV simulations while retaining computational tractability, a significant amount of recent literature has considered moving valve leaflets and pulsatile inlet conditions at the price of simplifying assumptions such as two-dimensionality or symmetry about a plane through the valve housing. Cheng et al. [6] performed a two-dimensional simulation of the closure phase using an Arbitrary Lagrangian-Eulerian method, and later extended the results to a 3D simulation where only one-quarter of the valve geometry was considered [19]. Krishnan et al. [7] studied the closure phase of a two-dimensional approximation of the Medtronic ADVANTAGE bileaflet valve in the mitral position using a sharp-interface method enhanced with adaptive quadtree meshing. It was observed that the flow in the gap between the housing and leaflet in the fully closed position involves a complex interaction of the strong and opposite signed vorticity attached at the leaflet and housing surfaces, which generates large shear stresses and high particle residence times in this region. It was noted that while platelet activation and thrombus formation will probably be experienced, thrombus deposition is not likely to occur in the gap. Unfortunately, the two-dimensional nature of the simulation precluded the study of the hinge region.

Very recently, literature has been published that attempts to consider the true physiological conditions of the mechanical heart valve problem. Tai et al. [102] used an artificial compressibility method in combination with an overlapping grid approach and loose FSI coupling between the flow field and the valve leaflets to simulate the opening phase dynamics of a 27 mm St. Jude’s bileaflet mechanical heart valve with a reduced peak Reynolds number. Nobili et al. [13] simulated the complete 27 mm St. Jude’s bileaflet mechanical heart valve including hinges by implementing a strongly-coupled partitioned FSI solver in FLUENT at the physiologic Reynolds number. However, the leaflets were scaled to 98 percent of their true size and the simulation was under-resolved downstream to capture the smallest scales in the flow field. Ge and Sotiropoulos [17] used a sharp immersed boundary method on a curvilinear background mesh (CURVIB) to solve for the flow in the 23 mm St. Jude’s bileaflet mechanical valve with prescribed leaflet motion, and Borazjani et al. [15] extended the work of [17] to include full FSI of the leaflets where the hinge region was neglected. The results of the simulation concluded that during the acceleration portion of systole, coherent vortical structures are generated at and shed from the leaflet surfaces as well as the valve housing. These structures interact with each other undergoing a transition to a turbulent like state. The authors also demonstrated that contrary to previous belief, the flow does not re-laminarize after breaking down into the small-scale state. It instead survives and is then washed downstream. The opening phase of the leaflet motion was demonstrated to be unstable in a weakly-coupled FSI scheme, while the closing phase was much more stable.

It should be noted that none of the 3D work cited has completely resolved all gap regions. Several studies have been undertaken to try to quantify the flow physics in the gaps. Most notably, Simon et al. [8] have used the large-scale simulation results of [16] to supply boundary conditions for a simulation at the scale of the hinge region of a St. Jude's mechanical heart valve. It was demonstrated that the shear stress in the hinge region is much higher during diastole, and that it can be expected that platelet damage will occur during diastole and thrombosis will occur during systole.

5.3 The Medtronic ADVANTAGE Valve

The current work investigates the flow in a Medtronic ADVANTAGE valve. The ADVANTAGE valve has been the subject of investigations by previous collaborators, and serves as a demonstration case in the current thesis (Figure 56). The ADVANTAGE valve is a variation on the bi-leaflet mechanical valve design with an increased central orifice area which is intended to reduce the central flow velocity and minimize flow separation and turbulence [3]. The hinge recess has a butterfly shape and a widened outflow area to promote washing of blood elements throughout the cardiac cycle.

The housing is approximately 25 mm in diameter, and the leaflets are approximately 1.31 cm long and 0.09 cm thick [103]. Each leaflet has two hinge tabs which fit into butterfly-shaped hinge recesses formed in the valve housing. The leaflets rotate through an angle of approximately 63.8 degrees about an axis connecting each tab.

The length from the pivot axis to the leaflet edge is 1.13 cm. The leaflet density and moment of inertia are 2000 kg/m3 and 3.3x10-9 kg-m2.

The idealized system studied in this work is shown in Figure 57. The valve housing is inserted in a tube which extends in both the upstream and downstream directions by 5 tube diameters. The Sinus of Valsava is approximated as three bulbs which are located just downstream of the housing. The bulbs are oriented at 120 degree increments around the tube. The valve assembly, which is shown more clearly in Figure 3, was presented in Chapter 1. The hinge recess is shown from an upstream axial view Figure 3(a). The manner in which the leaflet tabs insert into the butterfly-shaped recess is clarified in Figure 3(b). A zoomed in view of the leaflet insertion (Figure 3(c)) emphasizes the extremely small gap formed between the leaflet tab and the hinge recess wall. The hinge recess gap is approximately 0.15 cm. The centerline and housing gaps that exist when the valve is in the fully closed position are shown in Figure 2. The centerline gap is approximately 0.017 cm, while the housing gap is 0.03 cm.

5.4 Fluid-Structure Interaction Model

A weakly coupled fluid-structure algorithm is implemented in the current work. The results of [16] indicate that the opening phase of leaflet motion would be unstable with such a scheme due to the added mass effect. As a fully coupled FSI algorithm is not the purpose of the present work, only the closing phase of leaflet motion is simulated. A fully-coupled sub-iteration scheme with Aiteken acceleration has been constructed for flexible boundary calculations, and the extension of this work to the rigid case is the subject of current work.

The angular velocity of the leaflet is computed using the non-dimensionalized Euler equation of rotation,

$$\gamma I^* \frac{d\vec{\Omega}^*}{dt^*} = T_F^* + \frac{(\gamma-1)V^*}{Fr}\left((\vec{x}_g^* - \vec{x}_a^*) \times \vec{e}_g\right). \tag{5.1}$$

Here, V^* and $I^*$are the non-dimensional volume and moment of inertia per unit mass of the leaflet, $\gamma = \rho_s/\rho_f$ is the density ratio of the leaflet and fluid, Ω^* and T_F^* are the non-dimensional angular velocity and fluid torque about the fixed rotation axis, $Fr = U^2/gL$ is the Froude number, $\vec{x}_g^* - \vec{x}_a^*$ is the relative position vector between the center of gravity and the rotation axis, and $\vec{e}_g$ is the direction gravity acts. The rigid rotation velocity of the leaflet is then computed as $u_l(\vec{x}^*) = \vec{\Omega}^* \times \vec{x}^*$.

5.5 Simulation Setup

The current simulations consider the closure of the fully three-dimensional Medtronic ADVANTAGE valve with artificially increased gaps. The leaflets are initialized in the fully open position to avoid FSI stability issues and are scaled to 98 percent of their original size to decrease the computational burden of the calculations. A constant density Newtonian fluid is assumed to be representative of blood at high shear rates. The viscosity and density of the fluid are 0.0035 cP and 1000 kg/m3. Therefore, the density ratio of the leaflet to the fluid is $\gamma = 2$. The effects of gravity are not considered in the current thesis.

Time-varying plug flow velocity boundary conditions were prescribed from the experimental volumetric flow rate measured through a St. Jude’s MHV in [104]. The peak Reynolds number associated with the current setup was 6,000. A cubic spline fit was used to compute flow rates at times between the digitized samplings (Figure 58). The location of the inlet condition was dynamically altered during the simulation to ensure that inlets always had a mean velocity coming into the domain. A convective outflow condition was specified at the location of the outlet boundary.

The level set fields describing the valve were constructed from unstructured surface mesh representations of the mechanical valve housing and leaflets which were created using the Gambit mesh generation package. The housing is extended 5 valve diameters in both the upstream and downstream directions to provide room for flow development. The housing surface mesh has approximately 627,000 nodes and 1.25 million triangular elements. The leaflet surface meshes each contain roughly 15,000 nodes and 29,000 triangular elements.

The time-step was set to satisfy a CFL criteria of 0.3, and the Cartesian base mesh spacing was set to 1 mm. Three levels of refinement were used to enrich the mesh which resulted in a fine mesh spacing of 0.125 mm. This grid spacing is of the order of the smallest structures observed in [14] which may make the simulation under-resolved. A grid independence study was not under taken in this work. Refining the current results to a physiologically correct condition is the subject of ongoing work.

5.6 Results

The MHV simulation was run for one complete cardiac cycle. The evolution of the complex flow structures, shown in Figure 59 and Figure 60, is depicted using the λ-2 method [105]. The λ-2 method identifies a vortical structure as a coherent region over which the second eigenvalue of $DD + \Omega\Omega^T$ is negative. Here D and Ω are the symmetric and anti-symmetric components of the velocity gradient tensor. The isocontours are colored by vorticity magnitude.

In Figure 59, the forward flow phase is shown at times of 96, 125, 162, 194, 228, 260, 297, and 333 ms. Figure 60 shows the flow field during the valve closure process at times of 366, 383, 400, 420, and 435 ms. In Figure 59(a)-(c), the formation of the sinus vortices is clearly observed. The vortical structure in the top most sinus is stripping a filament of vorticity being shed from the left leaflet which evolves to wrap around the main sinus structure. In Figure 59(d), the vortical structures shed from the leaflet surfaces are clearly visible as an organized system of intertwined horse shoe vortices of alternating orientations. The sinus structures have begun to strip vortical filaments from the axial jets which emanate through the valve orifices. If the stripped filaments are on the sinus side of a separating manifold, they are entrained into the sinus structure, otherwise they are washed downstream. The filament entrainment leads to an increasingly complex sinus structure with a large-scale low vorticity structure mixed with higher vorticity filaments. In Figure 59(e)-(f), the sinus vortices continue to strip vorticity from the orifice jets. As the flow rate decreases in Figure 59(f)-(h), the flow completely breaks down into a small-scale turbulent state similar to that observed in [16].

In Figure 60(a), the reverse flow rate has just initiated. The leaflets are still in the fully open position in Figure 60(b), but the vortical structures generated during the forward flow phase are washed back into the ventricular side of the device. The leaflets start to rotate in Figure 60(c), and the vortical structures being pushed through the closing central orifice are observed to be stretched into elongated vortex loops in Figure 60(d). Finally, the leaflets accelerate and quickly obtain attain a fully closed position in Figure 60(e). As the leaflet rotates, viscous cross diffusion causes boundary vorticity to connect with the vortex structures such that the structures reorient themselves into horseshoe vortices emanating from the leaflet surface.

In order to more clearly understand the flow evolution, Figure 61 and Figure 62 show slices of the vorticity field through the symmetry plane of the valve. In Figure 61, the vorticity in the x-, y-, and z-planes are shown during the forward flow at times of (a) 96 ms, (b) 125 ms, (c) 162 ms, (d) 194 ms, (e) 228 ms, (f) 260 ms, (g) 297 ms, and (h) 333 ms. In Figure 61(a)-(d), the vorticity is shed from the leaflet surfaces and moves downstream, where the wake becomes unstable. The vorticity cross-braids in the x- and y-directions amplifies in Figure 61(e)-(f) as the structures intensify in strength. As the flow decelerates in Figure 61(g)-(h), a strong asymmetry forms in the vorticity shed from the bottom leaflet. The closure phase of the cycle is shown in Figure 62(a)-(d). It is observed that as the leaflets close, structures created during the forward phase are swept back through the leaflets. This brings regions of opposite signs of streamwise-oriented vorticity into close proximity, and these aggregate patches are stretched in the streamwise direction as the leaflets close. This process may lead to a significant stress environment, and a more detailed understanding of the stress displayed in this time interval must be obtained.

The time-history of leaflet motion is compared with the results of Borazjani et al [15] in Figure 63. In the current simulation, the leaflets attain a closed state in approximately 100 ms. This value is 3 times the closure time reported in [7, 13, 16]. There are several potential sources of this discrepancy. First, the sinuses may not have a physically relevant shape or orientation. In fact, in the current simulation, the sinuses are not oriented symmetrically with respect to the valves, and they are far larger than other similar simulations. It is well known that the sinuses are important in determining the closure dynamics of the leaflets [106]. If the vortical structures are too strong, as may be induced by the artificial initial position of the leaflets utilized in the current simulation, the leaflets closure time may be significantly increased. Second, there is a factor of 3 discrepancy in the maximum reverse flow rate used in [16] and [13]. Therefore, the flow conditions here may have too weak of a peak reverse flow rate. Finally, the simulation may not exhibit grid independent results. Determining the reason for the discrepancy in valve motion is the subject of current work.

In Figure 64, the size of the computation grid is shown versus the simulation time step. The grid initially had approximately 10 million cells, and through the course of the simulation, the mesh size was increased to 30 million cells. The beginning of the reverse phase of the flow cycle is evident near 7500 time steps, as the movement of vortical structures to the ventricular side of the valve causes the grid size to increase after a period of quasi-steady behavior. Figure 65(a) shows the wall clock time required to perform each step of the simulation, while Figure 65(b) gives a zoomed in view of the wall clock time per step at the transition between stationary and moving leaflets. Before this transition, the mesh was only updated during the refinement procedure which occurred every 10 time steps. As observed in Figure 65(b), the refinement steps were nearly 2.5 times more expensive than the steps for which refinement did not occur. During this part of the calculation, the Poisson equation preconditioner was only reconstructed during the refinement steps. After the leaflets started to move the preconditioner was required to be rebuilt each step. This manifests itself as the large increase in wall clock time for the time steps after motion had initiated. In this phase of the simulation, the refinement steps were only 20 percent more expensive than the remaining time steps. Therefore, it is noted that the current refinement procedure does not significantly alter the cost of the simulation.

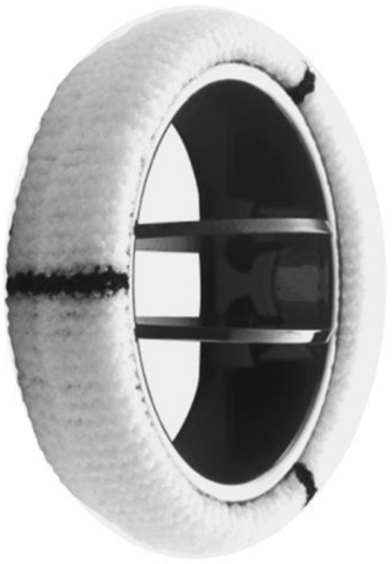


Figure 56 The Medtronic ADVANTAGE bi-leaflet mechanical heart valve. The valve is composed of two pyrolytic occluders which rotate in butterfly-shaped recesses in a pyrolytic housing.

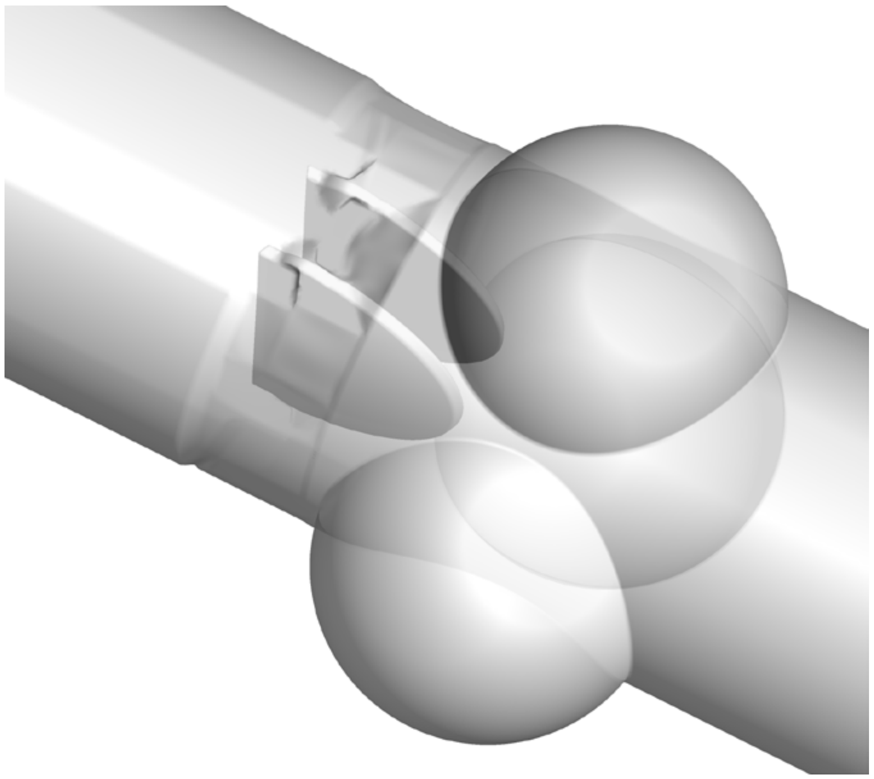


Figure 57 The geometry of the bi-leaflet mechanical heart valve. The valve is three components: a housing and two leaflets. The leaflets have tabs at each end which insert into butterfly-shaped recesses in the housing. In the current setup, the housing is inserted in a tube which has three bulbs to approximate the Sinus of Valsalva.

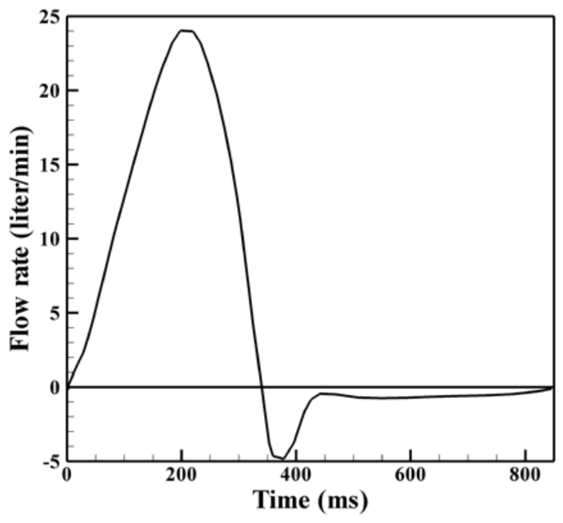


Figure 58 The unsteady flow rate used in the mechanical valve problem. The flow rate is constructed from a cubic-spline fit to data digitized from the experimental results of [104].

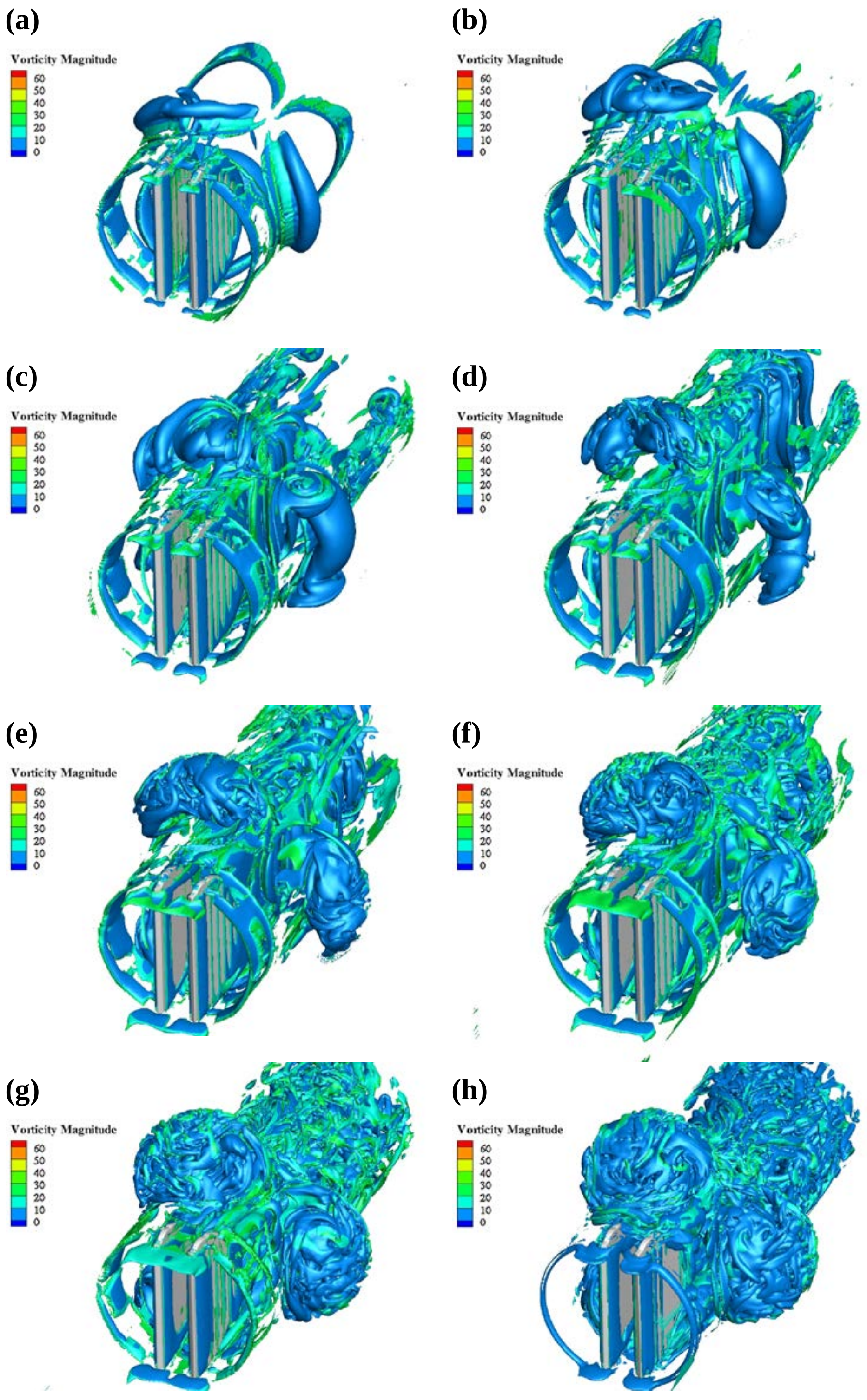


Figure 59 The forward flow in the Medtronic ADVANTAGE mechanical heart valve as indicated by λ-2 isocontours colored by vorticity magnitude. The flow is shown at times of (a) 96 ms, (b) 125 ms, (c) 162 ms, (d) 194 ms, (e) 228 ms, (f) 260 ms, (g) 297 ms, and (h) 333 ms.

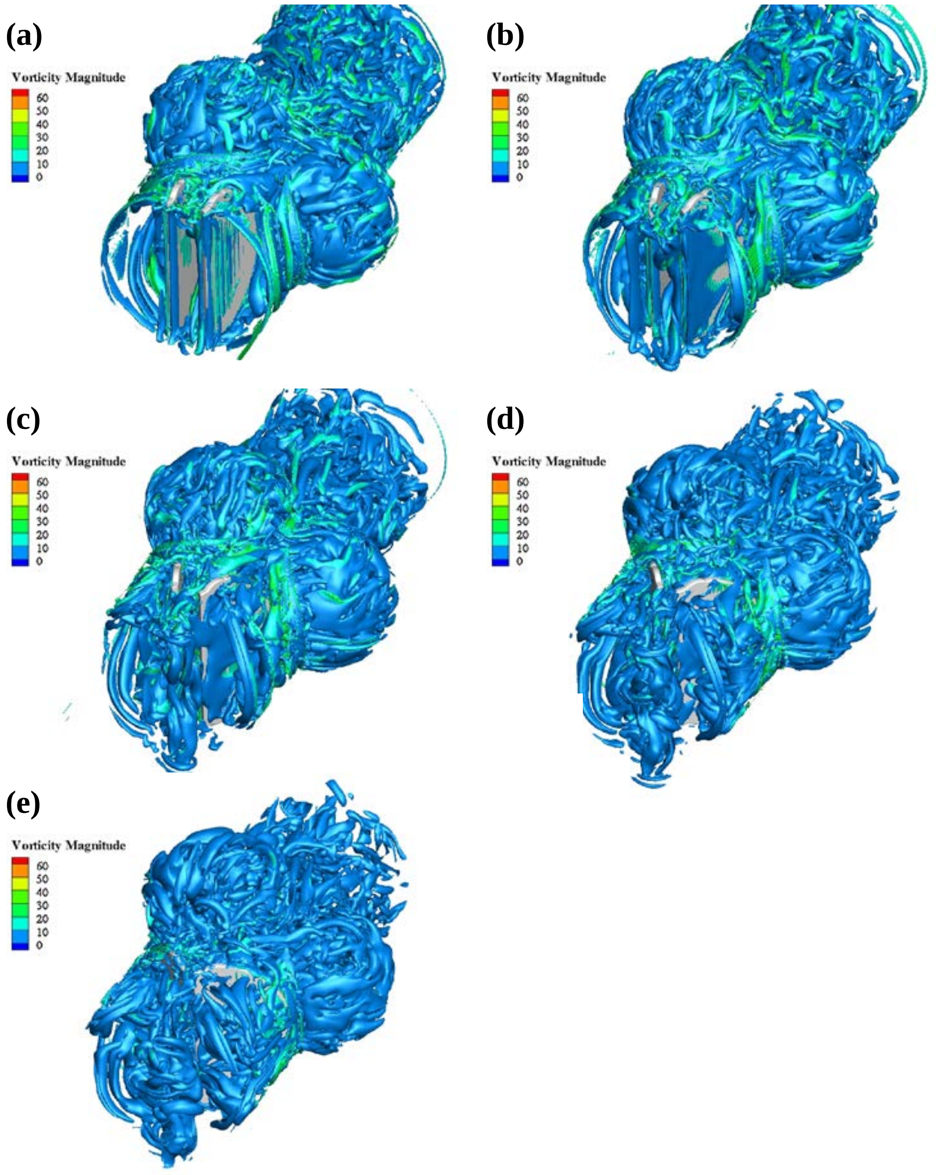


Figure 60 The closure of the leaflets during reverse flow in the Medtronic ADVANTAGE mechanical heart valve as indicated by λ-2 isocontours. The isocontours are colored by vorticity magnitude. The flow is shown at times of (a) 366 ms, (b) 383 ms, (c) 400 ms, (d) 420 ms, and (e) 435 ms.

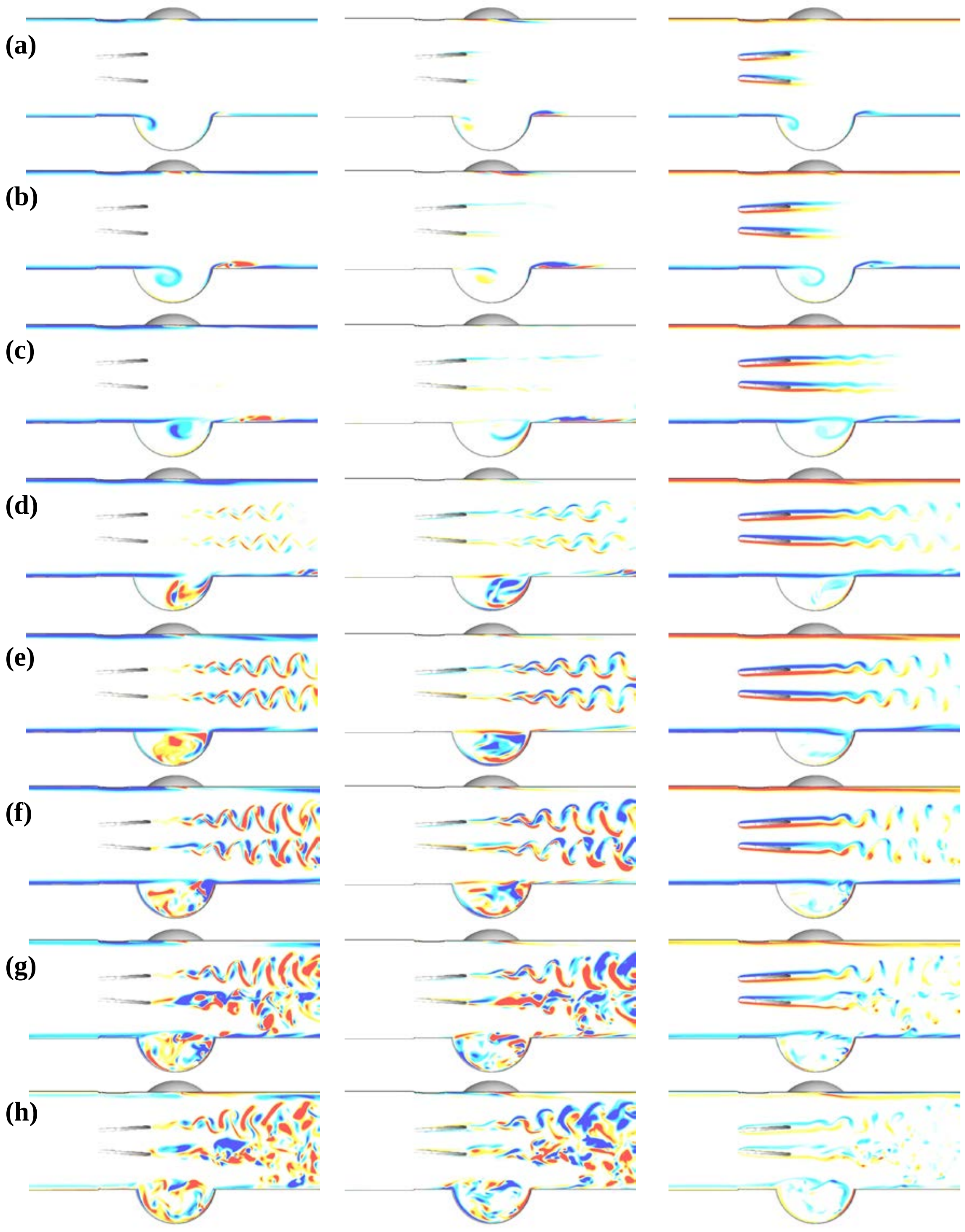


Figure 61 The forward flow in the Medtronic ADVANTAGE mechanical heart valve as indicated by slices of x-vorticity (left), y-vorticity (center) and z-vorticity (right) on the z = 1 plane. The flow is shown at times of (a) 96 ms, (b) 125 ms, (c) 162 ms, (d) 194 ms, (e) 228 ms, (f) 260 ms, (g) 297 ms, and (h) 333 ms.

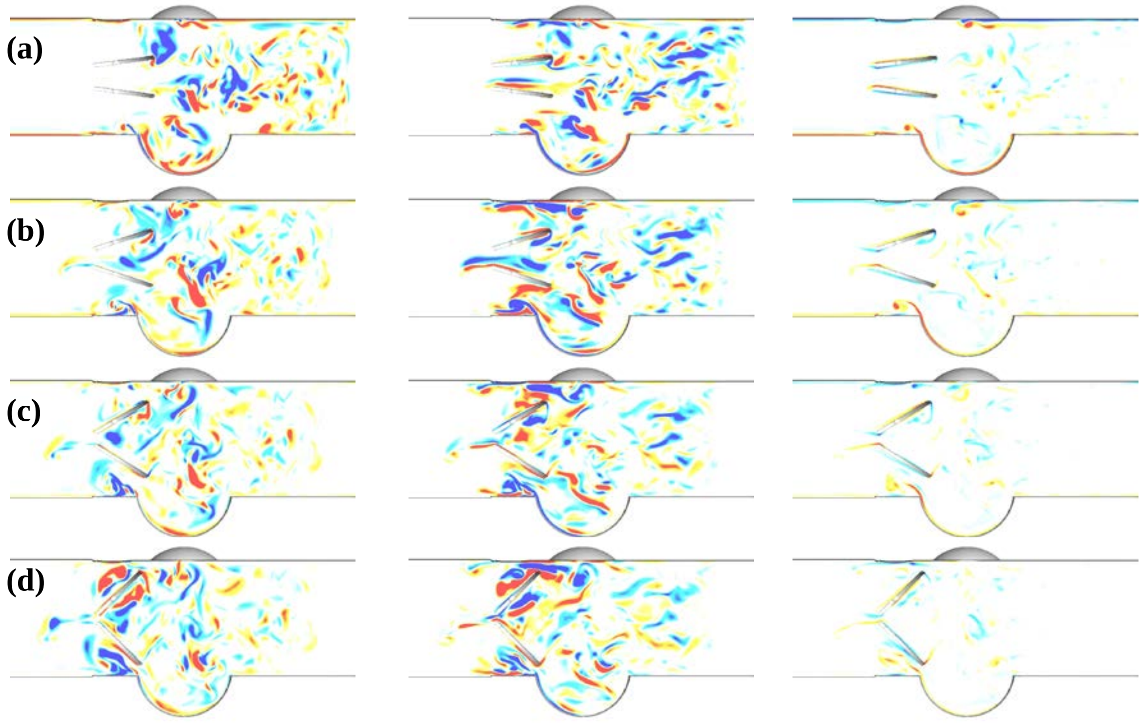


Figure 62 The reverse flow in the Medtronic ADVANTAGE mechanical heart valve as indicated by slices of x-vorticity (left), y-vorticity (center) and z-vorticity (right) on the z = 1 plane. The flow is shown at times of (a) 366 ms, (b) 383 ms, (c) 400 ms, and (d) 420 ms.

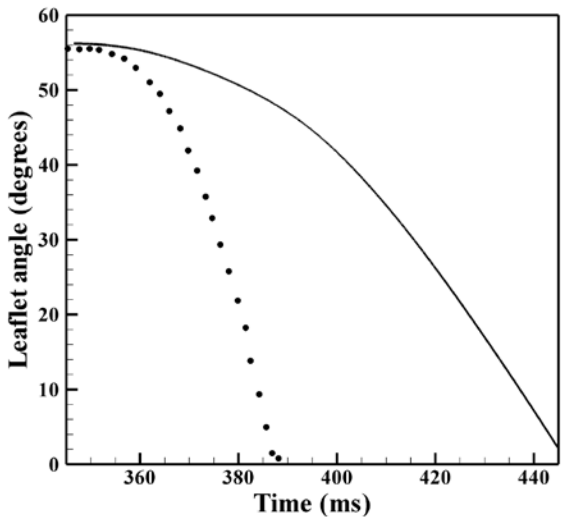


Figure 63 The angle of the valve leaflets versus time during valve closure for the current work (line) and [15] (circles). The closure time is approximately 100 ms in the current results.

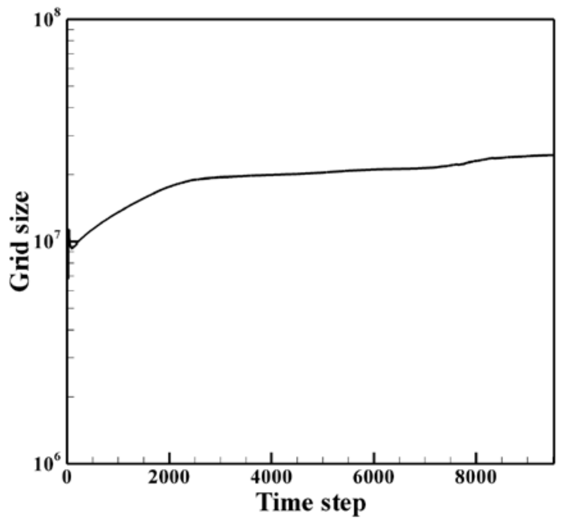


Figure 64 The size of the computational grid versus time step for the MHV simulation. The grid is initially approximately 10 million grid cells, and increases to nearly 30 million cells at the termination of the simulation. The start of valve closure is evidenced by the return to increasing behavior of the grid size around time step 7500.

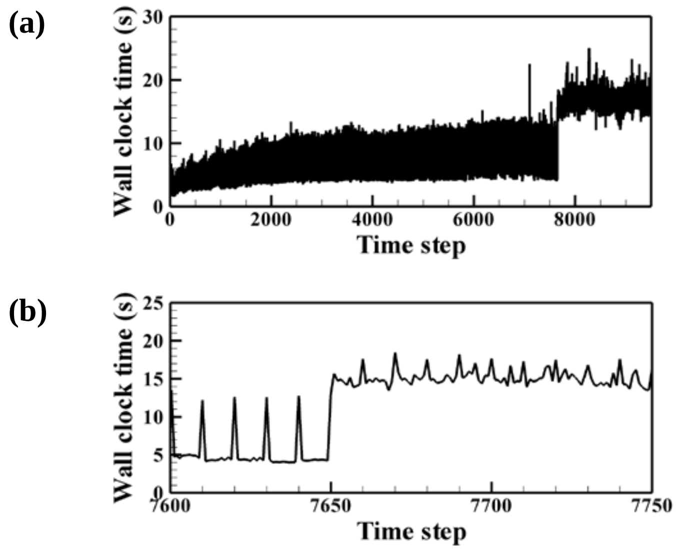


Figure 65 In (a), the wall clock time required to apply the flow solver versus time step is shown, while in (b), a zoomed in view shows the wall clock time behavior at the transition between the leaflets being stationary and moving.

CHAPTER 6

IMAGE TO COMPUTATION OF AN INTRACRANIAL ANEURYSM

6.1 Introduction

In the current chapter, the framework developed in the thesis to this point is extended to facilitate a direct image to computation technique. The central role of Cartesian grids and level set representations in the field of image processing make this extension very natural. The goal is to formulate a seamless and robust methodology to extract level set representations of boundaries directly from medical images, and then use the level set directly in the current Cartesian grid solver. When formulated in a single unifying framework, such an approach circumvents the need for the user to extract the geometry surface by hand, and perform the time intensive tasks of surface and volume mesh construction. Additionally, velocity boundary conditions can be obtained for moving objects from time-series of images through the application of an image processing technique known as optical flow. While not discussed here, this added benefit enhances the attraction of the current approach.

The first section of this chapter is a review of the image processing concept of segmentation which is the basis of constructing boundary representations from image data. An emphasis is placed on the active contours method, which determines the boundary of an object in an image through an energy minimization technique which is formulated in terms of a level set field. We then describe the specific approach taken in this work. Finally, we apply the proposed methodology to analyze the flow in an intracranial aneurysm where the detailed process is performed on a set of data obtained from CT imaging.

6.2 Segmentation in Image Processing

One of the fundamental processes of image-based modeling is segmentation. Segmentation is the act of computing a disjoint partitioning of image pixels in a manner that mimics human visualization [107, 108]. Pixels (or voxels, in 3D) in a digital image are grouped together, or segmented, based on properties like brightness intensity or texture, and are partitioned by different labels or by contours computed on the image. It is this partitioning that delineates object boundaries and makes surface modeling possible.

Segmentation methods can be divided into two categories; clustering and contour evolution. Clustering methods group image pixels/voxels based on some characteristic defining the cluster. It can be as simple as intensity thresholding, whereby representative voxels are selected based upon their grey scale levels; i.e. voxels with intensities falling between predetermined minimum and maximum threshold levels are retained and smoothed to yield a modeled surface. Alternatively, clustering may involve more complex statistical methods e.g. adaptive clustering [109, 110], which iteratively compares individual voxels with local properties of a set of neighborhoods, Ω_l, as well as the global properties of the image domain. By making use of local statistics, such cluster-based methods can be advantageous when it is desirable to segment images polluted with high levels of noise or a large global intensity variation. They can also easily be extended to include an arbitrary number of clusters, facilitating the classification of multiple imaged objects. The disadvantage of clustering methods is that they do not produce smooth object surfaces on their own. The grouping of voxels gives a pixelated representation that must be smoothed in a post-processing step. Additionally, the parallelization of such methods is not straight-forward, as the comparison region, Ω_1, may span several processes. Furthermore, as the comparison regions become coarser and coarser, the small nature of the problem may lead to severe load imbalance, similar to the issues experienced in the implementation of parallel multigrid schemes.

Unlike clustering, contour evolution segmentation methods separate images into smoothly delineated regions, usually through the solution of a partial differential equation derived using energy minimization techniques. Segmentation of the image domain proceeds from the placement of an arbitrary seed contour, which is designed to evolve from its initial state to a minimum energy state where the energy is a function of image characteristics like intensity, intensity gradient, etc. The clear advantage of contour evolution methods is that they produce smooth surfaces that require little or no post-processing. Parallel implementations are also quite straight-forward. However, they can be sensitive to noise since they have no intrinsic statistical noise removal capabilities. In addition, the final segmentation result can depend strongly on the number of objects of different intensity levels existing within the image domain. While some more recent developments have reduced this dependence by enabling multiple object classification to take place in the active contours framework [111], the technique for evolving multiple contours is less straightforward than cluster labeling.

Active contours methods are based on early applications of elastic deformation theory to computer vision [112], where contours are modeled as having elastic properties that serve to minimize the energy of mismatch between the evolving contour and the boundary of the imaged object. Mumford and Shah [113] applied the concept of active contours to the segmentation problem by proposing an energy functional that decomposes image domain Ω into piecewise smooth regions delineated by segmentation contour C:

$$E(f, C) = \mu L(C) + \lambda \int_{\Omega} (I_o - I)^2 d\Omega + \int_{\Omega \setminus C} |\nabla I|^2 d\Omega \ . \quad (6.1)$$

In (6.1), $L(C)$ is the total arc length of the segmentation contour and $\lambda > 0$, $\mu \geq 0$ are weighting parameters. The arc length penalty term ensures that segmentation contours follow object boundaries smoothly, while the second term quantifies the mismatch between a representative function I and the original underlying image intensity, I_o, and the third term constrains the spatial variation of I to be smooth within each segment. This allows for an image to be decomposed into segments that are discontinuous across contour boundaries while maintaining smoothness within each region.

Chan and Vese [114] later modified (6.1) to delineate images into piecewise constant regions of average intensity rather than piecewise smooth regions of slowly varying intensity, thus eliminating the need for a smoothness constraint. They also recast the energy minimization problem in terms of level sets, so that segmentation surfaces could be represented implicitly as zero level set isocontours. The modified formulation also made possible the addition of a second regularization term that restricts the area of a segment along with the original curve length restriction, giving energy functional

$$E(\phi, c_1, c_2) = \mu \int_{\Omega} |\nabla \mathcal{H}(\phi)| d\Omega + \nu \int_{\Omega} \mathcal{H}(\phi) d\Omega + \lambda_1 \int_{\Omega} (I_0 - c_1)^2 \mathcal{H}(\phi) d\Omega +$$

$$\lambda_2 \int_{\Omega} (I_0 - c_2)^2 \big(1 - \mathcal{H}(\phi)\big) d\Omega \ . \quad (6.2)$$

In (6.2), μ, ν, and λ_i are weighting parameters and $\mathcal{H}(\phi)$ is the Heaviside function of level set field ϕ, which can be approximated on a Cartesian mesh with grid spacing Δ as

$$\mathcal{H}_{\epsilon}(\phi) = \frac{1}{2}\left[1 + \frac{2}{\pi} \arctan\left(\frac{\phi}{\Delta}\right)\right] \ . \quad (6.3)$$

The constants c_i in (6.2) represent the average brightness intensity in each segmentation region i, and are defined as

$$c_1 = \frac{\int_\Omega I_0 \mathcal{H}_\epsilon(\phi)d\Omega}{\int_\Omega \mathcal{H}_\epsilon(\phi)d\Omega}, \qquad c_2 = \frac{\int_\Omega I_0\left(1-\mathcal{H}_\epsilon(\phi)\right)d\Omega}{\int_\Omega\left(1-\mathcal{H}_\epsilon(\phi)\right)d\Omega} . \tag{6.4}$$

Euler-Lagrange minimization of (6.2) leads to a level set evolution equation

$$\frac{d\phi}{dt} = \delta_\epsilon(\phi)\left[\mu\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - \nu - \lambda_1(I_0 - c_1)^2 + \lambda_2(I_0 - c_2)^2\right] , \tag{6.5}$$

which acts on the $\phi = 0$ segmentation contour by making use of a regularized delta function,

$$\delta_\epsilon(\phi) = \mathcal{H}_\epsilon{}'(\phi) = \frac{1}{\pi}\frac{\Delta}{\Delta^2+\phi^2} . \tag{6.6}$$

6.3 The Current Image-to-Computation Algorithm

In this thesis, we propose a method that combines pixel intensity thresh holding with active contour evolution to produce a smooth segmentation in a reliable and robust fashion. One disadvantage of active contours is its sensitivity to the presence of multiple regions of different brightness levels within the image domain. This sensitivity is due to the two-phase approach of (6.5); curve evolution is driven in large part by average intensity values calculated in the two regions of Ω. Thus, if the object of interest is bright, low-intensity objects in Ω will act to steer the evolving contour away from the desired boundary. We circumvent this problem by making use of intensity thresh holding where voxels with a brightness intensity in a certain range are identified. Active contour evolution then takes place on the set of voxels identified by the thresholding operation, similar in spirit to the mesh pruning algorithm applied to the Cartesian flow solver. This acts to eliminate much of the global dependency of the segmentation contour on imaged objects defined by intensity values that differ greatly from intensities in the region of interest (ROI).

We also incorporate an object-labeling scheme [115] that allows connected surfaces found during the segmentation process to be distinguished from each other within the context of smooth contours. The tagging operation proceeds by assigning all points a null object tag. The domain is scanned in its natural ordering until a point that is inside an object and has a null tag is found. All neighbors of this point which are inside the object are tagged with an object number of 1. Then, all the newly tagged points scan their neighbors, and tag the neighbors which are inside the object. The iterative neighbor tagging continues until no null-tagged neighbors are found. The procedure then continues by scanning for the next point in the natural ordering which is inside an object and has a null tag. The classification of the domain into a set of objects reduces the need for a closely cropped ROI, which in turn lessens the need for the addition of artificial extensions for inlets and outlets.

Evolution of (6.5) proceeds from an initial level set field that is usually defined as one or more closed contours with interior regions possessing level set values $\phi \leq 0$ and exterior regions having $\phi > 0$. For this we have adopted a multiple seed approach:

$$\phi(x, y, z) = \left|\prod_{i=1}^{3} \sin\left(\frac{\pi x_i}{d_s}\right)\right| - \frac{1}{2} \ , \qquad (6.7)$$

where x_i are spatial coordinates in each of the three dimensions of image Ω and d_s is level set seed diameter. The result is a field of $\phi = 0$ spheres of diameter d_s, spaced d_s apart, embedded in the image domain. This multiple seed initialization approach was taken as a way to handle generalized threshold-based domain pruning, as a large number of small seeds are in general more likely to end up being placed in regions of Ω that remain after thresholding than would a single initialization contour. In our experience, neither the speed nor the accuracy of active contours is sensitive to initial seed size, provided that the seeds are small enough to occupy (at least in part) all regions of the thresholded domain while still being large enough to define coherent interior and exterior regions. Given voxel grid spacing Δ, we have, in practice, found $d_s = 5\Delta$ to be a reliable initial seed diameter on a range of images.

The current algorithm for stationary boundaries can be summarized in the following set of steps:

1) Crop the image to isolate the region of interest.

2) Prune the domain by thresh holding intensity to remove outlying voxels from the segmentation process.

3) Initialize the pruned domain with spherical level set seeds.

4) Perform segmentation via active contours evolution on the pruned domain. Re-initialize the level set to make it a signed distance function.

5) Tag connected regions as distinct objects.

6) Interpolate the level set field of desired objects onto the base mesh of the flow solver.

7) Iteratively refine the level set field by interpolation during mesh refinement.

6.4 Direct Image-to-Computation of an Intracranial Aneurysm

In this section, we apply the proposed methodology to extract the flow characteristics of an intracranial aneurysm. The geometry of the aneurysm is obtained from a set of 128 CT scan images. In Figure 66, the steps outlined in the previous section are applied to the image data set. First, the approximate region of the aneurysm is extracted from the data set of the entire skull, as is shown in the inset figure in step 1. Thresholding is applied to the cropped image, and the voxels that fall in the specified range are shown in red in step 2 of Figure 66. The voxels identified in step 2 are then initialized with a set of uniformly spaced spherical level sets (red spheres in step 3 of Figure 66) of the form described in the previous section. The seeds are evolved by the active contours method (step 4), and the shape of the aneurysm is now clearly observed as the intersection of a bulge with 4 vessel segments.

However, the segmentation has identified a total of three surfaces, two of which are inconsequential to the current purposes. In order to extract the surface of interest, we apply the object tagging algorithm to the segmentation results, which in step 5 of Figure 66 has clearly identified three distinct connected regions, of which object 3 is the aneurysm that we wish to study. The level set field for object 3 is retained and interpolated onto the base mesh of the flow solver. Because the shape of the aneurysm is well described on the base mesh, a sufficient representation of the geometry is obtained on the refined mesh by interpolation from the image-based level set representation.

The final extracted aneurysm geometry is shown in Figure 67, along with the boundary conditions applied to each of the vessel segments emanating from the aneurysm bulb. The system contains two inlets and two outlets as shown. A uniform flow profile is assigned at both inlets with $U_{I2} = 0.9U_{I1}$, where U_{I1} and U_{I2} are the velocity magnitudes at the inlets. A convective outflow condition is applied at both outlets. The simulation is performed at a Reynolds number of 219, where the Reynolds number is defined based on the maximum width of inlet 1 and U_{I1}. The base mesh spacing is set to 0.2, and 3 additional levels of refinement are applied during the simulation. The solution is run for 20,000 time steps, which is sufficient to obtain the steady solution.

The streamlines in the aneurysm bulb at steady state are shown in Figure 68. The pressure contours are displayed on the streamlines, which gives an interpretation to how the flow is accelerating along the trajectory. The streamlines in the inlet sections of the flow indicate the presence of secondary flows in the curved vessels similar to that observed in the bent tube case presented in Chapter 4. The two inlet jets merge which induces a strongly tortuous streamline pattern in the aneurysm bulb. A separating manifold exists for which streamlines originating from an inlet in a certain region do not become entangled in the streamline bulb, but instead pass around the bulb and directly to an outlet.

The pressure and wall shear stress, defined as the magnitude of the tangent component of the viscous stress on the aneurysm surface, are shown in Figure 69 and Figure 70. It is observed that the pressure on the surface in the vicinity of the attachments of the inlet vessels to the bulb decreases quickly in response to the strong acceleration of the two inlet streams as they merge. A pressure maximum is observed on the concave intersection of outlet 2 with the bulb as the fluid must strongly change trajectory to flow into the outlet vessel. The shear stress attains a maximum at the shoulder region where the vessels meet the aneurysm bulb.

6.5 Conclusions

In this chapter, an image to computation algorithm which minimizes user intervention was proposed. The current methodology builds on the Cartesian grid and level set based framework developed in the rest of the thesis which makes the extension very natural. The algorithm was then applied to obtain the geometry describing an intracranial aneurysm and subsequently compute the steady flow in the aneurysm bulb. In this application, image segmentation was performed using the active contours algorithm, where the segmentation arises as the energy minimizing state of an elastic curve evolution equation.

This work is a first step in formulating a general approach for simulating moving systems from four-dimensional data sets. The geometry of the problem is computed according to the current technique. The additional component for moving boundary problems is obtaining a smooth boundary velocity. To this end, one can apply optical flow methods which obtain velocities at each pixel/voxel through considerations of physical laws governing the time-varying behavior of pixel intensity. With this approach in place, the current methodology provides a comprehensive and powerful approach to modeling biological fluid dynamics in patient specific cases.

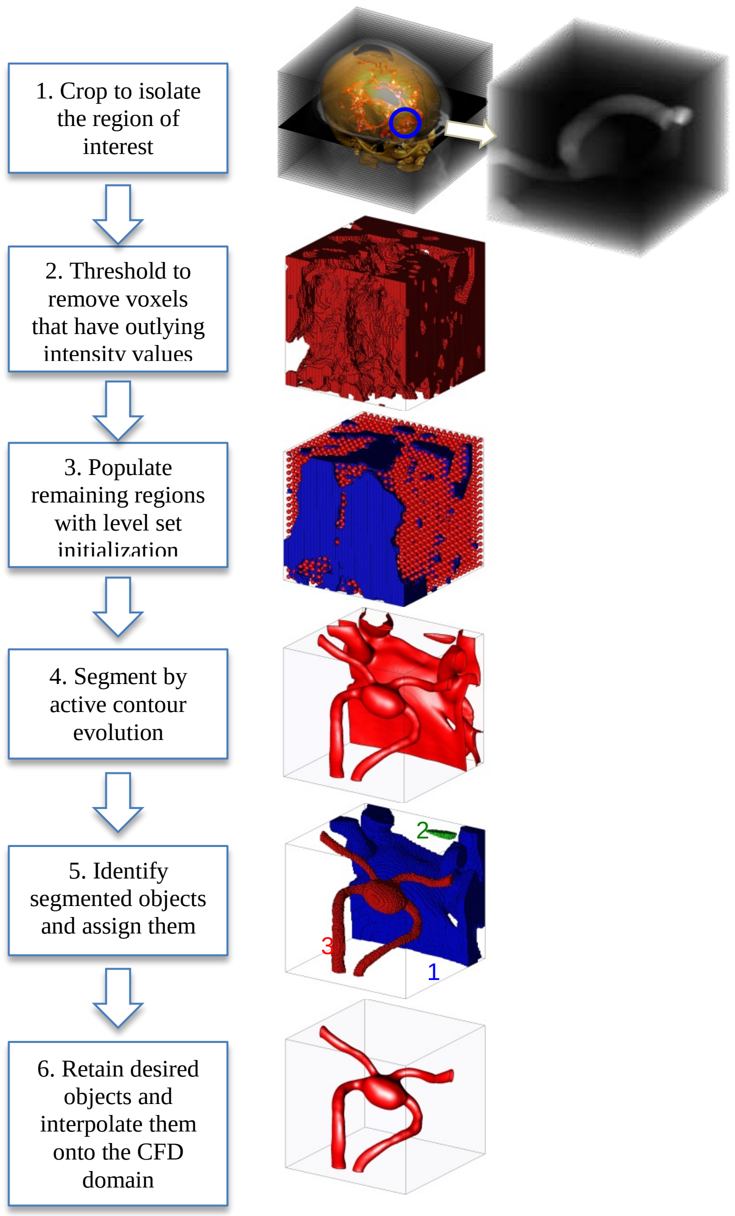

1. Crop to isolate the region of interest
2. Threshold to remove voxels that have outlying intensity values
3. Populate remaining regions with level set initialization
4. Segment by active contour evolution
5. Identify segmented objects and assign them
6. Retain desired objects and interpolate them onto the CFD domain
2
3
1

Figure 66 A flow chart depicting the current image-to-computation algorithm. In steps 1 and 2, pre-processing improves the robustness of the algorithm. In steps 3 and 4, the image is segmented which results immediately in level set representations. In step 5, connected regions are identified and given unique numbering. In step 6, the user selects the region of interest, and all other objects are discarded.

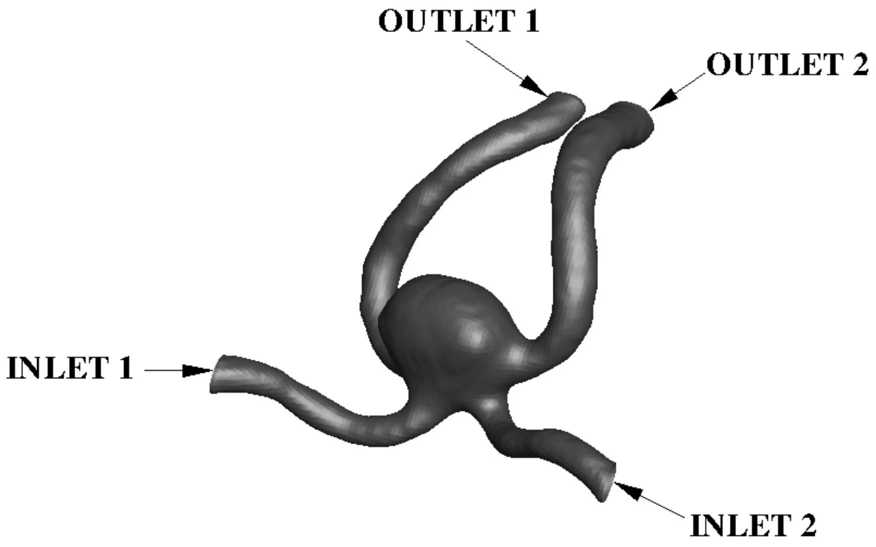


Figure 67 A schematic of the segmented intracranial aneurysm. The aneurysm bulb is attached to four vessels. Two vessels are treated as inlets, and two are specified as outlets as shown.

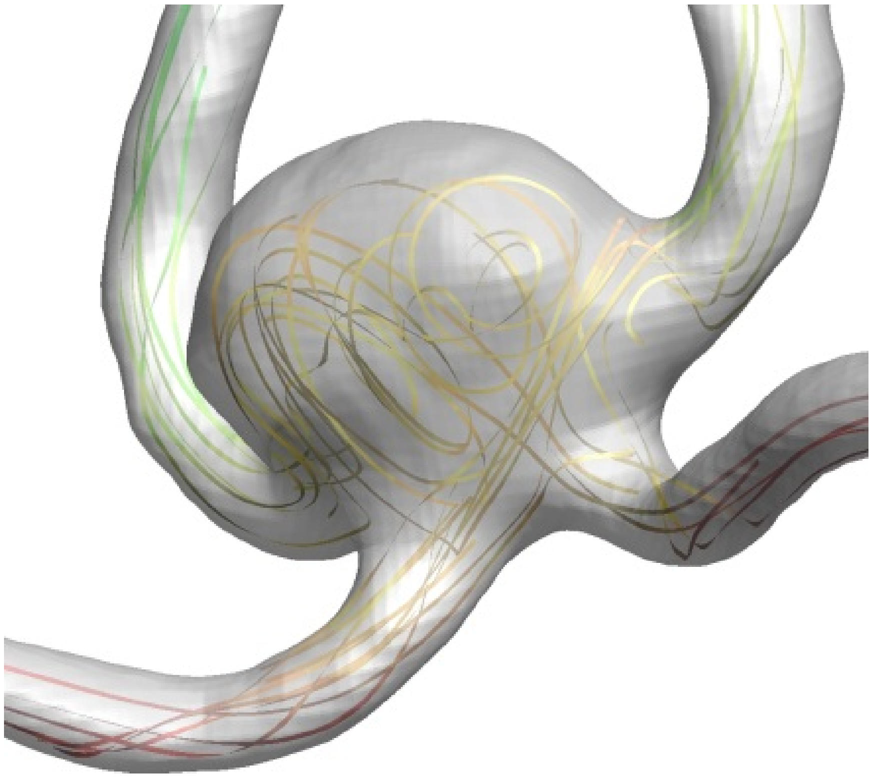


Figure 68 The flow pattern in the aneurysm is elucidated by plotting streamlines. The streamlines are colored by pressure value to indicate the acceleration behavior of the flow.

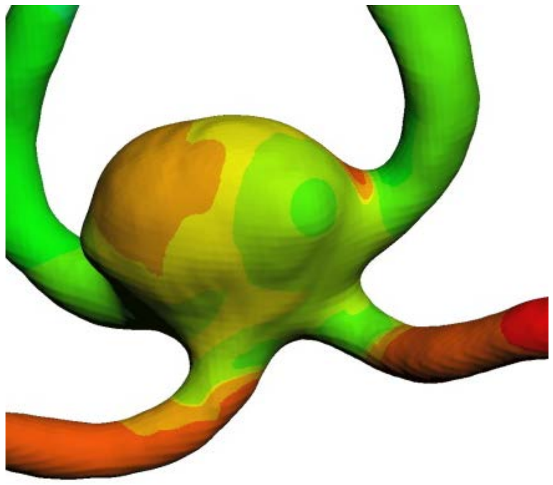


Figure 69 A plot of the surface pressure induced by flow in the intracranial aneurysm.

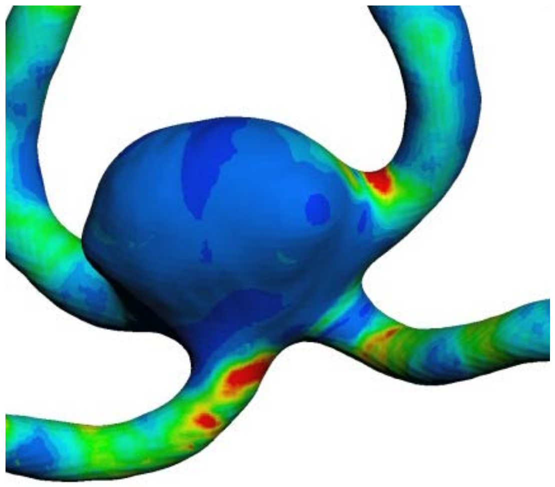


Figure 70 A plot of the wall shear stress induced by the flow intracranial aneurysm. The stress is a maximum at the shoulders of the geometry where the bulb meets the vessels.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Contributions of the Current Thesis

The current thesis has developed a massively parallel framework for efficiently solving complex incompressible flow problems with the presence of moving boundaries. In summary, the specific contributions of this thesis are:

1. A massively parallel adaptive grid framework has been developed which operates efficiently on tens of thousands of computational cores. This has been shown to facilitate the solution of problems where the computational grid is upwards of half of a billion grid cells in size.
2. A novel combination of a dynamic tubular grid storage scheme for the Cartesian base mesh with octree mesh refinement leads to an increased range of applicability of the current method to geometries where a large amount of Cartesian mesh would be extraneously placed outside the solution domain.
3. A novel hybridized least-squares ghost fluid method has been developed which is robust in the presence of complex geometries with both strong concave and convex curvatures. The hybridization of the scheme significantly reduces the magnitude of numerical pressure oscillations that are generated as an immersed boundary moves through the mesh.
4. A completely distributed narrow band level set field implementation was developed. The narrow band tube was stored using the memory efficient dynamic

tubular grid approach. The narrow band tubes can be adaptively repartitioned during the computation in an efficient manner.

5. A direct image to computation technique was developed by drawing on the connection between level sets and segmentation techniques of image analysis. The methodology was demonstrated through an analysis where the flow through an intracranial aneurysm was computed using level sets extracted from CT images.
6. The fully three-dimensional closure flow through a bi-leaflet mechanical heart valve was computed including the gap region between the leaflet edge and the valve housing. Both the forward and reverse flows were computed, but motion of the valves only occurred during the closure phase of the calculation.

7.2 Future Work

The framework developed in this thesis has been shown to allow one to solve very complex computationally challenging problems in an efficient manner. However, there are many avenues for future work, both in the development of the numerical package as well as the use of the framework to examine more complex problems.

1. The major thrust of the current work was the development of a solver scalable to tens of thousands of cores. However, partitioning of the mesh occurs at a very coarse-grained level. This removes a considerable level of control that may introduce difficulty in obtaining acceptable load-balanced partitions when many levels of refinement are used. Additionally, there exists the possibility that most of the ghost and hybrid cells may end up on a small number of partitions. As the least-squares solves can become computationally expensive in large numbers, this

may introduce a large load-imbalance and increase simulation times unnecessarily. To mitigate these problems, the option of partitioning at the leaf cell level should be implemented.

2. The current framework has required the development of a large amount of source code. A package of this magnitude must be easily maintainable and modifiable if it is to appeal to a large user base. LOCI is a framework that constructs program schedules from user-provided facts and rules in a way that the program is guaranteed to be internally consistent. This strategy provides a partial verification of program correctness and significantly reduces the risk of faulty results. The current techniques should be implemented in the LOCI framework to enhance the attractiveness and ease of use for future users.
3. The data structures used to construct and contain the adaptive mesh do not have optimal cache coherence behavior. To improve memory performance, the current fully-threaded octree should be re-implemented as a flat octree. Threaded trees can be reconstructed at times when neighbor finding is crucial and difficult, such as 2:1 balancing, and then subsequently destroyed.
4. To make the current mechanical heart valves simulation tractable on the available computational resources, the grid resolution was not able to adequately resolve the hinge region. To avoid numerical complications, the hinge region was slightly enlarged. The current package is capable of running on vastly greater computational resources than were available, and the current MHV problem should be recomputed using these resources where the hinge region is properly resolved. This type of resolution could require billions of grid points.

5. The current simulations were only able to be carried out to capture the closure dynamics of the mechanical heart valve, as the opening phase requires a strongly coupled fluid-structure interaction algorithm, which was beyond the scope of the present thesis. A strongly coupled model should be implemented and the simulation described in the previous task should be carried out with the proper leaflet dynamics.
6. The image to computation technique described in Chapter 6 has currently only been implemented for stationary boundaries. To extend the algorithm to moving boundaries, a method of extracting boundary velocities must be implemented. Such a scheme can be built around the optical flow technique. This technique uses rules governing the evolution of image intensity values to compute a velocity of each pixel. This allows one to extract the velocity of a surface by combining optical flow with image segmentation.

In conclusion, pELAFINT3D is a powerful framework which holds promise in studying many complex problems. The numerical schemes presented in this thesis are easily implemented, and the package has been validated against many benchmark problems. The large-scale computing facility of pELAFINT3D has been demonstrated, but there is much room for optimization. The Cartesian fabric of the package makes extensibility of the physics and numerics easy to accomplish, which makes the pELAFINT3D framework even more powerful.

REFERENCES

1. Lloyd-Jones, D., et al., *Heart disease and stroke statistics—2009 update.* Circulation, 2009. **119**(3): p. 480-486.

2. Chandran, K.B., S.E. Rittgers, and A.P. Yoganathan, *Biofluid mechanics: the human circulation*. 2007: CRC Press.

3. Korfer, R., et al., *The Worldwide Mid-Term Experience with the Medtronic ADVANTAGE? Bileaflet Mechanical Heart Valve.* JOURNAL OF HEART VALVE DISEASE, 2006. **15**(3): p. 404.

4. Bluestein, D., Y. Li, and I. Krukenkamp, *Free emboli formation in the wake of bi-leaflet mechanical heart valves and the effects of implantation techniques.* Journal of biomechanics, 2002. **35**(12): p. 1533-1540.

5. Bluestein, D., E. Rambod, and M. Gharib, *Vortex shedding as a mechanism for free emboli formation in mechanical heart valves.* TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF BIOMECHANICAL ENGINEERING, 2000. **122**(2): p. 125-134.

6. Cheng, R., Y. Lai, and K.B. Chandran, *Two-dimensional fluid-structure interaction simulation of bileaflet mechanical heart valve flow dynamics.* The Journal of heart valve disease, 2003. **12**(6): p. 772.

7. Krishnan, S., et al., *Two-dimensional dynamic simulation of platelet activation during mechanical heart valve closure.* Annals of biomedical engineering, 2006. **34**(10): p. 1519-1534.

8. Simon, H.A., et al., *Numerical Investigation of the Performance of Three Hinge Designs of Bileaflet Mechanical Heart Valves.* Annals of biomedical engineering, 2010. **38**(11): p. 3295-3310.

9. Quinlan, N.J. and P.N. Dooley, *Models of flow-induced loading on blood cells in laminar and turbulent flow, with application to cardiovascular device flow.* Annals of biomedical engineering, 2007. **35**(8): p. 1347-1356.

10. Dooley, P.N. and N.J. Quinlan, *Effect of eddy length scale on mechanical loading of blood cells in turbulent flow.* Annals of biomedical engineering, 2009. **37**(12): p. 2449-2458.

11. Ellis, J., T. Wick, and A. Yoganathan, *Prosthesis-induced hemolysis: mechanisms and quantification of shear stress.* The Journal of heart valve disease, 1998. **7**(4): p. 376.

12. Liu, J., P. Lu, and S. Chu, *Turbulence characteristics downstream of bileaflet aortic valve prostheses.* Journal of biomechanical engineering, 2000. **122**(2): p. 118.

13. Nobili, M., et al., *Numerical simulation of the dynamics of a bileaflet prosthetic heart valve using a fluid-structure interaction approach.* Journal of biomechanics, 2008. **41**(11): p. 2539-2550.

14. Bellofiore, A. and N.J. Quinlan, *High-Resolution Measurement of the Unsteady Velocity Field to Evaluate Blood Damage Induced by a Mechanical Heart Valve.* Annals of biomedical engineering, 2011. **39**(9): p. 2417-2429.

15. Borazjani, I., L. Ge, and F. Sotiropoulos, *Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies.* Journal of Computational Physics, 2008. **227**(16): p. 7587-7620.

16. Borazjani, I., L. Ge, and F. Sotiropoulos, *High-Resolution Fluid–Structure Interaction Simulations of Flow Through a Bi-Leaflet Mechanical Heart Valve in an Anatomic Aorta.* Annals of biomedical engineering, 2010. **38**(2): p. 326-344.

17. Ge, L. and F. Sotiropoulos, *A numerical method for solving the 3D unsteady incompressible Navier-Stokes equations in curvilinear domains with complex immersed boundaries.* Journal of Computational Physics, 2007. **225**(2): p. 1782-1809.

18. Chandran, K., C. Lee, and L. Chen, *Pressure field in the vicinity of mechanical valve occluders at the instant of valve closure: correlation with cavitation initiation.* The Journal of heart valve disease, 1994. **3**: p. S65.

19. Cheng, R., Y.G. Lai, and K.B. Chandran, *Three-dimensional fluid-structure interaction simulation of bileaflet mechanical heart valve flow dynamics.* Annals of biomedical engineering, 2004. **32**(11): p. 1471-1483.

20. Taylor, C.A., T.J.R. Hughes, and C.K. Zarins, *Effect of exercise on hemodynamic conditions in the abdominal aorta.* Journal of vascular surgery, 1999. **29**(6): p. 1077-1089.

21. Zhang, Y., et al., *Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow.* Computer Methods in Applied Mechanics and Engineering, 2007. **196**(29): p. 2943-2959.

22. Pekkan, K., et al., *Physics-driven CFD modeling of complex anatomical cardiovascular flows—A TCPC case study.* Annals of biomedical engineering, 2005. **33**(3): p. 284-300.

23. Ferziger, J.H. and M. Perić, *Computational methods for fluid dynamics*. Vol. 2. 1999: Springer Berlin etc.

24. de Zélicourt, D.A., et al., *Individualized computer-based surgical planning to address pulmonary arteriovenous malformations in patients with a single ventricle with an interrupted inferior vena cava and azygous continuation.* The Journal of thoracic and cardiovascular surgery, 2011. **141**(5): p. 1170-1177.

25. Tseng, Y.H. and J.H. Ferziger, *A ghost-cell immersed boundary method for flow in complex geometry*. Journal of Computational Physics, 2003. **192**(2): p. 593-623.

26. Mittal, R. and G. Iaccarino, *Immersed boundary methods.* Annu. Rev. Fluid Mech., 2005. **37**: p. 239-261.

27. Sethian, J.A., *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. 1999: Cambridge Univ Pr.

28. Burstedde, C., L.C. Wilcox, and O. Ghattas, *p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees.* Submitted to SIAM Journal on Scientific Computing, 2010.

29. Wissink, A.M., et al., *Parallel multi-physics AMR applications using the SAMRAI Library*, in *Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande*2001, ACM: Palo Alto, California, United States. p. 184.

30. Luo, H., et al., *On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries.* Computers &amp; Fluids, 2012. **56**(0): p. 61-76.

31. Peskin, C.S., *Flow patterns around heart valves: a numerical method.* Journal of Computational Physics, 1972. **10**(2): p. 252-271.

32. Peskin, C.S., *The immersed boundary method.* Acta numerica, 2002. **11**(1): p. 479-517.

33. Peskin, C.S., *Numerical analysis of blood flow in the heart.* Journal of Computational Physics, 1977. **25**(3): p. 220-252.

34. McQueen, D.M. and C.S. Peskin, *Heart simulation by an immersed boundary method with formal second-order accuracy and reduced numerical viscosity.* Mechanics for a New Mellennium, 2002: p. 429-444.

35. Bagchi, P., P.C. Johnson, and A.S. Popel, *Computational fluid dynamic simulation of aggregation of deformable cells in a shear flow.* Journal of biomechanical engineering, 2005. **127**: p. 1070.

36. Eggleton, C. and A. Popel, *Large deformation of red blood cell ghosts in a simple shear flow.* Physics of Fluids, 1998. **10**: p. 1834.

37. Marella, S.V. and H. Udaykumar, *Computational analysis of the deformability of leukocytes modeled with viscous and elastic structural components.* Physics of Fluids, 2004. **16**: p. 244.

38. Leveque, R.J. and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources.* SIAM Journal on Numerical Analysis, 1994: p. 1019-1044.

39. Lai, M.C. and C.S. Peskin, *An immersed boundary method with formal second-order accuracy and reduced numerical viscosity.* Journal of Computational Physics, 2000. **160**(2): p. 705-719.

40. Roma, A.M., C.S. Peskin, and M.J. Berger, *An adaptive version of the immersed boundary method.* Journal of Computational Physics, 1999. **153**(2): p. 509-534.

41. Glowinski, R., et al., *A distributed Lagrange multiplier/fictitious domain method for flows around moving rigid bodies: application to particulate flow.* International journal for numerical methods in fluids, 1999. **30**(8): p. 1043-1066.

42. Wang, X. and W.K. Liu, *Extended immersed boundary method using FEM and RKPM.* Computer Methods in Applied Mechanics and Engineering, 2004. **193**(12-14): p. 1305-1321.

43. Balaras, E., *Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations.* Computers & Fluids, 2004. **33**(3): p. 375-404.

44. Kim, J., D. Kim, and H. Choi, *An immersed-boundary finite-volume method for simulations of flow in complex geometries.* Journal of Computational Physics, 2001. **171**(1): p. 132-150.

45. Fadlun, E., et al., *Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations.* Journal of Computational Physics, 2000. **161**(1): p. 35-60.

46. Lai, M.C. and Z. Li, *A remark on jump conditions for the three-dimensional Navier-Stokes equations involving an immersed moving membrane.* Applied mathematics letters, 2001. **14**(2): p. 149-154.

47. Li, Z. and M.C. Lai, *The immersed interface method for the Navier–Stokes equations with singular forces.* Journal of Computational Physics, 2001. **171**(2): p. 822-842.

48. Fedkiw, R.P., et al., *A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method).* Journal of Computational Physics, 1999. **152**(2): p. 457-492.

49. Gibou, F., et al., *A second-order-accurate symmetric discretization of the Poisson equation on irregular domains.* Journal of Computational Physics, 2002. **176**(1): p. 205-227.

50. Mittal, R., et al., *A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries.* Journal of Computational Physics, 2008. **227**(10): p. 4825-4852.

51. Udaykumar, H., et al., *A sharp interface Cartesian grid method for simulating flows with complex moving boundaries.* Journal of Computational Physics, 2001. **174**(1): p. 345-380.

52. Udaykumar, H., R. Mittal, and P. Rampunggoon, *Interface tracking finite volume method for complex solid–fluid interactions on fixed meshes.* Communications in numerical methods in engineering, 2002. **18**(2): p. 89-97.

53. Udaykumar, H., S. Marella, and S. Krishnan, *Sharp-interface simulation of dendritic growth with convection: benchmarks.* International journal of heat and mass transfer, 2003. **46**(14): p. 2615-2627.

54. Chessa, J. and T. Belytschko, *An extended finite element method for two-phase fluids: Flow simulation and modeling.* Journal of Applied Mechanics, 2003. **70**(1): p. 10-17.

55. Chessa, J., P. Smolinski, and T. Belytschko, *The extended finite element method (XFEM) for solidification problems.* International Journal for Numerical Methods in Engineering, 2002. **53**(8): p. 1959-1977.

56. Dolbow, J., N. Moes, and T. Belytschko, *An extended finite element method for modeling crack growth with frictional contact.* Computer Methods in Applied Mechanics and Engineering, 2001. **190**(51-52): p. 6825-6846.

57. Ingram, D.M., D.M. Causon, and C.G. Mingham, *Developments in Cartesian cut cell methods.* Mathematics and Computers in Simulation, 2003. **61**(3): p. 561-572.

58. Hartmann, D., M. Meinke, and W. Schröder, *A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids.* Computer Methods in Applied Mechanics and Engineering, 2011. **200**(9): p. 1038-1052.

59. Murman, S.M., M.J. Aftosmis, and M.J. Berger, *Implicit approaches for moving boundaries in a 3-D Cartesian method.* AIAA paper, 2003. **1119**: p. 2003.

60. Gibou, F. and R. Fedkiw, *A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem.* Journal of Computational Physics, 2005. **202**(2): p. 577-601.

61. Lee, J., et al., *Sources of spurious force oscillations from an immersed boundary method for moving-body problems.* J. Comput. Phys., 2011. **230**(7): p. 2677-2695.

62. Seo, J.H. and R. Mittal, *A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations.* Journal of Computational Physics, 2011.

63. Sethian, J.A., *Theory, algorithms, and applications of level set methods for propagating interfaces.* Acta numerica, 1996. **5**(1): p. 309-395.

64. Jiang, G.S. and D. Peng, *Weighted ENO schemes for Hamilton-Jacobi equations.* SIAM Journal on Scientific computing, 2000. **21**(6): p. 2126-2143.

65. Losasso, F., R. Fedkiw, and S. Osher, *Spatially adaptive techniques for level set methods and incompressible flow.* Computers & Fluids, 2006. **35**(10): p. 995-1010.

66. Adalsteinsson, D., *A fast level set method for propagating interfaces*, 1994, University of California.

67. Nielsen, M.B. and K. Museth, *Dynamic Tubular Grid: An efficient data structure and algorithms for high resolution level sets.* Journal of Scientific Computing, 2006. **26**(3): p. 261-299.

68. Sussman, M. and E. Fatemi, *An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow.* SIAM Journal on Scientific computing, 1999. **20**(4): p. 1165-1191.

69. Choi, H. and P. Moin, *Effects of the computational time step on numerical solutions of turbulent flow.* Journal of Computational Physics, 1994. **113**(1): p. 1-4.

70. Yue, W., C.L. Lin, and V.C. Patel, *Numerical simulation of unsteady multidimensional free surface motions by level set method.* International journal for numerical methods in fluids, 2003. **42**(8): p. 853-884.

71. Brown, D.L., R. Cortez, and M.L. Minion, *Accurate projection methods for the incompressible Navier-Stokes equations.* Journal of Computational Physics, 2001. **168**(2): p. 464-499.

72. Marella, S., et al., *Sharp interface Cartesian grid method I: an easily implemented technique for 3D moving boundary computations.* Journal of Computational Physics, 2005. **210**(1): p. 1-31.

73. Baek, H. and G.E. Karniadakis, *A convergence study of a new partitioned fluid-structure interaction algorithm based on fictitious mass and damping.* J. Comput. Phys., 2012. **231**(2): p. 629-652.

74. Lorensen, W.E. and H.E. Cline. *Marching cubes: A high resolution 3D surface construction algorithm*. 1987. ACM.

75. Gropp, W., E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message passing interface*. 1999.

76. Berger, M.J. and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics.* Journal of Computational Physics, 1989. **82**(1): p. 64-84.

77. Berger, M.J. and R.J. LeVeque, *Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems.* SIAM Journal on Numerical Analysis, 1998. **35**(6): p. 2298-2316.

78. MacNeice, P., et al., *PARAMESH: A parallel adaptive mesh refinement community toolkit.* Computer physics communications, 2000. **126**(3): p. 330-354.

79. Popinet, S., *Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries.* Journal of Computational Physics, 2003. **190**(2): p. 572-600.

80. Zélicourt, D., et al., *Flow simulations in arbitrarily complex cardiovascular anatomies–An unstructured Cartesian grid approach.* Computers & Fluids, 2009. **38**(9): p. 1749-1762.

81. Balay, S., et al., *PETSc Users Manual Revision 3.2.* 2011.

82. Karypis, G., *Metis/Parmetis web page, University of Minnesota, 2008*.

83. Tu, T. and D. O'Hallaron, *Balance refinement of massive linear octrees*, 2004, Technical Report CMU-CS-04-129, Carnegie Mellon School of Computer Science.

84. Tu, T., D.R. O'Hallaron, and O. Ghattas. *Scalable parallel octree meshing for terascale applications*. 2005. IEEE.

85. Isaac, T., C. Burstedde, and O. Ghattas. *Low-cost parallel algorithms for 2: 1 octree balance*. in *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*. 2012. IEEE.

86. Krishnan, S., *An adaptively refined Cartesian grid method for moving boundary problems applied to biomedical systems.* 2006.

87. Saad, Y. and Y. Saad, *Iterative methods for sparse linear systems*. Vol. 620. 1996: PWS publishing company Boston.

88. Chow, E., A.J. Cleary, and R.D. Falgout. *Design of the hypre preconditioner library*. in *SIAM Workshop on Object Oriented Methods for Inter-operable Scientific and Engineering Computing*. 1998.

89. Falgout, R., J. Jones, and U. Yang, *The design and implementation of hypre, a library of parallel high performance preconditioners.* Numerical solution of partial differential equations on parallel computers, 2006: p. 267-294.

90. Laboratory, O.R.N., 2012.

91. Zalesak, S.T., *Fully multidimensional flux-corrected transport algorithms for fluids.* Journal of Computational Physics, 1979. **31**(3): p. 335-362.

92. Koumoutsakos, P. and A. Leonard, *High-resolution simulations of the flow around an impulsively started cylinder using vortex methods.* Journal of Fluid Mechanics, 1995. **296**(1): p. 1-38.

93. Johnson, T. and V. Patel, *Flow past a sphere up to a Reynolds number of 300.* Journal of Fluid Mechanics, 1999. **378**(1): p. 19-70.

94. Mittal, R. and F. Najjar, *Vortex dynamics in the sphere wake.* Red, 1999. **2**: p. 2.

95. Van de Vosse, F., et al., *A finite element analysis of the steady laminar entrance flow in a 90 curved tube.* International journal for numerical methods in fluids, 2005. **9**(3): p. 275-287.

96. Chisari, N., G. Artana, and D. Sciamarella, *Vortex dipolar structures in a rigid model of the larynx at flow onset.* Experiments in Fluids, 2011. **50**: p. 397-406.

97. Shim, E. and K. Chang, *Three-dimensional vortex flow past a tilting disc valve using a segregated finite element scheme.* Comput Dyn J, 1994. **3**: p. 205.

98. Shim, E.B. and K.S. Chang, *Numerical analysis of three-dimensional Björk–Shiley valvular flow in an aorta.* Journal of biomechanical engineering, 1997. **119**: p. 45.

99. Kiris, C., et al., *Computational approach for probing the flow through artificial heart devices.* Journal of biomechanical engineering, 1997. **119**: p. 452.

100. King, M., et al., *A three-dimensional, time-dependent analysis of flow through a bileaflet mechanical heart valve: comparison of experimental and numerical results.* Journal of biomechanics, 1996. **29**(5): p. 609-618.

101. Ge, L., et al., *Numerical simulation of flow in mechanical heart valves: grid resolution and the assumption of flow symmetry.* Journal of biomechanical engineering, 2003. **125**: p. 709.

102. Tai, C., K. Liew, and Y. Zhao, *Numerical simulation of 3D fluid-structure interaction flow using an immersed object method with overlapping grids.* Computers & structures, 2007. **85**(11-14): p. 749-762.

103. Shu, M.C.S., et al., *Flow Characterization of the ADVANTAGE" and St. Jude Medical" Bileaflet Mechanical Heart Valves.* JOURNAL OF HEART VALVE DISEASE, 2004. **13**(5): p. 814-822.

104. Dasi, L., et al., *Vorticity dynamics of a bileaflet mechanical heart valve in an axisymmetric aorta.* Physics of Fluids, 2007. **19**: p. 067105.

105. Jeong, J. and F. Hussain, *On the identification of a vortex.* Journal of Fluid Mechanics, 1995. **285**(69): p. 69-94.

106. Grigioni, M., et al., *Three-dimensional numeric simulation of flow through an aortic bileaflet valve in a realistic model of aortic root.* ASAIO journal, 2005. **51**(3): p. 176-183.

107. Brice, C.R. and C.L. Fennema, *Scene analysis using regions.* Artificial intelligence, 1970. **1**(3): p. 205-226.

108. Pal, N.R. and S.K. Pal, *A review on image segmentation techniques.* Pattern recognition, 1993. **26**(9): p. 1277-1294.

109. Pappas, T.N., *An adaptive clustering algorithm for image segmentation.* Signal Processing, IEEE Transactions on, 1992. **40**(4): p. 901-914.

110. Ashton, E.A. and K.J. Parker, *Multiple resolution Bayesian segmentation of ultrasound images.* Ultrasonic imaging, 1995. **17**(4): p. 291-304.

111. Vese, L.A. and T.F. Chan, *A multiphase level set framework for image segmentation using the Mumford and Shah model.* International Journal of Computer Vision, 2002. **50**(3): p. 271-293.

112. Terzopoulos, D., et al. *Elastically deformable models*. in *ACM Siggraph Computer Graphics*. 1987. ACM.

113. Mumford, D. and J. Shah, *Optimal approximations by piecewise smooth functions and associated variational problems.* Communications on pure and applied mathematics, 2006. **42**(5): p. 577-685.

114. Chan, T.F. and L.A. Vese, *Active contours without edges.* Image Processing, IEEE Transactions on, 2001. **10**(2): p. 266-277.

115. Herrmann, M., *A parallel Eulerian interface tracking/Lagrangian point particle multi-scale coupling procedure.* Journal of Computational Physics, 2010. **229**(3): p. 745-759.