RADIAL BASIS FUNCTION-BASED NUMERICAL METHODS FOR THE SIMULATION OF PLATELET AGGREGATION

by

Varun Shankar

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

August 2014

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Varun Shankar**

has been approved by the following supervisory committee members:

Robert M. Kirby	, Chair	**04/18/2014** Date Approved
Aaron L. Fogelson	, Member	**04/18/2014** Date Approved
Christopher R. Johnson	, Member	**04/18/2014** Date Approved
Adam W. Bargteil	, Member	**04/18/2014** Date Approved
Grady B. Wright	, Member	**04/22/2014** Date Approved

and by **Ross T. Whitaker** , Chair/Dean of

the Department/College/School of **Computing**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Platelet aggregation, an important part of the development of blood clots, is a complex process involving both mechanical interaction between platelets and blood, and chemical transport on and off the surfaces of those platelets. Radial Basis Function (RBF) interpolation is a meshfree method for the interpolation of multidimensional scattered data, and therefore well-suited for the development of meshfree numerical methods. This dissertation explores the use of RBF interpolation for the simulation of both the chemistry and mechanics of platelet aggregation.

We first develop a parametric RBF representation for closed platelet surfaces represented by scattered nodes in both two and three dimensions. We compare this new RBF model to Fourier models in terms of computational cost and errors in shape representation. We then augment the Immersed Boundary (IB) method, a method for fluid-structure interaction, with our RBF geometric model. We apply the resultant method to a simulation of platelet aggregation, and present comparisons against the traditional IB method.

We next consider a two-dimensional problem where platelets are suspended in a stationary fluid, with chemical diffusion in the fluid and chemical reaction-diffusion on platelet surfaces. To tackle the latter, we propose a new method based on RBF-generated finite differences (RBF-FD) for solving partial differential equations (PDEs) on surfaces embedded in 2D domains. To robustly tackle the former, we remove a limitation of the Augmented Forcing method (AFM), a method for solving PDEs on domains containing curved objects, using RBF-based symmetric Hermite interpolation.

Next, we extend our RBF-FD method to the numerical solution of PDEs on surfaces embedded in 3D domains, proposing a new method of stabilizing RBF-FD discretizations on surfaces. We perform convergence studies and present applications motivated by biology.

We conclude with a summary of the thesis research and present an overview of future research directions, including spectrally-accurate projection methods, an extension of the Regularized Stokeslet method, RBF-FD for variable-coefficient diffusion, and boundary conditions for RBF-FD.

I lovingly dedicate this dissertation to my wife, Dr. Divya Raghavan, for her unwavering love, patience, encouragement, and support. She is my best friend and my first love.

I also dedicate this dissertation to my amazing parents, Latha and Ramalingam, who always listened, never judged, and continue to inspire.

CONTENTS

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to express my deepest gratitude to my advisor, Professor Robert M. Kirby, for being the most intelligent, perceptive, and compassionate mentor a student could ask for, for teaching me how to be a scientist, and for "going meta" and explaining to me his methodology for advising students. I would like to thank my co-advisor, Professor Aaron Fogelson, for spending huge amounts of time bouncing around ideas with me, for his wry humour, his wisdom, and his support. I would also like to thank Professor Grady Wright, who has been a third mentor to me over the past few years, for his guidance and support and for showing me the intricacies and the beauty of RBFs. I thank Professor Adam Bargteil for spending months answering my questions on physical simulation and for being a great mentor when I really needed one. I thank Professor Chris Johnson for listening and offering comments on my research. I also thank Professor Sarah Olson for her excellent advice and support.

Second, I wish to acknowledge the different sources of financial support that I received over the course of my studies. This research was supported by a National Institutes of General Medical Sciences grant (NIGMS R01-GM090203). I truly appreciate the investment made by NIGMS (and, more broadly speaking, the United States) in training a foreign citizen for so many years, and shall endeavor to give back to this country by teaching at the University level and collaborating with researchers here. I am grateful to the Bharat Petroleum Corporation Ltd. (BPCL) for awarding me the Bharat Petroleum Scholarship for Higher Studies; this scholarship partially supported me during the first two years of grad school. I am also grateful to have been awarded the TataChem Golden Jubilee Foundation Scholarship, which assisted me in coming abroad to study.

I also express my gratitude to Konstantin Shkurko, Dan Maljovec, Scott Ruhnau, Mark Kim, and Jon Bronson for their support and friendship over the past few years.

Lastly, I wish to thank my wife and my parents for all the support, joy, and love they have given me for the past ten years. I also thank the rest of my family for their support and encouragement, and for being the coolest family anyone could ask for.

CHAPTER 1

INTRODUCTION

The goal of this dissertation is to develop efficient numerical methods based on Radial Basis Function interpolation for the simulation of models of important subprocesses of platelet aggregation. The thesis underlying this work, then, is that Radial Basis Function interpolation forms an excellent foundation for the development of numerical methods in multiple dimensions, especially in the complex geometries and rheologies arising from the study of biological systems. This work was done under the guidance of Professor Robert M. Kirby and Professor Aaron L. Fogelson, and in collaboration with Professor Grady B. Wright. Each chapter of the dissertation will indicate or reference the published or submitted journal article that originally documented the research which the chapter discusses.

Intravascular blood clots (thrombi) are initiated by damage to the endothelial cell lining of a blood vessel and involve the formation of clumps of cells intermixed with a fibrous protein gel on the damaged surface. The cells involved in this process are platelets. This introduction will present the background material for the dissertation by first introducing the mechanics of platelet aggregation and the primary tool for the simulation of these mechanics; next, we will discuss the chemistry of platelet aggregation, and the state of the art in simulating those dynamics. We will then present an overview of Radial Basis Function (RBF) interpolation, the primary tool in this dissertation for developing and/or extending the current state of the art; this overview will attempt to outline the important theoretical and algorithmic developments related to RBF interpolation. Finally, we will discuss the current state of the art in RBF-based numerical methods for the solution of PDEs. At the appropriate points in the narrative, we will allude to our extensions to the current state of the art, with each major extension being documented in its own chapter. We will conclude the introduction by presenting an overview of the remainder of the dissertation.

1.1 Overview

In general, this introduction and this dissertation focus on two specific aspects of platelet aggregation: the mechanics and the chemistry. Both problems are essentially fluid-structure interaction problems, with platelets deforming in response to fluid flow and in turn affecting the fluid as they deform, and with chemistry in the fluid and on the platelet surfaces dependent on the geometry of the domain (including the platelets in the domain). We focus on the Immersed Boundary (IB) method for platelet mechanics and the Augmented Forcing method (AFM) for platelet chemistry; our focus on the IB method and the AFM is due to the fact that the implementations of current platelet aggregation and chemistry models depend on and use these two methods. However, to gain an understanding of the alternatives, we will now present a brief overview of some recent fluid-structure interaction methods for moving objects on irregular domains from the literature.

A specific type of Finite Element Method (FEM) has recently been developed to handle fluid-structure interactions [96]. This so-called Immersed Finite Element (IFE) method can be viewed as a generalization of the IB method to use Finite Elements for both the solid and fluid representations. A unified representation was thus developed for both the solid and the fluid using the weak formulation of the Navier-Stokes equations. To delineate the Lagrangian solid from the Eulerian fluid, different velocities are used for each and they are coupled using a Reproducing Kernel Particle Method (RKPM) delta function. This method is quite promising and continues to be researched. In a different paradigm, researchers have investigated a unified Eulerian description for fluid-structure interactions [79]. A volume fraction formulation is used to keep track of solid and fluid separately, but the entire system, solid immersed in the fluid, is treated as an incompressible continuum. A stress balance equation is written for both solid and fluid stresses. The left Cauchy-Green deformation tensor for the solid is updated (the stress tensor arising from that work's choice of constitutive models) in order to reflect the deformation of the solid by motion within the fluid. The authors evade numerical instability in the fluid by modifying the deformation tensor, which would otherwise exhibit a rough deformation within the fluid domain. The advection terms are discretized using fifth-order differences, while all other spatial terms are discretized with second-order differences. The time-stepping scheme is a combination of second-order Adams-Bashforth for the advection terms and Crank-Nicolson for the implicit terms. However, both these methods are much more expensive than the IB method, since they use a single paradigm for both the fluid and the structure; in addition, they are first-order accurate just like the IB method. An interesting hybrid method is the so-called

Finite Element IB method which is a method that uses Finite Element elasticity for platelets but retains the fluid solver of the traditional IB method. The method has two formulations: the partitioned formulation maintains two separate Lagrangian forces, an internal force density (throughout the volume) and a transmission force density (restricted to the surface), while the other maintains a unified formulation of both forces. For more, see recent work by Boyce Griffith [43]. This method uses a coarser Lagrangian mesh than the traditional IB method, despite higher accuracy. It also exhibits improved volume conservation properties over both the traditional IB method and the IFE method. The method also allows for the easy evaluation and inclusion of constitutive models in the well-studied finite element formulation. The unified formulation conserves energy. The Finite Element formulation works well with both structured and unstructured surfaces (and/or volumes). In a sense, this method is similar to the method we will introduce in Chapters 2 and 3, but is too costly to use for platelets, given that platelets are usually modeled as simply connected objects that are homeomorphic to the sphere (either $\mathbb{S}^1$ or $\mathbb{S}^2$).

1.2 Simulating Platelet Mechanics

In the unactivated state in which platelets circulate, they are fairly rigid ellipsoidal cells with a major axis of 2 to 3 microns and an aspect ratio of close to four. Unactivated platelets do not adhere to one another or to the intact endothelial cell lining of the blood vessels. While very numerous in the blood ($\approx$ 250,000/μliter), they are much smaller and far less numerous than the red blood cells, which comprise 40-45% of the blood's volume. It is the red blood cells rather than individual platelets that determine the blood's rheological properties [84].

Among the many changes that an activating platelet undergoes, two are most relevant for modeling and simulating the mechanics of aggregation. One is that $\alpha_{IIb}\beta_3$ integrin receptors embedded in its surface membrane become activated and capable of binding dimeric fibrinogen molecules and multimeric vWF molecules from the blood plasma. By binding to receptors on two platelets, these molecules serve as links between the platelets that allow them to cohere. The second is that the platelet's cytoskeleton is reorganized, and as a result, the platelet initially becomes more spherical. However, it also becomes sufficiently flexible that over time, it can spread out over the surface to which it is adhered. Activation can also be triggered when a platelet is exposed to sufficiently high concentrations of specific chemicals that are secreted into the plasma by other activated platelets. Thus, platelets near the injury site (but which do not directly contact the damaged vascular

wall) can be activated, and these chemically-activated platelets can also cohere with one another and with the wall-adherent platelets. Through these mechanisms, a platelet clump or aggregate grows at the injury site. As it grows, the aggregate can profoundly change the local fluid dynamics (to the extent that vessel occlusion can occur). Conversely, the changing fluid dynamics influence the further growth of the platelet aggregate both in terms of changes in the transport of platelets and chemicals to and from the injury site, and in terms of fluid forces which affect the binding and unbinding kinetics of the bonds between platelets and the subendothelium and between pairs of platelets.

Disruption of the endothelial cell lining exposes collagen and adsorbed von Willebrand factor (vWF) molecules in the subendothelial matrix to the blood. Platelets adhere to both molecules via specific receptor molecules on the platelets' surfaces. In addition to slowing or stopping platelet motion over the subendothelium, this binding triggers intracellular signaling pathways that lead to platelet activation [49, 71].

The platelet aggregation models developed by Fogelson *et al.* [17, 21, 24, 23, 95] track the motion and behavior of a collection of individual platelets as they interact with the suspending fluid, one another, and the vessel walls. They also track fluid concentrations of platelet activating chemicals, cell-cell and cell-surface forces, fluid motion, and the local fluid forces on the growing thrombus. In the models, nonactivated platelets are activated by proximity to reactive sites on the injured wall, or through exposure to a sufficiently high concentration of activator in the fluid. Activation enables a platelet to cohere with other activated platelets, and to secrete additional activator. The platelets and the secreted chemical are advected with the fluid and diffuse relative to it. Each platelet is represented as an IB object, *i*.e., as a collection of elastically-linked Lagrangian points that each move at the local fluid velocity. New elastic links are created dynamically to model the adhesion of a platelet to the injured wall or the cohesion of activated platelets to one another. Multiple links can form between a pair of activated model platelets or between a model platelet and the injured wall, and these links collectively represent the ensemble of molecular bridges binding real platelets to one another or to the damaged vessel. The links exert forces on the surrounding fluid to resist motions which would otherwise separate the linked entities. Links may break if subjected to sufficiently high stress. Model variables are fully coupled: the fluid carries the activator and platelets, while the interplatelet forces, potentiated by chemically-induced activation of the platelets, determine the local flow.

The Immersed Boundary (IB) method was developed to study the interactions of a viscous incompressible fluid with one or more moving and/or deformable elastic objects in

contact with that fluid. To review the IB method, we focus on a simple two-dimensional model problem in which a single fluid-filled closed elastic membrane is immersed in a viscous fluid.

The physics of the model problem is that an elastic membrane is under tension and exerts forces on the adjacent fluid. These forces may cause the fluid to move and, correspondingly, cause points on the membrane to move along with the fluid. In the IB method, the fluid is described in the Eulerian frame through a velocity field $\boldsymbol{u}(\boldsymbol{x},t)$ and pressure field $p(\boldsymbol{x},t)$ defined at every point $\boldsymbol{x}$ in the physical domain Ω. The elastic membrane is described in the Lagrangian frame. Let the elastic membrane be parameterized by q, and denote by $\boldsymbol{X}(q,t)$ the spatial coordinates at time t of the membrane point labeled by q. The IB equations are the following coupled equations of motion for the fluid variables $\boldsymbol{u}(\boldsymbol{x},t)$ and $p(\boldsymbol{x},t)$ and the membrane configuration $\boldsymbol{X}(q,t)$.

$$\rho(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}) = -\nabla p + \mu\Delta\boldsymbol{u} + \boldsymbol{f}, \qquad \nabla\cdot\boldsymbol{u} = 0, \tag{1.1}$$

$$\boldsymbol{F}(q,t) \;=\; \boldsymbol{F}\left(\boldsymbol{X}(q,t), \frac{\partial}{\partial q}\boldsymbol{X}(q,t)\right), \tag{1.2}$$

$$\boldsymbol{f}(\boldsymbol{x},t) \;=\; \int \boldsymbol{F}(q,t)\;\delta(\boldsymbol{x}-\boldsymbol{X}(q,t))\;dq, \tag{1.3}$$

$$\frac{\partial\boldsymbol{X}}{\partial t}(q,t) = \int_\Omega \boldsymbol{u}(\boldsymbol{x},t)\;\delta(\boldsymbol{x}-\boldsymbol{X}(q,t))d\boldsymbol{x}. \tag{1.4}$$

Equations (1.1) are the Navier Stokes equations which describe the dynamics of a viscous incompressible fluid, of constant density ρ and constant viscosity μ, driven by a force density $\boldsymbol{f}$ which here arises because of the elastic deformation of the immersed membrane. Equation (1.2) specifies the elastic force (per unit q) at each point of the immersed boundary object. The functional dependence of this force on the state of the boundary is specified appropriately to the material being modeled. Equation (1.3) defines the fluid force density $\boldsymbol{f}(\boldsymbol{x},t)$ in terms of the immersed boundary elastic force density $\boldsymbol{F}$. Equation (1.4) specifies that the velocity of each immersed boundary point equals the fluid velocity at the same location, a formulation of the no-slip boundary condition for viscous flows. In the model problem and the platelet applications, we assume that the IB objects are neutrally buoyant; the IB membrane itself carries no mass and each object's mass is attributed to the fluid in which it sits. For more on the IB method, see [67].

Chapters 2 and 3 of this dissertation will focus on extending the IB method for more efficient platelet simulations. Chapter 2 will focus on exploring different modeling strategies for static platelet-like shapes, and compare them to the traditional modeling strategies used for Lagrangian structures in the IB method. Chapter 3 will then augment the IB

method with the model developed in Chapter 2 and explore the ramifications within a full fluid-structure interaction setting.

1.3 Simulating Platelet Chemistry

Consider a stationary fluid in which there are a number of suspended objects on whose surfaces chemical reactions may occur. Some of the chemical may unbind from the surface of a particular object, thus entering the fluid phase, and undergo diffusion in the fluid. This chemical may bind to the surface of the same or a different one of the suspended objects, and when bound may diffuse on the surface of the suspended object. At each point on the objects, the flux of chemical to (from) that surface should exactly balance the rate of consumption (production) of the chemical on that surface. That is, there should be no flux of the chemical *across* the surfaces of the moving objects. We wish to determine the surface density (amount/area) of the chemical bound at each point of the suspended objects' surfaces and determine the concentration of chemical at points of the fluid phase as well.

The specific situation we have in mind is intravascular blood clotting. The fluid is blood plasma, and the objects are small blood cells called platelets that normally circulate in the blood. During the clotting process, platelets can become activated, allowing them to stick to one another and to the vascular wall. We have modeled this process (following [22]) using the Immersed Boundary method [67] to determine the coupled motion of the fluid and platelets. In these calculations, the no-slip condition holds on the platelet surfaces, *i.e.,* the velocity at each point of the platelet matches that of the immediately adjacent fluid. The chemicals of interest in the current work are those involved in conveying 'activation' signals between platelets [22] and those involved in the coagulation enzyme network [54]. In summary, we seek to solve a reaction-diffusion equation for each chemical on a platelet surface, where the reactions are coupled to diffusion equations in the blood around the platelet. Appropriate boundary conditions are to be satisfied at all points of the surfaces of the platelets (and external boundaries).

There are a variety of Cartesian grid methods that can be used to solve partial differential equations (PDEs) in the presence of irregularly-shaped objects within the domain. The widely-used Immersed Boundary (IB) method introduced by Peskin [65, 66] uses a discrete delta function to spread boundary forces from the IB surface to the fluid, and then the discrete fluid dynamic equations are solved using a regular discretization on a rectangular grid everywhere in the domain. The forcing methods introduced by Goldstein [39], Mohd-Yusof [61], and Kim *et al.* [53] follow the idea of the IB method in using forces to represent

objects embedded in a flow, but calculate the forces using feedback terms or numerical corrections to approximately enforce boundary conditions. Fadlun [14] introduces direct forcing without modification of the stencil; but, in the end, he applies the forcing in an implicit way by modifying the stencil at grid points near the irregular boundaries. The Immersed Interface method of LeVeque and Li [57], the Embedded Boundary (EB) method of Johansen and Collela [51], the sharp interface method of Udaykumar and coworkers [85, 94], and the capacity function finite volume method of Calhoun and LeVeque [7] all modify the stencil at grid points near the irregular surfaces. Because of the explicit inclusion of the boundary conditions in the linear system, the methods with changed stencil often have better accuracy and stability than the direct forcing methods, while the simpler grid and uniform stencils of the latter make them easier to implement and allow use of fast solvers.

The Augmented Forcing Method (AFM) was developed to solve the problem of simulating chemical diffusion for stationary fluid and platelets, with the end goal of eventually simulating chemical transport for full platelet simulations within the IB method [93]. The key idea of the AFM (as presented in [93]) is to solve a discrete PDE at all N_T grid points, except that at some specific N_F forcing points on the grid, the discrete PDE is modified by the addition of a forcing term that enforces boundary conditions. The hope is to capture the accuracy of implicit forcing while retaining some measure of the computational efficiency of direct forcing.

We discretize the PDE for fluid-phase chemical concentrations using a second-order five point stencil for the Laplacian in space and the second-order Crank-Nicolson scheme in time. Let A be the matrix formed from the discretization of a chemical diffusion equation (see Equation (4.1)) and $\mathbf{r}$ be the right-hand side vector from that discretization. Let P be an $N_T \times N_F$ matrix that maps each forcing point index to the index of the corresponding grid point in the overall ordering of the grid unknowns used in the vector of chemical concentrations $\mathbf{c}$, *i.e.*, all the entries of P are zero except for those locations corresponding to forcing point locations, which are one. Let $\mathbf{F}$ be a vector whose N_F entries contain the forcing values. Let E be an $N_F \times N_T$ matrix that enforces boundary conditions as described below. Then, we require the solution of the following block system of equations:

$$\begin{pmatrix} A & P \\ E & 0 \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \mathbf{F} \end{pmatrix}^{n+1} = \begin{pmatrix} \mathbf{r} \\ \mathbf{r}_{\text{bc}} \end{pmatrix}. \tag{1.5}$$

This system is solved in two stages.

- First, we find $\mathbf{F}$ by solving the Schur Complement system of the above block system using the BiCGSTAB iterative method. This system is as follows:

$$-EA^{-1}P\mathbf{F} = \mathbf{r}_{\text{bc}} - EA^{-1}\mathbf{r}. \tag{1.6}$$

- Having solved for $\mathbf{F}$, we then solve for the chemical concentrations by solving $A\mathbf{c} = \mathbf{r} - P\mathbf{F}$. We use a conjugate gradient solver preconditioned by the modified incomplete Cholesky factorization of A.

In the AFM as implemented in [93], for each forcing point, the boundary condition at the corresponding boundary point (see Section 4.3) and the concentrations at five nearby fluid points are used to construct a bivariate quadratic interpolant tht satisfies the boundary condition at the boundary point. A value of c at the forcing point is obtained by evaluating this polynomial at the forcing point. Since the fluid concentrations are still to be determined, this gives an implicit relationship between the forcing point concentration and those at the five fluid points. This relationship is used to populate one row of the matrix E. With this approach, if two platelets are close to one another or the shape of the platelet is concave, there may not be a sufficient number of points necessary to perform this interpolation. In such a case, grid refinement is necessary to introduce sufficient spacing between objects. Chapter 4 in this dissertation will focus partly on overcoming this limitation of the AFM using symmetric Hermite interpolation (to be discussed shortly). The primary focus of Chapter 4, however, will be the development of a new method based on RBF-generated finite differences for the simulation of reaction-diffusion equations in the context of our Hermite interpolation variant of the AFM.

1.4 Radial Basis Function Interpolation

1.4.1 Overview

In this section, we will present background information and the formulation for two important variations of RBF interpolation that will be used in this thesis. First, we will discuss RBF interpolation of function samples on scattered node sets; this could be called RBF Lagrange interpolation, but we do not use that title and so avoid any inadvertent confusion about Lagrange interpolating polynomials. Next, we will discuss RBF Hermite interpolation, an important generalization of polynomial Hermite interpolation. RBF interpolation will be used extensively in this thesis, and RBF Hermite interpolation will be used in Chapter 4.

1.4.2 RBF Interpolation

We present an overview of RBF interpolation, which is essential to understanding the RBF-based geometric modeling approach outlined in Chapters 2 and 3, the RBF-FD approach used in Chapters 3 and 4, and the symmetric Hermite interpolation approach used in Chapter 4. This section also presents notation that is more convenient for expressing the RBF interpolation problem on point clouds, which will be modified (as appropriate) in later Chapters depending on the context in which RBF interpolation is used. Let $\Omega \subseteq \mathbb{R}^d$, and $\phi : \Omega \times \Omega \to \mathbb{R}$ be a kernel with the property $\phi(\boldsymbol{X}, \boldsymbol{Y}) := \phi(\|\boldsymbol{X} - \boldsymbol{Y}\|)$ for $\boldsymbol{X}, \boldsymbol{Y} \in \Omega$, where $\|\cdot\|$ is the standard Euclidean norm in $\mathbb{R}^d$. We refer to kernels with this property as *radial kernels* or *radial functions*. Given a set of nodes $X = \{\boldsymbol{X}_k\}_{k=1}^N \subset \Omega$ and a continuous target function $f : \Omega \to \mathbb{R}$ sampled at the nodes in X, we consider constructing an RBF interpolant to the data of the following form:

$$I_\phi f(\boldsymbol{X}) = \sum_{k=1}^{N} c_k \phi(\|\boldsymbol{X} - \boldsymbol{X}_k\|) + c_{N+1}. \tag{1.7}$$

The interpolation coefficients $\{c_k\}_{k=1}^{N+1}$ are determined by enforcing $I_\phi f|_X = f|_X$ and $\sum_{k=1}^N c_k = 0$. This can be expressed as the following linear system:

$$\underbrace{\begin{bmatrix} \phi(r_{1,1}) & \phi(r_{1,2}) & \dots & \phi(r_{1,N}) & 1 \\ \phi(r_{2,1}) & \phi(r_{2,2}) & \dots & \phi(r_{2,N}) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi(r_{N,1}) & \phi(r_{N,2}) & \dots & \phi(r_{N,N}) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{bmatrix}}_{A_X} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \\ c_{N+1} \end{bmatrix}}_{c_f} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \\ 0 \end{bmatrix}}_{f_X}, \tag{1.8}$$

where $r_{i,j} = ||\boldsymbol{X}_i - \boldsymbol{X}_j||$. If ϕ is a positive-definite radial kernel or an order one conditionally positive-definite kernel on $\mathbb{R}^d$, and all nodes in X are distinct, then the matrix A_X above is guaranteed to be invertible (see, for example, [88, Ch. 6–8]).

For completeness, it is useful to extend the above discussion to the interpolation of vector-valued functions $\mathbf{g}(\boldsymbol{X}) : \Omega \to \mathbb{R}^d$ sampled at a set of nodes $X = \{\boldsymbol{X}_k\}_{k=1}^N \subset \Omega$. For this problem, we simply apply scalar RBF interpolation as given in Equation (1.7) to each component of $\mathbf{g}(\boldsymbol{X})$ and represent the resulting interpolant as $I_\Phi \mathbf{g}$. For example, if $d = 3$ and $\mathbf{g} = \begin{bmatrix} g^x & g^y & g^z \end{bmatrix}^T$, then the vector interpolant is given as

$$I_\Phi \mathbf{g}(\boldsymbol{X}) = \begin{bmatrix} I_\phi g^x(\boldsymbol{X}) & I_\phi g^y(\boldsymbol{X}) & I_\phi g^z(\boldsymbol{X}) \end{bmatrix}. \tag{1.9}$$

The interpolation coefficients for each component of $I_\Phi \mathbf{g}$ can be determined by solving a system of equations similar to Equation (1.8), but with the right-hand side replaced with

the respective component of $\mathbf{g}$ sampled on X. This allows some computational savings for determining the interpolation coefficients for $I_\phi f$ and $I_\Phi \mathbf{g}$ with a direct solver since the matrix A_X then only needs to be factored once. We exploit this fact in Chapter 5.

There are many choices of positive definite or order one conditionally positive definite radial kernels that can be used in applications; see [15, Ch. 4, 8, 11] for several examples. These kernels can be classified into two types: finitely smooth and infinitely smooth. It is still very much an open question about which kernel is optimal for which application. Typically infinitely smooth kernels such as the Gaussian ($\phi(r) = \exp(-(\varepsilon r)^2)$), multiquadric ($\phi(r) = \sqrt{1 + (\varepsilon r)^2}$), and inverse multiquadric ($\phi(r) = 1/\sqrt{1 + (\varepsilon r)^2}$) are used in the RBF methods for numerically solving PDEs [77, 91, 3, 12, 18, 36].

All infinitely smooth kernels feature a free "shape parameter" ε, which can be used to change the kernels from peaked (large ε) to flat (small ε). For smooth target functions, smaller values of ε generally lead to more accurate RBF interpolants [28, 55]. The standard way of computing the interpolant by solving Equation (1.8) becomes increasingly ill-conditioned as the size of the system grows, or as $\varepsilon \to 0$ [29]. While stable algorithms have been developed for bypassing this ill-conditioning (see [27]), these algorithms require knowledge of surface topology. In Chapter 5, we will detail strategies for selecting the shape parameter based on condition numbers of RBF interpolation matrices. We will also introduce (in the same chapter) a strategy for modifying the shape parameter to produce interpolants that compensate for irregularities in point spacing.

1.4.3 Symmetric Hermite Interpolation

Thus far, we have seen how RBFs can be used for the interpolation of function data on scattered node sets. This can be viewed as the RBF-analog of Lagrange interpolation, with the term "Lagrange interpolation" not referring to the use of a Lagrange basis for a polynomial interpolant, but rather the fact that polynomial interpolation is being used to reconstruct a function using samples of that function. However, it is possible with polynomial methods to incorporate extra information in the form of samples of successive derivatives of functions into the interpolant. This form of interpolation (of function values and successive derivative values) is called Hermite interpolation and is well-known in the literature. While this is commonly used in 1D settings, polynomial Hermite interpolants are often difficult or impossible to generalize to higher dimensions or to more general linear functionals than derivatives, especially when the data locations are nonuniform. For these problems, the RBF Hermite interpolation method [15, Chapter 36] offers a powerful solution that can be generalized to scattered nodes and various differential and integral constraints

(*e.g.*, see [4, 91]). This method is also called symmetric Hermite interpolation, or generalized Hermite interpolation. We now present an overview of Hermite interpolation in the RBF context. The following discussion is adapted primarily from [15, Chapter 36].

Let $\Omega \subseteq \mathbb{R}^d$, and $\phi : \Omega \times \Omega \to \mathbb{R}$ be a radial kernel. Given a set of nodes $X = \{\boldsymbol{X}_k\}_{k=1}^{N+M} \in \Omega$, a continuous target function $f : \Omega \to \mathbb{R}$ sampled at a subset of the nodes in X so that we have $\{f_i\}_{i=1}^{N}$, and samples of linear functionals applied to that function $\{\mathcal{L}_j f_j\}_{j=N+1}^{M}$, we consider constructing an RBF interpolant of the following form:

$$I_\phi f(\boldsymbol{X}) = \sum_{i=1}^{N} c_i \phi(\|\boldsymbol{X} - \boldsymbol{X}_i\|) + \sum_{j=N+1}^{M} c'_j \mathcal{L}_j^{\boldsymbol{X}_j} \phi(\|\boldsymbol{X} - \boldsymbol{X}_j\|), \tag{1.10}$$

where the superscript in $\mathcal{L}_j^{\boldsymbol{X}_j}$ refers to the argument with respect to which $\mathcal{L}_j$ is applied. We obtain a 2×2 block system of linear equations by imposing the interpolation conditions

$$I_\phi f(\boldsymbol{X}_i) = f_i, i = 1, 2, \ldots, N, \tag{1.11}$$

$$\mathcal{L}_j I_\phi f(\boldsymbol{X}_j) = \mathcal{L}_j f_j, j = N+1, \ldots, M. \tag{1.12}$$

Note that despite our selection of continuous indices for notational convenience, this method requires no ordering on function samples or samples of linear functionals applied to the function. Indeed, both the function data and derivative data can be scattered over the nodes in X, possibly overlapping on some nodes and not overlapping on others. It is also worth noting that the second interpolation condition requires applying functionals to functionals. This block system is symmetric and nonsingular for both positive-definite and conditionally positive-definite kernels provided the functionals are linearly independent [80, 62, 48]. This now provides us a powerful tool to interpolate scattered samples of linear functionals applied to functions in arbitrary dimensions. For completion, we note that much like in Section 1.4.2, this discussion can be easily extended to vector-valued data as well. In the context of this work, however, we will be applying symmetric Hermite interpolation to scalar-valued data in 2D domains, with linear functionals arising from Robin boundary conditions on PDEs.

1.5 RBFs for the solution of PDEs

In this section, we present an overview of numerical methods for the solution of PDEs based on RBF interpolation. Every scheme discussed in this section uses some form of RBF interpolation and differentiation for the spatial discretization of a PDE, leveraging the advantages of RBFs to enable the solution of PDEs on point clouds and in a meshfree fashion. We will discuss the RBF-based Pseudospectral (RBF-PS) method, the RBF-generated finite

difference (RBF-FD) method, and some other popular methods that we do not use in this work. In our discussion, we will focus on methods applicable to platelet aggregation. Since current discretizations of the IB method and the AFM use Eulerian grids for fluid dynamics and are reasonably numerically efficient, we will emphasize the use of RBF methods for the solution of PDEs on closed surfaces (like platelets and red blood cells) without boundary conditions.

1.5.1 RBF-based Pseudospectral Method (RBF-PS)

The idea behind pseudospectral (PS) methods is to represent the solution of a PDE using a smooth and globally-supported set of basis functions [25]. Unlike numerical methods based on the weak form of a PDE, PS methods are collocating methods, *i.e.*, they attempt to satisfy the PDE at a discrete set of points in the domain (known as collocation points or nodes). If $\hat{u}$ is the approximate solution to a PDE, we write

$$\hat{u}_i = \sum_{k=1}^{N} c_k B_k(x_i), \tag{1.13}$$

$$\implies \boldsymbol{u} = A\mathbf{c}. \tag{1.14}$$

where A is the matrix of basis functions evaluated at the collocation points x_i, $\boldsymbol{u}$ is the vector of samples of the solution $\hat{u}$ at the collocation points, and $\mathbf{c}$ is the vector of coefficients. Given a linear PDE of the form $\mathcal{L}u = f$, where $\mathcal{L}$ is a linear differential operator, we may write

$$\boldsymbol{u}' = A_{\mathcal{L}}\mathbf{c}, \tag{1.15}$$

where $\boldsymbol{u}'$ is the discrete form of the linear operator $\mathcal{L}$ applied to the vector $\boldsymbol{u}$ and $A_{\mathcal{L}}$ is the matrix formed by applying the operator $\mathcal{L}$ to the basis functions and evaluating the resulting function at the collocation points. Simplifying, we have $\boldsymbol{u}' = L\boldsymbol{u}$, where $L = A_{\mathcal{L}}A^{-1}$. The matrix L is traditionally called a differentiation matrix because it usually involves derivatives of the basis functions. For a fixed set of collocation points, L can be precomputed and reused through the simulation. A popular choice (in 1D) of basis functions is Chebyshev polynomials, an orthogonal family of polynomials, with the collocation points being selected as Chebyshev points. With appropriate boundary conditions, the Chebyshev differentiation matrix L_c is usually invertible and therefore usable in an implicit time-stepping scheme for the PDE, though this is not always the case [83]. When one opts to use smooth RBFs with global support (like the (Inverse) Multiquadric or the Gaussian) as the basis by setting $B_k(x) = \phi(||x - x_k||)$, we obtain an RBF-PS method; this approach generalizes

to higher dimensions because of the previously discussed features of RBF interpolation. Boundary conditions can be incorporated into the differentiation matrix using the Hermite interpolation approach outlined previously. If the solution to the PDE is smooth, this method can give spectral accuracy and convergence. Further, if the RBF is positive definite and $\mathcal{L}$ elliptic, the differentiation matrix L is always invertible, even on ill-posed problems! For a more detailed discussion on RBF-PS methods on planar domains, see [15].

An application of great interest is the solution of PDEs on closed surfaces (and manifolds), where the only boundary conditions are the restriction of the solution to the surface itself. Several of the applications of RBF-PS to the solution of PDEs on surfaces focused on approximating differential operators on the sphere [19, 20, 26]; these methods were so-called *intrinsic* methods, in that they used points only on the sphere $\mathbb{S}^2$ to solve PDEs on that surface. Another class of methods is the set of *embedded, narrow-band* methods, that extend the PDE to the embedding space and solve it in a narrow-band around the surface, thus enabling the method to handle more arbitrary surfaces. One such method is the popular Closest Point method, a polynomial-based method [60]. A similar method is the recently-developed (RBF-based) Orthogonal Gradients method [68]. However, narrow-band methods are not as computationally efficient as intrinsic methods. This problem was addressed by Fuselier and Wright in their recent work [36]. This is the first RBF-PS method that works on general surfaces. The method combines the efficiency of intrinsic methods with the stability of the narrow-band approaches. It was tested on diffusion and reaction-diffusion equations. Chapter 5 of this thesis is focused on modifying Fuselier and Wright's method to use a finite-difference approach rather than a pseudospectral approach.

1.5.2 RBF-generated Finite Differences (RBF-FD)

Finite-difference (FD) methods have been used for decades in the spatial and temporal discretization of PDEs. As in the previous subsection, we focus on the spatial discretization. While PS methods offer an excellent accuracy-cost profile, they depend on the solution being globally smooth. In addition, PS approximations of linear operators result in dense matrices, which can be very expensive to invert especially if the collocation nodes themselves change as a function of time (which will happen on an evolving surface or a problem with mesh/node refinement). FD methods offer a compromise between cost and accuracy by approximating derivatives. To compute an $n-$point finite difference formula, one uses the function value at a point and its $n-1$ nearest neighbors, where $n << N$, the total number of nodes in the discretization of the domain . They thus have a cost scaling as $O(N)$ when $n << N$. This is in contrast to the $O(N^3)$ cost incurred by PS methods. FD methods, on

the other hand, display algebraic convergence rather than spectral convergence. However, they can be more accurate than PS methods on problems that have rough solutions.

There are many ways to generate FD approximations to derivatives. The simplest way is by returning to the limit definition of a derivative, dropping the limit, and using a very small space (or time) step. A more principled way of generating FD approximations to the derivatives of a function is by expanding the Taylor series of the function in the neighborhood of the point at which we wish the derivative, rearranging terms in the Taylor series to express the derivatives that appear in the Taylor expansion in terms of linear combinations of function values, and then truncating the Taylor expansion at a desired number of terms. The leading order of the Taylor expansion determines the order of the FD approximation to the derivative. However, this procedure can grow tedious as one proceeds to higher-order approximations or higher derivatives. Further, this procedure can be even more cumbersome when the points at which we require derivatives are not evenly-spaced.

In 1D, the simple solution is to realize that standard FD formulae for some derivative can also be generated by interpolating those derivatives with a monomial basis and computing the interpolation weights (which turn out to be identical to the finite difference weights from Taylor polynomials). This procedure was described in detail by Fornberg [30], and has a cost of $O(n^3)$ for an $n-$point FD formula. If all points in the domain are equispaced, one can calculate the weights once and reuse them throughout the domain, resulting in a sparse $N \times N$ differentiation matrix. This approach generalizes to equispaced nodes in higher dimensions as well. However, given a set of scattered nodes in two or three dimensions, the multivariate polynomial interpolation matrix may be singular. Instead, one can use the fact that RBFs are extensions of polynomial interpolants to scattered node sets (for a shape parameter of $\varepsilon > 0$) and replace the monomial basis with an RBF basis, allowing us to generate finite-difference formulae on platelet surfaces.

For clarity, consider generating a 3-point RBF-FD approximation for the first derivative at some point $x = x_c$ using points x_1, x_2 and x_3 in 1D. Defining the vector $\mathbf{b} = [\frac{d}{dx}\phi(||x - x_1||)\big|_{x=x_c}, \frac{d}{dx}\phi(||x - x_2||)\big|_{x=x_c}, \frac{d}{dx}\phi(||x - x_3||)\big|_{x=x_c}]^T$, the unknown vector of RBF-FD weights $\mathbf{w}_1$ and the RBF interpolation matrix $A_{ij} = \phi(||x_i - x_j||)$, we have

$$A\mathbf{w}_1 = \mathbf{b}. \tag{1.16}$$

The only difference from the monomial case is the way the matrix A is generated. Note that we could have taken another approach, one that is similar to the PS approach but using only $n = 3$ points out of all N points in the domain. This approach involves forming and inverting the RBF interpolation matrix A, then premultiplying it by the matrix of

derivatives of basis functions evaluated at $x = x_c$. In the notation we just developed, this matrix is simply $\mathbf{b}^T$. We thus have a second possible set of weights $\mathbf{w}_2 = \mathbf{b}^T A^{-1}$. It is easy to see that $\mathbf{w}_1 = \mathbf{w}_2^T$ if $A = A^T$. In other words, if the matrix is symmetric (as it is when using RBF interpolation), both approaches yield the same set of weights. In Chapter 4, we will use the approach described by Equation (1.16) to generate the RBF-FD weights for the gradient of a function on a 1D surface, and then combine those weights appropriately to generate an approximation to the surface Laplacian. However, in Chapter 5, we will use the second approach of interpolating and differentiating successively to compute Laplacians on a 2D surface. The rationale for this will be explained in Chapter 5. Also, as we will mention in Chapter 5, even when solving PDEs on arbitrary manifolds, it is sufficient to perform RBF interpolation in the embedding space, despite the overall method being an embedded method.

We note that while the RBF-FD method has proven successful for a number of other applications in planar domains in two and higher dimensions (*e.g.*, [82, 77, 8, 91, 9, 78]), and on the surface of a sphere [26, 18], this work represents the first application of RBF-FD to general 1D surfaces (Chapter 4) as well as 2D surfaces (Chapter 5).

1.5.3 RBF Partition of Unity Method (RBF-PUM)

We will now discuss the use of RBFs within the so-called Partition of Unity method (PUM) for interpolation and the solution of PDEs. Let $\Omega \subseteq \mathbb{R}^d$ be the domain of interest. The idea of the Partition of Unity method is to partition this domain into some set of M overlapping subdomains. Each domain has a compactly-supported, nonnegative continuous function w_j associated with it such that for every point $\boldsymbol{x}$ in Ω, we have

$$\sum_{k=1}^{M} w_j(\boldsymbol{x}) = 1. \tag{1.17}$$

The next step is approximate the function of interest and its derivatives on each subdomain by constructing a local RBF interpolant $u(\boldsymbol{x})$. The global approximation to the function on the domain is then $\sum_{k=1}^{M} u_k(\boldsymbol{x}) w_k(\boldsymbol{x})$. Note the similarity to the RBF-FD approach in the local interpolation strategy; however, the RBF-PUM has the added advantage of "stitching" together the local interpolants into a global approximant using the partition of unity functions w_k. If the M partitions are chosen well, this method can give very high orders of convergence, with the computational cost growing as $O(N)$, N being the total number of collocation points in the domain. The constant associate with this method is higher than that in the RBF-FD method, but again, this method is very promising. Recent

work by Safdari, Heryudono, and Larsson (see [70]) has demonstrated the viability of using the RBF-PUM method for the solution of convection-diffusion equations on planar domains. Work is currently in progress on implementing the RBF-PUM for the solution of transport equations on the sphere. For a detailed review on the RBF-PUM, see [15]. In this work, we will not focus on the RBF-PUM.

1.6 Overview of Thesis Topics

To substantiate the thesis that RBF interpolation is well-suited to the development of numerical methods for platelet aggregation, we propose the following contributions, outlined in each chapter of this dissertation:

- In Chapter 2, we present a new RBF-based parametric approach to modeling platelets, to facilitate simulations of mechanical interactions between platelets as well as simulations of chemical transport during platelet aggregation. This work was published in Applied Numerical Mathematics [72].
- In Chapter 3, we augment the IB method with our RBF-based parametric model, and explore the effects of this choice on the IB method. We compare the new RBF-IB method to the traditional IB method in terms of accuracy and computational cost for two-dimensional problems, and present the results of a simple platelet aggregation simulation. This work has been submitted to the Journal of Computational Physics [74].
- In Chapter 4, we present the first RBF-FD method for the simulation of reaction-diffusion equations on arbitrary surfaces embedded in 2D domains. We also modify the Augmented Forcing Method (AFM) with symmetric Hermite interpolation to remove a significant limitation of that method. We present a numerical method that uses our RBF-FD method within the modified AFM for the solution of a coupled problem involving chemical reaction-diffusion on platelet surfaces and chemical diffusion in the domain in which the platelets are embedded. We show that this method exhibits second-order convergence in space and time. This work was accepted to the International Journal for Numerical Methods in Fluids [73]. This chapter also used the RBF geometric model developed in Chapter 2.
- In Chapter 5, we extend the RBF-FD method from Chapter 4 to surfaces of co-dimension 1 embedded in 3D domains. This is the first RBF-FD method for diffusion and reaction-diffusion equations on general surfaces. We also present a method based on modification of shape parameters for the stabilization of the differential operators

arising in the context of reaction-diffusion equations, removing the need for popular hyperviscosity-based stabilization approaches. We present convergence results on the sphere and torus, and present applications based on Turing pattern generation on surfaces and point clouds. This work has been submitted to the Journal of Scientific Computing [75].

- In Chapter 6, we summarize the contributions of this dissertation as outlined in Chapters 2-5. We then present four potential lines of research as a natural extension of the research outlined in this dissertation; all four research projects continue to develop the theme of RBF-based numerical methods for biological applications.

CHAPTER 2

GEOMETRIC MODELING OF PLATELETS

2.1 Introduction

In IB modeling, inactive platelets are approximately elliptical or ellipsoidal in 2D and 3D, respectively, while activated platelets are approximately circular in 2D and spherical in 3D. Piecewise linear approximations of platelets are currently used within IB methods applied to platelet aggregation (*e.g.*, [17, 21, 24]). We seek to explore alternative methods for the modeling of platelets that might decrease the computational time necessary to maintain and update platelet geometry and motion with comparable or better error characteristics to the standard piecewise linear models.

In this chapter, we examine two alternative representations for platelets: interpolation with Fourier-based techniques (trigonometric polynomials in 2D and spherical harmonics in 3D) and interpolation with radial basis functions (restricted to the unit circle in 2D and unit sphere in 3D). Fourier methods have frequently been used for the modeling of circular and spherical objects (*e.g.*, [76]). A recent result of Fornberg and Piret shows that both trigonometric polynomials and spherical harmonics are just special cases of radial basis functions (RBFs) when one chooses the shape parameter in a particular limit [27]. Additionally, error estimates for RBF interpolation on the circle and sphere have been given for a much wider range of target functions than just C^∞ [47, 50, 63].

To perform a platelet IB computation, one must (1) have a representation of the surface of the platelet and (2) be able to compute forces (internal structural forces) at a specified collection of material points on the platelet surface. Once forces are determined, they are "projected" to an Eulerian mesh in which they are incorporated into the solution of the Navier-Stokes equations for determining the motion of the fluid. Based upon the updated fluid velocity field, the platelet's position and shape are updated. We will not detail how the projection and interpolation are accomplished as this has been amply discussed in other works (*e.g.*, [64]). Our focus is instead restricted to models for representing the platelet objects and how these can be used for constructing and maintaining the object's representation, computing the normal vectors to the object, and computing the internal

structural forces.

For results, we will compare the piecewise linear, Fourier, and RBF methods for two different shapes in 2D and two different shapes in 3D that typify observed platelet geometries. We compare the errors in reconstructing these shapes, computing the normal vectors, and computing the forces. We provide a discussion of the engineering trade-offs we observe with respect to error and computational costs. Our results indicate that the RBF and Fourier models are viable alternatives to the piecewise linear models for platelet-like geometries in terms of errors versus computational cost. We furthermore find that the RBF models give better results for objects of varying smoothness than the Fourier models, and thus appear to be more promising in applications.

2.2 Geometric Modeling Strategies

In this section, we present the three different geometric modeling approaches to be examined. We first present the (traditional) piecewise linear approach for modeling two- and three-dimensional platelet structures. We then present our two alternative strategies based on a parametric representation of the surface: Fourier-based models (trigonometric series in 2D and spherical harmonic series in 3D) and radial basis function (RBF) models. Implementation details for all three methodologies are provided in Section 2.4.

2.2.1 Piecewise Linear Models

In the traditional (IB) method, parametric representations of the surface are rarely formed explicitly. Typically, a piecewise linear representation of the boundary is maintained. In 2D, the piecewise linear interpolant is a set of line segments between pairs of IB points. However, to perform secondary computations (such as computing normals) with a greater level of accuracy than what the piecewise linear interpolant would offer, piecewise quadratic interpolants are typically fitted to a set of IB points (*e.g.*, [92, §3.1.1]).

Given a parameter λ, the piecewise quadratic representation is therefore defined as:

$$x(\lambda) \approx a_x\lambda^2 + b_x\lambda + c_x, \tag{2.1}$$

$$y(\lambda) \approx a_y\lambda^2 + b_y\lambda + c_y. \tag{2.2}$$

The coefficients are computed by solving two linear systems of equations for each IB point; the right-hand sides to these systems of equations are simply the x and y coordinates of the three IB points to which the piecewise quadratics are fitted. Once the coefficients are obtained, one can now compute derivatives (and therefore normals and other quantities) at each IB point.

In 3D, the piecewise linear interpolant is a triangulation of the IB points (*e.g.*, [24]). An example of such a triangulated surface is given in Figure 2.1. Secondary computations, such as computing normals and forces, are computed from the triangulation as discussed in Section 2.4.1.

2.2.2 Parametric Models

The Fourier and RBF models we propose are both based on explicit parametric representations of the objects. Since our target objects are platelets, which in 2D models are nearly elliptical or circular and in 3D models are nearly ellipsoidal or spherical, we choose circular (or polar) and spherical parameterizations in 2D and 3D, respectively. Before discussing the two modeling approaches, we introduce some notation and put the modeling problem in the context of a reconstruction problem using interpolation.

In 2D, we use the following polar parametric notation to represent any of the objects:

$$\boldsymbol{x}(\lambda) = (x(\lambda), y(\lambda)), \tag{2.3}$$

where $-\pi \leq \lambda \leq \pi$ and $\boldsymbol{x}(-\pi) = \boldsymbol{x}(\pi)$. In the case the object is a circle of radius r, $\boldsymbol{x}(\lambda) = (r\cos\lambda, r\sin\lambda)$. In general, given a finite collection of values of the object, $\{\boldsymbol{x}(\lambda_k)\}_{k=1}^{N_\mathrm{d}} = \{(x(\lambda_k), y(\lambda_k))\}_{k=1}^{N_\mathrm{d}}$, our goal is to reconstruct $\boldsymbol{x}(\lambda)$ from smooth interpolations of each of

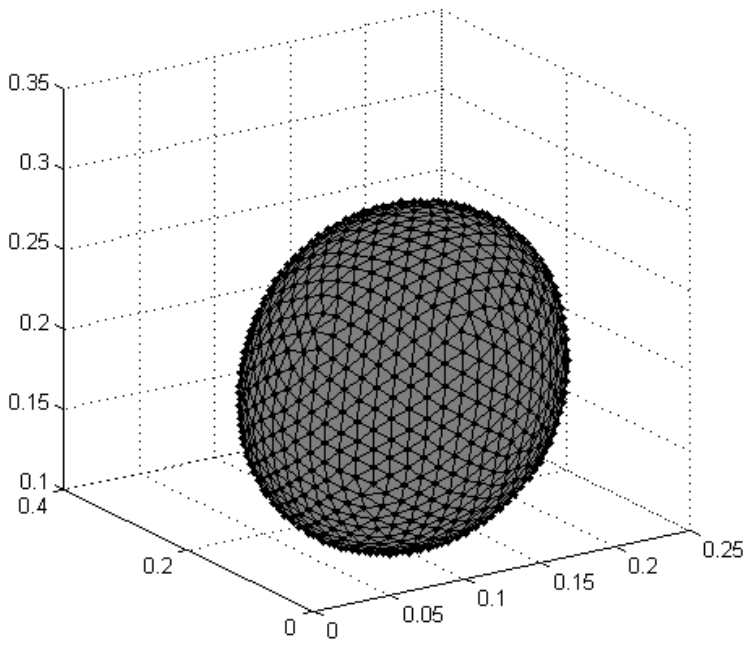

Figure 2.1: Illustration of the triangulation of a set of IB points in 3D.

its components. We refer to these values as the *data sites* and the set of values $\{\lambda_k\}_{k=1}^{N_\mathrm{d}}$ as the *nodes*. Figure 2.2 illustrates this reconstruction problem, of which the main ingredient is the interpolation of a function defined on the unit circle.

In 3D, we represent any of the objects using the following spherical parametric notation:

$$\boldsymbol{x}(\lambda,\theta) = (x(\lambda,\theta), y(\lambda,\theta), z(\lambda,\theta)), \tag{2.4}$$

where $-\pi \leq \lambda \leq \pi$ and $-\pi/2 \leq \theta \leq \pi/2$. Here the end conditions on $\boldsymbol{x}$ in λ are $\boldsymbol{x}(-\pi,\theta) = \boldsymbol{x}(\pi,\theta)$, while the end conditions in θ are $\boldsymbol{x}(\lambda,\pi/2) = \boldsymbol{x}(\lambda+\pi,\pi/2)$ and $\boldsymbol{x}(\lambda,-\pi/2) = \boldsymbol{x}(\lambda+\pi,-\pi/2)$ for $-\pi \leq \lambda \leq 0$ and $\boldsymbol{x}(\lambda,\pi/2) = \boldsymbol{x}(\lambda-\pi,\pi/2)$ and $\boldsymbol{x}(\lambda,-\pi/2) = \boldsymbol{x}(\lambda-\pi,-\pi/2)$ for $0 < \lambda \leq \pi$. These end conditions on θ are to enforce continuity of $\boldsymbol{x}$ at the poles of the spherical coordinate system. In the case the object is a sphere of radius r, $\boldsymbol{x}(\lambda,\theta) = (r\cos\lambda\cos\theta, r\sin\lambda\cos\theta, r\sin\theta)$. Similar to 2D, our goal is to reconstruct a general object $\boldsymbol{x}(\lambda,\theta)$ from smooth interpolations of each of its components which are given

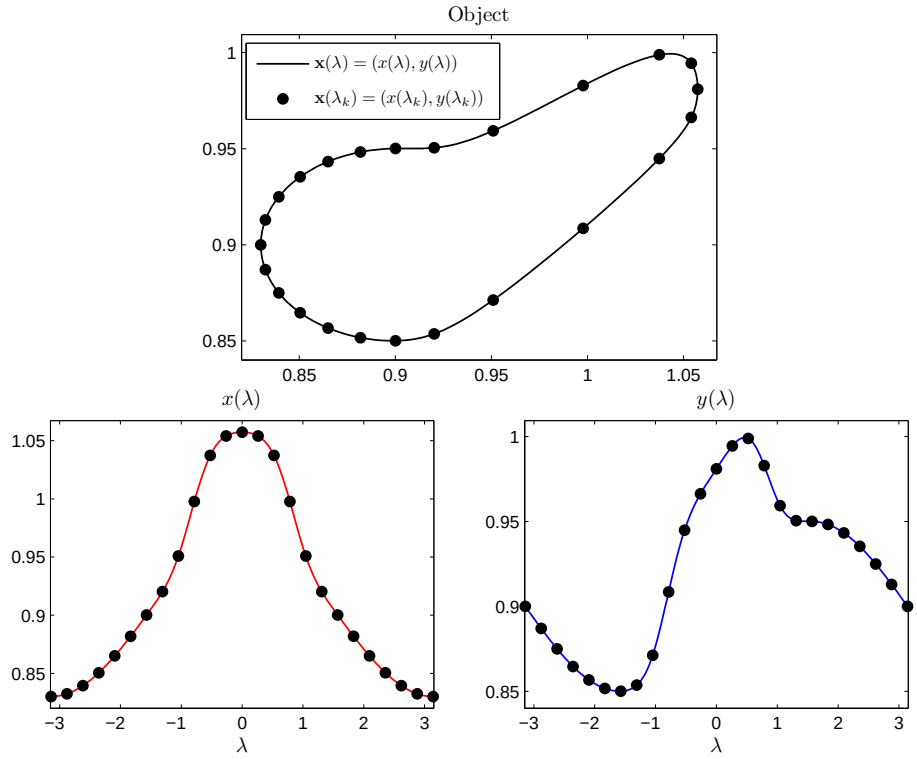

Figure 2.2: Illustration of the parametric representation of a 2D object $\boldsymbol{x}(\lambda)$ and the reconstruction from a finite number of data sites. The top figure shows the 2D object together with discrete data sites $\boldsymbol{x}(\lambda_k) = (x_k, y_k)$. The bottom left figure shows the x component of the object in parametric space and its values at the node set $\{\lambda_k\}_{k=1}^{N_\mathrm{d}}$, while the right figure shows the y component and its corresponding values. The goal is to reconstruct $x(\lambda)$ and $y(\lambda)$ from interpolations of these values at the node sets shown and then use these to reconstruct $\boldsymbol{x}(\lambda)$.

at some finite collection of locations $\{\boldsymbol{x}(\lambda_k, \theta_k)\}_{k=1}^{N_\mathrm{d}} = \{(x(\lambda_k, \theta_k), y(\lambda_k, \theta_k), z(\lambda_k, \theta_k))\}_{k=1}^{N_\mathrm{d}}$. We again refer to these values as the *data sites* and $\{(\lambda_k, \theta_k)\}_{k=1}^{N_\mathrm{d}}$ as the *nodes*. Figure 2.3 illustrates this reconstruction problem, of which the main ingredient is the interpolation of a function defined on the unit sphere.

2.2.2.1 Fourier Models

Since the modeling problems involve interpolation on the unit circle in 2D and the unit sphere in 3D, a natural choice for constructing these interpolants are Fourier-based methods: trigonometric function for 2D objects and spherical harmonics for 3D objects. These methods have been used extensively for geometric modeling (see, for, example [76] and the references therein). We briefly review both of these interpolation techniques in the context of Figures 2.2 and 2.3.

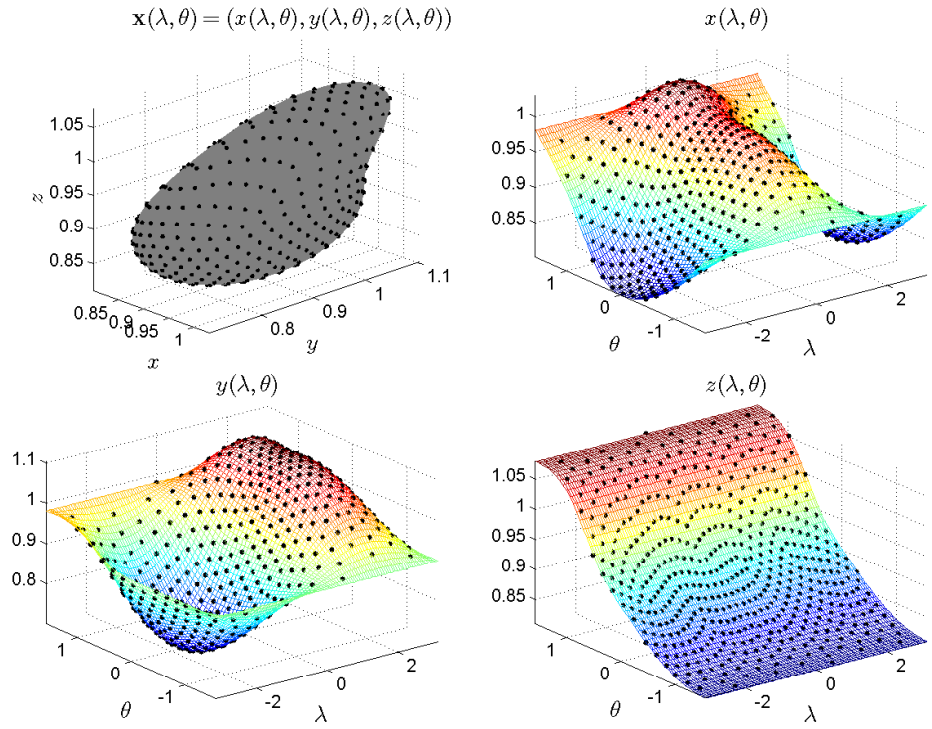

Figure 2.3: Illustration of the parametric representation of a 3D object $\boldsymbol{x}(\lambda, \theta)$ and the reconstruction from a finite number of data sites. Top left figure shows the 3D object together with discrete data sites represented as black solid spheres. Top right figure shows the x component of the object in spherical parametric space and its values at the node set $\{(\lambda_k, \theta_k)\}_{k=1}^{N_\mathrm{d}}$ (marked by black solid spheres), while the bottom left and right figures show the respective y and z components and their corresponding values. The goal is to reconstruct $x(\lambda, \theta)$, $y(\lambda, \theta)$, and $z(\lambda, \theta)$ from interpolations of the values at the node sets shown and then use these to reconstruct $\boldsymbol{x}(\lambda, \theta)$.

Using the notation from Figure 2.2, we first discuss the case of reconstructing the $x(\lambda)$ component of $\boldsymbol{x}(\lambda)$. In the case that the number of nodes N_{d} is even, we consider a trigonometric interpolant to these data of the form

$$p^x(\lambda) = c_0^x + \sum_{k=1}^{N_{\mathrm{d}}/2} c_{2k-1}^x \cos k\lambda + \sum_{k=1}^{N_{\mathrm{d}}/2-1} c_{2k}^x \sin k\lambda. \tag{2.5}$$

While there is an analogous formula for odd values of N_{d}, we omit this discussion and limit our current study to even values of N_{d}. The coefficients c_k^x are determined by the interpolation conditions $p^x(\lambda_k) = x(\lambda_k)$, $k = 1, \ldots, N_{\mathrm{d}}$. The solution to this problem can be written in terms of the following linear system:

$$\begin{bmatrix} 1 & \cos\lambda_1 & \sin\lambda_1 & \cdots & \cos\frac{N_{\mathrm{d}}-2}{2}\lambda_1 & \sin\frac{N_{\mathrm{d}}-2}{2}\lambda_1 & \cos\frac{N_{\mathrm{d}}}{2}\lambda_1 \\ 1 & \cos\lambda_2 & \sin\lambda_2 & \cdots & \cos\frac{N_{\mathrm{d}}-2}{2}\lambda_2 & \sin\frac{N_{\mathrm{d}}-2}{2}\lambda_2 & \cos\frac{N_{\mathrm{d}}}{2}\lambda_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 1 & \cos\lambda_N & \sin\lambda_N & \cdots & \cos\frac{N_{\mathrm{d}}-2}{2}\lambda_N & \sin\frac{N_{\mathrm{d}}-2}{2}\lambda_N & \cos\frac{N_{\mathrm{d}}}{2}\lambda_N \end{bmatrix} \begin{bmatrix} c_0^x \\ c_1^x \\ \vdots \\ c_{N_{\mathrm{d}}-1}^x \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_{\mathrm{d}}} \end{bmatrix}, \tag{2.6}$$

where $x_k = x(\lambda_k)$, $k = 1, \ldots, N_{\mathrm{d}}$. A similar construction to Equation (2.5) is given for the $y(\lambda)$ component of $\boldsymbol{x}(\lambda)$, which we denote by $p^y(\lambda)$. Our trigonometric representation of a 2D object like the one in Figure 2.2 is given by

$$\boldsymbol{p}(\lambda) = (p^x(\lambda), p^y(\lambda)). \tag{2.7}$$

We turn our attention now to interpolation with spherical harmonics and use the notation from Figure 2.3 to describe the reconstruction of the $x(\lambda, \theta)$ component of $\boldsymbol{x}(\lambda, \theta)$. The dimension of the space of all spherical harmonics of degree N_{s} is given by $(N_{\mathrm{s}} + 1)^2$. For simplicity, we thus restrict our attention to the case that the number of nodes is given by $N_{\mathrm{d}} = (N_{\mathrm{s}} + 1)^2$. In this case, we look for a spherical harmonic interpolant of the form

$$p^x(\lambda, \theta) = \sum_{\ell=0}^{N_{\mathrm{s}}} \left[\sum_{m=0}^{\ell} c_{\ell,2m}^x Y_\ell^{2m}(\lambda, \theta) + \sum_{m=1}^{\ell} c_{\ell,2m-1}^x Y_\ell^{2m-1}(\lambda, \theta) \right], \tag{2.8}$$

where Y_ℓ^{2m} and Y_ℓ^{2m-1} are defined as follows:

$$Y_\ell^{2m}(\lambda, \theta) := \sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}} \cos(m\lambda) P_\ell^m(\sin\theta), \quad m = 0, \ldots, \ell, \tag{2.9}$$

$$Y_\ell^{2m-1}(\lambda, \theta) := \sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}} \sin(m\lambda) P_\ell^m(\sin\theta), \quad m = 1, \ldots, \ell. \tag{2.10}$$

Here P_ℓ^m is an associated Legendre function of degree ℓ and order m. The coefficients c_k^x are determined by the interpolation conditions $p^x(\lambda_k, \theta_k) = x(\lambda_k, \theta_k)$, $k = 1, \ldots, N_\mathrm{d}$. The linear system corresponding to these conditions is given by

$$\begin{bmatrix} Y_0^0(\lambda_1,\theta_1) & Y_1^0(\lambda_1,\theta_1) & Y_1^1(\lambda_1,\theta_1) & Y_1^2(\lambda_1,\theta_1) & \cdots \\ Y_0^0(\lambda_2,\theta_2) & Y_1^0(\lambda_2,\theta_2) & Y_1^1(\lambda_2,\theta_2) & Y_1^2(\lambda_2,\theta_2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_0^0(\lambda_N,\theta_N) & Y_1^0(\lambda_N,\theta_N) & Y_1^1(\lambda_N,\theta_N) & Y_1^2(\lambda_N,\theta_N) & \cdots \end{bmatrix} \begin{bmatrix} c_1^x \\ c_2^x \\ \vdots \\ c_N^x \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_\mathrm{d}} \end{bmatrix}, \tag{2.11}$$

where $x_k = x(\lambda_k, \theta_k)$, $k = 1, \ldots, N_\mathrm{d}$. Unlike the trigonometric case, this linear system can be singular depending on how the nodes are arranged [16, §2]. We avoid this possibility by choosing the nodes in an "optimal" manner as discussed in Section 2.4.2. For a good review of the properties of spherical harmonic interpolants, see [89]. A similar construction to Equation (2.8) is given for the $y(\lambda, \theta)$ and $z(\lambda, \theta)$ components of $\boldsymbol{x}(\lambda, \theta)$, which we denote by $p^y(\lambda, \theta)$ and $p^z(\lambda, \theta)$. Our spherical harmonic representation of a 3D object like the one in Figure 2.3 is given by

$$\boldsymbol{p}(\lambda, \theta) = (p^x(\lambda, \theta), p^y(\lambda, \theta), p^z(\lambda, \theta)). \tag{2.12}$$

2.2.2.2 RBF Models

While Chapter 1 presents a detailed overview of RBF interpolation, we use a variant of the standard RBF interpolant in this chapter, and in Chapter 3, based on restriction of the RBF method to interpolation on a circle and on a sphere. This approach began to receive considerable attention from a theoretical standpoint starting in the mid-1990s (see [16, §6] for a discussion). When restricted to these domains, the RBF method is sometimes referred to as the *zonal basis function* (ZBF) or *spherical basis function* (SBF) method in the literature [88, Ch. 17]. We will, however, use the more popular term RBF to describe the interpolation technique. Several studies have been devoted to providing error estimates for RBF interpolation on circles and spheres; see, for example, [50, 63]. In the first of these papers, it is shown these interpolants can provide spectral accuracy provided the underlying target function is sufficiently smooth. The latter of these studies gives error estimates in the case that the target function belongs to some Sobolev space. As was mentioned in Chapter 1, the RBF method has been successfully used for approximating derivatives of scalar- and vector-valued quantities on the surface of a sphere and incorporated into methods for solving partial differential equations numerically in spherical geometries [38, 19, 20].

The construction of the 2D and 3D RBF models of the objects is similar, so we discuss them together. Using the notation of Figures 2.2 and 2.3, and focusing on the reconstruc-

tions of the $x(\lambda)$ and $x(\lambda,\theta)$ components of the objects, the corresponding RBF interpolants are given by

$$\text{2D}: \qquad s^x(\lambda) \quad = \sum_{k=1}^{N_\text{d}} c_k^x \phi\left(\sqrt{2-2\cos(\lambda-\lambda_k)}\right), \tag{2.13}$$

$$\text{3D}: \qquad s^x(\lambda,\theta) \quad = \sum_{k=1}^{N_\text{d}} c_k^x \phi\left(\sqrt{2(1-\cos\theta\cos\theta_k\cos(\lambda-\lambda_k)-\sin\theta\sin\theta_k)}\right). \tag{2.14}$$

Here ϕ is some scalar-valued, positive (semi-) definite radial kernel. The square root term in Equation (2.13) is just the Euclidean distance between the points described in polar coordinates by λ and λ_k, while the square root term in Equation (2.14) is similarly the Euclidean distance between the points described in spherical coordinates by (λ,θ) and (λ_k,θ_k). The coefficients c_k^x in either Equation (2.13) or Equation (2.14) are again determined by the interpolation conditions. These conditions lead to the following linear system of equations:

$$\underbrace{\begin{bmatrix} \phi(r_{1,1}) & \cdots & \phi(r_{1,N_\text{d}}) \\ \phi(r_{2,1}) & \cdots & \phi(r_{2,N_\text{d}}) \\ \vdots & \ddots & \vdots \\ \phi(r_{N_\text{d},1}) & \cdots & \phi(r_{N_\text{d},N_\text{d}}) \end{bmatrix}}_{A} \begin{bmatrix} c_1^x \\ c_2^x \\ \vdots \\ c_N^x \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \tag{2.15}$$

where $x_k = x(\lambda_k)$ and $r_{j,k} = \sqrt{2-2\cos(\lambda_j-\lambda_k)}$ for 2D objects, and $x_k = x(\lambda_k,\theta_k)$ and $r_{j,k}$ $= \sqrt{2(1-\cos\theta_j\cos\theta_k\cos(\lambda_j-\lambda_k)-\sin\theta_j\sin\theta_k)}$ for 3D objects. Note that $r_{j,k} = r_{k,j}$ so that the linear system Equation (2.15) is symmetric. More importantly, this linear system is guaranteed to be nonsingular for the appropriate choice of ϕ. In this study, we restrict our attention to the multiquadric (MQ) and inverse multiquadric (IMQ) radial kernels, which are popular in applications and are given explicitly by

$$\text{MQ:} \quad \phi(r) = \sqrt{1+(\varepsilon r)^2}, \tag{2.16}$$

$$\text{IMQ:} \quad \phi(r) = \frac{1}{\sqrt{1+(\varepsilon r)^2}}. \tag{2.17}$$

Here ε is called the shape parameter. For both the MQ and IMQ, the linear system Equation (2.15) is guaranteed to be nonsingular (provided $\varepsilon > 0$). Furthermore, for the IMQ, the A matrix in this linear system is guaranteed to be positive definite. A full discussion of the nonsingularity of Equation (2.15) for various radial kernels can be found in either [15] or [88]. We postpone the discussion of choosing ε to Section 2.4.3.2. However, we note that in the limit that $\varepsilon \to 0$ a RBF interpolant on a circle converges to a trigonometric interpolant, while a RBF interpolant on a sphere converges to a spherical harmonic interpolant [27] (strictly

speaking this was only shown for the case of the sphere, but the arguments from [27] carry directly over to the case of the circle as well). Thus, trigonometric and spherical harmonic interpolation can be viewed as a special case of RBF interpolation.

We denote the RBF representations of a 2D object like the one in Figure 2.2 by

$$\mathbf{s}_{\boldsymbol{B}}(\lambda) = (s^x(\lambda), s^y(\lambda)), \tag{2.18}$$

where $s^y(\lambda)$ interpolates $y(\lambda)$ and has the form of Equation (2.13). Similarly, we denote the RBF representation of a 3D object like the one in Figure 2.3 by

$$\mathbf{s}_{\boldsymbol{B}}(\lambda, \theta) = (s^x(\lambda, \theta), s^y(\lambda, \theta), s^z(\lambda, \theta)), \tag{2.19}$$

where $s^y(\lambda, \theta)$ and $s^z(\lambda, \theta)$ interpolate $y(\lambda, \theta)$ and $z(\lambda, \theta)$, respectively, and have the form of Equation (2.14). We conclude this section by noting that the RBF method is more flexible than the Fourier-based methods in regard to altering the parameterization for the objects. For example, if one were to find that a more general ellipse or ellipsoid provided a better parameterization of the object than a circle or sphere, then the RBF method can be naturally extended to this new parameterization. The only change to Equation (2.13) or Equation (2.14) would be to replace the distance measure in the argument of ϕ with the appropriate (Euclidean) distance measure on the target object for the parametrization. More general objects, including ones with higher genus, are also possible; see [31] for a theoretical and numerical discussion.

2.3 Immersed Boundary Modeling

In this section, we review the components needed for an immersed boundary model of platelets. Our focus here is on the computation of normal vectors and on the modeling of elasticity. For a discussion of how forces generated from immersed objects are transferred to the underlying Eulerian mesh and how the fluid velocity is updated, see, for example, [64].

Normal vectors do not play a prominent role in most traditional IB calculations. Our interest in them is motivated by their other uses in the modeling of platelet aggregation. In addition to the fluid-structure interactions modeled using the IB method, the platelet problem requires solution of advection-diffusion equations for chemicals in the fluid domain outside of the moving platelets, along with boundary conditions on the chemical concentration at the fluid-platelet interface. Normal vectors along the platelet boundary are needed for determining when an Eulerian grid point is inside or outside of the platelet, and for imposing the boundary conditions. For further discussion of this, see [93].

2.3.1 Components for 2D

We denote the 2D platelet using the parametric representation $\boldsymbol{x}(\lambda)$ given in Equation (2.3) and define

$$\boldsymbol{\tau} := \frac{\partial}{\partial\lambda}\boldsymbol{x}(\lambda) = \left(\frac{\partial}{\partial\lambda}x(\lambda), \frac{\partial}{\partial\lambda}y(\lambda)\right) = (\boldsymbol{\tau}_x, \boldsymbol{\tau}_y). \tag{2.20}$$

The unit tangent and normal vectors to $\boldsymbol{x}(\lambda)$ are then given as

$$\hat{\boldsymbol{\tau}} := \frac{\boldsymbol{\tau}}{\|\boldsymbol{\tau}\|} = (\hat{\boldsymbol{\tau}}_x, \hat{\boldsymbol{\tau}}_y), \tag{2.21}$$

$$\hat{\boldsymbol{\eta}} := (-\hat{\boldsymbol{\tau}}_y, \hat{\boldsymbol{\tau}}_x) \tag{2.22}$$

For the force model in 2D, we use the fiber model defined in [67]. According to this model, the elastic force density on $\boldsymbol{x}(\lambda)$ at the location $\boldsymbol{x}(\lambda_i)$ is given by

$$\mathbf{F}(\boldsymbol{x}(\lambda_i)) = \frac{\partial}{\partial\lambda}(T\hat{\boldsymbol{\tau}})\bigg|_{\lambda_i}, \tag{2.23}$$

where $T = K(\|\boldsymbol{\tau}\|)$ is the fiber tension. In our platelet model, we choose K as a linear function, $K = K_0\|\boldsymbol{\tau}\|$, where K_0 is the Hookean spring constant. In this case, Equation (2.23) reduces to

$$\mathbf{F}(\boldsymbol{x}(\lambda_i)) = K_0\,\frac{\partial}{\partial\lambda}\,(\|\boldsymbol{\tau}\|\hat{\boldsymbol{\tau}})\bigg|_{\lambda_i} = K_0\,\frac{\partial^2}{\partial\lambda^2}\boldsymbol{x}(\lambda)\bigg|_{\lambda_i}. \tag{2.24}$$

The 2D spring force model traditionally used in piecewise linear representations is a scaled second-order, central-difference approximation to the above fiber model (assuming springs of zero rest length). From the physical standpoint, each IB point in a 2D object is thought to be connected to each of its neighbors via springs. For tension forces, there are only two neighbors attached to each IB point via springs. This spring force is expressed as:

$$\mathbf{F}(\boldsymbol{x}_i) = K_0(\boldsymbol{x}_{i+1} - 2\boldsymbol{x}_i + \boldsymbol{x}_{i-1}). \tag{2.25}$$

2.3.2 Components for 3D

We denote the 3D platelet using the parametric representation $\boldsymbol{x}(\lambda, \theta)$ given in Equation (2.4) and define

$$\boldsymbol{\tau}^{\lambda} := \frac{\partial}{\partial\lambda}\boldsymbol{x}(\lambda,\theta) = \left(\frac{\partial}{\partial\lambda}x(\lambda,\theta), \frac{\partial}{\partial\lambda}y(\lambda,\theta), \frac{\partial}{\partial\lambda}z(\lambda,\theta)\right), \tag{2.26}$$

$$\boldsymbol{\tau}^{\theta} := \frac{\partial}{\partial\theta}\boldsymbol{x}(\lambda,\theta) = \left(\frac{\partial}{\partial\theta}x(\lambda,\theta), \frac{\partial}{\partial\theta}y(\lambda,\theta), \frac{\partial}{\partial\theta}z(\lambda,\theta)\right). \tag{2.27}$$

The unit tangent vectors to $\boldsymbol{x}(\lambda, \theta)$ are then given by

$$\hat{\boldsymbol{\tau}}^{\lambda} := \frac{\boldsymbol{\tau}^{\lambda}}{\|\boldsymbol{\tau}^{\lambda}\|} \quad \text{and} \quad \hat{\boldsymbol{\tau}}^{\theta} := \frac{\boldsymbol{\tau}^{\theta}}{\|\boldsymbol{\tau}^{\theta}\|}, \tag{2.28}$$

while the unit normal vector is given by

$$\hat{\boldsymbol{\eta}} := \frac{\boldsymbol{\tau}^{\lambda} \times \boldsymbol{\tau}^{\theta}}{\|\boldsymbol{\tau}^{\lambda} \times \boldsymbol{\tau}^{\theta}\|}. \tag{2.29}$$

The force model we use in 3D differs depending on whether a piecewise linear representation for the object is used or a parametric representation. Traditionally, piecewise linear representations (triangulated surfaces) in 3D have been used in conjunction with spring forces. In this model, a spring is assumed to be placed along each triangle edge (again, we assume these springs have a rest length of zero). Then, the total force acting on an IB point at $\boldsymbol{x}_i$ due to its k nearest neighbors is:

$$\mathbf{F}(\boldsymbol{x}_i) = K_0 \sum_{j \neq i} (\boldsymbol{x}_i - \boldsymbol{x}_j), \tag{2.30}$$

where the sum is over k IB points. The nearest neighbors are typically defined from the triangulation, *i.e.*, as members of the adjacency list of $\boldsymbol{x}_i$. This is the same strategy that we follow.

For our parametric representation of platelets, we use surface tension as the model to compute tension forces. The force due to surface tension is then given by

$$\mathbf{F} = \gamma(2H)\hat{\boldsymbol{\eta}}, \tag{2.31}$$

where γ is the coefficient of surface tension. H is the mean curvature of the surface, and can be computed as [41, §16.5]

$$H = \frac{eG - 2fF + gE}{2(EG - F^2)}, \tag{2.32}$$

where E, F, and G are coefficients of the first fundamental form,

$$E = \boldsymbol{\tau}^{\lambda} \cdot \boldsymbol{\tau}^{\lambda},\ F = \boldsymbol{\tau}^{\lambda} \cdot \boldsymbol{\tau}^{\theta},\ G = \boldsymbol{\tau}^{\theta} \cdot \boldsymbol{\tau}^{\theta}, \tag{2.33}$$

and e, f, and g are coefficients of the second fundamental form,

$$e = \left(\frac{\partial}{\partial \lambda}\boldsymbol{\tau}^{\lambda}\right) \cdot \hat{\boldsymbol{\eta}},\ f = \left(\frac{\partial}{\partial \theta}\boldsymbol{\tau}^{\lambda}\right) \cdot \hat{\boldsymbol{\eta}},\ g = \left(\frac{\partial}{\partial \theta}\boldsymbol{\tau}^{\theta}\right) \cdot \hat{\boldsymbol{\eta}}. \tag{2.34}$$

$$\tag{2.35}$$

2.4 Implementation Details

In this section, we present the implementation details for evaluating the positions on the Lagrangian objects, computing normals to the surface of the object, and computing the internal forces as presented in the previous section. For the piecewise linear representation,

these surface normals and forces are computed at the IB points. For the parametric representations using Fourier and RBF models, these values are computed at some set of *sample sites*, which do not necessarily correspond to the *data sites*. With these operations defined, it is possible to employ the traditional spreading and interpolation operators for transferring the forces and velocity respectively between the Lagrangian and Eulerian discretizations.

2.4.1 Piecewise Linear Models

In 2D, normals are computed at the IB points using the piecewise quadratic representation presented in Section 2.2.1. For each IB point, we first solve for the coefficients in Equation (2.1) and Equation (2.2) using the IB point and its two neighbors. Using Equation (2.1) and Equation (2.2), we next compute the tangent vector at each IB point using Equation (2.21) and then determine the normal vector using Equation (2.22).

In 3D, we compute the normal vectors at each IB point by first computing the normal vector at the circumcenter of each of the triangles. We then obtain the normal vector at a vertex (IB point) by a weighted average of the values of the normal vectors at the circumcenters of the triangles connected to the vertex. Specifically, we weight these facet normals by the angle at which that facet is incident on the vertex at which we require a normal. This approach takes into account the geometric configuration of each facet [81].

The implementation of the forces follows directly from the simple spring force model in both 2D Equation (2.25) and 3D Equation (2.30). We note that while the 2D implementation follows naturally from a constitutive model, the 3D implementation is a purely algorithmic extension of the 2D case.

2.4.2 Parametric Models

For the parametric models, we use the continuous representations of the objects from either the Fourier- or RBF-based interpolants to approximate the normal vectors and forces. This involves analytically computing derivatives of these interpolants and then evaluating the derivatives at some set of N_s locations in the parametric space that corresponds to the set of sample sites. In 2D, we denote the set of sample sites by $\{\boldsymbol{x}(\lambda_j^\mathrm{e})\}_{j=1}^{N_\mathrm{s}}$ and refer to the set of parametric values $\{\lambda_j^\mathrm{e}\}_{j=1}^{N_\mathrm{s}}$ as the *evaluation points*. Similarly for 3D, we denote the sample sites by $\{\boldsymbol{x}(\lambda_j^\mathrm{e}, \theta_j^\mathrm{e})\}_{j=1}^{N_\mathrm{s}}$ and refer to $\{(\lambda_j^\mathrm{e}, \theta_j^\mathrm{e})\}_{j=1}^{N_\mathrm{s}}$ as the *evaluation points*. The method we use is similar to the pseudospectral or spectral collocation method (*e.g.*, [25, 83]), except that the derivatives are not evaluated at interpolation nodes.

Before describing the implementation details for the Fourier and RBF models, we discuss the node and evaluation points used.

2.4.3 Node and Evaluation Points

For our 2D objects, we use N_{d} equally-spaced points on the interval $(-\pi, \pi]$ as the node set $\{\lambda_k\}_{k=1}^{N_{\mathrm{d}}}$, and take N_{d} to be even. This gives a uniform sampling in the parametric space and allows fast algorithms to be used for computing the interpolants as discussed below. Additionally, since the shape of our target objects are near circular or elliptical, these nodes give a good distribution of data sites on the object. We also use $N_{\mathrm{s}} >> N_{\mathrm{d}}$ equally-spaced points in the interval $(-\pi, \pi]$ as the set of evaluation points $\{\lambda_j^{\mathrm{e}}\}_{j=1}^{N_{\mathrm{s}}}$ since this also results in a set of sample sites that are well distributed over the object.

To get a good sampling of our nearly ellipsoidal or spherical objects in 3D, we cannot resort to using equally spaced points in the spherical coordinate system as our node sets $\{(\lambda_k, \theta_k)\}_{k=1}^{N_{\mathrm{d}}}$ because of the inherent "pole problem". Instead we use node sets that give a quasi-uniform distribution of data sites on the unit sphere. Since only a maximum of 20 points can be evenly distributed on a sphere, there are a myriad of methods to define and generate a quasi-uniform distribution for larger numbers of points [45]. We use two of these methods: maximal determinant (MD) for our spherical harmonic models and minimal energy (ME) for our RBF models. Both of these methods are discussed in [89] and many of these two point sets for various N_{d} can be downloaded from [90]. The MD points are generated by finding a distribution of points that maximize the determinant of a certain "Gram matrix" related to Equation (2.11). The ME points are generated by finding a distribution of nodes that minimize an electrostatic type energy potential. For spherical harmonic interpolation, the MD points lead to much better results both in terms of accuracy and stability [89]. For RBF interpolation, the ME points typically yield better results in terms of accuracy [19, 20] for larger shape parameters ε. For smaller values, the MD points give better results because of the connection to spherical harmonics as $\varepsilon \to 0$ [27]. For the set of evaluation points, $\{(\lambda_j^{\mathrm{e}}, \theta_j^{\mathrm{e}})\}_{j=1}^{N_{\mathrm{s}}}$, we use $N_{\mathrm{s}} >> N_{\mathrm{d}}$ minimum energy points for both the spherical harmonic and RBF models, which again results in a well-distributed set of sample sites on the object.

2.4.3.1 Fourier Models

The first step in computing the normal vectors and forces for the 2D trigonometric model Equation (2.7) is to compute the interpolation coefficients c_k^x and c_k^y, $k = 1, \ldots, N_{\mathrm{d}}$ (see Equation (2.5)). Since we are using equally spaced node points $\{\lambda_k\}_{k=1}^{N_{\mathrm{d}}}$, we can avoid

having to solve Equation (2.6) directly for these coefficients and can instead compute them by means of the fast Fourier transforms (*e.g.*, [83, §3]) at a cost of $O(N_{\mathrm{d}} \log N_{\mathrm{d}})$.

We next compute the derivatives of the interpolants to obtain the following approximation to Equation (2.20):

$$\frac{\partial}{\partial \lambda} \boldsymbol{x}(\lambda)\big|_{\lambda=\lambda_j^{\mathrm{e}}} \approx \frac{\partial}{\partial \lambda} \boldsymbol{p}(\lambda)\big|_{\lambda=\lambda_j^{\mathrm{e}}},\ j = 1, \ldots, N_{\mathrm{s}}. \tag{2.36}$$

We then determine the normal vector at $\boldsymbol{x}(\lambda_j^{\mathrm{e}})$ by normalizing the vector above and switching the components according to Equation (2.21) and Equation (2.22). We similarly obtain an approximation of the force Equation (2.24) from the second derivative of the interpolants:

$$\frac{\partial^2}{\partial \lambda^2} \boldsymbol{x}(\lambda)\big|_{\lambda=\lambda_j^{\mathrm{e}}} \approx \frac{\partial^2}{\partial \lambda^2} \boldsymbol{p}(\lambda)\big|_{\lambda=\lambda_j^{\mathrm{e}}},\ j = 1, \ldots, N_{\mathrm{s}}. \tag{2.37}$$

For the 3D spherical harmonic model Equation (2.12), the first step in computing the normal vectors and forces is again to compute the interpolation coefficients c_k^x, c_k^y, and c_k^z, $k = 1, \ldots, N_{\mathrm{d}}$, (see Equation (2.8)). Unlike the trigonometric interpolant, there are unfortunately no fast algorithms for computing these coefficients. Since we use relatively small values of N_{d}, we thus resort to determining the coefficients by solving the linear system Equation (2.11) using a direct LU factorization of the interpolation matrix. By using the MD points as the nodes in this model, we are guaranteed that this system is nonsingular and relatively well conditioned [89]. We note that in context of the IB method simulation, the node points will stay fixed throughout the simulation so that the LU factorization of the interpolation matrix from Equation (2.11) needs to be done only once at the initial time-step and then stored. Thus, for all other time-steps, the coefficients can be determined in $O(N_{\mathrm{d}}^2)$ computations.

After the coefficients are determined, we compute the following six derivatives to obtain approximations to Equation (2.26) and Equation (2.27):

$$\frac{\partial}{\partial \lambda} \boldsymbol{x}(\lambda, \theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})} \approx \frac{\partial}{\partial \lambda} \boldsymbol{p}(\lambda, \theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})},\ j = 1, \ldots, N_{\mathrm{s}}, \tag{2.38}$$

$$\frac{\partial}{\partial \theta} \boldsymbol{x}(\lambda, \theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})} \approx \frac{\partial}{\partial \theta} \boldsymbol{p}(\lambda, \theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})},\ j = 1, \ldots, N_{\mathrm{s}}. \tag{2.39}$$

We then compute the normal vectors using these approximations in Equation (2.28) and Equation (2.29).

The computation of the force requires the approximation to the normal vectors and an approximation to the mean curvature Equation (2.32). For the values of E, F, and G in the

mean curvature computation (see Equation (2.33)), we use the approximations Equation (2.38) and Equation (2.39). For the values of e, f, and g, we use the approximations

$$\frac{\partial^2}{\partial\lambda^2}\boldsymbol{x}(\lambda,\theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})} \approx \frac{\partial^2}{\partial\lambda^2}\boldsymbol{p}(\lambda,\theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})},\ j=1,\ldots,N_{\mathrm{s}}, \tag{2.40}$$

$$\frac{\partial^2}{\partial\theta\partial\lambda}\boldsymbol{x}(\lambda,\theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})} \approx \frac{\partial^2}{\partial\theta\partial\lambda}\boldsymbol{p}(\lambda,\theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})},\ j=1,\ldots,N_{\mathrm{s}}, \tag{2.41}$$

$$\frac{\partial^2}{\partial\theta^2}\boldsymbol{x}(\lambda,\theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})} \approx \frac{\partial^2}{\partial\theta^2}\boldsymbol{p}(\lambda,\theta)\big|_{(\lambda,\theta)=(\lambda_j^{\mathrm{e}},\theta_j^{\mathrm{e}})},\ j=1,\ldots,N_{\mathrm{s}}. \tag{2.42}$$

2.4.3.2 RBF Models

The normal vectors and forces for the RBF models are computed in the same fashion as for the Fourier models discussed above; one just needs to replace the Fourier interpolants $\boldsymbol{p}(\lambda)$ and $\boldsymbol{p}(\lambda,\theta)$ with the RBF interpolants $\mathbf{s}_{\boldsymbol{B}}(\lambda)$ from Equation (2.18) and $\mathbf{s}_{\boldsymbol{B}}(\lambda,\theta)$ from Equation (2.19), respectively. We thus omit a full description. We will, however, discuss the shape parameter ε and the computation of the interpolation coefficients.

As was mentioned in Chapter 1, infinitely smooth radial kernels like the MQ Equation (2.16) and IMQ Equation (2.17) feature a free shape parameter ε. It has generally been reported in the literature that there is typically an optimal value of ε that produces the best accuracy in the interpolants with these kernels and that this value tends to decrease with increasing smoothness of the underlying function being approximated (*e.g.*, [69]). However, as ε decreases to zero, these smooth kernels become increasingly flat and the shifts of ϕ in Equation (2.13) and Equation (2.14) become less and less distinguishable from one another. If one follows the direct approach of solving for the expansion coefficients via Equation (2.15) and then evaluating the interpolant via Equation (2.13) or Equation (2.14) (which is denoted by RBF-Direct in the current literature) for ε in this flat regime, then ill-conditioning can entirely contaminate the computation. For RBF interpolation on a sphere, this ill-conditioning can be completely bypassed by replacing the RBF-Direct algorithm with the RBF-QR algorithm of Fornberg and Piret [27]. The framework for this algorithm can also naturally be adapted to the task of computing RBF interpolants on the unit circle in a stable manner for all ε.

We have implemented both the RBF-QR algorithm and the RBF-Direct approach and present results in Sections 2.5.2.1 and 2.6.2.1 illustrating the behavior of the RBF interpolants for the full range of ε and the connection to Fourier-based methods. However, we have opted to use the RBF-Direct approach in implementation since it is computationally more efficient and the coding is much less involved for computing the normals and forces. Additionally, we have found that with the RBF-Direct approach and the values of N_{d} that

we considered, it is possible to get as good or better results than the Fourier-based methods. For increasingly large values of N_d, or objects whose parameterizations are very smooth, it may be necessary to switch to the RBF-QR algorithms to exploit the better accuracy that can sometimes be achieved for increasingly small values of ε.

For the RBF-Direct approach, the interpolation coefficients for both the 2D and 3D objects can be determined by solving the linear system Equation (2.15) (with the appropriate choice of $r_{j,k}$ for the dimension of interest). In the case of 2D objects with equally spaced points, solving this system directly can be bypassed by means of the fast Fourier transform and the coefficients can be computed in $O(N_\mathrm{d} \log N_\mathrm{d})$ operations [46]. This follows by observing that the matrix in Equation (2.15) is *circulant* (for any radial kernel ϕ) and can be diagonalized via the discrete Fourier transform matrix [40, §4.7.7]. For the 2D models, we use the MQ radial kernel Equation (2.16).

As in the case of the spherical harmonic model, there are no fast direct algorithms for determining the interpolation coefficients for the 3D RBF model Equation (2.19) and we thus resort to using a direct method. However, unlike the spherical harmonic model, the system is symmetric and, as discussed in Section 2.2.2.2, for the right choice of ϕ, is positive definite. Thus, a Cholesky factorization of the matrix can be used which reduces the memory costs and the need for pivoting over the LU factorization method used in the spherical harmonic model. We also note that the initial cost of computing the Cholesky factorization is lower than the LU factorization, but since this is only done once initially, there is no real savings in an IB simulation. We have opted to use the IMQ kernel Equation (2.16) to exploit the use of the Cholesky factorization.

2.5 2D Platelet Modeling Results

In this section, we present the results of our comparative study between using the piecewise linear approach as traditionally used within the IB method and our two alternative parametric approaches in 2D: RBF and Fourier (trigonometric polynomials) interpolation. Recall that within an IB time-step, the typical procedure employed is as follows. Given the locations of the immersed boundaries, both the normals and forces on an object are computed. The forces are then projected to an Eulerian grid and used as right-hand-side forcing to the Navier-Stokes equations. Based upon an update velocity field, the positions of the IB points are updated. In our comparison, we thus examine the geometric modeling capabilities, accuracy of the normal computations, and accuracy of the computation of the forces. As discussed in the previous section, we distinguish between the data sites and

sample sites for the parametric models. Data sites are the positions along the object at which the parametric models are interpolating. It is at these positions that we propose updating the geometric information of the object (for instance, at the conclusion of a time-step when the object's movement within the flow field is updated). Sample sites (which are normally more numerous compared to the data sites) are the positions along the object at which normals and forces are computed. It is from these positions that we propose projecting the IB forces. In all experiments, 100 sample sites are used as this represents the typical number of IB points that would be used per platelet object in a traditional 2D immersed boundary computation (and hence a reasonable standard against which to compare our new methods for the purposes of determining the feasibility of replacement). All errors are computed by taking the maximum of the two-norm difference between the approximations and the true values.

2.5.1 Test Cases

We consider two prototypical test objects and define them based upon perturbations of idealized shapes (an ellipse and a circle). Let $\mathbf{x}_{ideal}$ be a function representing the idealized, unperturbed shapes as given by the following equation:

$$\boldsymbol{x}_{ideal} = (x_c + a\cos\lambda, y_c + b\sin\lambda) \tag{2.43}$$

where $-\pi \leq \lambda \leq \pi$. Here (x_c, y_c) denotes the object center and a and b denote the radii. The two objects used for our comparison are defined as follows:

$$\underline{\text{Object 1}}: \quad \boldsymbol{x}_{2d\,obj1} = \left[1.0 + A\exp\left(\frac{-(1-\cos\lambda)^2}{\sigma_1}\right)\right]\boldsymbol{x}_{ideal}, \tag{2.44}$$

$$\underline{\text{Object 2}}: \quad \boldsymbol{x}_{2d\,obj2} = \left[1.0 + B\exp\left(\frac{(-(1-\cos^2\lambda)^{1.5})}{\sigma_2}\right)\right]\boldsymbol{x}_{ideal}. \tag{2.45}$$

For Object 1, we use the following parameters: $x_c = y_c = 0.9$, $a = 0.04$, $b = 0.05$, $A = 0.09$, and $\sigma_1 = 0.1$. For Object 2, we use the following parameters: $x_c = y_c = 0.2$, $a = b = 0.1$, $B = 0.04$, and $\sigma_2 = 0.9$.

Figure 2.4 displays the two test objects Equation (2.44) and Equation (2.45). Object 1 is a smooth (in terms of regularity) yet highly perturbed ellipse, while Object 2 is a rough perturbation of a circle. It can be shown that the parameterization Equation (2.45) for this object has only two continuous derivatives.

2.5.2 Comparison of Reconstructing the Objects

RBF and Fourier approaches. In Figure 2.5 we present the errors in reconstructing the objects as a function of the number of data sites. The error at the sample sites gives an

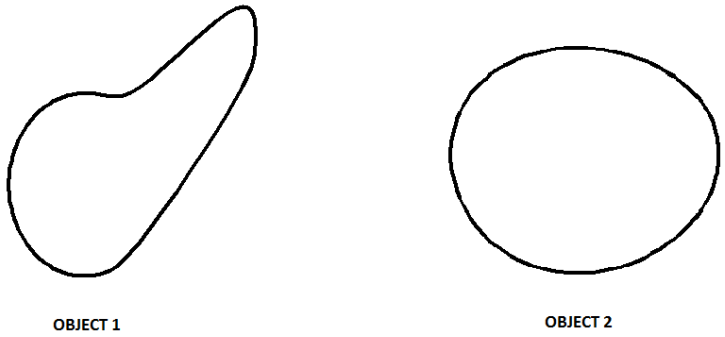

Figure 2.4: The test objects Equation (2.44) and Equation (2.45) for the 2D study.

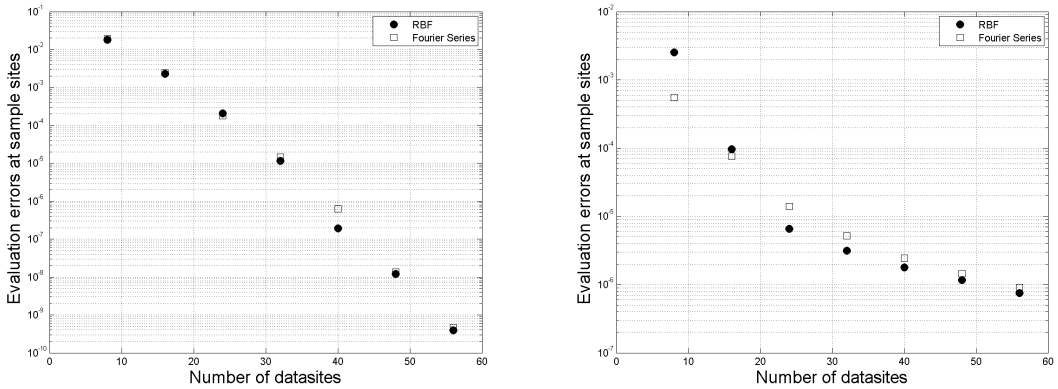

Figure 2.5: Error in the reconstruction of the shape of the objects (left is Object 1 and right is Object 2) evaluated at $N_s = 100$ sample sites as a function of the number of data sites. Circles denote the errors in the RBF model and squares denote the errors for the Fourier model. For the RBF model, the shape parameter for Object 1 was set to $\varepsilon = 0.9$ and for Object 2, it was set to $\varepsilon = 3.6$.

indication of the modeling capability of the RBF and Fourier methods. We can see from this figure that both the RBF and Fourier models are converging at a spectral rate for Object 1 (left figure), but at a much slower rate for Object 2 (right figure). This is expected since Object 1 is infinitely smooth, while Object 2 has only two continuous derivatives. The RBF and Fourier models perform similarly for Object 1. For Object 2, the RBF model shows better reconstruction properties as the number of sample sites increases above 20. No direct comparison with the piecewise linear model is given as the piecewise linear IB method always samples at the interpolating points.

2.5.2.1 Shape Parameter Study

In this section, we examine the impact of the shape parameter on the reconstruction errors of the RBF model. Figure 2.6 (left) displays the reconstruction errors for the two objects as a function of the shape parameter using $N_{\mathrm{d}} = 24$ data sites. A similar comparison for $N_{\mathrm{d}} = 56$ data sites is given in Figure 2.6 (right). For $\varepsilon \lesssim 0.85$, it was necessary to use the RBF-QR algorithm [27] (adapted to the unit circle) to compute the model in a numerically stable manner for the $N_{\mathrm{d}} = 56$ case. We can see from both figures that the errors are smallest for $\varepsilon \approx 0$ for the smooth Object 1 and increase quite dramatically as ε increases. For the rough Object 2, there is a much larger range of ε for which the errors are small, and this range includes values for which the RBF-Direct approach can be used without issues of numerical instabilities.

We used Figure 2.6 to help guide our selection of ε for the numerical experiments. However, we found from extensive tests on other objects that if the object is smooth, and RBF-Direct is to be used, then one generally wants to choose ε as small as the numerical conditioning allows. For rough objects, there is much more freedom in the choice and the results will not vary that greatly. It is unclear if we should expect smooth or rough objects in an IB simulation.

We conclude this section by noting that there are several algorithms that have been devoted to selecting an "optimal" shape parameter [15, §17]. However, these are too costly to be used every time-step of an IB simulation. We are thus advocating using a fixed ε for all time-steps. This value could be selected based on an expected typical shape for the

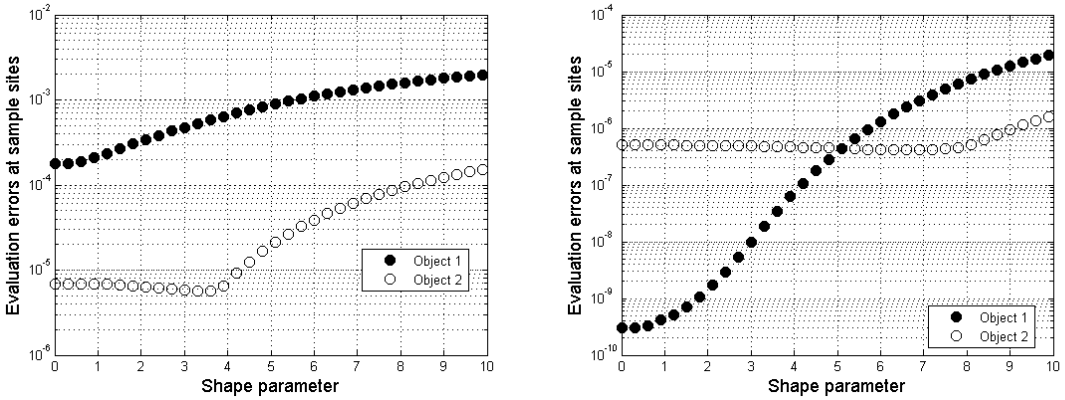

Figure 2.6: The figure on the left shows errors in the RBF reconstructions of the objects using $N_{\mathrm{d}} = 24$ data sites and $N_{\mathrm{s}} = 100$ sample sites as a function of the shape parameter. The figure on the right shows errors in the RBF reconstructions using $N_{\mathrm{d}} = 56$ data sites and $N_{\mathrm{s}} = 100$ sample sites as a function of the shape parameter.

immersed objects and one of the algorithms from [15, §17] or from trial and error. We will report on these strategies in the next chapter.

2.5.3 Comparison of Normal Vectors and Forces

We next focus on the errors in the parametric models in the approximation of the normal vectors to the objects and the forces. In this case, we compare the results to the traditional piecewise linear models.

Figure 2.7 displays the errors in the normal vectors at $N_{\mathrm{s}} = 100$ sample sites as a function of the number of data sites N_{d} for both the RBF and Fourier models. A solid line denoting the errors in the normal vectors for 100 IB points is given for comparison using the method for the piecewise linear models discussed in Section 2.4.1. We can see from this figure that at about $N_{\mathrm{d}} = 18$ data sites, the errors for both the RBF and Fourier models of Object 1 are lower than the piecewise linear model. The errors are similar between both parametric models and decrease rapidly with increasing N_{d}. The results for Object 2 are even more favorable for the parametric models compared to the piecewise linear model. For increasing N_{d}, the RBF model appears to have an advantage over the Fourier model.

We lastly examine the errors in the force computation incurred by the two parametric models and the traditional piecewise linear model. Figure 2.8 shows the errors in forces evaluated at 100 sample sites as a function of the number of data sites N_{d}. In all experiments, the force constant K_0 is set to 0.2. The solid line in Figure 2.8 denotes the error for the

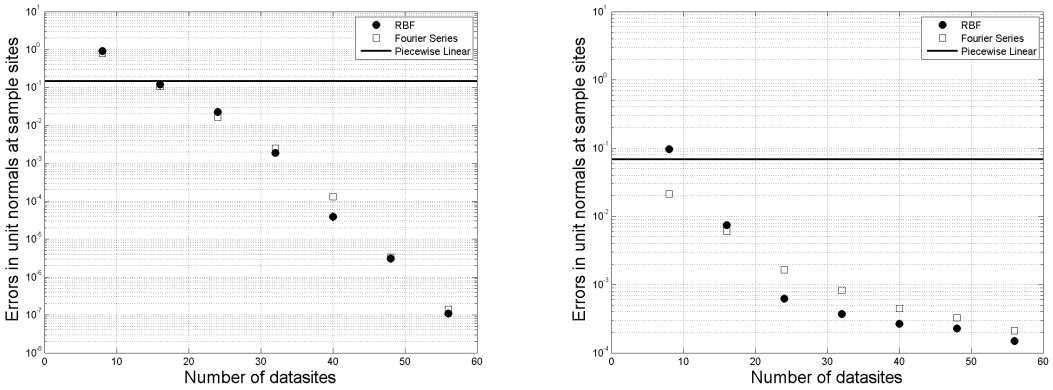

Figure 2.7: Errors in the approximations of the normal vectors to the objects at 100 sample sites as a function of the number of data sites N_{d}. The left plot is for Object 1, while the right one is for Object 2. The line denotes the error for the method used in the piecewise linear model with 100 IB points. Circles denote the errors for the RBF model and squares denote the Fourier model. For the RBF model, $\varepsilon = 0.9$ for Object 1 and $\varepsilon = 3.6$ for Object 2.

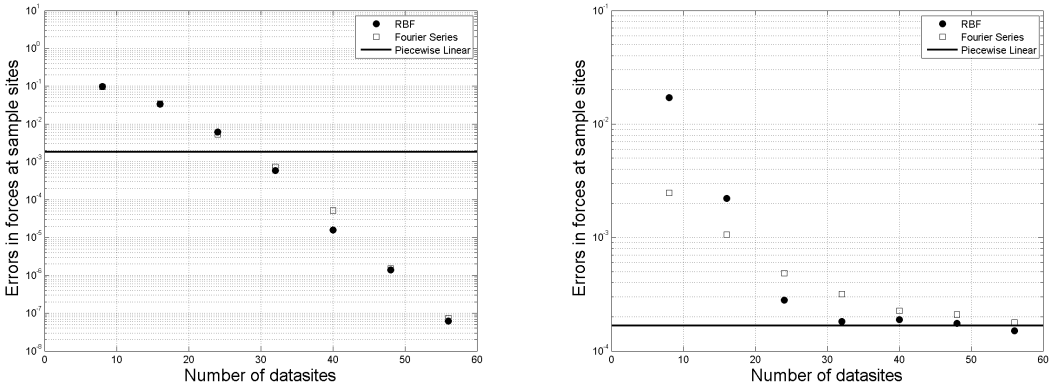

Figure 2.8: Errors in the approximation of the forces evaluated at $N_s = 100$ sample sites as a function of the number of data sites N_d for Object 1 (left) and Object 2 (right). The black line denotes the errors for a piecewise linear model with 100 IB points. Circles denote the errors for the RBF model and squares denote the errors for the Fourier model. For the RBF model, $\varepsilon = 0.9$ for Object 1 and $\varepsilon = 3.6$ for Object 2.

piecewise linear model computed at 100 IB points. For Object 1, we can see from the left plot of this figure that the errors for both parametric models are lower than the piecewise linear model starting at about $N_d = 30$ data sites. Again, both the RBF and Fourier models give similar results for this object. For the rough Object 2, it requires about $N_d = 32$ data sites for the RBF model to match the errors of the piecewise linear model, while it takes approximately $N_d = 56$ data sites for the Fourier model to give similar errors. We note that the errors for both the RBF and Fourier models do not fall as sharply for the rough Object 2 as the number of datasites is increased. This is because Object 2 is generated from a function that has only two derivatives, and the force computation involves computing a second derivative. It therefore follows that these global methods would therefore not converge as they would in the case of the smooth Object 1.

2.5.4 Comparison of the Computational Cost

We conclude the 2D results experiments with an examination of the computational cost associated with the three methods. We measure the computational cost as the elapsed wallclock time required to compute the interpolation coefficients, evaluate the interpolants, compute the normal vectors, and compute the forces. Under the assumption that all objects will be evaluated at the same parametric sites at each time-step, for both parametric models, we precompute the matrices for evaluating the interpolants, the derivatives, and the force operator once the interpolation coefficients have been determined (see Section 2.4.2 for details). We do not account for this setup time in our timing results.

Since for the piecewise linear model the number of evaluation sites is the same as the number of data sites, the total computational cost includes only the time required to compute the normal vectors and forces (see Section 2.4.1 for details).

All computations were performed in MATLAB version 7.10.0499 (64-bit) on a Windows desktop with a Intel Core i7 Sandy Bridge 3.4 GHz processor and 4 GB of 1600 MHz RAM. Times were measured using the tic and toc functions in MATLAB . All results presented are averages of 100 trials and are in units of seconds.

Figure 2.9 displays the elapsed time between the RBF, Fourier, and traditional piecewise linear models. The results for the RBF and Fourier models are displayed as a function of the number of data sites N_{d} for a fixed number of $N_{\mathrm{s}} = 100$ sample sites. The results for the piecewise linear model are for a fixed number of 100 IB points. We can see from the figure that the parametric models require significantly less time than the piecewise linear model. For $N_{\mathrm{d}} = 56$ data sites, the parametric models are over one order of magnitude faster.

We note that all the evaluation and derivative computations for the parametric models can be formulated in terms of matrices in order to avoid the need to first solve for the coefficients every time-step of the IB simulation. Thus, the results we present are not optimal in terms of computational time. If, however, during the IB simulation the sample sites change, then the step of going first through the coefficients as we have done will be necessary.

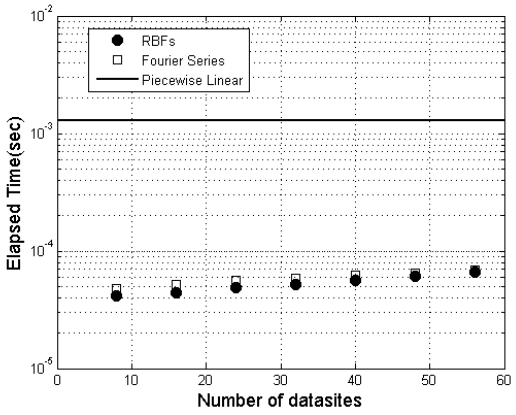

Figure 2.9: Elapsed wallclock time (in seconds) for one object to perform interpolation, evaluation, the computation of normal vectors, and the computation of forces at $N_{\mathrm{s}} = 100$ sample sites as a function of the number of data sites N_{d}. The piecewise linear computations were done with 100 IB points for comparison.

2.6 3D Platelet Modeling Results

Following a similar approach to the last section, we present here the results from a comparative study between using the traditional piecewise linear approach as used within the IB method and our two alternative parametric approaches in 3D: RBF and Fourier (spherical harmonics) interpolation. We examine the reconstruction capabilities of the models and the accuracy in computing normal vectors and forces. As in the 2D tests, we distinguish between data sites and sample sites. In all experiments unless otherwise specified, $N_s = 1024$ sample sites are used as this represents the typical number of IB points that would be used per platelet object in a traditional 3D IB computation (and hence a reasonable standard against which to compare our new methods for the purposes of determining the feasibility of replacement). All errors are computed by taking the maximum of the two-norm difference between the approximations and the true values.

2.6.1 Test Cases

We again consider two prototypical test objects and define them based on perturbations of idealized shapes (an ellipsoid and a sphere). Let $\mathbf{x}_{ideal}$ be a function representing the idealized, unperturbed shapes as given by the following equation:

$$\boldsymbol{x}_{ideal} = (x_c + a\cos\lambda\cos\theta, y_c + b\sin\lambda\cos\theta, z_c + c\sin\theta), \tag{2.46}$$

where $-\pi \leq \lambda \geq \pi$ and $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$. Here (x_c, y_c, z_c) denotes the object center, a and b are the equatorial radii, and c is the polar radius. The two objects used for our comparison are defined as follows:

$$\underline{\text{Object 1}}: \quad \boldsymbol{x}_{3d\,obj1} = \left[1.0 + A\exp\left(\frac{r_c^2}{\sigma_1}\right)\right]\boldsymbol{x}_{ideal}, \tag{2.47}$$

$$\underline{\text{Object 2}}: \quad \boldsymbol{x}_{3d\,obj2} = \left[1.0 + B\exp\left(\frac{r_c^{2.5}}{\sigma_2}\right)\right]\boldsymbol{x}_{ideal}. \tag{2.48}$$

where $r_c = 1 - \cos\theta\cos\theta_c\cos(\lambda - \lambda_c) - \sin\theta\sin\theta_c$. For Object 1, we use the following parameters: $x_c = y_c = z_c = 0.9$, $a = 0.1$, $b = 0.2$, $c = 0.09$, $A = 0.09$, and $\sigma_1 = 0.2$. For Object 2, we use the following parameters: $x_c = y_c = 0.1$, $z_c = 0.2$, $a = b = c = 0.1$, $B = 0.04$, and $\sigma_2 = \frac{16}{25}$. For both objects, $\lambda_c = 0$ and $\theta_c = \frac{\pi}{2}$.

Figure 2.10 displays the two test objects Equation (2.47) and Equation (2.48). Object 1 is a smooth (in terms of regularity) yet highly perturbed ellipsoid, while the Object 2 is a rough perturbation of a sphere. It can be shown that the parameterization Equation (2.48) has only three continuous derivatives.

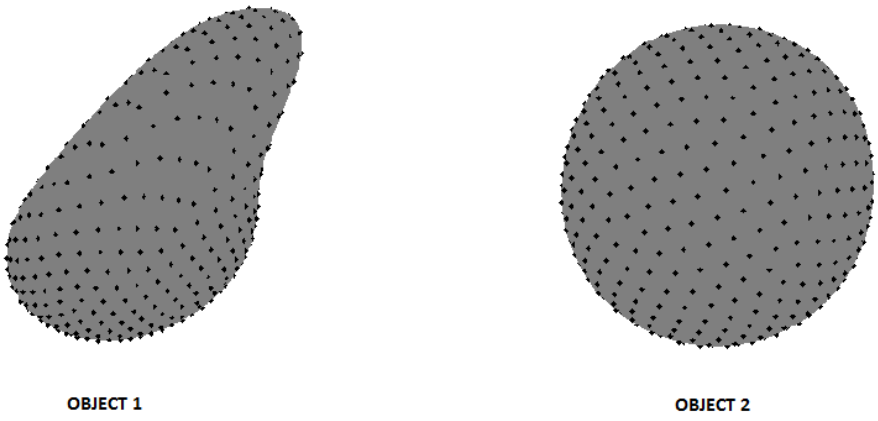

Figure 2.10: The test objects Equation (2.47) and Equation (2.48) for the 3D study.

2.6.2 Comparison of Reconstructing the Objects

As in the 2D results, we first examine the errors in reconstructing the objects using the two parametric models. Figure 2.11 displays the errors in reconstructing the objects as a function of the square root of the number of data sites N_{d}. We use $\sqrt{N_{\mathrm{d}}}$ since these are 2D objects and thus the reciprocal of this value gives a good measure of the spacing between data sites. These errors give a indication of the modeling capability of the RBF and Fourier

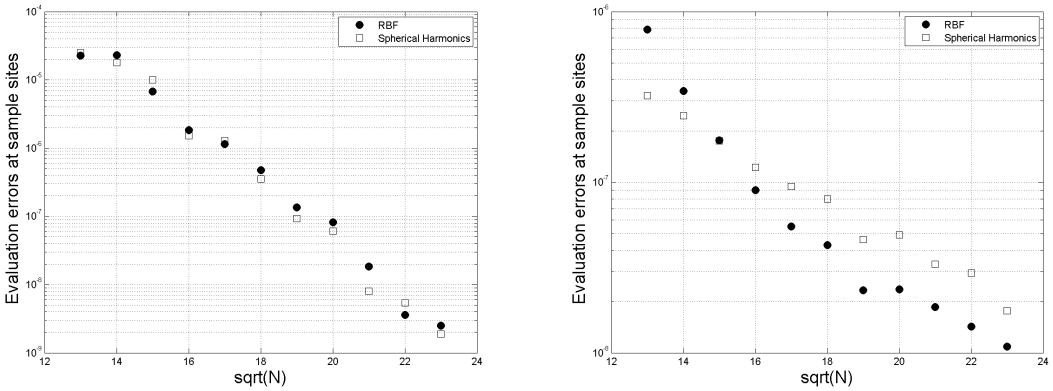

Figure 2.11: Error in the reconstruction of the shape of the objects (left is Object 1 and right is Object 2) evaluated at $N_{\mathrm{s}} = 1024$ sample sites as a function of the square root of the number of data sites N_{d}. Circles denote the errors in the RBF model and squares denote the errors for the Fourier model. For the RBF model, the shape parameter for Object 1 was set to $\varepsilon = 0.9$ and for Object 2 was set to $\varepsilon = 1.5$. Data sites for the RBF model are the ME points, while the data sites for the Fourier model are the MD points.

methods.

The results are similar to what we observed in 2D. For the smooth Object 1 (left plot in Figure 2.11), the RBF and Fourier models are converging at a spectral rate, but at a much slower rate for rough Object 2. The RBF and Fourier models are giving similar errors for Object 1, with a few values of N_{d} where the spherical harmonic method is clearly better. For Object 2, the RBF model consistently gives better results than the spherical harmonic model as N_{d} increases. No direct comparison with the piecewise linear model is given as the piecewise linear IB method always samples at the interpolating points.

2.6.2.1 Shape Parameter Study

Figures 2.12 and 2.13 display the reconstruction errors of the RBF model for the two objects as a function of the shape parameter using $N_{\mathrm{d}} = 256$ data sites and $N_{\mathrm{d}} = 529$ data sites, respectively. The left plot of each of these figures contains the results for the ME points, while the right plot contains the results for the MD points. For $\varepsilon \lesssim 0.85$, it was necessary to use the RBF-QR algorithm [27] to compute the model in a numerically stable manner for the $N_{\mathrm{d}} = 529$ case.

We see similar results to the 2D shape parameter study from Section 2.5.2.1. For the smooth Object 1 and the MD points, the errors decrease rapidly as ε decreases and reach a minimum near $\varepsilon = 0$ (at $\varepsilon = 0$ in the $N_{\mathrm{d}} = 256$ case), which correspond to a spherical harmonic interpolant on these nodes. For the ME points, we see the error rise right as ε gets to zero. For the rough Object 2 and both types of nodes, we see that the error reaches a minimum at a larger value of ε that is well within the numerically safe range of RBF-Direct.

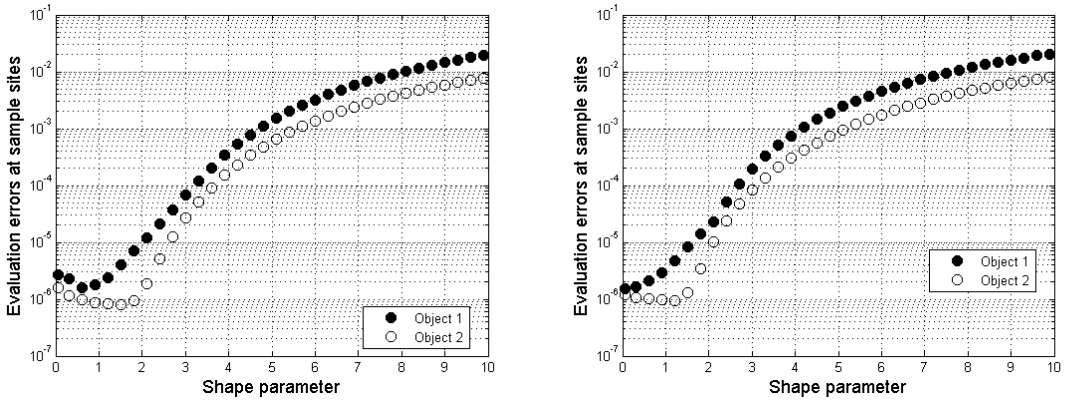

Figure 2.12: Error in the shape at 1024 sample sites as a function of the shape parameter for 256 data sites on Object 1 (solid circles) and Object 2 (open circles) using minimal energy points (left) and maximal determinant points (right) for the data sites.

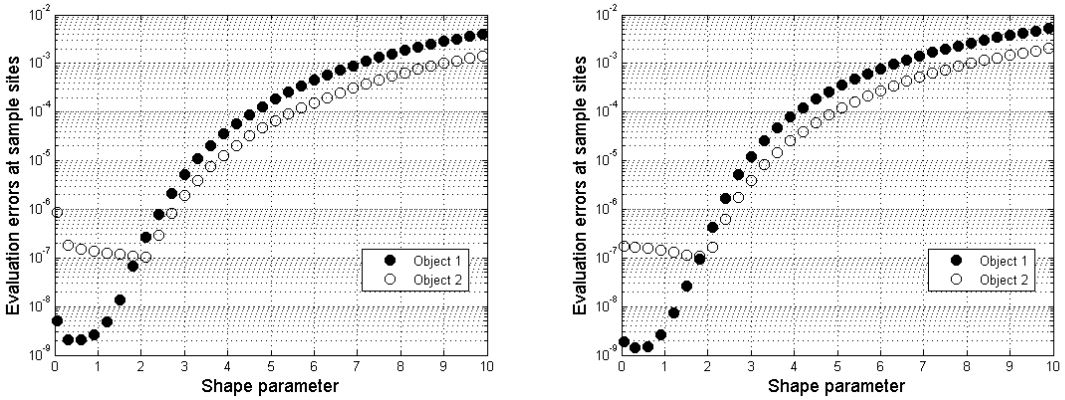

Figure 2.13: Error in the shape at 1024 sample sites as a function of the shape parameter for 529 data sites on Object 1 (solid circles) and Object 2 (open circles) using minimal energy points (left) and maximal determinant points (right) for the data sites.

The errors then increase slightly as ε decreases toward zero (with a jump up at $\varepsilon = 0$ in the case of the ME points). From both Figures 2.12 and 2.13, we see that the errors in the RBF model are much better for the MD points when ε is near zero, but as ε increases away from zero, the errors are better for the ME points.

We make similar comments to those at the end of Section 2.5.2.1 in regards to selection of the shape parameter for the 3D case, and thus refer the reader there.

2.6.3 Comparison of Normal Vectors and Forces

We next focus on the errors in the parametric models in the approximation of the normal vectors to the objects and the forces. In the case of computing the normal vectors, we compare the results to the traditional piecewise linear models based on triangulations of the surface. As discussed in Section 2.4.1, a comparison against the traditional piecewise linear 3D force model is not appropriate since this model is described purely algorithmically, and hence the underlying material constitutive model is not known and cannot be computed exactly.

Figure 2.14 displays the errors in the normal vectors at $N_{\mathrm{s}} = 1024$ sample sites as a function of the square root of the number of data sites N_{d}. The solid and dashed lines in both plots from this figure denote the errors in the normal vectors at 1024 and 10242 IB points. We see that increasing the number of IB points, decreases the errors in the normal vectors. However, unlike the 2D case, both parametric models always give better results in the normal vector computations even for the high value of 10242 IB points. Additionally, the errors in these computations for Object 1 are similar for the RBF and Fourier models. For Object 2, the RBF model gives consistently better results for increasing data sites N_{d}.

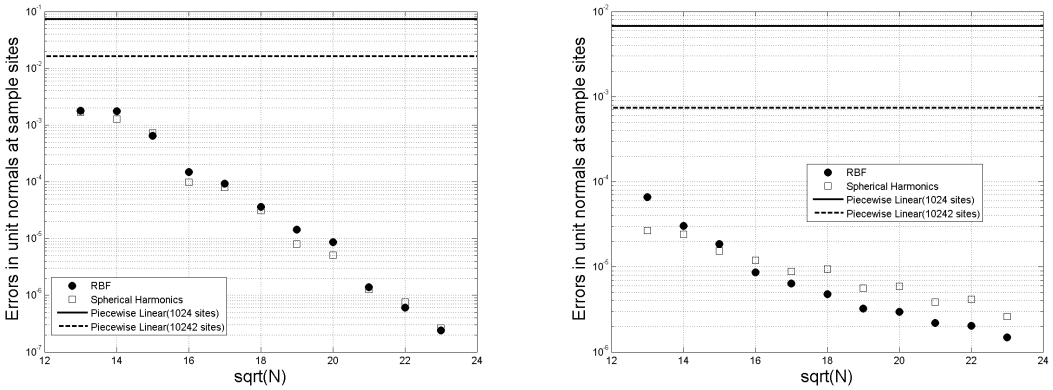

Figure 2.14: Errors in the approximations of the normal vectors to the 3D objects at $N_s = 1024$ sample sites as a function of the square root of the number of data sites N_d for Object 1 (left) and Object 2 (right). The solid line denotes the error for the piecewise linear model with 1024 IB points and the dashed line corresponds to the error with 10242 IB points. Circles denote the errors for the RBF model and squares denote the Fourier model. For the RBF model, $\varepsilon = 0.9$ for Object 1 and $\varepsilon = 1.5$ for Object 2. Data sites for the RBF model are the ME points, while the data sites for the Fourier model are the MD points.

We lastly focus on the errors in the computation of the forces in both parametric models. Figure 2.15 displays the errors in forces evaluated at 1024 sample sites as a function of the square root of the number of data sites N_d. In all experiments, both the coefficient of surface tension γ and the spring constant K_0 are set to 0.2. We see that the results are consistent with those from the shape reconstruction and normal vector approximations.

2.6.4 Comparison of the Computational Cost

We conclude the 3D results experiments by examining the computational cost associated with the three methods. As in the 2D experiments, we measure the computational cost as the elapsed wallclock time required to compute the interpolation coefficients, evaluate the interpolants, compute the normal vectors, and compute the forces. We precompute and store the LU decomposition of the spherical harmonic interpolation matrix Equation (2.11) and the Cholesky decomposition LL^T of the RBF interpolation matrix Equation (2.15). We also precompute matrices for evaluating the interpolants, the derivatives, and the force operator once the interpolation coefficients have been determined (see Section 2.4.2 for details). We do not account for these precomputations in our timing results. As in 2D, all computations were performed in MATLAB using the machine described in Section 2.5.4.

Since for the piecewise linear model the number of evaluation sites is the same as the number of data sites, the total computational cost includes only the time required to

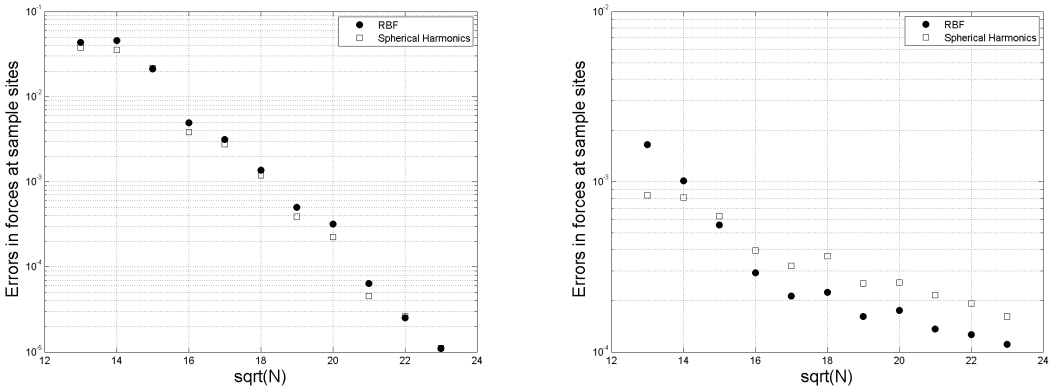

Figure 2.15: Errors in the approximations of the forces evaluated at $N_s = 1024$ sample sites as a function of the square root of the number of data sites N_d for Object 1 (left) and Object 2 (right). Circles denote the errors for the RBF model and squares denote the Fourier model. For the RBF model, $\varepsilon = 0.9$ for Object 1 and $\varepsilon = 1.5$ for Object 2. Data sites for the RBF model are the ME points, while the data sites for the Fourier model are the MD points.

compute the normal vectors and forces (see Section 2.4.1 for details), we do not include the time to compute the triangulation of the surface.

Figure 2.16 displays the elapsed time between the RBF, Fourier, and traditional piecewise linear models. The results for the RBF and Fourier models are displayed as a function of the number of data sites N_d for a fixed number of $N_s = 1024$ sample sites. Two results are presented for the piecewise linear model: one with 1024 IB points (solid) line and one with 10242 IB points (dashed line). We can see from the figure that the parametric models

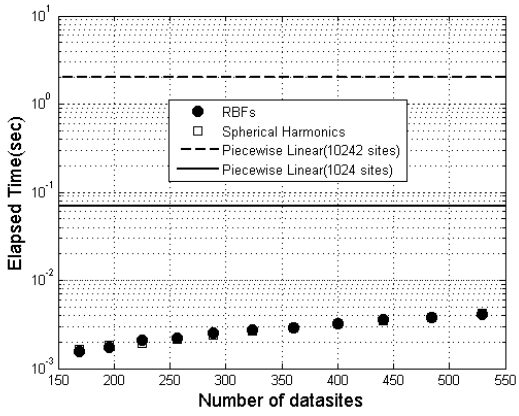

Figure 2.16: Elapsed wallclock time (in seconds) for one object to perform interpolation, evaluation, the computation of normal vectors, and the computation of forces at $N_s = 1024$ sample sites as a function of the number of data sites N_d. The piecewise linear computations were done with 1024 IB points (solid line) and 10242 IB points (dashed line) for comparison.

require significantly less time than the piecewise linear model, especially for the 10242 case. For $N_{\mathrm{d}} = 529$ data sites, the parametric models are over an order of magnitude faster than the piecewise linear model with 1024 IB points and nearly three orders of magnitude better with 10242 IB points.

2.7 Summary

One of the fundamental ingredients in platelet aggregation simulations with the IB method (and many others involving immersed structures) is how to model the platelets geometrically, so that internal structural forces can be computed at specified locations on the platelet surface. The current strategy is to use piecewise linear models for representing the platelets. In this work, we have presented two alternative geometric models for platelets: RBFs and Fourier-based methods. Both of these models are based on a parametric representation of the surface using polar coordinates in 2D and spherical coordinates in 3D. This choice of parameterization is motivated by the observed shape of platelets both during their inactive and active states. We have described how these new models can be used for constructing and maintaining the platelet's representation, computing the normal vectors to the platelet surface, and computing the internal structural forces. We have presented numerical comparisons between the traditional piecewise linear models and the new RBF and Fourier-based models in both 2D and 3D. Our findings indicate that both the RBF and Fourier methods provide viable alternatives to the traditional approach in terms of geometric modeling accuracy, force accuracy, and computational efficiency.

Although both the RBF and Fourier-based methods provided comparable results in terms of error characteristics and computational efficiency, we would advocate the use of the RBF-based models for the following reasons:

- they are easier to implement;
- they have accuracy similar to that of Fourier methods for smoothly-perturbed objects with similar computational costs;
- they are more accurate than Fourier methods for roughly perturbed objects with similar computational costs;
- they are more flexible than Fourier methods in terms of changing the underlying parameterizations of the objects (*e.g.*, changing to an elliptical parameterization rather than polar) [31].

One issue with the RBF models is how to choose an appropriate shape parameter. We will study this issue as part of our next step in applying the RBF-based models in an

IB simulation. This step will involve implementing the RBF-based models in a full IB simulation of platelet aggregation. The simulation will require projection of the forces from the sample points to the Eulerian mesh, computation of the Navier-Stokes system with forcing based upon the platelets, and then movement of the platelets via updating of the RBF data points. We will study how the shape parameter affects the simulations and compare the results of these simulations to those based on the traditional piecewise linear models for platelets. We will also attempt to understand the effects of the RBF-based models (if any) on the fluid solver used in the full IB simulation.

CHAPTER 3

THE RBF-IMMERSED BOUNDARY METHOD

3.1 Introduction

In Chapter 2 (and in previous work [72]), we explored alternative methods for the modeling of platelets, focusing on methods that decreased the computational time necessary to maintain and update platelet geometry and motion and had comparable or better error characteristics to the standard models. Specifically, we compared interpolation with Fourier-based techniques (trigonometric polynomials in 2D and spherical harmonics in 3D) and interpolation with radial basis functions (restricted to the unit circle in 2D and unit sphere in 3D). We found that interpolation with radial basis functions offered accuracy and computational cost comparable to that offered by Fourier-based methods (and better than that offered by standard finite-differences and piecewise-quadratic interpolation used in conjunction with traditional platelet simulations) in modeling a target shape, its normals, and tension forces computed on its surface when the target shape was infinitely smooth. Furthermore, interpolation with radial basis functions resulted in better accuracy than that offered by both Fourier-based methods and the standard combination of finite-differences and piecewise-quadratic interpolation used in many versions of the IB method (*e.g.*, [17, 21, 24]) when the target shape had only one or two underlying derivatives. In this situation, use of radial basis functions led to a computational cost comparable to that of Fourier-based methods and lower than that of the standard combination of techniques used in many IB methods.

We now turn our attention to exploring the consequences of using a parametric RBF geometric model within the full IB method, with platelet aggregation as our target application. In this work, we propose a new immersed boundary algorithm that utilizes the features afforded by our RBF geometric model. We will compare the convergence of the RBF-IB method to that of the traditional IB method on a fluid-structure interaction test used frequently in the IB literature. We also determine the extent of volume loss and the maximum time-step size allowed by each method. We then provide timing comparisons

between the two methods as a function of grid size and the number of platelets in a simulation. Our results show that the RBF-IB algorithm offers significant savings in terms of computational cost with greater accuracy than in the traditional IB method. We then present results from a 2D platelet aggregation simulation computed using the RBF-IB method.

3.2 Geometric Modeling of Platelets

In this section, we review the two geometric modeling strategies to be compared in the context of the Immersed Boundary method applied to platelet aggregation. For a full description of these strategies, see Chapter 2.

3.2.1 Piecewise Linear Model

In the traditional IB method, the surfaces of platelets are represented by a collection of Immersed Boundary points. We henceforth alternatively refer to the IB points in the traditional IB method as *sample sites*, and denote them by $\boldsymbol{X}_{\mathrm{s}}(t) = \boldsymbol{X}(q,t)$ for each discrete $q \in \Gamma$ and at a particular time t. The surface elastic forces of the platelets are spread from these sample sites into the neighboring fluid. Both tension and bending forces are computed using a collection of springs (typically linear springs) between pairs of sample sites. An explicit piecewise linear interpolant of the surface is not formed. If other information (such as normal vectors) is needed at the sample points, an approximation to the surface represented by the sample sites may be formed by a piecewise quadratic interpolation of the sample sites (*e.g.*, [93]). After the incompressible Navier Stokes equations are solved, velocities from the portions of the Eulerian grid surrounding the sample sites are interpolated to the sample sites using a discretization of Equation 1.4 and used to move the platelets.

3.2.2 Parametric RBF Model

Here, we briefly review the RBF model developed in Chapter 2, with the notation adapted for use with the IB method. Throughout this chapter, we use lowercase letters for Eulerian quantities and uppercase letters for Lagrangian quantities. We denote vectors with as many components as the spatial dimension in bold. We denote vectors with as many components as the number of data sites (N_{d}) or sample sites (N_{s}) by underlining. We indicate matrices with (N_{d}) or (N_{s}) rows and two columns in bold with underlining.

We use our model to define operators necessary for the computation of geometric and mechanical quantities required by the IB method.

We represent a platelet surface at any time t parametrically by

$$\boldsymbol{X}(\lambda, t) = (X(\lambda, t), Y(\lambda, t)) \tag{3.1}$$

where $0 \leq \lambda \leq 2\pi$ is the parametric variable and $\boldsymbol{X}(0,t) = \boldsymbol{X}(2\pi, t)$. We explicitly track a finite set of N_d points $\boldsymbol{X}_1^\mathrm{d}(t), \ldots, \boldsymbol{X}_2^\mathrm{d}(t)$, which we refer to as *data sites*. Here $\boldsymbol{X}_j^\mathrm{d}(t) := \boldsymbol{X}(\lambda_j, t)$, $j = 1, \ldots, N_\mathrm{d}$, and we refer to the parametric coordinates $\lambda_1, \ldots, \lambda_{N_\mathrm{d}}$ as the *data site nodes* (or simply *nodes*). We construct each component of $\boldsymbol{X}$ by using a smooth RBF interpolant of the data sites in parameter space as discussed in detail below. We also make use of derivatives of the interpolant at the data sites and we use the interpolant and its derivatives at another set of prescribed *sample points* or *sample sites*, which correspond to N_s parameter values: $\lambda_1^\mathrm{e}, ..., \lambda_{N_\mathrm{s}}^\mathrm{e}$.

We first explain how to construct an RBF interpolant to the X component of $\boldsymbol{X}$ using the data $(\lambda_1, X_1^\mathrm{d}(t)), ..., (\lambda_{N_\mathrm{d}}, X_{N_\mathrm{d}}^\mathrm{d}(t))$; the construction of the Y component follows in a similar manner. Let $\phi(r)$ be a scalar-valued radial kernel, whose choice we discuss below. Define $X(\lambda, t)$ by

$$X(\lambda, t) = \sum_{k=1}^{N_d} c_k^X \phi\left(\sqrt{2 - 2\cos(\lambda - \lambda_k)}\right). \tag{3.2}$$

Note that the square root term in Equation (3.2) is the Euclidean distance between the points on the unit circle whose angular coordinates are λ and λ_k. For this paper, we use the multiquadric (MQ) radial kernel function, given by

$$\text{MQ:} \quad \phi(r) = \sqrt{1 + (\varepsilon r)^2}, \tag{3.3}$$

where ε is called the shape parameter. To have $X(\lambda, t)$ interpolate the given data, we require that the coefficients $c_k^X, k = 1, ..., N_\mathrm{d}$ satisfy the following system of equations:

$$\underbrace{\begin{bmatrix} \phi(r_{1,1}) & \cdots & \phi(r_{1,N_\mathrm{d}}) \\ \phi(r_{2,1}) & \cdots & \phi(r_{2,N_\mathrm{d}}) \\ \vdots & \ddots & \vdots \\ \phi(r_{N_\mathrm{d},1}) & \cdots & \phi(r_{N_\mathrm{d},N_\mathrm{d}}) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1^X \\ c_2^X \\ \vdots \\ c_{N_\mathrm{d}}^X \end{bmatrix}}_{\vec{c}_\mathrm{d}^X} = \underbrace{\begin{bmatrix} X_1^\mathrm{d}(t) \\ X_2^\mathrm{d}(t) \\ \vdots \\ X_{N_\mathrm{d}}^\mathrm{d}(t) \end{bmatrix}}_{\vec{X}_\mathrm{d}(t)}, \tag{3.4}$$

where $r_{j,k} = \sqrt{2 - 2\cos(\lambda_j - \lambda_k)}$. Since $r_{j,k} = r_{k,j}$, the matrix A in this system is symmetric. More importantly, for the MQ kernels, A is nonsingular, with the global support and infinite smoothness of $\phi(r)$ lending itself to spectral accuracy and convergence on smooth problems [15, 88].

In our application, we want to be able to evaluate $X(\lambda, t)$ at sample sites corresponding to parameter values $\lambda_1^{\mathrm{e}}, ..., \lambda_{N_{\mathrm{s}}}^{\mathrm{e}}$, that stay fixed over time. While we could use Equation (3.2) to do this, it is much more convenient from a notational and computational perspective to construct an *evaluation matrix* that combines the linear operations of constructing the interpolant to $\vec{\boldsymbol{X}}_{\mathrm{d}}(t) = [\vec{X}_{\mathrm{d}}(t),\ \vec{Y}_{\mathrm{d}}(t)]$, for any t, and evaluating it at $\lambda_1^{\mathrm{e}}, ..., \lambda_{N_{\mathrm{s}}}^{\mathrm{e}}$. The evaluation matrix can be constructed by first noting that Equation (3.2) can be written as

$$X(\lambda, t) = \underbrace{\left[\phi\left(\sqrt{2 - 2\cos(\lambda - \lambda_1)}\right) \quad \cdots \quad \phi\left(\sqrt{2 - 2\cos(\lambda - \lambda_{N_{\mathrm{d}}})}\right)\right]}_{\vec{b}(\lambda)^T} \vec{c}_{\mathrm{d}}^{\,X}. \tag{3.5}$$

Since $\vec{c}_{\mathrm{d}}^{\,X} = A^{-1}\vec{X}_{\mathrm{d}}(t)$, we can write Equation (3.2) as $X(\lambda, t) = \vec{b}(\lambda)^T A^{-1}\vec{X}_{\mathrm{d}}(t)$. The evaluation of $X(\lambda, t)$ at $\lambda_1^{\mathrm{e}}, ..., \lambda_{N_{\mathrm{s}}}^{\mathrm{e}}$ can then be obtained as follows:

$$\underbrace{\begin{bmatrix} X(\lambda_1^{\mathrm{e}}, t) \\ \vdots \\ X(\lambda_{N_{\mathrm{s}}}^{\mathrm{e}}, t) \end{bmatrix}}_{\vec{X}_{\mathrm{s}}(t)} = \underbrace{\begin{bmatrix} \vec{b}(\lambda_1^{\mathrm{e}})^T \\ \vdots \\ \vec{b}(\lambda_{N_{\mathrm{s}}}^{\mathrm{e}})^T \end{bmatrix}}_{B} A^{-1}\vec{X}_{\mathrm{d}}(t) = \underbrace{BA^{-1}}_{\mathcal{E}_{\mathrm{s}}}\vec{X}_{\mathrm{d}}(t). \tag{3.6}$$

So, given the data sites $\vec{X}_{\mathrm{d}}(t)$ at any time t, we can interpolate their coordinates with an RBF expansion *and* evaluate the interpolant at the sample site nodes $\lambda_1^{\mathrm{e}}, ..., \lambda_{N_{\mathrm{s}}}^{\mathrm{e}}$ to get $\vec{X}_{\mathrm{s}}(t)$ by the matrix-vector product $\mathcal{E}_{\mathrm{s}}\vec{X}_{\mathrm{d}}(t)$. In fact, this same procedure can be used to give values at sample site nodes for any quantity whose values we have at data site nodes and which we represent using an RBF expansion (*e.g.*, $\vec{Y}_{\mathrm{d}}(t) = [Y_1^{\mathrm{d}}(t) \cdots Y_{N_{\mathrm{d}}}^{\mathrm{d}}(t)]^T$). Furthermore, the evaluation matrix $\mathcal{E}_{\mathrm{s}}$ can be precomputed once at $t = 0$ and used for all subsequent times.

We also need to compute geometric quantities such as tangent vectors, and mechanical quantities such as forces at data sites and/or sample sites. These quantities require computing derivatives with respect to λ of the platelet surface coordinates $(X(\lambda, t), Y(\lambda, t))$. We use the RBF-based representation of the surface to compute these derivatives, and we will express derivatives of the RBF interpolant in matrix-vector form. Toward this end, we use similar notation to Equation (3.5) and define the vector

$$\begin{aligned} \vec{b}_{\lambda}^{n}(\tilde{\lambda}) &:= \frac{\partial^n}{\partial\lambda^n}\vec{b}(\lambda)\bigg|_{\lambda=\tilde{\lambda}} \\ &= \left[\frac{\partial^n}{\partial\lambda^n}\phi\left(\sqrt{2 - 2\cos(\lambda - \lambda_1)}\right)\bigg|_{\lambda=\tilde{\lambda}} \quad \cdots \quad \frac{\partial^n}{\partial\lambda^n}\phi\left(\sqrt{2 - 2\cos(\lambda - \lambda_{N_{\mathrm{d}}})}\right)\bigg|_{\lambda=\tilde{\lambda}}\right]^T, \end{aligned}$$

for any $0 \leq \tilde{\lambda} \leq 2\pi$. Just as $\vec{b}(\tilde{\lambda})^T A^{-1} \vec{X}_{\rm d}(t)$ gives the value of $X(\tilde{\lambda}, t)$, we can use $\vec{b}^n_\lambda(\tilde{\lambda})$ to obtain the $n^{\rm th}$ derivative of $X(\lambda, t)$ with respect to λ as

$$\frac{\partial^n}{\partial \lambda^n} X(\lambda, t)\bigg|_{\lambda=\tilde{\lambda}} = \vec{b}^n_\lambda(\tilde{\lambda})^T A^{-1} \vec{X}_{\rm d}(t).$$

The evaluation of the $n^{\rm th}$ derivative of $X(\lambda, t)$ at the data site nodes $\lambda_1, \ldots, \lambda_{N_{\rm d}}$ can then be obtained as follows:

$$\begin{bmatrix} \frac{\partial^n}{\partial \lambda^n} X(\lambda, t)\Big|_{\lambda=\lambda_1} \\ \vdots \\ \frac{\partial^n}{\partial \lambda^n} X(\lambda, t)\Big|_{\lambda=\lambda_{N_{\rm d}}} \end{bmatrix} = \underbrace{\begin{bmatrix} \vec{b}^n_\lambda(\lambda_1)^T \\ \vdots \\ \vec{b}^n_\lambda(\lambda_{N_{\rm d}})^T \end{bmatrix}}_{B^n_\lambda} A^{-1} \vec{X}_{\rm d}(t) = \underbrace{B^n_\lambda A^{-1}}_{\mathcal{D}^n_\lambda} \vec{X}_{\rm d}(t). \tag{3.7}$$

In a similar manner, the evaluation of the $n^{\rm th}$ derivative of $X(\lambda, t)$ at the sample site nodes $\lambda^{\rm e}_1, \ldots, \lambda^{\rm e}_{N_{\rm d}}$ can be obtained by

$$\begin{bmatrix} \frac{\partial^n}{\partial \lambda^n} X(\lambda, t)\Big|_{\lambda=\lambda^{\rm e}_1} \\ \vdots \\ \frac{\partial^n}{\partial \lambda^n} X(\lambda, t)\Big|_{\lambda=\lambda^{\rm e}_{N_{\rm s}}} \end{bmatrix} = \underbrace{\begin{bmatrix} \vec{b}^n_\lambda(\lambda^{\rm e}_1)^T \\ \vdots \\ \vec{b}^n_\lambda(\lambda^{\rm e}_{N_{\rm s}})^T \end{bmatrix}}_{B^n_{\lambda^{\rm e}}} A^{-1} \vec{X}_{\rm d}(t) = \underbrace{B^n_{\lambda^{\rm e}} A^{-1}}_{\mathcal{D}^n_{\lambda^{\rm e}}} \vec{X}_{\rm d}(t). \tag{3.8}$$

For given data sites $\vec{X}_{\rm d}(t)$ at any time t, we can interpolate these values with an RBF expansion *and* evaluate the $n^{\rm th}$ derivative of the interpolant at the data site nodes by the matrix-vector product $\mathcal{D}^n_\lambda \vec{X}_{\rm d}(t)$ and at the sample site nodes by $\mathcal{D}^n_{\lambda^{\rm e}} \vec{X}_{\rm d}(t)$. We refer to the $N_{\rm d} \times N_{\rm d}$ matrices $\mathcal{D}^n_\lambda$ and the $N_{\rm s} \times N_{\rm d}$ matrices $\mathcal{D}^n_{\lambda^{\rm e}}$ as *RBF differentiation matrices*.

The matrices $\mathcal{D}^n_\lambda$ and $\mathcal{D}^n_{\lambda^{\rm e}}$ can be used to give values at respective data site or sample site nodes of the $n^{\rm th}$ derivative of the RBF interpolant of any quantity whose values we have at the data site nodes (*e.g.*, $\vec{Y}_{\rm d}(t) = [Y^{\rm d}_1(t) \cdots Y^{\rm d}_{N_{\rm d}}(t)]^T$). These matrices can also be precomputed once at $t = 0$ and used for all subsequent times.

Having defined the operators to compute derivatives of the RBF interpolant, we define the quantity

$$\boldsymbol{\tau} := \frac{\partial}{\partial \lambda} \boldsymbol{X}(\lambda, t) = \left(\frac{\partial}{\partial \lambda} X(\lambda, t), \frac{\partial}{\partial \lambda} Y(\lambda, t) \right) = (\tau_X, \tau_Y). \tag{3.9}$$

The unit tangent vector to $\boldsymbol{X}(\lambda, t)$ is then given by

$$\hat{\boldsymbol{\tau}} := \frac{\boldsymbol{\tau}}{\|\boldsymbol{\tau}\|} = (\hat{\tau}_X, \hat{\tau}_Y). \tag{3.10}$$

In our experiments, we assume that the Lagrangian force at a point on a platelet is the sum of a tension force, a bending-resistant force, and possibly a force due to a bond between

that point and a point on another platelet or the vessel wall. For the tension force, we use the fiber model defined in [67], according to which the elastic tension force density at $\boldsymbol{X}(\lambda_i, t_k)$ is given by

$$\mathbf{F}^{\mathrm{T}}(\lambda_i, t_k) = \frac{\partial}{\partial \lambda}(T\hat{\boldsymbol{\tau}})\Big|_{\lambda_i, t_k}, \tag{3.11}$$

where $T = k_{\mathrm{t}}(\|\boldsymbol{\tau}\| - l_0)$ is the fiber tension and $k_{\mathrm{t}} > 0$ is constant. We set $l_{0,i} = \|\boldsymbol{\tau}\|_{\lambda_i, t_0}$, where t_0 is the initial time of the simulation. For a bending-resistant force, we use a linear variant of the force defined in [42] and define the elastic force density at $\boldsymbol{X}(\lambda_i, t_k)$ due to how much the platelet surface there is bent to be

$$\mathbf{F}^{\mathrm{B}}(\lambda_i, t_k) = -\, k_{\mathrm{b}} \left(\frac{\partial^4 \boldsymbol{X}}{\partial \lambda^4} - \frac{\partial^4 \boldsymbol{X}^0}{\partial \lambda^4} \right)\Big|_{\lambda_i, t_k}. \tag{3.12}$$

Here $\boldsymbol{X}^0 = \boldsymbol{X}(\lambda_i, t_0)$ is the initial configuration of the platelet and $k_{\mathrm{b}} > 0$ is constant. Ideally, the constants k_{t} and k_{b} would be chosen to reflect values determined from experiments involving real platelets. In our work, we choose k_{t} and k_{b} that keep the platelets rigid, and scale them as we refine the background Eulerian grid.

We defer discussion of how we compute the forces given by Equations (3.11) and (3.12) to the next section, since the implementation is different for the RBF and piecewise-linear representations of the platelet boundary. However, the force acting on a platelet due to other platelets (and/or walls) is common to both methods. We use the spring force defined in [24]: let $p_1, p_2, ..., p_{N_p}$ be the indices corresponding to the platelets in the domain. Let p_1 and p_2 be the indices of two platelets which are linked at sample sites $\boldsymbol{X}_{p_1}(\lambda^s_{i_1})$ and $\boldsymbol{X}_{p_2}(\lambda^s_{i_2})$. The force at $\boldsymbol{X}_{p_1}(\lambda^s_{i_1})$ is then given by:

$$\mathbf{F}^{\mathrm{C}}_{p_1}(\lambda^s_{i_1}, t_k) = K_{\mathrm{C}}(||\boldsymbol{X}_{p_2}(\lambda^s_{i_2}) - \boldsymbol{X}_{p_1}(\lambda^s_{i_1})|| - l_{0,\mathrm{C}}) \frac{\boldsymbol{X}_{p_2}(\lambda^s_{i_2}) - \boldsymbol{X}_{p_1}(\lambda^s_{i_1})}{||\boldsymbol{X}_{p_2}(\lambda^s_{i_2}) - \boldsymbol{X}_{p_1}(\lambda^s_{i_1})||}, \tag{3.13}$$

where K_{C} and $l_{0,\mathrm{C}}$ are the interplatelet cohesion spring stiffness and the resting length, respectively; we also set $\mathbf{F}^{\mathrm{C}}_{p_2}(\lambda^s_{i_2}, t_k) = -\mathbf{F}^{\mathrm{C}}_{p_1}(\lambda^s_{i_1}, t_k)$. The formulation for platelet-wall links is similar.

3.3 Numerical Discretization

In this section, we present the implementation details for both IB methods. For each method, we briefly describe the spatial discretization for both the Lagrangian and Eulerian quantities. We then describe the time-stepping scheme for each method.

3.3.1 The Piecewise-Linear IB Method

Traditionally, finite-difference approximations of Equations (3.11) and (3.12) are used in conjunction with piecewise linear methods in 2D (*e.g.*, [42]). We use a second-order central difference involving triads of sample sites or IB points to discretize the derivatives involved in the computation of both the tension and bending forces (including tangent lengths). It is useful to think of these finite difference approximations to the constitutive model as Hookean springs connecting pairs of IB points. Note that these differences are only second-order assuming a near-uniform sampling. This is one of the sources of error for the IB method.

For the Eulerian spatial discretization, we use a second-order centered finite-difference approximation to the Laplacian on a staggered grid. We discretize the advection term using second-order centered differences, averaging quantities to cell edges or nodes as required. For the approximate δ-function, we use the "cosine" form described by Peskin [67] which ensures that the entire IB force is transmitted to the grid, that the force density on the grid is a continuous function of the IB point locations, and that the communication between grid and IB points is very localized. After each update of the IB point locations, new links are formed and existing ones are broken using the model's rules for these types of events. To prevent leakage, it is common to enforce a restriction that the IB point spacing on the surface be no greater than $0.5h$, where h is the Eulerian grid cell width.

We use the formally second-order Runge-Kutta time-stepping scheme outlined in [13]. This time-stepping scheme appears to show second-order convergence in time for a smooth forcing function, or for an elastic material that fills the whole domain, as demonstrated in [13]. However, as our results will show, this scheme will produce only first-order convergence in time in the presence of a sharp interface between the fluid and the elastic material, as is typical of IB methods. The full scheme is presented below:

1. Advance the structure to time level $t^{n+1/2}$ using the current velocity field on the grid $\boldsymbol{u}_g^n$. This is done by updating each IB point $\boldsymbol{X}_q$ (for each q) using the equation

$$\boldsymbol{X}_q^{n+1/2} = \boldsymbol{X}_q^n + \frac{\Delta t}{2}\boldsymbol{U}_q^n \equiv \boldsymbol{X}_q^n + \frac{\Delta t}{2}\sum_g \boldsymbol{u}_g^n \delta_h(\boldsymbol{x}_g - \boldsymbol{X}_q^n)h^2, \tag{3.14}$$

   where h is the fluid grid spacing, and δ_h is a discrete approximation to a two-dimensional δ-function. Here, $\boldsymbol{x}_g$ and $\boldsymbol{X}_q^{n+\frac{1}{2}}$ are the coordinates of grid point g and IB point q, respectively.
2. The resultant $\boldsymbol{F}_q^{n+\frac{1}{2}}$ of all of the force contributions that act on an IB point $\boldsymbol{X}_q^{n+\frac{1}{2}}$ is calculated for each q.

3. These forces are distributed to the Eulerian grid used for the fluid dynamics equations using a discrete version of Equation (1.3):

$$\boldsymbol{f}_g^{n+\frac{1}{2}} \equiv \boldsymbol{f}^{n+\frac{1}{2}}(\boldsymbol{x}_g) = \sum_q \boldsymbol{F}_q^{n+\frac{1}{2}} \delta_h(\boldsymbol{x}_g - \boldsymbol{X}_q^{n+\frac{1}{2}}) dq. \tag{3.15}$$

   Here, $\boldsymbol{F}_q^{n+\frac{1}{2}}$ is the Lagrangian force (per unit q) on the IB point, dq is the increment in parameter q between consecutive discrete sample sites, and δ_h is the same approximate δ-function as used in Equation 3.14.

4. With the fluid force density $\boldsymbol{f}_g^{n+\frac{1}{2}}$ now known at each grid point, the fluid velocity is updated taking a half step ($\Delta t/2$) with a discrete Navier-Stokes solver. As in [13], we use a fractional-step projection method. First, a backward Euler discretization of the momentum equations is used. The pressure that enforces discrete incompressibility is determined [44]. This gives us the velocity field $\boldsymbol{u}_g^{n+\frac{1}{2}}$, the mid-step approximation required in an RK2 method.

5. Using the mid-step fluid velocity $\boldsymbol{u}_g^{n+\frac{1}{2}}$ and the mid-step IB point positions $\boldsymbol{X}_q^{n+\frac{1}{2}}$, update the IB points $\boldsymbol{X}_q^n$ for each q to the time level t^{n+1} using

$$\boldsymbol{X}_q^{n+1} = \boldsymbol{X}_q^n + \Delta t \boldsymbol{U}_q^{n+\frac{1}{2}} \equiv \boldsymbol{X}_q^n + \Delta t \sum_g \boldsymbol{u}_g^{n+\frac{1}{2}} \delta_h(\boldsymbol{x}_g - \boldsymbol{X}_q^{n+\frac{1}{2}}) h^2, \tag{3.16}$$

   where δ_h is the same approximate δ-function we have used throughout.

6. Update the velocity $\boldsymbol{u}_g^n$ to time-level t^{n+1} using the mid-step velocity $\boldsymbol{u}_g^{n+\frac{1}{2}}$ and the force $\boldsymbol{f}_g^{n+\frac{1}{2}}$. The mid-step velocities are advected, while a Crank-Nicolson scheme is used for time-stepping the momentum equations. The pressure projection gives us the discretely-incompressible velocity field $\boldsymbol{u}_g^{n+1}$. Note that this step could have been performed as soon as $\boldsymbol{u}_g^{n+\frac{1}{2}}$ was computed. It is independent of step (5).

3.3.2 The RBF-IB Method

In order to construct the operators utilized by our algorithm, we must first choose an appropriate node set. We use N_d equally-spaced values on the interval $(0, 2\pi]$ as the data site node set $\{\lambda_k\}_{k=1}^{N_\mathrm{d}}$. This gives a uniform sampling in the parametric space. More importantly, since our target objects are either circular or elliptical in 2D, these node points correspond to a nearly uniform distribution of data sites on the object. We also use $N_\mathrm{s} >> N_\mathrm{d}$ equally-spaced points in the interval $(0, 2\pi]$ as the set of sample site nodes $\{\lambda_j^\mathrm{e}\}_{j=1}^{N_\mathrm{s}}$ since this results in a set of sample sites that are well distributed over the object. As in the piecewise-linear IB method, we make sure to start with enough points that the

sample site spacing is never greater than $0.5h$, though the data site spacing can be much greater. In the results section, we explore the ramifications of this choice.

We have formulated our operators to ensure that operations like evaluation of the interpolant and computing derivatives (and therefore the constitutive model) do not require solving a linear system for any time step of the platelet simulation except the initial step. This is possible because, though the data sites and sample sites move over the course of the simulation, their values in parameter space do not change. For the RBF model of the platelets, the evaluation matrix $\mathcal{E}_{\mathrm{s}}$ in Equation (3.6) and differentiation matrices $\mathcal{D}_{\lambda}^{n}$ and $\mathcal{D}_{\lambda^{\mathrm{e}}}^{n}$ in Equations (3.7) and (3.8), respectively, can be computed using the FFT as discussed in our previous work [72]. This is possible since the data site nodes $\{\lambda_k\}_{k=1}^{N_{\mathrm{d}}}$ are equally-spaced, which results in the A matrix defined Equation (3.4) having a circulant matrix structure. The costs and accuracy of the RBF models are elaborated upon in the discussion of the results. The algorithm to compute forces on platelets using these operators is presented below.

In the description of the algorithm below, we use standard matrix-vector operations such as multiplication as well as nonstandard operations like element-by-element multiplication of matrices and vectors (sometimes called the Hadamard product). We denote this latter operation with the $\circ$ operator. For example, if $\vec{\boldsymbol{J}}$ and $\vec{\boldsymbol{L}}$ are $N_{\mathrm{d}} \times 2$ matrices and $\vec{R}$ is a vector of length N_{d}, then the i^{th} row of $\vec{\boldsymbol{J}} \circ \vec{\boldsymbol{L}}$ and $\vec{R} \circ \vec{\boldsymbol{J}}$ are given by

$$
\begin{aligned}
(\vec{\boldsymbol{J}} \circ \vec{\boldsymbol{L}})_{i,1:2} &= [(\vec{\boldsymbol{J}})_{i,1}(\vec{\boldsymbol{L}})_{i,1},\ (\vec{\boldsymbol{J}})_{i,2}(\vec{\boldsymbol{L}})_{i,2}] \\
(\vec{R} \circ \vec{\boldsymbol{L}})_{i,1:2} &= [(\vec{R})_{i}(\vec{\boldsymbol{L}})_{i,1},\ (\vec{R})_{i}(\vec{\boldsymbol{L}})_{i,2}]
\end{aligned}
$$

where $i = 1, \ldots, N_{\mathrm{d}}$. We define $\vec{\boldsymbol{\tau}}_{\mathrm{d}} = \mathcal{D}_{\lambda}^{1}\vec{\boldsymbol{X}}_{\mathrm{d}}(t)$, the $N_{\mathrm{d}} \times 2$ matrix of tangent vectors at the data sites at time t and $\|\vec{\boldsymbol{\tau}}_{\mathrm{d}}\|$, the N_{d} vector containing the two-norm of each row of $\vec{\boldsymbol{\tau}}_{\mathrm{d}}$. The algorithm for computing platelet elasticity is as follows:

1. Initialization ($t = t_0$): After creating and storing the RBF evaluation matrix as in Equation (3.6) and differentiation matrices as in Equations (3.7) and (3.8), compute for each platelet:
   (a) The rest lengths for the tension force at the data sites: $\vec{l_0} = \vec{\boldsymbol{\tau}}_{\mathrm{d}} = \mathcal{D}_{\lambda}^{1}\vec{\boldsymbol{X}}_{\mathrm{d}}(t_0)$.
   (b) The bending-resistant force term for the platelet's initial configuration at the data sites, $\mathcal{D}_{\lambda^{\mathrm{e}}}^{4}\vec{\boldsymbol{X}}_{\mathrm{d}}(t_0)$.
2. For each time-step ($t = t_k$, $k \geq 1$), compute for each platelet:
   (a) The length of the tangent vectors $\vec{\boldsymbol{\tau}}_{\mathrm{d}} = \mathcal{D}_{\lambda}^{1}\vec{\boldsymbol{X}}_{\mathrm{d}}(t_k)$ at the data sites: $\|\vec{\boldsymbol{\tau}}_{\mathrm{d}}\|$; and the unit tangents at the data sites: $\hat{\vec{\boldsymbol{\tau}}}_{\mathrm{d}}$.

(b) The tension at the data sites, using the constitutive model: $\vec{T}_{\rm d} = k_{\rm t}(\|\vec{\boldsymbol{\tau}}_{\rm d}\| - \vec{l}_0)$.

(c) The tension force at sample sites: $\vec{\boldsymbol{F}}_{\rm s}^{\rm T} = \mathcal{D}^1_{\lambda^{\rm e}}\vec{\boldsymbol{Z}}_{\rm d}$, where $\vec{\boldsymbol{Z}}_{\rm d} = \vec{T}_{\rm d} \circ \vec{\hat{\boldsymbol{\tau}}}_{\rm d}$.

(d) The bending force at sample sites: $\vec{\boldsymbol{F}}_{\rm s}^{\rm B} = -k_{\rm b}\left(\mathcal{D}^4_{\lambda^{\rm e}}\vec{\boldsymbol{X}}_{\rm d}(t_k) - \mathcal{D}^4_{\lambda^{\rm e}}\vec{\boldsymbol{X}}_{\rm d}(t_0)\right)$.

(e) The interplatelet cohesion force from Equation (3.13) at the sample sites: $\vec{\boldsymbol{F}}_{\rm s}^{\rm C}$.

(f) The total Lagrangian force at the sample sites: $\vec{\boldsymbol{F}}_{\rm s} = \vec{\boldsymbol{F}}_{\rm s}^{\rm T} + \vec{\boldsymbol{F}}_{\rm s}^{\rm B} + \vec{\boldsymbol{F}}_{\rm s}^{\rm C}$.

The RBF-IB method uses the same time-stepping scheme and Eulerian discretization as the piecewise linear IB method, with one important difference. When computing the forces at time level $n+\frac{1}{2}$, we advance the *data sites* to time level $n+\frac{1}{2}$, generate a set of sample sites at that time level, and compute forces at the sample sites. Similarly, we use the mid-step approximation to the velocity field to advance the *data sites* to time level $n+1$. We thus generate only a single set of sample sites every time-step, since the sample sites are only needed when the data sites are advanced to time level $n+\frac{1}{2}$. It is clear that if the number of data sites is fewer than the number of sample sites, this results in improved computational efficiency over the piecewise linear IB method. However, it is important to explore the effect of our changes on the convergence of the algorithm. We explore these questions in the results section. We present the full algorithm below:

1. Advance the structure to time level $t^{n+1/2}$ using the current velocity field $\boldsymbol{u}^n$. This is done by updating the *data sites* $(\boldsymbol{X}_{\rm d})_j^n$ by a discrete analog of Equation (1.4)

$$(\boldsymbol{X}_{\rm d})_j^{n+\frac{1}{2}} = (\boldsymbol{X}_{\rm d})_j^n + \frac{\Delta t}{2}(\boldsymbol{U}_{\rm d})_j^n \equiv (\boldsymbol{X}_{\rm d})_j^n + \frac{\Delta t}{2}\sum_g \boldsymbol{u}_g^n \delta_h(\boldsymbol{x}_g - (\boldsymbol{X}_{\rm d})_j^n)h^2. \tag{3.17}$$

2. Generate a new set of sample sites $\vec{\boldsymbol{X}}_{\rm s}(t_{n+\frac{1}{2}})$ by applying the RBF evaluation operator to the data sites $\vec{\boldsymbol{X}}_{\rm d}^{n+\frac{1}{2}} := \vec{\boldsymbol{X}}_{\rm d}(t_{n+\frac{1}{2}})$, *i.e.*,

$$\vec{\boldsymbol{X}}_{\rm s}(t_{n+\frac{1}{2}}) = \mathcal{E}_{\rm s}\vec{\boldsymbol{X}}_{\rm d}(t_{\rm new}). \tag{3.18}$$

3. The total force at the sample sites $\vec{\boldsymbol{F}}_{\rm s}^{n+\frac{1}{2}}$ is calculated using the algorithm from above.
4. These forces are distributed to the Eulerian grid used for the fluid dynamics equations using a discrete version of Equation (1.3):

$$\boldsymbol{f}_g^{n+\frac{1}{2}} \equiv \boldsymbol{f}^{n+\frac{1}{2}}(\boldsymbol{x}_g) = \sum_q \boldsymbol{F}_q^{n+\frac{1}{2}}\delta_h(\boldsymbol{x}_g - (\boldsymbol{X}_{\rm s})_q^{n+\frac{1}{2}})dq. \tag{3.19}$$

Here, $\boldsymbol{x}_g$ and $(\boldsymbol{X}_{\rm s})_q^{n+\frac{1}{2}}$ are the coordinates of grid point g and sample site q, respectively, $\boldsymbol{F}_q^{n+\frac{1}{2}}$ is the Lagrangian force (per unit q) on the sample site, dq is the increment in parameter q between consecutive discrete sample sites, and δ_h is the same approximate δ-function as used in Equation 3.17.

5. With the fluid force density $\boldsymbol{f}_g^{n+\frac{1}{2}}$ now known at each grid point, the fluid velocity is updated taking a half step ($\Delta t/2$) with a discrete Navier-Stokes solver. Again, we use a fractional-step projection method, with a backward Euler discretization of the momentum equation, and a projection to determine the pressure that enforce incompressibility [44]. This gives us the velocity field $\boldsymbol{u}_g^{n+\frac{1}{2}}$, the mid-step approximation required in an RK2 method.
6. Using the mid-step fluid velocity and the mid-step *data site* positions $(\boldsymbol{X}_{\mathrm{d}})_j^{n+\frac{1}{2}}$, update the *data sites* $(\boldsymbol{X}_{\mathrm{d}})_j^{n+\frac{1}{2}}$ for each j to the time level t^{n+1} using

$$(\boldsymbol{X}_{\mathrm{d}})_j^{n+1} = (\boldsymbol{X}_{\mathrm{d}})_j^n + \Delta t(\boldsymbol{U}_{\mathrm{d}})_j^{n+\frac{1}{2}} \equiv (\boldsymbol{X}_{\mathrm{d}})_j^n + \Delta t \sum_g \boldsymbol{u}_g^{n+\frac{1}{2}} \delta_h(\boldsymbol{x}_g - (\boldsymbol{X}_{\mathrm{d}})_j^{n+\frac{1}{2}}) h^2, \quad (3.20)$$

   where δ_h is the same approximate δ-function we have used throughout.
7. Update the velocity $\boldsymbol{u}_g^n$ to time-level t^{n+1} using the mid-step velocity $\boldsymbol{u}_g^{n+\frac{1}{2}}$ and the force $\boldsymbol{f}_g^{n+\frac{1}{2}}$. The mid-step velocities are advected, while a Crank-Nicolson scheme is used for time-stepping the momentum equations. The pressure projection gives us the discretely-incompressible velocity field $\boldsymbol{u}_g^{n+1}$.

Observe that the data sites are updated twice per time-step in the RK2 scheme, but the sample sites are only generated once. Since the data sites are typically a fraction of the number of IB points from the PL-IB method, the computational cost is significantly lower for the RBF-IB method, even factoring in the interpolation and the sample site generation.

3.4 Results

In this section, we first compare the convergence of both methods on a canonical test problem. We also use this test problem to explore the relationship between the number of data sites (N_{d}) and the Eulerian grid spacing (h). We then compare the area loss in an elastic object simulated by each method on the same problem, and discuss the time-step sizes allowed by both methods. We follow with a discussion of the change in energy over time in the RBF-IB method. We then provide timings for platelet simulations and discuss both foreseen and unforeseen results of using the RBF model within the IB method. Finally, we present the results of platelet aggregation simulations conducted using the RBF-IB method.

3.4.1 Description of a Standard Fluid-structure Interaction Problem

We describe a standard fluid-structure interaction problem on which we test both versions of the IB method. This problem is commonly used in the IB literature (*e.g.*,

[64]). The problem involves placing an elliptical object with its center of mass at the center of the $[0,1]^2$ physical domain. The elliptical object has a circle of the same area as its rest configuration, and attempts to attain the rest configuration subject to a combination of tension and bending forces. The physical domain is filled with a fluid that is initially at rest, with periodic boundary conditions in the x-direction and no-slip Dirichlet boundary conditions in the y-direction. We set the radius of the target circle to be $r = 0.1$ units, with the ellipse initially having a major axis of $a = 2r$ and a minor axis of $b = 0.5r$. This test is visualized in Figure 3.1.

3.4.2 Convergence Studies

In previous work [72], we compared the accuracy and convergence of both RBF and traditional IB geometric modeling strategies for static platelet-like shapes. We now compare the accuracy and convergence of the full RBF-IB and PL-IB methods, both for the velocity field and for the immersed elastic structure.

For the fluid, on each grid with cell width h, we define the quantity $\boldsymbol{u}^{c,h}$, the coarsened discrete velocity field from the 256×256 grid. For each grid point i on a grid with cell width h, we compute the quantity $f_i^h = ||\boldsymbol{u}_i^h - \boldsymbol{u}_i^{c,h}||$. We define the l_2 error in the velocity field as $e_2(h) = \sqrt{\sum_i f_i^h h^2}$, and the l_∞ error in the velocity field to be $e_\infty(h) = \max_i f_i^h$.

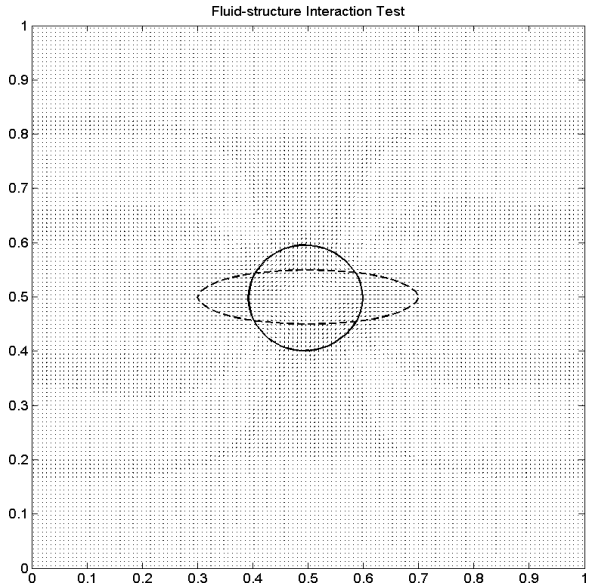

Figure 3.1: A visualization of the fluid-structure interaction test. The dashed lines show the initial ellipse, while the filled line indicates the near-circular object at the final time $t = 2.0$. The small arrows indicate the velocity field at the final time. The maximum velocity is very close to zero at this time as the object is almost at rest.

The convergence rate for errors $e(2h)$ and $e(h)$ is measured as $p_u = \log_2\left(\frac{e(2h)}{e(h)}\right)$.

For the Lagrangian markers (sample sites or IB points), we adopt the following procedure:

1. Given the number of sample sites N_s and the radius of the target circle r, we define $\lambda = \frac{2\pi}{N_s}$, the angle subtended at the center of the circle if the points were evenly-spaced.
2. We define the quantity $C = 2r\sin(0.5\lambda)$, the chord length between any two points in a set of evenly-spaced points on an ideal circle. We also define C_{exact} to be the ideal chord length for $N_s = 400$.
3. We compute the actual distances d_i between the sample sites (or IB points) for a simulation computed on the 256×256 grid with $N_s = 400$. We then compute the quantities $s^e_\infty = \max_i |d_i - C_{exact}|$ and $s^e_2 = (1/N_s)\sqrt{\sum_i |d_i - C_{exact}|}$.
4. We compute $s^{N_s}_\infty$ and $s^{N_s}_2$ for $N_s = 50, 100, 200$. We then define the l_2 error to be $e_2(N_s) = |s^{N_s}_2 - s^e_2|$ and the l_∞ error to be $e_\infty(N_s) = |s^{N_s}_\infty - s^e_\infty|$.

We define the convergence rate for errors $e(N_s)$ and $e(2N_s)$ to be $p_X = \log_2\left(\frac{e(N_s)}{e(2N_s)}\right)$.

3.4.2.1 Convergence of the Fluid Solver

Before testing both IB methods, we verify the convergence of our fluid solver. The RK2 time-stepping scheme used for the IB method in this paper is designed to give second-order convergence for smooth problems. To test this scheme, we specify a smooth initial condition to the Navier-Stokes equations on a $[0, 1]^2$ domain. The initial condition is a simple parabolic velocity profile. The maximum fluid velocity was set to $u_{max} = 5.0$, the fluid density to $\rho = 1.0$, and the nondimensionalized viscosity to $\mu = 8.0$. We also force the Navier-Stokes equations with a constant forcing function to give us a Poisseuille flow. We impose periodic boundary conditions in the x direction and no-slip boundary conditions in the y direction.

We then simulate on successfully finer grids and compute errors in the velocity field as outlined previously, using the solution computed on a 256×256 grid as our gold standard. We run the simulation to time $t = 1.0$. Our results (see Table 3.1) show that the fluid solver exhibits second-order convergence in both space and time for a smooth initial condition.

Table 3.1: Results of a refinement study on the fluid solver. Errors were measured against the solution computed on 256×256 grid with a time-step of $\Delta t = 6.25 \times 10^{-4}$.

Grid Size	Δt	L_2 error	Order	L_∞ error	Order
32×32	0.0050	6.4090e-03		6.4176e-03	
64×64	0.0025	1.5259e-03	2.07	1.5281e-03	2.07
128×128	0.00125	3.0518e-04	2.322	3.0572e-04	2.321

We note that for full Immersed Boundary simulations involving thin interfaces, it is unlikely that we will see second-order convergence.

3.4.2.2 Convergence of the PL-IB Method

We now test the convergence of the PL-IB method on the standard fluid-structure interaction problem. We compare the velocity field and IB point positions to those computed for the same test problem on a 256×256 grid with $N_s = 400$ IB points, and compute errors as outlined previously. It is important to note that the time-steps used for IB simulations with explicit time integrators are necessarily smaller than what we could use for a standard fluid solver. We simulate to final time $t = 2$. Our results are shown in Tables 3.2 and 3.3. For both the velocity field and the IB points, we see high convergence when refining from a 32×32 grid to a 64×64 grid, and closer to first-order convergence when refining from the 64×64 grid to the 128×128 grid. Indeed, as was mentioned earlier, one expects first-order convergence on this problem with the IB method. The high-order convergence seen with the first refinement is likely due to the solution being fairly inaccurate on the coarsest grid. For completeness, we note that $s_2^e = 5.0994e-07$ and $s_\infty^e = 4.3042e-05$. Below, we compare these quantities to the corresponding ones computed from the RBF-IB method.

Table 3.2: Results of a refinement study with the PL-IB method. We show the convergence of the velocity field, with errors measured against the velocity field of a simulation on a 256×256 grid with $N_s = 400$ sample sites and $\Delta t = 2.5 \times 10^{-5}$.

Grid Size	N_s	Δt	L_2 error	Order	L_∞ error	Order
32×32	50	2×10^{-4}	4.8302e-03		2.6640e-02	
64×64	100	1×10^{-4}	4.4352e-04	3.45	3.5463e-03	2.91
128×128	200	5×10^{-5}	1.1651e-04	1.93	1.6403e-03	1.11

Table 3.3: Results of a refinement study with the PL-IB method. We show the convergence in the IB point positions, with errors measured against the IB point positions from a simulation on a 256×256 grid with $N_s = 400$ IB points and $\Delta t = 2.5 \times 10^{-5}$.

N_s	Grid Size	Δt	L_2 error	Order	L_∞ error	Order
50	32×32	2×10^{-4}	3.1545e-05		5.9530e-04	
100	64×64	1×10^{-4}	4.4447e-06	6.15	8.7363e-05	2.76
200	128×128	5×10^{-5}	9.2942e-07	2.26	2.5866e-05	1.76

3.4.2.3 Convergence of the RBF-IB Method for $N_{\mathrm{d}} = 25$

Here, we test the convergence of the RBF-IB method on the fluid-structure interaction problem described above. Once again, we compare the velocity field and sample site positions to those computed for the same test problem on a 256×256 grid with $N_{\mathrm{s}} = 400$ sample sites and $N_{\mathrm{d}} = 100$ data sites. Tables 3.4 and 3.5 show the results obtained with $N_{\mathrm{d}} = 25$ data sites for the fluid and the object, respectively.

Table 3.4 shows that the errors produced in the RBF-IB method (for the same initial configuration and final time as in the PL-IB method) are lower for the same grid resolution. We see that the RBF-IB method for $N_{\mathrm{d}} = 25$ does not offer good convergence rates for the fluid velocity in any norm, despite producing lower errors on each grid level than the PL-IB method. Table 3.5 shows results for the errors on the object. The errors on the coarsest grid are high, leading to a higher-than-expected convergence rate in both norms when we measure the errors on a 64×64 grid. However, even on the coarsest grid level, the errors are already lower than those shown by the PL-IB method for the same number of sample sites and initial configuration. Further refinement gives us lower errors than the PL-IB method, but with lower convergence rates as well.

We note that the l_2 and l_∞ errors for the structure on the 256×256 grid for $N_{\mathrm{d}} = 25$ data sites are $s_2^e = 4.3694e-08$ and $s_\infty^e = 1.1113e-06$. These are each an order of magnitude

Table 3.4: Results of a refinement study with the RBF-IB method with $N_{\mathrm{d}} = 25$ data sites. We show the convergence of the velocity field, with errors measured against the velocity field of a simulation on a 256×256 grid with $N_{\mathrm{s}} = 400$ sample sites, $N_{\mathrm{d}} = 100$ data sites, and $\Delta t = 2.5 \times 10^{-5}$.

Grid Size	N_{s}	Δt	L_2 error	Order	L_∞ error	Order
32×32	50	2×10^{-4}	1.7666e-04		8.8682e-04	
64×64	100	1×10^{-4}	1.5097e-04	0.23	5.4255e-04	0.71
128×128	200	5×10^{-5}	8.4247e-05	0.84	3.0738e-04	0.82

Table 3.5: Results of a refinement study with the RBF-IB method with $N_{\mathrm{d}} = 25$ data sites. We show the convergence in the sample site positions, with errors measured against the sample site positions of a simulation on a 256×256 grid with $N_{\mathrm{s}} = 400$ sample sites, $N_{\mathrm{d}} = 100$ data sites, and $\Delta t = 2.5 \times 10^{-5}$.

N_{s}	Grid Size	Δt	L_2 error	Order	L_∞ error	Order
50	32×32	2×10^{-4}	3.1188e-06		2.3238e-05	
100	64×64	1×10^{-4}	3.5898e-07	3.12	3.0048e-06	2.95
200	128×128	5×10^{-5}	1.5310e-07	1.23	1.7905e-06	0.75

lower than those produced by the PL-IB method on the 256×256 grid.

3.4.2.4 Convergence of the RBF-IB Method for $N_d = 50$

We repeat the above test problem with $N_d = 50$ data sites. As before, we compare the velocity field and sample site positions to those computed for the same test problem on a 256×256 grid with $N_s = 400$ sample sites and $N_d = 100$ data sites. Tables 3.6 and 3.7 show the results for the fluid and the object, respectively.

Examining Table 3.6, we see high convergence when going from the coarsest grid to a finer grid. However, we see that when moving to the finest grid, we have recovered first-order convergence. The errors on the 128×128 grid for $N_d = 50$ with the RBF-IB method are close to those on the same grid with $N_d = 25$, and much lower than those produced by the PL-IB method in both norms. Table 3.7 shows errors similar to those seen in Table 3.5, albeit with less erratic convergence. Indeed, we recover first-order convergence in the l_2 norm and close to second-order convergence in the l_∞ norm. Of course, the errors are much lower than those seen in Table 3.3.

We note that using $N_d = 50$ data sites does not result in significantly better convergence on the structure than $N_d = 25$. There are two possible explanations. The first is that the function representing the shape of the object is of limited smoothness (as seen in our previous work [72]), with higher values of N_d causing the interpolation error to saturate

Table 3.6: Results of a refinement study with the RBF-IB method with $N_d = 50$ data sites. We show the convergence of the velocity field, with errors measured against the velocity field of a simulation on a 256×256 grid with $N_s = 400$ sample sites, $N_d = 100$ data sites, and $\Delta t = 2.5 \times 10^{-5}$.

Grid Size	N_s	Δt	L_2 error	Order	L_∞ error	Order
32×32	50	2×10^{-4}	5.5617e-03		3.7404e-02	
64×64	100	1×10^{-4}	2.2436e-04	4.63	1.2403e-03	4.91
128×128	200	5×10^{-5}	7.8934e-05	1.51	2.8859e-04	2.10

Table 3.7: Results of a refinement study with the RBF-IB method with $N_d = 50$ data sites. We show the convergence in the sample site positions, with errors measured against the sample site positions of a simulation on a 256×256 grid with $N_s = 400$ sample sites, $N_d = 100$ data sites, and $\Delta t = 2.5 \times 10^{-5}$.

N_s	Grid Size	Δt	L_2 error	Order	L_∞ error	Order
50	32×32	2×10^{-4}	3.6212e-05		5.2794e-04	
100	64×64	1×10^{-4}	3.8824e-07	6.54	7.8472e-06	6.07
200	128×128	5×10^{-5}	1.7236e-07	1.17	2.0636e-06	1.93

or even increase. The alternate (and more likely) explanation is that, since our RBFs are parametrized on the circle, $N_d = 25$ would already have a very high accuracy when the object becomes a circle, considering the spectral accuracy of RBF interpolation on the circle; in such a scenario, using $N_d = 50$ data sites would only serve to increase the rounding errors in the representation of the structure. However, it is clear that using $N_d = 50$ results in better convergence in the velocity field.

The l_2 and l_∞ errors for the structure on the 256×256 grid for $N_d = 50$ data sites are the same as those for $N_d = 25$ data sites. Once again, these are each an order of magnitude lower than those produced by the PL-IB method on the 256×256 grid.

3.4.2.5 Convergence of the RBF-IB Method for $N_d = 0.25N_s$

In the PL-IB method, the number of IB points depends on the grid spacing h. Specifically, the number of IB points is chosen so that the distance between any two sample sites is always less that $0.5h$. In all the tests above, we have maintained that relationship for the IB points in the PL-IB method and the sample sites in the RBF-IB method. In the RBF-IB method, we always use fewer data sites than sample sites, *i.e.*, $N_d < N_s$, with the choice of N_d being justified by the results in previous work [72]. Furthermore, in the tests above, we fix N_d even as we refine the fluid grid. For $N_d = 25$, this means that as we refine N_s the distance between data sites increases from $0.8h$ to $3.2h$ (at the start of the simulation).

In order to gain intuition on the relationship between N_d and h, we now perform a convergence study (using the same test problem given above) with increasing values of N_d as h is reduced. To accomplish this, we use values of $N_d = 12, 25, 50, 100$ for $N_s = 50, 100, 200, 400$, *i.e.*, we enforce $N_d = 0.25N_s$. We use the solution computed with $N_d = 100$ and $N_s = 400$ on a 256×256 grid as our gold standard, just as we have in all the other tests.

Table 3.8 shows the results for the fluid. Clearly, the errors are higher and the con-

Table 3.8: Results of a refinement study with the RBF-IB method with $N_d = 0.25N_s$ data sites. We show the convergence of the velocity field, with errors measured against the velocity field of a simulation on a 256×256 grid with $N_s = 400$ sample sites, $N_d = 100$ data sites, and $\Delta t = 2.5 \times 10^{-5}$.

Grid Size	N_s	Δt	L_2 error	Order	L_∞ error	Order
32×32	50	2×10^{-4}	4.7909e-04		2.4700e-03	
64×64	100	1×10^{-4}	1.5097e-04	1.67	5.4255e-04	2.19
128×128	200	5×10^{-5}	9.0802e-05	0.73	3.3020e-04	0.72

vergence more erratic than for the fixed $N_d = 50$ tests previously presented, but varying N_d certainly seems to give better convergence than fixing it to $N_d = 25$. However, the convergence in the structure is comparable, with lower errors being achieved than both $N_d = 25$ and $N_d = 50$ (and the PL-IB method as well). This can be seen in Table 3.9. The advantages of varying N_d with N_s are not clear. Using $N_d = 50$ yields the lowest errors in the fluid on the finest grid level, and reasonably low errors on the structure for all grid levels. Given the similarity of the errors achieved with $N_d = 50$ to those achieved with increasing N_d, we choose the simpler strategy of using a fixed value of $N_d = 50$ for our tests, though we present timings with $N_d = 25$ as well.

3.4.2.6 Effect of the Shape Parameter ε

In previous work [72], we found that the RBF shape parameter $\varepsilon > 0$ had to be selected carefully to achieve spectral accuracy in the representation of the elastic structure. In that work, we found that small values of ε were ideal for interpolating smooth target shapes and larger ones for rougher target shapes. In our tests, we found that the errors depended on ε even in the case of fluid-structure interaction, with smaller values of ε giving the lowest values of s_2^e and s_∞^e on the 256×256 grid. However, as we mentioned in our previous work, lower values of ε can make the RBF interpolation matrix more ill-conditioned. While methods (such as RBF-QR and RBF-RA) have been developed to overcome this poor conditioning [27], they are much more expensive than forming and inverting the standard RBF interpolation matrix. We thus choose a small value of $\varepsilon = 1.2$ for all our tests. When using $N_d = 100$, we use $\varepsilon = 2.0$. These are the smallest values we were able to pick without the interpolation matrix becoming ill-conditioned, a strategy consistent with the one used in our previous work [72].

Table 3.9: Results of a refinement study with the RBF-IB method. We show the convergence in the sample site positions, with errors measured against the sample site positions of a simulation on a 256×256 grid with $N_s = 400$ sample sites, $N_d = 100$ data sites, and $\Delta t = 2.5 \times 10^{-5}$.

N_s	Grid Size	Δt	L_2 error	Order	L_∞ error	Order
50	32×32	2×10^{-4}	9.7439e-06		7.0459e-05	
100	64×64	1×10^{-4}	3.5898e-07	4.76	3.0048e-06	4.55
200	128×128	5×10^{-5}	1.2694e-07	1.50	1.4362e-06	1.07

3.4.3 Area Loss and Time-step Size

In this section, we study the area loss in the RBF-IB method in a refinement study. We then explore the maximum stable time-step size afforded by each IB method.

The PL-IB method enforces an IB point separation distance of $0.5h$ in order to reduce area loss over the coarse of the simulation. In the RBF-IB method, while the sample site spacing is maintained at $0.5h$, we use a much coarser data site discretization, with the data site separation being almost $3.2h$ in some cases. In addition, we use the same strategy for interpolating velocities that we do in the PL-IB method, *i.e.*, we interpolate velocities to data sites from a 4×4 patch of fluid around each data site. While this can result in signifcant computational savings, it is important to explore the area loss in our discretization. We turn once again to our standard fluid-structure interaction problem. We run that simulation on successively finer grids until time $t = 2$. For both the RBF-IB method and the PL-IB method, we measure the initial area of the object for the same initial configuration of points. We then measure the area at time $t = 2$ and compute the percentage change in area.

In order to get an accurate estimate of the area in both methods, we fit an RBF interpolant to each object's Lagrangian markers (data sites for the RBF-IB method and a subset of the sample sites for the PL-IB method). We then sample that interpolant at a fixed number of points (400 points), and use the trapezoidal rule to compute the area. As was mentioned earlier, we ensure that the initial ellipse has the same area as the target circle by picking its radii to be $a = 2r$ and $b = 0.5r$, where $r = 0.1$ is the radius of the target circle. The exact area is then $\frac{\pi}{100}$. Our approach of sampling each object and computing the area with the trapezoidal rule gives an area estimate that agrees with this value up to 7 digits at $t = 0$. We record the results of our refinement study in Table 3.10. From the table, it is clear that the area loss for fixed $N_\mathrm{d} = 25, 50$ and $N_\mathrm{d} = 0.25N_\mathrm{s}$ are all close to each other. On the coarsest grid, it appears that smaller values of N_d result in

Table 3.10: Percentage area loss in the RBF-IB method as a function of grid size, the number of sample sites N_s, and the time-step Δt. The PL-IB method gives area losses similar to the $N_\mathrm{d} = 50$ case, except on the coarsest grid, where the percentage area loss is three times that of the RBF-IB method.

N_s	Grid Size	Δt	Percentage area loss		
			$N_\mathrm{d} = 25$	$N_\mathrm{d} = 50$	$N_\mathrm{d} = 0.25N_\mathrm{s}$
50	32×32	2×10^{-4}	0.0680	0.3081	0.0450
100	64×64	1×10^{-4}	0.0047	0.0049	0.0047
200	128×128	5×10^{-5}	0.0023	0.0025	0.0025
400	256×256	2.5×10^{-5}	0.0015	0.0015	0.0015

lower area loss. The area losses for $N_{\mathrm{d}} = 50$ match with those given by the PL-IB method (results not shown), except in the case of the coarsest grid, where the PL-IB method gives almost a 1% area loss. The convergence is initial second-order but quickly saturates. This saturation is likely due to two sources of error: the first is the interpolation of velocities to the Lagrangian markers, which does not preserve the divergence-free nature of the fluid velocity; the second is the fact that the time-integration itself is not specifically designed to preserve area. Nevertheless, it is clear from this study that the RBF-IB method produces similar area losses to the PL-IB method despite using a smaller number of Lagrangian markers to move the structure through the fluid.

Another measure of interest is the maximum stable time-step size afforded by each method. We measure this by increasing the time-step size in small increments and observing the forces produced on the structure in the fluid-structure interaction test. Using a time-step that is too large can result in the forces blowing up and the simulation halting. We immediately note that the PL-IB method allows a maximum time-step size of $\Delta t = 2 \times 10^{-4}$ on the 32×32 grid when $N_{\mathrm{s}} = 50$ IB points are used, and a maximum time-step size of $\Delta t = 10^{-4}$ on the 64×64 grid when $N_{\mathrm{s}} = 100$ IB points are used. We use these values of Δt as the starting point when testing for the time-step sizes allowed by the RBF-IB method, and increase the value of Δt in increments of 10^{-4}. We found that on the 32×32 grid, the RBF-IB method allows us to take time-steps that are $3\times$ larger than the time-steps allowed by the traditional IB method; on the 64×64 grid, the RBF-IB method can use time-steps that are $1.5\times$ larger than the time-steps allowed by the traditional IB method. This pattern holds both when $N_{\mathrm{d}} = 25$ and $N_{\mathrm{d}} = 50$ data sites are used.

In simulations involving platelet-like shapes (ellipses that attempt to maintain their elliptical configuration), we found that the RBF-IB method allows time-step sizes that are $6\times$ larger than those allowed by the PL-IB method on a 32×32 grid, and $3\times$ larger than those allowed by the PL-IB method on a 64×64 grid. This is likely due to the fact that platelet simulations involve smaller deformations than those seen in the standard fluid-structure interaction test.

3.4.4 Energy Estimates

In this section, we compute energy estimates for the RBF-IB method in the context of our standard fluid-structure interaction problem. We run our simulation out to time $t = 2.0$ on two grid sizes, 32×32 and 64×64, with time-step sizes $\Delta t = 2 \times 10^{-4}$ and $\Delta t = 10^{-4}$ respectively. We use $N_{\mathrm{d}} = 50$ data sites.

In this test, one expects the changes in energy to be mainly due to the deformation of

the stiff elastic object. Eventually, the energy of the system must damp out as the elastic object reaches its target configuration. We compute the change in energy to demonstrate that the energy is bounded within the RBF-IB simulation. The energy change in a time-step is computed as the sum of the difference in kinetic energy of the fluid over the time-step and the change in potential energy of the elastic object. This can be written as

$$E = \int_{fluid} \rho \boldsymbol{u}^{n+1} \cdot \boldsymbol{u}^{n+1} - \int_{fluid} \rho \boldsymbol{u}^{n} \cdot \boldsymbol{u}^{n} + \Delta t \int_{\boldsymbol{X}} \boldsymbol{F} \cdot \frac{\partial \boldsymbol{X}}{\partial t} \tag{3.21}$$

Here, the Lagrangian force $\boldsymbol{F}$ is computed at time level $n+1$ at the sample sites. The $\frac{\partial \boldsymbol{X}}{\partial t}$ term is computed by applying the evaluation matrix $\mathcal{E}_{\mathrm{s}}$ to the velocities obtained at the data sites. This gives us sample site velocities, allowing us to compute dot products with the $\boldsymbol{F}$ terms. We compute the discrete analog of Equation (3.21) for the fluid-structure interaction problem.

The results of this test are shown in Figure 3.2. Both plots show the change in energy of the system for the fluid-structure interaction problem on a 32×32 grid (left) and a 64×64 grid (right). Here, the fluid starts off stationary, so the initial kinetic energy is zero. However, the elliptical elastic object starts off under tension, since its target configuration is a circle. This means that the initial elastic potential energy of the system is high (though negative by convention). As the elastic object attempts to minimize its elastic potential

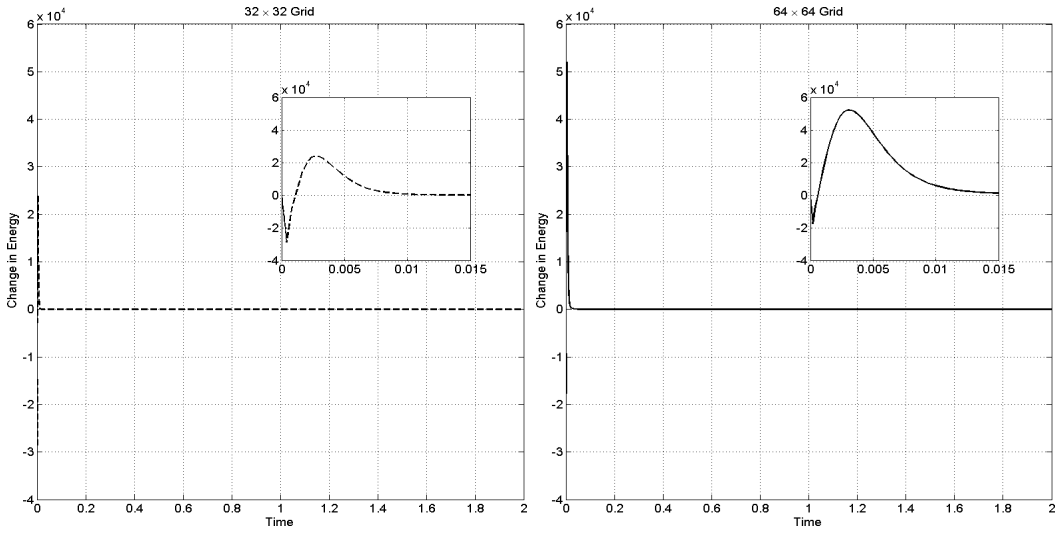

Figure 3.2: Change in energy as a function of time in the RBF-IB method. The figure on the top left shows the change in energy over a time-step as a function of time for the standard fluid-structure interaction test on a 32×32 grid. The figure on the top right shows the same quantity on a 64×64 grid. We use $N_{\mathrm{d}} = 50$ data sites for both grid sizes. The inset plots show the initial spikes corresponding to the change from an ellipse to a circle which are difficult to see in the main plots.

energy, its deformation drives a change in the kinetic energy of the fluid, causing the kinetic energy of the fluid to increase from its initial value of zero to some maximum. However, the elastic object soon attains something close to its reference configuration, causing the kinetic energy of the fluid to drop sharply. The spikes in both the left and right sides of Figure 3.2 correspond to that rise and fall in kinetic energy and the trending of the potential energy to zero on both grids, and can be seen more clearly in the inset plots. The viscosity of the fluid causes the kinetic energy to eventually damp out almost completely, with minor perturbations due to possible deformations of the elastic object. The energy of the system continues to decrease as the object becomes more and more circular. In fact, our estimates show that the change in energy is negative, indicating that our method is dissipative. The results are similar for $N_{\mathrm{d}} = 25$ (not shown), though using more data sites appears to make our method less dissipative on this particular test problem.

3.4.5 Timings for Platelet Simulations

We now present timings of simulations involving platelet-like shapes. The setup here is different from the standard fluid-structure interaction test. We place ellipses ($r = 0.05$, $a = 2r$, $b = 0.5r$) at the left end of a $[0, 2] \times [0, 1]$ domain that resembles a channel. These ellipses represent platelets, and they attempt to maintain their elliptical shapes, *i.e.*, their configuration at time $t = 0$ is their preferred configuration. We apply a background force that would result in parabolic velocity field in the absence of the platelets, with a density $\rho = 1.0$, and a nondimensionalized viscosity of $\mu = 8.0$. The field has a maximum velocity value of $u_{max} = 5.0$, with no-slip boundary conditions on the top and bottom of the domain and periodic boundary conditions at the left and right ends. A platelet is removed from the domain if its center of mass crosses the location $x = 1.9$.

Figure 3.3 shows timings for three grid sizes for each method as a function of the number of platelets (N_{p}) being simulated. The number of sample sites was fixed at $N_{\mathrm{s}} = 100$ for both methods and the number of data sites for the RBF-IB method was set to $N_{\mathrm{d}} = 25$ for one set of tests and then to $N_{\mathrm{d}} = 50$ for the next set. We plot the average time per time-step as a function of the number of platelets; this was computed by running simulations on each grid for 10^5 time-steps, and dividing the total wall clock time by the number of time-steps. We average this over three runs of each simulation.

While the cost of platelet operations always increases as we increase the number of platelets, the increase in cost is slower for the RBF-IB method due to the RBF representation. For example, for $N_{\mathrm{p}} = 60$, the PL-IB method directly spreads forces from, interpolates to, and moves a total of 6000 IB points (twice per time-step due to the RK2

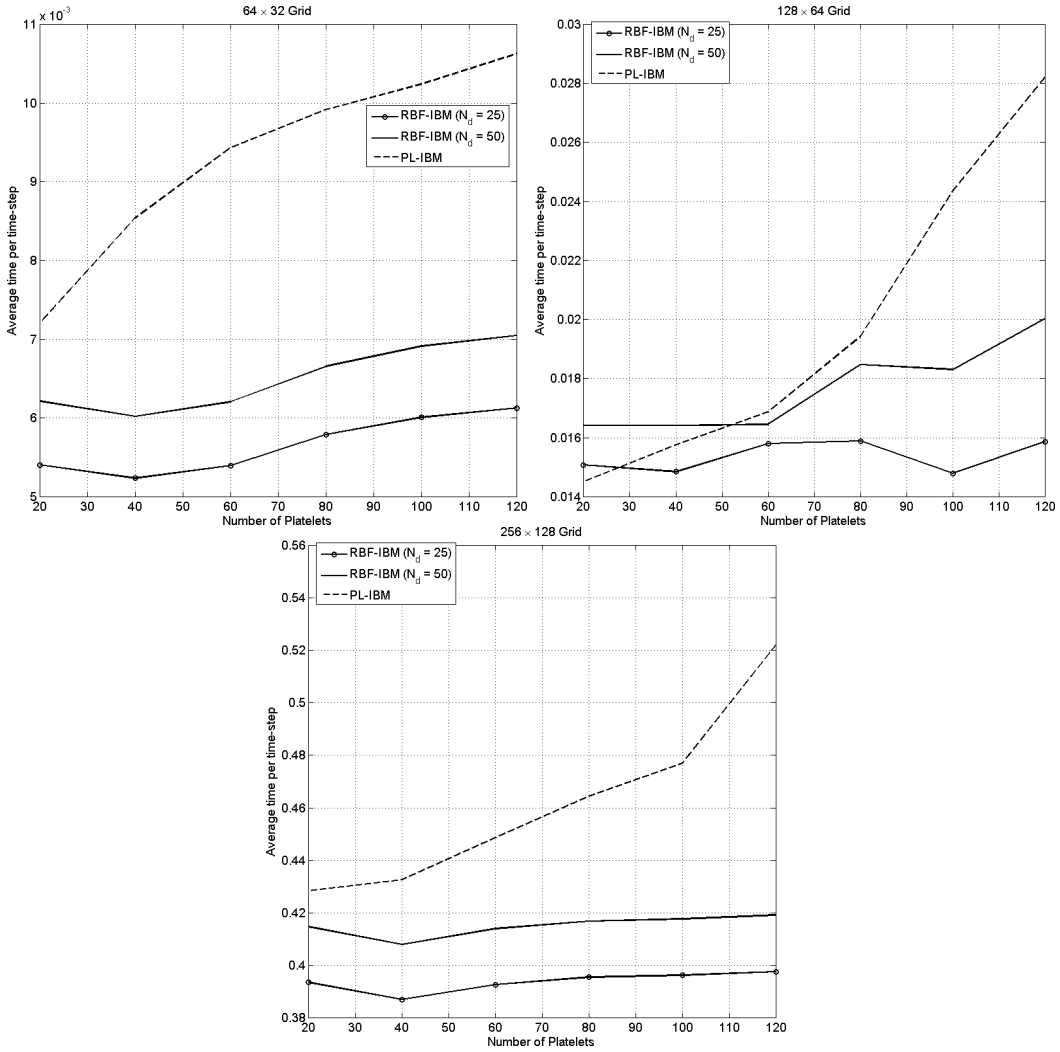

Figure 3.3: Average time per time-step for 10^5 time-steps of each simulation method as a function of the number of platelets. In the first row, the figure on the left shows timings on a 64×32 grid and the figure on the right on a 128×64 grid. The figure below shows timings on a 256×128 grid. The time-step was set to $\Delta t = 10^{-4}$ for the figures on the top row, and was set to $\Delta t = 10^{-5}$ for the figure on the bottom.

scheme) while the RBF-IB method with $N_{\mathrm{d}} = 25$ data sites computes forces at 6000 points, and interpolates velocities to (and moves) only 1620 points twice per time-step. If the number of platelets is doubled to $N_{\mathrm{p}} = 120$, the PL-IB method now computes forces at, spreads from, interpolates to, and moves 12000 points twice per time-step, whereas the RBF-IB method computes forces at 12000 points, but interpolates velocities to and moves only 3240 points twice per time-step. The cost of the RBF-IB method shows better than linear scaling with respect to the number of platelets on all the tested grid sizes for these reasons. Furthermore, it is clear that there is not much of a difference in computational

cost between using $N_d = 25$ data sites and $N_d = 50$ data sites. It is clear that the RBF-IB method is more computationally efficient than the PL-IB method.

3.4.5.1 Effect of the RBF Representation on the Fluid Solver

In previous work [72], we showed that using an RBF interpolant for geometric modeling is more computationally efficient (for a given accuracy) than using piecewise quadratics and finite differences. However, that benefit alone does not explain the computational efficiency of the RBF-IB method over the PL-IB method that we see in Figure 3.3.

To fully understand the speedup seen with the RBF-IB method, it is important to understand how the costs are distributed between the different operations (platelet operations and fluid solves) in both IB methods. We show the results for $N_p = 60$ platelets in Table 3.11. Clearly, as h is reduced, both IB codes spend more time in the fluid solver than on platelet operations. However, the RBF-IB method clearly spends less time in the fluid solver than the PL-IB method does as we refine the background Eulerian grid.

Indeed, this unexpected result is what gives the RBF-IB method an edge even when the cost of the fluid solver dominates the cost of platelet operations. We hypothesize that this may be caused by the RBF representation producing smoother Lagrangian forces than the finite difference model used in the PL-IB method. Our experiments show that the RBF-IB code needs fewer iterations in the linear solver used in the pressure projection– anywhere from $10 - 30\%$ fewer than the identical fluid solver used in the PL-IB method, depending on the time-step size and the grid resolution, with larger savings on finer grids and smaller time-step sizes.

Table 3.11: Percentage of time per time-step spent in fluid solver as a function of grid size by both methods for $N_p = 60$ platelets. The percentages for the RBF-IB method are the same for both $N_d = 25$ and $N_d = 50$ data sites, with the total time for the latter being larger. All results use $N_s = 100$ sample sites (or IB points in the PL-IB method) per platelet.

Grid Size	Percentage time in fluid solver	
	RBF-IB method	PL-IB method
32×32	33.7	32.1
64×64	56.2	58.0
128×128	65.4	79.3

3.4.6 Platelet Aggregation

We now present the results of a platelet aggregation simulation with the RBF-IB simulation. We used the same boundary conditions, domain size, fluid properties, and Poisseuille flow as in the previous subsection, but allow platelets to form links with other platelets and a portion of the chamber wall ($x = 0.4$ to $x = 0.7$) at the sample sites ($N_s = 100$ per platelet). We used $N_d = 50$ data sites per platelet, making the data sites a subset of the sample sites for convenience of visualization, and then visualize the data sites and the links between sample sites. We allowed each platelet to form up to 10 links in total, either with the wall or with a neighbor; we allow links to cross each other for the purpose of simplicity, though this is usually prohibited in a platelet simulation. The simulation was run on a 128×64 grid with a time-step of $\Delta t = 10^{-4}$.

Each platelet is initially an ellipse with radii $a = 0.06$ and $b = 0.015$. We initialize the platelets so that their centers of masses are at locations $(0.5, 0.02)$, $(0.64, 0.02)$, $(0.78, 0.02)$, $(0.55, 0.07)$, $(0.68, 0.07)$, $(0.4, 0.045)$, $(0.23, 0.045)$, and $(0.65, 0.14)$. We chose these locations to ensure that three platelets lay on the wall, with three close enough to bind to the three bound to the wall, and two slightly further away. Each platelet attempts to maintain its initial elliptical shape. We then started the simulation and ran it to time $t = 2.4$. The results are shown in Figure 3.4. The figure shows both the velocity field and the platelet aggregate for a portion of $[0, 2] \times [0, 1]$ domain, the data sites on each platelet and the links

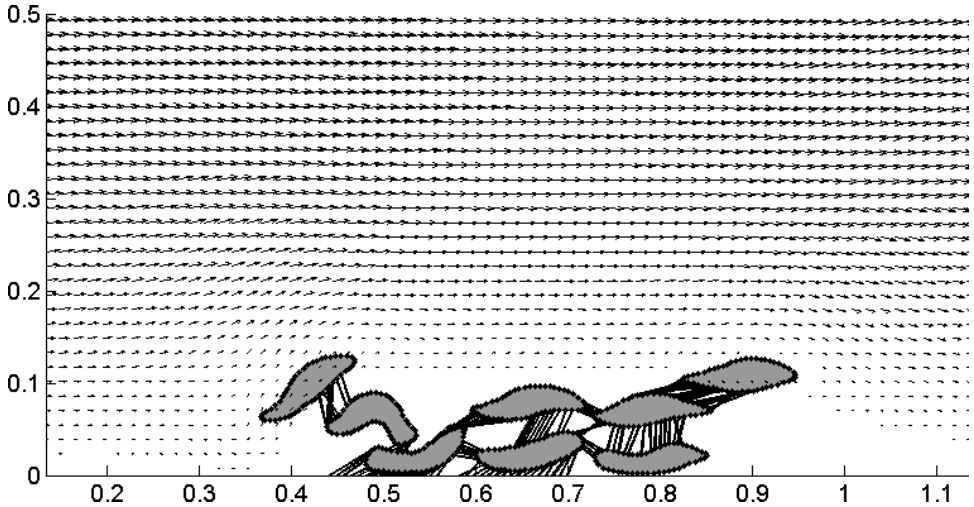

Figure 3.4: Results of a platelet aggregation simulations with the RBF-IB method. The figure shows a snapshot of a platelet aggregation simulation achieved with the RBF-IB method with a time-step of $\Delta t = 10^{-4}$. The snapshot was taken at simulation time $t = 2.4$. The simulation was run on a 128×64 grid on a $[0, 2] \times [0, 1]$ domain. The arrows show the magnitude and direction of the velocity field.

between the sample sites corresponding to those particular data sites on the platelet.

There are two interesting features in Figure 3.4. The first is that the fluid flow gets diverted around the growing aggregate, a consequence of the size of the aggregate and the dynamics of the problem that mimics what one would hope to see in a realistic platelet aggregation simulation. The second feature is that some platelets are quite deformed, *e.g.*, the platelet with center of mass approximately at $(0.5, 0.02)$, or its neighbor above and to its left. This is a consequence (and function) of the stiffness of each platelet, the shear rate of the flow, and the number of links we allow each platelet to form. Higher platelet stiffness, lower shear rates, and/or fewer (or weaker) links would lead to less deformation. The breaking model for interplatelet and platelet-wall links can also affect the mechanics of aggregation. We note that our RBF model did not run into any instabilities in this simulation even when we ran it out to a time at which all the platelets (except the three closest to the wall) had left the domain.

3.5 Summary

In this chapter, we explored the ramifications of using the RBF geometric model developed in Chapter 2 within the IB method, and compared the behavior of this new RBF-IB method against that of the traditional IB method. We discussed the issue of selecting an appropriate shape parameter for the RBF-IB method. We then presented a series of convergence studies for measuring errors and convergence both in the velocity field and in the representation of the immersed elastic structure. We went on to compare the computational costs incurred by both methods in the context of platelet simulations. We then compared the area conservation properties of both methods and also the time-step restrictions on both. We also remarked on the energy properties of our method.

We conclude the following:

- The RBF-IB method facilitates the use of constitutive models within the IB method without having to resort to lower-order approximations;
- The RBF-IB method (especially with $N_{\mathrm{d}} = 50$) produces lower errors in both the velocity field and the immersed elastic structure in convergence studies;
- The RBF-IB is more computationally efficient than the traditional IB method for both $N_{\mathrm{d}} = 25$ and $N_{\mathrm{d}} = 50$, both due to the utilization of a small number of data sites and due to smoother forces being spread into the fluid resulting in a faster convergence from the fluid solver;

- The RBF-IB allows for larger time-step sizes than those allowed in the traditional IB method for a given grid size.

In previous work [64], a sufficient condition for unconditional stability of an implicit IB method was established. The proof relied on the assumption that the set of points from which IB forces are spread is the same as that to which grid velocities are interpolated to update IB point positions. The RBF-IB method does not meet that condition, and it remains to be seen how this would impact an implict version of our method. Finally, an issue with the RBF-IB method is that it is dependent on the parametrization of the immersed elastic objects. For objects that are not easily parameterized in terms of circles and ellipses, the use of the RBF model as presented in our work (wherein the RBFs are restricted to the circle) may not be ideal. In the future, we thus hope to explore the use of RBFs in a meshfree variational form within the IB method so as to be able to easily evaluate constitutive models on arbitrary shapes.

CHAPTER 4

RBF-FD ON 1D SURFACES WITHIN THE AFM

4.1 Introduction

Thus far in this dissertation, we have presented an RBF geometric model for platelet modeling, and tested the use of this model within the IB method. We now turn our focus away from the mechanics of platelet aggregation, and toward the chemistry involved in the process.

In this chapter, we present a new numerical methodology for the simulation of reaction-diffusion equations on 2D stationary platelets that are suspended in blood and for simulating diffusion of chemical species in the blood based on boundary conditions derived from those reaction-diffusion equations. This methodology consists of two components: a new method to solve reaction-diffusion equations on curves (1D surfaces) using radial basis function generated finite differences (RBF-FD), and a version of the Augmented Forcing method (AFM) for the simulation of the fluid-phase diffusion equations modified with symmetric Hermite RBF interpolation to enforce boundary conditions. This is a documentation of previously published work [73]. We also utilize the parametric RBF model from Chapter 2. Since the accuracy of normals can affect the enforcement of boundary conditions in the AFM, our RBF geometric model for platelets is an important part of our methodology.

4.2 Problem Statement

Consider a two-dimensional region Ω consisting of those points $\boldsymbol{x} = (x, y)$ within the rectangle Ω_0 that are external to all of the nonoverlapping subregions $\Omega_i,\ i = 1, ..., N_o$ (see Figure 4.1). Let Γ_i denote the boundary of Ω_i.

Let $c(\boldsymbol{x}, t) = c(x, y, t)$ be a concentration field defined for $\boldsymbol{x} \in \Omega$ and $t \geq 0$. For each i, let $C_i^{\mathrm{b}}(\boldsymbol{X}, t)$ and $C_i^{\mathrm{u}}(\boldsymbol{X}, t)$ be chemical density fields defined for each point $\boldsymbol{X} = (X, Y) \in \Gamma_i$ and $t \geq 0$, representing the surface densities of occupied and unoccupied binding sites respectively. We will describe these quantities in greater detail below.

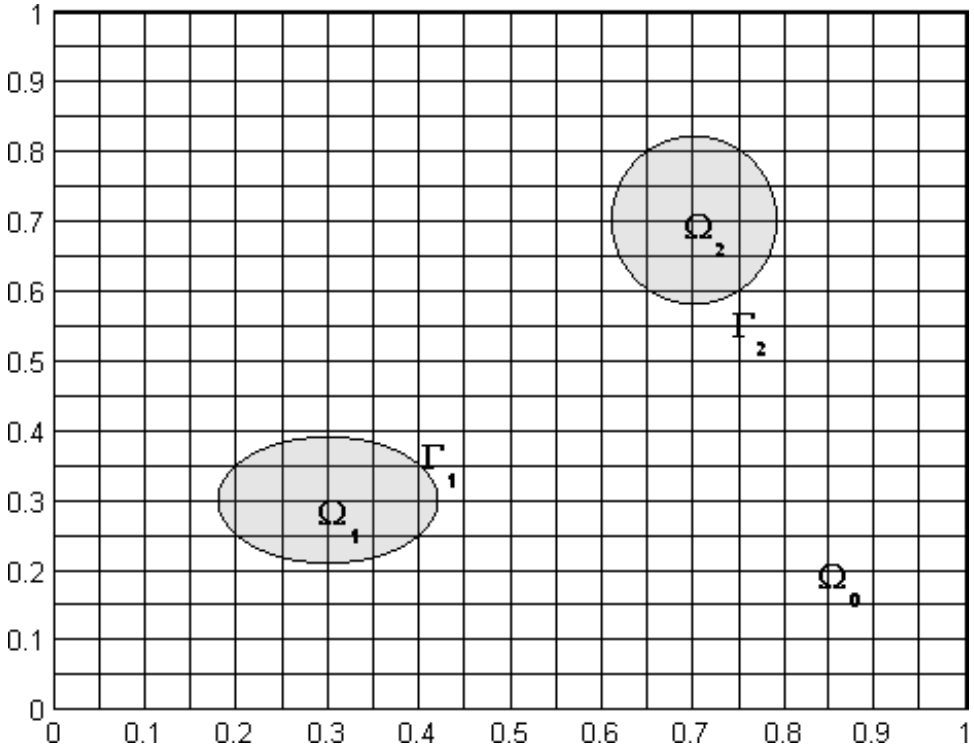

Figure 4.1: Illustration of a rectangular domain and grid with irregular objects embedded.

We assume that $c(\boldsymbol{x}, t)$ satisfies the inhomogeneous diffusion equation

$$\frac{\partial c}{\partial t} = D\Delta c + s, \tag{4.1}$$

at each point $\boldsymbol{x} \in \Omega$, and that it satisfies the boundary condition

$$-D\frac{\partial c}{\partial \hat{\boldsymbol{\eta}}} = k_{on} C_i^{\mathrm{u}} c - k_{off} C_i^{\mathrm{b}}, \tag{4.2}$$

at each point $\boldsymbol{X} \in \Gamma_i$. In these equations, s is a specfied source term, D is a diffusion coefficient, k_{on} and k_{off} are second-order and first-order rate constants, respectively, and $\hat{\boldsymbol{\eta}}$ is the unit normal to Γ_i pointing into the domain Ω. Initial data for c is given at all points of Ω.

For the surface densities C_i^{b} and C_i^{u}, we consider two variants of a reaction-diffusion model. For model 1, we imagine that the density of binding sites $C_i^{\mathrm{tot}}(\boldsymbol{X})$ is a prescribed constant at each point $\boldsymbol{X}$ on Γ_i, and we assume that the density of occupied binding sites $C_i^{\mathrm{b}}(\boldsymbol{X}, t)$, which we also refer to as the bound chemical density, satisfies

$$\frac{\partial C_i^{\mathrm{b}}}{\partial t} = k_{on}(C_i^{\mathrm{tot}} - C_i^{\mathrm{b}})c_f - k_{off} C_i^{\mathrm{b}} + D_s \Delta_{\boldsymbol{X}} C^{\mathrm{b}} b_i \tag{4.3}$$

at each point $\boldsymbol{X} \in \Gamma_i$. Here, c_f is the value of the fluid-phase chemical concentration in the fluid adjacent to $\boldsymbol{X}$, D_s is the surface diffusion coefficient, and $\Delta_{\boldsymbol{X}}$ is the Laplace-Beltrami operator on the surface.

In model 2, we instead consider a pair of coupled reaction-diffusion equations on each surface Γ_i

$$\frac{\partial C_i^{\mathrm{b}}}{\partial t} = k_{on} C_i^{\mathrm{u}} c_f - k_{off} C_i^{\mathrm{b}} + D_s^{\mathrm{b}} \Delta_{\boldsymbol{X}} C_i^{\mathrm{b}} \tag{4.4}$$

$$\frac{\partial C_i^{\mathrm{u}}}{\partial t} = -k_{on} C_i^{\mathrm{u}} c_f + k_{off} C_i^{\mathrm{u}} + D_s^{\mathrm{u}} \Delta_{\boldsymbol{X}} C_i^{\mathrm{u}}. \tag{4.5}$$

Here, the quantity $C_i^{\mathrm{u}}(\boldsymbol{X}, t)$ is the surface density of unoccupied binding sites at $\boldsymbol{X} \in \Gamma_i$ at time t, and D_s^u is the surface diffusion coefficient for these sites. In this variant, all binding sites diffuse on the surface, so the total density of binding sites at $\boldsymbol{X}$, $C_i^{\mathrm{tot}}(\boldsymbol{X}, t) = C_i^{\mathrm{b}}(\boldsymbol{X}, t) + C_i^{\mathrm{u}}(\boldsymbol{X}, t)$, can change in time. The setup of model 2 is intended to better represent the biological fact that the binding sites are proteins embedded in the platelet's lipid membrane and that both occupied and unoccupied proteins may diffuse. For both problems, initial data for C_i^{b} and C_i^{u} are given at all points on Γ_i.

The surface chemistry and chemical transport are coupled to that in the fluid because of the appearance of c_f in the surface Equations (4.3) or (4.4) and (4.5), and because of the appearance of C_i^{b} and C_i^{u} in the boundary conditions (4.2) for the fluid-phase chemical. (For Problem 1, we set $C_i^{\mathrm{u}} = C_i^{\mathrm{tot}} - C_i^{\mathrm{b}}$ for each point $\boldsymbol{X} \in \Gamma_i$.) In our earlier work [93], only the reaction portions of these equations were considered, that is, there was no surface diffusion, and the chemical surface densities at different points were coupled only indirectly through the diffusion of fluid-phase chemicals.

4.3 Grid and Platelet Geometry

In this section, we introduce some terminology relevant to the rectangular Eulerian grid and to the Augmented Forcing Method.

We overlay a uniform Cartesian grid with spacing h over the domain of interest, Ω_0. Let $(x_i, y_j) = (ih, jh)$ denote a point of the grid. We require c only within the domain Ω but nevertheless define c_{ij} for all points of the mesh. Let $t_n = n\Delta t$ be the current time, where Δt is the time-step.

To simplify the exposition, we assume that there exists a single irregular object Ω_1 with boundary Γ_1. We may now classify each grid point based on its relation to the irregular object. Grid points in the domain Ω are called *fluid points*; a grid point that is covered by the object with at least one neighboring grid point not covered by the object is called a *forcing point*; finally, the grid points covered by the object that are not forcing points are called *solid points*. We also define *boundary points*, which are points on the boundary of the object whose inward normal vectors pass through forcing points; consequently, there

are as many boundary points as there are forcing points. This labeling process extends to the case when multiple irregular objects exist in the domain.

We model our platelets using the RBF geometric model developed in Chapter 2. For our convenience, we also define an *evaluation matrix*, as described in Chapter 3.

4.4 Numerical Solution of Reaction-diffusion Models for Platelet Chemistry

In this section, we discuss our numerical method for simulating the two models presented in Section 4.2, and our choices for approximating the different components of those models. Before we proceed, it is useful to discuss our time-stepping scheme for the numerical solution to the coupled problem presented in Section 4.2. As in [93], we use a fractional step approach in which during each time-step, we first update the surface densities C_i^{b} and C_i^{u} for each platelet i using known values of the fluid-phase concentrations to determine the values of c_f in Equation (4.3) or Equations (4.4)-(4.5). Then, using the new as well as older surface densities in the boundary conditions (4.2), we update the fluid-phase concentration by solving Equation (4.1). Hence, in describing how we advance each of the surface densities or fluid-phase concentration, we regard the other as known.

In order to obtain the numerical solution of the PDEs of models 1 and 2, several components are required. An approximation of the local fluid-phase chemical concentration c_f must be obtained at each sample site; then, an approximation to the surface Laplacian must be computed; finally, stable and efficient time-stepping schemes must be selected to advance the solutions in time.

4.4.1 Interpolating Fluid-phase Concentrations

In [93], Moving Least Squares (MLS) was used in order to construct a smooth approximation to c_f using chemical concentrations from nearby patches of fluid points. This performed better than an alternate approach with bivariate quadratic interpolation, which produced undesirable spatial oscillations in C^{b}. However, there are two potential issues with the use of MLS. First, it requires the solution of several (small) linear systems [86], in this case constructed from the background Eulerian grid; on a very fine grid, this cost may not be trivial. Second, if two platelets are very close to one another, an insufficient number of fluid points may be available for the construction of the MLS approximation to c_f for each of those platelets.

In order to avoid these potential issues, we use bilinear interpolation, as shown in Figure 4.2; our reasoning is that bilinear interpolation has fewer degrees of freedom than bivariate

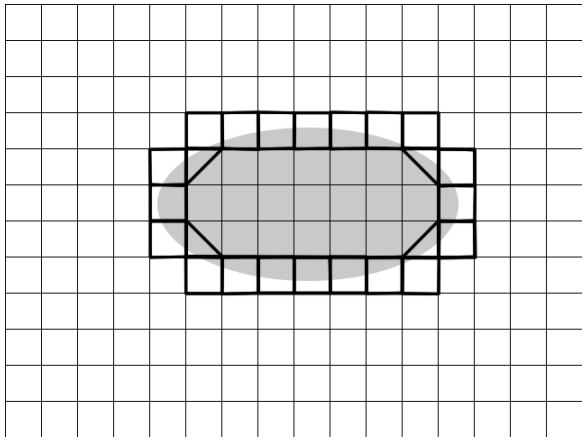

Figure 4.2: Illustration of the bilinear interpolation stencil for a platelet.

quadratic interpolation while not requiring the solution of several linear systems, like MLS does. Our original approach, which we modify somewhat below, is as follows: For each sample site, we first find the Eulerian grid cell in which the sample site is located. If all four corners of the cell are fluid points or forcing points, we use bilinear interpolation of the physically meaningful concentrations at the corners to the sample site. It is possible that one of the corner points is a solid point for which there is no physically meaningful concentration. In that case, we linearly interpolate concentrations from the other three corner points to this fourth corner point.

When we used this approach, our experiments showed that the resulting interpolated chemical concentration field c_f was insufficiently smooth and that the overall accuracy was lower than we expected. Therefore, we instead use this procedure to first interpolate grid concentrations to *data sites* rather than sample sites. We then construct a parametric RBF interpolant of these data in the manner described in Section 4.3. Lastly, we evaluate the RBF interpolant at the sample sites, but with a shape parameter value slightly smaller (by a factor of 0.99), than the value used to construct the interpolant. The theory (and rationale) behind the procedure is described in detail in [5]; essentially, this procedure is a smoothing operation where we interpolate the data with a basis function, and then replace the basis function with a smoother basis function during evaluation. Figure 1 in [5] shows this process with the Multiquadric RBF, where increasing the parameter (c) (equivalent to reducing the shape parameter ϵ_{geom} in our work to some ϵ_{eval}) smoothes a noisy Lidar scan. Section 4 in [5] explains why this is equivalent to using a low-pass filter on the interpolated data by writing out the procedure in terms of convolutions against a smoother basis function. This procedure therefore gives us a smoother concentration field c_f at the

sample sites. In our platelet applications, we use the same parametrization for the platelets even when they are moving. This means that we can precompute the RBF interpolation and evaluation operators and use them with a single matrix-vector multiplication per platelet for each fluid-phase chemical species. Clearly, this approach retains the advantages of our global RBF model while also not deviating from the philosophy of using a least-squares approximant (like MLS) for this purpose.

This method can be used even when two platelets are a single grid cell apart, which is an improvement over the method presented in [93] that requires that platelets be no closer than two grid widths. This feature is required for physically-relevant modeling of aggregation; in platelet aggregation simulations run by the IB method, platelets may indeed be very close to one another when in an aggregate (*e.g.*, see Figure 3.4).

4.4.2 Approximating the Surface Laplacian

Recently, RBFs have been used to compute an approximation to the surface Laplacian in the context of a pseudospectral method for reaction-diffusion equations on manifolds [36], as mentioned in Chapter 1. In that study, *global* RBF interpolants were used to approximate the surface Laplacian at a set of "scattered" nodes on a given surface. To develop a less costly method that is still sufficiently accurate for our purposes, we here use finite difference (FD)-style approximations based on RBFs for the surface Laplacian. This is the first application of the method to general 1D surfaces (curves).

We elect to use Cartesian coordinates rather than surface-based coordinates to formulate the surface Laplacian. This is not very important for 1D surfaces, but very important for generalizing our method to 2D surfaces in the future since a Cartesian-based formulation completely avoids singularities that are associated with surface-based coordinates (*e.g.*, the pole singularity in spherical coordinates). Additionally, the Cartesian-based formulation is quite suitable in the context of approximation with RBFs [36]. Central to this formulation is the projection operator that takes an arbitrary 2D vector field at a point $\boldsymbol{X}$ on the surface and projects it onto the tangent line to the surface at $\boldsymbol{X}$. Letting $\hat{\boldsymbol{\eta}} = (\eta_x, \eta_y)$ denote the *unit* normal vector to the surface at $\boldsymbol{X}$ (which in our applications is obtained from the RBF parametric model of the platelets described in the previous section), this operator is given by

$$\mathcal{P} = \mathcal{I} - \hat{\boldsymbol{\eta}}\hat{\boldsymbol{\eta}}^T = \begin{bmatrix} (1-\eta_x\eta_x) & -\eta_x\eta_y \\ -\eta_x\eta_y & (1-\eta_y\eta_y) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_x^T \\ \mathbf{p}_y^T \end{bmatrix}, \tag{4.6}$$

where $\mathcal{I}$ is the 2-by-2 identity matrix, and $\mathbf{p}_x$ and $\mathbf{p}_y$ represent the projection operators in the x and y directions, respectively. We can combine $\mathcal{P}$ with the standard gradient operator

in $\mathbb{R}^2$, $\nabla = \begin{bmatrix}\partial_x & \partial_y\end{bmatrix}^T$, to define the *surface* gradient operator $\nabla_{\boldsymbol{X}}$ in Cartesian coordinates as

$$\nabla_{\boldsymbol{X}} := \mathcal{P}\nabla = \begin{bmatrix}\mathbf{p}_x \cdot \nabla \\ \mathbf{p}_y \cdot \nabla\end{bmatrix} = \begin{bmatrix}\mathcal{G}_x \\ \mathcal{G}_y\end{bmatrix}. \tag{4.7}$$

Noting that $\Delta_{\boldsymbol{X}} = \nabla_{\boldsymbol{X}} \cdot \nabla_{\boldsymbol{X}}$, the surface Laplacian can then be written in Cartesian coordinates as

$$\Delta_{\boldsymbol{X}} := (\mathcal{P}\nabla\cdot)\mathcal{P}\nabla = \mathcal{G}_x\mathcal{G}_x + \mathcal{G}_y\mathcal{G}_y. \tag{4.8}$$

The approach we use to approximate the surface Laplacian mimics the formulation given in (4.8) and is conceptually similar to that based on global RBFs given in [36]. It is worth noting at this point that though the normal vector is obtained from the parametric representation of the platelet, one could certainly use normal vectors derived from level set representations or, more generally, signed-distance representations of the data sites. Since the literature on computing normal vectors on point clouds is fairly rich, we focus our attention on the exposition of the RBF-FD method, assuming that we are given reasonably smooth normal vectors. The RBF-FD method we now describe is therefore a Cartesian method that can easily handle nonparametrizeable geometries.

Given a set of N nodes, we first construct discrete approximations to $\mathcal{G}_x$ and $\mathcal{G}_y$ using n-node RBF-FD formulas (as explained below). Letting G_x and G_y denote these respective discrete approximations (or differentiation matrices), we then obtain the discrete approximation to the surface Laplacian, L, using the matrix-products as follows:

$$L := G_xG_x + G_yG_y.$$

This approach avoids the need to compute derivatives of the normal vectors of the surfaces, but does have the effect of doubling the bandwidth of the L compared to G_x and G_y.

We explain the RBF-FD method for approximating the $\mathcal{G}_x$ component of the surface gradient in (4.7) as the procedure for $\mathcal{G}_y$ is similar. Without loss of generality, let the sample site where we wish to approximate $\mathcal{G}_x$ be $\boldsymbol{X}_1^s$, and let $\boldsymbol{X}_2^s, \ldots, \boldsymbol{X}_n^s$ be the $n-1$ nearest neighboring sample sites to $\boldsymbol{X}_1^s$. Given samples of a scalar valued function (say chemical density) $C(\boldsymbol{X})$ at these nodes, $C_1, \ldots, C_n$, the goal is to approximate $\mathcal{G}_xC(\boldsymbol{X})$ at $\boldsymbol{X} = \boldsymbol{X}_1^s$ using a linear combination of these samples:

$$\mathcal{G}_xC(\boldsymbol{X})\Big|_{\boldsymbol{X}=\boldsymbol{X}_1^s} \approx \sum_{i=1}^{n} \gamma_i C_i. \tag{4.9}$$

In the RBF-FD method, the differentiation weights, γ_i, are computed by enforcing that this linear combination be exact for each of the RBFs $\left\{\phi(\|\boldsymbol{X}-\boldsymbol{X}_j^s\|)\right\}_{j=1}^n$, *i.e.*,

$$\sum_{i=1}^{n}\gamma_i\phi(\|\boldsymbol{X}_i-\boldsymbol{X}_j^s\|)=\mathcal{G}_x\phi(\|\boldsymbol{X}-\boldsymbol{X}_j^s\|)\Big|_{\boldsymbol{X}=\boldsymbol{X}_1^s}, \tag{4.10}$$

for $j=1,...,n$. Note that $\|\cdot\|$ is the standard two-norm (Euclidean distance) between nodes on the surface and does not depend on any surface metrics (see [31] for a theoretical discussion on using these types of RBF approximations on general surfaces). It has also been shown through experience and studies [91, 26] that better accuracy is gained by additionally requiring that the linear combination (4.9) be exact for a constant. Hence, we also impose the following constraint on the weights γ_i:

$$\sum_{i=1}^{n}\gamma_i=\mathcal{G}_x 1\Big|_{\boldsymbol{X}=\boldsymbol{X}_1^s}=0. \tag{4.11}$$

The conditions (4.10) and (4.11) can be combined into the following linear system for determining the RBF-FD weights γ_i:

$$\begin{bmatrix}\phi(\|\boldsymbol{X}_1^s-\boldsymbol{X}_1^s\|) & \cdots & \phi(\|\boldsymbol{X}_1^s-\boldsymbol{X}_n^s\|) & 1\\ \vdots & \ddots & \vdots & \vdots\\ \phi(\|\boldsymbol{X}_n^s-\boldsymbol{X}_1^s\|) & \cdots & \phi(\|\boldsymbol{X}_n^s-\boldsymbol{X}_n^s\|) & 1\\ 1 & \cdots & 1 & 0\end{bmatrix}\begin{bmatrix}\gamma_1\\ \vdots\\ \gamma_n\\ \gamma_{n+1}\end{bmatrix}=\begin{bmatrix}\mathcal{G}_x\phi(\|\boldsymbol{X}-\boldsymbol{X}_1^s\|)\big|_{\boldsymbol{X}=\boldsymbol{X}_1^s}\\ \vdots\\ \mathcal{G}_x\phi(\|\boldsymbol{X}-\boldsymbol{X}_n^s\|)\big|_{\boldsymbol{X}=\boldsymbol{X}_1^s}\\ 0\end{bmatrix}, \tag{4.12}$$

where γ_{n+1} is a dummy value that is not actually used in RBF-FD approximation after this system is solved.

The solution to (4.12) gives the weights for the first row of the RBF-FD differentiation matrix G_x corresponding to $\boldsymbol{X}_1^s$ (recall the discussion in Section 1.5.2 on the equivalency of the two different methods of computing RBF-FD weights). The weights for the second row corresponding to $\boldsymbol{X}_2^s$ are obtained by finding the $n-1$ nearest neighbors to $\boldsymbol{X}_2^s$ and solving an analogous system to (4.12). This procedure is repeated for determining the weights for the remaining rows of G_x corresponding to sample sites $\boldsymbol{X}_3^s,\ldots,\boldsymbol{X}_{N_s}^s$. The differentiation matrix G_y is obtained using the same procedure, but with the operator $\mathcal{G}_x$ in (4.12) replaced with $\mathcal{G}_y$. Note that each row of G_x and G_y contain only n non-zero entries.

In all the numerical results presented in Section 4.6, we used the Gaussian RBF $\phi(r)=e^{-(\varepsilon r)^2}$ in (4.12) for computing the RBF-FD weights and set $n=3$. In the definition of $\phi(r)$, ε is again called the shape parameter. Provided it is positive and the sample sites are distinct, the linear system (4.12) is guaranteed to be nonsingular, which means the weights are unique. Although not presented here, we did test other RBFs (such as the multiquadric

and inverse multiquadric), but found that the Gaussian generally gave better results for the experiments we ran.

4.4.3 Simulating Models 1 and 2

We use the discretization of the surface Laplacian just described with an implicit-explicit (IMEX) time-stepping scheme, specifically the second-order accurate semi-implicit backward differentiation formula (SBDF2) [1]. For model 1, Equation (4.3), this corresponds to the following discretization:

$$\begin{aligned}\left(I - \frac{2}{3}\Delta t D_s L\right) C^{n+1} =& \frac{4}{3}\left(C^n + \Delta t k_{\text{on}}\left(C^{\text{tot}} - C^n\right) c_f^n - \Delta t k_{\text{off}} C^n\right) \\ & - \frac{1}{3}\left(C^{n-1} + 2\Delta t k_{\text{on}}\left(C^{\text{tot}} - C^{n-1}\right) c_f^{n-1} - 2\Delta t k_{\text{off}} C^{n-1}\right),\end{aligned} \tag{4.13}$$

where Δt is the time-step, and C^{n+1}, C^n, and C^{n-1} denote vectors containing values of the density of unoccupied surface binding sites at the N_s sample sites and at time-steps $n+1$, n, and $n-1$, respectively. Note that (4.13) results in an $N_s \times N_s$ sparse system of equations to solve for C^{n+1}. Since N_s is small, we opt to use a direct method to solve this system of equations, although an iterative method such as BiCGSTAB could have also been used. We note that we bootstrap (4.13) with one step of SBDF1 in the initial time-step.

The discretization for Equations (4.4)–(4.5) in model 2 is similar, but contains a pair of coupled equations. However, the implicit systems that result in these two equations are not coupled, since the coupling is purely through the reaction terms, which are discretized explicitly in time.

4.5 Symmetric Hermite Interpolation for the AFM

In this section, we present our modifications to the AFM based on RBF Hermite interpolation. We apply so-called symmetric RBF Hermite interpolation (presented in Chapter 1) to the problem of enforcing boundary conditions on the fluid-phase chemical concentrations within the AFM, exploiting the general nature of this formulation to overcome the separation constraints imposed by the AFM on the irregular boundaries (platelets) and similar issues that can arise in handling concavities in platelet shapes. In the following subsections, we present our methods for computing the prescribed boundary conditions $\mathbf{r}_{\text{bc}}$ and the matrix E that enforces those boundary conditions in the AFM.

4.5.1 Computing $\mathbf{r}_{\mathrm{bc}}$

Upon rearranging the boundary condition given in Equation (4.2), we obtain the equation

$$\left(-D\frac{\partial}{\partial\hat{\boldsymbol{\eta}}} - k_{\mathrm{on}}C^{\mathrm{u}}\right)c = -k_{off}C^{\mathrm{b}}. \tag{4.14}$$

We use this condition at each boundary point when updating the fluid-phase concentration c^n to c^{n+1}. Because of our fractional-step approach to time-stepping, values of C^{u} and C^{b} are known at the needed times at the locations of the sample sites (for Model 1, we set $C^{\mathrm{u}} = C^{\mathrm{tot}} - C^{\mathrm{b}}$.) These are used to compute values at the boundary points, as described below, and so we can think of them as known in Equation (4.14) and regard this equation as a Robin condition on c^{n+1}. For later reference, we define the Robin boundary condition operator by $\mathcal{D} = -D\frac{\partial}{\partial\hat{\boldsymbol{\eta}}} - k_{\mathrm{on}}C^{\mathrm{u}}$.

The right-hand side of Equation (4.14) describes the known boundary conditions on the fluid-phase chemical concentration and therefore, values of it at the boundary points define the vector $\mathbf{r}_{\mathrm{bc}}$. It is important to note that $\mathbf{r}_{\mathrm{bc}}$ is modified by the procedure to compute the matrix E. It is this modified $\mathbf{r}_{\mathrm{bc}}$ that makes its way into the right-hand side of Equation (1.5).

Since boundary conditions are enforced at *boundary points* in the AFM, we require require values of C^{b} and C^{u} at those points. There are many ways this can be done. For example, in [93], piecewise quadratic interpolants were fit to the concentrations at the IB points, then evaluated at the boundary points. However, in this work, we have two considerations when making this choice. First, we would like the resulting concentration field to be smooth enough to ensure that the overall convergence of our method is not affected. Second, we require that the interpolant (or more generally, approximant) be efficient to compute and evaluate. This rules out directly interpolating concentrations at the sample sites as the number of sample sites (N_s) is much greater than the number of data sites (N_{d}) in our geometric model; it would be more efficient to construct an approximant that had as many coefficients as the number of data sites. With these considerations in mind, we determined that a parametric least-squares fit using the RBF geometric model, described below, would be a good choice.

Let $\vec{C} = [C_1, C_2, \ldots, C_{N_s}]^T$ be a vector of function values at the sample sites, representing either C^{u} or C^{b} values at those sites. As in Chapter 3, we define an $N_s \times N_{\mathrm{d}}$ RBF evaluation matrix, B. When B is applied to the known vector of coefficients of an RBF interpolant of some quantity defined at the data sites, we obtain values of that quantity at the sample sites. Here, we use B in a somewhat different way; as the coefficient

matrix in a least-squares problem. We seek N_{d} coefficients $\vec{g}$ that minimize the quantity $||B\vec{g}-\vec{C}||_2^2$; that is, we seek the coefficients of the N_{d}-term RBF expansion that best fits the N_s sample-site function values that are contained in $\vec{C}$. Since the matrix B depends only on the fixed parameter nodes of the data sites and sample sites, and not on their actual spatial locations, it does not change in time. Thus, we precompute QR-decomposition of B reuse in each time-step to solve these least squares problems.

Once we obtain the coefficients $\vec{g}$, we evaluate the least-squares approximant at the boundary points, giving us values of the chemical surface densities C^{u} and C^{b} at the N_F boundary points. This is done by building an $N_F \times N_{\mathrm{d}}$ evaluation matrix, $\hat{B}$, and applying it to the coefficient vector $\vec{g}$. In the current paper, since the platelets are stationary, $\hat{B}$ can be precomputed and reused every time-step. In the more general problem where platelets are advected and deformed by a background flow, $\hat{B}$ must be recomputed every time-step, since the parameter values corresponding to the boundary points change as the platelets move relative to the background grid.

4.5.2 Enforcing Boundary Conditions with Matrix E

We now introduce an alternative method for computing the interpolation matrix E, which we refer to as E_{rbf}. Our technique for enforcing boundary conditions is conceptually similar to the technique used in the original AFM. However, there are some significant differences, illustrated in Figure 4.3. For each forcing point, we now choose *three*, rather than five, nearby fluid points immediately outside the boundary to use in constructing an interpolant that satisfies the boundary conditions. Additionally, instead of using the boundary condition at a single boundary point, we now use the boundary conditions at *three* boundary points in constructing our interpolant. These changes (in conjunction with the bilinear interpolation scheme outlined in Section 4.3) allow platelets simulated by our modified AFM to be as close as a grid cell width apart, something which the original AFM

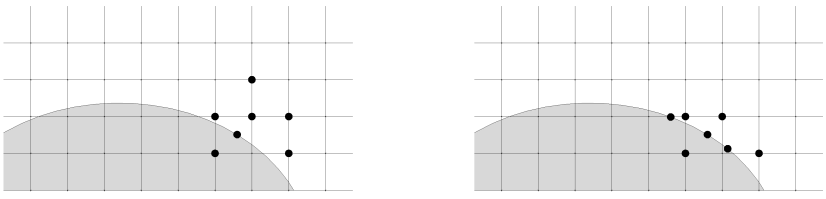

Figure 4.3: The figure on the left shows the number of fluid points and boundary points used in the original AFM. The figure on the right shows the number of fluid points and boundary points used within the modified AFM.

does not allow.

Suppose we wish to impose Robin boundary conditions for c along Γ_1 using the Robin boundary operator $\mathcal{D}$ from the previous subsection. Let forcing point $\boldsymbol{B}$ have coordinates (x_a, y_a) and let the corresponding boundary point $\boldsymbol{p}_a$ have coordinates (X_a, Y_a). As a prototypical example, consider the layout of points in Figure 4.4. Here, $\boldsymbol{p}_b$ and $\boldsymbol{p}_c$ are the two boundary points closest to $\boldsymbol{p}_a$, and $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$ are the three fluid points that are neighbors to the forcing point $\boldsymbol{B}$. Let c_1, c_2, and c_3 be the chemical concentrations at those fluid points, and recall that the boundary conditions are known at $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ as at $\boldsymbol{p}_c$. We note that fluid points need not necessarily be chosen as shown in Figures 4.3 and 4.4; our method only requires that the selected fluid points be close to the boundary of the platelet. We use the symmetric RBF Hermite interpolation technique to obtain an expression for the chemical concentration at each forcing point. In this approach, we construct interpolants of the form:

$$s_{\boldsymbol{B}}(\boldsymbol{p}) = \sum_{i=1}^{3} a_i \phi(\|\boldsymbol{p} - \boldsymbol{p}_i\|) + b_1 \mathcal{D}_{\boldsymbol{p}_a} \phi(\|\boldsymbol{p} - \boldsymbol{p}_a\|) + b_2 \mathcal{D}_{\boldsymbol{p}_b} \phi(\|\boldsymbol{p} - \boldsymbol{p}_b\|) + b_3 \mathcal{D}_{p_c} \phi(\|\boldsymbol{p} - \boldsymbol{p}_c\|), \tag{4.15}$$

where the boundary condition operator with subscripts is defined as

$$\mathcal{D}_{\boldsymbol{p}_\iota} \phi(\|\boldsymbol{p} - \boldsymbol{p}_\iota\|) := \mathcal{D}\phi(\|\boldsymbol{p} - \boldsymbol{x}\|)\Big|_{\boldsymbol{x}=\boldsymbol{p}_\iota}, \quad \iota = a, b, c,$$

i.e., $\mathcal{D}$ acts on ϕ as a function of the subscript variable put on $\mathcal{D}$, with the other variable fixed. The interpolation conditions are given as

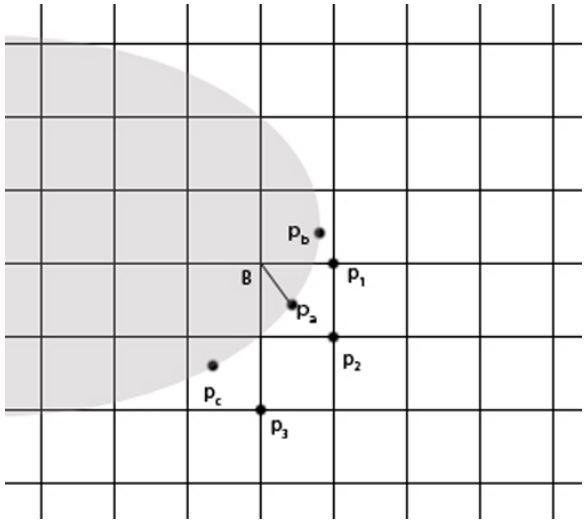

Figure 4.4: Illustration of the symmetric Hermite RBF interpolation stencil.

$$s_{\boldsymbol{B}}(\boldsymbol{p}_j) = c_j, \quad j = 1, 2, 3, \tag{4.16}$$

$$\mathcal{D}s_{\boldsymbol{B}}(\boldsymbol{p})\Big|_{\boldsymbol{p}=\boldsymbol{p}_\iota} = \mathcal{D}c(\boldsymbol{p})\Big|_{\boldsymbol{p}=\boldsymbol{p}_\iota} = d_\iota, \quad \iota = a, b, c, \tag{4.17}$$

where the chemical concentrations c_j are unknown ones from the end of the current time-step, and the boundary conditions d_j are known. These interpolation conditions can be written as the following block 2×2 linear system of equations for determining the unknown coefficients, a_i and b_i in (4.15)

$$\underbrace{\begin{bmatrix} G & R \\ R^T & H \end{bmatrix}}_{V_B} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_{\boldsymbol{B}} \\ \mathbf{d}_{\boldsymbol{B}} \end{bmatrix}^{n+1}, \tag{4.18}$$

where $\mathbf{a}$ and $\mathbf{b}$ are vectors containing the unknown interpolation coefficients, $\mathbf{c}_{\boldsymbol{B}}$ and $\mathbf{d}_{\boldsymbol{B}}$ are vectors containing the respective chemical concentration (4.16) and boundary condition data (4.17) for the forcing point $\boldsymbol{B}$, and the $n+1$ superscript denotes that the values are given at the next time-level. The matrix blocks in this system are defined as follows:

$$G = \begin{bmatrix} \phi(\|\boldsymbol{p}_1 - \boldsymbol{p}_1\|) & \phi(\|\boldsymbol{p}_1 - \boldsymbol{p}_2\|) & \phi(\|\boldsymbol{p}_1 - \boldsymbol{p}_3\|) \\ \phi(\|\boldsymbol{p}_2 - \boldsymbol{p}_1\|) & \phi(\|\boldsymbol{p}_2 - \boldsymbol{p}_2\|) & \phi(\|\boldsymbol{p}_2 - \boldsymbol{p}_3\|) \\ \phi(\|\boldsymbol{p}_3 - \boldsymbol{p}_1\|) & \phi(\|\boldsymbol{p}_3 - \boldsymbol{p}_2\|) & \phi(\|\boldsymbol{p}_3 - \boldsymbol{p}_3\|) \end{bmatrix}, \tag{4.19}$$

$$R = \begin{bmatrix} \mathcal{D}_{\boldsymbol{p}_a}\phi(\|\boldsymbol{p}_1 - \boldsymbol{p}_a\|) & \mathcal{D}_{\boldsymbol{p}_b}\phi(\|\boldsymbol{p}_1 - \boldsymbol{p}_b\|) & \mathcal{D}_{\boldsymbol{p}_c}\phi(\|\boldsymbol{p}_1 - \boldsymbol{p}_c\|) \\ \mathcal{D}_{\boldsymbol{p}_a}\phi(\|\boldsymbol{p}_2 - \boldsymbol{p}_a\|) & \mathcal{D}_{\boldsymbol{p}_b}\phi(\|\boldsymbol{p}_2 - \boldsymbol{p}_b\|) & \mathcal{D}_{\boldsymbol{p}_c}\phi(\|\boldsymbol{p}_2 - \boldsymbol{p}_c\|) \\ \mathcal{D}_{\boldsymbol{p}_a}\phi(\|\boldsymbol{p}_3 - \boldsymbol{p}_a\|) & \mathcal{D}_{\boldsymbol{p}_b}\phi(\|\boldsymbol{p}_3 - \boldsymbol{p}_b\|) & \mathcal{D}_{\boldsymbol{p}_c}\phi(\|\boldsymbol{p}_3 - \boldsymbol{p}_c\|) \end{bmatrix}, \tag{4.20}$$

and

$$H = \begin{bmatrix} \mathcal{D}_{\boldsymbol{p}_a}\big(\mathcal{D}_{\boldsymbol{p}_a}\phi(\|\boldsymbol{p}_a - \boldsymbol{p}_a\|)\big) & \mathcal{D}_{\boldsymbol{p}_a}\big(\mathcal{D}_{\boldsymbol{p}_b}\phi(\|\boldsymbol{p}_a - \boldsymbol{p}_b\|)\big) & \mathcal{D}_{\boldsymbol{p}_a}\big(\mathcal{D}_{\boldsymbol{p}_c}\phi(\|\boldsymbol{p}_a - \boldsymbol{p}_c\|)\big) \\ \mathcal{D}_{\boldsymbol{p}_b}\big(\mathcal{D}_{\boldsymbol{p}_a}\phi(\|\boldsymbol{p}_b - \boldsymbol{p}_a\|)\big) & \mathcal{D}_{\boldsymbol{p}_b}\big(\mathcal{D}_{\boldsymbol{p}_b}\phi(\|\boldsymbol{p}_b - \boldsymbol{p}_b\|)\big) & \mathcal{D}_{\boldsymbol{p}_b}\big(\mathcal{D}_{\boldsymbol{p}_c}\phi(\|\boldsymbol{p}_b - \boldsymbol{p}_c\|)\big) \\ \mathcal{D}_{\boldsymbol{p}_c}\big(\mathcal{D}_{\boldsymbol{p}_a}\phi(\|\boldsymbol{p}_c - \boldsymbol{p}_a\|)\big) & \mathcal{D}_{\boldsymbol{p}_c}\big(\mathcal{D}_{\boldsymbol{p}_b}\phi(\|\boldsymbol{p}_c - \boldsymbol{p}_b\|)\big) & \mathcal{D}_{\boldsymbol{p}_c}\big(\mathcal{D}_{\boldsymbol{p}_c}\phi(\|\boldsymbol{p}_c - \boldsymbol{p}_c\|)\big) \end{bmatrix}. \tag{4.21}$$

The matrices G and H are symmetric so that the composite matrix V_B in Equation (4.18) is symmetric. Moreover, for our choice of ϕ (again, the multiquadric RBF), the matrix is guaranteed to be nonsingular provided the nodes $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_3$, and $\boldsymbol{p}_a$, $\boldsymbol{p}_b$, $\boldsymbol{p}_c$ are distinct [15, Chapter 36]. Finally, we note that in our numerical tests, V_B is also well-conditioned, thus allowing us to not only use very closely spaced fluid points from a fine grid to perform the interpolation, but also a low value for the shape parameter associated with the multiquadric RBF. The goal is to use the interpolant (4.15) to construct the matrix E_{rbf} for enforcing boundary conditions on the interpolated chemical concentrations at the forcing points at time-level $n+1$. This matrix has dimensions $N_F \times N_T$, where N_F is the number of forcing points and N_T is the total number of grid points, and serves the same purpose as E does in the original AFM matrix (1.5). The entries of E_{rbf} can be obtained from (4.15) as

follows. First, we express the interpolated chemical concentration at the forcing point $\boldsymbol{B}$ as a linear combination of the chemical concentrations at the fluid grid points and the boundary conditions at the boundary points. The former are unknown as they are specified at time $n+1$, while the latter are known (see Section 5.2). The weights in this linear combination can be determined by noting that the value of the interpolant (4.15) at the forcing point $\boldsymbol{p} = \boldsymbol{B}$ can be written as

$$s_{\boldsymbol{B}}(\boldsymbol{B}) = S_{\boldsymbol{B}} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \underbrace{S_{\boldsymbol{B}} V_{\boldsymbol{B}}^{-1}}_{Q_{\boldsymbol{B}}} \begin{bmatrix} \mathbf{c}_{\boldsymbol{B}} \\ \mathbf{d}_{\boldsymbol{B}} \end{bmatrix}^{n+1}, \tag{4.22}$$

where $S_{\boldsymbol{B}}$ is the row vector

$$S_{\boldsymbol{B}} = \begin{bmatrix} \phi\left(\|\boldsymbol{B} - \boldsymbol{p}_1\|\right) \\ \phi\left(\|\boldsymbol{B} - \boldsymbol{p}_2\|\right) \\ \phi\left(\|\boldsymbol{B} - \boldsymbol{p}_3\|\right) \\ \mathcal{D}_{\boldsymbol{p}_a}\phi(\|\boldsymbol{B} - \boldsymbol{p}_a\|) \\ \mathcal{D}_{\boldsymbol{p}_b}\phi(\|\boldsymbol{B} - \boldsymbol{p}_b\|) \\ \mathcal{D}_{\boldsymbol{p}_c}\phi(\|\boldsymbol{B} - \boldsymbol{p}_c\|) \end{bmatrix}^T.$$

Thus, $Q_{\boldsymbol{B}}$ contains the weights for the linear combination of chemical concentrations and boundary conditions. Letting $c_{\boldsymbol{B}} := s_{\boldsymbol{B}}(\boldsymbol{B})$ and q_1, q_2, …, q_6 denote the entires of $Q_{\boldsymbol{B}}$, we next write this linear combination as

$$q_1 c_1 + q_2 c_2 + q_3 c_3 - c_{\boldsymbol{B}} = -q_4 d_a - q_5 d_b - q_6 d_c, \tag{4.23}$$

where we have arranged the unknown values of the chemical concentration at time-level $n+1$ on the left-hand side and the known values of the boundary conditions on the right-hand side.

The weights on the left-hand side of (4.23) constitute the entries in one row of the evaluation matrix E_{rbf} corresponding to the forcing point $\boldsymbol{B}$. The columns for these entries correspond to the indices of the matching grid points for $\boldsymbol{B}$, $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$. Specifically, if $\boldsymbol{B}$ is the k^{th} boundary point and has lexicographic grid-index j_1, while $\boldsymbol{p}_1$ $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$ have lexicographic indices j_2, j_3, and j_4, then the k^{th} row of E_{rbf} has non-zero entries

$$(E_{\text{rbf}})_{k,j_1} = -1,\ (E_{\text{rbf}})_{k,j_2} = -q_1,\ (E_{\text{rbf}})_{k,j_3} = -q_2,\ \text{and}\ (E_{\text{rbf}})_{k,j_4} = -q_3.$$

Similarly, the vector of known boundary conditions $\mathbf{r}_{\text{bc}}$ in (1.5) is populated with values from the right-hand side of (4.23). Specifically,

$$(\mathbf{r}_{\text{bc}})_k = -q_4 d_a - q_5 d_b - q_6 d_c,$$

where d_a, d_b, and d_c depend on the location of the boundary point $\boldsymbol{B}$ (see Figure 4.4). Note that prior to modification, $(\mathbf{r}_{\text{bc}})_k$ had the value d_a, which was in turn determined according to the method outlined in Section 5.2.

The above procedure is repeated for each of the N_F forcing points $\boldsymbol{B}$. The resulting matrix E_{rbf} is clearly sparse, with at most four non-zero entries per row (the matrix E from [93] has up to six non-zero entries per row). This procedure can be used to enforce Dirichlet and Neumann boundary conditions as well. We note that the matrix E_{rbf} serves the same function as the matrix E mentioned in Section 5.1 and described in [93], but requires fewer fluid points in its construction. It is thus far more flexible in handling geometric features of the immersed objects.

4.6 Results

In this section, we present the results of the numerical experiments performed to analyze the effects of the changes made to the AFM, as well as the results of experiments performed to analyze the new method for the coupled problems proposed in this paper. We first comment on the selection of the various shape parameters used in this work. Then, we analyze the properties of the RBF-FD discretization scheme for solving pure diffusion equations on platelet surfaces. Next, we examine the behavior of the modified AFM when using analytic boundary conditions (as opposed to deriving boundary conditions from the reaction-diffusion equations on the irregular boundaries). Having tested the convergence of the modified AFM with analytic boundary conditions, we examine the effect of varying the distance between forcing points and boundary points on the accuracy of the modified AFM. We then test the convergence of the combined method on two coupled problems, where the boundary conditions for the AFM are derived from platelet surface reaction model 1. Finally, we test the convergence of the combined method on a single coupled problem with boundary conditions for the AFM derived from platelet surface reaction model 2. Throughout this section, we compute absolute errors on the Cartesian grid and RMS (root mean squared) errors on the surfaces (as in [93]).

4.6.1 Selection of Shape Parameters

In this paper, we use RBFs in several contexts. Here, we list the shape parameters for each of those RBFs and describe the process of obtaining those shape parameters.

1. Geometry: the RBF used for the geometric modeling of the platelets is a parametric interpolant. For the selection of the shape parameter for this RBF, we follow the results obtained in Chapter 2. For this work, we set that shape parameter to $\varepsilon_{geom} = 0.9$.
2. Smoothing c_f: we use ε_{geom} as the shape parameter for the parametric fit of c_f at the data sites. To evaluate the RBF interpolant at sample sites and also smooth it, we

use $\varepsilon_{eval} = 0.99\varepsilon_{geom}$.

3. Surface Laplace-Beltrami operator: local RBFs are used for computing the RBF-FD approximation to the surface Laplace-Beltrami operator. For these RBFs, for all tests, the shape parameter was set to $\varepsilon_{fd} = 35$. This choice was motivated, in part, by the desire to compensate for irregular point spacings on some of the perturbed objects in the tests. We note that the comparitively large value of ε_{fd} is due to the partial dependence of the PDE to the Cartesian grid via the c_f term, and also due to the fact that we are using the Gaussian RBF with small node spacings for the interpolation.
4. Chemical densities on platelet surfaces: we once again use the parametric model, albeit for a least-squares fit. We use the same value for the shape parameter as we do for the geometric modeling.
5. Hermite interpolant: we set the shape parameter of all the RBF Hermite interpolants (one for each forcing point) to $\varepsilon_{herm} = 5$. We found that a wide range of values could be used for ε_{herm} without adversely affecting the accuracy of the AFM.

4.6.2 RBF-FD on a Circle

We test the RBF-FD method for solving a pure diffusion equation on the surface of a platelet. In order to test the effect of the geometric model on the solution of the diffusion equations on the irregular boundaries by the RBF-FD method, we prescribe an initial chemical density $C(\lambda, 0) = (\cos\lambda + \sin\lambda)$ on the unit circle with $0 \leq \lambda < 2\pi$. For $t \geq 0$, the function $C(\lambda, t) = e^{-t}(\cos\lambda + \sin\lambda)$ is then an exact solution to the diffusion equation on the circle when $D_s = 1$. We fix the number of data sites to $N_{\rm d} = 50$ and vary the number of sample sites. To test the errors in the spatial discretization, we fix the time-step at $\Delta t = 10^{-4}$. The test was run from $t = 0$ to $t = 2$. The results of this test are shown in Table 4.1. The results demonstrate that the RBF-FD solution to the diffusion equation on a circle exhibits second-order convergence in the sample site spacing. Similar experiments with irregularly-spaced points around the circle (results not shown) show that the convergence of the RBF-FD method gradually decreases to first-order as the points

Table 4.1: The effect of geometric accuracy on the RBF-FD solution to the diffusion equation. The errors were measured against the exact solution at $t = 2$.

$N_{\rm s}$	L_2 error	Order	L_∞ error	Order
50	2.0591e-03		2.9106e-03	
100	5.0705e-04	2.02	7.1672e-04	2.02
200	1.2152e-04	2.06	1.7185e-04	2.06

become more irregularly spaced. However, the method appears tolerant to mildly uneven point spacings, both on the circle and on the test objects in Coupled Problems 2 and 3.

4.6.3 Convergence of the Modified AFM

Table 4.2 shows the results of a refinement study conducted with the modified AFM. The solution was taken to be $c(x, y, t) = \sin(\pi x)\sin(\pi y)e^{-\pi^2 t}$. The initial chemical concentration was prescribed using the values of this function at $t = 0$, and analytic boundary conditions were prescribed by applying the boundary condition operator $\mathcal{D}$ to c. These boundary conditions were enforced at boundary points on platelet surfaces in the modified AFM calculations, and, on the computational domain boundary, the exact solution satisfies periodic boundary conditions in the x-direction and Neumann boundary conditions in the y-direction (though our results were similar for Dirichlet and Neumann boundary conditions as well). For our refinement study, we used the Robin boundary condition operator $\mathcal{D} = -D\frac{\partial}{\partial \hat{\boldsymbol{\eta}}} + 1$ with a diffusion coefficient $D = 0.1$. Two objects were embedded in the domain, a circle C1 and an ellipse E1. C1 has its center at (0.2,0.4) and a radius of 0.0995, while E1 has its center at (0.8,0.4), a semimajor axis of length 0.15 and a semiminor axis of length 0.1. We compare the solution on grids of several sizes to a solution computed on a 256×256 grid. We also reduce the time-step by half for each progressively finer grid. The test was run from $t = 0$ to $t = 3$. The results demonstrate that the modified AFM exhibits second-order convergence in both space and time when analytic boundary conditions are prescribed.

4.6.4 Effect of Forcing Point Locations

We wished to test whether convergence of the modified AFM is sensitive to the distance between the boundary of an irregular object and the forcing points on the grid. To accomplish this, we placed an object that looks like a square with rounded corners in the center of the domain; technically, this object is a superquadric and is shown in Figure 4.5. We generated the object parametrically as follows:

Table 4.2: Results of a refinement study for the modified AFM. The errors were measured against a solution computed on a 256×256 grid as a gold standard. The errors were measured at $t = 3$.

Grid Size	Δt	L_2 error	Order	L_∞ error	Order
32×32	0.0050	9.0012e-07		3.3407e-06	
64×64	0.0025	2.2716e-07	1.99	8.8616e-07	1.92
128×128	0.00125	5.2988e-08	2.10	2.0742e-07	2.10

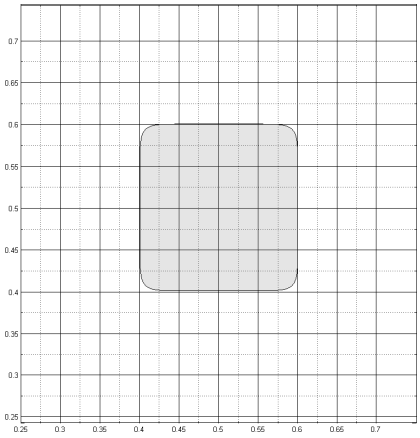

Figure 4.5: Illustration of the quadric object used to test the effect of forcing point locations on the modified AFM.

$$X = x_c + r * \text{sign}(\cos\lambda)(p_x|\cos\lambda|)^m \tag{4.24}$$

$$Y = y_c + r * \text{sign}(\sin\lambda)(p_y|\sin\lambda|)^m \tag{4.25}$$

where $(x_c, y_c) = (0.5, 0.5)$, $m = 0.2$, $r = 0.0995$, and $0 \leq \lambda < 2\pi$. The test involved squeezing the sides (or top and bottom) of the object in such a way that the boundary shifts *between* grid lines without its actually crossing a grid line and thus causing generation of a new set of forcing points. We accomplished this by varying the parameters p_x and p_y; reducing p_x or p_y squeezes the object either along the horizontal or the vertical, respectively, while increasing these parameters stretches the object. For this test, we successively reduced either p_x or p_y from 1.1 to 0.7, with $p_x = 1$ and $p_y = 1$ corresponding to the unchanged object.

We measured the error in approximating the manufactured solution to the test function $c(x, y, t) = \sin(\pi x)\sin(\pi y)e^{-\pi^2 t}$ for these different values of p_x and p_y. We used analytic normals and sample sites locations for the rounded square so as to remove the effect of interpolation error. We set $D = 0.2$ and perform our tests on a 64×64 grid with time-step $\Delta t = 0.0025$. We found that that the errors were unaffected by the distance between the boundary and the forcing points.

4.6.5 Convergence on Coupled Problems for Model 1

We next report on tests of the convergence of the modified AFM in conjunction with the RBF-FD method on two coupled problems. In Coupled Problem 1, the boundary conditions at boundary points for the modified AFM were obtained from the solution of reaction-diffusion equations on the surfaces of platelets C1 and E1. The diffusion coefficient for the fluid-phase chemical concentrations was set to $D = 0.1$ and that for the surface of

the platelets was set to $D_s = 1$ for both platelets. The reaction rates were set to $k_{on} = 0.2$ and $k_{off} = 0.4$ for C1, and to set to $k_{on} = 0.4$ and $k_{off} = 0.2$ for E1. The fluid-phase concentrations were initialized to $c(x, y, 0) = \sin(\pi x)\sin(\pi y)$ while the platelet densities were initialized to $C(\lambda, 0) = \cos(\lambda)$, for $0 \leq \lambda < 2\pi$ for both C1 and E1. The test was run from $t = 0$ to $t = 3$. Convergence was measured for both the fluid-phase concentrations and the platelet-surface concentrations. The results shown in Table 4.3 and Table 4.4 demonstrate that the modified AFM with boundary conditions derived from the RBF-FD solution of reaction-diffusion equations on simple platelet surfaces exhibits second-order convergence in both space and time on Coupled Problem 1.

Coupled Problem 2 uses the same parameters as Coupled Problem 1, but differs from that problem in solving surface reaction-diffusion equations on the ellipse E1 and on a smoothly perturbed version of ellipse E1 that we will call PE1 (Perturbed Ellipse 1). The motivation for this test was to study the behavior of the AFM on platelets which may be oddly shaped or stretched ellipses (for example, as they may be when bound to other platelets within a clot). The points on PE1 are given by Equation (2.44) from Chapter 2, with the same parameters used there.

The results of a convergence study of the combined method on Coupled Problem 2 are shown in Table 4.5 and Table 4.6. These results show that the modified AFM in conjunction

Table 4.3: Results of a refinement study for the modified AFM on Coupled Problem 1. The errors were measured by using a solution computed on a 256×256 grid as a gold standard. The number of sample sites was also increased from $N_s = 50$ to $N_s = 200$ as the grid was refined. All errors were measured at $t = 3$.

Grid Size	N_s	Δt	L_2 error	Order	L_∞ error	Order
32×32	50	0.0050	1.2841e-03		1.9135e-03	
64×64	100	0.0025	3.2477e-04	1.98	4.9864e-04	1.94
128×128	200	0.00125	7.5756e-05	2.10	1.2041e-04	2.05

Table 4.4: Results of a refinement study for the RBF-FD solution to reaction-diffusion equations on the surface of platelets in Coupled Problem 1. The errors were measured using a solution computed on a 256×256 grid with analytically computed normals at $N_s = 400$ sample sites as a gold standard. The fluid grid was also refined as the number of sample sites was increased. All errors were measured at $t = 3$.

N_s	Grid Size	Δt	L_2 error	Order	L_∞ error	Order
50	32×32	0.0050	1.5567e-03		2.1497e-03	
100	64×64	0.0025	3.6238e-04	2.10	5.0534e-04	2.09
200	128×128	0.00125	8.3943e-05	2.11	1.1706e-04	2.11

Table 4.5: Results of a refinement study for the modified AFM on Coupled Problem 2. The errors were measured by using a solution computed on a 256×256 grid as a gold standard. The number of sample sites was also increased from $N_s = 50$ to $N_s = 200$ as the grid was refined. All errors were measured at $t = 3$.

Grid Size	N_s	Δt	L_2 error	Order	L_∞ error	Order
32×32	50	0.0050	9.8668e-04		1.5650e-03	
64×64	100	0.0025	2.5374e-04	1.96	4.1418e-04	1.92
128×128	100	0.00125	5.8373e-05	2.12	9.7048e-05	2.09

Table 4.6: Results of a refinement study for the RBF-FD solution to reaction-diffusion equations on the surface of platelets in Coupled Problem 2. The errors were measured using a solution computed on a 256×256 grid with analytically computed normals at $N_s = 400$ sample sites as a gold standard. The grid was refined as the number of sample sites was increased. All errors were measured at $t = 3$.

N_s	Grid Size	Δt	L_2 error	Order	L_∞ error	Order
50	32×32	0.0050	1.1976e-03		1.6179e-03	
100	64×64	0.0025	2.7351e-04	2.13	3.6505e-04	2.15
200	128×128	0.00125	6.1624e-05	2.15	8.3978e-05	2.13

with the RBF-FD method for solving reaction-diffusion equations on perturbed platelet surfaces exhibits second-order convergence in both space and time.

4.6.6 Convergence on a Coupled Problem for Model 2

Having tested the convergence of the combined method on Coupled Problems 1 and 2 that used model 1, we now test the convergence of the combined method on a coupled problem that uses model 2. For this new coupled problem (Coupled Problem 3), we simulate the equations of model 2 on the objects E1 and PE1 using the RBF-FD method within the AFM. The reaction rates were set to the same as those in Coupled Problem 2, as were the platelet positions. The bound and unbound chemical density fields were initialized to $C^b(\lambda) = \cos(\lambda)$ and $C^u(\lambda) = 1 - C^b(\lambda)$, for $0 \leq \lambda < 2\pi$, respectively. The simulation was run from $t = 0$ to $t = 3$.

The results of the convergence studies are shown in Tables 4.7, 4.8, and 4.9. Having used the same initial conditions and platelet configurations as in Coupled Problem 2, we see identical results in terms of errors and convergence on Coupled Problem 3 for the AFM and for the PDE for C^b. Furthermore, the errors and convergence for the PDE for C^u are identical to the errors and convergence for C^b. We thus observe second-order convergence using our methods on Coupled Problem 3 as well. The advantage of model 2 over model 1, of course, is that one has greater flexibility in model 2, in terms of selecting initial conditions

Table 4.7: Results of a refinement study for the modified AFM on Coupled Problem 3. The errors were measured by using a solution computed on a 256×256 grid as a gold standard. The number of sample sites was also increased from $N_s = 50$ to $N_s = 200$ as the grid was refined. All errors were measured at $t = 3$.

Grid Size	N_s	Δt	L_2 error	Order	L_∞ error	Order
32×32	50	0.0050	9.8668e-04		1.5650e-03	
64×64	100	0.0025	2.5374e-04	1.96	4.1418e-04	1.92
128×128	200	0.00125	5.8373e-05	2.12	9.7048e-05	2.09

Table 4.8: Results of a refinement study for the RBF-FD solution to the reaction-diffusion equations for bound chemical concentrations on the surface of platelets in Coupled Problem 3. The errors were measured using a solution computed on a 256×256 grid with analytically computed normals at $N_s = 400$ sample sites as a gold standard. The grid was refined as the number of sample sites was increased. All errors were measured at $t = 3$.

N_s	Grid Size	Δt	L_2 error	Order	L_∞ error	Order
50	32×32	0.0050	1.1976e-03		1.6179e-03	
100	64×64	0.0025	2.7351e-04	2.13	3.6505e-04	2.15
200	128×128	0.00125	6.1624e-05	2.15	8.3978e-05	2.13

Table 4.9: Results of a refinement study for the RBF-FD solution to the reaction-diffusion equations for unbound chemical concentrations on the surface of platelets in Coupled Problem 3. The errors were measured using a solution computed on a 256×256 grid with analytically computed normals at $N_s = 400$ sample sites as a gold standard. The grid was refined as the number of sample sites was increased. All errors were measured at $t = 3$.

N_s	Grid Size	Δt	L_2 error	Order	L_∞ error	Order
50	32×32	0.0050	1.1976e-03		1.6179e-03	
100	64×64	0.0025	2.7351e-04	2.13	3.6505e-04	2.15
200	128×128	0.00125	6.1624e-05	2.15	8.3978e-05	2.13

for C^u and C^b, and different coefficients of diffusion as well.

4.7 Summary

The Augmented Forcing Method (AFM) was developed in [93] for the simulation of chemical transport in a stationary fluid in the presence of irregular boundaries (platelets). In that work, an ODE model for chemistry on platelet surfaces was also presented, with the ODEs contributing boundary conditions to the fluid-phase chemical diffusion equation and the fluid-phase chemical diffusion equation contributing to the ODEs. When the AFM was used in conjunction with a Crank-Nicolson time-stepping method for the simulation of the combined problem, the resulting method was shown to have second-order accuracy and

convergence. However, the method had the following limitations:

- the ODE model was only a simple approximation to true platelet chemistry; a reaction-diffusion PDE model would be more appropriate;
- the use of Moving Least Squares (MLS) scheme to obtain fluid-phase chemical concentrations at points on the platelet surface imposed a separation constraint on platelets – the platelets had to be at least $2h$ apart, where h is the Cartesian grid spacing; and
- the AFM itself imposed another separation constraint of $2h$ on platelets because of the biquadratic interpolation stencil chosen to enforce boundary conditions on the fluid-phase chemical diffusion equation.

In this work, we introduced more complete models of platelet surface chemistry involving diffusion of chemical densities on the surface. Two models (models 1 and 2) were presented. Model 1 is a simple update to the ODE model that involved adding a surface diffusion term to the ODE (thereby giving a PDE), while model 2 aims to better match the biology of the problem by using a pair of PDEs at each point on the boundary (these PDEs are coupled to each other through their equal and opposite reaction terms).

In order to facilitate the simulation of models 1 and 2 on oddly-shaped platelets (typically seen in platelet aggregation simulations) in 2D and to remove the limitations of the AFM in its original form, we presented the following numerical methodology:

- the first application of Radial Basis Function-Finite Differences (RBF-FD) to the simulation of reaction-diffusion equations on surfaces in 2D;
- a modification to the AFM involving symmetric RBF Hermite interpolation (instead of biquadratic interpolation) to enforce boundary conditions on the fluid-phase chemical diffusion equation, thus eliminating the separation constraint on platelets simulated by the AFM; and
- a replacement for the MLS scheme used in [93] with a simple bilinear interpolation and a parametric RBF-based smoothing scheme, thereby eliminating the other separation constraint on platelets in that work.

Through numerical experiments, we analyzed the behavior of our proposed methodology and draw the following conclusions:

- the RBF-FD approximation to the surface Laplacian, when used in conjunction with a BDF2 scheme, resulted in a method that exhibited second-order convergence in both space and time when applied to the simulation of pure diffusion equations on circles;

- the symmetric RBF Hermite interpolation scheme for enforcing boundary conditions within the AFM gave a modified AFM that also exhibited second-order convergence in both space and time for diffusion of fluid-phase concentrations; and
- the combined methodology involving RBF-FD and the AFM showed second-order convergence in both space and time on three coupled problems involving reaction-diffusion equations on platelet surfaces and a diffusion equation for the fluid-phase concentrations; Coupled Problems 1 and 2 used model 1 (simulated with SBDF2), while Coupled Problem 3 used model 2 (also simulated with SBDF2).

While we have indeed shown that the RBF-FD method can be successfully applied to the simulation of reaction-diffusion equations on platelet-like surfaces in 2D, we have yet to explore the effects on this method of using different stencil sizes and different point spacings on surfaces embedded in 3D domains. We will discuss this in Chapter 5. Also, like [93], while our results are valid for stationary platelets in stationary fluid, we have yet to explore the modified AFM and the RBF-FD method for platelets interacting with a moving fluid as simulated by the Immersed Boundary method. This is a subject for future work.

CHAPTER 5

RBF-FD FOR TWO-DIMENSIONAL SURFACES

5.1 Introduction

We now turn our attention to simulating reaction-diffusion equations on surfaces embedded in 3D domains. The first natural step was to attempt to simply use the method from Chapter 4, albeit adapted to 2D surfaces. However, in our experiments, a straightforward extension of the RBF-FD approach presented in the previous chapter to 2D surfaces proved to be unstable, requiring hyperviscosity-based stabilization, an approach commonly used for the stabilization of RBF-FD operators.

In this work, we modify the RBF-FD formulation presented in Chapter 4, and present numerical and algorithmic strategies for generating stable RBF-FD operators on general point sets. This combined approach appears to do away with the need for hyperviscosity-based stabilization. While this work deals with the formulation of RBF-FD surface Laplacians on 2D surfaces, it easily generalizes to both 1D and higher-dimensional surfaces of co-dimension 1 embedded in higher-dimensional domains.

We note that though we frequently interpolate function samples at scattered nodes that lie on manifolds in this work, we still measure distances only in the embedding space of the manifold, *i.e.,* we use straight line distances rather than distances intrinsic to the manifold. In previous work [31], the authors proved that favorable error estimates can be achieved even when using such distance measures with RBF interpolation when applied to reconstruction problems on general manifolds. We will take advantage of this result to formulate our method, and mention some of the ramifications of the choice to use straight-line distances in a later section.

For the remainder of this chapter, we will revert to the notation used in Section 1.5.1, since that is the notation most convenient for discussing interpolation on point clouds and scattered node sets on arbitrary surfaces.

5.2 Surface Laplacian in Cartesian Coordinates

Again, as in the previous chapter, we elect to use Cartesian coordinates rather than surface-based coordinates to formulate the surface Laplacian. Working with the operator in Cartesian coordinates is fundamental to our proposed method as it completely avoids singularities that are associated with using intrinsic, surface-based coordinates (*e.g.*, the pole singularity in spherical coordinates).

Let $\mathcal{P}$ denote the projection operator that takes an arbitrary vector field in $\mathbb{R}^3$ at a point $\boldsymbol{X} = (x, y, z)$ on the surface and projects it onto the tangent plane to the surface at $\boldsymbol{X}$. Letting $\hat{\boldsymbol{\eta}} = (n^x, n^y, n^z)$ denote the *unit* normal vector to the surface at $\boldsymbol{X}$, this operator is given by

$$\mathcal{P} = \mathcal{I} - \hat{\boldsymbol{\eta}}\hat{\boldsymbol{\eta}}^T = \begin{bmatrix} (1-n^x n^x) & -n^x n^y & -n^x n^z \\ -n^x n^y & (1-n^y n^y) & -n^y n^z \\ -n^x n^z & -n^y n^z & (1-n^z n^z) \end{bmatrix} = \begin{bmatrix} \mathbf{p}^x & \mathbf{p}^y & \mathbf{p}^z \end{bmatrix}, \tag{5.1}$$

where $\mathcal{I}$ is the 3-by-3 identity matrix, and $\mathbf{p}^x$, $\mathbf{p}^y$, and $\mathbf{p}^z$ are vectors representing the projection operators in the x, y, and z directions, respectively. We can combine $\mathcal{P}$ with the standard gradient operator in $\mathbb{R}^3$, $\nabla = \begin{bmatrix} \partial_x & \partial_y & \partial_z \end{bmatrix}^T$, to define the *surface* gradient operator $\nabla_{\mathbb{M}}$ in Cartesian coordinates as

$$\nabla_{\mathbb{M}} := \mathcal{P}\nabla = \begin{bmatrix} \mathbf{p}^x \cdot \nabla \\ \mathbf{p}^y \cdot \nabla \\ \mathbf{p}^z \cdot \nabla \end{bmatrix} = \begin{bmatrix} \mathcal{G}^x \\ \mathcal{G}^y \\ \mathcal{G}^z \end{bmatrix}. \tag{5.2}$$

Noting that the surface Laplacian $\Delta_{\mathbb{M}}$ is given as the surface divergence of the surface gradient, this operator can be written in Cartesian coordinates as

$$\Delta_{\mathbb{M}} := \nabla_{\mathbb{M}} \cdot \nabla_{\mathbb{M}} = (\mathcal{P}\nabla) \cdot \mathcal{P}\nabla = \mathcal{G}^x\mathcal{G}^x + \mathcal{G}^y\mathcal{G}^y + \mathcal{G}^z\mathcal{G}^z. \tag{5.3}$$

The approach we use to approximate the surface Laplacian mimics the formulation given in Equation (5.3) and is conceptually similar to the formulation given in [36], with the important difference being that we use local RBF interpolants.

5.3 RBF-FD Approximation to the Surface Laplacian

Let $X = \{\boldsymbol{X}_k\}_{k=1}^N$ denote a set of (scattered) node locations on a surface $\mathbb{M}$ of dimension two embedded in $\mathbb{R}^3$ and suppose $f : \mathbb{M} \to \mathbb{R}$ is some differentiable function sampled on X. Our goal is to approximate $\Delta_{\mathbb{M}} f|_X$ with finite-difference-style local approximations to the operator $\Delta_{\mathbb{M}}$. Without loss of generality, let the node where we want to approximate $\Delta_{\mathbb{M}} f$

be $\boldsymbol{X}_1$, and let $\boldsymbol{X}_2, \ldots, \boldsymbol{X}_n$ be the $n-1$ nearest neighbors to $\boldsymbol{X}_1$, in terms of Euclidean distances measured in $\mathbb{R}^3$. We refer to $\boldsymbol{X}_1$ and its $n-1$ nearest neighbors as the *stencil* on the surface corresponding to $\boldsymbol{X}_1$ and denote this stencil as $P_1 = \{\boldsymbol{X}_k\}_{k=1}^n$. We seek an approximation to $\Delta_{\mathbb{M}} f$ at $\boldsymbol{X}_1$ that involves a linear combination of the values of f over the stencil P_1 of the form

$$(\Delta_{\mathbb{M}} f)\big|_{\boldsymbol{X}=\boldsymbol{X}_1} \approx \sum_{j=1}^{n} w_j f(\boldsymbol{X}_j). \tag{5.4}$$

The weights $\{w_j\}_{j=1}^n$ in this approximation will be computed using RBFs, and will be referred to as RBF-FD weights.

The first step to computing the RBF-FD weights is to construct an RBF interpolant of f similar to Equation (1.7), but now only over the nodes in P_1, *i.e.*,

$$I_\phi f(\boldsymbol{X}) = \sum_{j=1}^{n} c_j \phi(r_j(\boldsymbol{X})) + c_{n+1}, \tag{5.5}$$

where $r_j(\boldsymbol{X}) = \|\boldsymbol{X} - \boldsymbol{X}_j\|$. The interpolation coefficients c_j can be determined by the solution to the system of equations given in Equation (1.8), but with X replaced with P_1; we denote this system by $A_{P_1} c_f = f_{P_1}$. Second, we compute the surface gradient of the above interpolant using Equation (5.2) and evaluate it at the nodes in P_1. In the case of the $\mathcal{G}^x$ component of the gradient, this is given as

$$(\mathcal{G}^x I_\phi f(\boldsymbol{X}))\big|_{\boldsymbol{X}=\boldsymbol{X}_i} = \sum_{j=1}^{n} c_j \underbrace{(\mathcal{G}^x \phi(r_j(\boldsymbol{X})))\big|_{\boldsymbol{X}=\boldsymbol{X}_i}}_{\left(B^x_{P_1}\right)_{i,j}}, i = 1, \ldots, n, \tag{5.6}$$

where the constant term from Equation (5.5) has vanished since its gradient is zero. We can rewrite Equation (5.6) in matrix-vector form using the fact that $c_f = A_{P_1}^{-1} f_{P_1}$ as follows:

$$(\mathcal{G}^x I_\phi f)\big|_{P_1} = B^x_{P_1} c_f = \left(B^x_{P_1} A^{-1}_{P_1}\right) f_{P_1} = G^x_{P_1} f_{P_1}. \tag{5.7}$$

Here $G^x_{P_1}$ is an n-by-n differentiation matrix that represents the RBF approximation to the x-component of the surface gradient operator over the set of nodes in P_1. Similar approximations can be obtained to the y- and z-components of the surface gradient operator on this stencil as follows:

$$(\mathcal{G}^y I_\phi f)\big|_{P_1} = \left(B^y_{P_1} A^{-1}_{P_1}\right) f_{P_1} = G^y_{P_1} f_{P_1}, \tag{5.8}$$

$$(\mathcal{G}^z I_\phi f)\big|_{P_1} = \left(B^z_{P_1} A^{-1}_{P_1}\right) f_{P_1} = G^z_{P_1} f_{P_1}, \tag{5.9}$$

where the entries of $B^y_{P_1}$ and $B^z_{P_1}$ are given as

$$(B^y_{P_1})_{i,j} = (\mathcal{G}^y \phi(r_j(\boldsymbol{X})))\big|_{\boldsymbol{X}=\boldsymbol{X}_i} \text{ and } (B^z_{P_1})_{i,j} = (\mathcal{G}^z \phi(r_j(\boldsymbol{X})))\big|_{\boldsymbol{X}=\boldsymbol{X}_i}.$$

In the third step, we mimic the continuous formulation of the surface Laplacian in Equation (5.3) using the differentiation matrices $G^x_{P_1}$, $G^y_{P_1}$, and $G^z_{P_1}$ in place of the operators $\mathcal{G}^x$, $\mathcal{G}^y$, and $\mathcal{G}^z$, respectively, which gives the following approximation to the surface Laplacian of f at all the nodes in P_1:

$$(\Delta_{\mathbb{M}} f)\big|_{P_1} \approx \underbrace{\left(G^x_{P_1} G^x_{P_1} + G^y_{P_1} G^y_{P_1} + G^z_{P_1} G^z_{P_1}\right)}_{L_{P_1}} f_{P_1}. \tag{5.10}$$

This approximation is equivalent to the following operations: construct an interpolant of f over P_1, compute its surface gradient, interpolate each component of the surface gradient, apply the surface divergence, and evaluate it at P_1. Hence, we can use the vector interpolant notation from Equation (1.9), to write Equation (5.10) equivalently as

$$(\Delta_{\mathbb{M}} f)\big|_{P_1} \approx (\nabla_{\mathbb{M}} \cdot I_\Phi (\nabla_{\mathbb{M}} I_\phi f))\big|_{P_1}.$$

This approach of repeated interpolation and differentiation avoids the need to analytically differentiate the surface normal vectors of $\mathbb{M}$, which imples closed form expressions for these values are not needed. This simplifies the computations and makes the method applicable to surfaces defined by point clouds (as illustrated in Section 5.6).

While the approximation in Equation (5.10) is for all the nodes in P_1, we are only interested in the approximation at $\boldsymbol{X} = \boldsymbol{X}_1$ (the "center" point of the stencil P_1) according to Equation (5.4). Because of the ordering of nodes in P_1, the value of Equation (5.4) is given by the first value in the vector that results from the product on the right of Equation (5.10). Thus, the weights w_j in Equation 5.4 are given by the entries in the first row of the matrix L_{P_1} from Equation (5.10). Extracting these entries from this matrix, and disregarding the rest, then completes the steps for determining the RBF-FD weights for the node $\boldsymbol{X}_1$.

For each node $\boldsymbol{X}_j \in X$, $j = 1, \ldots, N$, we repeat the above procedure of finding its $n-1$ nearest neighbors (stencil P_j), computing the corresponding matrix L_{P_j} according to Equation (5.10), and extracting out of this matrix the row of RBF-FD weights for $\boldsymbol{X}_j$. These weights are then arranged into a *sparse* N-by-N differentiation matrix L_X for approximating the surface Laplacian over all the nodes in X.

The computational cost of computing each matrix L_{P_j} is $O(n^3)$, and there are N such stencils, so that the total cost of computing the entries of L_X is $O(n^3 N)$ (this is apart

from the cost of determining the stencil nodes, for which an efficient method is discussed below). In practice, $n << N$ and would typically be fixed as N increases, so that the total cost scales like $O(N)$. Furthermore, each L_{P_j} can be computed independently from the others and is thus a embarrassingly parallel computation. In contrast, the method from [36], requires $O(N^3)$ operations and results in a dense differentiation matrix. However, the accuracy of this global method is better than the local RBF-FD approach.

5.3.1 Implementation Details

To efficiently determine the members of stencils P_k, $k = 1, \ldots, N$, we first build a k-d tree for the full set of N nodes in X. The k-d tree is constructed in $O(dN \log N)$ operations, where d is the number of dimensions. The members of stencil P_k can then be determined from the k-d tree in $O(\log N)$ operations. Combining the computational cost of the k-d tree construction and look-ups with computing the RBF-FD weights, the total cost of building L_X is $O(N \log N) + O(N)$, where the constants in the last term depend on the cube of n.

We note that points in each of the N stencils P_k on the surfaces are selected merely using a distance criterion; in other words, for a node $\boldsymbol{X}_k$, the stencil only comprises of its $n-1$ nearest neighbors, with the distances measured in $\mathbb{R}^3$, rather than along the surface. While it is possible to include more information to form more regular or biased stencils, we do not explore these possibilities in our current work. One consequence of using distances in the embedding space is that one must exercise caution when simulating PDEs on the surfaces of thin objects, or thin features of more general surfaces. If the distance between points across a thin feature is smaller than the distance between points on the same sides of the surface of the thin feature, a poor approximation to the surface Laplacian will result in that region. We will not address this issue in our work, except by taking care to have a sufficiently dense sampling of the surface around thin features.

In general, the nodes in X will lack any ordering, which may negatively impact the fill-in of the sparse matrix L_X. We therefore reorder the matrix using the Reverse Cuthill-McKee algorithm [37] for all our tests. This usually results in faster iterations in an iterative solver involving L_X, and improved sparsity in stored lower triangular and upper triangular factors within a sparse direct solver involving L_X. Figure 5.1 shows two different surfaces, a idealized red blood cell and a double-torus, with corresponding nodes X, and the resulting sparsity pattern of L_X after reordering with Reverse Cuthill-McKee.

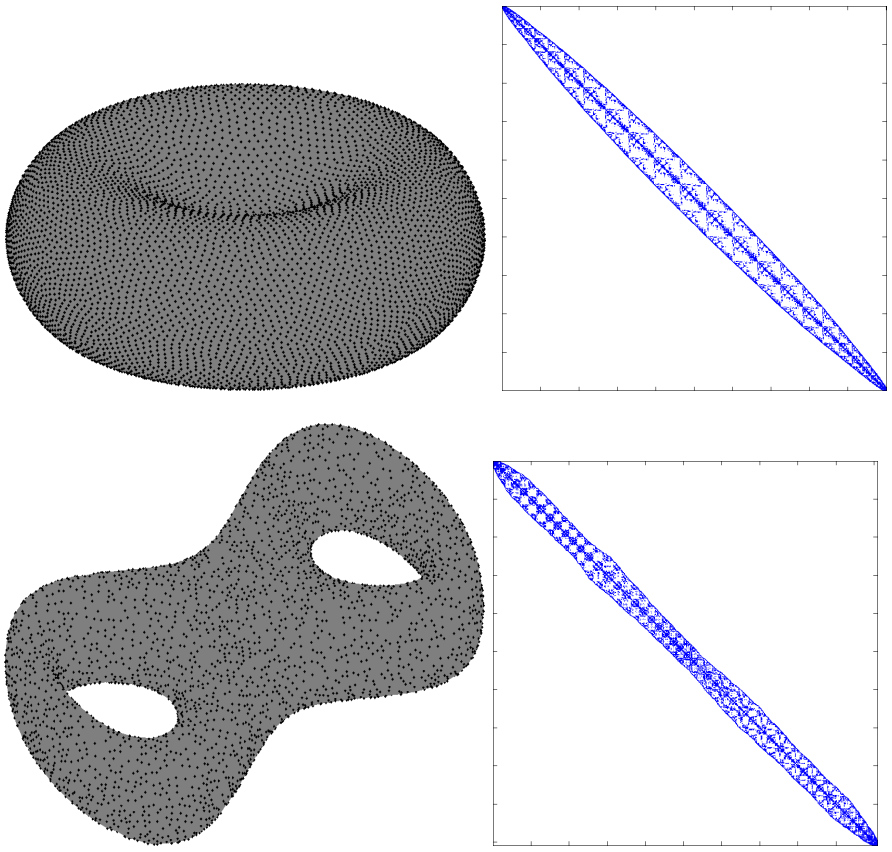

Figure 5.1: The figure on the top left shows maximum determinant nodes mapped from the sphere to the surface of the Red Blood Cell. The figure on the top right shows the reordered matrix L_X, obtained by applying the Reverse Cuthill-McKee reordering algorithm. 0.31% of the entries of the matrix are non-zeros for the Red Blood Cell. The figure on the bottom left shows a node set obtained on the double-torus. The figure on the bottom right shows the reordered matrix L_X, obtained by applying the Reverse Cuthill-McKee reordering algorithm. 0.62% of the entries of the matrix are nonzeros for the double-torus. We use a stencil size of $n = 31$ for both objects.

5.3.2 Method-of-lines

In Sections 5.5 and 5.6, we use the RBF-FD discrete approximation to the surface Laplacian in the method-of-lines (MOL) to simulate diffusion and reaction-diffusion equations on surfaces. We briefly review this technique for the former equation, as its generalization to the latter follows naturally.

The diffusion of a scalar quantity u on a surface with a (nonlinear) forcing term is given as

$$\frac{\partial u}{\partial t} = \delta \Delta_{\mathbb{M}} u + f(t, u), \tag{5.11}$$

where $\delta > 0$ is the diffusion coefficient, $f(t, u)$ is the forcing term, and an initial value of u at time $t = 0$ is given. Letting $X = \{\boldsymbol{X}_j\}_{j=1}^N \subset \mathbb{M}$ and $u_X \in \mathbb{R}^N$ denote the vector

containing the samples of u at the points in X, our RBF-FD method for (5.11) takes the form

$$\frac{d}{dt}u_X = \delta L_X u_X + f\left(t, u_X\right), \tag{5.12}$$

where L_X is an n-node RBF-FD differentiation matrix for approximating $\Delta_{\mathbb{M}}$ over the nodes in X, as described above. This is a (sparse) system of N coupled ODEs and, provided it is stable (see Section 5.4), can be advanced in time with a suitably chosen time-integration method. For an explicit time-integration method, L_X can be evaluated in $O(N)$ operations. For a method that treats the diffusion term implicitly, one can use an iterative solver such a $BICGSTAB$, or form the sparse upper and lower triangular factors obtained from the LU factorization of the implicit equations and use them for an efficient direct solver every time-step. These are the two respective approaches we use in our convergence studies in Section 5.5 and our applications in Section 5.6.

We conclude by noting that solving surface reaction-diffusion equations with an RBF-FD method was also considered in our paper [73] for the case of 1D surfaces embedded in $\mathbb{R}^2$. However, the approach used in that study for computing a discrete approximation to the surface Laplacian differs in an important way from the RBF-FD formulation of the surface Laplacian presented above. In that work, given a set of N nodes (X) on a surface, we start by using n-node RBF-FD formulas to construct differentiation matrices for the $\mathcal{G}^x$ and $\mathcal{G}^y$ over the node set X, which we denote by G_X^x and G_X^y. Next, the surface Laplacian was approximated from these matrices as $L_X = G_X^x G_X^x + G_X^y G_X^y$. As with the above approach, this formulation also avoids the need to compute derivatives of the normal vectors of the surfaces, but has the effect of doubling the bandwidth of the L_X compared to G_X^x and G_X^y. We tried extending this approach to two-dimensional surfaces in embedded in $\mathbb{R}^3$, but encountered stability issues when combining this with the method-of-lines, as the differentiation matrices L_X had eigenvalues with (sometimes large) positive real parts. The present method appears to be much less susceptible to these problems as discussed in the next section.

5.4 Shape Parameter and Eigenvalue Stability

A necessary condition for stability of the MOL approach described in the previous section is that the eigenvalues of the RBF-FD differentiation matrices L_X must be in the stability domain of the ODE solver used for advancing the system in time. As a minimum requirement, this will generally mean that all eigenvalues must, at the very least, be in the

left-half plane. The RBF-FD procedure does not guarantee that this property will hold for L_X, and it is possible to encounter situations in which this requirement is violated. In this section, we discuss a procedure related to choosing a stencil-dependent shape parameter ε_k when computing the RBF-FD weights that appears to ameliorate this issue and lead to L_X with eigenvalues in the left-half plane.

The idea is to choose a shape parameter $\varepsilon_k > 0$ for each stencil P_k that "induces" a particular target condition number κ_T for the RBF interpolation matrix on that stencil. In the previous section, we denoted this matrix by A_{P_k}, but now we denote it by $A_{P_k}(\varepsilon)$ since the entries of the matrix depend continuously on the shape parameter (see Equation (1.8)). The condition number of RBF interpolation matrices increase monotonically as the shape parameter decreases to zero (*cf.* [29]), so that the unique ε_k that induces the desired condition number κ_T is given as the zero of the function

$$F(\varepsilon, \kappa_T) = \log(\kappa(A_{P_k}(\varepsilon))/\kappa_T), \tag{5.13}$$

where $\kappa(A_{P_k})$ is the condition number of $A_{P_k}(\varepsilon)$ with respect to the two-norm. Since $A_{P_k}(\varepsilon)$ is symmetric, this is just the ratio of its largest singular value to its smallest. We view this process as a homogenization that compensates for irregularities in the node distribution. It is a generalization of the method from [18] for the surface of the sphere, where the nodes X are quasi-uniformly distributed so that one shape parameter gives roughly equal condition numbers amongst all the stencil interpolation matrices. In that study, the shape parameter is chosen to be proportional to $\sqrt{N}$, which keeps all the conditions number approximately equal as N grows.

We illustrate the effect of the proposed optimization process on the eigenvalues of the matrix approximation to the surface Laplacian L_X with two tests: one on a slightly distorted but somewhat regular set of nodes and one on a very irregular set of nodes.

For the first test, we start with the $N = 10000$ quasi-uniform Maximal Determinant (MD) node set for the unit sphere (obtained from [90]). We then map this point set to an idealized Red Blood Cell surface, which is biconcave in shape; see [36, Appendix B] for the analytical expression and the upper left picture in Figure 5.1 for a plot of these mapped nodes. While the MD points offer a quasi-uniform sampling of the sphere, they do not offer a good sampling when mapped to the Red Blood Cell (for a true quasi-uniform sampling of the latter, the correct procedure would be to solve an optimization problem and directly obtain MD points on the Red Blood Cell). Next, we form two RBF-FD matrix approximations to the surface Laplacian on the Red Blood Cell using $n = 31$ point stencils. The first approximation uses an optimized shape parameter on each stencil with the target

condition number set to $\kappa_T = 10^{12}$ in Equation (5.13). The second approximation uses a single shape parameter of $\varepsilon = 2.51$ across all stencils. This value is the mean of the shape parameters obtained in the first approximation. The eigenvalues of the corresponding differentiation matrices for these two procedures are shown in the top row of Figure 5.2, with the optimized ε per stencil on the left and the single ε on the right. We can see from the figure that optimized version produces eigenvalues all in the left-half plane, while the single-ε version results in one large positive eigenvalue. For the second test, we start

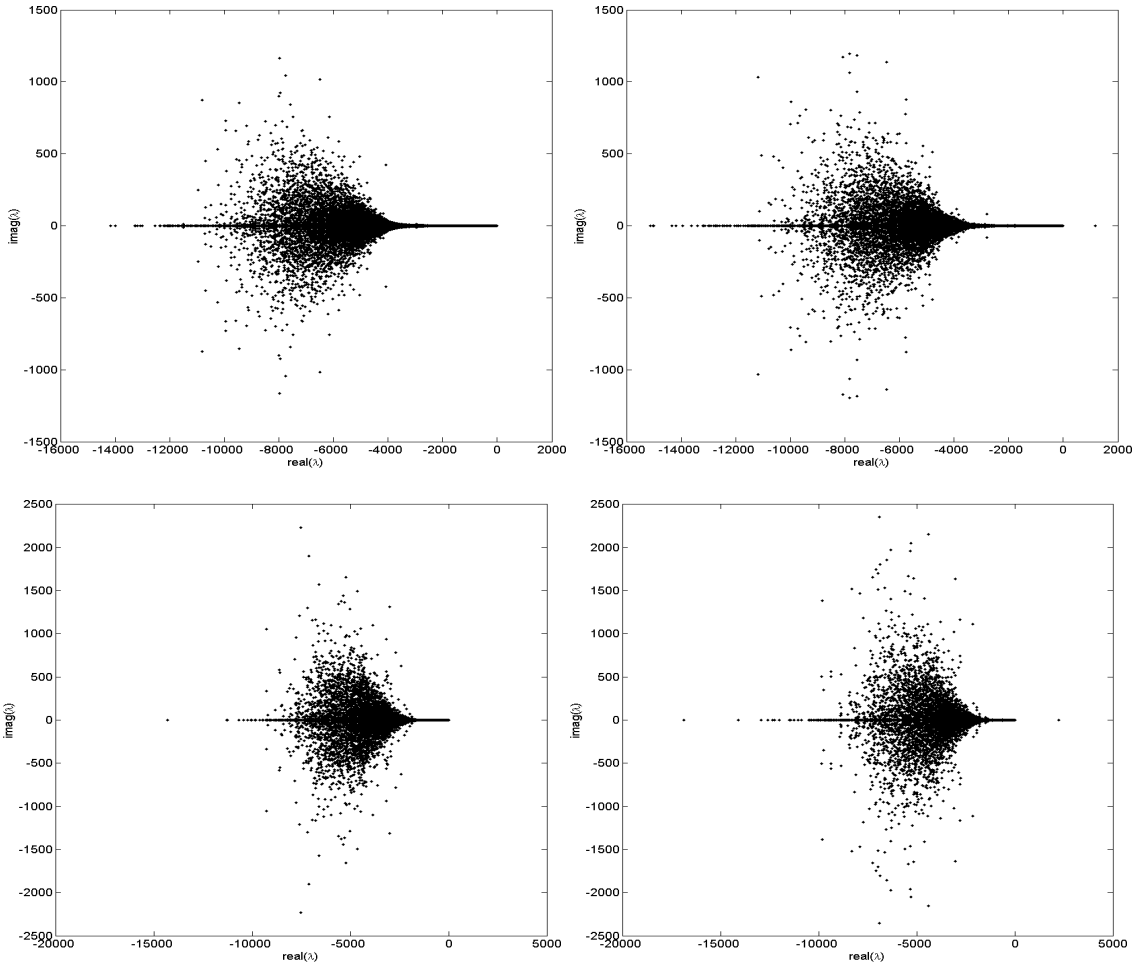

Figure 5.2: The figure on the left of the top row shows the eigenvalues of the $n = 31$ RBF-FD matrix L_X for the surface Laplacian on the Red Blood Cell using $N = 10000$ MD nodes mapped to the Red Blood Cell and the per-stencil shape parameter optimization strategy with $\kappa_T = 10^{12}$. The right figure on the top row is similar, but shows the eigenvalues of L_X using a single shape parameter of $\varepsilon = 2.51$, which is the mean of the shape parameters from L_X in the left figure. The figures on the bottom row are similar to the top, but show the eigenvalues of L_X for the double tours using $N = 5041$ scattered nodes. In the left one, the per-stencil shape parameter optimization strategy was used, while the one on the right used the mean of the shape parameters from the right which was $\varepsilon = 2.47$.

with an $N = 5041$ set of nodes on the double-torus that were obtained from the program 3D-XplorMath. This node set offers a fairly irregular sampling of the double-torus. For more on how the nodes were generated, see [36]. As on the Red Blood Cell, we form two RBF-FD matrix approximations to the surface Laplacian on the double-torus using $n = 31$ point stencils. The first approximation uses an optimized shape parameter on each stencil with the target condition number set to $\kappa_T = 10^{11}$ in Equation (5.13), the largest condition number that we could safely use on the irregular node set for $n = 31$ nodes (with larger condition numbers giving us eigenvalues with positive real parts). The second approximation uses a single shape parameter of $\varepsilon = 2.47$ across all stencils. Again, this value is the mean of the shape parameters obtained in the first approximation. The eigenvalues of the corresponding differentiation matrices are shown in the bottom row of Figure 5.2, with the optimized ε per stencil on the left and the single ε on the right. Again, we can see from the figure that the optimized version produces eigenvalues all in the left-half plane, while the single-ε version results in one large positive eigenvalue.

We note that it is possible to choose a single shape parameter in the above examples that is sufficiently large so that L_X have all eigenvalues in the left-half plane. However, this does not produce equally good results. The reason is that smaller shape parameters generally give better accuracy (*cf.* [91, 56]). Using the optimization procedure for selecting ε allows us to benefit from the accuracy afforded by smaller shape parameters where possible, as well as the stability afforded by larger shape parameters (when required by the irregularity of the node set). The trade-off in this procedure is that optimizing the shape parameter adds the cost of root-finding to the RBF-FD method. Additionally, fixing a target condition number across all stencils could mean that we end up choosing a lower condition number on some stencil than the condition number naturally dictated by the minimum width on that stencil. In this scenario, we are sacrificing some degree of local accuracy for the overall stability of the method. However, our tests did not reveal any impact of this on the convergence of the method.

While the shape parameter optimization procedure adds to the cost of our method, there are a few mitigating factors. First, we only solve the optimization problem to an absolute tolerance of 10^{-4}; this proved sufficient for the purpose of stability and achieving the target condition number. Second (and more important), since the optimization is done on a per-stencil basis and the stencil computations themselves are easily parallelized, the overall optimization procedure itself is also embarassingly parallel. These advantages are retained even if the surface sampled by the node set is evolving in time.

We conclude by noting that algorithms for the stable computation of RBF-FD matrices for all values of the shape parameters are available [56], but efforts to make these work for the case when the nodes are distributed on a lower dimensional surface embedded in $\mathbb{R}^d$ is still needed. Though successful outcomes in this area would mean that we could use larger target condition numbers in our method, this does not necessarily imply the obsolescence of our optimization procedure. Given that current methods to stably compute RBF interpolants in planar domains are currently at least $5 - 10$ times as costly as the standard RBF interpolation method, it is probable that new methods will have this drawback as well. In such a scenario, our shape parameter optimization procedure will likely allow for cost-efficient implementations of the RBF-FD method, allowing trade-offs between accuracy and computational cost.

5.5 Convergence Studies

We now present the results illustrating the convergence of our method for the (forced) diffusion equation given in Equation (5.11) on some standard surfaces. We present experiments with two different optimization strategies. First, we present results for studies where we fix the condition number across all stencils for a given N, but allow the target condition number to grow with increasing N. Then, we present results for studies involving fixing the condition number for increasing N (equivalent to increasing the shape parameter for increasing N). In the latter case, we run into saturation errors [15] due to the employment of stationary interpolation.

We use the Backward Difference Formula of Order 4 (BDF4) for all tests and set the time-step to $\Delta t = 10^{-4}$, a time-step that allows the spatial errors to dominate the temporal error. We use $BICGSTAB$ to solve the implicit system arising from the BDF4 discretization; we noticed that the solver needed at most three iterations per time-step to converge to a relative tolerance of 10^{-12}.

For convenience, we measure errors using the ℓ_2 and ℓ_∞ norms rather than approximations to the continuous versions of these norms on the test surfaces. The convergence of our method is a function of the fill distance h_X, defined as the radius of the largest ball that is completely contained on the manifold which does not contain a node in X. For quasi-uniformly distributed nodes on our test surfaces, we expect that $h \propto \frac{1}{\sqrt{N}}$, where N is the total number of nodes on the surface. In the following subsections, we therefore examine convergence as a function of $\sqrt{N}$.

5.5.1 Convergence Studies with Increasing Condition Number

In this section, we present the results of numerical convergence studies where the uniform condition number across the RBF-FD matrices is allowed to grow as the number of points (N) on the surface increases. In the absence of the shape parameter optimization procedure, this would be equivalent to fixing the shape parameter while increasing N, the simplest approach to take with RBF interpolation.

First, we examine the convergence of our MOL formulation by approximating the diffusion equation on a sphere. Then, we examine the convergence of our method on simulating two forced diffusion equations, one on the sphere and one on a torus. We present results for different stencil sizes n and examine convergence as the total number of nodes N increases.

5.5.1.1 Diffusion on the Sphere

This test problem was presented in [60], and involves solving the heat equation on a unit sphere $\mathbb{S}^2$. The exact solution to this problem is given as a series of spherical harmonics $u(t,\theta,\phi) = \frac{20}{3\pi}\sum_{l=1}^{\infty} e^{-l^2/9}e^{-tl(l+1)}Y_{ll}(\theta,\phi)$, where θ and ϕ are longitude and latitude, respectively, and Y_{lm} is the degree l order m real spherical harmonic. Since the coefficients decay rapidly, the series is truncated after 30 terms. As in [60], we evolve the PDE until $t = 0.5$, using the exact solutions to boot-strap our BDF4 scheme. We test the RBF-FD method for $n = 11, 17$, and 31 for $N = 642, 2562, 10242$, and 40962 icosahedral points on the sphere [2], and plot the relative error in the numerical solutions. For all tests, we start with a target condition number of $\kappa_T = 10^5$ and allow it to grow with increasing N to $\kappa_T = 10^{18}$. The results of this study are shown in Figure 5.3.

In addition to the errors, Figure 5.3 shows dashed lines corresponding to ideal p-order convergence, where $p = 2, 3, 4, 5$. It is clear that our method gives convergence between orders two and three for $n = 11$, close to order four for $n = 17$, and slightly higher than order five for $n = 31$, both in the ℓ_2 and the ℓ_∞ norms. Our method achieves similar results for smaller values of N than were used by the Closest Point method in [60], as is to be expected from a method that uses points only in the embedded space $\mathbb{S}^2$. However, it is important to be cautious when comparing errors against the Closest Point method. The values of N given in the results in [60] are greater than the actual number of points used in that work to compute approximations to the Laplace-Beltrami operator. The values of N in that work correspond to all the points used in the embedding space $\mathbb{R}^3$.

We note that the RBF-FD weights for $n = 31$ and $N = 40962$ were computed in quad-

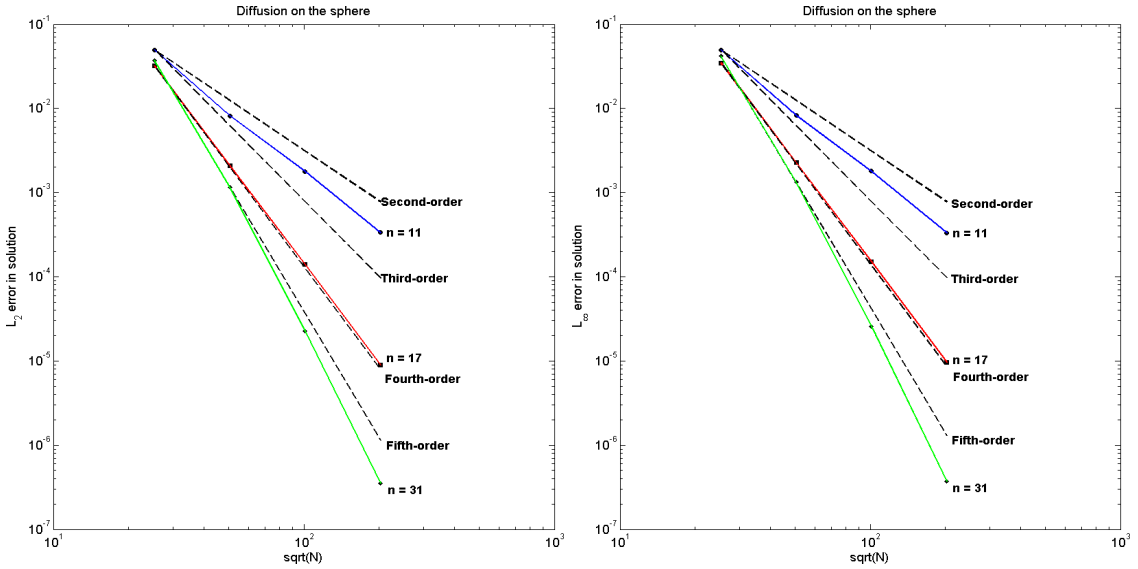

Figure 5.3: The figure on the left shows the ℓ_2 error in the numerical solution to the diffusion equation on the sphere as a function of $\sqrt{N}$, while the figure on the right shows the ℓ_∞ error. Both figures use a log-log scale, with colored lines indicating the errors in our method and dashed lines showing ideal p-order convergence for $p = 2, 3, 4, 5$. All errors were measured against the exact solution. The errors for $n = 31$ and $N = 40962$ were computed in quad-precision.

precision, though the simulations that used the weights were only run in double-precision. This is because our approach of allowing the condition number to grow with N leads to condition numbers of 10^{18} for very high N and n, which correspond to nearly-singular or singular matrices in double-precision. A possible way of remedying this is to start with a smaller target condition number for $N = 642$. Of course, this will lead to a higher error for each N, but can help offset the ill-conditioning for very large N and n. Later in this section, we will present an alternative way of ameliorating this issue.

5.5.1.2 Forced Diffusion on the Sphere

This problem was first presented in [6], and used in [36] as well. For this test, we manufacture a solution to the diffusion equation on the sphere, with the forcing term $f(t, u)$ in Equation (5.11) chosen so that this solution is maintain for all time. The manufactured solution is given by

$$u(t, \boldsymbol{X}) = e^{-5t} \sum_{k=1}^{23} e^{-10 \cos^{-1}(\xi_k \cdot \boldsymbol{X})}, \tag{5.14}$$

where $\xi_k, k = 1, \ldots, 23$ are randomly placed points on the surface of the sphere. The solution is $C^\infty(\mathbb{S}^2)$. As in [36], we compute the forcing function analytically and evalute it implicitly

in time. We compare errors in the numerical solution of the forced diffusion equation at time $t = 0.2$ for different values of N and n. Again, we use $N = 642, 2562, 10242$, and 40962 icoshedral points on the sphere, with $n = 11, 17$, and 31 points in each of the N stencils on the surface. The random placement of Gaussian centers makes this a more difficult test than diffusion of a spherical harmonic. The results are shown in Figure 5.4.

Again, the figure shows dashed lines corresponding to ideal p-order convergence, where $p = 2, 3, 4, 5$. For this test, our method gives convergence of order two for $n = 11$, close to order four for $n = 17$, and close to order five for $n = 31$, both in the ℓ_2 and the ℓ_∞ norms, which are similar to the previous experiment. Again, the weights for $n = 31$ and $N = 40962$ were computed in quad-precision, for the same reasons as before.

5.5.1.3 Forced Diffusion on a Torus

This test is similar to the test involving randomly placed Gaussians on the sphere, except that this procedure is done on a torus. We consider the torus given by the implicit equation:

$$\mathbb{T}^2 = \left\{ \boldsymbol{X} = (x, y, z) \in \mathbb{R}^3 \;\middle|\; \left(1 - \sqrt{x^2 + y^2}\right)^2 + z^2 - \frac{1}{9} = 0 \right\},$$

which can be parameterized using intrinsic coordinates φ and λ as follows:

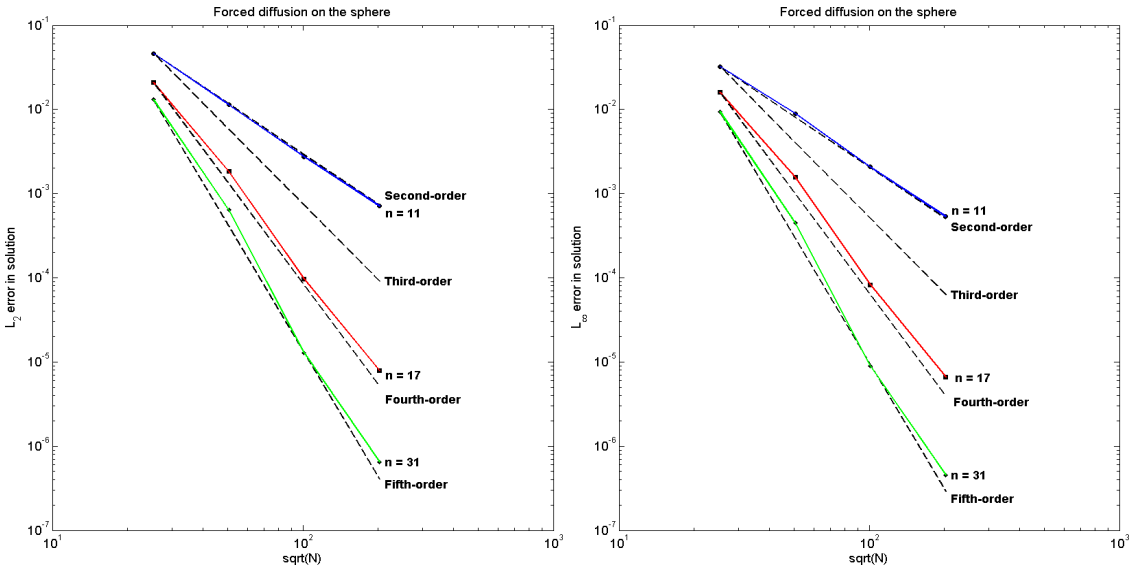

Figure 5.4: The figure on the left shows the ℓ_2 error in the numerical solution to the forced diffusion equation on the sphere given by Equation (5.14) as a function of $\sqrt{N}$, while the figure on the right shows the ℓ_∞ error. Both figures use a log-log scale, with colored lines indicating the errors in our method and dashed lines showing ideal p-order convergence for $p = 2, 3, 4, 5$. All errors were measured against the solution given by Equation (5.14). The errors for $n = 31$ and $N = 40962$ were computed in quad-precision.

$$x = \left(1 + \frac{1}{3}\cos(\varphi)\right)\cos(\lambda),\ y = \left(1 + \frac{1}{3}\cos(\varphi)\right)\sin(\lambda),\ z = \frac{1}{3}\sin(\varphi), \tag{5.15}$$

where $-\pi \leq \varphi, \lambda \leq \pi$. The surface Laplacian of a scalar function $f : \mathbb{T}^2 \rightarrow \mathbb{R}$ in this intrinsic coordinate system is given as

$$\Delta_{\mathbb{M}} f(\varphi, \lambda) = \frac{1}{\left(1 + \frac{1}{3}\cos(\varphi)\right)^2}\frac{\partial^2 f}{\partial \lambda^2} + \frac{9}{\left(1 + \frac{1}{3}\cos(\varphi)\right)}\frac{\partial}{\partial \varphi}\left(\left(1 + \frac{1}{3}\cos(\varphi)\right)\frac{\partial f}{\partial \varphi}\right).$$

The manufactured solution to the diffusion equation (given by Equation (5.11) with $\mathbb{M} = \mathbb{T}^2$) is

$$u(t, \varphi, \lambda) = e^{-5t}\sum_{k=1}^{23} e^{-a^2(1-\cos(\lambda-\lambda_k))-b^2(1-\cos(\varphi-\varphi_k))}, \tag{5.16}$$

where $a = 9$, $b = 3$, and (φ_k, λ_k) are randomly chosen values in $[-\pi, \pi]^2$. The solution is $C^\infty(\mathbb{T}^2)$ and a visualization at $t = 0$ is given in Figure 5.5 (left). While the solution and forcing function are all specified using intrinsic coordinates, the RBF-FD method uses only extrinsic (Cartesian coordinates) without requiring knowledge of the underlying intrinsic coordinate system. As with the forced diffusion test on the sphere, we compute the forcing function corresponding to Equation (5.16) analytically and evaluate it implicitly. We similarly compare errors in the numerical solution of the forced diffusion equation at time $t = 0.2$ for stencils of size $n = 11, 17$ and 31 nodes. The node sets we use for experiments on the torus are generated from a "staggered" grid in intrinsic variable space, and are determined as follows:

1. Given m, choose $m + 1$ equally spaced angles on $[-\pi, \pi]$ in φ and $3m + 1$ equally spaced angles on $[-\pi, \pi]$ in λ.

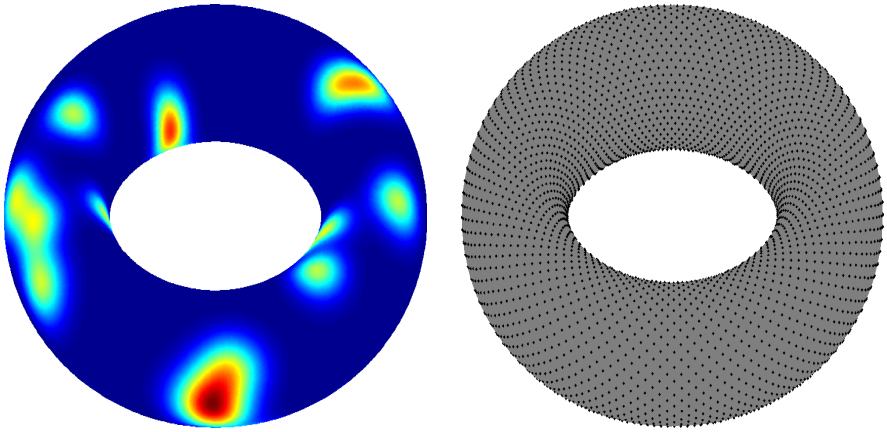

Figure 5.5: Forced diffusion on a torus. The figure on the left shows the intial condition for the forced diffusion problem on the torus given by Equation (5.15). The figure on the right shows a node set containing $N = 5400$ quasi-uniformly spaced nodes on the same torus.

2. Disregard the values of φ and λ at π and take a direct product of the remaining points to obtain $N = 3m^2$ points on $[-\pi, \pi)^2$. Map these points to $\mathbb{T}^2$ using Equation (5.15) and call the set of nodes X_1.
3. Next, generate another set of $N = 3m^2$ gridded points in $[-\pi, \pi)^2$ from the previous set by offsetting the φ coordinate by π/m and the λ coordinate by $\pi/(3m)$, so they lie at the midpoints of the previous gridded values. Map these to $\mathbb{T}^2$ and call the set of nodes X_2.
4. The final set of nodes is given by $X = X_1 \cup X_2$.

In the experiments, we use $m = 10, 20, 30, 60, 80$, corresponding to node sets of size $N = 600, 2400, 5400, 21600, 38400$. A plot of the nodes for $N = 5400$ is shown in Figure 5.5 (right). These points remain more or less uniformly spaced on the torus as N grows.

The results for the experiments are shown in Figure 5.6. Again, the figure shows dashed lines corresponding to ideal p-order convergence, where $p = 2, 3, 4, 6$. On this test, our method gives convergence of order two for $n = 11$, close to order four for $n = 17$, and between orders five and six for $n = 31$, both in the ℓ_2 and the ℓ_∞ norms. The convergence rates are comparable to the results seen for diffusion on the sphere, and slightly better than those seen for the forced diffusion problem on the sphere. The results for $n = 31$ and

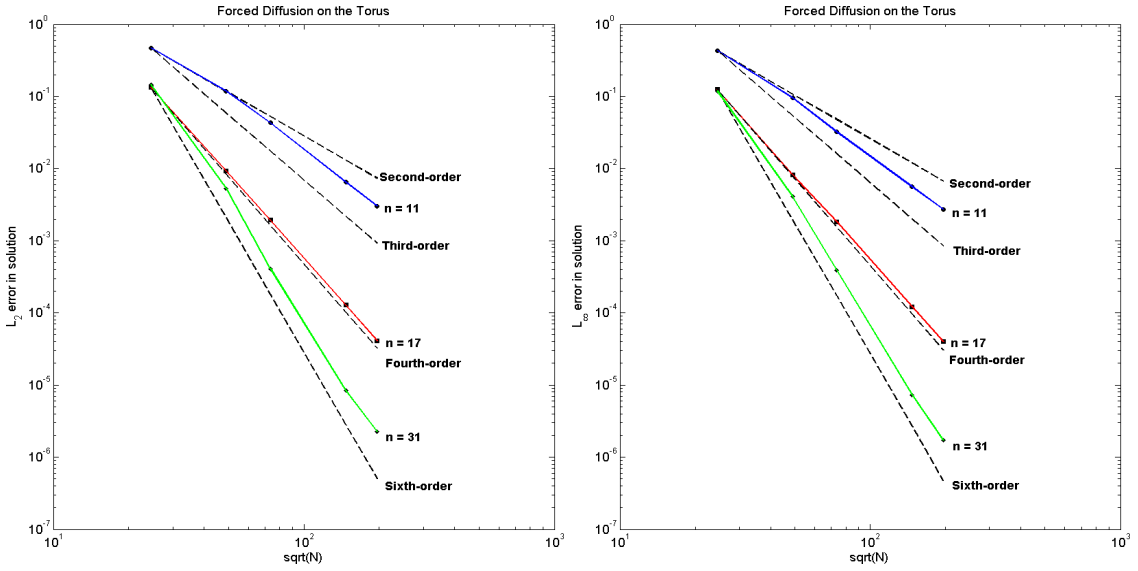

Figure 5.6: The figure on the left shows the ℓ_2 error in the numerical solution to the forced diffusion equation on the torus given by Equation (5.15) as a function of $\sqrt{N}$, while the figure on the right shows the ℓ_∞ error. Both figures use a log-log scale, with colored lines indicating the errors in our method and dashed lines showing ideal p-order convergence for $p = 2, 3, 4, 6$. All errors were measured against the solution given by Equation (5.16). The errors for $n = 31$ and $N = 38400$ were computed in quad-precision.

$N = 38400$ were computed in quad-precision, for the same reasons as before.

5.5.2 Convergence Studies with Fixed Condition Number

In Section 5.5.1, we saw that allowing the condition number to grow as N increases by fixing the mean shape parameter can give excellent results, but will eventually cause the RBF interpolation matrices to be ill-conditioning for large N and n.

In this section, we present an alternate approach. We choose to fix the target condition number at a particular (reasonably large) value for all values of N and n. As N increases, this has the effect of increasing the value of the average shape parameter. Our goal here is to understand the relationship between the magnitude of the target condition number and the value of n and N at which saturation errors can set in. This would also give us intuition on the connection between the target condition number and the order of convergence of our method.

With this in mind, we present the results of numerical convergence studies of forced diffusion on a sphere and on a torus for two fixed condition numbers, $\kappa_T = 10^{14}$ and $\kappa_T = 10^{20}$, for increasing values of N and n. For $\kappa_T = 10^{20}$, the RBF-FD weights on each patch were run in quad-precision using the Advanpix Multicomputing Toolbox. However, once the weights were obtained, they were converted back to double-precision and the simulations were carried out only in double-precision.

We test the behavior of our MOL formulation for forced diffusion on the sphere and for forced diffusion on the torus, with the solution to the former given by Equation (5.14) and the solution to the latter given by Equation (5.16). We again use a BDF4 method with $\Delta t = 10^{-4}$, with the forcing term computed analytically and evaluated implicitly. The errors in the ℓ_2 and ℓ_∞ measured at $t = 0.2$ are shown in Figures 5.7–5.8 for the sphere, and in Figures 5.9–5.10 for the torus.

5.5.2.1 Forced Diffusion on the Sphere

The results for $\kappa_T = 10^{14}$ are shown in Figure 5.7, and those for $\kappa_T = 10^{20}$ are shown in Figure 5.8. Figure 5.7 shows that fixing the target condition number at $\kappa_T = 10^{14}$ produces no saturation errors in the ℓ_2 or ℓ_∞ norms for $n = 11$. In fact, this value of κ_T appears to produce saturation in the ℓ_2 norm only for large values of N for $n = 17$. We see similar saturation errors in the ℓ_∞ norm. For $n = 31$, we see saturation for $N > 2562$; this is a clear indication that generating high-order RBF-FD methods requires the ability to use large target condition numbers (small values of the shape parameter ε).

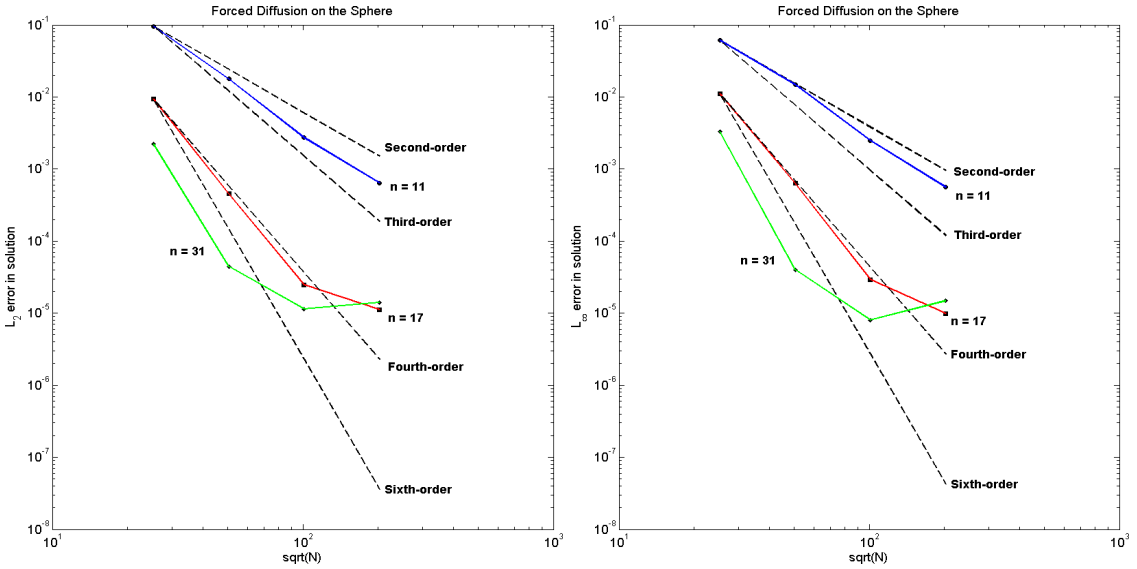

Figure 5.7: The figure on the left shows the ℓ_2 error in the numerical solution to the forced diffusion equation on the sphere given by Equation (5.14) as a function of $\sqrt{N}$, while the figure on the right shows the ℓ_∞ error. Both figures use a log-log scale, with colored lines indicating the errors in our method and dashed lines showing ideal p-order convergence for $p = 2, 3, 4, 6$. The target condition number was set to $\kappa_T = 10^{14}$.

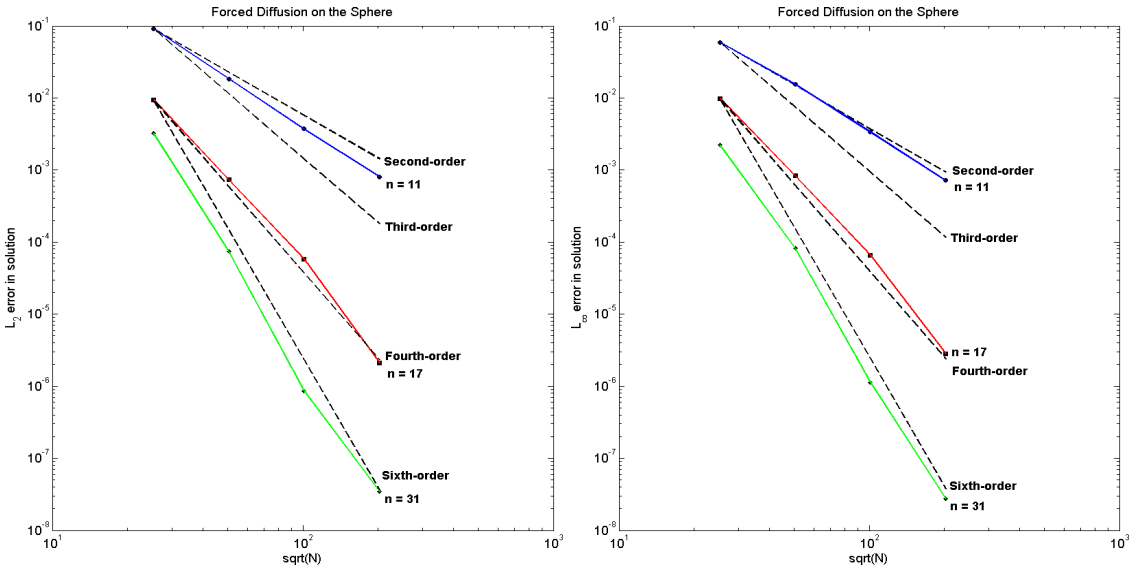

Figure 5.8: The figure on the left shows the ℓ_2 error in the numerical solution to the forced diffusion equation on the sphere given by Equation (5.14) as a function of $\sqrt{N}$, while the figure on the right shows the ℓ_∞ error. Both figures use a log-log scale, with colored lines indicating the errors in our method and dashed lines showing ideal p-order convergence for $p = 2, 3, 4, 6$. The target condition number was set to $\kappa_T = 10^{20}$ and all RBF-FD weights were computed in quad-precision.

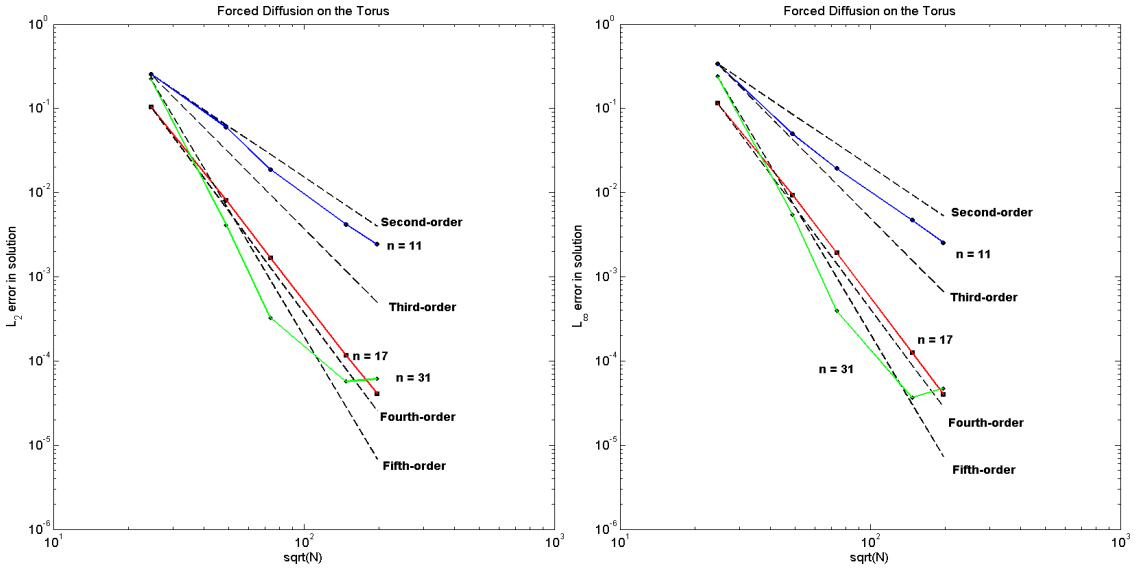

Figure 5.9: The figure on the left shows the ℓ_2 error in the numerical solution to the forced diffusion equation on the torus given by Equation (5.15) as a function of $\sqrt{N}$, while the figure on the right shows the ℓ_∞ error. Both figures use a log-log scale, with colored lines indicating the errors in our method and dashed lines showing ideal p-order convergence for $p = 2, 3, 4, 5$. The target condition number was set to $\kappa_T = 10^{14}$.

It is clear from Figure 5.8 that there are no saturation errors for $n = 11$ and $n = 17$ for the values of N tested when $\kappa_T = 10^{20}$. In addition, for $n = 31$, we see convergence that is close to order six with no saturation errors for the values of N used. This confirms that using small shape parameters within the RBF-FD method can give higher convergence. This is further motivation for the development of algorithms to stably compute the RBF interpolation matrices as $\varepsilon \to 0$.

5.5.2.2 Forced Diffusion on a Torus

The results for $\kappa_T = 10^{14}$ are shown in Figure 5.9, and those for $\kappa_T = 10^{20}$ are shown in Figure 5.10. Figure 5.9 shows that fixing the target condition number at $\kappa_T = 10^{14}$ produces no saturation errors in the ℓ_2 or ℓ_∞ norms for $n = 11$ or $n = 17$ in either norm, in contrast to forced diffusion on the sphere. Indeed, $\kappa_T = 10^{14}$ seems sufficient for methods up to order 4 for the values of N tested. However, for $n = 31$, we see saturation errors for $N > 5400$, again showing that larger target condition numbers are required for high-order RBF-FD methods.

Figure 5.10 shows that convergence on the torus is a bit more erratic than on the sphere for the forced diffusion problem when $\kappa_T = 10^{20}$. While there are no saturation errors for $n = 11$ and $n = 17$ for the values of N tested, the convergence is slightly lower

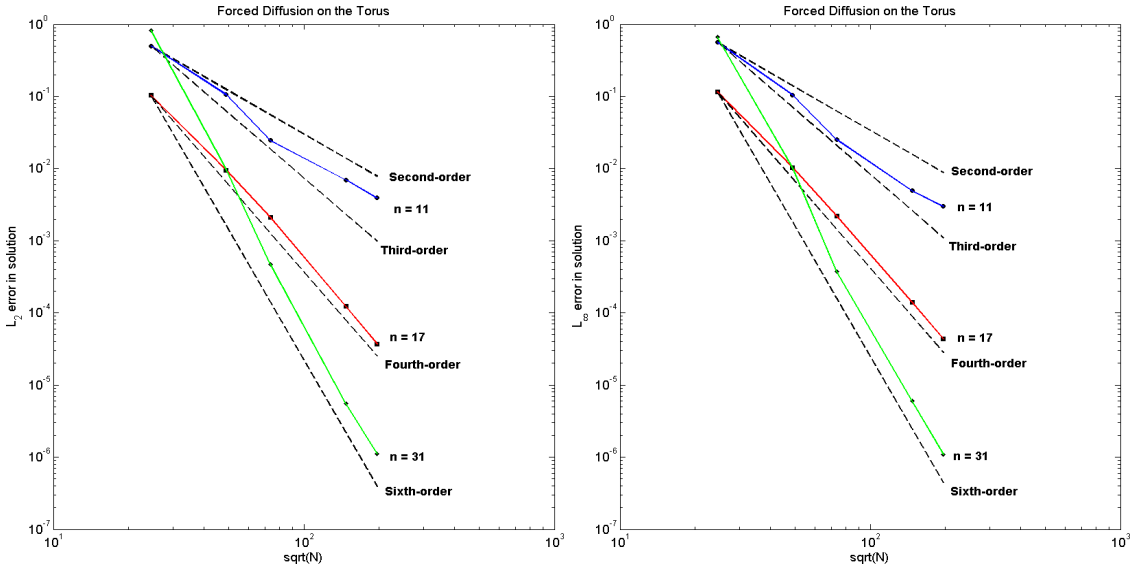

Figure 5.10: The figure on the left shows the ℓ_2 error in the numerical solution to the forced diffusion equation on the torus given by Equation (5.15) as a function of $\sqrt{N}$, while the figure on the right shows the ℓ_∞ error. Both figures use a log-log scale, with colored lines indicating the errors in our method and dashed lines showing ideal p-order convergence for $p = 2, 3, 4, 6$. The target condition number was set to $\kappa_T = 10^{20}$ and all RBF-FD weights were computed in quad-precision.

for smaller values of N, possibly indicating that the node set on the torus is not quite as uniformly-spaced as the one on the sphere for those values of N. However, as N is increased, the rate of convergence seems to be slightly better than what was seen on the sphere. This becomes apparent when looking at the line for $n = 31$. We see large errors for $N = 600$, but a rapid fall-off as N is increased, with the overall order of convergence for $n = 31$ being between five and six.

5.6 Application: Turing Patterns

This section presents an application of our RBF-FD method to solving a two-species Turing system (two coupled reaction-diffusion equations) on different surfaces. We present two types of results, the first for surfaces where parameterizations or implicit equations describing the surface are known, and the second where they are not.

To facilitate comparison, we use the Turing system first described for the surface of the sphere in [87] and applied to more general surfaces in [36]. The system describes the interaction of an activator u and inhibitor v according to

$$\frac{\partial u}{\partial t} = \alpha u(1-\tau_1 v^2) + v(1-\tau_2 u) + \delta_u \Delta_{\mathbb{M}} u, \tag{5.17}$$

$$\frac{\partial v}{\partial t} = \beta v\left(1 + \frac{\alpha\tau_1}{\beta} uv\right) + u(\gamma + \tau_2 v) + \delta_v \Delta_{\mathbb{M}} v. \tag{5.18}$$

If $\alpha = -\gamma$, then $(u, v) = (0, 0)$ is a unique equilibrium point of this system. Altering the diffusivity rates of u and v can lead to instabilities which manifest as pattern formations. The coupling parameter τ_1 favors stripe formations, while τ_2 favors spots. Stripe formations take much longer to attain "steady-state" than spot formations. In the following subsections, We use the Semi-implicit Backward Difference Formula of order 2 (SBDF2) as the time-stepping scheme, and set the time-step to $\Delta t = 0.01$ for all tests. Since the diffusion terms are handled implicitly, the RBF-FD matrix needs to be inverted every time-step. We accomplish this by precomputing a sparse LU decomposition of the matrix, and using the triangular factors for forward and back solves every time-step. The values for all parameters for Equations (5.17) and (5.18), including final times for simulations, are presented in Table 5.1. The parameters used in the RBF-FD discretizations of Equations (5.17) and (5.18) on different surfaces are shown in Table 5.2.

Table 5.1: Parameters of Equations (5.17) and (5.18) used in the numerical experiments shown in Figures 5.11 and 5.12. In all cases, we set $\delta_u = 0.516\delta_v$. The last column shows the final time t_{final} to which each of the simulations was run.

Surface/Pattern	δ_v	α	β	γ	τ_1	τ_2	t_{final}
RBC/spots	4.5×10^{-3}	0.899	-0.91	-0.899	0.02	0.2	800
RBC/stripes	2.1×10^{-3}	0.899	-0.91	-0.899	3.5	0	6500
Bumpy sphere/spots	4.5×10^{-3}	0.899	-0.91	-0.899	0.02	0.2	800
Bumpy sphere/stripes	2.1×10^{-3}	0.899	-0.91	-0.899	3.5	0	7000
Double-torus/spots	2.1×10^{-3}	0.899	-0.91	-0.899	0.02	0.2	700
Double-torus/stripes	8.87×10^{-4}	0.899	-0.91	-0.899	3.5	0	6000
Frog/spots	2.87×10^{-4}	0.899	-0.91	-0.899	0.02	0.2	600
Bunny/stripes	2.87×10^{-4}	0.899	-0.91	-0.899	3.5	0	6000

Table 5.2: Parameters used in the RBF-FD discretization of Equations (5.17) and (5.18) for the numerical experiments shown in Figures 5.11 and 5.12. In all cases, the time-step was set to $\Delta t = 0.01$.

Surface	Number of nodes (N)	Stencil size (n)	Target Cond. No. (κ_T)
RBC	10000	31	10^{12}
Bumpy sphere	10000	31	10^{12}
Double-torus	12100	31	10^{11}
Frog	7458	31	10^{10}
Bunny	11339	31	10^{10}

5.6.1 Turing Patterns on Manifolds

We first solve the Turing system on three surfaces: the Red Blood Cell (RBC) and the double-torus described earlier, and on the Bumpy Sphere detailed in [36]. RBCs are biconcave surfaces and can be represented parametrically, as described earlier. The Bumpy Sphere is a point set downloaded from an online repository, and equipped with point unit normals by parametric interpolation with the RBF parametric model presented in [72]. Also, since the downloaded model had $N = 5256$ vertices and we wished to demonstrate the viability of the RBF-FD method on far more vertices, we sampled our parametric model to generate $N = 10000$ vertices, and solve the Turing system on that point set. The first three rows of Table 5.2 list all the parameters used in the RBF-FD discretizations of Equations (5.17) and (5.18) on each of these three surfaces, and the last column of Table 5.1 lists the final times used for these simulations. The results of these simulations are shown in Figure 5.11. The spot and stripe patterns are qualitatively similar to those shown in [36].

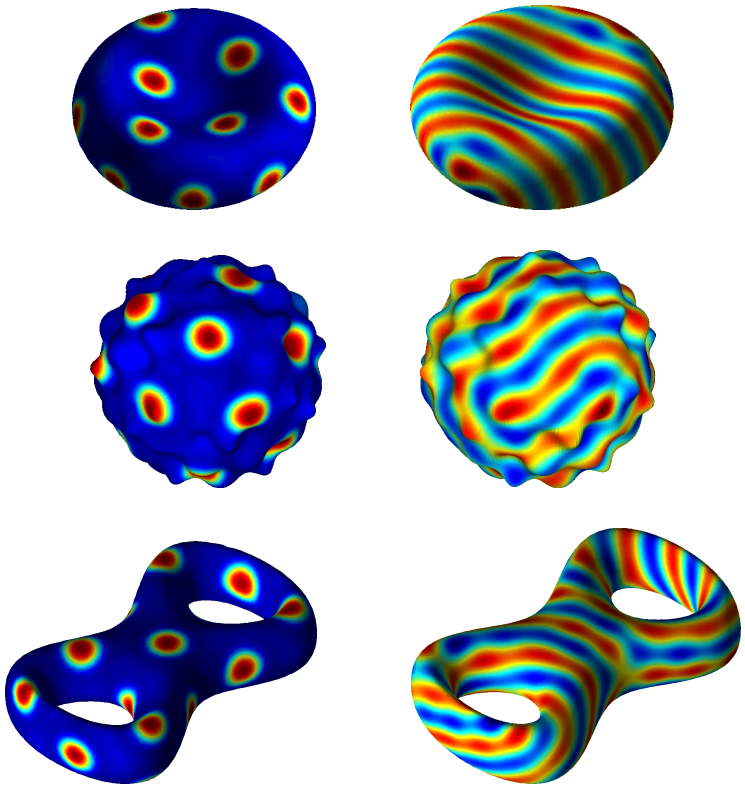

Figure 5.11: Steady Turing spots and stripe patterns resulting from solving Equations (5.17) and (5.18) on the Red Blood Cell, the Bumpy Sphere, and the double-torus. In all plots, red and blue correspond to a high and low concentration of u, respectively.

5.6.2 Turing Patterns on More General Surfaces

We now turn our attention to more general point sets: a Frog model (obtained from the AIM@SHAPE Shape Repository) and the Stanford Bunny model (obtained from the Stanford 3D Scanning Repository). Rather than as point clouds, these models are available in the form of meshes and approximate normal data. In contrast to the previous two examples, it is not clear if the surfaces represented by these meshes can be analytically parametrized. To prepare these point sets for simulations, we first run the Poisson surface reconstruction algorithm [52] to generate a water-tight implicit surface that fits the point cloud. This algorithm requires both the point cloud and the approximate normals as input. Forming an implicit surface smoothes the approximate normal vectors input into the Poisson surface reconstruction, resulting in a more stable RBF-FD discretization. Having generated an implicit surface, we sample that with the Poisson disk sampling algorithm to generate a point cloud with the desired number of points. The other rationale for employing Poisson disk sampling is that while the Poisson surface reconstruction will fix any holes in the mesh, those former holes may not be sufficiently sampled. This preprocessing was performed entirely in MeshLab [10].

After this preprocessing, we run a Turing spot simulation on the Frog model, and a Turing stripe simulation on the Stanford Bunny. The last two rows of Table 5.2 list all the parameters used in the RBF-FD discretizations of Equations (5.17) and (5.18) on each of these surfaces, and the last column of Table 5.1 lists the final times used for these simulations. The results are shown in Figure 5.12. Before the color-mapping for aesthetics, the results are qualitatively similar to those shown in Figure 5.11.

5.7 Summary

In this chapter, we introduced a new numerical method based on RBF-generated Finite Differences (RBF-FD) for computing a discrete approximation to the Laplace-Beltrami operator on surfaces of codimension one embedded in $\mathbb{R}^3$. The method uses scattered nodes on the surface, without requiring expansion into the embedding space. We improved on the method presented in Chapter 4, designing a stable numerical method that does not stabilization with artificial viscosity (a feature of most RBF-FD methods). This development was facilitated by an algorithm to optimize the shape parameter for each interpolation patch on the surface. We demonstrated that this optimization procedure can compensate for irregularities in the sampling of the surface. We then presented error and convergence estimates for our method using two approaches: allowing the condition number to grow

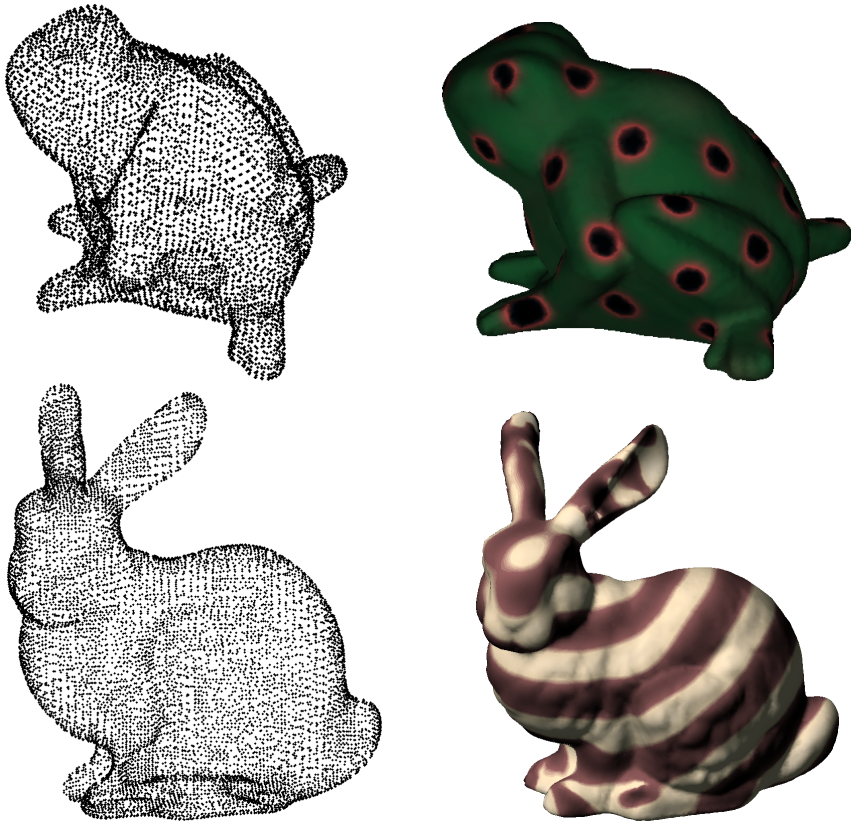

Figure 5.12: The figure on the top right shows a Turing spot pattern on a Frog model. Green corresponds to a low concentration, and brown and black to higher concentrations. The figure on the bottom right shows a Turing stripe pattern on the Stanford Bunny model. Here, the lightest browns (almost white) correspond to low concentrations, and darker browns correspond to higher concentrations. Both the figures on the left show the point clouds used for the solution of the Turing system.

with the number of points on the surface, and fixing the condition number for an increasing number of points. We discussed the trade-offs inherent in each approach, and provided intuition as to the relationship between the condition number, shape parameter, and the order of convergence of our method on the diffusion equation on a sphere and a torus. We presented an application of our method to simulating reaction-diffusion equations on surfaces; specifically, we demonstrated the solution of Turing PDEs on several interesting shapes, both parametrizable and more general.

While our method currently works for static objects, our goal is to apply RBF-FD to the solution of PDEs on evolving surfaces, with the evolution dictated by the interaction of a fluid with the object as in Chapter 3. This will require efficient kd-tree implementations, including algorithms for dynamically updating and/or rebalancing the kd-tree as the point

set evolves. The method will need to be parallelized to be efficient. Another issue with the method is its ability to handle thin features on surfaces. For RBF-FD to be competent on more general surfaces, it will be necessary to combine our method with an adaptive refinement code that detects thin features and samples sides of the feature sufficiently (the alternative would be to find an efficient way to measure distances along arbitrary surfaces).

A natural extension of this work would be to adapt the method to handle spatially-variable (possibly anisotropic) diffusion. While this extension is not conceptually difficult, the realization of this extension would make our method even more useful for biological applications, like the simulation of gels or viscoelastic materials on surfaces. We briefly discuss this idea for 1D surfaces in Chapter 6 and present some preliminary results there. We intend to fully address this problem in a follow-up study. Finally, it would be interesting to apply the method to solving PDEs on problems with boundary conditions. We discuss this briefly in the 1D surface context in Chapter 6, but intend to fully study the problem in a follow-up study as well.

CHAPTER 6

SUMMARY AND FUTURE WORK

6.1 Summary

Before proceeding to describe future lines of research, we summarize the contributions of this dissertation that substantiate the thesis that RBF interpolation forms an excellent foundation for the development of numerical methods in the complex geometries and rheologies arising from the study of biological systems.

In Chapter 1, we reviewed the state of the art for fluid-structure interaction, with a focus on the Immersed Boundary (IB) method and the Augmented Forcing method (AFM). We presented an overview of RBF interpolation, laying the foundation for its use throughout this dissertation. We also discussed symmetric Hermite interpolation, an extension of Hermite interpolation to handle scattered data and independent linear functionals. We then described the state of the art in RBF-based numerical methods for the solution of PDEs, focusing on RBF-PS, RBF-FD, and RBF-PUM.

In Chapter 2, we developed a parametric framework for modeling platelets. We then used three different interpolants within that framework: RBF interpolants restricted to the circle/sphere, Fourier interpolants (Fourier series in 2D, spherical harmonics in 3D), and piecewise quadratic interpolants. We compared these three interpolation strategies on two types of target shapes generated from perturbations of ideal shapes: an infinitely-smooth target shape generated from a sharp perturbation of an ellipse (ellipsoid in 3D), and a rough target shape generated from a small perturbation of a cirlce (sphere in 3D). Specifically, we compared these interpolants in terms of accuracy, convergence, and computational cost. In preparation for Chapter 3, we also computed tension forces on these shapes that were similar to those used within the IB method. We concluded that interpolation with RBFs offered the right balance between robustness, accuracy, and computational cost, by having costs similar to those of Fourier methods and greater accuracy and convergence when interpolating rough target functions (with identical accuracy on smooth target functions).

Having selected the RBF geometric model, we went on to augment the IB method with RBFs for the modeling of platelets in the context of hemodynamic flows in Chapter 3.

We compared the convergence and accuracy of this new RBF-IB method to that of the traditional IB method on a standard test problem (using an RK2 time-stepping scheme), and concluded that the RBF-IB method had greater accuracy and similar convergence rates to the traditional IB method. We also tested the area loss in both methods and determined that the RBF-IB method had lower area loss on coarse grids; in addition, we tested the time-step restrictions on both methods, and determined that the RBF-IB method allowed larger time-steps on the same grid sizes than the traditional IB method. We estimated the change in energy over a time-step for the RBF-IB method and determined that it was stable and even dissipative for small N_{d}. Next, we compared the costs of the RBF-IB method for different numbers of data sites against the traditional IB method in the context of simulating platelet motion in channel flow; this test uncovered an unforeseen advantage of the RBF representation, namely, faster convergence from the fluid solver due to smoother forces being spread into the fluid. This allowed the RBF-IB method to be much more efficient than the traditional IB method even on grid resolutions where the fluid solver had greater costs than platelet operations. Finally, we presented the results of a simple 2D platelet aggregation simulation, concluding that the RBF-IB method was indeed well suited to the simulation of platelets in hemodynamic flows.

In Chapter 4, we turned our attention to developing numerical methods to enable simulations of platelet chemistry. We extended Ordinary Differential Equation (ODE) models of platelet surface chemistry to account for platelet geometry, developing two PDE models for describing the evolution of chemical density on platelet surfaces. We developed the first RBF-FD method for simulating reaction-diffusion equations on arbitrary geometries, and applied this method to solving the new PDE models on platelets in 2D domains. These PDEs were coupled to diffusion equations that described how chemicals diffused in the stationary fluid around stationary platelets, a special case of the situation encountered in full platelet simulations with fluid flow and moving platelets. The state of the art in solving PDEs on domains containing platelets, the AFM, had a significant limitation: it could not handle the situation where two platelets were closer than two Eulerian grid cells apart. However, the IB method that simulates the mechanics of platelets allows platelets to be within the same grid cell. To overcome the limitation of the AFM and prepare it for use in conjunction with the IB method, we modified the quadratic interpolation scheme used within the AFM with symmetric Hermite interpolation, thereby "tightening" the interpolation stencils around each platelet and allowing platelets to be in contact. The original AFM used Moving Least Squares (MLS) to handle the coupling between surface

chemistry and fluid chemistry, imposing another restriction on how close platelets could be (due to the wide stencils required by MLS). We removed this restriction using a combination of bilinear interpolation and parametric RBF-based least-squares, thereby also improving the computational efficiency of the numerical method. We demonstrated that our overall numerical method to handle this coupled problem exhibited second-order convergence in space and time.

The numerical method from Chapter 4 contains three components: RBF-FD on surfaces, symmetric Hermite interpolation for the AFM, and a combination of bilinear interpolation and parametric RBF-based least squares for the coupling terms. Of these three components, the second and third are easily extended to platelets in 3D domains: symmetric Hermite interpolation, like standard RBF interpolation, is guaranteed to be well-posed in arbitrary dimensions; also, bilinear interpolation can be trivially extended to trilinear interpolation, and the parametric RBF least-squares model can be trivially extended as well. The untested component for platelets with 2D surfaces (embedded in 3D domains) was the new RBF-FD method. An attempt to extend the RBF-FD method from Chapter 4 to arbitrary 2D surfaces resulted in approximations to elliptic operators that contained eigenvalues with large, positive real parts. A different approach to RBF-FD was required to handle platelet surface chemistry problems for platelets in three dimensions (and, in general, the problem of solving PDEs on 2D surfaces). This motivated the work in Chapter 5.

In Chapter 5, we developed a new RBF-FD method for reaction-diffusion equations on 2D surfaces. This method is more stable than the method developed in Chapter 4 and is more robust to irregular node layouts. The method has two key features. The first is that the surface Laplacian is approximated on a per-stencil basis (instead of approximating surface gradients on each stencil and combining them at the global level into a surface Laplacian, as in Chapter 4). The second feature was a per-stencil optimization of the RBF shape parameter so that the condition number of the interpolation matrix matched a global target condition number. This had the effect of producing stable approximations to the surface Laplacian on many surfaces, given reasonably uniformly distributed nodes on those surfaces. We showed the convergence of our method on (forced) diffusion equations on the sphere and torus for different stencil sizes, generated second-, fourth-, and fifth-order finite difference methods, and achieving the same accuracy as popular narrow-band methods using fewer points. We demonstrated two possible approaches to selecting shape parameters as the total number of nodes on the surface increased, and established the advantages of using small shape parameters (large target condition numbers) in the RBF-FD method. Finally,

we applied this RBF-FD method to the generation of Turing patterns on two types of surfaces: implicit/parametric surfaces and more general surfaces represented only by point clouds.

This work shows that the ability of RBF interpolation to interpolate function values (and derivatives) at scattered node locations can indeed be leveraged to develop numerical methods for biological problems. Much of the content developed in Chapters 2-5 can be applied not only to platelet aggregation, but to many biological problems involving coupled PDEs on irregular domains and surfaces. Further, much of this work offers potential for future research. For example, the work in Chapter 3 could be extended by using meshfree Petrov-Galerkin formulations within the IB method; the work in Chapter 5 could be extended to handle advection on arbitrary surfaces. In the section that follows, we will describe some possible lines of research, continuing the theme of developing numerical methods based on RBF interpolation for the simulation of problems in biology.

6.2 Future Work

In this final section, we will discuss possible extensions of the work conducted in this dissertation, motivated both by the platelet aggregation problem and other biology problems. We will discuss spectrally-accurate projection methods based on Divergence-free RBFs, an extension of the geometric model presented in Chapter 2 to bounded surfaces and for use in the Regularized Stokeslet method (a "cousin" of the IB method), an extension of RBF-FD methods to the solution of variable-coefficient diffusion problems on surfaces and the extension of RBF-FD methods on surfaces to include boundary conditions via the incorporation of symmetric Hermite interpolation.

6.2.1 A Spectrally-accurate Projection Method

Divergence-free RBFs are matrix-valued RBF kernels that were introduced by Narcowich and Ward in the context of the RBF Hermite interpolation problem [62]. Error estimates for interpolation and vector decomposition with divergence-free RBFs were presented by Lowitzsch [59] for functions within the associated reproducing kernel Hilbert spaced of the divergence-free RBF (also known as the native space of the RBF). This work was extended by Fuselier, who provided Sobolev error estimates for target functions that were not as smooth as those in the native space of the divergence-free RBF [34]. Fuselier et al. then provided error estimates for divergence-free RBF interpolants on the sphere [32]. Fuselier and Wright provided estimates for vector field decomposition on the sphere with divergence-free RBFs [33].

One possible line of research is to apply divergence-free RBFs to the Stokes and Navier-Stokes equations, with the goal of projecting intermediate velocity fields onto the space of divergence-free velocity fields at a discrete set of points on some irregular domain. If the domain is fixed, the divergence-free RBF projection matrix can be precomputed just like any other RBF-PS differentiation matrix, and reused over the course of the simulation. The strength of this approach lies in its ability to project vector fields on irregular domains through scattered node placement strategies. This approach would also do away with the need to prescribe pressure boundary conditions for the Poisson problem that arises in conjunction with finite-difference-based projection methods in the Navier-Stokes equations; indeed, with this approach, one only needs boundary conditions on the velocity field. This is an ongoing project with Professors Grady Wright and Edward Fuselier. One possible extension of this project (if it is successfully completed) is to the projection of the Navier-Stokes equations on surfaces using surface-based divergence-free RBFs.

6.2.2 RBFs in the Regularized Stokeslet Method

The Regularized Stokeslet method is a fluid-structure interaction method for the simulation of elastic objects immersed in fluids in a Stokes flow regime. It was developed (and extended) by Ricardo Cortez [11]. The Stokeslet method has since been used in various biological applications.

An ongoing project with Professor Sarah Olson involves augmenting the Regularized Stokeslet method with the RBF geometric model developed in Chapter 2 (and used in the IB method in Chapter 3). Since the Regularized Stokeslet method directly solves for velocities at the Lagrangian points (unlike in the IB method) at a significant computational cost that depends on the number of points, the RBF representation would allow one to get spectrally-accurate elastic forces and lower computational cost without even needing the use of sample sites (unlike in the RBF-IB method), though sample sites could be used if necessary. Beyond this straightforward extension of the Regularized Stokeslet method, we also wish to extend the RBF geometric model to open (bounded) surfaces embedded in 2D domains, both for use in the Regularized Stokeslet and the IB method. Finally, we seek to incorporate recent results from Fuselier and Wright on superconvergent derivative computations with RBFs [35] into our RBF geometric model for use in the Regularized Stokeslet method (and consequently the IB method).

An important feature of the Regularized Stokeslet method is that it uses smoothing functions to smooth forces and generate regularized Stokeslets. One interesting observation is that this smoothing function can be regarded as an RBF. One direction for future research

would be to explore if one can simplify the computations in the Regularized Stokeslet method by using the same RBF to express both the Lagrangian structure and the smoothing function, possibly reducing some of the computational cost of this method.

6.2.3 RBF-FD for Variable-coefficient Diffusion on Surfaces

For the next two future directions, let us consider a simple and illustrative model problem. Consider what would happen if one of the platelets from Figure 4.1 were resting on (or within one grid cell width of) the bottom of the domain in that figure. This is certainly possible (and indeed likely) in a platelet aggregation simulation (see Figure 3.4). In this scenario, binding sites on the part of the platelet resting against the wall would not be free to diffuse outward on the platelet surface– they would be trapped between the bulk of the platelet and the wall. This setting can be numerically modeled in two different ways: the first would be to impose no-flux boundary conditions on an IB point or sample site close to the wall but not under the platelet; this would model the constraints on binding sites by modeling the effect that constraint would have on the chemical density on the platelet surface. The other approach (which would be useful in other scenarios as well) would be to model the coefficient of diffusion associated with those trapped binding sites to zero and "turning off" the reactions at those sites, thus not allowing any chemicals to bind or unbind from there.

The second approach would require an interesting extension of RBF-FD on surfaces to the case of variable-coefficient diffusion. Since we know the approach from Chapter 5 is superior to the one from Chapter 4 in terms of stability, we will discuss the RBF-FD method in that context, though the approach from Chapter 4 would also work in principle. Consider a simple variable-coefficient surface diffusion equation for a scalar variable C:

$$\frac{\partial C}{\partial t} = \nabla_{\mathbb{M}} \cdot (D_{\mathbb{M}} \nabla_{\mathbb{M}} C), \tag{6.1}$$

where $D_{\mathbb{M}}$ is the spatially-varying function on the surface representing the coefficients of surface diffusion. For some RBF-FD stencil P, we may numerically compute this with the RBF-FD approach as

$$\nabla_{\mathbb{M}} \cdot (D_{\mathbb{M}} \nabla_{\mathbb{M}} C)|_P \approx (\nabla_{\mathbb{M}} \cdot I_\phi (D_{\mathbb{M}} \nabla_{\mathbb{M}} I_\phi C))|_P . \tag{6.2}$$

Put simply, we could use the approach from Chapter 5 to compute the Laplacian, but multiply the function $D_{\mathbb{M}}$ in before the second interpolation and differentiation; again, as in Chapter 5, this would be used in a Method of Lines. As a preliminary test of this approach,

we placed a circle of radius $r = 0.0995$ and an ellipse with radii $a = 0.15$ and $b = 0.1$ in a $[0, 1] \times [0, 1]$ domain like the one seen in Figure 4.1. We initialized both surfaces with a chemical concentration of $C(\lambda) = cos(\lambda)$, $0 \leq \lambda \leq 2\pi$. We represented each object with $N_{\mathrm{d}} = 50$ data sites and $N_{\mathrm{s}} = 400$ sample sites and initialized the diffusion coefficients to $D_{\mathbb{M}}(\lambda) = 0.01|(cos(\lambda)|$, $0 \leq \lambda \leq 2\pi$, a choice that gives positive coefficients which dip to zero at sample site indices 100 and 300. We use Model 1 from Chapter 4, adding linear reaction terms to the diffusion equation in Equation (6.1). We use $k_{on} = 0.3$ and $k_{off} = 0.6$ for both platelets. We run the simulation from $t = 0$ to $t = 3$ and plot the results for both $D(\lambda) = 0.01$ and the variable-coefficient diffusion case in Figure 6.1.

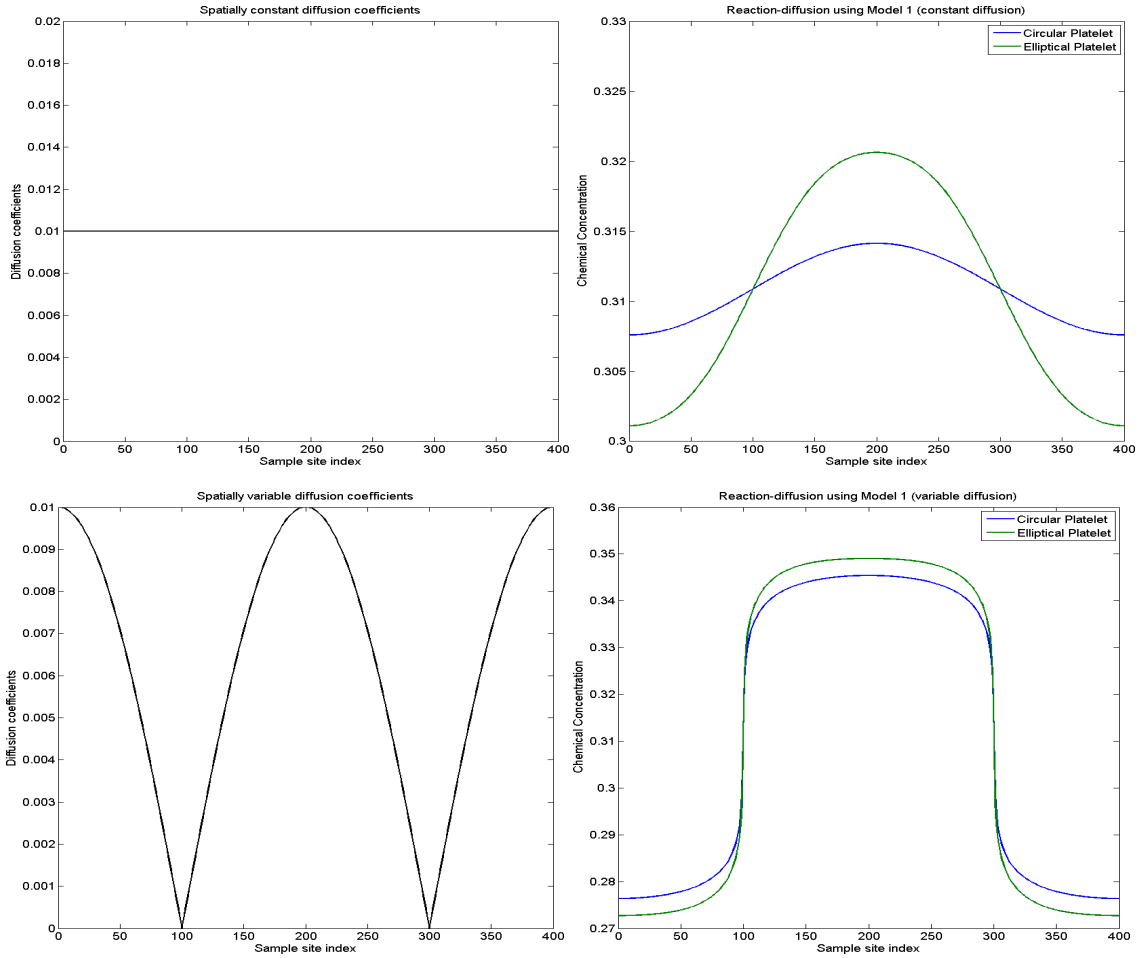

Figure 6.1: Constant and variable-coefficient diffusion on surfaces. The figure on the top left shows the constant diffusion coefficient as a function of sample site index, while the figure on the top right shows the concentrations at time $t = 3$ when using Model 1 and the parameters in the text. The figure on the bottom left shows the spatially variable diffusion coefficents as a function of sample site index, while the figure on the bottom right shows the concentrations at time $t = 3$.

Clearly, using a spatially varying diffusion coefficient results in a very different profile. For this test, we used the RBF-FD method from Chapter 5 and solve the reaction-diffusion equations with an SBDF2 scheme at the sample sites. We set the target condition number to $\kappa_T = 10^{10}$ and the stencil size to $n = 5$. The stability of the RBF-FD method on this test problem is reassuring, especially since $D(\lambda)$ is not an analytic (or even smooth) function.

Future work would include testing the stability and convergence of the RBF-FD scheme for different stencil sizes and point sets on different classes of functions $D(\lambda)$ with varying smoothness, and an extension of the problem to 2D surfaces like the ones studied in Chapter 5.

6.2.4 Boundary Conditions for RBF-FD on Surfaces

Consider the same problem as the one presented in Section 6.2.3, where a platelet is placed on (or very near) the bottom wall of a domain. We proposed a modeling strategy based on variable-coefficient diffusion to capture the physics of the problem, but a more natural approach would be to impose boundary conditions at sample sites that are not quite between the platelet and the wall, but close to the bottom of the platelet. This is illustrated in Figure 6.2. Essentially, if $\boldsymbol{x}$ is the direction along one of the arrows in the figure, we will require $\boldsymbol{x} \cdot \nabla_{\mathbb{M}} C = 0$.

This is an open research question, but one possible approach to tackling this problem would be to use the symmetric Hermite interpolation problem presented in Chapter 1 and used in Chapter 4, albeit on the surface of the platelet. Since the symmetric Hermite interpolation problem has been tackled on the sphere (for a review, see [15]), this project will involve generalizing the Hermite interpolation problem to arbitrary surfaces. The Hermite

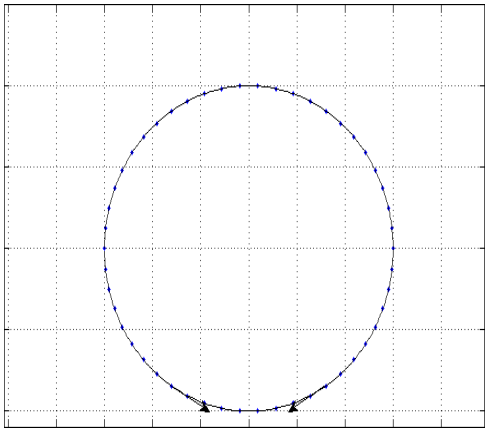

Figure 6.2: Illustration of the imposition of boundary conditions on a platelet. The arrows indicate the directions along which the chemical flux *must not* diffuse.

interpolation approach could also be applied to the imposition of boundary conditions on open 1D curves or 2D sheets. In that scenario, however, the domain is bounded. If boundary conditions are required on the edges of the domain, we will need to adapt the fictitious node strategy for Hermite interpolation developed by Lohmeier (under the guidance of Grady Wright) [58] to open surfaces.

REFERENCES

[1] Uri M. Ascher, Steven J. Ruuth, and Brian T. R. Wetton, *Implicit-explicit methods for time-dependent pdes*, SIAM J. Numer. Anal **32** (1997), 797–823.

[2] John R. Baumgardner and Paul O. Frederickson, *Icosahedral discretization of the Two-Sphere*, SIAM Journal on Numerical Analysis **22** (1985), no. 6, 1107–1115.

[3] V. Bayona, M. Moscoso, M. Carretero, and M. Kindelan, *Rbf-fd formulas and convergence properties*, Journal of Computational Physics **229** (2010), no. 22, 8281–8295.

[4] R. Beatson and M. Langton, *Integral interpolation*, Algorithms for Approximation (2007), 199–218.

[5] R. K. Beatson and H. Q. Bui, *Mollification formulas and implicit smoothing*, Advances in Computational Mathematics **27** (2007), 125–149 (English).

[6] D. Calhoun and C. Helzel, *A finite volume method for solving parabolic equations on logically cartesian curved surface meshes*, SIAM Journal on Scientific Computing **31** (2010), no. 6, 4066–4099.

[7] Donna Calhoun and Randall J. LeVeque, *A cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries*, J. Comput. Phys. **157** (2000), no. 1, 143–180.

[8] T. Cecil, J. Qian, and S. Osher, *Numerical methods for high dimensional Hamilton-Jacobi equations using radial basis functions*, J. Comput. Phys. **196** (2004), 327–347.

[9] G. Chandhini and Y. Sanyasiraju, *Local RBF-FD solutions for steady convection–diffusion problems*, International Journal for Numerical Methods in Engineering **72** (2007), no. 3, 352–378.

[10] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia, *Meshlab: an open-source 3d mesh processing system*, ERCIM News (2008), no. 73, 45–46.

[11] R. Cortez, *The method of regularized stokeslets*, SIAM Journal on Scientific Computing **23** (2001), no. 4, 1204–1225.

[12] O. Davydov and D.T. Oanh, *Adaptive meshless centres and rbf stencils for poisson equation*, Journal of Computational Physics **230** (2011), no. 2, 287–304.

[13] Dharshi Devendran and Charles S. Peskin, *An immersed boundary energy-based method for incompressible viscoelasticity*, Journal of Computational Physics **231** (2012), no. 14, 4613–4642.

[14] E.A. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof, *Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations*, Journal of Computational Physics **161** (2000), 35–60.

[15] G. E. Fasshauer, *Meshfree approximation methods with MATLAB*, Interdisciplinary Mathematical Sciences - Vol. 6, World Scientific Publishers, Singapore, 2007.

[16] G. E. Fasshauer and L. L. Schumaker, *Scattered data fitting on the sphere*, Mathematical Methods for Curves and Surface, Vol.2 of the Proceedings of the 4th Int. Conf. on Mathematical Methods for Curves and Surfaces, Lillehammer, Norway (M. Daehlen, T. Lyche, and L. L. Schumaker, eds.), Vanderbilt University Press, Nashville Tennessee, 1998.

[17] L. J. Fauci and A. L. Fogelson, *Truncated newton methods and the modeling of complex immersed elastic structures*, Communications on Pure and Applied Mathematics **66** (1993), 787–818.

[18] N Flyer, E Lehto, S Blaise, G.B. Wright, and A St-Cyr, *A guide to RBF-generated finite differences for nonlinear transport: shallow water simulations on a sphere*, J. Comput. Phys. **231** (2012), 4078–4095.

[19] N. Flyer and G. B. Wright, *Transport schemes on a sphere using radial basis functions*, Journal of Computational Physics **226** (2007), 1059–1084.

[20] N. Flyer and G. B. Wright, *A radial basis function method for the shallow water equations on a sphere*, Proc. Roy. Soc. A **465** (2009), 1949–1976.

[21] A. L. Fogelson, *A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting*, Journal of Computational Physics **1** (1984), 111–134.

[22] A. L. Fogelson and R. D. Guy, *Immersed-boundary-type models of intravascular platelet aggregation*, Comput. Meth. Appld. Mech. Eng. **197** (2008), 2087–2104.

[23] Aaron Fogelson, Andrew Kuharsky, and Haoyu Yu, *Computational modeling of blood clotting: Coagulation and three-dimensional platelet aggregation*, Polymer and Cell Dynamics: Multicsale Modeling and Numerical Simulations (W. Alt, M. Chaplain, M. Griebel, and J. Lenz, eds.), Birkhaeuser-Verlag, Basel, 2003, pp. 145–154.

[24] Aaron L. Fogelson and Robert D. Guy, *Immersed-boundary-type models of intravascular platelet aggregation*, Computer Methods in Applied Mechanics and Engineering **197** (2008), 2087 – 2104.

[25] B. Fornberg, *A practical guide to pseudospectral methods*, Cambridge University Press, Cambridge, 1996.

[26] B. Fornberg and E. Lehto, *Stabilization of RBF-generated finite difference methods for convective PDEs*, J. Comput. Phys. **230** (2011), 2270–2285.

[27] B. Fornberg and C. Piret, *A stable algorithm for flat radial basis functions on a sphere*, SIAM Journal on Scientific Computing **30** (2007), 60–80.

[28] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Comput. Math. Appl. **48** (2004), 853–867.

[29] B. Fornberg and J. Zuev, *The Runge phenomenon and spatially variable shape parameters in RBF interpolation*, Comput. Math. Appl. **54** (2007), 379–398.

[30] Bengt Fornberg, *Calculation of weights in finite difference formulas*, SIAM Rev **40** (1998), 685–691.

[31] E. Fuselier and G.B. Wright, *Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates*, SIAM Journal on Numerical Analysis **50** (2012), no. 3, 1753–1776.

[32] E J Fuselier, F J Narcowich, J D Ward, and G B Wright, *Error and Stability Estimates for Divergence-Free {RBF} Interpolants on the Sphere*, Math. Comp. **78** (2009), 2157–2186.

[33] E. J. Fuselier and G. B. Wright, *Stability and error estimates for vector field interpolation and decomposition on the sphere with {RBF}s*, {SIAM} J. Num. Anal. **47** (2009), 3213–3239.

[34] Edward J. Fuselier, *Sobolev-type approximation rates for divergence-free and curl-free rbf interpolants.*, Math. Comput. **77** (2008), no. 263, 1407–1423.

[35] Edward J. Fuselier and Grady B. Wright, *Order-preserving approximations of derivatives with periodic radial basis functions*, Submitted.

[36] Edward J Fuselier and Grady B Wright, *A high-order kernel method for diffusion and reaction-diffusion equations on surfaces*, Journal of Scientific Computing **56** (2013), no. 3, 535–565.

[37] Alan George and Joseph W. Liu, *Computer solution of large sparse positive definite*, Prentice Hall Professional Technical Reference, 1981.

[38] Q. T. Lê Gia, *Approximation of parabolic pdes on spheres using spherical basis functions*, Adv. Comput. Math. **22** (2005), 377–397.

[39] D. Goldstein, R. Handler, and L. Sirovich, *Modeling a no-slip flow boundary with an external force field*, Journal of Computational Physics **105** (1993), 354–366.

[40] G. H. Golub and C. F. van Loan, *Matrix computations*, third ed., Johns Hopkins University Press, Baltimore, 1996.

[41] A. Gray, *Modern differential geometry of curves and surfaces with mathematica*, CRC Press, Boca Raton, FL, 1997.

[42] B.E. Griffith, *Simulating the fluid dynamics of natural and prosthetic heart valves using the immersed boundary method.*, Int J Appl Mech. **1** (2009), 137–177.

[43] B.E. Griffith and X. Luo., *Immersed boundary method with finite element elasticity.*, Submitted.

[44] R. D. Guy and A. L. Fogelson, *Stability of approximate projection methods on cell-centered grids*, Journal of Computational Physics **203** (2005), 517–538.

[45] D. P. Hardin and E. B. Saff, *Discretizing manifolds via minimum energy points*, Notices Amer. Math. Soc. **51** (2004), 1186–1194.

[46] S. Hubbert and S. Müller, *Interpolation with circular basis functions*, Numerical Algorithms **42** (2006), 75–90.

[47] Simon Hubbert and Tanya M. Morton, *l_p-error estimates for radial basis function interpolation on the sphere*, J. Approx. Theory **129** (2004), 58–77.

[48] A. Iske et al., *Reconstruction of functions from generalized hermite-birkhoff data*, Series in Approximations and Decompositions **6** (1995), 257–264.

[49] S.P. Jackson, W.S. Nesbitt, and S. Kulkarni, *Signaling events underlying thrombus formation*, J Thromb Haemost **1** (2003), 1602–1612.

[50] Kurt Jetter, Joachim Stöckler, and Joseph D. Ward, *Error estimates for scattered data interpolation on spheres*, Math. Comput. **68** (1999), no. 226, 733–747.

[51] Hans Svend Johansen and Phillip Colella, *A cartesian grid embedded boundary method for poisson's equation on irregular domains*, Journal of Computational Physics **147** (1998), 60–85.

[52] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe, *Poisson surface reconstruction*, Proceedings of the Fourth Eurographics Symposium on Geometry Processing (Aire-la-Ville, Switzerland, Switzerland), SGP '06, Eurographics Association, 2006, pp. 61–70.

[53] Jungwoo Kim, Dongjoo Kim, and Haecheon Choi, *An immersed-boundary finite-volume method for simulations of flow in complex geometries*, Journal of Computational Physics **171** (2001), 132–150.

[54] A. L. Kuharsky and A. L. Fogelson, *Surface-mediated control of blood coagulation: The role of binding site densities and platelet deposition*, Biophysical Journal **80** (2001), 1050–1074.

[55] E. Larsson and B. Fornberg, *Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions*, Comput. Math. Appl. **49** (2005), 103–130.

[56] E. Larsson, E. Lehto, A. Heryudono, and B. Fornberg, *Stable computation of differentiation matrices and scattered node stencils based on gaussian radial basis functions*, SIAM Journal on Scientific Computing **35** (2013), no. 4, A2096–A2119.

[57] R. J. LeVeque and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal. **31** (1994), 1001–1025.

[58] Joseph Lohmeier, *A fictitious point method for handling boundary conditions in the rbf-fd method*, Master's thesis, Boise State University, 2011.

[59] Svenja Lowitzsch, *Error estimates for matrix-valued radial basis function interpolation*, Journal of Approximation Theory **137** (2005), no. 2, 238 – 249.

[60] C. Macdonald and S. Ruuth, *The implicit closest point method for the numerical solution of partial differential equations on surfaces*, SIAM Journal on Scientific Computing **31** (2010), no. 6, 4330–4350.

[61] J. Mohd-Yusof, *Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries*, Annu. Res.Briefs, Cent. Turbul. Res. (1997), 317–328.

[62] F.J. Narcowich and J.D. Ward, *Generalized hermite interpolation via matrix-valued conditionally positive definite functions*, Mathematics of Computation **63** (1994), no. 208, 661–688.

[63] Francis J. Narcowich, Xingping Sun, Joseph D. Ward, and Holger Wendland, *Direct and inverse Sobolev error estimates for scattered data interpolation via spherical basis functions*, Found. Comput. Math. **7** (2007), no. 3, 369–390. MR MR2335250

[64] Elijah P. Newren, Aaron L. Fogelson, Robert D. Guy, and Robert M. Kirby, *Unconditionally stable discretizations of the immersed boundary equations*, Journal of Computational Physics **222** (2007), 702–719.

[65] C. S. Peskin, *Flow pattern around heart valves: a numerical method*, Journal of Computational Physics **10** (1972), 252–271.

[66] C. S. Peskin, *Numerical analysis of blood flow in the heart*, Journal of Computational Physics **25** (1977), 220–252.

[67] Charles. S. Peskin, *The immersed boundary method*, Acta Numerica **11** (2002), 479–517.

[68] C. Piret, *The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces*, Journal of Computational Physics **231** (2012), no. 20, 4662–4675.

[69] S. Rippa, *An algorithm for selecting a good value for the parameter c in radial basis function interpolation*, Adv. Comput. Math. **11** (1999), 193–210.

[70] Ali Safdari-Vaighani, Alfa Heryudono, and Elisabeth Larsson, *A radial basis function partition of unity collocation method for convection-diffusion equations*, Tech. Report 2013-023, Department of Information Technology, Uppsala University, November 2013.

[71] Brian Savage, Enrique Saldivar, and Zaverio M. Ruggeri, *Initiation of platelet adhesion by arrest onto fibrinogen or translocation on von Willebrand factor*, Cell **84** (1996), 289–297.

[72] Varun Shankar, Grady B. Wright, Aaron L. Fogelson, and Robert M. Kirby, *A study of different modeling choices for simulating platelets within the immersed boundary method*, Appl. Numer. Math. **63** (2013), 58–77.

[73] ———, *A radial basis function (rbf) finite difference method for the simulation of reactiondiffusion equations on stationary platelets within the augmented forcing method*, International Journal for Numerical Methods in Fluids **75** (2014), no. 1, 1–22.

[74] Varun Shankar, Grady B. Wright, Robert M. Kirby, and Aaron L. Fogelson, *Augmenting the immersed boundary method with radial basis functions (rbfs) for the modeling of platelets in hemodynamic flows*, Submitted.

[75] ———, *A radial basis function (rbf)-finite difference (fd) method for diffusion and reaction-diffusion equations on surfaces*, Submitted.

[76] L. Shen, H. Farid, and M. A. McPeek, *Modeling three-dimensional morphological structures using spherical harmonics*, Evolution **4** (2009), no. 63, 1003–1016.

[77] C. Shu, H. Ding, and KS Yeo, *Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible navier–stokes equations*, Computer Methods in Applied Mechanics and Engineering **192** (2003), no. 7, 941–954.

[78] D. Stevens, H. Power, M. Lees, and H. Morvan, *The use of PDE centers in the local RBF Hermitean method for 3D convective-diffusion problems*, Journal of Computational Physics **228** (2009), 4606–4624.

[79] Kazuyasu Sugiyama, Satoshi II, Shintaro Takeuchi, Shu Takagi, and Yoichiro Matsumoto, *A full eulerian finite difference approach for solving fluid-structure coupling problems*, Journal of Computational Physics **230** (2010), no. 3, 38.

[80] X. Sun, *Scattered hermite interpolation using radial basis functions*, Linear algebra and its applications **207** (1994), 135–146.

[81] Grit Thrmer and Charles A. Wthrich, *Computing vertex normals from polygonal facets*, journal of graphics, gpu, and game tools **3** (1998), no. 1, 43–46.

[82] AI Tolstykh and DA Shirobokov, *On using radial basis functions in a finite difference mode with applications to elasticity problems*, Computational Mechanics **33** (2003), no. 1, 68–79.

[83] Lloyd N. Trefethen, *Spectral methods in MATLAB*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[84] V. T. Turitto and H. L. Goldsmith, *Rheology, transport and thrombosis in the circulation*, Textbook of Vascular Medicine (J. Loscalzo, M. Creager, and V. Dzau, eds.), Little, Brown & Co., New York, 2nd ed., 1996, pp. 141–184.

[85] H. S. Udaykumar, R. Mittal, and Wei Shyy, *Computation of solid-liquid phase fronts in the sharp interface limit on fixed grids*, Journal of Computational Physics **153** (1999), 535–574.

[86] Marcos Vanella and Elias Balaras, *Short note: A moving-least-squares reconstruction for embedded-boundary formulations*, J. Comput. Phys. **228** (2009), 6617–6628.

[87] C. Varea, J. Aragon, and R. Barrio, *Turing patterns on a sphere*, Phys. Rev. E **60** (1999), 4588–4592.

[88] Holger Wendland, *Scattered data approximation*, Cambridge Monographs on Applied and Computational Mathematics, vol. 17, Cambridge University Press, Cambridge, 2005. MR MR2131724 (2006i:41002)

[89] R. S. Womersley and I. H. Sloan, *How good can polynomial interpolation on the sphere be?*, Adv. Comput. Math. **23** (2001), 195–226.

[90] R. S. Womersley and I. H. Sloan, *Interpolation and cubature on the sphere*, Website, 2007, http://web.maths.unsw.edu.au/ rsw/Sphere/.

[91] Grady B. Wright and Bengt Fornberg, *Scattered node compact finite difference-type formulas generated from radial basis functions*, Journal of Computational Physics **212** (2006), no. 1, 99 – 123.

[92] Jianming Yang, *An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries*, Ph.D. thesis, University of Maryland, College Park, College Park, Maryland, 2005.

[93] Lingxing Yao and Aaron L. Fogelson, *Simulations of chemical transport and reaction in a suspension of cells i: an augmented forcing point method for the stationary case*, International Journal for Numerical Methods in Fluids **69** (2012), no. 11, 1736–1752.

[94] T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy, *An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries*, Journal of Computational Physics **156** (1999), 209–240.

[95] Haoyu Yu, *Three dimensional computational modeling and simulation of platelet aggregation on parallel computers*, Ph.D. thesis, University of Utah, 2000.

[96] Lucy Zhang, Axel Gerstenberger, Xiaodong Wang, and Wing Kam Liu, *Immersed finite element method*, Computer Methods in Applied Mechanics and Engineering **193** (2004), no. 21-22, 2051 – 2067, Flow Simulation and Modeling.