DEVELOPMENT OF GENERAL FINITE DIFFERENCES FOR COMPLEX GEOMETRIES USING IMMERSED BOUNDARY METHOD

A Thesis
Presented to
The Academic Faculty

by

Yaroslav V. Vasyliv

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology
December 2015

DEVELOPMENT OF GENERAL FINITE DIFFERENCES FOR COMPLEX GEOMETRIES USING IMMERSED BOUNDARY METHOD

Approved by:

Professor Alexander Alexeev, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Professor Satish Kumar
School of Mechanical Engineering
Georgia Institute of Technology

Professor Edmond Chow
School of Computational Science and Engineering
Georgia Institute of Technology

Date Approved: 4 November 2015

To my parents,

who came to the United States with four kids, a hundred bucks, and a dream.

ACKNOWLEDGEMENTS

I would like to thank Dr. Alexeev for his practical research advise as well as his dedication to furnishing an encouraging and enabling lab environment. I would also like to acknowledge the rest of the committee for their valuable feedback. Lastly, I am grateful for the financial support provided by the National Science Foundation (NSF) Graduate Research Fellowship, Grant No. DGE-1148903.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In meshfree methods, partial differential equations are solved on an unstructured cloud of points distributed throughout the computational domain. In collocated meshfree methods, the differential operators are directly approximated at each grid point based on a local cloud of neighboring points. The set of neighboring nodes used to construct the local approximation is determined using a variable search radius which establishes an implicit nodal connectivity and hence a mesh is not required. As a result, meshfree methods have the potential flexibility to handle problem sets where the computational grid may undergo large deformations as well as where the grid may need to undergo adaptive refinement. In this work we develop an immersed boundary framework for collocated meshfree approximations. We use the framework to implement three meshfree methods: General Finite Differences (GFD), Smoothed Particle Hydrodynamics (SPH), and Moving Least Squares (MLS). We evaluate the numerical accuracy and convergence of these methods by solving the 2D Poisson equation. We demonstrate that GFD is computationally more efficient than MLS and show that its accuracy is superior to a popular corrected form of SPH and comparable to MLS. We then use GFD to solve several canonic steady state fluid flow problems on meshfree grids generated using uniform and variable radii Poisson-disk algorithm.

1.1 Outline

In Chapter 2, we introduce collocated meshfree methods - as described by Belytschko et al. and Onate et al. - briefly touching on how boundary conditions are typically handled. Before proceeding to discuss various meshfree methods, we introduce shape functions and explain why in general meshfree shape functions lose their interpolating property. We then proceed to give an initial introduction to General Finite Differences (GFD), holding off on a detailed explanation until Section 3.1. Instead, we focus on tracking the history of GFD, introducing popular pseudonyms for the method. In Section 2.4, we show that the backbone of the approximation in the Diffuse Element Method is simply the GFD approximation using a complete basis. We next introduce the Moving Least Squares (MLS) approximation and derive how a collocated version may appear where the constant basis has been removed. We show that for function and 1^{st} derivative approximations GFD and MLS will produce the same estimate, while for 2^{nd} derivative approximations, MLS carries an additional term

related to the spatial variation of the weight functions. We later use collocated MLS to solve the 2D Poisson equation in Section 6.1.

We spend the remainder of the Chapter 2 introducing Smoothed Particle Hydrodynamics (SPH) with the majority of the time devoted to the various corrections proposed to restore consistency. The approaches to restore consistency can be grouped into Taylor-series based corrections and reproducing corrections. We first summarize the main Taylor series based corrections which we further subdivide into implicit and semi-implicit corrections. Under the semi-implicit category, higher order derivative approximations are constructed sequentially from lower order derivative approximations. We introduce Chen and Beraun's Corrected Smoothed Particle Method (CSPM) which lacks consistent 2^{nd} derivatives due to propagation of same order errors from the 1^{st} derivative approximations. Subsequently, we introduce Fatehi and Manzari's consistent approach for 2^{nd} derivatives. Contrary to CSPM, Fatehi and Manzari keep track of the same order truncation error resulting from the first gradient correction and use it in the construction of the second correction. Last under this category is a simple form for second derivatives based on Randles and Libersky popular gradient correction applied to Brookshaw's approximation. Here we call this 2^{nd} derivative approximation, Brookshaw Corrected Gradient (BCG). We test the BCG approximation on the simple 2D Poisson example in Section 6.1. Moving to the implicit category, we introduce Zhang and Batra's consistent Modified Smoothed Particle Hydrodynamics (MSPH). In MSPH, the Taylor series is integrated with respect to spatial derivatives of the kernel resulting in a system of equations which allows for higher and lower order derivatives to be approximated simultaneously via a single matrix inversion.

Following Belytschko et al., we introduce the reproducing conditions and outline how they can be applied to correct the kernel and the gradient of the kernel. When the polynomial reproducing conditions are enforced on the kernel approximation, the corrected kernel ultimately leads to the MLS approximation. In this manner, SPH using a completely corrected weight function is simply the MLS approximation. We then summarize the reproducing corrections as they apply to the gradient of the kernel. We show that Method 2, proposed by Belytschko et al., is identical to GFD with a complete basis while Method 3 - another gradient correction approach proposed by the same group - leads to MSPH.

In Chapter 3, we formally introduce GFD as the minimization of the euclidean error norm with respect to the Taylor coefficients. In order to use the approximation to solve boundary value problems such as those that may be encountered in a fractional step Navier-Stokes solver, we show how to rewrite the approximation in terms of i neighboring scaled shaped functions evaluated at the collocation point. To simplify the discretization of operators, we introduce the $m \times n$ matrix of scaled shape functions $\underline{\underline{\phi}}$, where the rows m correspond to

contributions to the m^{th} Taylor coefficient while the columns n correspond to the i^{th} neighbor. In this manner, a particular row contains the scaled coefficients necessary to discretize the m^{th} Taylor coefficient for a particular collocation point (i.e., star node a). In Section 3.2, we present a novel generalization of the sharp interface form of the Immersed Boundary method to collocated meshfree methods. By using linear extrapolation matrices $\underline{\underline{Q}}$ and $\underline{\underline{R}}$ - for Dirichlet and Neumann boundary conditions respectively - we extrapolate appropriate values to ghost nodes such that boundary conditions are fulfilled at all intersecting boundary points between a star node a and ghost nodes g_i in the star node's support domain. Continuing further, since the ghost node values are simply a linear combination of the neighboring values of node a, we modify the i neighboring coefficients in the computational stencil of a accordingly. During this process, we keep track of the additional known boundary terms that may fall out to the RHS for row a of the corresponding system. As we show through numerical tests in Chapter 6, the combination of modifying the coefficients and RHS - for a star node a near the boundary - allows us to enforce the boundary conditions without actually having equations corresponding to nodes on the boundary.

In Chapter 4, we address automatic and (potentially) adaptive grid generation for meshfree methods. We introduce uniform and variable radii Poisson disk sampling - a random grid generation technique mostly used by the graphics community. In Poisson disk sampling, the grid points are generated in such a way as to respect a certain spacing between grid points (i.e., each grid point has a disk or bubble which no other grid points may occupy). The spacing between grid points may be specified arbitrarily by some function. As example, we use the variable radii Poisson disk algorithm to generate a grid based on an image's normalized greyscale. We later use the algorithm to generate the uniform and variable meshfree grids used in the validation test cases presented in Chapter 6. We conclude the chapter with a brief discussion on a neighbor search algorithm.

In Chapter 5, we introduce the backbone of the algorithm use to solve the fluid test cases presented in Chapter 6 - an explicit fractional step solver for the Navier-Stokes equations. In the method, the momentum equation is split into a viscous and non-viscous parts. In the viscous equation, the pressure term is dropped and an intermediate velocity may be obtained which does not satisfy the divergence-free condition but does satisfy both normal and tangential velocity boundary conditions. In the non-viscous equation, the intermediate velocity field is decomposed into a divergence free velocity field and curl free vector field (i.e., the gradient of pressure). By taken the divergence of the non-viscous equation, a Poisson Pressure Equation (PPE) may be obtained with a right hand side equal to the divergence of the intermediate velocity field. Solving the boundary value problem for the pressure, the intermediate velocity field can be corrected using the non-viscous equation to

obtain a divergence free velocity field which satisfies the normal boundary condition and approximately satisfies the tangential condition. Concluding the chapter, we outline the overall meshfree algorithm within the Immersed Boundary framework, list the discretization of various operators in cartesian and axisymmetric cylindrical coordinates, and lastly briefly elaborate on post-processing.

In Chapter 6, we present validation test cases solved within the generalized sharp interface framework using grids generated by the uniform and variable radii Poisson disk algorithm. We first solve the 2D Poisson equation with Dirichlet boundary conditions considering three different meshfree approximation techniques: GFD, MLS, and BCG. In all three methods, we remove the constant basis. Using the same meshfree grids we evaluate the discrete L_2 relative error norm and spatial convergence rate and compare results to central differences. Numerical tests indicate that when the constant basis is removed GFD has accuracy comparable to MLS and superior to the BCG approximation and moreover is computationally more efficient than MLS. Furthermore, our 2D numerical tests used on average 14 - 20 neighbors, a factor of 2 - 3 times lower than the average number of neighbors typically reported in the inconsistent but conservative SPH approximations. Choosing GFD to explore further, we solve two canonical steady state fluid flow problems - the lid driven cavity and uniform flow over a cylinder. For the lid driven cavity, we compare the horizontal and vertical velocity solutions to the Ghia data set using Reynolds number of Re = 100, 400, 1000, 3200. While for the uniform flow over a cylinder test case, we compare the skin friction and pressure coefficients to a boundary fitted solution for $Re = 40$. We conclude the validation examples with the 2013 Food and Drug Administration cardiovascular benchmark. Using an axisymmetric model and a Poisson-disk grid, we first compare the axial velocity profiles and the centerline velocity profile to an axisymmetric Lattice Boltzmann Method (LBM) solution for a low Reynolds number $Re = 50$. Unable to resolve stability issues encountered on the Poisson-disk grids at higher Reynolds numbers, we switch to a simple lattice arrangement for $Re = 500$, and using $n = 13$ neighbors compare GFD results to the LBM solution as well as to the available experimental data sets.

CHAPTER II

BACKGROUND AND RELATED TOPICS

2.1 Collocated Meshfree Method

We first introduce the traditional collocated meshfree method as described by Belytschko et al. and Onate et al. [3, 25]. Supposed we have the following boundary value problem:

$$\mathcal{L}u(\vec{x}) = f(\vec{x}) \quad , \quad \vec{x} \in \Omega - \Gamma \tag{2.1}$$

where $\mathcal{L}$ is a differential operator acting on the unknown field $u(\vec{x})$. With $u(\vec{x})$ subject to the following boundary conditions:

$$\begin{aligned} u(\vec{x}) &= g(\vec{x}) \quad , \quad \vec{x} \in \Gamma_d \\ \nabla u(\vec{x}) \cdot \hat{n} &= h(\vec{x}) \quad , \quad \vec{x} \in \Gamma_n \end{aligned} \tag{2.2}$$

The boundary Γ is split into the portion associated with Dirichlet conditions Γ_d and the portion associated with Neumann conditions Γ_n. Here $\hat{n}$ represents the unit normal at the boundary and so the quantity $\nabla u(\vec{x}) \cdot \hat{n}$ is the gradient in the normal direction. We can construct an approximate solution $u^h(\vec{x})$ as the following linear combination:

$$u^h(\vec{x}) = \sum_{i}^{NP} \phi_i(\vec{x}) u_i \tag{2.3}$$

where $i \in 1, 2, ...NP$ and where $\phi_i(\vec{x})$ are a set of linearly independent functions called shape functions. Here $NP = N_d + N_n + N$ and is the total number of nodes discretizing Ω. In a collocation approach, the approximation given by Eq. 2.3 is directly substituted into Eq. 2.2 and is evaluate at the NP nodes discretizing the domain. In other words, the residual is minimized at a set of NP discrete points. Upon substitution we arrive at the following $NP \times NP$ system:

$$\mathcal{L}u^h(\vec{x}_a) = f(\vec{x}_a) \quad , \quad \vec{x}_a \in N \tag{2.4}$$

$$u^h(\vec{x}_a) = g(\vec{x}_a) \quad , \quad \vec{x}_a \in N_d \tag{2.5}$$

$$\nabla u^h(\vec{x}_a) \cdot \hat{n} = h(\vec{x}_a) \quad , \quad \vec{x}_a \in N_n \tag{2.6}$$

where the differential operators are simply passed off onto the shape functions. Note that here we have $(N_d + N_n)$ equations stemming from nodes that lay directly on the boundary.

As $NP \to \infty$, we except the residual to be zero everywhere and - stability aside - expect the approximate solution to converge to the exact solution (i.e., the approximation is consistent).

To ensure a sparse matrix, a local support domain for the node a is enforced via a weight function, also called a kernel function. The weight function's support domain is typically chosen to be a circle (2D) with radius kh, where h is the smoothing length and k is a factor specific to the weight function. Outside of the support domain, the weight function is zero. Consequently, the neighbors contributing to the local approximation at node a are found within a search radius kh. Moreover, establishing the nodal connectivity via a search radius circumvents the requirement for an explicit meshing procedure. Note that some authors use rectangular support domains. Here we only consider circular domains. Continuing further, the approximation evaluated a collocation point $u^h(\vec{x}_a)$ will read:

$$u^h(\vec{x}_a) = \sum_i^n \phi_i(\vec{x}_a) u_i \tag{2.7}$$

where n is the number of neighbors found in the local support domain for node a. In this work, sometimes we will drop the the h superscript and just list the approximation as u_a, a commonly used shorthand in the cited texts. Similarly for the derivatives we would have:

$$\left.\frac{\partial u^h}{\partial x}\right|_{\vec{x}_a} = \sum_i^n \left.\frac{\partial \phi_i}{\partial x}\right|_{\vec{x}_a} u_i \tag{2.8}$$

Note that finding the right number of neighbors is critical as it will determine whether shape functions can be found (i.e., is the matrix singular?). Moreover, as will be shown using GFD in Chapter 6, too large of a support domain will reduce the accuracy of the approximation. It is clear, finding the right spatial distribution of neighbors ultimately determines the stability and accuracy of the approximation. In this work, we try to alleviate these concerns by producing grids based on the Poisson-disk algorithm discussed later. On a final note, we will not implement boundary conditions as described above but rather we will generalize the sharp interface variant of the Immersed Boundary method to directly modify the computational stencil such that boundary conditions are approximately enforced [23]. As such, the framework we present in Section 3.1 will only require an $N \times N$ system to be solved rather than an $NP \times NP$ system. Note that the approximations provided can easily be used on regular grids. In this work we choose to work with irregular grids because of the possible extension of our code to Lagrangian particle-based solvers. This is discussed further in Chapter 5.

2.2 Shape Functions

Before proceeding, it is necessary to understand shape functions a little better. We give a simple example using the 1D linear polynomial basis as our approximation space (i.e., $\mathbf{P}^T = \begin{bmatrix} 1 & x \end{bmatrix}$). If the number of points is equivalent to the number of basis, then we will have the following system in order to find a_0 and a_1 such that $u^h(x) = \mathbf{P}^T\mathbf{a}$ interpolates values u_1 and u_2:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.9}$$

and so $\mathbf{a} = \underline{\underline{A}}^{-1}\mathbf{u}$, where we have let $\mathbf{u}$ be the vector of nodal values and have let $\underline{\underline{A}}$ denote the matrix [1]. Substituting the found coefficients, we now have the linear approximation $u^h(x) = \mathbf{P}^T\underline{\underline{A}}^{-1}\mathbf{u}$, where the vector of shape functions $\mathbf{\Phi}(x) = \begin{bmatrix} \phi_1(x) & \phi_2(x) \end{bmatrix}$ is:

$$\mathbf{\Phi} = \mathbf{P}^T\underline{\underline{A}}^{-1} \tag{2.10}$$

Here the number of unknown coefficients is equivalent to the number of nodal values and so the shape functions are just the Lagrange interpolation polynomials. That is, the shape functions have the following property:

$$\phi_i(x_j) = \delta_{ij} \tag{2.11}$$

In meshfree methods, the interpolating property of shape functions is generally lost unless the number of coefficients are equal to the number of unknown nodal values. However, this does not mean if there are more nodes than coefficients that the approximation can not interpolate all nodal values in its cloud. If the shape functions enforce the reproducing conditions and if the function being approximated is spanned by the basis then the approximation $u^h(\vec{x})$ will still interpolate all points. Typically, the underlying function is rarely spanned by the basis. This implies for a particular point i in the cloud of star node a that $u^h(\vec{x}_i) \neq u_i$, which in turn implies that the locally constructed approximations are multivalued between clouds. It is often argued that when using "point collocation" [25, 1] it is okay for the approximation to be multivalued as long as you restrict use of the approximation to the star node, however, it is unclear what underlying implications may exist. We do not discuss this further, but simply point it out to the reader.

[1] To improve the conditioning number of the matrix, typically the basis is shifted about point x_a and the polynomial approximation is now simply the truncated Taylor series with the unknown coefficients now the Taylor coefficients.

2.3 General Finite Differences (GFD)

Smoothed Particle Hydrodynamics (SPH) is widely attributed as the eldest of the mesh-free family of methods, dating back to an astrophysics paper in 1977 by Gingold and Monaghan [13]. Predating SPH by five years[2] was Jensen when he published finite differences for arbitrary grids (FIDAG) in 1972 [16]. In FIDAG, Jensen constructed a m^{th} order Taylor series expansion about a node a by interpolating the $(m+1)(m+2)/2$ required nodes in the neighboring 2D area to construct a nodal matrix which could be used to approximate the derivative vector (i.e., the Taylor coefficients) to $O(h^{m+1-i})$, where i refers to the i^{th} derivative. As one may expect, this approach was found to be extremely sensitive to the nodal distributions.

Subsequently in 1974, Perrone and Kao extended these ideas by considering u_a as known for the approximation, proposing a 9 point control scheme to improve the accuracy and conditioning of the nodal matrix. Perrone and Kao point out that if the mixed derivative is dropped and u_a is considered known, the scheme collapses to the well known central difference estimates for the 1^{st} and 2^{nd} derivatives when using equally spaced points. Motivated by recovering the central difference estimate for the mixed derivative, they showed that by taking the 5^{th} point to be one of the four corners, 4 sets of 5×5 systems could be obtained which when averaged recover the central difference scheme for the mixed derivative. They extended this notion to an irregular grid by considering the 4 points closest to the orthogonal axes as nodes 1-4, and choosing the 5th point - four times - from the remaining points and averaging the resulting systems [26].

The main ideas that have been introduced so far are not new and in fact mathematically probably have been around for quiet sometime. It was not until they became computational practical around the 70's did a number of papers get published on the subject – see Reference [19]. Ultimately, they were overshadowed by the simultaneous rise of the Finite Element Method (FEM). Nonetheless, by 1979 Liszka and Orkisz introduced a set of computer programs FIDAM (Finite Difference-Arbitrary Mesh)[3]. Generalizing Perrone and Kao's work, they introduced more nodes in the approximation leading to an overdetermined system. They obtained a solution for the Taylor coefficients by solving a weighted least squares problem under the squared error norm. They reported better accuracy for derivatives than traditional central difference estimates or Perrone's elaborate averaging process.

Over the years, the basic approximation as outlined in Liszka's paper has resurfaced multiple times under various pseudonyms: Least Square Finite Differences (LSFD), Finite

[2]See Perrone and Kao [26] references 8-9 for even earlier texts from the 1960s.

[3]To paint the state of computational power at this time, their program was limited to about 1000 nodes.

Point Method (FPM)[4], Finite Cloud Method (FCM) [5] and as the backbone of the approximation in Diffuse Element Method (DEM) [24]. It even has been mistakenly labeled as Moving Least Squares (MLS) by Gossler at Sandia Labs [14]. A quick inspection shows that the differentiation only took place with respect to the polynomial basis leading to an identical approximation as found in Reference [19]. Here we will refer to this local Taylor series based approximation as General Finite Differences (GFD) since it can be shown to reduce to classical finite difference estimates under special circumstances. We hold off on a detailed explanation of GFD until Section 3.1 and instead choose to first describe other meshfree methods.

2.4 Diffuse Element Method (DEM)

In 1992, Nayroles et al. introduced the Diffuse Element Method (DEM)[24]. In DEM, the weak form of a PDE is solved using a weighted least squares fitting performed on "diffuse" elements consisting of a cloud of neighboring points surrounding an evaluation point (i.e., the sole integration point for the "diffuse element"). As they point out, by shifting the origin to occur about the evaluation point (i.e., shift the basis from $\mathbf{P}^T(\vec{x})$ to $\mathbf{P}^T(\vec{x}-\vec{x}_a)$), DEM's formulation ultimately approximates the Taylor coefficients. When the coordinate shift is performed their approximation for the solution over an element with nodal values $\mathbf{u}$ and evaluation point located at $\vec{x}_a$ reads:

$$u^h(\vec{x}) = \mathbf{P}^T(\vec{x}-\vec{x_a})\mathbf{a}(\vec{x}_a) \tag{2.12}$$

Solving the least squares problem for $\mathbf{a}(\vec{x}_a)$ and upon substitution:

$$u^h(\vec{x}) = \mathbf{P}^T(\vec{x}-\vec{x_a})\underline{\underline{A}}^{-1}(\vec{x}_a)\underline{\underline{B}}(\vec{x}_a)\mathbf{u} \tag{2.13}$$

where the vector of shape functions corresponding to each of the n neighboring nodes is:

$$\mathbf{\Phi}(\vec{x}) = \mathbf{P}^T(\vec{x}-\vec{x_a})\underline{\underline{A}}^{-1}\underline{\underline{B}} \tag{2.14}$$

From the collocation perspective, here the matrices $\underline{\underline{A}}$ and $\underline{\underline{B}}$ are identical to those presented in the the GFD section when the constant basis has not been removed. Note that Nayroles et al. assumed the weight function was a continuous function as opposed to discrete weights, however, when evaluating derivatives, ultimately, only the polynomial term in the shape

[4]Onate et. al. refered to a variety of meshfree methods under this name but - as they themselves point out - they ultimately implemented Liszka's method referring to it as Weighted Least Squares (WLS)[25].

[5]In FCM, Aluru and Li referred to the approximation as the "fixed reproducing kernel approximation" [1].

function was differentiated while the rest of the terms were assumed constant. For instance, for $\dfrac{\partial}{\partial x}$ we have:

$$\frac{\partial \mathbf{\Phi}}{\partial x} = \frac{\partial \mathbf{P}^T(\vec{x} - \vec{x}_a)}{\partial x} \underline{\underline{A}}^{-1} \underline{\underline{B}} \tag{2.15}$$

We remark that if the DEM approximation is used with a collocation technique then it will be equivalent to the GFD approximation. In DEM and GFD the approximation is m times differentiable (as determined by the order of the basis) while MLS (as shown next) will produce an approximation k times differentiable (as determined by the continuity of the weight function). For example, if we wished to approximate derivatives using DEM with a collocation approach, we can evaluate the derivative at the star node a using the quadratic basis:

$$\left.\frac{\partial u^h}{\partial x}\right|_{\vec{x}_a} = \left.\frac{\partial \mathbf{\Phi}}{\partial x}\right|_{\vec{x}_a} \mathbf{u} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \underline{\underline{A}}^{-1} \underline{\underline{B}} \mathbf{u} = \sum_i^n \phi_{2i} u_i \tag{2.16}$$

or for $\dfrac{\partial^2}{\partial x^2}$:

$$\left.\frac{\partial^2 u^h}{\partial x^2}\right|_{\vec{x}_a} = \left.\frac{\partial^2 \mathbf{\Phi}}{\partial x^2}\right|_{\vec{x}_a} \mathbf{u} = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix} \underline{\underline{A}}^{-1} \underline{\underline{B}} \mathbf{u} = 2 \sum_i^n \phi_{4i} u_i \tag{2.17}$$

Here we see that the estimate of derivatives are identical to those presented in Section 3.1 when u_a is treated as unknown. In fact Nayroles et al. mentions that one of the possible uses of their approximation is to "generate new 'finite difference like' formula based on any irregular set of discretization points". Apparently, the group was not aware of the work done a decade earlier by Liszka and Orkisz [19].

2.5 Moving Least Squares (MLS) and Element-Free Galerkin Method (EFG)

In 1994, Belytschko et al. extended the ideas presented in DEM in their EFG Method where again the Galerkin method is used with shape functions constructed from a weighted least squares fitting over a cloud of neighbors [4]. Besides two important differences related to boundary conditions and numerical evaluation of resulting integrals, Belytschko et al., citing Lancaster and Salkauskas [18], include the spatial variation of the matrices $\underline{\underline{A}}$ and $\underline{\underline{B}}$, making their approximation the moving least squares (MLS) approximation which reads:

$$u^h(\vec{x}) = \mathbf{P}^T(\vec{x} - \vec{x}_a)\mathbf{a}(\vec{x}) \tag{2.18}$$

Here it is important to emphasize that $\mathbf{a}(\vec{x})$ is a vector of "coefficients" which are functions of space – they are not constant. The least squares problem under the weighted squared error norm reads:

$$E = \sum_{i}^{n} \left(\mathbf{P}^T(\vec{x}_i - \vec{x}_a)\mathbf{a}(\vec{x}) - u_i\right)^2 W(\vec{x} - \vec{x}_i) \tag{2.19}$$

where $W(\vec{x} - \vec{x}_i) = W(\vec{x}_i - \vec{x})$ is now a continuous weight function, rather than discrete weights as in GFD, making $\mathbf{a}(\vec{x}_a)$ now a function $\mathbf{a}(\vec{x})$. The basis $\mathbf{P}^T$ can be any orthogonal basis and any function included in this basis is reproduced exactly (i.e., the approximation is consistent to the k^{th} order if the polynomial basis is chosen and is complete to order k) [3]. After minimizing $\dfrac{\partial E}{\partial \mathbf{a}} = 0$, solving for $\mathbf{a}(\vec{x})$, and substituting, the final approximation reads:

$$u^h(\vec{x}) = \mathbf{P}^T(\vec{x} - \vec{x}_a)\underline{\underline{A}}^{-1}(\vec{x})\underline{\underline{B}}(\vec{x})\mathbf{u} \tag{2.20}$$

with the shape function vector ("MLS interpolants"):

$$\mathbf{\Phi}(\vec{x}) = \begin{bmatrix}\phi_1(\vec{x}) & \phi_2(\vec{x}) & \phi_3(\vec{x}) & \ldots & \phi_n(\vec{x})\end{bmatrix} = \mathbf{P}^T(\vec{x} - \vec{x}_a)\underline{\underline{A}}^{-1}(\vec{x})\underline{\underline{B}}(\vec{x}) \tag{2.21}$$

where the moment matrix is defined as:

$$\underline{\underline{A}}(\vec{x}) = \sum_{i}^{n} W\left(\vec{x} - \vec{x}_i\right)\mathbf{P}(\vec{x}_{ia}) \otimes \mathbf{P}(\vec{x}_{ia}) \tag{2.22}$$

We reiterate that the entries of the moment matrix are now actually functions. Similarly $\underline{\underline{B}}$ (entries also functions) is defined as:

$$\underline{\underline{B}} = \begin{bmatrix} W(\vec{x} - \vec{x}_1)P_1(\vec{x}_{1a}) & W(\vec{x} - \vec{x}_3)P_1(\vec{x}_{3a}) & \ldots & W(\vec{x} - \vec{x}_n)P_1(\vec{x}_{na}) \\ W(\vec{x} - \vec{x}_1)P_2(\vec{x}_{1a}) & W(\vec{x} - \vec{x}_2)P_2(\vec{x}_{2a}) & \ldots & W(\vec{x} - \vec{x}_n)P_2(\vec{x}_{na}) \\ \ldots & \ldots & \ldots & \ldots \\ W(\vec{x} - \vec{x}_1)P_m(\vec{x}_{1a}) & W(\vec{x} - \vec{x}_2)P_m(\vec{x}_{2a}) & \ldots & W(\vec{x} - \vec{x}_n)P_m(\vec{x}_{na}) \end{bmatrix} \tag{2.23}$$

Using numerical tests, Belyschtko et al. showed that not accounting for the spatial variation of the weight function reduces the accuracy of their integral formulation. Accounting for this spatial variation is necessary when using the Galerkin method since the matrices $\underline{\underline{A}}(\vec{x})$ and $\underline{\underline{B}}(\vec{x})$ will vary at different points in the integration region. However, the question remains whether the spatial variation will need to be accounted for when using a collocation technique or if GFD will suffice.

In this work, we investigate a collocated version of MLS where we treat the value at the collocation point u_a as known leading to the removal of the constant basis and correspondingly decreased matrix sizes (as is done with GFD). The collocated MLS approximation using a quadratic basis (constant basis removed) may be obtained as follows:

1. Drop the constant basis in $\mathbf{P}^T(\vec{x}-\vec{x}_a)$ treating u_a as known.

$$u^h(\vec{x}) = u_a + \mathbf{P}^T(\vec{x}-\vec{x}_a)\mathbf{a}(\vec{x}) \tag{2.24}$$

2. Solve for $\mathbf{a}(\vec{x})$ using n neighboring points:

$$\mathbf{a}(\vec{x}) = \underline{\underline{A}}^{-1}(\vec{x})\underline{\underline{B}}(\vec{x})\left(\mathbf{u}-\mathbf{u}_a\right) \tag{2.25}$$

3. Substitute in $\mathbf{a}(\vec{x})$ and differentiate twice:

$$u^h(\vec{x}) = u_a + \mathbf{P}^T(\vec{x}-\vec{x}_a)\mathbf{a}(\vec{x}) \tag{2.26}$$

$$\frac{\partial u^h}{\partial x} = \frac{\partial \mathbf{P}^T}{\partial x}\mathbf{a} + \mathbf{P}^T\frac{\partial \mathbf{a}}{\partial x} \tag{2.27}$$

$$\frac{\partial^2 u^h}{\partial x^2} = \frac{\partial^2 \mathbf{P}^T}{\partial x^2}\mathbf{a} + 2\frac{\partial \mathbf{P}^T}{\partial x}\frac{\partial \mathbf{a}}{\partial x} + \mathbf{P}^T\frac{\partial^2 \mathbf{a}}{\partial x^2} \tag{2.28}$$

4. Evaluating at $\vec{x}_a$, $\mathbf{P}^T(\vec{x}_a-\vec{x}_a)=\vec{0}$ and $\mathbf{a}(\vec{x}_a) = \underline{\underline{A}}^{-1}(\vec{x}_a)\underline{\underline{B}}(\vec{x}_a)(\mathbf{u}-\mathbf{u}_a)$:

$$\left.\frac{\partial u^h}{\partial x}\right|_{\vec{x}_a} = \begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}\mathbf{a}(\vec{x}_a) \tag{2.29}$$

$$\left.\frac{\partial^2 u^h}{\partial x^2}\right|_{\vec{x}_a} = \begin{bmatrix}0 & 0 & 2 & 0 & 0\end{bmatrix}\mathbf{a}(\vec{x}_a) + \begin{bmatrix}2 & 0 & 0 & 0 & 0\end{bmatrix}\left.\frac{\partial \mathbf{a}}{\partial x}\right|_{\vec{x}_a} \tag{2.30}$$

A similar procedure can be followed for other derivatives. Note that the first derivative is identical to the derivative obtained by GFD in Section 3.1, however, for the second derivative an additional term related to the spatial derivatives of the matrices $\underline{\underline{A}}(\vec{x})$ and $\underline{\underline{B}}(\vec{x})$ is included. Moreover, note that if the constant basis was not dropped, for 2^{nd} derivatives an additional term related to $\left.\dfrac{\partial^2 \mathbf{a}}{\partial x^2}\right|_{\vec{x}_a}$ would need to be included while for 1^{st} derivatives the term related to $\left.\dfrac{\partial \mathbf{a}}{\partial x}\right|_{\vec{x}_a}$ would also need to be included. Applying the product rule, the spatial variation of the coefficients can be calculated as follows:

$$\frac{\partial \mathbf{a}}{\partial x} = \left(\frac{\partial \underline{\underline{A}}^{-1}}{\partial x}\underline{\underline{B}} + \underline{\underline{A}}^{-1}\frac{\partial \underline{\underline{B}}}{\partial x}\right)\left(\mathbf{u}-\mathbf{u}_a\right) \tag{2.31}$$

where the spatial derivative of the inverse is found by rearranging $\dfrac{\partial\left(\underline{\underline{A}}^{-1}\underline{\underline{A}}\right)}{\partial x} = \underline{\underline{0}}$:

$$\frac{\partial \underline{\underline{A}}^{-1}}{\partial x} = -\underline{\underline{A}}^{-1}\frac{\partial \underline{\underline{A}}}{\partial x}\underline{\underline{A}}^{-1} \tag{2.32}$$

To approximate the Laplacian of a scalar field we would have the following discretization using a quadratic basis with the constant basis removed:

$$\nabla \cdot \nabla u_a = 2 \left(\sum_i^n \left(\phi_{4i} + \phi_{2i,x} + \phi_{6i} + \phi_{3i,y} \right) u_i - u_a \sum_i^n \left(\phi_{4i} + \phi_{2i,x} + \phi_{6i} + \phi_{3i,y} \right) \right) \tag{2.33}$$

where the following terms correspond to the GFD approximation, with $\underline{\underline{\phi}}$ the $5 \times n$ matrix presented in the Section 3.1 :

$$2 \left(\sum_i^n \left(\phi_{4i} + \phi_{6i} \right) u_i - u_a \sum_i^n \left(\phi_{4i} + \phi_{6i} \right) \right) \tag{2.34}$$

and where the following are additional terms related to MLS:

$$2 \left(\sum_i^n \left(\phi_{2i,x} + \phi_{3i,y} \right) u_i - u_a \sum_i^n \left(\phi_{2i,x} + \phi_{3i,y} \right) \right) \tag{2.35}$$

The additional terms are the appropriate rows of $\underline{\underline{\phi_x}}$ and $\underline{\underline{\phi_y}}$, $m \times n$ matrices related to $\dfrac{\partial}{\partial x}, \dfrac{\partial}{\partial y}$ of $\underline{\underline{A}}^{-1}(\vec{x})\underline{\underline{B}}(\vec{x})$ evaluated at $\vec{x}_a$. For example, $\phi_{2i,x}$ corresponds to the 1^{st} row of the $5 \times n$ matrix $\underline{\underline{\phi_x}}$. The convention used here is that the row index refers to absolute position in the quadratic basis (i.e., the index related to x in $\begin{bmatrix} 1 & x & y & x^2 & xy & y^2 \end{bmatrix}$ will have a value of 2 regardless if the constant basis is dropped). Notice that the polynomial vector simply acts to select (and scale) a given row of the resulting $5 \times n$ matrix. We use Eq. 2.33, to solve the 2D Poisson problem in Chapter 6.

To conclude the section, we must note that Belystchko et al. - citing the original authors - points out the MLS approximation is actually a subset of the Partition of Unity (PU) methods. We will not discuss PU methods here but reference the reader to Belytschko et al. and the papers cited therein [3]. Moreover, we leave exploring the consequences of dropping the constant basis in MLS to future work.

2.6 Smoothed Particle Hydrodynamics (SPH)

Smoothed Particle Hydrodynamics (SPH) was initially introduced by the astrophysics community in 1977 but was later adopted by other communities as an attractive alternative to modeling free surfaces and interfacial flows due to its Lagrangian nature [10]. In Lagrangian methods, nodes in the computational grid are treated as particles occupying a certain volume and having a certain mass with a position which is evolved in time according to Newton's laws (contrary to the Eulerian viewpoint where the grid points are fixed and are treated simply as observation points). Typically in SPH, the governing equations are discretized

using the collocation technique described earlier. In the worst case scenario - due to the evolving nature of the grid - the neighboring particles must be found at the beginning of each time step via a neighbor search algorithm. Here we will not focus on the Lagrangian nature of SPH but instead focus on the underlying function and derivative approximations.

To introduce the approximations behind SPH, we start with a general Taylor series expansion of the field value at a neighboring point $u(\vec{x}_i)$ about the field value at our point of interest $u(\vec{x}_a)$. Here we loosely follow the notation used by Fatehi and Manazari [11].

$$u_i = u_a + \vec{x}_{ia} \cdot \nabla u_a + \frac{1}{2}\vec{x}_{ia} \otimes \vec{x}_{ia} : \nabla\nabla u_a + \frac{1}{6}\vec{x}_{ia} \otimes \vec{x}_{ia} \otimes \vec{x}_{ia} \vdots \nabla\nabla\nabla u_a + \cdots \tag{2.36}$$

where $\otimes$ refers to the tensor product and $\cdot, :, \vdots$ refer to the the appropriate tensor contraction (e.g., for rank 2 tensors $A : B = A_{ij}B_{ij}$). Weighting each term with a function $W(\vec{x} - \vec{x}_i)$, integrating over the domain, and finally discretizing and evaluating at the collocation point:

$$\sum u_i W_{ai} \Delta V_i = u_a \sum_i W_{ai} \Delta V_i + \nabla u_a \cdot \sum_i \vec{x}_{ia} W_{ai} \Delta V_i + \frac{1}{2}\nabla\nabla u_a : \sum_i \vec{x}_{ia} \otimes \vec{x}_{ia} W_{ai} \Delta V_i + \cdots \tag{2.37}$$

where ΔV_i is the volume (3D) or area (2D) assigned to a neighboring node. Here we have pulled u_a, ∇u_a and $\nabla\nabla u_a$ outside the summation over the i neighboring nodes and have additionally commuted the tensor contraction. The weight function is chosen such that it is symmetric, positive and monotonically decreasing with a compact support domain. If we differentiate the Taylor series with respect to $\vec{x}$ before collocation and then collocate we obtain:

$$\sum u_i \nabla W_{ai} \Delta V_i = u_a \sum_i \nabla W_{ai} \Delta V_i + \nabla u_a \cdot \sum_i \vec{x}_{ia} \otimes \nabla W_{ai} \Delta V_i + \frac{1}{2}\nabla\nabla u_a : \sum_i \vec{x}_{ia} \otimes \vec{x}_{ia} \otimes \nabla W_{ai} \Delta V_i \tag{2.38}$$

Here the notation W_{ai} corresponds to the i^{th} weight function evaluated at the collocation point $\vec{x}_a$ (i.e., $W_{ai} = W(\vec{x}_a - \vec{x}_i)$). Similarly, $\nabla W_{ai} = \nabla W(\vec{x} - \vec{x}_i)|_{\vec{x}_a}$ corresponds to the gradient of the i^{th} weight function with respect to $\vec{x}$, evaluated at $\vec{x}_a$. The notation carries over to other variables such as the position, $\vec{x}_{ia} = \vec{x}_i - \vec{x}_a$. For a discussion on weight functions and computing their derivatives see Section 2.6.3. At this point we recognize the truncation error terms as the required "moment conditions" needed to have a desired consistency [6]. In general, these moment conditions are not met even when using a symmetric and normalized

[6]Similar moment conditions can be shown for higher order approximations as well as higher derivatives, in fact these moment conditions are equivalent to the reproducing conditions mentioned in the subsequent sections.

kernel due to potential disorder in particle positions (i.e., "particle inconsistency") as well as support domain truncation near boundaries.

$$\begin{aligned} M_{0F} &= \sum_i W_{ai}\Delta V_i \neq 1 \qquad M_{1F} = \sum_i \vec{x}_{ia} W_{ai}\Delta V_i \neq \vec{0} \\ M_{0G} &= \sum_i \nabla W_{ai}\Delta V_i \neq \vec{0} \quad M_{1G} = \sum_i \vec{x}_{ia} \otimes \nabla W_{ai}\Delta V_i \neq \underline{\underline{I}} \end{aligned} \tag{2.39}$$

The first index refers to the degree of the polynomial which can be reproduced exactly if the condition is met, while the second index refers to whether it is "F", a function approximation or "G", a gradient approximation. In the continuous case, the moment conditions hold leading to second order accurate expressions. For a detailed error analysis of the discrete case, we refer the reader to Fatehi and Manzari and the references therein [11]. If we ignore these error terms, we have the following uncorrected approximations for functions and derivatives:

$$\begin{aligned} \langle u_a \rangle &= \sum_i u_i W_{ai}\Delta V_i \\ \langle \nabla u_a \rangle &= \sum_i u_i \nabla W_{ai}\Delta V_i \end{aligned} \tag{2.40}$$

Notice that the spatial derivatives are simply passed off onto the weight function. In this manner, the weight functions are analogous to shape functions which lack the interpolating property. If we use these uncorrected forms, the numerical solution is not guaranteed to converge to the actual solution due to large numerical errors present wherever the moment conditions are violated.

To reconcile these issues, SPH practitioners introduced various corrections to restore the moment conditions in the discrete setting. As Belytschko et al. points out there are two basic approaches [2]:

1. enforcing reproducing conditions on either the kernel or gradients of the kernel such that desired functions are reproduced exactly

2. manipulation of terms in the Taylor series expansion such that the truncation error in the approximation is of the desired order

We will first introduce Taylor series based approaches and later introduce the approach via reproducing conditions as is done by Beltyschko et al. [2]

2.6.1 Taylor Series Corrections

The promising corrections that have been proposed in SPH literature to correct the gradient $\nabla u(\vec{x})$ and the Laplacian $\nabla \cdot \nabla u(\vec{x})$ can be categorized as either semi-implicit or

implicit. In semi-implicit corrections, the gradient is first approximated considering u_a as known and then normalized by the truncation error term M_{1G} to restore consistency. The gradient estimate is then considered as a known when constructing higher derivatives. In this manner, a better estimate for higher order derivatives may be sequentially obtained. Chen and Beraun called this approach Corrected Smoothed Particle Method (CSPM) [7].

Building on CSPM, Zhang and Batra recognized that if u_a is treated as an unknown the function and derivatives maybe approximated simultaneously [35]. The implications of this approach are that the truncation error for ∇u_a will not propagate into the estimate of higher order terms as occurs in CSPM. Zhang and Batra called this approach Modified Smoothed Particle Hydrodynamics (MSPH). Within a year, Liu et al. arrived at the same approach calling it Finite Particle Method (FPM) [21].

Over a decade later, Fatehi and Manzari proposed a semi-implicit correction to construct a consistent Laplacian[7]. Using Brookshaw's approximation[6] as a starting point, they kept track of the truncation error associated with lower derivatives and included the error when constructing the normalization matrix needed to correct second derivatives. Before demonstrating each of these corrections we summarize the corrections below:

1. *Semi-Implicit corrections*

   (a) CSPM - Excludes same order truncation error stemming from lower derivatives and considers u_a as known. In 2D, this approach involves a 2×2 matrix inversion in order to reproduce the gradient of a linear field, followed by a 3×3 matrix inversion with the corrected gradient now treated as as known leading to an improved but still inconsistent Laplacian.

   (b) Fatehi and Manzari - Includes same order truncation error stemming from lower derivatives and considers u_a as known. In 2D, this approach involves a 2×2 matrix inversion in order to reproduce the gradient of a linear field, followed by a 3×3 matrix inversion where the corrected gradient is treated as known. By including the associated leading error term of the gradient approximation in the construction of the 3×3 matrix, they obtain a consistent Laplacian estimate.

2. *Implicit correction.* (MSPH, FPM) Implicitly includes lower truncation error with u_a considered unknown. In 2D, this involves a 6×6 matrix inversion to arrive at first and second derivatives estimates exact for quadratic functions.

[7]Fatehi and Manzari mention in their paper they were not aware of any consistent second order derivative scheme in SPH literature. We note that at the time of their writing Belytschko et al. and others made it clear over a decade earlier that MLS could be used to restore consistency [3]. Moreover, MSPH or GFD could have been used.

2.6.1.1 Corrected Smoothed Particle Method (CSPM)

Starting with the differentiated Taylor series, dropping the $\nabla\nabla u_a$ error term, and subtracting u_a we have:

$$\sum_i (u_i - u_a) \nabla W_{ai} \Delta V_i = \nabla u_a \cdot \sum_i \vec{x}_{ia} \otimes \nabla W_{ai} \Delta V_i \tag{2.41}$$

In order for the estimate ∇u_a to be exact for constant and linear functions, the 1^{st} gradient moment condition must be equal to $\underline{\underline{I}}$. This can be achieved by taken the inverse of the moment condition and dotting the gradient of the kernel:

$$\left(\sum_i \vec{x}_{ia} \otimes \nabla W_{ai} \Delta V_i \right)^{-1} \cdot \sum_i (u_i - u_a) \nabla W_{ai} \Delta V_i = \nabla u_a \cdot \underline{\underline{I}} \tag{2.42}$$

More compactly we have:

$$\langle \nabla u_a \rangle = \sum_i (u_i - u_a) \tilde{\nabla} W_{ai} \Delta V_i \tag{2.43}$$

where $\tilde{\nabla} W_{ai}$ is the corrected gradient of the kernel:

$$\tilde{\nabla} W_{ai} = M_{1G}^{-1} \cdot \nabla W_{ai} \tag{2.44}$$

where M_{1G} is the symmetric matrix corresponding to 1^{st} gradient moment listed earlier. Here the notation $\langle \nabla u_a \rangle$ is necessary to indicate an estimate since we will later substitute the expression in the Taylor series and it will be necessary to distinguish the estimate of the gradient from the actual gradient at $\vec{x}_a$. Randles and Libersky were the first to propose the above normalization to construct an exact gradient estimate for a linear tensor field [28]. Bonet and Lok point out that the approach is equivalent to enforcing rotational invariance [5]. Lastly, we note the above gradient correction is the first step of CSPM. For convenience we list out the equations for the 2D case:

$$\begin{bmatrix} \sum_i x_{ia} \frac{\partial W}{\partial x} \Delta V_i & \sum_i y_{ia} \frac{\partial W}{\partial x} \Delta V_i \\ \sum_i x_{ia} \frac{\partial W}{\partial y} \Delta V_i & \sum_i y_{ia} \frac{\partial W}{\partial y} \Delta V_i \end{bmatrix} \begin{bmatrix} \frac{\partial u_a}{\partial x} \\ \frac{\partial u_a}{\partial y} \end{bmatrix} = \begin{bmatrix} \sum_i (u_i - u_a) \frac{\partial W}{\partial x} \Delta V_i \\ \sum_i (u_i - u_a) \frac{\partial W}{\partial y} \Delta V_i \end{bmatrix} \tag{2.45}$$

here the notation $\dfrac{\partial W}{\partial x}$ indicates the derivative with respect to x of $W(\vec{x} - \vec{x}_i)$ and then evaluated at $\vec{x}_a$. Notice that if M_{1G}^{-1} is dropped, the gradient approximation reduces to only being exact for constant functions. Furthermore, if we reverse the sign for u_a we arrive

at the popular pairwise symmetric formulation which will in general not even be exact for constant functions. We see there exists a trade off between restoring consistency and preserving pairwise symmetry . Typically (to restore consistency) a unique matrix inversion is necessary at each node since in the general setting node a and neighboring node i will likely have different distribution of neighbors. As a result, pairwise symmetry is broken.

Continuing with the CSPM approach, a system maybe obtained for 2^{nd} derivatives by considering the gradient as known for a second system constructed by integrating the Taylor series with respect to second derivatives of the weight function:

$$\sum (u_{ia})\,\nabla\nabla W_{ai}\Delta V_i - \langle\nabla u_a\rangle\cdot\sum_i \vec{x}_{ia}\otimes\nabla\nabla W_{ai}\Delta V_i = \frac{1}{2}\nabla\nabla u_a : \sum_i \vec{x}_{ia}\otimes\vec{x}_{ia}\otimes\nabla\nabla W_{ai}\Delta V_i + \cdots \tag{2.46}$$

where $\nabla\nabla$ indicates the gradient of the gradient. Dropping the higher order terms, combining equivalent mixed derivatives, a simplified system of equations may be obtained for the second derivatives. In the general case, this approximation will not be able to reproduce second derivatives for a quadratic function since the leading error term dropped in the approximation for $\langle\nabla u_a\rangle$ actually is proportional to $\nabla\nabla u_a$ [11]. This leading error term $\nabla u_a - \langle\nabla u_a\rangle$ is given by:

$$\nabla u_a - \langle\nabla u_a\rangle = -\frac{1}{2}\nabla\nabla u_a : M_{2G}\cdot M_{1G}^{-1} \tag{2.47}$$

where the symmetric second gradient moment tensor is given by:

$$M_{2G} = \sum_i \vec{x}_{ia}\otimes\vec{x}_{ia}\otimes\nabla W_{ai}\Delta V_i \tag{2.48}$$

We leave expanding the compact form to the reader but note that it results in a 3×3 system (in 2D) for the second derivatives. Furthermore, while the approximation is not fully consistent it is still improved in the sense it no longer has sources of error proportional to the lower order terms. We next explain Fatehi and Manzari's approach in more detail in the following sections.

2.6.1.2 Fatehi and Manzari Approach

The starting point in their approximation is the Brookshaw approximation [6], where the first gradient is approximated using a finite difference while the second gradient is passed off onto the weight function[8]:

$$\langle\nabla\nabla u_a\rangle = 2\sum_i \frac{u_i - u_a}{|\vec{x}_{ia}|}\vec{e}_{ia}\otimes\nabla W_{ai} \tag{2.49}$$

[8]Fatehi and Manzari actually introduce $\nabla\cdot\nabla u_a$ first but it is clear from their work that they construct their final approximation actually starting from $\nabla\nabla u_a$.

where $|\vec{x}_{ia}|$ is the distance between node i and node a while $\vec{e}_{ia} = \dfrac{\vec{x}_{ia}}{|\vec{x}_{ia}|}$ is the unit vector between them. Expanding u_i using a Taylor series and absorbing the magnitude:

$$\langle \nabla\nabla u_a \rangle = 2\sum_i \left(u_a + \vec{e}_{ia} \cdot \nabla u_a + \frac{1}{2}\vec{x}_{ia} \otimes \vec{e}_{ia} : \nabla\nabla u_a + \cdots - u_a \right) \vec{e}_{ia} \otimes \nabla W_{ai} \tag{2.50}$$

Simplifying using the identities $(\vec{a}\cdot\vec{b})\vec{c} = \vec{b}\cdot(\vec{a}\otimes\vec{c})$ and $(\vec{a}\otimes\vec{b} : \underline{\underline{B}})\vec{c}\otimes\vec{d} = \underline{\underline{B}} : (\vec{a}\otimes\vec{b}\otimes\vec{c}\otimes\vec{d})$ we have:

$$\langle \nabla\nabla u_a \rangle = 2\nabla u_a \cdot \sum_i \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \nabla W_{ai} + \nabla\nabla u_a : \sum_i \vec{x}_{ia} \otimes \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \nabla W_{ai} + \cdots \tag{2.51}$$

As in CSPM, the leading error term is approximated using the corrected gradient $\langle \nabla u_a \rangle$. Subtracting $2\langle \nabla u_a \rangle \cdot \sum_i \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \nabla W_{ai}$ we have:

$$\langle \nabla\nabla u_a \rangle^* = 2\left(\nabla u_a - \langle \nabla u_a \rangle\right) \cdot \sum_i \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \nabla W_{ai} + \nabla\nabla u_a : \sum_i \vec{x}_{ia} \otimes \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \nabla W_{ai} + \cdots \tag{2.52}$$

where $\langle \nabla\nabla u_a \rangle^* = \langle \nabla\nabla u_a \rangle - 2\langle \nabla u_a \rangle \cdot \sum_i \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \nabla W_{ai}$. The novelty of Fatehi and Manzari's approach occurs at this point in the process. Instead of ignoring error from $\nabla u_a - \langle \nabla u_a \rangle$ as is done in CSPM, they instead calculate the leading truncation error term for the gradient as given by Equation 2.47. The idea is that the corrected gradient's leading error term will contribute a term proportional to $\nabla\nabla u_a$ and therefore must be considered during the construction process. After substitution of Equation 2.47 we have:

$$\langle \nabla\nabla u_a \rangle^* = \left(-\nabla\nabla u_a : M_{2G} \cdot M_{1G}^{-1}\right) \cdot \sum_i \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \nabla W_{ai} + \nabla\nabla u_a : \sum_i \vec{x}_{ia} \otimes \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \nabla W_{ai} + \cdots \tag{2.53}$$

Grouping terms in front of $\nabla\nabla u_a$ and rewriting in terms of moment gradient tensors we have:

$$\langle \nabla\nabla u_a \rangle^* = \nabla\nabla u_a : \underline{\underline{\underline{C}}} + \cdots \tag{2.54}$$

where $\underline{\underline{\underline{C}}}$ is a fully symmetric fourth order tensor given by:

$$\underline{\underline{\underline{C}}} = \left(\frac{1}{|\vec{x}_{ia}|^2} \left(M_{3G} - M_{2G} \cdot M_{1G}^{-1} \cdot M_{2G} \right) \right) \tag{2.55}$$

where M_{3G} is the third gradient moment tensor defined as:

$$M_{3G} = \sum_i \vec{x}_{ia} \otimes \vec{x}_{ia} \otimes \vec{x}_{ia} \otimes \nabla W_{ai} \tag{2.56}$$

To see that $\underline{\underline{\underline{C}}}$ is fully symmetric, note that $\nabla W_{ai} = -\vec{e}_{ia}\dfrac{\partial W}{\partial r}$ and $\vec{x}_{ia} = |\vec{x}_{ia}|\vec{e}_{ia}$. Leading the moment gradient tensors M_{1G}, M_{2G} and M_{3G} to be the summation of individual fully symmetric tensors constructed from some scalar multiple of $\vec{e}_{ia} \otimes \vec{e}_{ia}$, $\vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \vec{e}_{ia}$, and $\vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \vec{e}_{ia}$, respectively. Furthermore, the left and right product of M_{1G}^{-1} in $M_{2G} \cdot M_{1G}^{-1} \cdot M_{2G}$ will contract the third and first indexes of each M_{2G} into ultimately some scalar in front of $\vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \vec{e}_{ia} \otimes \vec{e}_{ia}$, leading to the final product to also simply be the summation of individually fully symmetric fourth order tensors. Now that the full symmetry of $\underline{\underline{\underline{C}}}$ is established we look for a second order tensor $\underline{\underline{B}}$ such that $\underline{\underline{\underline{C}}} : \underline{\underline{B}} = \underline{\underline{I}}$ since $\nabla\nabla u_a : \underline{\underline{I}} = \nabla \cdot \nabla u_a$.

$$\langle \nabla\nabla u_a \rangle^* : \underline{\underline{B}} = \nabla\nabla u_a : \underline{\underline{\underline{C}}} : \underline{\underline{B}} \tag{2.57}$$

Fatehi and Manzari state that $\underline{\underline{B}}$ is symmetric but do not explain why. Consider the 4×4 system in 2D given by $C_{ijkl}B_{kl} = I_{ij}$. Expanding we have:

$$\begin{bmatrix} C_{1111} & C_{1112} & C_{1121} & C_{1122} \\ C_{1211} & C_{1212} & C_{1221} & C_{1222} \\ C_{2111} & C_{2112} & C_{2121} & C_{2122} \\ C_{2211} & C_{2212} & C_{2221} & C_{2222} \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.58}$$

Using the full symmetry of C_{ijkl} we reduce the 16 components to only 5 potentially unique components:

$$\begin{bmatrix} c_1 & c_2 & c_1 & c_3 \\ c_2 & c_3 & c_3 & c_4 \\ c_2 & c_3 & c_3 & c_4 \\ c_3 & c_4 & c_4 & c_5 \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.59}$$

This leads to a singular system since row 2 and row 3 are identical. Choosing the free variable to be B_{21} and by setting it equal to B_{12} we reduce the system to a 3×3 system where $\underline{\underline{B}}$ is now symmetric:

$$\begin{bmatrix} c_1 & c_2 & c_3 \\ c_2 & c_3 & c_4 \\ c_3 & c_4 & c_5 \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \tag{2.60}$$

where $c_1 = C_{1111}, c_2 = C_{1112}, c_3 = C_{1122}, c_4 = C_{1222}$ and $c_5 = C_{2222}$. Once $\underline{\underline{B}}$ is found, the final corrected approximation for the Laplacian reads:

$$\langle \nabla \cdot \nabla u_a \rangle = \underline{\underline{B}} : 2\sum_i \left(\frac{u_i - u_a}{|\vec{x}_{ia}|} - \vec{e}_{ia} \cdot \langle \nabla u_a \rangle \right) \vec{e}_{ia} \otimes \nabla W_{ai} \tag{2.61}$$

We do not consider this scheme here although it is consistent for second derivatives. We simply present their unique approach and clarify some details left out of the original paper.

We refer the reader to numerical results presented by Fatehi and Manzari or more recently Trask et al., who applied it to solve the Poisson equation [32].

2.6.1.3 Modified Smoothed Particle Hydrodynamics

Building on CSPM, Zhang and Batra introduced a strictly implicit procedure for obtaining consistent estimates [35]. By simultaneously solving for derivatives in one step, they bypassed the issue in CSPM where truncation errors from lower derivative estimates propagate into the estimates of higher derivatives. The MSPH approximation starts from integrating the Taylor series with respect to the weight function and its derivatives. The system of equations for u_a and ∇u_a reads:

$$\begin{aligned} u_a \sum_i W_{ai} \Delta V_i + \nabla u_a \cdot \sum_i \vec{x}_{ia} W_{ai} \Delta V_i &= \sum u_i W_{ai} \Delta V_i \\ u_a \sum_i \nabla W_{ai} \Delta V_i + \nabla u_a \cdot \sum_i \vec{x}_{ia} \otimes \nabla W_{ai} \Delta V_i &= \sum u_i \nabla W_{ai} \Delta V_i \end{aligned} \tag{2.62}$$

To obtain u_a and ∇u_a either a 3×3 system (2D) or a 4×4 (3D) would need to be inverted. The system can be written as follows where the matrix entries are the moment conditions M_{0F}, M_{1F}, M_{0G}, and M_{1G}:

$$\begin{bmatrix} \sum_i W_{ai} \Delta V_i & \left(\sum_i \vec{x}_{ia} W_{ai} \Delta V_i \right)^T \\ \sum_i \nabla W_{ai} \Delta V_i & \sum_i \vec{x}_{ia} \otimes \nabla W_{ai} \Delta V_i \end{bmatrix} \begin{bmatrix} u_a \\ \nabla u_a \end{bmatrix} = \begin{bmatrix} \sum u_i W_{ai} \Delta V_i \\ \sum u_i \nabla W_{ai} \Delta V_i \end{bmatrix} \tag{2.63}$$

The above matrix approximates the appropriately sized identity matrix. We see if u_a is considered as known, then the system collapses to a 2×2 (2D) or 3×3 (3D) system which is equivalent to the corrected gradient approximation first introduced by Randles and Libersky and later introduced by Chen and Beraun in CSPM under the context of a Taylor series expansion. To approximate 2^{nd} derivatives consistently, the unknowns u_a, ∇u_a, and $\nabla \nabla u_a$ may be found by integrating the Taylor series with respect to $W, \nabla W$ and $\nabla \nabla W$ resulting in the following 6×6 (2D) or 10×10 (3D) system:

$$\begin{aligned} u_a \sum_i W_{ai} + \nabla u_a \cdot \sum_i \vec{x}_{ia} W_{ai} + \frac{1}{2} \nabla \nabla u_a : \sum_i \vec{x}_{ia} \otimes \vec{x}_{ia} W_{ai} &= \sum_i u_i W_{ai} \\ u_a \sum_i \nabla W_{ai} + \nabla u_a \cdot \sum_i \vec{x}_{ia} \otimes \nabla W_{ai} + \frac{1}{2} \nabla \nabla u_a : \sum_i \vec{x}_{ia} \otimes \vec{x}_{ia} \otimes \nabla W_{ai} &= \sum_i u_i \nabla W_{ai} \\ u_a \sum_i \nabla \nabla W_{ai} + \nabla u_a \cdot \sum_i \vec{x}_{ia} \otimes \nabla \nabla W_{ai} + \frac{1}{2} \nabla \nabla u_a : \sum_i \vec{x}_{ia} \otimes \vec{x}_{ia} \otimes \nabla \nabla W_{ai} &= \sum_i u_i \nabla \nabla W_{ai} \end{aligned} \tag{2.64}$$

where the mixed derivatives in $\nabla\nabla$ need to be combined before the system can be formed and where we have dropped ΔV_i for convenience. We do not consider this scheme here as the matrices are asymmetric, thereby requiring more storage. Additionally, computing the matrix entries involves not only evaluating the weight function at each point but also the first and second partial derivatives of the weight functions at each point.

2.6.1.4 Brookshaw Corrected Gradient (BCG)

A popular form to estimate second derivatives is Brookshaw's approximation [6]. In Brookshaw's approximation, $\frac{(u_i - u_a)}{|\vec{x}_{ia}|}\vec{e}_{ia}$ approximates the first gradient followed by passing the second gradient onto the weight function. Some authors have gone a step further and combined this form with the gradient correction for the weight function to obtain the following Laplacian estimate:

$$\langle \nabla \cdot \nabla u_a \rangle = 2 \sum_i \frac{u_i - u_a}{|\vec{x}_{ia}|} \vec{e}_{ia} \cdot \tilde{\nabla} W_{ai} \tag{2.65}$$

where $\tilde{\nabla} W_{ai}$ is given by Eq. 2.44. We call this form Brookshaw Corrected Gradient (BCG). If we expand u_i via the Taylor series given by Equation 2.36:

$$\langle \nabla \cdot \nabla u_a \rangle = 2 \sum_i \frac{\nabla u_a \cdot \vec{x}_{ia} + 1/2 \nabla\nabla u_a : (\vec{x}_{ia} \otimes \vec{x}_{ia})}{|\vec{x}_{ia}|} \vec{e}_{ia} \cdot \tilde{\nabla} W_{ai} \tag{2.66}$$

Rearranging and simplifying:

$$\langle \nabla \cdot \nabla u_a \rangle = 2 \nabla u_a \cdot \sum_i \tilde{\nabla} W_{ai} + \nabla\nabla u_a : \sum_i \vec{x}_{ia} \otimes \tilde{\nabla} W_{ai} \tag{2.67}$$

where $\langle \nabla \cdot \nabla u_a \rangle - \nabla \cdot \nabla u_a$:

$$\langle \nabla \cdot \nabla u_a \rangle - \nabla \cdot \nabla u_a = 2 \nabla u_a \cdot \sum_i \tilde{\nabla} W_{ai} + \nabla\nabla u_a : \left(\sum_i \vec{x}_{ia} \otimes \tilde{\nabla} W_{ai} - \underline{\underline{I}} \right) \tag{2.68}$$

The leading error term is proportional to $2\nabla u_a \cdot \sum_i \tilde{\nabla} W_{ai}$ which is not equal to $\vec{0}$ unless a regular grid is used. Moreover, note that even if an irregular grid is used, since the kernel is corrected using the 2×2 (2D) or 3×3 (3D) matrix M_{1G}^{-1}, the term $\sum_i \vec{x}_{ia} \otimes \tilde{\nabla} W_{ai} = \underline{\underline{I}}$. As a result, we will not have the error term proportional to $\nabla\nabla u_a$. Khorasanizade et al. used this Laplacian estimate when discretizing the Poisson equation in their incompressible Lagrangian solver [17]. Despite the inconsistency, their numerical tests for lid driven cavity solution at $Re = 100, 400, 1000, 3200$ had excellent agreement with the Ghia data set. Based on their results, we decide to explore the BCG approximation further in Chapter 6, although it should be clear based on the above that the approximation is not consistent.

2.6.2 Reproducing Corrections

Belytschko et al. describes how reproducing conditions can be used to restore consistency of the function or gradient approximation by modifying the kernel (i.e., weight function) or gradient of the kernel[2]. In fact, if the reproducing conditions are used to reproduce the polynomial basis, they will be equivalent to the moment conditions arrived at earlier using the Taylor series approach. As a result, we should expect some of the previous approximations to reappear in the following sections. Following Belytschko et al., we first introduce the polynomial reproducing conditions.

2.6.2.1 Polynomial Reproducing Conditions

In terms of shape functions, the function approximation can be written as:

$$u^h(\vec{x}) = \sum_i \phi_i(\vec{x}) u(\vec{x}_i) \tag{2.69}$$

where in the case of SPH, $\phi_i(\vec{x}) = W(\vec{x}-\vec{x}_i)\Delta V_i$ with the shape functions generally no longer having the interpolating property. The reproducing conditions require that a particular set of linearly independent functions are reproduced exactly by the approximation $u^h(\vec{x})$. Typically, the polynomial reproducing condition is enforced where the set of functions to be reproduced will be a linear combination of the polynomial basis (e.g., 1D quadratic basis $[1 \quad x \quad x^2]$). Take for example the constant reproducing condition which requires that any constant function be reproduced exactly. Substituting $u^h(\vec{x}) = c_o$ and $u(\vec{x}_i) = c_o$ into Eq. 2.69:

$$c_o = \sum_i \phi_i(\vec{x}) c_o \quad \rightarrow \quad \sum_i \phi_i(\vec{x}) = 1 \tag{2.70}$$

Here it is no coincidence that we have arrived at the zeroth moment condition M_{0F} or more commonly known as the unity condition. In fact, it should be clear that the polynomial reproducing condition used with $\phi_i(\vec{x}) = W(\vec{x} - \vec{x}_i)\Delta V_i$, will be equivalent to the function moment conditions (e.g., $M_{0F}, M_{1F}, M_{2F}, ...M_{mF}$) introduced earlier in the weighted Taylor series based approach. To see this insert the general form of a polynomial shifted about $\vec{x}_a$ for $u^h(x)$ and $u(\vec{x}_i)$ into Eq. 2.69:

$$\sum_m c_m P_m(\vec{x}) = \sum_i \phi_i(\vec{x}) \left(\sum_m c_m P_m(\vec{x}_i) \right) \tag{2.71}$$

where $P_m(\vec{x})$ is the m^{th} polynomial basis shifted about $\vec{x}_a$. Rearranging we arrive at:

$$\sum_m c_m P_m(\vec{x}) = \sum_m c_m \left(\sum_i \phi_i(\vec{x}) P_m(\vec{x}_i) \right) \tag{2.72}$$

which gives us the reproducing conditions to recover a m^{th} order polynomial:

$$P_m(\vec{x}) = \sum_i \phi_i(\vec{x}) P_m(\vec{x}_i) \tag{2.73}$$

Taking the gradient we arrive at the gradient reproducing conditions:

$$\nabla P_m(\vec{x}) = \sum_i \nabla\phi_i(\vec{x}) P_m(\vec{x}_i) \tag{2.74}$$

When evaluated at the collocation point $\vec{x}_a$, the previously seen moment conditions on functions and gradients $(M_{0F}, M_{1F}, M_{0G}, M_{1G})$ are recovered as given by Eq. 2.39.

2.6.2.2 Kernel Correction

The simplest correction is normalizing by the unity condition as is done in the Shepard interpolant $\phi_i^S(\vec{x})$. By enforcing the unity condition, zeroth order consistency is restored to both the function and derivative approximations [30]:

$$\phi_i^S(\vec{x}) = \frac{\phi_i(\vec{x})}{\sum_j \phi_j(\vec{x})} = \frac{W(\vec{x} - \vec{x}_i)\Delta V_i}{\sum_j W(\vec{x} - \vec{x}_j)\Delta V_j} \tag{2.75}$$

Here the index j still refers to same set of local neighboring nodes. Upon substituting and evaluating at $\vec{x}_a$ the function approximation reads:

$$u_a = \frac{\sum_i u_i W_{ai} \Delta V_i}{\sum_j W_{aj} \Delta V_j} \tag{2.76}$$

Here we see if $u_i = c_0$, the constant can be pulled out and the two sums will always cancel, reproducing constant functions exactly. Moreover if we take the gradient before collocation:

$$\nabla\phi_i^S = \frac{\nabla\phi_i \sum_j \phi_j - \phi_i \sum_j \nabla\phi_j}{\left(\sum_j \phi_j\right)^2} \tag{2.77}$$

and upon substitution we arrive at the derivative estimate:

$$\nabla u_a = \frac{\sum_i u_i \nabla\phi_i \sum_j \phi_j - \sum_i u_i \phi_i \sum_j \nabla\phi_j}{\left(\sum_j \phi_j\right)^2} \tag{2.78}$$

We see that for a constant field the derivative will also be computed exactly since the numerator terms will cancel. The Shepard interpolant can be interpreted as a correction to the kernel function by a constant function which normalizes the estimate. Lancaster and Salkauskas point out that the Shepard interpolant is identical to the constant basis MLS approximation [18]. Moreover, this means the non-corrected SPH estimates given by Eq. 2.40 are in fact a constant basis MLS approximation where the normalizing function in the denominator has been dropped and as a result the approximation is incapable of even reproducing constant functions.

The approach of modifying the weight function by a constant function can be generalized to a correction function comprised of a linear combination of unknown functions. For example in 2D the linearly corrected kernel (denoted by the tilde) would be:

$$\tilde{\phi}_i(\vec{x}) = (\alpha_1(\vec{x}) + \alpha_2(\vec{x})(x_i - x_a) + \alpha_3(\vec{x})(y_i - y_a)\phi_i(\vec{x}) \tag{2.79}$$

where if we wish for the corrected shape functions to reproduce linear functions we would have:

$$\sum_i \tilde{\phi}_i = 1 \quad \sum_i \tilde{\phi}_i x_{ia} = 0 \quad \sum_i \tilde{\phi}_i y_{ia} = 0 \tag{2.80}$$

After enforcing the linear reproducing conditions, a system maybe obtained for the unknown functions α_1, α_2 and α_3 which correct $\phi_i(\vec{x})$ such that reproducing conditions are enforced. Ultimately the corrected kernels are identical to the MLS shape functions when $\Delta V_i = 1$, see Reference [3]. This is in fact the approach introduced by Liu et al. as the Reproducing Kernel Particle Method (RPKM) [22]. Consequently, the m^{th} order consistent SPH approximation - obtained by correcting the weight function to enforce the reproducing (moment) conditions on the kernel - is simply the m^{th} order MLS approximation either obtained by solving the least squares problem or by enforcing the reproducing conditions on the weight function. To estimate the k^{th} derivative, the function approximation would simply be differentiated k times before collocation as was shown in Section 2.5.

2.6.2.3 Mixed Kernel and Gradient Corrections

So far the reproducing conditions have been placed on the weight function, leading to the MLS approximation. Alternatively, reproducing conditions may instead be placed on a combination of the kernel and the gradient of the weight function as was done by Belytschko et al. and Bonet and Lok [2, 5]. Bonet and Lok proposed a single approach they called a "mixed kernel and gradient correction" while Belytschko et al. refered to the overall approach as simply "derivative corrections", proposing three schemes. To start, the linear gradient

reproducing conditions are:

$$\begin{aligned} \sum_i \nabla\phi_i &= \vec{0} \\ \sum_i \vec{x}_{ia} \otimes \nabla\phi_i &= \underline{\underline{I}} \end{aligned} \tag{2.81}$$

Here we will go over the three approaches - Method 1, Method 2, and Method 3 - proposed by Belytschko et al. We will focus on linear reproducing conditions for the function and gradient approximations but it should be straightforward from the examples to arrive at higher order reproducing conditions as well as construct higher derivative estimates. We will show Method 2 is equivalent to the GFD approximation while Method 3 is equivalent to the MSPH approximation, which will provides us with an additional interpretation of GFD and MSPH.

Method 1

In Method 1, the constant reproducing conditions on derivatives are ignored. Instead the constant reproducing condition is enforced by replacing ϕ_i with the shepard interpolant ϕ_i^S, thereby sidestepping the first two conditions (2D). The remaining reproducing conditions are now written as:

$$\sum_i \vec{x}_{ia} \otimes \nabla\phi_i^S = \underline{\underline{I}} \tag{2.82}$$

The corrected gradient of the shepard interpolant $\tilde{\nabla}\phi_i^S$ is then constructed as a linear combination of unknown correction functions applied to the uncorrected shepard gradient $\nabla\phi_i^S$:

$$\tilde{\nabla}\phi_i^S = \underline{\underline{\alpha}} \cdot \nabla\phi_i^S \tag{2.83}$$

After requiring the reproducing conditions given by Eq. 2.82 to hold on $\tilde{\nabla}\phi_i^S$ we arrive at the following sets of symmetric systems:

$$\underline{\underline{\alpha}} \cdot \sum_i \vec{x}_{ia} \otimes \nabla\phi_i^S = \underline{\underline{I}} \tag{2.84}$$

After solving for the unknown correction functions $\underline{\underline{\alpha}}$, the following gradient approximation will be exact for linear functions:

$$\nabla u_a = \left(\sum_i \vec{x}_{ia} \otimes \nabla\phi_i^S \right)^{-1} \cdot \sum_i u_i \nabla\phi_i^S \tag{2.85}$$

Note that Method 1 is equivalent to the approach presented in Bonet and Lok as the mixed kernel and gradient approach [5]. Moreover, Belytschko et al. points out Method 1 differs from the Randles and Libersky gradient approximation (Equation 2.43) simply by the choice of weight function, which no longer requires the error term associated with u_a to be subtracted in order for the derivative approximation to be exact for constant functions.

Method 2

Here Belytschko et al. proposes looking for correction functions for both the kernel and derivatives using the shepard interpolant. The linear 2D example given for the corrected kernels and kernel derivatives is:

$$\begin{aligned}
\tilde{\phi}_i &= (\alpha_{11}(\vec{x}) + \alpha_{12}(\vec{x})x_{ia} + \alpha_{13}(\vec{x})y_{ia})\phi_i^S(\vec{x}) \\
\frac{\partial\tilde{\phi}_i}{\partial x} &= (\alpha_{21}(\vec{x}) + \alpha_{22}(\vec{x})x_{ia} + \alpha_{23}(\vec{x})y_{ia})\phi_i^S(\vec{x}) \\
\frac{\partial\tilde{\phi}_i}{\partial y} &= (\alpha_{31}(\vec{x}) + \alpha_{32}(\vec{x})x_{ia} + \alpha_{33}(\vec{x})y_{ia})\phi_i^S(\vec{x})
\end{aligned} \tag{2.86}$$

Where the basis evaluated at point $\vec{x}_i$ about point $\vec{x}_a$ is included (i.e., x_{ia}, y_{ia}) when constructing the approximation. The reproducing conditions require:

$$\begin{bmatrix}
\sum_i \phi_i^S & \sum_i \frac{\partial\phi_i^S}{\partial x} & \sum_i \frac{\partial\phi_i^S}{\partial y} \\
\sum_i x_{ia}\phi_i^S & \sum_i x_{ia}\frac{\partial\phi_i^S}{\partial x} & \sum_i x_{ia}\frac{\partial\phi_i^S}{\partial y} \\
\sum_i y_{ia}\phi_i^S & \sum_i y_{ia}\frac{\partial\phi_i^S}{\partial x} & \sum_i y_{ia}\frac{\partial\phi_i^S}{\partial y}
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.87}$$

After imposing the above conditions on $\tilde{\phi}_i, \frac{\partial\tilde{\phi}_i}{\partial x}, \frac{\partial\tilde{\phi}_i}{\partial y}$, the following symmetric system is obtained which may be solved for the correction functions:

$$\begin{bmatrix}
\sum_i \phi_i^S & \sum_i x_{ia}\phi_i^S & \sum_i y_{ia}\phi_i^S \\
\sum_i x_{ia}\phi_i^S & \sum_i x_{ia}^2\phi_i^S & \sum_i x_{ia}y_{ia}\phi_i^S \\
\sum_i y_{ia}\phi_i^S & \sum_i x_{ia}y_{ia}\phi_i^S & \sum_i y_{ia}^2\phi_i^S
\end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} \\ \alpha_{12} & \alpha_{22} & \alpha_{32} \\ \alpha_{13} & \alpha_{33} & \alpha_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.88}$$

Notice that the linearly corrected kernel is obtained again via the first row and correction functions $\alpha_{11}, \alpha_{12}, \alpha_{13}$. However, instead of differentiating the linearly corrected kernel to obtain derivatives, the linearly corrected derivatives are obtained by imposing the gradient reproducing conditions on the above linear combinations. After collocation, Method 2 will be identical to GFD when $\phi_i = W(\vec{x} - \vec{x}_i)$ is used as opposed to ϕ_i^S. Note that the correction functions evaluated at $\vec{x}_a$ are equivalent to the inverse of the symmetric moment matrix for GFD presented in Section 3.1:

$$\underline{\underline{\alpha}}(\vec{x}_a) = \underline{\underline{A}}^{-1} \tag{2.89}$$

To see that the above formulation will recover GFD, substitute the correction functions evaluated at $\vec{x}_a$ into the approximation given by Eq. 2.86, evaluated at $\vec{x}_a$:

$$\begin{bmatrix} u_a \\ \dfrac{\partial u_a}{\partial x} \\ \dfrac{\partial u_a}{\partial y} \end{bmatrix} = \begin{bmatrix} \displaystyle\sum_i u_i \tilde{\phi}_i \\ \displaystyle\sum_i u_i \frac{\partial \tilde{\phi}_i}{\partial x} \\ \displaystyle\sum_i u_i \frac{\partial \tilde{\phi}_i}{\partial y} \end{bmatrix} = \begin{bmatrix} \alpha_{11} \displaystyle\sum_i u_i \phi_i + \alpha_{12} \sum_i u_i x_{ia} \phi_i + \alpha_{13} \sum_i u_i y_{ia} \phi_i \\ \alpha_{21} \displaystyle\sum_i u_i \phi_i + \alpha_{22} \sum_i u_i x_{ia} \phi_i + \alpha_{23} \sum_i u_i y_{ia} \phi_i \\ \alpha_{31} \displaystyle\sum_i u_i \phi_i + \alpha_{32} \sum_i u_i x_{ia} \phi_i + \alpha_{33} \sum_i u_i y_{ia} \phi_i \end{bmatrix} \tag{2.90}$$

where we split the vector into a matrix-vector product:

$$\begin{bmatrix} \alpha_{11} \displaystyle\sum_i u_i \phi_i + \alpha_{12} \sum_i u_i x_{ia} \phi_i + \alpha_{13} \sum_i u_i y_{ia} \phi_i \\ \alpha_{21} \displaystyle\sum_i u_i \phi_i + \alpha_{22} \sum_i u_i x_{ia} \phi_i + \alpha_{23} \sum_i u_i y_{ia} \phi_i \\ \alpha_{31} \displaystyle\sum_i u_i \phi_i + \alpha_{32} \sum_i u_i x_{ia} \phi_i + \alpha_{33} \sum_i u_i y_{ia} \phi_i \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \begin{bmatrix} \displaystyle\sum_i u_i \phi_i \\ \displaystyle\sum_i u_i x_{ia} \phi_i \\ \displaystyle\sum_i u_i y_{ia} \phi_i \end{bmatrix} \tag{2.91}$$

taking $\phi_i = W(\vec{x}_a - \vec{x}_i) = W_{ai}$ and using the symmetric matrix of what are now correction coefficients $\underline{\underline{\alpha}} = \underline{\underline{A}}^{-1}$ we arrive at the GFD approximation. Note the LHS is simply the vector of Taylor coefficients $\mathbf{a}$ (constant basis included), while the RHS vector can be rewritten in terms of $\underline{\underline{B}}\mathbf{u}$ (see Section 3.1):

$$\mathbf{a}(\vec{x}_a) = \underline{\underline{A}}^{-1} \underline{\underline{B}} \mathbf{u} \tag{2.92}$$

Method 3

In the last method, the linearly corrected kernels and kernel derivatives are written as the following linear combination of correction functions:

$$\begin{aligned} \tilde{\phi}_i &= \beta_{11}(\vec{x}) \phi_i^S + \beta_{12}(\vec{x}) \frac{\partial \phi_i^S}{\partial x} + \beta_{13}(\vec{x}) \frac{\partial \phi_i^S}{\partial y} \\ \frac{\partial \tilde{\phi}_i}{\partial x} &= \beta_{21}(\vec{x}) \phi_i^S + \beta_{22}(\vec{x}) \frac{\partial \phi_i^S}{\partial x} + \beta_{23}(\vec{x}) \frac{\partial \phi_i^S}{\partial y} \\ \frac{\partial \tilde{\phi}_i}{\partial y} &= \beta_{31}(\vec{x}) \phi_i^S + \beta_{32}(\vec{x}) \frac{\partial \phi_i^S}{\partial x} + \beta_{33}(\vec{x}) \frac{\partial \phi_i^S}{\partial y} \end{aligned} \tag{2.93}$$

Compared to Method 2, the linear combination no longer has a term proportional to the polynomial terms in its construction. The following system for the correction functions can be obtained by similarly enforcing the linear reproducing conditions given by Equation 2.87

on $\tilde{\phi}_i, \dfrac{\partial \tilde{\phi}_i}{\partial x}, \dfrac{\partial \tilde{\phi}_i}{\partial y}$:

$$\begin{bmatrix} \sum_i \phi_i^S & \sum_i \dfrac{\partial \phi_i^S}{\partial x} & \sum_i \dfrac{\partial \phi_i^S}{\partial y} \\ \sum_i x_{ia}\phi_i^S & \sum_i x_{ia}\dfrac{\partial \phi_i^S}{\partial x} & \sum_i x_{ia}\dfrac{\partial \phi_i^S}{\partial y} \\ \sum_i y_{ia}\phi_i^S & \sum_i y_{ia}\dfrac{\partial \phi_i^S}{\partial x} & \sum_i y_{ia}\dfrac{\partial \phi_i^S}{\partial y} \end{bmatrix} \begin{bmatrix} \beta_{11} & \beta_{21} & \beta_{31} \\ \beta_{12} & \beta_{22} & \beta_{32} \\ \beta_{13} & \beta_{33} & \beta_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.94}$$

As we showed that Method 2 collocated is actually is just the GFD approximation, similarly we can show that replacing the Shepard interpolant above with $\phi_i = W(\vec{x} - \vec{x}_i)$ and then collocating will lead to Zhang and Batra's MSPH approximation. The linearly corrected approximations after collocation reads:

$$\begin{bmatrix} u_a \\ \dfrac{\partial u_a}{\partial x} \\ \dfrac{\partial u_a}{\partial y} \end{bmatrix} = \begin{bmatrix} \beta_{11}\sum_i u_i\phi_i + \beta_{12}\sum_i u_i\dfrac{\partial \phi_i}{\partial x} + \beta_{13}\sum_i u_i\dfrac{\partial \phi_i}{\partial y} \\ \beta_{21}\sum_i u_i\phi_i + \beta_{22}\sum_i u_i\dfrac{\partial \phi_i}{\partial x} + \beta_{23}\sum_i u_i\dfrac{\partial \phi_i}{\partial y} \\ \beta_{31}\sum_i u_i\phi_i + \beta_{32}\sum_i u_i\dfrac{\partial \phi_i}{\partial x} + \beta_{33}\sum_i u_i\dfrac{\partial \phi_i}{\partial y} \end{bmatrix} = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \beta_{31} & \beta_{32} & \beta_{33} \end{bmatrix} \begin{bmatrix} \sum_i u_i\phi_i \\ \sum_i u_i\dfrac{\partial \phi_i}{\partial x} \\ \sum_i u_i\dfrac{\partial \phi_i}{\partial y} \end{bmatrix} \tag{2.95}$$

Noting that the system for the correction functions given by Eq. 2.94 reads:

$$\begin{aligned} \underline{\underline{A}}_3 \underline{\underline{\beta}}^T &= \underline{\underline{I}} \\ \underline{\underline{\beta}} &= \left(\underline{\underline{A}}_3^T\right)^{-1} \end{aligned} \tag{2.96}$$

where we have defined $\underline{\underline{A}}_3$ to be the moment matrix associated with Method 3 given in Equation 2.94. Taking the transpose, $\underline{\underline{A}}_3^T$ is equivalent to the linear moment matrix defined in Equation 2.63. Hence, Method 3 implicitly approximates the Taylor series coefficients and is equivalent to MSPH. It is no coincidence that enforcing the reproducing conditions leads us to MSPH. The equivalence between the two approaches is due to the fact the truncation error terms (i.e., moment conditions) in the integrated Taylor series are simply the reproducing conditions.

2.6.3 Weight Functions

Many different weight functions are possible. As illustrated in Figure 1, here the cubic W_3, quintic W_5 and Gaussian W_e kernels are considered [20]:

$$W_3(r,h) = \frac{15}{7\pi h^2}\begin{cases} 2/3 - \left(\frac{r}{h}\right)^2 + \frac{1}{2}\left(\frac{r}{h}\right)^3 & \frac{r}{h} \leq 1 \\ \frac{1}{6}\left(2 - \frac{r}{h}\right)^3 & 1 < \frac{r}{h} \leq 2 \\ 0 & \frac{r}{h} \geq 2 \end{cases} \tag{2.97}$$

$$W_5(r,h) = \frac{7}{478\pi h^2}\begin{cases} \left(3 - \frac{r}{h}\right)^5 - 6\left(2 - \frac{r}{h}\right)^5 + 15\left(1 - \frac{r}{h}\right)^5 & \frac{r}{h} \leq 1 \\ \left(3 - \frac{r}{h}\right)^5 - 6\left(2 - \frac{r}{h}\right)^5 & 1 < \frac{r}{h} \leq 2 \\ \left(3 - \frac{r}{h}\right)^5 & 2 < \frac{r}{h} < 3 \\ 0 & \frac{r}{h} \geq 3 \end{cases} \tag{2.98}$$

$$W_e(r,h) = \frac{k}{\pi h^2} e^{-k\left(\frac{r}{h}\right)^2} \tag{2.99}$$

where $r = ||\vec{x} - \vec{x}_i||$ and h is the smoothing length which defines the compacts support. The coefficients in front are the 2D normalization constants such that the continuous integral of the kernels is unity. Except for the Gaussian kernel, the kernels have a built in cut off radius r_c. For example, the cubic kernel only assigns non-zero values to nodes within $r < 2h$. For the Gaussian kernel, r_c must be specified. For MLS and SPH, derivatives of the weight function are necessary. They can be obtained via repeated application using the chain rule. For $\frac{\partial}{\partial x}$:

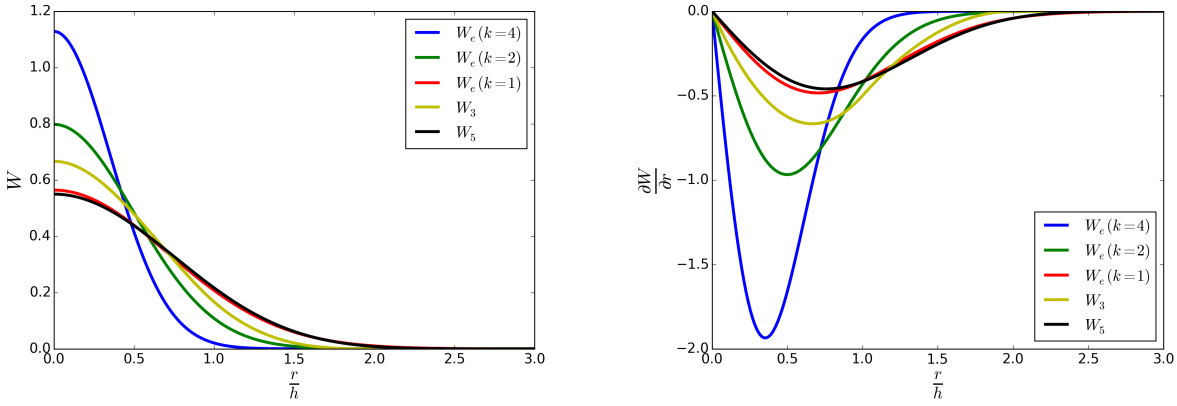
$$\frac{\partial W}{\partial x} = \frac{\partial W}{\partial r}\frac{\partial r}{\partial x} = \frac{(x - x_i)}{r}\frac{\partial W}{\partial r} \tag{2.100}$$

Notice that for SPH and MLS, each weight function is actually associated with the i^{th} neighbor and as such is centered about $\vec{x}_i$. Similarly, so are the derivatives of the weight function. For example, in $1D$ the derivative at x_a using the uncorrected SPH form reads:

$$\left.\frac{du}{dx}\right|_{x_a} = \sum_i u_i \left.\frac{dW_i}{dx}\right|_{x_a} \Delta x_i \tag{2.101}$$

Consider a support domain which only has three nodes u_r, u_c, u_l - a right, center, and left node. If we evaluate the derivative we would see that using Eq. 2.100, $\left.\frac{dW_i}{dx}\right|_{x_a}$ for the right node evaluates to a positive weight, for the left node it evaluates to a negative weight, and

for the center node it evaluates to zero. Hence, on a regular grid we would have a finite difference like expression for $\frac{d}{dx}$.



(a) Cubic, quintic, and Gaussian kernels.

(b) 1^{st} derivative of kernels.

Figure 1: Weight functions and their respective derivatives as a function of r/h.

CHAPTER III

GENERAL FINITE DIFFERENCES (GFD)

3.1 Method

One route to obtaining the GFD approximation is to use a weighted least squares fitting to the Taylor series expansion of a function $u(\vec{x})$ expanded about the star node located at $\vec{x}_a$:

$$u^h(\vec{x}) = \sum_j^m P_j(\vec{x} - \vec{x}_a) a_j(\vec{x}_a) = \mathbf{P}^T(\vec{x} - \vec{x}_a)\mathbf{a}(\vec{x}_a) \tag{3.1}$$

with the polynomial basis:

$$\mathbf{P}^T(\vec{x} - \vec{x}_a) = \begin{bmatrix} 1 & x - x_a & y - y_a & (x - x_a)^2 & (x - x_a)(y - y_a) & (y - y_a)^2 & \dots & P_m \end{bmatrix} \tag{3.2}$$

and Taylor coefficients:

$$\mathbf{a}^T(\vec{x}_a) = \begin{bmatrix} u(\vec{x}_a) & \left.\frac{\partial u}{\partial x}\right|_{\vec{x}_a} & \left.\frac{\partial u}{\partial y}\right|_{\vec{x}_a} & \left.\frac{1}{2}\frac{\partial^2 u}{\partial x^2}\right|_{\vec{x}_a} & \left.\frac{\partial^2 u}{\partial x \partial y}\right|_{\vec{x}_a} & \left.\frac{1}{2}\frac{\partial^2 u}{\partial y^2}\right|_{\vec{x}_a} & \dots & \frac{1}{c!}a_m \end{bmatrix} \tag{3.3}$$

Here the index m refers to the absolute position of the Taylor coefficient starting with the first index at $m = 1$, while c corresponds to the coefficient (i.e., for $\dfrac{\partial^2}{\partial x^2}$, $c = 2$). Using the squared error norm as the metric to describe the error of the constructed approximation:

$$E = \sum_i^n \left(\mathbf{P}^T(\vec{x}_i - \vec{x}_a)\mathbf{a}(\vec{x}_a) - u_i\right)^2 W(\vec{x}_a - \vec{x}_i) \tag{3.4}$$

where n is the number of neighboring points in the support domain for node a determined by the smoothing length h of the symmetric weight function $W(\vec{x}_a - \vec{x}_i) = W(\vec{x}_i - \vec{x}_a)$. Note that $\vec{x}_a$ was substituted for $\vec{x}$, making this approximation GFD and not MLS. Taking the partial derivatives with respect to the Taylor coefficients [1]:

$$\frac{\partial E}{\partial \mathbf{a}} = 2\sum_i^n W(\vec{x}_a - \vec{x}_i)\mathbf{P}(\vec{x}_{ia}) \otimes \mathbf{P}(\vec{x}_{ia})\mathbf{a}(\vec{x}_a) - 2\sum_i^n \mathbf{P}(\vec{x}_{ia})u_i W(\vec{x}_a - \vec{x}_i) \tag{3.5}$$

[1] The resulting equations are called the "normal equations" and are equivalently obtained by making the residual vector orthogonal to the space spanned by the basis. The approaches yield the same equations.

and finding the minimum by setting $\dfrac{\partial E}{\partial \mathbf{a}} = 0$:

$$\sum_i^n W(\vec{x}_a - \vec{x}_i)\mathbf{P}(\vec{x}_{ia}) \otimes \mathbf{P}(\vec{x}_{ia})\mathbf{a}(\vec{x}_a) = \sum_i^n \mathbf{P}(\vec{x}_{ia})u_i W(\vec{x}_a - \vec{x}_i) \tag{3.6}$$

where $\otimes$ represents a tensor product of two vectors. At this point, if the PDE can be recast to a decoupled set of ODEs as would be the case in an explicit version of Chorin's artificial compressibility method for the Navier Stokes equations [8], then the LHS can be inverted and the Taylor coefficients can be used to discretize the derivatives as $\approx c! a_m$. However, if a system of equations needs to be solved, as would be the case using a variant of Chorin's projection method [9], then a little bit more work is necessary to recast the above formulation in terms of n neighboring shape functions evaluated at node a. Start by defining the symmetric moment matrix $\underline{\underline{A}}$ with size $(m \times m)$:

$$\underline{\underline{A}} = \sum_i^n W(\vec{x}_a - \vec{x}_i)\mathbf{P}(\vec{x}_{ia}) \otimes \mathbf{P}(\vec{x}_{ia}) \tag{3.7}$$

Next rewrite the RHS as follows:

$$\sum_i^n \mathbf{P}(\vec{x}_{ia})u_i W(\vec{x}_a - \vec{x}_i) = \underline{\underline{B}}\mathbf{u} \tag{3.8}$$

where $\underline{\underline{B}}$ is a $m \times n$ matrix with each column as the weighted basis vector evaluated at the i^{th} neighbor:

$$\underline{\underline{B}} = \begin{bmatrix} W(\vec{x}_a - \vec{x}_1)P_1(\vec{x}_{1a}) & W(\vec{x}_a - \vec{x}_2)P_1(\vec{x}_{2a}) & \dots & W(\vec{x}_a - \vec{x}_n)P_1(\vec{x}_{na}) \\ W(\vec{x}_a - \vec{x}_1)P_2(\vec{x}_{1a}) & W(\vec{x}_a - \vec{x}_2)P_2(\vec{x}_{2a}) & \dots & W(\vec{x}_a - \vec{x}_n)P_2(\vec{x}_{na}) \\ \dots & \dots & \dots & \dots \\ W(\vec{x}_a - \vec{x}_1)P_m(\vec{x}_{1a}) & W(\vec{x}_a - \vec{x}_2)P_m(\vec{x}_{2a}) & \dots & W(\vec{x}_a - \vec{x}_n)P_m(\vec{x}_{na}) \end{bmatrix} \tag{3.9}$$

and where $\mathbf{u}$ is the vector of the field values $u(\vec{x}_i)$ at the n neighboring points. Solving for $\mathbf{a}(\vec{x}_a)$ and substituting into the approximation $u(\vec{x})$ given by Eq. 3.1:

$$u^h(\vec{x}) = \mathbf{P}^T(\vec{x} - \vec{x}_a)\underline{\underline{A}}^{-1}(\vec{x}_a)\underline{\underline{B}}(\vec{x}_a)\mathbf{u} \tag{3.10}$$

where the shape function vector is:

$$\mathbf{\Phi}(\vec{x}) = \begin{bmatrix} \phi_1(\vec{x}) & \phi_2(\vec{x}) & \phi_3(\vec{x}) & \dots & \phi_n(\vec{x}) \end{bmatrix} = \mathbf{P}^T(\vec{x} - \vec{x}_a)\underline{\underline{A}}^{-1}(\vec{x}_a)\underline{\underline{B}}(\vec{x}_a) \tag{3.11}$$

As was shown before, derivatives are approximated by simply taking the appropriate derivative of $u(\vec{x})$, which in GFD results in a differentiation of $\mathbf{P}^T(\vec{x} - \vec{x}_a)$ while the other terms are held constant. The resulting matrix coefficients for a given row a of the system will correspond to the n neighboring shape functions (and or their derivatives) evaluated at the star node a. When compared to MLS we have the following differences:

1. The GFD approximation is m times differentiable where m is the highest order in the basis, while MLS is k times differentiable where k is how many times the kernel is differentiable.

2. If a polynomial basis is used, the GFD shape functions are polynomials while the MLS shape functions are not.

3. Both GFD and MLS will reproduce any function that can be written as a linear combination of the basis. [2]

4. Both GFD and MLS are in general approximations and not interpolations except for special circumstances.

Continuing, instead of keeping tracking of a specific derivative via appropriate differentiation of $\mathbf{P}^T(\vec{x}-\vec{x}_a)$, let us keep track of all Taylor coefficients given in Eq. 3.6 and then later scale them by the appropriate constant $c!$. The Taylor coefficients can be written as:

$$\mathbf{a}(\vec{x}_a) = \underline{\underline{\phi}}\mathbf{u} \tag{3.12}$$

where $\underline{\underline{\phi}}$ is a $m \times n$ matrix defined as:

$$\underline{\underline{\phi}} = \underline{\underline{A}}^{-1}\underline{\underline{B}} \tag{3.13}$$

Each row of the matrix $\underline{\underline{\phi}}$ can be thought of as the m^{th} Taylor derivative of the n neighboring shape functions evaluated at the star node a scaled by $1/c!$. The first row of shape function values can be used as the coefficients for a smoothing filter for $u(\vec{x}_a)$. Additionally, it may be used to estimate $u(\vec{x})$ where $\vec{x}$ is a location which happens to not coincide with any of the nodes. Alternatively, one can also think of the matrix $\underline{\underline{\phi}}$ as a container of finite difference coefficients where the m^{th} row contains scaled coefficients to discretize the m^{th} Taylor derivative. Equivalently we have:

$$a_m = \sum_i^n \phi_{mi} u_i \tag{3.14}$$

Up to now we have not been clear about one important question and that is whether to include u_a as a known variable when solving the normal equations. If we treat u_a as known, the local approximation will fit through u_a and our matrix sizes will reduce by one row and one column. Taking the first Taylor coefficient as known and repeating the above procedure we have:

[2] An intuitive analogy is that just like any vector in a 2D plane can be represented by two orthonormal vectors, so can any function which happens to live in the space spanned by the basis.

$$\mathbf{a} = \underline{\underline{A}}^{-1}\underline{\underline{B}}(\mathbf{u} - \mathbf{u}_a) \tag{3.15}$$

equivalently written as:

$$a_m = \sum_i^n \phi_{mi} u_i - u_a \sum_i^n \phi_{mi} \tag{3.16}$$

Here ϕ_{mi} are the entries of the shape function matrix $\underline{\underline{\phi}}$ which is now a $(m-1) \times (n-1)$ matrix. The Taylor coefficients $\mathbf{a}$ will also be reduced to a size of $m-1$ since the first entry u_a is known. For the examples presented in this work, we use the quadratic basis without the constant basis:

$$\mathbf{P}^T(\vec{x} - \vec{x}_a) = \begin{bmatrix} x - x_a & y - y_a & (x - x_a)^2 & (x - x_a)(y - y_a) & (y - y_a)^2 \end{bmatrix} \tag{3.17}$$

$$\mathbf{a}^T(\vec{x}_a) = \begin{bmatrix} \left.\dfrac{\partial u}{\partial x}\right|_{\vec{x}_a} & \left.\dfrac{\partial u}{\partial y}\right|_{\vec{x}_a} & \left.\dfrac{1}{2}\dfrac{\partial^2 u}{\partial x^2}\right|_{\vec{x}_a} & \left.\dfrac{\partial^2 u}{\partial x \partial y}\right|_{\vec{x}_a} & \left.\dfrac{1}{2}\dfrac{\partial^2 u}{\partial y^2}\right|_{\vec{x}_a} & \cdots & \dfrac{1}{c!}a_m \end{bmatrix} \tag{3.18}$$

where the final approximation reads:

$$u^h(\vec{x}) = u_a + \mathbf{P}^T(\vec{x} - \vec{x}_a)\underline{\underline{A}}^{-1}(\vec{x}_a)\underline{\underline{B}}(\vec{x}_a)(\mathbf{u} - \mathbf{u}_a) \tag{3.19}$$

As before, derivatives can be evaluated by differentiating the approximation and then collocating:

$$\left.\frac{\partial u^h}{\partial x}\right|_{\vec{x}_a} = \left.\frac{\partial \mathbf{\Phi}}{\partial x}\right|_{\vec{x}_a} \mathbf{u} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \underline{\underline{A}}^{-1}\underline{\underline{B}}(\mathbf{u} - \mathbf{u}_a) = \sum_i^n \phi_{2i} u_i - u_a \sum_i^n \phi_{2i} \tag{3.20}$$

or for $\dfrac{\partial^2}{\partial x^2}$:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{\vec{x}_a} = \left.\frac{\partial^2 \mathbf{\Phi}}{\partial x^2}\right|_{\vec{x}_a} \mathbf{u} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 \end{bmatrix} \underline{\underline{A}}^{-1}\underline{\underline{B}}(\mathbf{u} - \mathbf{u}_a) = 2\left(\sum_i^n \phi_{4i} u_i - u_a \sum_i^n \phi_{4i}\right) \tag{3.21}$$

Note that here ϕ_{2i} uses the convention we mentioned previously where we number the rows of the container matrix using absolute indexes (i.e., if a row is removed in the event a basis is removed, the row numbering does not change).

3.2 Boundary Conditions

One way to implement boundary conditions is the traditional collocation approach described in the introduction where nodes are positioned directly on the boundary allowing for an approximation to be constructed at the boundary [3, 25]. In the resulting $NP \times NP$ system, some of the rows will correspond to Γ while the majority will correspond to $\Omega - \Gamma$. The

boundary operators directly act on the approximation at the boundary and will determine the coefficients and RHS for a particular row corresponding to a node belonging to Γ, while the differential operator corresponding to the PDE will act on the nodes corresponding to $\Omega - \Gamma$ and will determine the corresponding row and RHS. While this approach is conceptually simple, one of the limitations in positioning nodes on the boundary is what Liszka and Orkisz referred to as the problem of an "unbalanced star" wherever Neumann conditions are applied. At the boundary, the support domain will be biased in the normal direction towards the interior and as a result so will the derivative estimate.

To balance the star, Liszka and Orkisz proposed adding "auxillary" nodes outside the computational domain near Neumann boundaries to which they extrapolated a value such that the star is balanced. Following this reasoning, here we generalize the sharp interface immersed boundary approach (as classified by Mittal and Iaccarino) to meshfree grids. When the immersed boundary is treated as a sharp interface, the boundary conditions are imposed by modifying the computational stencil for nodes near the boundary [23]. For example, consider the central difference stencil truncated near the boundary element $\partial\Omega_e$ as shown below.

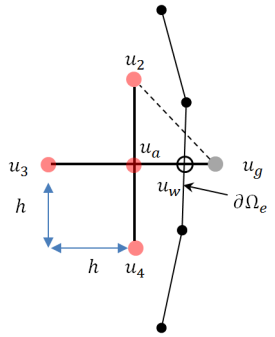


Figure 2: Truncated central difference stencil near the linear boundary element $\partial\Omega_e$.

Writing a central difference for the star node a we have:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{\vec{x}_a} = \frac{u_3 - 2u_a + u_g}{h^2} \tag{3.22}$$

Suppose we have a Dirichlet boundary condition u_w imposed at the intersection of the line segment ag with $\partial\Omega_e$. We wish to find a value for u_g such that the boundary condition at $\vec{x}_w$ is approximately enforced. One possible approach is to linearly interpolate u_a, u_w, and u_2 by the matrix $\underline{\underline{Q}}$ given by Equation 3.28 and then extrapolate the value to $\vec{x}_g$ using $\underline{\underline{Q}}^{-1}$ where $u_g = Q_{11}^{-1}u_a + Q_{12}^{-1}u_w + Q_{13}^{-1}u_2$. We see that u_g is linearly dependent on u_a, u_2, and

u_w. Upon substitution, we have:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{\vec{x}_a} = \frac{u_3 - 2u_a + Q_{11}^{-1}u_a + Q_{12}^{-1}u_w + Q_{13}^{-1}u_2}{h^2} \tag{3.23}$$

after regrouping terms:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{\vec{x}_a} = \frac{u_3 + (Q_{11}^{-1} - 2)u_a + Q_{13}^{-1}u_2}{h^2} + \frac{Q_{12}^{-1}u_w}{h^2} \tag{3.24}$$

We see two coefficients in row a would be modified to incorporate the Dirichlet boundary conditions, while an additional term related to u_w would need to be subtracted from the RHS for row a. Note that the coefficients that need to be modified will correspond to the nodes used in the interpolation matrix.

The above example is potentially misleading since $u_2 - u_a$ is orthogonal to $u_a - u_g$ and will not contribute to the spatial derivatives in the x direction (i.e., $Q_{13}^{-1} = 0$). Only the coefficient for u_a would actually be modified, while the known boundary term would be subtracted from the RHS. Another approach would be to construct a quadratic polynomial centered about $\vec{x}_a$ that interpolates u_3, u_a, u_w and then to differentiate, however, this could not be used to implement a Neumann boundary condition at $\partial\Omega_e$, since u_3 and u_w are co-linear with u_a and u_g, but could potentially be non co-linear with the boundary normal of $\partial\Omega_e$.

We circumvent these problems by using enough points in the support domain of node a (near the boundary) such that appropriate interpolation points can be found (see Figure 4). We note that our approach differs from Tseng and Ferziger's approach [33] in the following ways:

1. they designed for and limited their approach to Cartesian grids.
2. they do not modify the computational stencil near the boundary but instead extrapolate a value to ghost nodes to be used as a forcing function.
3. they consider higher order interpolations.
4. we explicitly place a restriction that the interpolation points must come from the computational stencil of node a.
5. our interpolation matrices are linear approximations and are centered about the ghost nodes.

To generalize enforcing boundary conditions via the sharp interface approach for an irregular grid, some extra accounting is necessary. We start by expressing the Taylor coefficient given by Eq. 3.16 over different sets of local indexes:

$$a_m = \sum_{i \in f} \phi_{mi} u_i + \sum_{i \in g_d} \phi_{mi} u_i + \sum_{i \in g_n} \phi_{mi} u_i - u_a \sum_{i \in n} \phi_{mi} \tag{3.25}$$

where the sets of local indexes for star node a are:

1. n - set of local indexes of fluid and ghost points in the support domain of the star node
2. f - set of local indexes of the nodes which are strictly in the fluid.
3. g_d - set of local indexes of the ghost nodes for which the relative vector $\vec{x}_a - \vec{x}_g$ intersects a boundary element $\partial\Omega_e$ which has Dirichlet boundary conditions imposed.
4. g_n - set of local indexes of ghost nodes for which the relative vector $\vec{x}_a - \vec{x}_g$ intersects a boundary element $\partial\Omega_e$ which has Neumann boundary conditions imposed.

In the following sections, we will show how each u_i, in the ghost sets g_d and g_n, can be represented as a linear combination of the star node a, another fluid node o, and a boundary point w. Depending on the boundary conditions needed, an appropriate value is extrapolated to u_g by using either linear interpolation matrix $\underline{Q}$ or $\underline{\underline{R}}$, for Dirichlet and Neumann boundary conditions respectively. By choosing the linear interpolation matrices to be centered about the ghost point, the linear relation for a ghost node is simplified to only three coefficients (corresponding to the first row of the matrix inverse) making it easier to regroup terms, modify coefficients, and determine the additional terms that need to be subtracted from the RHS.

3.2.1 Dirichlet

The field function value at a ghost node located at $\vec{x}_g$ which is a neighbor of the star node a can be extrapolated using a Taylor polynomial expanded about $\vec{x}_g$:

$$u(\vec{x}) = a_o + a_1(x - x_g) + a_2(y - y_g) + O(h^2) \tag{3.26}$$

Here we will consider a linear basis and interpolate three nodes - two in the fluid and one on the boundary domain - to construct the following (3×3) system:

$$\begin{bmatrix} 1 & x_a - x_g & y_a - y_g \\ 1 & x_w - x_g & y_w - y_g \\ 1 & x_o - x_g & y_o - y_g \end{bmatrix} \begin{bmatrix} a_o \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} u(\vec{x_a}) \\ u(\vec{x_w}) \\ u(\vec{x_o}) \end{bmatrix} \tag{3.27}$$

As shown in Figure 4, u_a is the field function value at the star node, u_w is the known boundary value at the intersection point of line ag with the piecewise linear element $\partial\Omega_e$, and u_o is field value at a third interpolation point in the support domain of $\vec{x}_a$. We choose $\vec{x}_o$ which has the minimum distance to $\vec{x}_w$ such that line ag and ao are not co-linear[3]. Define the linear interpolation matrix for Dirichlet boundary conditions as $\underline{\underline{Q}}$:

$$\underline{\underline{Q}} = \begin{bmatrix} 1 & x_a - x_g & y_a - y_g \\ 1 & x_w - x_g & y_w - y_g \\ 1 & x_o - x_g & y_o - y_g \end{bmatrix} \tag{3.28}$$

By choosing the approximation to occur around $\vec{x}_g$, the extrapolated value at the ghost node will simply be the dot product of the first row of $\underline{\underline{Q}}^{-1}$ and the RHS:

$$u(\vec{x}_g) = a_o = Q_{11}^{-1} u_a + Q_{12}^{-1} u_w + Q_{13}^{-1} u_o \tag{3.29}$$

As shown in Figure 4, each ghost node will potentially have a unique matrix $\underline{\underline{Q}}$, unique u_w, and unique u_o. As such, we must add a subscript i to each term upon substitution into the Dirichlet summation term of Eq. 3.25:

$$\sum_{i \in g_d} \phi_{mi} u_i = u_a \sum_{i \in g_d} \phi_{mi} Q_{11,i}^{-1} + \sum_{i \in g_d} \phi_{mi} Q_{12,i}^{-1} u_{w,i} + \sum_{i \in g_d} \phi_{mi} Q_{13,i}^{-1} u_{o,i} \tag{3.30}$$

Since each ghost node (see Figures 4a and 4b) may potentially share the same u_o in their extrapolation matrices, a particular u_o may be modified more than once. As such, it will be necessary to reduce each $u_{o,i}$ into a unique set of $u_{o,j}$ in order to correctly group the modifying coefficients:

$$\sum_{i \in g_d} \phi_{mi} Q_{13,i}^{-1} u_{o,i} = \sum_j u_{o,j} \left(\sum_{i \in q} \phi_{mi} Q_{13,i}^{-1} \right) \tag{3.31}$$

here j corresponds to the unique set of $u_{o,i}$, while set q corresponds to the set of ghosts nodes with Dirichlet boundary conditions which share the same u_o and u_a. Substituting, the summation term over set g_d reads:

$$\sum_{i \in g_d} \phi_{mi} u_i = u_a \sum_{i \in g_d} \phi_{mi} Q_{11,i}^{-1} + \sum_{i \in g_d} \phi_{mi} Q_{12,i}^{-1} u_{w,i} + \sum_j u_{o,j} \left(\sum_{i \in q} \phi_{mi} Q_{13,i}^{-1} \right) \tag{3.32}$$

Here we see that the set g_d contributes a modification to the coefficient of u_a and to the coefficients of each $u_{o,j}$ used. In addition, a known term related to each $u_{w,i}$ is produced which will be subtracted from the RHS.

[3]Note that we always position grid points approximately half a unit away from the boundary.

3.2.2 Neumann

Following the procedure that was outlined in the preceding section for Dirichlet boundary conditions, we now consider the modification of the computational stencil that occurs when Neumann conditions are imposed. For Neumann boundary conditions we must satisfy:

$$\frac{\partial u}{\partial n} = \nabla u \cdot \hat{n} = \frac{\partial u}{\partial x} n_x + \frac{\partial u}{\partial y} n_y \tag{3.33}$$

where the unit normal of the piecewise linear element is defined as:

$$\hat{n} = -sin\theta \hat{i} + cos\theta \hat{j} \tag{3.34}$$

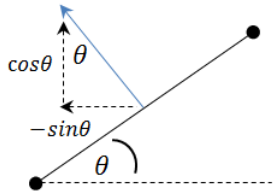


Figure 3: Unit normal definition.

Replacing row 2 in the preceding system with the Neumann boundary condition:

$$\begin{bmatrix} 1 & x_a - x_g & y_a - y_g \\ 0 & -sin\theta & cos\theta \\ 1 & x_o - x_g & y_o - y_g \end{bmatrix} \begin{bmatrix} b_o \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} u(\vec{x_a}) \\ \left.\frac{\partial u}{\partial n}\right|_{\vec{x}_w} \\ u(\vec{x_o}) \end{bmatrix} \tag{3.35}$$

where we define the Neumann interpolation matrix $\underline{\underline{R}}$ which will be used to extrapolate a value to the ghost node for a particular star - ghost pair:

$$\underline{\underline{R}} = \begin{bmatrix} 1 & x_a - x_g & y_a - y_g \\ 0 & -sin\theta & cos\theta \\ 1 & x_o - x_g & y_o - y_g \end{bmatrix} \tag{3.36}$$

Upon substituting each ghost value u_g into the Neumann ghost set we arrive at:

$$\sum_{i \in g_n} \phi_{mi} u_i = u_a \sum_{i \in g_n} \phi_{mi} R^{-1}_{11,i} + \sum_{i \in g_n} \phi_{mi} R^{-1}_{12,i} \frac{\partial u_{w,i}}{\partial n} + \sum_{i \in g_n} \phi_{mi} R^{-1}_{13,i} u_{o,i} \tag{3.37}$$

Similarly:

$$\sum_{i \in g_n} \phi_{mi} R^{-1}_{13,i} u_{o,i} = \sum_j u_{o,j} \left(\sum_{i \in r} \phi_{mi} R^{-1}_{13,i} \right) \tag{3.38}$$

where j is again the unique set of "other" fluid nodes used during extrapolation while r is the set of ghost nodes with Neumann boundary conditions which share the same u_o and u_a. After substituting the Neumann summation term over ghosts reads:

$$\sum_{i \in g_n} \phi_{mi} u_i = u_a \sum_{i \in g_n} \phi_{mi} R^{-1}_{11,i} + \sum_{i \in g_n} \phi_{mi} R^{-1}_{12,i} \frac{\partial u_{w,i}}{\partial n} + \sum_j u_{o,j} \left(\sum_{i \in r} \phi_{mi} R^{-1}_{13,i} \right) \tag{3.39}$$

We see that the Neumann set will produce modifications to u_a, each $u_{o,j}$, and will produce a known term that will be subtracted from the RHS.

3.2.3 Putting it Together...

After substituting Equations 3.32 and 3.39 into Equation 3.25 and grouping terms in front of unique nodal values, the general form for the Taylor coefficients evaluated at a given star node a with a set of n neighboring support points using linear extrapolation is:

$$a_m = \sum_{i \in (f \notin j)} \phi_{mi} u_i + u_a \left(\sum_{i \in g_d} \phi_{mi} Q^{-1}_{11,i} + \sum_{i \in g_n} \phi_{mi} R^{-1}_{11,i} - \sum_{i \in n} \phi_{mi} \right) + \sum_j k_j u_{o,j} + B_{RHS} \tag{3.40}$$

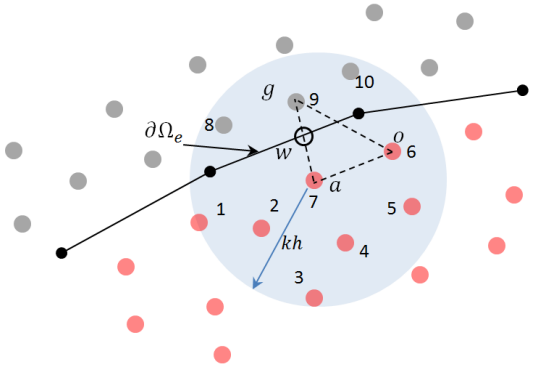
where the coefficient in front of the "other" nodes used in extrapolation is given as:

$$k_j = \phi_{mj} + \sum_{i \in q} \phi_{mi} Q^{-1}_{13,i} + \sum_{i \in r} \phi_{mi} R^{-1}_{13,i} \tag{3.41}$$

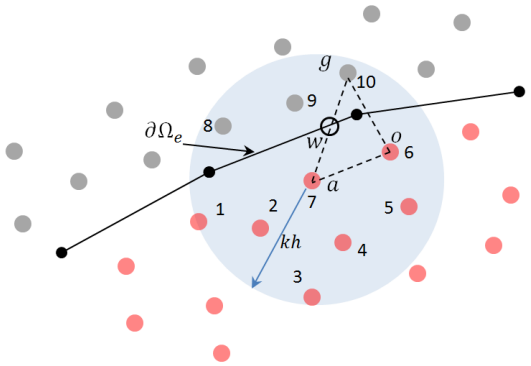
B_{RHS} are the terms falling out of the extrapolation matrix involving the boundary and are considered knowns.

$$B_{RHS} = \sum_{i \in g_d} \phi_{mi} Q^{-1}_{12,i} u_{w,i} + \sum_{i \in g_n} \phi_{mi} R^{-1}_{12,i} \frac{\partial u_{w,i}}{\partial n} \tag{3.42}$$

Note that Equation 3.40 is the general form of the Taylor coefficients in terms of i scaled shape functions evaluated at node a. The additional terms will only appear if the star node's support domain overlaps a boundary element, in which case modifications to coefficients are performed using appropriate linear extrapolation to fulfill boundary conditions at intersection points between the star node and each of its ghost nodes. Multiplying the Taylor coefficient by $c!$, gives us the appropriate derivatives to approximate a differential operator and will ultimately define the coefficients of row a. Note that this form can be used with GFD, SPH, or MLS, as long as the approximations are collocated with u_a treated as a known variable and where i scaled shape functions maybe evaluated at $\vec{x}_a$.



(a) Matrix $\underline{\underline{Q_9}}$ or $\underline{\underline{R_9}}$ used to extrapolate approach values to the ghost node at $\vec{x}_9$ to fulfill the boundary condition at $\vec{x}_w$



(b) Matrix $\underline{\underline{Q_{10}}}$ or $\underline{\underline{R_{10}}}$ used to extrapolate approach values to the ghost node at $\vec{x}_{10}$ to fulfill the boundary condition at $\vec{x}_w$.

Figure 4: Star node a with example support domain which intersects a boundary element $\partial\Omega_e$. For linear interpolation two additional points located at $\vec{x}_w$ and $\vec{x}_o$ are used to linearly extrapolate values to the ghost nodes falling in the support domain. The boundary conditions on the element $\partial\Omega_e$ will determine whether Dirichlet or Neumann conditions will be imposed at $\vec{x}_w$. Note that point $\vec{x}_6$ is again used in Figure 4b, however, this may not always be the case.

3.3 Recovering Finite Differences

Consider the star node u_a shown in Figure 5 with the 4 neighbor support domain. Dropping the cross derivative term we use the following basis with a uniform weight function:

$$\mathbf{P}^T(\vec{x} - \vec{x}_a) = \begin{bmatrix} x - x_a & y - y_a & (x - x_a)^2 & (y - y_a)^2 \end{bmatrix} \tag{3.43}$$

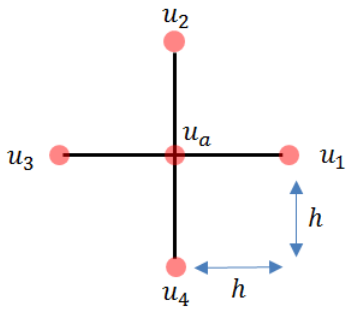


Figure 5: Central difference stencil.

Calculating $\underline{\underline{A}}^{-1}$, $\underline{\underline{B}}$, and $\underline{\underline{\phi}}$:

$$\underline{\underline{A}}^{-1} = \begin{bmatrix} \frac{1}{2h^2} & 0 & 0 & 0 \\ 0 & \frac{1}{2h^2} & 0 & 0 \\ 0 & 0 & \frac{1}{2h^4} & 0 \\ 0 & 0 & 0 & \frac{1}{2h^4} \end{bmatrix} \quad \underline{\underline{B}} = \begin{bmatrix} h & 0 & -h & 0 \\ 0 & h & 0 & -h \\ h^2 & 0 & h^2 & 0 \\ 0 & h^2 & 0 & h^2 \end{bmatrix} \quad \underline{\underline{\phi}} = \begin{bmatrix} \frac{1}{2h} & 0 & -\frac{1}{2h} & 0 \\ 0 & \frac{1}{2h} & 0 & -\frac{1}{2h} \\ \frac{1}{2h^2} & 0 & \frac{1}{2h^2} & 0 \\ 0 & \frac{1}{2h^2} & 0 & \frac{1}{2h^2} \end{bmatrix} \tag{3.44}$$

Resulting in the classic central difference estimates when using Eq. 3.16:

$$\left.\frac{\partial u}{\partial x}\right|_{\vec{x}_a} \approx a_2 = \frac{u_1 - u_3}{2h} \tag{3.45}$$

$$\left.\frac{\partial u}{\partial y}\right|_{\vec{x}_a} \approx a_3 = \frac{u_2 - u_4}{2h} \tag{3.46}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{\vec{x}_a} \approx 2a_4 = \frac{u_1 - 2u_a + u_3}{h^2} \tag{3.47}$$

$$\left.\frac{\partial^2 u}{\partial y^2}\right|_{\vec{x}_a} \approx 2a_6 = \frac{u_2 - 2u_a + u_4}{h^2} \tag{3.48}$$

By using the above fact, we were able to facilitate the debugging phase of code development by limiting the support domain to four neighbors and dropping the mixed basis to see if the corresponding central difference estimates would be recovered when using a lattice arrangement.

CHAPTER IV

GRID GENERATION

4.1 Uniform Poisson Disc Sampling

Spatially distributing nodes for an unstructured grid is a non trivial matter. Choosing a random distribution of points will lead to highly ill-conditioned moment matrices $\underline{\underline{A}}$ due to the linear dependence that results when nodes inevitably cluster. To alleviate the clustering seen in random distributions, Poisson disk sampling is used. We follow the algorithm as described by Schechter and Bridson [29]. In uniform Poisson disk sampling, the goal is to create a random distribution of nodes with a guaranteed minimum spacing s. Without going into implementation details, the algorithm is as follows:

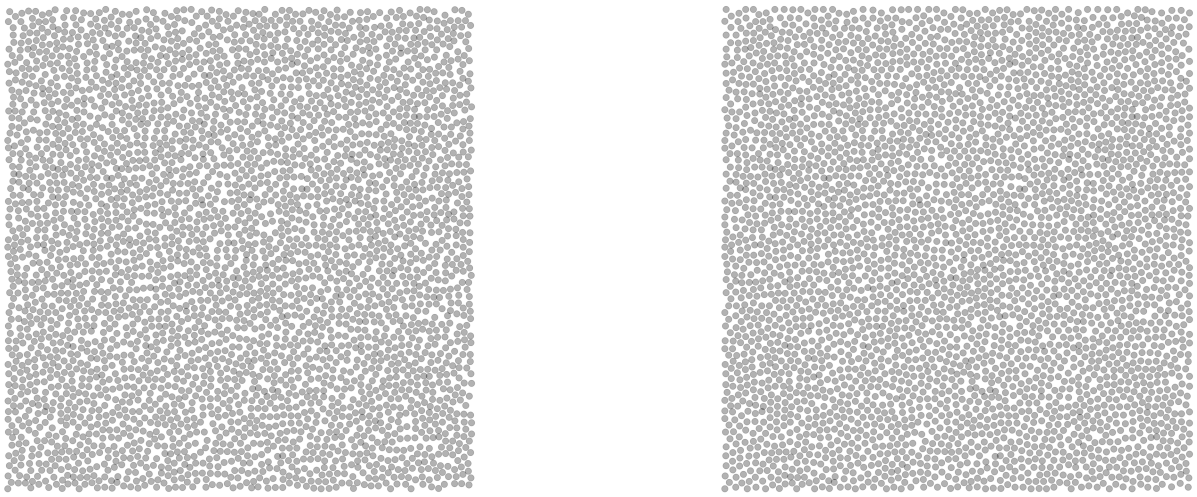
1. Choose initial point $\vec{x}_o$.
2. Randomly generate k candidates within an annular disk ranging from s to $s + \Delta s$.
3. Add candidate to the "active" list if it is 1) within the domain and 2) $> s$ away from all existing nodes. If no "active" candidates are generated, mark node a as "inactive".
4. Choose a random "active" node a from the list and go back to step 2. When there are no more "active" nodes, the algorithm ends.

After the generation step, an optional relaxation step may be applied to each node a:

1. Calculate r_{min}, the minimum distance to neighboring nodes for node a.
2. Generate k candidates whose radial distance τ away from from the selected node is given by the following decreasing function $\tau = \left(\frac{k-i}{k}\right) s$, where $i \in [0, k-1]$ and is the candidate id
3. If candidate i is in the domain, calculate r^*_{min} the minimum distance to neighboring nodes.
4. If $r^*_{min} > s$ and $r^*_{min} > r_{min}$, set $r_{min} = r^*_{min}$ and update the tracking index for the candidate id.

5. Test the next candidate starting with step 3. If no candidate remains to be tested, update the position of node a to the candidate position corresponding to the tracking index.

Figure 6 shows a Poisson distribution and the effect of relaxation on an initial distribution is to increase the characteristic spacing s between nodes (i.e., filling in empty regions). Ideally, this relaxation should decrease the condition number of $\underline{\underline{A}}$ at each of our star nodes, however this is hard to guarantee.



(a) Before relaxation.

(b) After relaxation.

Figure 6: Poisson disk sampling on a square using $k = 30$ with relaxation applied 10 times.

4.2 Variable Poisson Disk Sampling

In variable Poisson disk sampling, the spacing can be a function of space and time $s(\vec{x}, t)$ as shown in Figure 7. In step 2, now the active node a generates k candidates in an annulus $s(\vec{x}_a, t) + \Delta s$. In addition, step 3 is modified to consider a non constant "bubble" around each neighbor. Letting the distance between the candidate c and an arbitrary neighbor point p be R_{cp}, then two conditions are required: $R_{cp} > s(\vec{x_c})$ and $R_{cp} > s(\vec{x_p})$. In other words, no two points can reside within each others bubbles. Note that if there are sharp spatial gradients in the spacing function then the algorithm will potentially fail to fill empty coarse regions when the active node a resides in the fine region and tries to produce a candidate in the coarse region if $s(\vec{x}_a) + \Delta s < s(\vec{x}_c)$. If $s(\vec{x}_a) + \Delta s > s(\vec{x}_c)$ is not satisfied, the active node a will always produce rejected candidates since the node a will always reside in the bubble of each candidate.

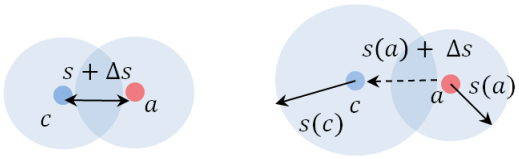
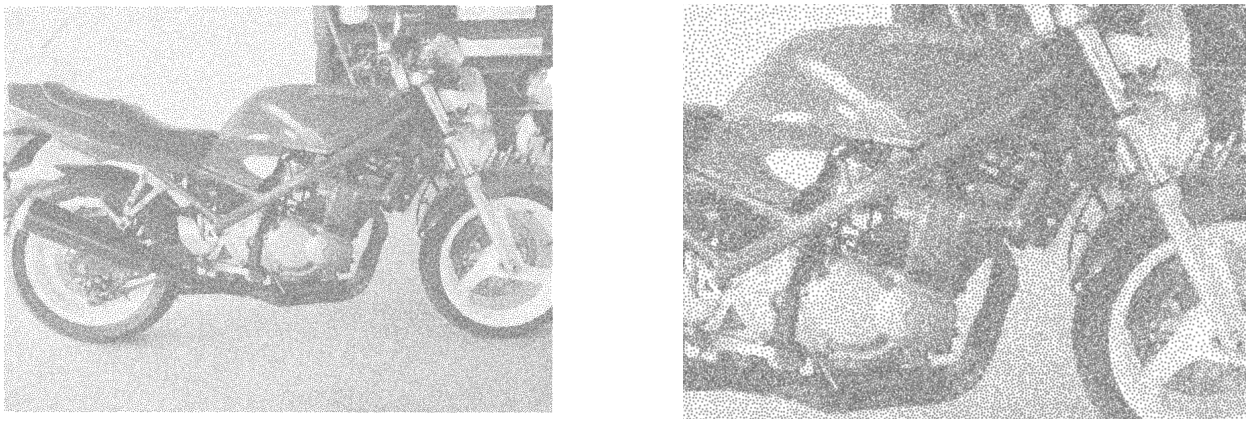


Figure 7: Uniform (left) versus variable (right) Poisson disk sampling.

The algorithm was tested on a series of images converted to greyscale and normalized resulting in a spatial function ranging from 0 (black) to 1 (white). The greyscale function was used to describe the varying radii of each bubble as follows:

$$s(\vec{x}) = s_{min} + (s_{max} - s_{min})G(\vec{x}) \tag{4.1}$$

where s_{min} is the minimum spacing of the grid, s_{max} is the maximum spacing, and $G(\vec{x})$ is the normalized greyscale value. The image on the left in Figure 8 is actually a variable Poisson disk distribution where the nodes are represented by gray markers as shown in the zoomed in image. The nodes do not actually overlap but rather their markers do. The dark areas correspond to fine areas of a grid where the spacing between nodes would be small. The visual effect is very similar to the pointillism painting technique, though we do not color the dots here.



(a) Zoomed out.

(b) Zoomed in.

Figure 8: Variable poisson disk sampling using an image's greyscale.

4.3 Neighbor Search

An important detail left out so far is an efficient algorithm for a neighbor search. The typical approach taken in particle-based methods is to divide the domain into cells of equal

sizes so that only the surrounding cells need to be checked for neighbors. The particle positions are mapped to a cell index which is linked to a list of all particles ids residing in the cell. The linking can be achieved by maintaining two arrays: a head of chain array and a linked list array. The first array is indexed using the cell ids and returns the corresponding id of the "head of chain" in the linked list. The value at the "head of chain" is the index used to obtain the next node residing in the current cell from the linked list. If the end of the chain is reached, -1 is returned. This is shown graphically in Figure 9. Although it is incredibly inefficient, here we use the same algorithm for uniform and variable grids by setting cell sizes based on the coarsest resolution.

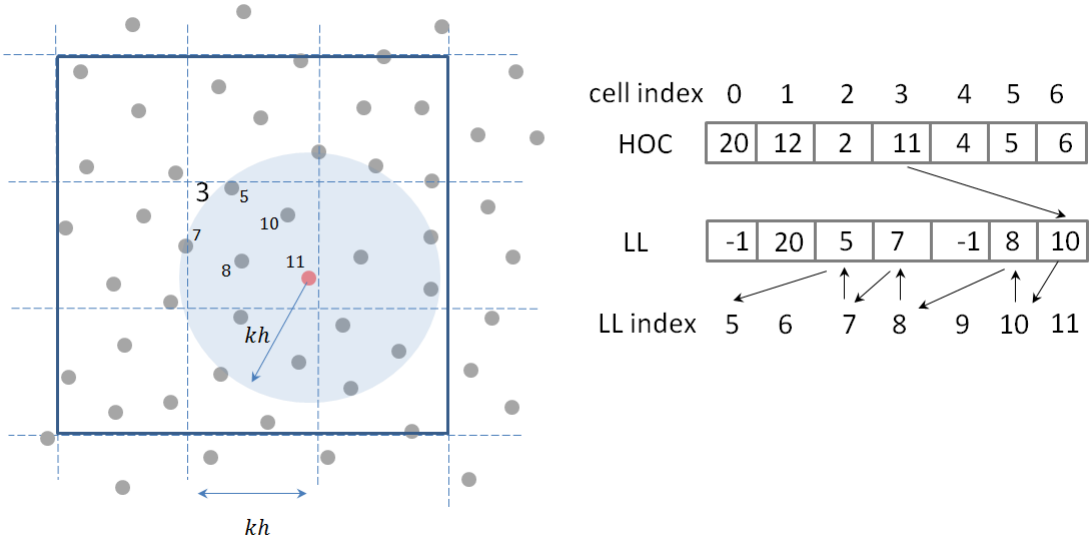


Figure 9: Head of chain and linked list arrays for an example grid.

CHAPTER V

ALGORITHM

5.1 Fractional Step Method for Navier-Stokes Equations

Written in terms of the primitive variables in non-conservative form, the momentum and mass conservation equations for a viscous, incompressible fluid from the Eulerian viewpoint reads:

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\, \vec{u} = -\nabla P + \nu \nabla^2 \vec{u} \tag{5.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{5.2}$$

where $\vec{u}$ is the velocity field, P is the density scaled pressure, and ν is the kinematic viscosity. A relatively simple scheme to solve the pressure-velocity coupled equations is an explicit fractional step projection method [9, 27]. In this method, the momentum equation is split into a viscous and inviscid equation:

$$\frac{\vec{u}^* - \vec{u}^n}{\Delta t} = \left(- (\vec{u}^n \cdot \nabla)\, \vec{u}^n + \nu \nabla^2 \vec{u}^n \right) \tag{5.3}$$

$$\frac{\vec{u}^{n+1} - \vec{u}^*}{\Delta t} = -\nabla P^{n+1} \tag{5.4}$$

Adding these two equations recovers the original momentum equation. We see the first equation predicts an intermediate velocity $\vec{u}^*$ only considering viscous effects. The predicted intermediate velocity satisfies both the tangential and normal boundary conditions, that is $\vec{u}^*|_\Gamma = \mathbf{b}^{n+1}$ [27]. However, $\vec{u}^*$ does not satisfy the divergence free condition since the pressure term was dropped (i.e., $\nabla \cdot \vec{u}^* \neq 0$).

In the second equation, the intermediate velocity is split into a divergence free field u^{n+1} and a curl free field ∇P^{n+1}. By taking the divergence of Eq. 5.4 and using the incompressibility condition $\nabla \cdot \vec{u}^{n+1} = 0$, we can obtain the following Pressure Poisson Equation (PPE):

$$\nabla \cdot \nabla P^{n+1} = \frac{1}{\Delta t} \nabla \cdot \vec{u}^* \tag{5.5}$$

Here we see that the RHS is the divergence of the intermediate velocity field. In order to solve the PPE, boundary conditions are needed on pressure. According to Quartapelle, the inviscid equation can only satisfy the normal boundary condition on $\vec{u}^{n+1}$ (i.e.,$\vec{u}^{n+1} \cdot \hat{n} = \mathbf{b}^{n+1} \cdot \hat{n}$) and so we project Eq. 5.4 at the boundary Γ onto the normal obtaining the following condition

on pressure:

$$\left.\frac{\partial P^{n+1}}{\partial n}\right|_{\Gamma} = 0 \tag{5.6}$$

After solving for P^{n+1}, ∇P^{n+1} is used to correct the intermediate field to obtain the divergence free field $\vec{u}^{n+1}$:

$$\vec{u}^{n+1} = \vec{u}^{*} - \Delta t \nabla P^{n+1} \tag{5.7}$$

As Quartapelle describes, the corrected field $\vec{u}^{n+1}$ will satisfy the normal boundary condition, but will only approximately satisfy the tangential condition [27]. The argument here is that the tangential velocity condition is imposed on the intermediate field and so the corrected field will approximately satisfy the tangential conditions as well. To march the momentum equation forward in time, the time step Δt is chosen as the minimum between the following convective and diffusive guidelines:

$$\begin{aligned} \Delta t_u &= 0.5 \frac{\Delta s}{u_{max}} \\ \Delta t_\nu &= 0.125 \frac{\Delta s^2}{\nu} \end{aligned} \tag{5.8}$$

where Δs is the minimum spacing between grid points, and u_{max} is the maximum velocity magnitude.

5.2 Algorithm

Using the sharp interface framework described previously, the algorithm can be summarized as follows:

1. Initialize the piecewise linear elements delineating the boundaries and describing the boundary conditions. Initialize the grid using the variable Poisson disk algorithm.

2. For each star node a , find the nearest neighbors within kh, where k depends on the weight function used and h is a variable smoothing length such that the specified number of neighbors are found.

3. For each star node a in $\Omega - \Gamma$, calculate $\underline{\underline{\phi}}$ - the $m \times n$ scaled shape function matrix evaluated using Equation 3.40.

4. Populate the $N \times N$ Laplacian system, where each row corresponds to a star node a.

5. Calculate $\vec{u}^{*}$ using Equation 5.3 and then calculate $\nabla \cdot \vec{u}^{*}$.

6. Solve the PPE for P^{n+1} using an appropriate linear solver. [1]

7. Calculate ∇P^{n+1} and correct $\vec{u}^*$ using Equation 5.7 to obtained u^{n+1}. Go back to Step 5 and repeat till terminating condition.

In Cartesian coordinates (x, y) the operators are approximated using Eq. 3.40 as follows:

1. ∇ (gradient): $\dfrac{\partial}{\partial x}\hat{i} + \dfrac{\partial}{\partial y}\hat{j} = a_2\hat{i} + a_3\hat{j}$

2. $\nabla\cdot$ (divergence): $\dfrac{\partial}{\partial x}u + \dfrac{\partial}{\partial y}v = a_2(u) + a_3(v)$

3. $\nabla^2 = \nabla\cdot\nabla()$ (laplacian): $\dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} = 2(a_4 + a_6)$

In axisymmetric cylindrical coordinates (r, z) some of the operators take on additional terms:

1. ∇ (gradient): $\dfrac{\partial}{\partial z}\hat{z} + \dfrac{\partial}{\partial r}\hat{r} = a_2\hat{z} + a_3\hat{r}$

2. $\nabla\cdot$ (divergence): $\dfrac{u_r}{r} + \dfrac{\partial u_z}{\partial z} + \dfrac{\partial u_r}{\partial r} = \dfrac{u_r}{r} + a_2(u_z) + a_3(u_r)$

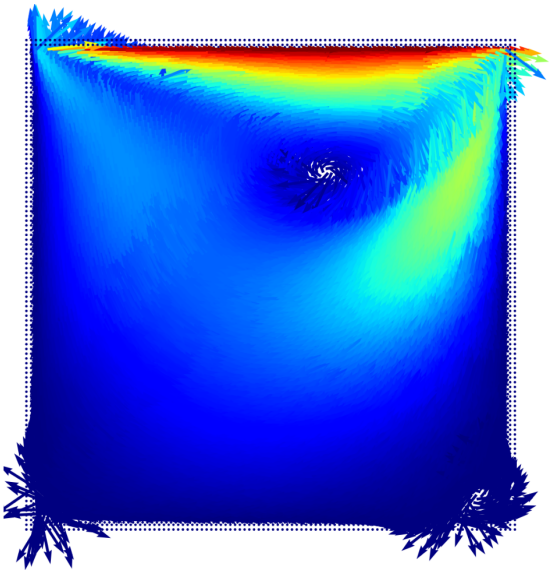
3. $\nabla^2 = \nabla\cdot\nabla()$ (laplacian): $\dfrac{1}{r}\dfrac{\partial}{\partial r} + \dfrac{\partial^2}{\partial z^2} + \dfrac{\partial^2}{\partial r^2} = \dfrac{1}{r}a_3 + 2(a_4 + a_6)$

where r is radial coordinate and z is the axial coordinate. The above algorithm can easily be modified into a Lagrangian solver where now Steps 2, 3, and 4 would need to be repeated at approximately every time step. However, it is not clear at this moment how well variable resolution will be handled since in the Lagrangian solver care must be taken to assign masses via some notion of the volume a particle occupies.

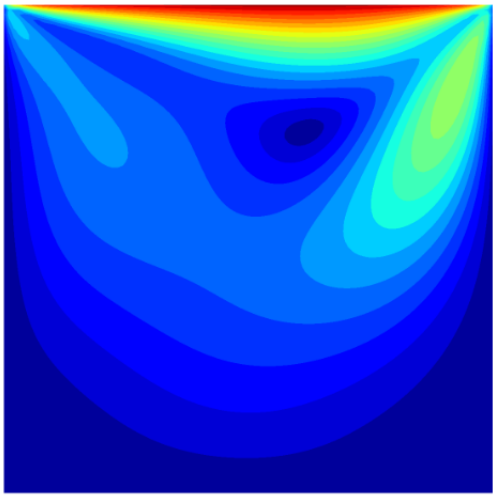
5.3 Post Processing

It is desirable to have the numerical solution on a regular lattice for processing reasons. As a result, it may be necessary for the meshfree solution to be sampled onto a lattice. To sample field values onto the lattice, GFD with a complete basis is used to approximate functions and their derivatives. Below we have given an example in Figure 10, where the meshfree velocity field in the image on the left was sampled onto a lattice in order to calculate contour plots of the velocity magnitude.

[1] In general, the system will be non-symmetric. Here we use the BiCGSTAB method as implemented in the Eigen library [15].



(a) Before sampling - unit velocity vectors colored by magnitude.



(b) After sampling - contour plot of velocity magnitude.

Figure 10: Post processing example. Lid driven cavity solution (Re = 100) sampled from Poisson disk distribution onto a lattice.

CHAPTER VI

VALIDATION

6.1 2D Poisson with Dirichlet Boundary Conditions

The two dimensional Poisson equation with a constant source term reads:

$$\left(\frac{d^2T}{dx^2} + \frac{d^2T}{dy^2}\right) + Q = 0 \tag{6.1}$$

with $Q = 1$ subject to $T(x, y) = 0$ at boundary of the unit square. The analytical solution is known and is given by:

$$T(x, y) = \frac{16}{\pi^4} \sum_{n}^{\infty} \sum_{m}^{\infty} \frac{sin(\pi nx) sin(\pi my)}{nm(n^2 + m^2)} \tag{6.2}$$

Here we discretize the equation on uniform Poisson disk grids using GFD, MLS, and BCG resulting in a sparse system of $\approx (N \times N)$ equations:

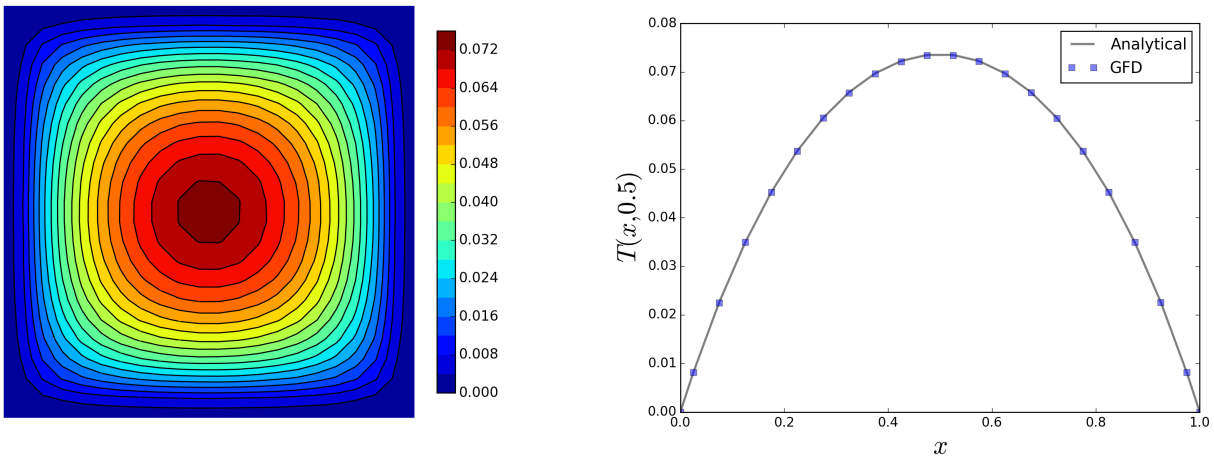
$$2(a_{4i} + a_{6i}) = -1 \tag{6.3}$$

where N is the target number of nodes we wish to span a given dimension and is used to scale the system such that $s \approx 1$, while a_{4i} and a_{6i} correspond to the appropriate Taylor coefficients given by Eq. 3.40 for a particular star node i. After populating the matrix coeffecients, this system is solved using Eigen's BiCGSTAB solver, where we set the residual to $\epsilon < 1 \times 10^{-10}$ [15]. Note that a quadratic basis is used for GFD and MLS with the constant basis treated as a known. The contours and center profile are presented in Figure 11.

The spatial convergence is presented in Figure 13 and was evaluated using the discrete L_2 relative error norm:

$$L_2 = \frac{\sum_i \left(u(x_i, y_i) - u^h(x_i, y_i)\right)^2}{\sum_i u(x_i, y_i)^2} \tag{6.4}$$

where $u^h(x_i, y_i)$ is the numerical solution and $u(x_i, y_i)$ is the analytical. In order to eliminate the error associated with the truncated infinite series, enough terms were carried out for both summations in Eq. 6.2 such that the norm maintained 3 significant digits as the number of terms in the series increased.



(a) $T(x, y) = c$, contour plot.

(b) Centerline profile comparison with analytical.

Figure 11: GFD solution for 2D Poisson equation with a constant source using the exponential weighting function where $k = 4, h = 1.5s, r_c = 1.5h$ and with $N \approx 20$ nodes spanwise.

Before proceeding we note that $W_3(h = 1.1s), W_e(h = 1.5s)$, and $W_5(h = 0.7s)$ correspond to $n \approx 16$ neighbors, $W_3(h = 1.25s)$ to $n \approx 20$, $W_5(h = 0.9s)$ to $n \approx 24$ and the range $W_e(h = 1.25s) - (h = 3.0s)$ corresponds to $n \approx 11$ to $n \approx 66$. Moreover, $W = 1$ corresponds to a uniform weighting function with a support domain of $n = 14$ neighbors. Numerical tests indicate that GFD, MLS, and BCG all under perform the classic central difference estimate as shown in Figure 13. The finite difference computations were performed using the sharp interface framework with the regular lattice positioned half a unit away from the wall. By dropping the mixed basis and reducing the search radius to four neighbors, the classic central difference stencil was obtained. In fact, within the sharp interface framework used here and with our treatment of the constant basis, it appears that the central difference curve is a lower limit for all three estimates. It is evident that tested forms of GFD and MLS have comparable accuracies with approximately quadratic convergence rates, while the inconsistent BCG estimate has error which is about 2-3 orders of magnitude higher with not even a linear convergence rate. BCG's poor convergence rate is expected as it was shown earlier using a Taylor series expansion that the scheme is inconsistent. The curve marked as $W_5 - B - (h = 0.7s)$ in Figure 13c corresponds to the Brookshaw estimate without the gradient correction. Compared to the corrected gradient curve (i.e., $W_5(h = 0.7s)$), we see applying the gradient correction slightly improves accuracy for some of the uniform Poisson disk grids used here.

Notice that in Figure 13b, MLS with the grid $N = 80$, required adjusting the smoothing length for each kernel in order to maintain the convergence rate. Plotting the error $|u_i - u_i^h|$

indicated an isolated region with error an order of magnitude higher than the rest of the domain. Since the collocated version of MLS used here differs from GFD only by accounting for the spatial variation of the moment matrix, we suspect these extra terms to potentially be the problem. On another note, Jensen argued that since not all derivatives appear in a PDE, they can be dropped in the moment matrix in order to reduce the matrix size [16]. Curve W_5^* in Figure 13a shows GFD with the mixed basis dropped. The curve indicates the reasoning holds up to a certain resolution, after which the convergence rate stagnates due to errors stemming from the dropped Taylor series term. Since the point at which truncation error due to the mixed derivative becomes significant is unknown apriori, this approach is not recommended.

Testing GFD (constant basis removed) with a uniform weight $W = 1$ indicates that - within our sharp interface formulation - convergence order has little to do with the choice of weight function and is actually close to quadratic convergence regardless of kernel choice and despite only using a quadratic basis to estimate second derivatives. Moreover, kernel choice in GFD or MLS, appears to affect accuracy minimally as indicated by the narrow spread of the curves[1]. Lastly, Figure 13d, demonstrates that increasing the smoothing length decreases the accuracy of the approximation but simultaneously decreases the required number of iterations as shown in Figure 12. In 2D, the number of neighbors grows quadratically with the smoothing length, so the computational cost of matrix multiplication will still increase since the number of iterations was found to only halve with every doubling of the smoothing length.

With GFD and MLS having comparable accuracy and convergence rates and with GFD requiring less computations it was decided to choose GFD to explore several other test cases.

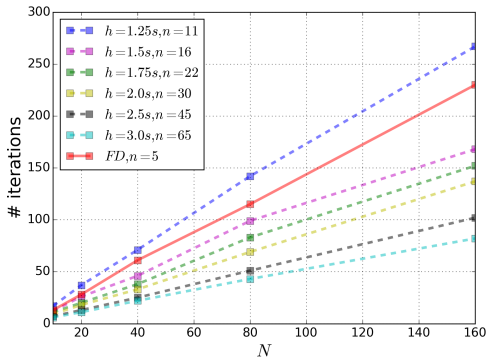
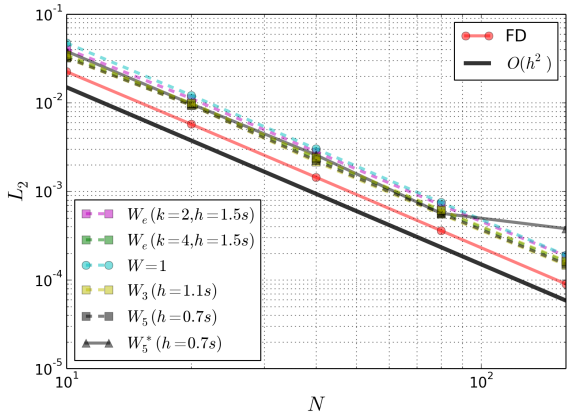
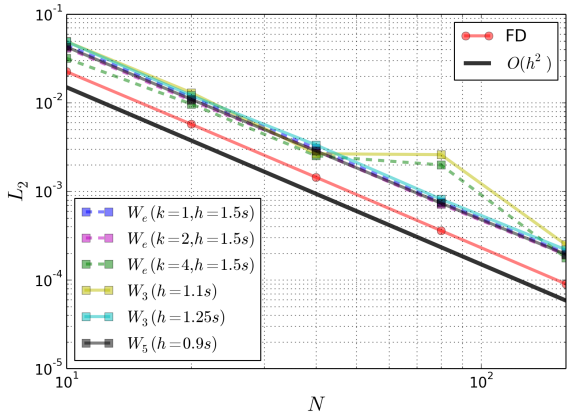


Figure 12: Number of iterations to reach $\epsilon < 1 \times 10^{-10}$ using BiCGSTAB as smoothing length h increased for GFD using $W_e(k = 1, r_{cutoff} = 1.5h)$.

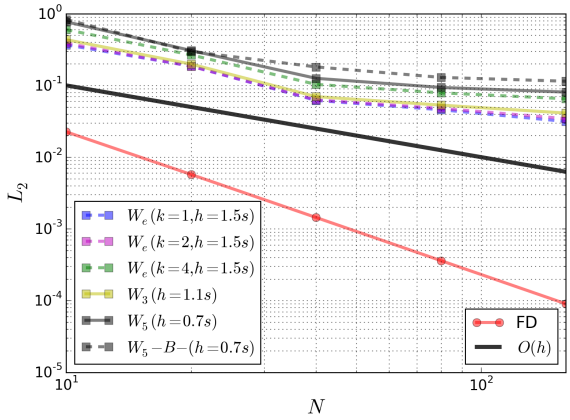
[1]This does not mean that the same results will hold if the constant basis is included. We will test this in future work.



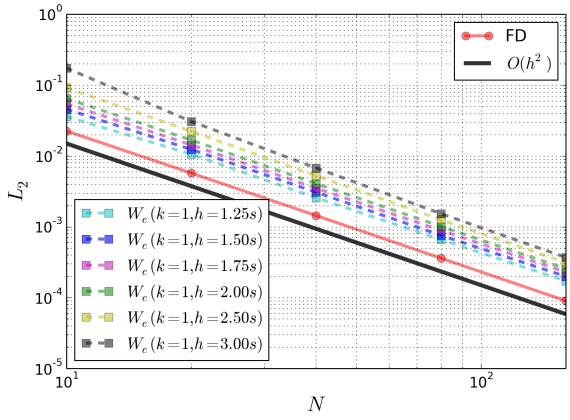
(a) GFD convergence using different kernels.



(b) MLS convergence using different kernels.



(c) BCG convergence using different kernels.



(d) Smoothing length varied (GFD).

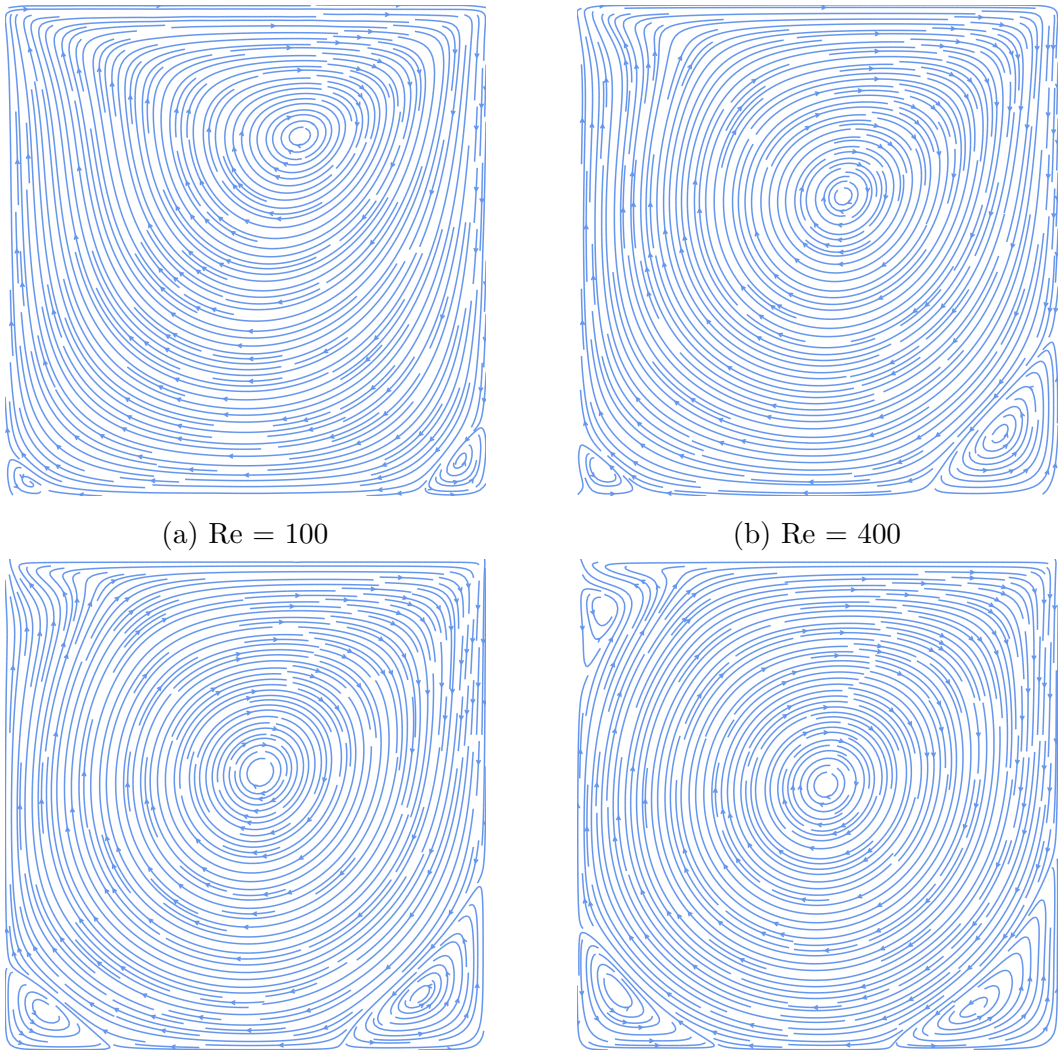
Figure 13: Spatial convergence for 2D Poisson equation for GFD, MLS, and BCG. Five irregular grids generated using uniform Poisson disk sampling corresponding to $N^2 \approx 10^2, 20^2, 40^2, 80^2, 160^2$ total nodes.

6.2 2D Cavity

The configuration we test is as follows: an incompressible, Newtonian fluid initially at rest is bounded by a square domain when at time $t = 0$ the top wall suddenly moves with a constant speed U to the right, while the rest of the walls remain stationary. Depending on the Reynolds number, eventually the fluid reaches a steady state solution. This test case does not have a known analytical solution so numerical results are compared to the available Ghia data set [12] as well as to a Lattice Boltzmann Method (LBM) solution. The explicit fractional step Navier-Stokes solver described in Section 5.1 is used to march the solution forward in time until steady state is reached. The collocation is performed using GFD with $W_5(h = 0.7s)$ resulting on average $n = 16$ neighbors. As in the 2D Poisson test case, the side of the square domain is scaled to N to improve conditioning of the moment matrices such that $s \approx 1$. Again N is the desired number of nodes to span a side. In addition, respective grids are generated using the uniform Poisson disk algorithm. The lid speed is set to $U = 1$ and so different Reynolds numbers are obtained via the kinematic viscosity as follows:

$$\nu = \frac{UN}{Re} \tag{6.5}$$

We simulate the following Reynolds numbers: 100, 400, 1000, and 3200. The cavity streamlines for the four cases are presented in Figure 14. In Figure 14, we see the GFD solution is able to capture the strong clockwise vortex as well as the vortex core's slow movement towards center of the cavity as the Reynolds number increases. Additionally, with increasing Reynolds number, we see the formation of secondary counter-clockwise vortices at the corners of the cavity. With the main flow characterstics captured, we next compare the velocity components in Figures 15, 16, 17, 18 to the Ghia data set as well as to an LBM solution. We see there is good agreement with both data sets for both components of velocity for the entire range of Reynolds numbers considered here.



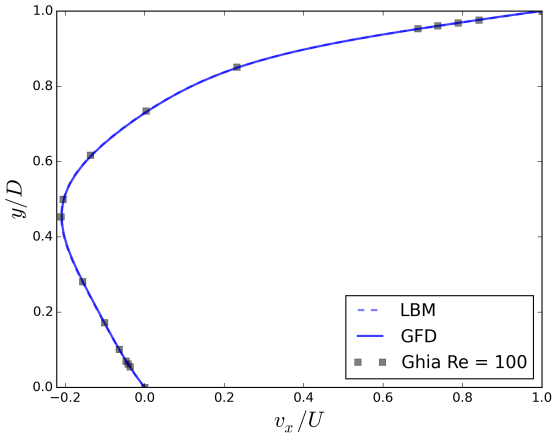
(a) Re = 100

(b) Re = 400

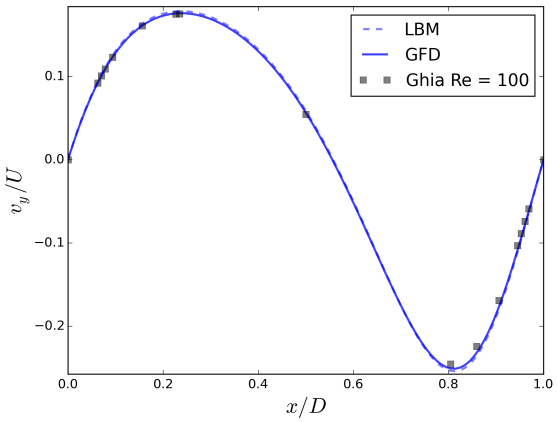
(c) Re = 1000

(d) Re = 3200

Figure 14: Lid driven cavity streamlines for Re=100,400,1000 and 3200.

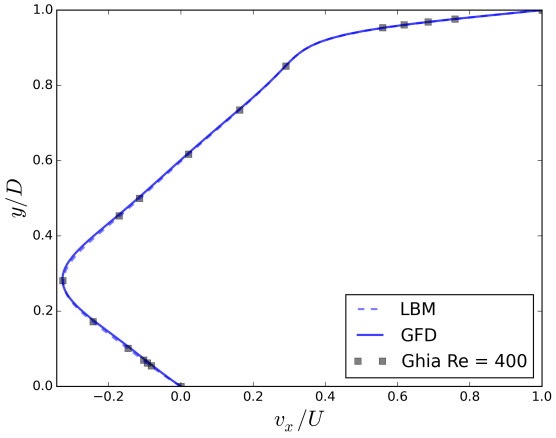


(a) Horizontal velocity component.

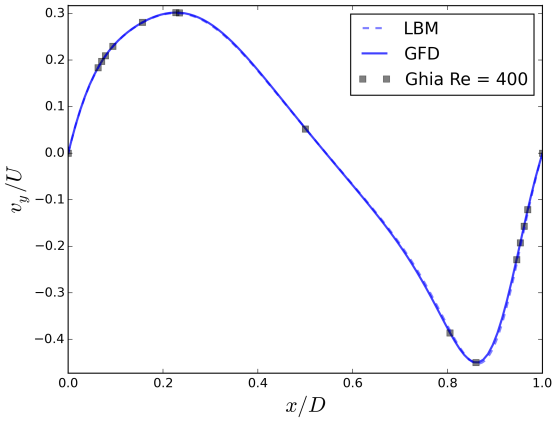


(b) Vertical velocity component.

Figure 15: Lid driven cavity steady state solution for velocity components, Re = 100. LBM grid size is 200×200, while GFD is $\approx 100 \times 100$.

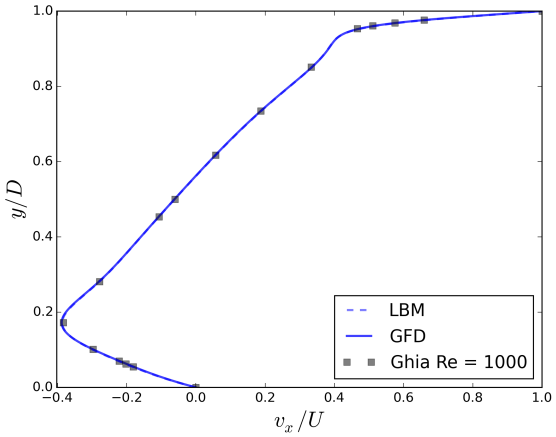


(a) Horizontal velocity component.

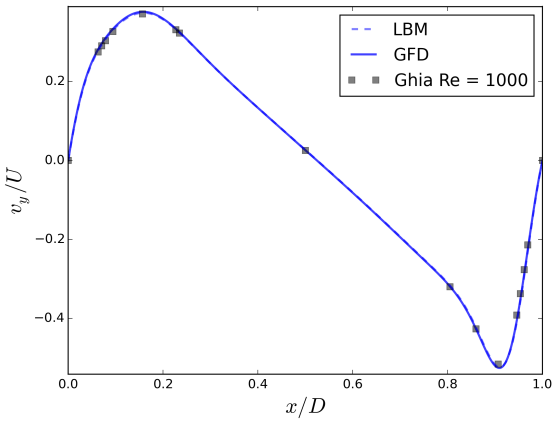


(b) Vertical velocity component.

Figure 16: Lid driven cavity steady state solution for velocity components, Re = 400. LBM grid size is 200×200, while GFD is $\approx 100 \times 100$.

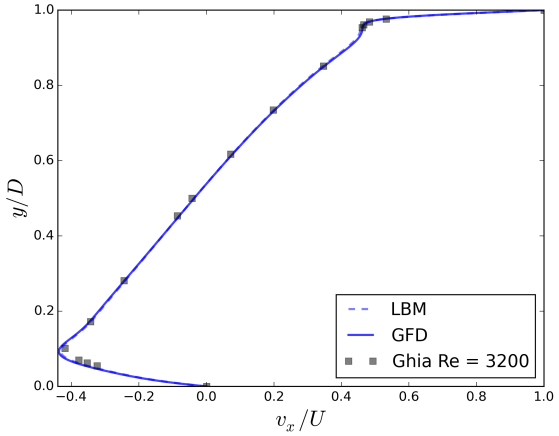


(a) Horizontal velocity component.

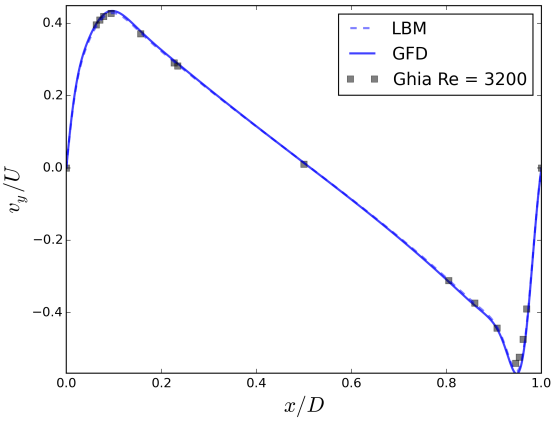


(b) Vertical velocity component.

Figure 17: Lid driven cavity steady state solution for velocity components, Re = 1000. LBM grid size is 200×200, while GFD is $\approx 300 \times 300$.



(a) Horizontal velocity component.

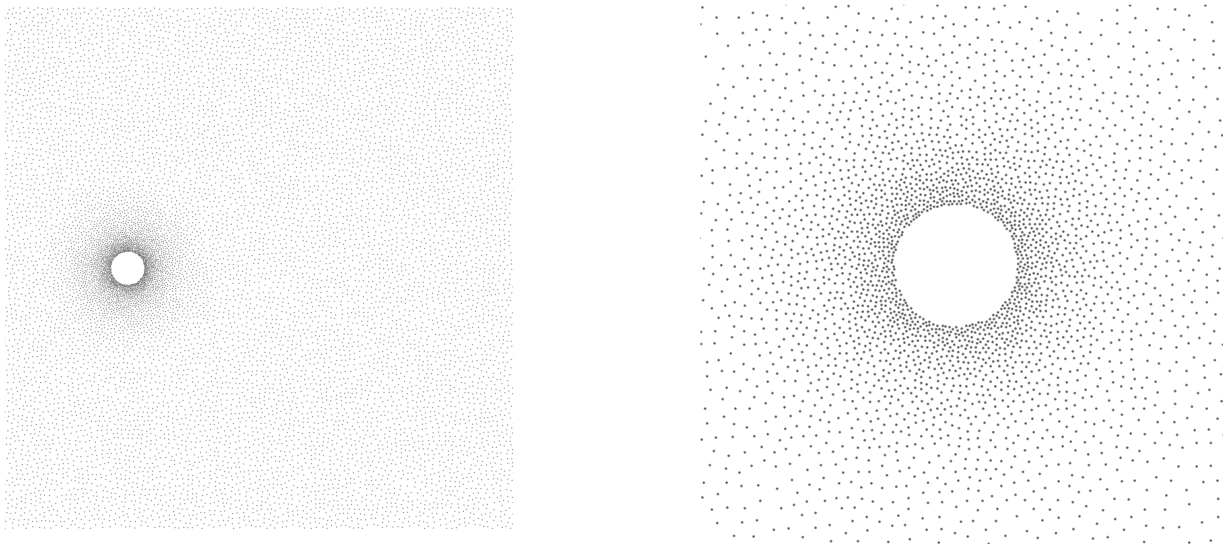


(b) Vertical velocity component.

Figure 18: Lid driven cavity steady state solution for velocity components, Re = 3200. LBM and GFD grid size is $\approx 500 \times 500$.

6.3 2D Uniform Flow Over Cylinder

The configuration tested here is steady 2D uniform flow over a cylinder in an "unbounded" domain with the Reynolds number defined as $Re = U_\infty D/\nu$, where D is the cylinder diameter. The test case has been thoroughly investigated by others and it is generally agreed that past a critical Reynolds number of $Re_c \approx 46$ the two standing symmetric vortices detach from the cylinder surface and begin to oscillate [34]. Here we consider the steady case where $Re = 40$. To model the "unbounded" domain we consider a $32D \times 32D$ square domain with the cylinder placed $8D$ from the inlet. Since the gradients with largest magnitude are localized around the cylinder wall and the cylinder dimensions are small relative to the computational domain, it is necessary to consider a variable resolution grid. We generate the grid using the variable radii Poisson disk algorithm with grid points clustered near the cylinder wall in order to capture the boundary layer. See Figure 19 below for an example grid used.



(a) Zoomed out. (b) Zoomed in.

Figure 19: Variable radii Poisson disk distribution with refinement around the cylinder using a maximum to minimum spacing ratio of 6. The variable radii function is simply a linear ramp function where the minimum spacing $s_{min} = 1/6 s_{max}$ is at the wall and the maximum spacing $s_{max} = 0.88$ is a few diameters away from the wall. Note this example is a $16D \times 16D$ domain, ghost nodes are not shown.

We compare the pressure coefficient C_p and skin friction coefficient C_f to the boundary fitted grid results presented in Tseng and Ferziger [33]. The pressure coefficient is calculated

as:

$$C_p = \frac{2(p - p_\infty)}{\rho U_\infty^2} \tag{6.6}$$

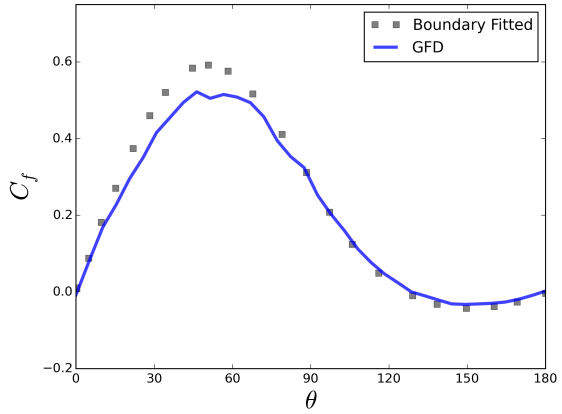
and the skin friction coefficient is calculated as:

$$C_f = \frac{2\tau_w}{\rho U_\infty^2} \tag{6.7}$$

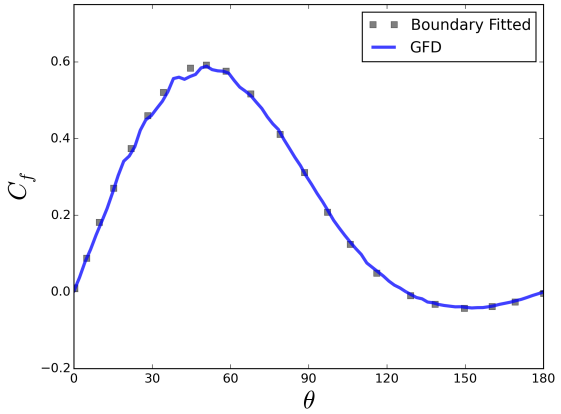
where τ_w is the wall shear stress defined as:

$$\tau_w = \nu \left(\nabla \vec{u}\right) \cdot \hat{n} \tag{6.8}$$

with the sign determined by whether or not it is aligned with the unit tangent. Both coefficients are calculated as a function of θ, with θ measured clockwise from the stagnation point $\theta = 0$. In order to calculate wall shear stress with an immersed boundary technique, some additional work must be done during post processing since the computational points do not lie on the boundary. We estimate $\nabla \vec{u}_w \cdot \hat{n}$ using the GFD approximation with a full basis. Here we construct the local approximation at each wall point $\vec{x}_w$ defining the immersed cylinder boundary. By taking $n \approx 20$ neighbors in the local approximation, we were able to get very smoothed profiles for C_f, despite the irregular grid points. As shown in the figures, both coefficients match the boundary fitted grid solution presented in Tseng and Ferziger. To get better agreement for the pressure coefficient, the height of the domain needed to be increased to $40D$ in order to simulate an "unbounded" domain.

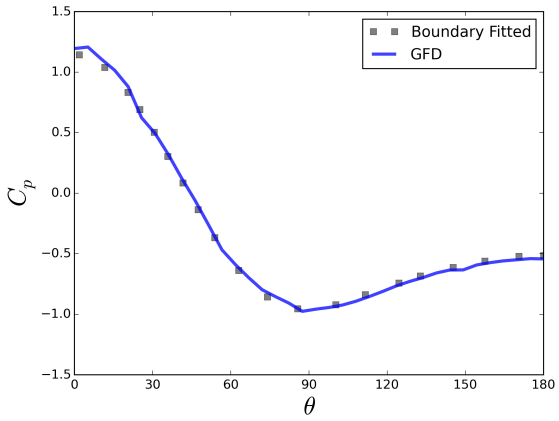


(a) C_f at lower resolution using $s_{min} = 1/10 s_{max}$ with a $32D \times 40D$ domain. Approximately 20 nodes across diameter with 9,400 total fluid nodes.

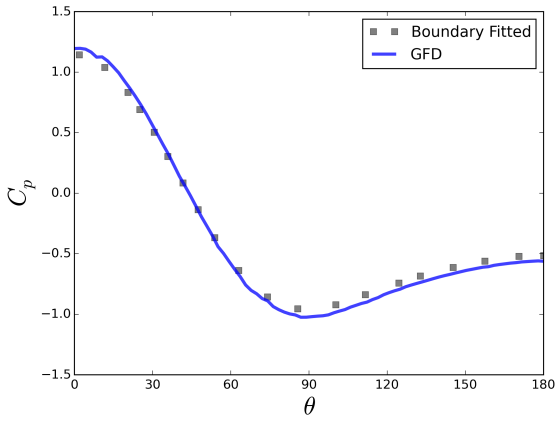


(b) C_f at higher resolution using $s_{min} = 1/6 s_{max}$ with a $32D \times 32D$ domain. Approximately 48 nodes across diameter with 84,000 total fluid nodes.

Figure 20: Comparison of skin friction coefficient C_f at the cylinder surface for $Re = 40$ using different resolutions compared to the boundary fitted solution presented in Reference [33].



(a) C_p at lower resolution using $s_{min} = 1/10 s_{max}$ but with a a $32D \times 40D$ domain. Approximately 20 nodes across diameter with 9,400 total fluid nodes.



(b) C_p at higher resolution using $s_{min} = 1/6 s_{max}$ but with a $32D \times 32D$ domain. Approximately 48 nodes across diameter with 84,000 total fluid nodes.

Figure 21: Comparison of pressure coefficient C_p at the cylinder surface for $Re = 40$ using different domain sizes compared to the boundary fitted solution presented in Reference [33].

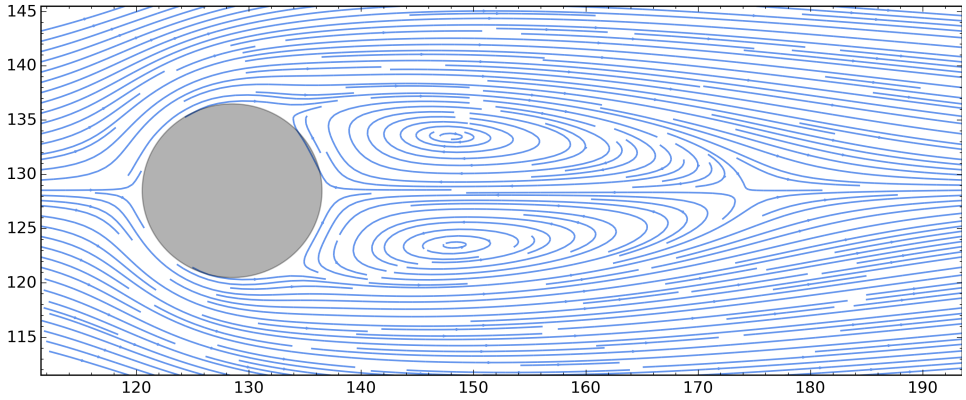


Figure 22: Uniform flow over cylinder, streamlines plotted for Re = 40 showing the two symmetrical standing vortices.

6.4 2013 FDA Cardiovascular Benchmark

In 2013, the U.S. Food and Drug Administration released a cardiovascular benchmark in order to compare experimental and computational modeling practices among different groups. As seen in Figure 23 the problem is a simple geometry with a sudden contraction and downstream conical diffuser. Stewart et al., compiled the 28 simulation data sets from around the world in addition to 4 experimental data sets for different Reynolds numbers - 500, 2000, 3500, 5000, and 6500 as measured at the nozzle [31]. We first consider a uniform Poisson disk grid and use GFD with $n = 18$ neighbors to compute the steady state solution for a low Reynolds number of $Re = 50$. We then consider a higher Reynolds number of $Re = 500$, using a uniformly spaced grid and compute the steady state solution using GFD with $n = 13$ neighbors. Due to unresolved stability issues, we were unable to use the Poisson disk grid at higher Reynolds numbers. For both cases, we compare results to a Lattice Boltzmann Method (LBM) solution. Additionally, for $Re = 500$, we compare GFD results to the experimental data sets presented in Reference [31].

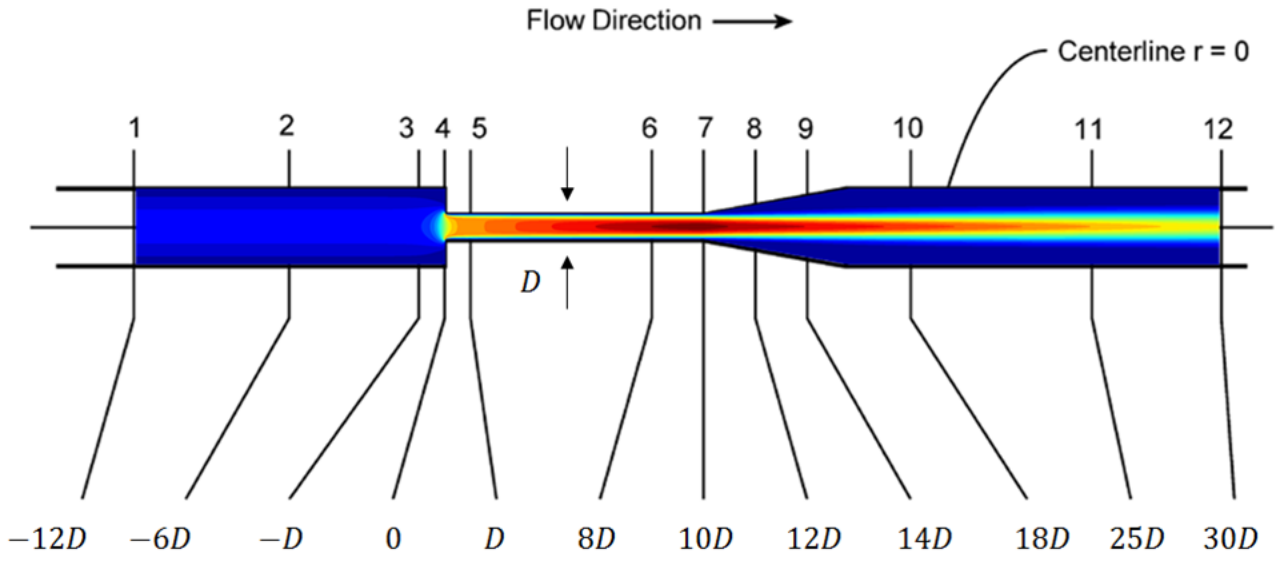


Figure 23: 2013 FDA Benchmark. Steady flow through a nozzle with a sudden contraction and downstream conical diffuser. Geometry modified from Stewart et al.[31]. The overlapped velocity contour plot is the Lattice Boltzmann Method (LBM) solution for Re = 500.

Model

Rather than use a $3D$ model, we use a $2D$ axisymmetric model by switching the cartesian operators with the cylindrical operators as is detailed in Chapter 5. The centerline is considered part of the immersed boundary and is located $s/2$ away from any node. When the uniform Poisson disk grid is used, we target a certain number of nodes n_n across the nozzle radius scaling the system by n_n such that $s \approx 1$. A sample uniform Poisson disk grid is shown in Figure 24. The Reynolds number is obtained by setting the kinematic viscosity such that $\nu = 6n_i u/Re$, where n_i is the number of nodes across the inlet, $u = 1$ and is the inlet velocity, and Re is the desired Reynolds number as measured at the nozzle. The factor of six results from doubling the radius to get the inlet diameter and the fact $Re_{inlet} = 1/3Re_{throat}$ due to the geometry and conservation of mass. At the inlet, we specify a parabolic velocity profile. Depending on the Reynolds number, we place the outlet far enough downstream such that fully developed flow is achieved (i.e., $\dfrac{\partial v_z}{\partial n} = 0$). At the centerline, a symmetry boundary condition is imposed (i.e., $v_r = 0, \dfrac{\partial v_z}{\partial r} = 0$).

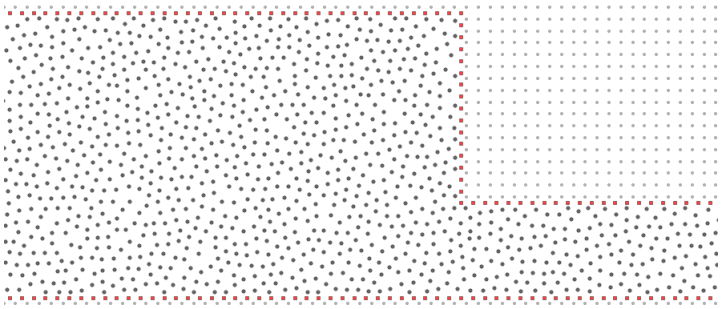


Figure 24: Axisymmetric grid used for FDA test case with $n_n \approx 8$ across the throat. The image is zoomed in on axial location z_4. The red nodes define the immersed boundary while the small gray nodes are ghost nodes used to modify the computational stencil.

Re = 50 (GFD with uniform Poisson disk grid)

We compare the GFD solution to an axisymmetric Lattice Boltzmann Method (LBM) solution as implemented by us and as described in Zhou [36]. To simplify comparison, rather than interpolate LBM and GFD solutions to the exact axial locations, we instead sample the GFD solution onto a lattice which is offset by $\Delta s/2$ and which is at an equivalent resolution as the meshfree grid. As a result, sampled axial locations are close to LBM lattice locations. We first introduce low resolution GFD results as compared to high resolution LBM resolutions and then later introduce higher resolution GFD results.

Figure 25 shows the sampled axial profiles as measured at several of the axial locations

when using a resolution of $n_n \approx 8$ while Figure 26 shows the centerline velocity profile. We see good agreement with the axisymmetric LBM solution except for locations z_4 and z_9. At z_4, due to the sudden contraction, the fluid abruptly accelerates to a plug like profile with a maximum normalized velocity of $v_z/U_{avg} = 10.8$. At z_4, the disagreement with the LBM solution is mainly with regard to the profile near the wall where sharp gradients exist. Since the GFD profile underestimates the solution as compared to the high resolution LBM solution, we see that the flow rate is not exactly conserved as indicated by Figure 27. This is to be expected, since GFD is a non-conservative approximation and furthermore, the non-conservative form of the Navier-Stokes equation were solved. Continuing down the nozzle, the flow develops into a parabolic profile with a maximum normalized velocity of $v_z/U_{avg} = 17.8$ by z_6 - underestimating the basic conservation analysis value of 18.0 [2]. As the flow enters the conical diffuser, we see the GFD solution captures the effect of the adverse pressure gradients on near wall velocity profiles as shown at z_8 and z_9. At higher Reynolds number, we expect flow reversals to occur in this region. Upon exiting the diffuser, since the Reynolds number is low, the flow redevelops into a parabola by z_{11} as expected.

Figure 25 also shows the axial profiles at an increased resolution of $n_n \approx 12$ nodes across the nozzle. We see the velocity profiles match better with the LBM solution with $v_z/U_{avg} = 10.7$ at z_4 and $v_x/U_{avg} = 18.0$ at z_6, matching the expected fully developed maximum velocity. More importantly, we see the flow rate is better conserved with a maximum change of only three percent $\Delta\bar{Q}_{max} = .03$. However, there are still noticeable discrepancies between the GFD and LBM solution near the wall of the sudden contraction, although visibly they are reduced. We expect as the resolution increases, for this discrepancy to be resolved. Lastly, we found we were unable to simulate higher Reynolds numbers for this particular test case using GFD with a uniform Poisson disk grid. As we increased the resolution (to handle higher Reynolds numbers), stability issues near the centerline were encountered. The oscillations grew overtime and occurred regardless of the size of the time step. Further investigation is needed to pin point the root of the problem.

Re = 500 (GFD with uniform lattice)

Unable to simulate higher Reynolds number using irregular grids, we decided to test GFD using a uniform lattice. It was found that using approximately $n_n = 30$ nodes across the nozzle allowed us to suppress the centerline oscillations. The smoothing length for the quintic kernel W_5 was set such that 13 neighbors were considered in the construction of the local

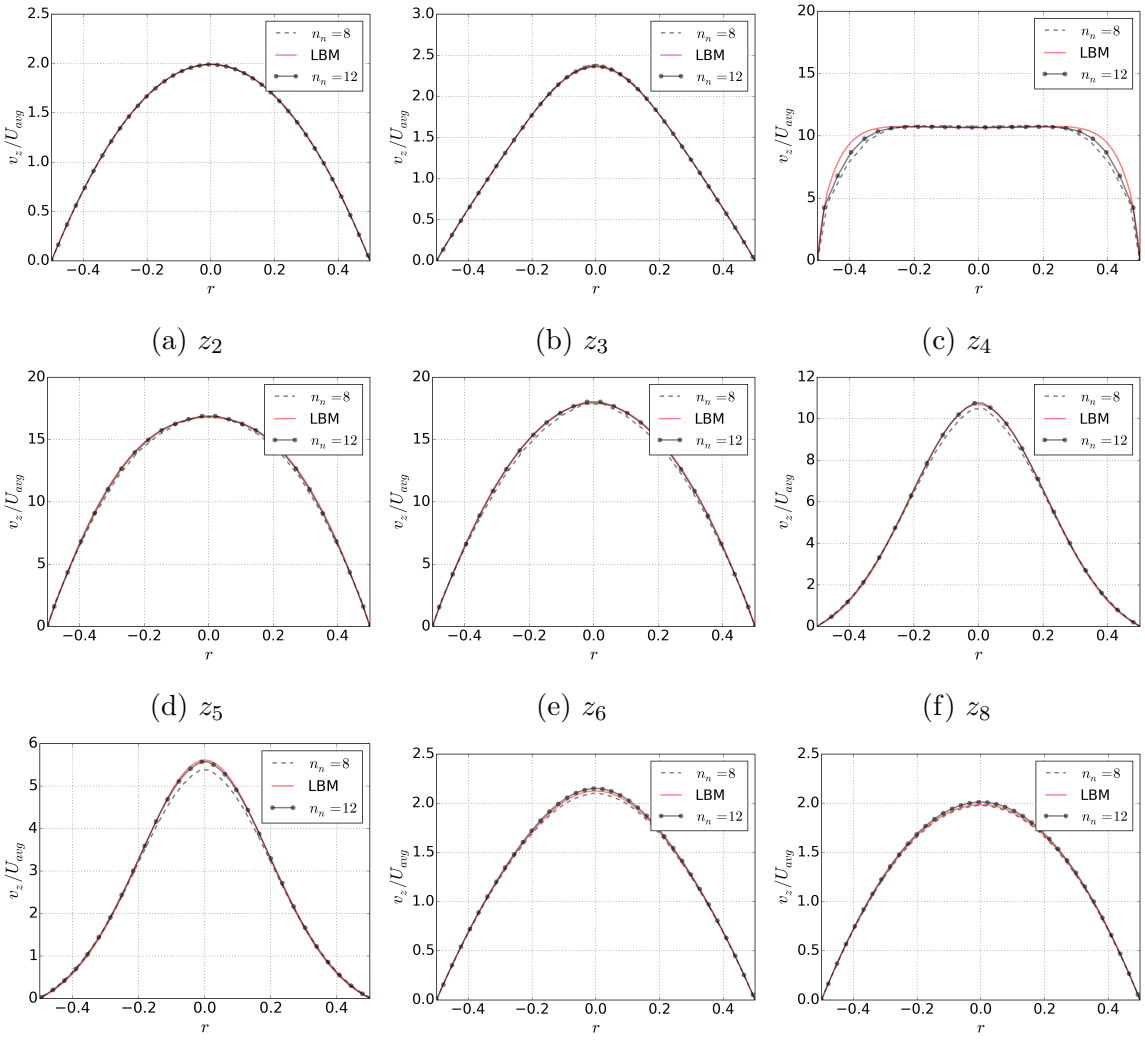
[2] At a higher Reynolds number, the $10D$ nozzle length will be insufficient for the flow to fully developed and as such we will not obtain the maximum velocity as predicted by assuming full developed flow.

approximations. In Figure 28, we compare the GFD solution to the four available experimental data sets as well as to the axisymmetric LBM solution. Note the large variance between the experimental data sets, especially at critical zones such as the sudden contraction. The variation suggests some of the groups were more systematic in carrying out planar particle image velocimetry (PIV) measurements than others. We note that even at a high Reynolds number, despite attacking the problem from different frameworks (i.e.,Navier-Stokes equations versus Boltzmann transport equation), GFD and LBM axial profiles overlap nearly perfectly with the exception of small differences at the sudden contraction [3].

To make Figure 28 more readable, we collapse the four experimental datasets into 95 percent confidence intervals plotted about the average of the four profiles. To calculate the confidence intervals, it was necessary to linearly interpolate the raw experimental axial velocity data onto the same r locations. The confidence intervals for the axial profiles are shown in Figure 29. We see that both GFD and LBM fall within the axial profile confidence intervals for all z locations with the exception of z_3 and z_4. Note that at z_4, most of the experimental curves - with the exception of EXP-763 - significantly underestimate the axial velocity profile. For $Re = 500$, we see that as the flow exits the nozzle and enters the conical diffuser, we start to see flow reversals at z_8 as is indicated by the negative slope of the velocity profile at the wall. Exiting the diffuser, the strong central jet that has formed continues down the tube, with pronounced flow reversal occurring at z_{10} and z_{11}.

On a final note, we compare the GFD solution to the centerline profile and respective confidence intervals in Figure 30a. As could be guessed from the axial profile confidence intervals, we see the GFD solution for the centerline profile falls within all confidence intervals and moreover, overlaps the LBM solution. The centerline velocity starts at $v_x/U_{avg} = 10.7$ at the throat, increasing to the maximum velocity attained at location z_7 with $v_x/U_{avg} = 16.3$. Exiting the nozzle, the centerline velocity steadily decreases to $v_x/U_{avg} = 10.3$ by z_{12}, confirming that applying an outlet extension was necessary. Lastly, we calculate the normalized flow rate in Figure 30b. We see the maximum change in flow rate is limited to $\Delta\bar{Q}_{max} \approx .02$ and is followed by a quick recovery after the sudden contraction.

[3]Note that the axisymmetric LBM grid is at twice the resolution.



(a) z_2 (b) z_3 (c) z_4

(d) z_5 (e) z_6 (f) z_8

(g) z_9 (h) z_{10} (i) z_{11}

Figure 25: 2013 FDA Benchmark (Re = 50), GFD solution for axial velocity profiles at various z locations using $n_n \approx 8$ nodes across nozzle radius (19,000 total) and $n_n \approx 12$ nodes (40,000 total). Compared to LBM solution obtained using $n_n = 40$ nodes across nozzle radius. All velocities are normalized with respect to the average inlet velocity while radial positions are normalized to fall between $(-0.5, 0.5)$. GFD solution used ≈ 18 neighbors.

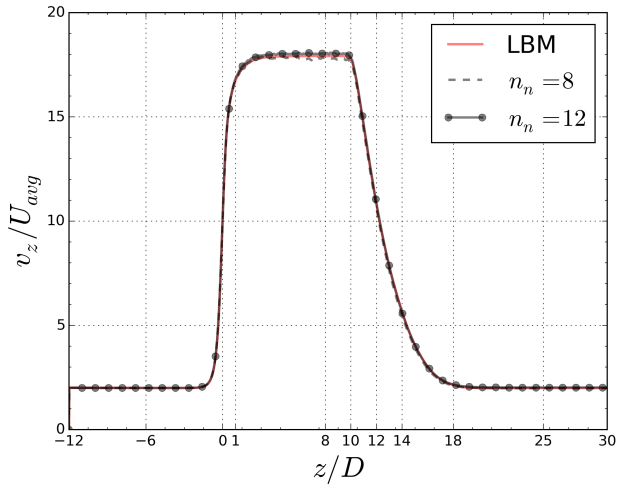


Figure 26: 2013 FDA Benchmark (Re = 50), GFD solution for centerline velocity profiles compared at two different resolutions.

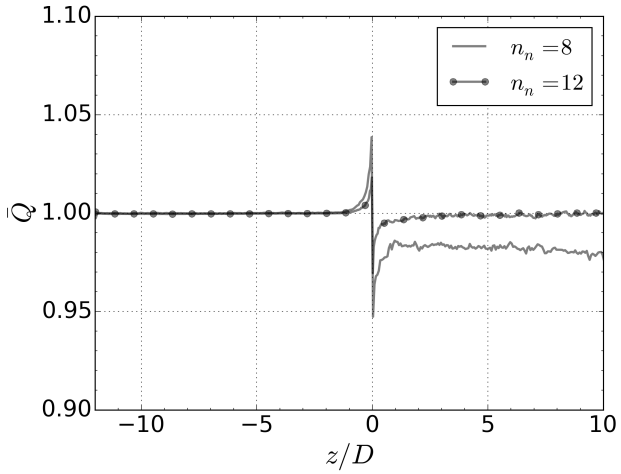


Figure 27: 2013 FDA Benchmark (Re = 50), normalized flow rates for GFD solution calculated using Simpson's rule at two different resolutions.

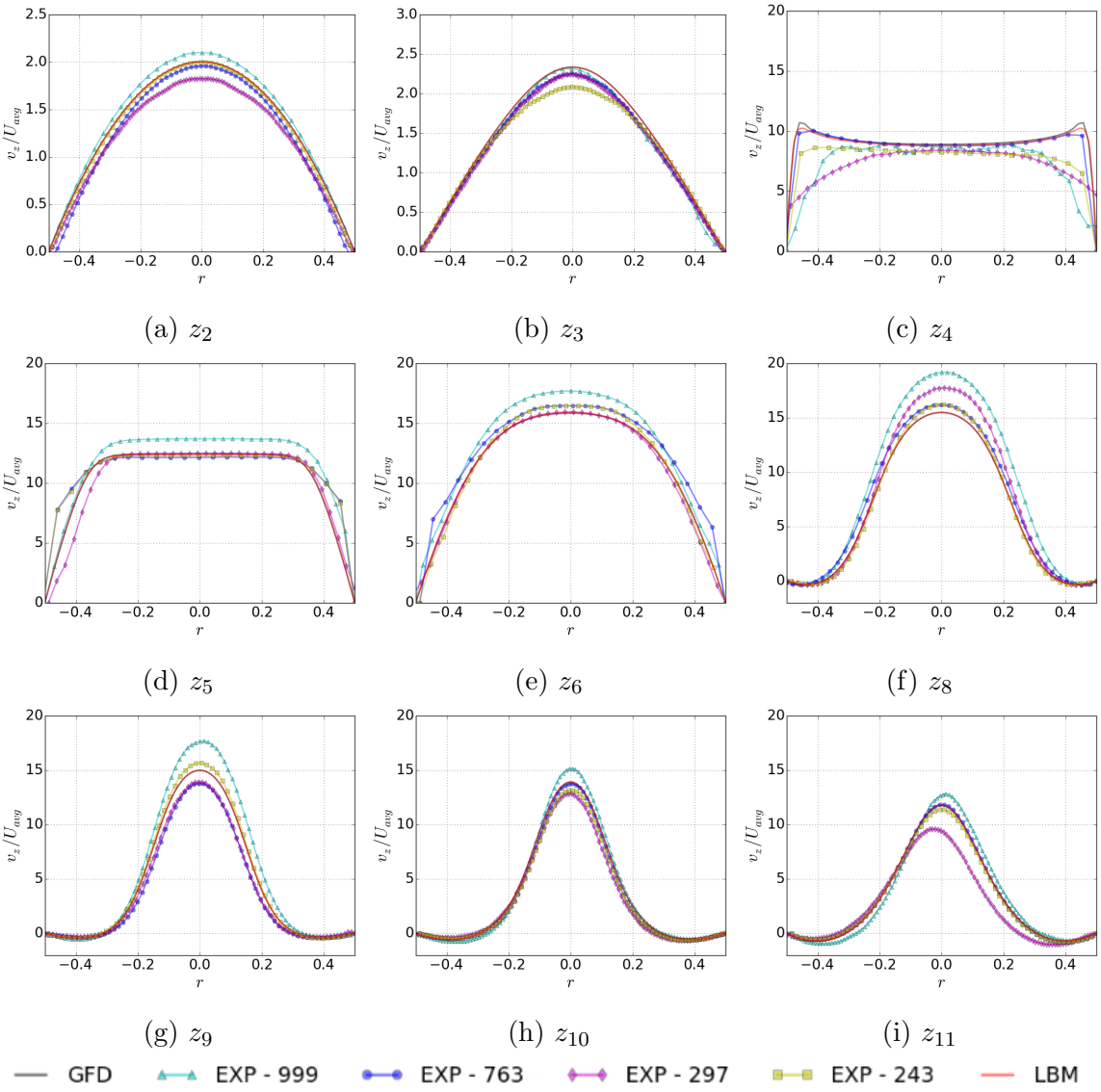
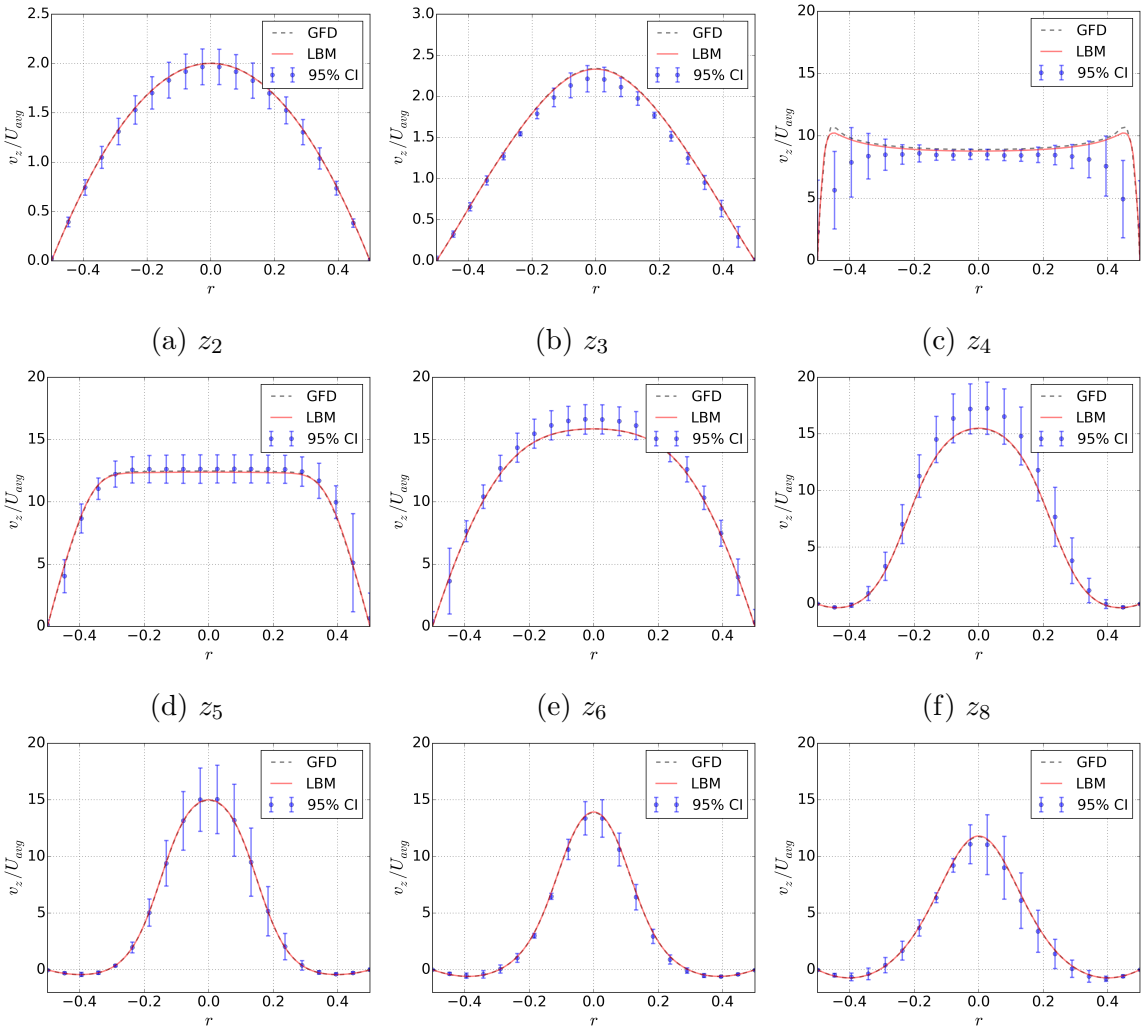


Figure 28: 2013 FDA Benchmark (Re = 500), GFD lattice solution ($n_n = 30$) for axial velocity profiles at various z locations compared to the LBM solution ($n_n = 60$) and to the four raw experimental data sets. All velocities are normalized with respect to the average inlet velocity while radial positions are normalized to fall between $(-0.5, 0.5)$. GFD solution used approximately 400,000 nodes with each node having 13 neighbors.

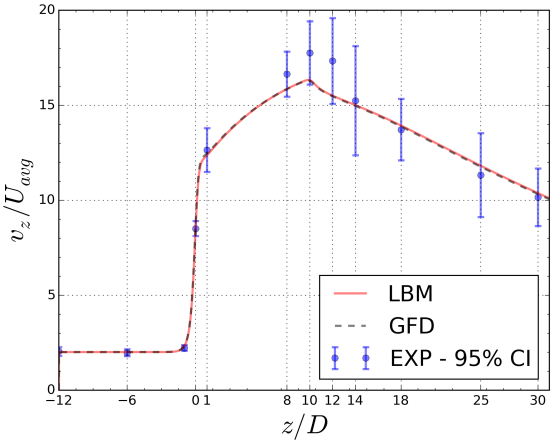


(a) z_2 (b) z_3 (c) z_4

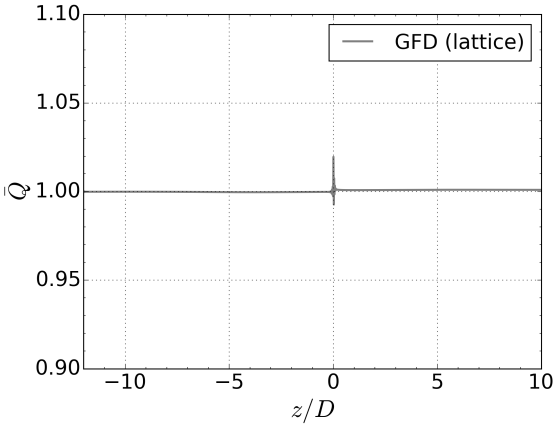
(d) z_5 (e) z_6 (f) z_8

(g) z_9 (h) z_{10} (i) z_{11}

Figure 29: 2013 FDA Benchmark (Re = 500). GFD lattice solution ($n_n = 30$) for axial profiles at various z locations compared to the 95 percent confidence intervals for the four experimental data sets, as well as to the LBM solution ($n_n = 60$).



(a) Centerline profile, $n_n = 30$.



(b) Normalized flow rate, $n_n = 30$.

Figure 30: 2013 FDA Benchmark (Re = 500), normalized flow rate and centerline profile using GFD on a lattice with 13 neighbors.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this work we first introduced and summarized the following collocated meshfree methods: General Finite Differences (GFD), Moving Least Squares (MLS) and Smoothed Particle Hydrodynamics (SPH). In GFD, the local approximation for function and derivative approximations is obtained after minimizing the weighted error of a local Taylor series approximation. As opposed to MLS, which introduces a continuous weight function at each neighboring point - instead of discrete weights - resulting in an approximation as many times differentiable as is the weight function or highest order polynomial. From a collocation viewpoint, MLS can be interpreted as a deviation about the GFD approximation. Moreover, MLS can also be interpreted as the form SPH takes after the weight function is corrected to enforce the polynomial reproducing conditions. For SPH, we summarize Taylor series based corrections aimed at restoring numerical consistency, namely CSPM, MSPH, Fatehi and Manzari's approach, and BCG. Additionally, following Belytschko et al. discussion on polynomial reproducing conditions, we show that Method 2 and Method 3, proposed by the group, are equivalent to GFD and MSPH.

Subsequently, we describe the main contribution of this work - an extension of the sharp interface variant of the Immersed Boundary method to collocated meshfree methods. Using linear interpolation matrices for Dirichlet and Neumann conditions, we modify the computational stencil for nodes whose support domain intersects a boundary element such that the appropriate boundary conditions are approximately enforced without the requirement that nodes be positioned exactly on the boundary. The stencil modification takes into account that extrapolated ghosts values are a linear combination of the star node, the intersecting boundary point, and a third interior node in the star node's support domain.

To generate the meshfree grids used in the validation test cases, we implemented a uniform and variable radii Poisson disk algorithm. In uniform Poisson disk sampling, grid points are randomly generated such that a minimum spacing between grid points is guaranteed. In variable Poisson disk sampling, the minimum spacing between grid points varies and is determined by a spacing function which describes the spatial variation of each grid point's "bubble". By using an image's greyscale as the spacing function, we showed the algorithm can be used to generate grids with varying resolution.

Using the sharp interface framework and uniform Poisson disk sampling, we then proceeded to evaluate the numerical accuracy and convergence rate of GFD, MLS, and BCG on a simple 2D Poisson equation. When the constant basis is removed, we determined that GFD and MLS have comparable accuracy and convergence rates with GFD requiring less computations, while BCG - as expected - suffered from significant numerical inconsistencies lacking even linear convergence.

Limiting our scope to GFD, we explored several incompressible fluid test cases using an explicit fractional step method to solve the Navier-Stokes equations. For the lid driven cavity case, numerical results agree favorably with the Ghia data set and Lattice Boltzmann Method (LBM) solutions throughout the range of Reynolds numbers considered (i.e., Re = 100 to 3200). We then presented results for uniform flow over a cylinder for Re = 40. To capture the localized gradients near the cylinder surface, the variable radii Poisson disk algorithm was used to cluster points near the cylinder surface according to a linear ramp function. The grids considered used a maximum to minimum spacing ratio of 6 and 10 respectively. We were able to accurately capture the skin friction and pressure coefficients evaluated at the cylinder's surface using the variable resolution meshfree grid as was shown with a direct comparison to a boundary fitted solution.

The last test case considered was the 2013 Food and Drug Administration cardiovascular benchmark. We first presented results for a low Reynolds number of $Re = 50$ obtained using a uniform Poisson disk grid. To model the $3D$ steady laminar flow, we assumed the flow is axisymmetric with respect to the centerline and as such discretized the simplified Navier-Stokes equations in cylindrical coordinates. Sampling from the uniform Poisson disk grid onto a lattice, we compared the axial profiles at various z locations to a high resolution axisymmetric LBM solution. It was determined that when using a low resolution, the non-conservative GFD approximation combined with the non-conservative form of the Navier-Stokes equations did not adequately conserve the flow rate. A significant change in flow rate of approximately five percent occurred at the sudden contraction z_4 when the meshfree grid consisted of $n_n \approx 8$ nodes across the nozzle. Increasing the resolution to $n_n \approx 12$ reduced this change to approximately three percent at the nozzle throat. Despite the disagreement at the sudden contraction, we found good agreement with the axial profiles at the other specified axial locations and were able to match centerline profiles.

Unable to overcome instabilities near the centerline for $Re = 500$, we decided to test GFD with the sharp interface boundary formulation on a uniformly spaced lattice using $n = 13$ neighbors in the support domain of each node. We found excellent agreement with LBM and more importantly showed the GFD solution fell within the 95 percent confidence intervals for the 4 available experimental data sets when comparing the 12 axial profiles as well as the

centerline velocity profile.

7.2 Future Work

There are a number of topics that we consider would be valuable to further explore. First, a direct comparison between the sharp interface formulation presented here and the direct boundary collocation approach would provide insight as to when to use one approach over the other. Additionally, we considered grids generated using uniform and variable Poisson disk sampling, it would be interesting to test GFD on grids generated using other techniques. Based on GFD lattice results for the FDA test case ($Re = 500$), we suspect a variable grid obtained by further recursively partitioning a lattice would be cheap to compute and would work well with the sharp interface framework presented here. Moreover, when using GFD with the sharp interface framework, the boundary condition is approximately imposed at several locations for a star node near the boundary. It is unclear whether this is advantageous. Exploring test cases where the immersed boundary is moving through the fluid could offer a potential answer to whether GFD has an advantage over other approaches when enforcing the sharp interface. On another note, our results are based on an explicit fraction step Navier-Stokes solver, naturally an extension of the code to a semi-implicit solver could potentially address stability issues encountered here. Furthermore, we did not test upwinding or filtering - two important topics traditionally used to improve stability. Incorporating these techniques would be necessary to simulate higher Reynolds number. Lastly, here the fluid test cases we have presented were more readily solved from the Eulerian frame, it would be valuable to extend the current code to a Lagrangian particle-based solver and address test cases more readily solved from the Lagrangian viewpoint.

REFERENCES

[1] Aluru, N. and Li, G., "Finite cloud method: a true meshless technique based on a fixed reproducing kernel approximation," *International Journal for Numerical Methods in Engineering*, vol. 50, no. 10, pp. 2373–2410, 2001.

[2] Belytschko, T., Krongauz, Y., Dolbow, J., and Gerlach, C., "On the completeness of meshfree particle methods," *International Journal for Numerical Methods in Engineering*, vol. 43, no. 5, pp. 785–819, 1998.

[3] Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., and Krysl, P., "Meshless methods: An overview and recent developments," *Computer Methods in Applied Mechanics and Engineering*, vol. 139, no. 14, pp. 3 – 47, 1996.

[4] Belytschko, T., Lu, Y. Y., and Gu, L., "Element-free galerkin methods," *International Journal for Numerical Methods in Engineering*, vol. 37, no. 2, pp. 229–256, 1994.

[5] Bonet, J. and Lok, T.-S., "Variational and momentum preservation aspects of smooth particle hydrodynamic formulations," *Computer Methods in applied mechanics and engineering*, vol. 180, no. 1, pp. 97–115, 1999.

[6] Brookshaw, L., "A method of calculating radiative heat diffusion in particle simulations," in *Proceedings of the Astronomical Society of Australia*, vol. 6, pp. 207–210, 1985.

[7] Chen, J. and Beraun, J., "A generalized smoothed particle hydrodynamics method for nonlinear dynamic problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 1, pp. 225–239, 2000.

[8] Chorin, A. J., "A numerical method for solving incompressible viscous flow problems," *Journal of computational physics*, vol. 2, no. 1, pp. 12–26, 1967.

[9] Chorin, A. J., "Numerical solution of the navier-stokes equations," *Mathematics of computation*, vol. 22, no. 104, pp. 745–762, 1968.

[10] Colagrossi, A. and Landrini, M., "Numerical simulation of interfacial flows by smoothed particle hydrodynamics," *Journal of Computational Physics*, vol. 191, no. 2, pp. 448–475, 2003.

[11] Fatehi, R. and Manzari, M., "Error estimation in smoothed particle hydrodynamics and a new scheme for second derivatives," *Computers & Mathematics with Applications*, vol. 61, no. 2, pp. 482–498, 2011.

[12] Ghia, U. and others, "High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method," *Journal of Computational Physics*, vol. 48, no. 3, pp. 387–411, 1982.

[13] Gingold, R. A. and Monaghan, J. J., "Smoothed particle hydrodynamics - theory and application to non-spherical stars," *Monthly Notices of the Royal Astronomical Society*, vol. 181, pp. 375–389, Nov. 1977.

[14] Gossler, A., "Moving least-squares: A numerical differentiation method for irregularly spaced calculation points," tech. rep., Sandia National Laboratories, 2001.

[15] Guennebaud, G., Jacob, B., and others, "Eigen v3." http://eigen.tuxfamily.org, 2010.

[16] Jensen, P. S., "Finite difference techniques for variable grids," *Computers and Structures*, vol. 2, no. 12, pp. 17–29, 1972.

[17] Khorasanizade, S. and Sousa, J. M., "A detailed study of lid-driven cavity flow at moderate reynolds numbers using incompressible sph," *International Journal for Numerical Methods in Fluids*, 2014.

[18] Lancaster, P. and Salkauskas, K., "Surfaces generated by moving least squares methods," *Math. Comp.*, vol. 37, no. 155, pp. 141–158, 1981.

[19] Liszka, T. and Orkisz, J., "The finite difference method at arbitrary irregular grids and its application in applied mechanics," *Computers and Structures*, vol. 11, pp. 83–95, 1979.

[20] Liu, M. and Liu, G., "Smoothed particle hydrodynamics (sph): an overview and recent developments," *Archives of computational methods in engineering*, vol. 17, no. 1, pp. 25–76, 2010.

[21] Liu, M., Xie, W., and Liu, G., "Modeling incompressible flows using a finite particle method," *Applied mathematical modelling*, vol. 29, no. 12, pp. 1252–1270, 2005.

[22] Liu, W. K., Jun, S., Zhang, Y. F., and others, "Reproducing kernel particle methods," *International journal for numerical methods in fluids*, vol. 20, no. 8-9, pp. 1081–1106, 1995.

[23] Mittal, R. and Iaccarino, G., "Immersed boundary methods," *Annu. Rev. Fluid Mech.*, vol. 37, pp. 239–261, 2005.

[24] Nayroles, B., Touzot, G., and Villon, P., "Generalizing the finite element method: Diffuse approximation and diffuse elements," *Computational Mechanics*, vol. 10, pp. 307–318, 1992.

[25] Onate, E., Idelsohn, S., Zienkiewicz, O. C., and Taylor, R. L., "A Finite Point Method in Computational Mechanics. Applications to Convective Transport and Fluid Flow," *International Journal for Numerical Methods in Engineering*, vol. 39, pp. 3839–3866, 1996.

[26] Perrone, N. and Kao, R., "A general finite difference method for arbitrary meshes," *Computers and Structures*, vol. 5, no. 1, pp. 45–57, 1975.

[27] Quartapelle, L., *Numerical solution of the incompressible Navier-Stokes equations*, vol. 113. Birkhäuser, 2013.

[28] Randles, P. and Libersky, L., "Smoothed particle hydrodynamics: some recent improvements and applications," *Computer methods in applied mechanics and engineering*, vol. 139, no. 1, pp. 375–408, 1996.

[29] Schechter, H. and Bridson, R., "Ghost sph for animating water," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)*, vol. 31, no. 4, 2012.

[30] Shepard, D., "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference*, pp. 517–524, ACM, 1968.

[31] Stewart, S. and others, "Results of FDAs First Interlaboratory Computational Study of a Nozzle with a Sudden Contraction and Conical Diffuser," *Cardiovascular Engineering and Technology*, vol. 4, no. 4, pp. 374–391, 2013.

[32] Trask, N., Maxey, M., Kim, K., Perego, M., Parks, M. L., Yang, K., and Xu, J., "A scalable consistent second-order sph solver for unsteady low reynolds number flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 289, pp. 155–178, 2015.

[33] Tseng, Y.-H. and Ferziger, J. H., "A ghost-cell immersed boundary method for flow in complex geometry," *J. Comput. Phys.*, vol. 192, pp. 593–623, Dec. 2003.

[34] Ye, T., Mittal, R., Udaykumar, H., and Shyy, W., "An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries," *Journal of computational physics*, vol. 156, no. 2, pp. 209–240, 1999.

[35] Zhang, G. and Batra, R., "Modified smoothed particle hydrodynamics method and its application to transient problems," *Computational mechanics*, vol. 34, no. 2, pp. 137–146, 2004.

[36] Zhou, J. G., "Axisymmetric lattice boltzmann method revised," *Phys. Rev. E*, vol. 84, Sep 2011.