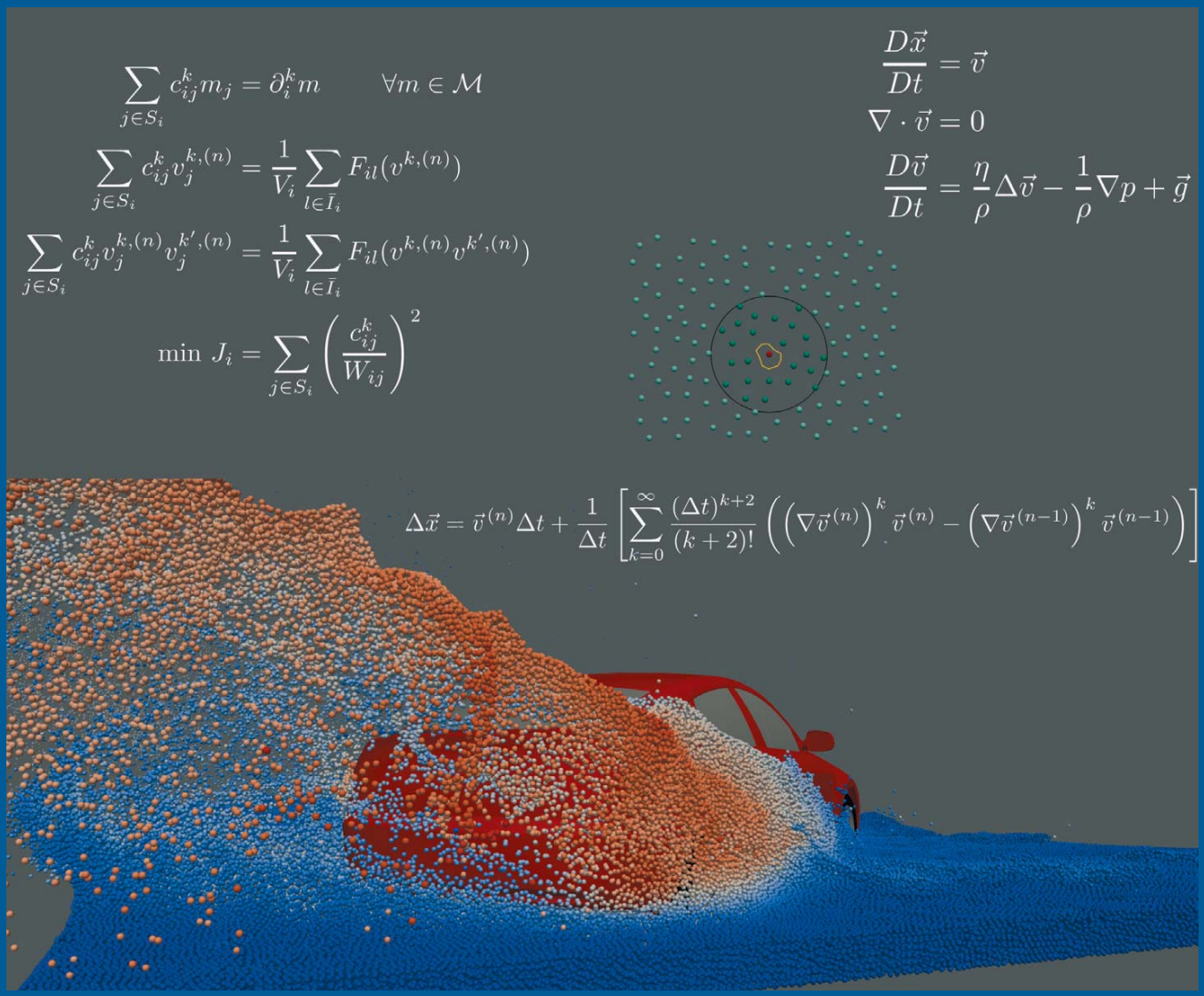
Pratik Suchde

Conservation and Accuracy in Meshfree Generalized Finite Difference Methods



Technische Universität Kaiserslautern

Fraunhofer ITWM

Fraunhofer-Institut für
Techno- und Wirtschaftsmathematik ITWM

Conservation and Accuracy in Meshfree Generalized Finite Difference Methods

Pratik Suchde

FRAUNHOFER VERLAG

Kontakt:
Fraunhofer-Institut für Techno- und Wirtschaftsmathematik ITWM
Fraunhofer-Platz 1
67663 Kaiserslautern
Telefon +49 631/31600-0
Fax +49 631/31600-1099
E-Mail info@itwm.fraunhofer.de
URL www.itwm.fraunhofer.de

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.
ISBN (Print): 978-3-8396-1325-2

D 386

Zugl.: Kaiserslautern, TU, Diss., 2018

Druck: Mediendienstleistungen des
Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart

Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 80 04 69, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 07 11 9 70-25 00
Telefax 07 11 9 70-25 08
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

Conservation and Accuracy in Meshfree Generalized Finite Difference Methods

Pratik Suchde
Department of Mathematics,
University of Kaiserslautern, Germany.

Vom Fachbereich Mathematik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.)
genehmigte Dissertation

Referee 1: Prof. Dr. Axel Klar, University of Kaiserslautern, Germany.
Referee 2: Associate Prof. Dr. Benjamin Seibold, Temple University, USA.

Datum der Disputation: 12.01.2018

D 386

Abstract

EN

The goal of this PhD is to improve various aspects of meshfree Generalized Finite Difference Methods (GFDMs). In this thesis, different meshfree GFDMs are compared, and their potential to solve over-determined problems is presented. A new method is presented that introduces conservation of fluxes in a meshfree setting, which reduces the problem of lack of conservation that has plagued meshfree methods. Special attention is paid on the application of meshfree GFDMs to simulate fluid flow modeled by the incompressible Navier–Stokes equations. A new meshfree GFDM scheme for the same is presented which improves local accuracy, and shows better approximations to the mass conservation condition. Further, different aspects of meshfree Lagrangian frameworks are studied, and new methods to improve accuracy in the Lagrangian movement process are also presented.

DE

Das Ziel dieser Dissertation ist die Verbesserung einiger Aspekte gitterfreier verallgemeinerter Finite Differenzen Methoden (engl. GFDM). In dieser Arbeit werden verschiedene solcher gitterfreier GFDMs verglichen und deren Potenzial zur Lösung überbestimmter Probleme herausgestellt. Es wird eine neue Methode präsentiert, welche die Erhaltung von Flüssen in den gitterfreien Kontext überträgt und somit das für gitterfreie Methoden typische Problem mangelnder Erhaltung reduziert. Besonderes Augenmerk wird auf die Anwendung gitterfreier GFDMs zur Simulation von Fluid Strömungen, welche durch die inkompressiblen Navier–Stokes Gleichungen beschrieben werden können, gelegt. Hierfür wird ein neues gitterfreies GDFM Verfahren vorgestellt, das die lokale Genauigkeit und die Approximation an die Bedingung der Massenerhaltung verbessert. Darüber hinaus werden verschiedene Aspekte des Lagrange-Formalismus im gitterfreien Kontext untersucht und neue Methoden zu Verbesserung der Genauigkeit im Lagrange'schen Bewegungsprozess präsentiert.

Contents

Acknowledgements

The work done for this thesis was made possible due to the support of many people, and I would like to express my sincere gratitude to each of them. Most importantly, Dr. Jörg Kuhnert for his constant help and input throughout this PhD. Dr. Sudarshan Tiwari for the many discussions on meshfree GFDMs. Prof. Axel Klar for being my supervisor. Dr. Simon Shröder for all things geometrical and programming. Dr. Timo Wächtler for being a great office mate and for the help with all the small things. And my fellow PhD students, Tobias Seifarth and Jens Bender, for the many discussions.

My thanks also goes to the entire FPM group at the Fraunhofer ITWM for the continuous support; and to the TV Mensa group.

I am also very grateful to my parents and my brother for supporting my choice to pursue this PhD, and for always believing in me. My deepest gratitude also goes to Sushma, for always being there for me through the ups and downs of my work.

Declaration

I hereby declare that this thesis, titled "Conservation and Accuracy in Meshfree Generalized Finite Difference Methods", and the work presented in it are my own. Except where specific reference is made to the work of others, the contents of this dissertation are original. The contents of this thesis have not been submitted, in whole or in part, for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. Parts of this work have been submitted to and published in peer-reviewed journals [107, 108, 109]. A list of these publications can be found on the next page.

Pratik Suchde
January 2018.

Publications

Parts of the work presented in this thesis have been published in the following peer-reviewed articles

[A] P. Suchde, J. Kuhnert, S. Schröder, A. Klar. A flux conserving meshfree method for conservation laws. *International Journal for Numerical Methods in Engineering.* 112(3) : 238-256. 2017. doi: 10.1002/nme.5511.

[B] P. Suchde, J. Kuhnert, S. Tiwari. On Meshfree GFDM Solvers for the Incompressible Navier–Stokes Equations. *Computer & Fluids.* 165: 1-12. 2018. doi: 10.1016/j.compfluid.2018.01.008.

[C] P. Suchde, J. Kuhnert. Point Cloud Movement For Fully Lagrangian Meshfree Methods. *Journal of Computational and Applied Mathematics.* 340: 89-100. 2018. doi: 10.1016/j.cam.2018.02.020.

Chapter 1

Introduction

Most numerical methods for solving partial differential equations require the generation of a mesh over the computational domain. Despite advances in mesh generation technology and computer hardware, the generation and management of meshes is often the most difficult and time consuming part of the simulation procedure. This is further compounded for complex, time-dependent geometries. In many practical scenarios, the simulation domain changes with time. In such situations, there may arise the need to remesh the domain repeatedly during the simulation. This causes a further bottleneck in numerical simulations. The quality of numerical solutions depends directly on the quality of the mesh. Thus, the efficiency of mesh generation limits the overall accuracy, robustness and speed of the numerical simulation process. Moreover, mesh generation can not always be entirely automated, and often requires a lot of man hours. In fact, in the simulation industry, there exist jobs called 'meshers' whose work is solely the creation of 3D meshes for simulation domains.

To avoid the task of meshing and remeshing, several classes of meshfree or meshless methods have been developed. These gridfree methods use the numerical basis of a set of nodes to cover the computational domain. Nodes need not be regularly distributed, and usually are arbitrarily spaced. These nodes are usually referred to either as points or particles, and are broadly of two types. A node could either be a mass-carrying entity, or simply a location where approximations are performed. For each node, the only geometrical information required is a local set of neighbouring nodes over which approximations are carried out. No additional connectivity information is needed. The generation of the set of nodes across the entire simulation domain, referred to as a point cloud, is much easier than mesh generation. Point cloud generation can be automated to a great extent, thus significantly reducing the necessary man hours to set up a simulation. For simulation domains that move or change with time, meshfree methods have the further advantage of local adaptivity. Modifying a point cloud is much easier than remeshing the entire domain. Due to these advantages, meshfree methods are becoming increasingly popular in industrial simulation applications. One of the biggest fields of application of meshfree methods is fluid flow problems, which often have an open free surface which rapidly changes with time.

Of course, these advantages of meshfree methods come with their own set of challenges. The lack of available mathematical rigour has driven several mathematicians away from meshfree methods. A further issue is the lack of global conservation. In meshed methods, the global mesh ensures that accurate local approximations provide a good global solution. The absence of such a mesh in most meshfree methods means

that there is often no direct relation between local accuracy and global solution quality. This lack of global conservation, and thus accuracy, in the numerical solution has been a big reason for the aversion to meshfree methods in many communities. The aim of this thesis is to address this issue. We try to improve conservation properties, and accuracy, both local and global, of a certain class of meshfree methods. This is done with a strong emphasis on (fully) Lagrangian meshfree methods for fluid flow, where the point cloud moves with the fluid velocity.

1.1 Thesis Outline

This thesis is organized as follows.

- In the remainder of this chapter, we introduce a few different meshfree methods, and talk about point cloud generation and neighbour search algorithms which are essential in all meshfree methods.
- *Chapter 2* goes into details about meshfree Generalized Finite Difference Methods (GFDMs) which are the central topic of this thesis. Their advantages and disadvantages are discussed, along with a comparison with other strong form meshfree methods. Different variations of meshfree GFDMs are presented, with numerical comparisons between them. The unexplored potential of extending meshfree GFDMs to solve over-determined problems is brought to light, and a few numerical examples of the same are also performed.
- *Chapter 3* introduces the topic of conservation in relation with meshfree GFDMs. Existing work to improve conservation is presented, and a new method to generalize the same is also developed. The inherent drawbacks of both these methods are discussed, which poses the need for more work in conservation with meshfree GFDMs.
- *Chapter 4* presents a novel method to introduce approximate conservation in meshfree GFDMs. This method is significantly more efficient than the existing work on the same which was discussed in the previous chapter. An idea to introduce local balances of numerical fluxes in meshfree methods is presented. This flux balance is done within the usual moving least squares framework. Unlike Finite Volume Methods, it is done on locally defined control cells, rather than a globally defined mesh. Applications of this method to an advection-diffusion equation and the Navier–Stokes equations are shown. Numerical simulations are presented to compare this new method to classical GFDMs, and the new method is shown to be superior.
- *Chapter 5* talks about meshfree GFDM schemes to solve fluid flow. Drawbacks specific to meshfree methods are brought to light, and a new scheme is presented which reduces those drawbacks. The new monolithic scheme presented improves accuracy of the mass conservation condition, while avoiding the difficult saddle point structure of linear systems arising in most monolithic schemes. In contrast with the previous chapter which aimed to improve global conservation (and accuracy), the work presented here aims to improve local accuracy of the solution.
- *Chapter 6* deals with the Lagrangian nature of meshfree methods. Different methods to move the point cloud with the fluid velocity are discussed. Inaccu-

racies in existing methods are highlighted, and new methods are presented for the same. A new method to approximate streamlines in a meshfree setting is presented. Point cloud movement along these approximate streamlines is shown to be more accurate than traditional methods for steady flows. A further new method is presented which moves points along pathlines which are computed by determining the change in the approximated streamlines between time steps. Numerical examples show this method to be the most accurate for rapidly changing flow profiles. Unlike the previous chapters, the work presented here applies to all Lagrangian meshfree methods, and not just meshfree GFDMs.
- *Chapter 7* presents an outlook about the work of this thesis, and talks about the potential to extend the different ideas presented.

- *Appendix A* presents some details about the construction of numerical differential operators for meshfree GFDM. This forms an extension of Chapters 2 and 3.
- *Appendix B* gives a short note on the over-determined systems which are referred to in Chapters 2 and 5.
- *Appendix C* shows the construction of local control cells which are used for local balances of fluxes in Chapter 4.

1.2 Different Meshfree Methods

Meshfree methods can be broadly classified into two classes based on the formulation of the partial differential equations being solved.

1.2.1 Weak Form Meshfree Methods

A wide variety of weak-form meshfree methods have been developed. Some important examples of the same are

- Element-Free Galerkin (EFG) [9]
- Meshless Local Petrov-Galerkin (MLPG) [5]
- Reproducing Kernel Particle Methods (RKPM) [70]
- Diffuse Element Method [81]
- Finite Volume Particle Method (FVPM) [38]

Some of these methods, such as the EFG, are based on global weak forms and perform integrations based on a global background mesh. Others, such as the MLPG, are based on local weak forms and integrations are performed only on local background meshes. Some methods, such as the MLPG arise from generalizing finite element and galerkin methods, while others such as the FVPM stem from generalizing finite volume methods.

Since this thesis focuses on strong form meshfree methods, we do not go into details about the advantages and disadvantages of meshfree weak form methods. The interested readers are referred to the book by Liu and Gu [65] and the habilitation of Schweitzer [95] for the same.

1.2.2 Strong Form Meshfree Methods

Several strong-form meshfree methods have been developed over the years. Below, we introduce three major classes of strong form meshfree methods.

Smoothed Particle Hydrodynamics (SPH)

SPH [66] is one of the first and most widely used meshless methods. In SPH, spatial discretizations are based on an integral representation of functions. The integral representation of a function f is motivated by the dirac-delta integral identity, and its value at point $\vec{x}$ is given as

$$f(\vec{x}) \approx \int_{\Omega} f(\vec{x}')W(\vec{x} - \vec{x}', h)\, d\vec{x}' \,, \tag{1.1}$$

where W is the so-called kernel function or smoothing function, and h is the spatial discretization parameter, referred to as the smoothing length. W is chosen such that it approaches the dirac-delta function as $h \to 0$. Derivatives are also approximated by integral operators. For example, if f in Eq. (1.1) is some derivative of a function g, the derivative is transferred to the known function W by performing an integration by parts, similar to that done in many weak-form methods. The final integral is then discretized at the particle locations.

SPH is a Lagrangian meshfree method which was originally developed for astrophysical applications in the late 1970s [34]. It has also been used a lot for fluid flow applications in the past two decades. The origins in astrophysical problems with no solid boundaries meant that SPH does not have a natural way to prescribe boundary conditions. As a result, one of the biggest drawbacks of SPH is the difficulty in enforcing boundary conditions [60, 68, 90]. While a lot of work has been done to address this issue, the SPH formulation still does not naturally include treatment of most boundary conditions and extra effort is needed to enforce them.

Another major drawback of SPH in its classical formulation is the so-called particle inconsistency issue which results in the absence of a valid approximation order. The original SPH does not even have C^0 particle consistency for irregularly distributed particles and boundary particles. Attempts to solve this problem include the kernel renormalization [16] which comes at the price of the loss of certain conservation properties for momentum and energy.

In SPH, the computational domain is discretized by particles which carry mass. Thus, mass conservation is directly guaranteed[1]. However, this means that particles can not be easily added or deleted. This results in clustering of particles and unbalanced or distorted particle distribution, which is related to the so-called tensile instability problem.

[1] While mass conservation is seemingly guaranteed in SPH, volume conservation is not. This is discussed in detail in Chapter 6. For incompressible flows, a constant density means that an error in volume conservation corresponds directly to an error in mass conservation.

Radial Basis Functions (RBF)

Radial functions are functions whose values only depend on the distance from a fixed point. Thus, a radial function ϕ centered at $\vec{c}$ satisfies

$$\phi(\vec{x}, \vec{c}) = \phi(\|\vec{x} - \vec{c}\|)\,. \tag{1.2}$$

Meshless methods based on RBFs stem from the use of radial functions in scattered data interpolation. In these methods, a function is approximated by a linear combination of a single radial function translated at different points

$$f(\vec{x}) = \sum_{i=1}^{N} c_i \phi(\|\vec{x} - \vec{x}_i\|)\,, \tag{1.3}$$

where the coefficients c_i are found by inverting a global matrix. Derivatives are defined based on Eq. (1.3) by considering the derivatives of the radial functions. For example,

$$\nabla f(\vec{x}) = \sum_{i=1}^{N} c_i \nabla \phi(\|\vec{x} - \vec{x}_i\|)\,. \tag{1.4}$$

To avoid the poor conditioning of the global systems and the high computational costs involved in inverting them, many meshless methods based on RBFs have started using local approximations [94], making RBFs resemble other meshless methods. One way of doing this is by replacing the summation in Eq. (1.3) by a local one depending only on the nodes in the support domain of the central node.

A drawback of meshfree RBF-based methods is that the choice of the basis function ϕ and the related shape parameters significantly affect the accuracy and stability of the method [79]. While a lot of has been done to determine the optimal choice for the same (for example, [28]), this choice remains application-specific and ad-hoc in nature. Depending on the choice of the parameter, extra stabilization techniques may be required, which are usually extremely expensive [30].

Meshfree Generalized Finite Difference Methods (GFDM)

As the name suggests, meshfree GFDMs generalize traditional finite difference methods to arbitrary point distributions. They are based on weighted least squares approximations. Derivatives are given by linear combinations of the function values on neighbouring points. Meshfree GFDMs are the central theme of this thesis, and a detailed introduction to them is given in Chapter 2.

1.3 Point Cloud Generation

An important aspect of meshfree methods is the generation of the initial set of nodes. A common misconception is that point cloud generation can be as tough as mesh generation. This stems from the fact that many meshfree communities initially used the nodes of a mesh as the point cloud. Some meshfree methods have also used randomly scattered nodal distributions, which further promotes the misconception.

A commonly used easy way to set up the initial point cloud is to start with adding points on the domain boundary with a certain spacing. These boundary points are used as a source to generate a first layer of interior points, which in turn act as a source for the next layer of interior points. This procedure is continued till the entire domain is filled. This process can be easily automated, and can even be used to generate point clouds on extremely complex geometries. A more detailed description of this procedure can be found in Drumm *et al.* [24]. Similar setup procedures have been commonly referred to as the advancing front technique [71].

1.4 Neighbour Search

Efficient neighbour searching is integral to the efficiency of meshfree methods. For each node, its neighbouring nodes are usually determined as the nodes within a certain distance h from it. The naive approach to neighbour searching would thus involve computing distances between every pair of nodes in the computational domain. However, this procedure is exorbitantly expensive. To avoid excessive distance computations, the domain is usually split into multiple regions referred to as boxes or cells. Using such a decomposition, for each node, distances only need to be computed with other nodes within the same box (or possibly also adjacent boxes). Several different data structures have been used to this end. One of the ways to do the same is to use quadtree or octree type searching algorithms. Efficient methods for neighbour searching have been studied in, for example, [22, 85].

Chapter 2

Meshfree Generalized Finite Difference Methods

Moving Least Squares (MLS) approximations used in data fitting have widely been used as a basis for many meshfree methods [8]. In MLS approximations, a function u, given only on a set of points, is approximated by a smooth function

$$v(\vec{x}) = \sum_{k=1}^{m} p_k(\vec{x}) a_k(\vec{x}) \,, \tag{2.1}$$

where $p_k(\vec{x})$ are basis functions, usually monomial or singular functions, $a_k(\vec{x})$ are their coefficients, and m is the number of basis functions. The coefficients $a_k(\vec{x})$ are obtained such that $v(\vec{x})$ provides the best approximation to u (and possibly its derivatives), in the least squares sense. This results in the minimization of the quadratic form

$$J = \sum_{\vec{x} \in S} W(\vec{x}) (v(\vec{x}) - u(\vec{x}))^2 \,, \tag{2.2}$$

where S is the set of points about which the approximation is being carried out, and $W(\vec{x})$ is a weighting function. The derivatives of u are approximated by the derivatives of v. Similar methods have also been reffered to as Weighted Least Squares (WLS or WLSQ) methods. Such least squares procedures have been used in a wide variety of meshfree methods including those based on both weak forms and strong forms. A comparison between different least squares procedures commonly used in meshfree methods has been done by several authors. For example, Oñate *et al.* [83] present an overview of several point data-interpolation based procedures used in meshfree methods, while Seibold [96] provides a more formal classification between the "local" and "moving" varieties, and the "interpolating" and "approximating" varieties of least squares approaches. Most of these details are skipped in this thesis, and we focus on the use of such procedures in meshfree GFDMs.

Meshfree GFDMs are one such class of meshfree methods based on MLS or WLS procedures. They have been widely used (for example, [32, 48, 89, 105, 121]) and are referred under various names, including FPM, which stands for both the Finite Pointset Method [111] and the Finite Point Method [83], the Kinetic Meshless Method (KMM) [89], and the Least Squares Kinetic Upwind Method (LSKUM) [33].

There are two variations of meshfree GFDMs that we consider in this thesis, both of which are based on MLS approximations. The first is the widely used approach based on the work of Liszka and Orkisz [63], which we refer to as the classical GFDM.

This method very closely resembles traditional finite differences, and is carried out by minimizing errors obtained from Taylor expansions. Most meshfree GFDMs fall under this class. The second approach is a modification of the classical GFDM based on the work of Tiwari and Kuhnert [111] in which the error in the PDE considered is minimized simultaneously with the errors obtained from Taylor expansions. We refer to this method as the direct GFDM. In both variations, we consider Taylor expansions up to second order terms. Higher order accuracy can be attained in both formulations and have been considered, for example, by Milewski [76].

2.1 Finite Pointset Method

The Finite Pointset Method (FPM) [47, 57, 110, 112] is one type of meshfree GFDM, and is used as the basic meshfree framework in this thesis. FPM is a fully Lagrangian meshfree method that evolved out of SPH while addressing the issues of boundary conditions, particle inconsistencies, and tensile instabilities [55]. Both meshfree GFDMs formulations considered in this thesis have been referred to under the name of FPM.

The FPM has been shown to be a robust method with many practical applications [24, 47, 110, 116]. The FPM is also used as the numerical basis of two commercially used meshfree simulation tools: NOGRID [77] and the meshfree module of VPS-PAMCRASH [113]. Further, it is also the numerical basis of the upcoming simulation software MESHFREE[1].

It must be noted that in the meshfree context, the acronym FPM is often a confusing one. It is used to represent not only the aforementioned Finite Pointset Method, but also the Finite Particle Method [69] and the Finite Point Method [83]. The confusion is compounded by the fact that each of these three methods are well established in different communities and have been around for well over a decade. Further, in the early days of meshfree methods, FPM was also used to denote the Free Points Method [25]. Thus, we henceforth drop the acronym FPM and refer to the Finite Pointset Method under the umbrella term of GFDM.

2.2 Comparison with Other Meshfree Methods

- In many meshfree particle methods like SPH, mass-carrying particles are used to discretize the computational domain. In contrast, meshfree GFDMs use numerical points which are simply locations where approximations are carried out. These numerical points do not have a mass[2], and thus points can easily be added or removed during a simulation [47]. This is especially relevant in Lagrangian frameworks where points move with a fluid velocity. This movement could cause clustering of points in one region, or the development of 'holes' with insufficient number of points. Points can easily be deleted in the first situation, and added in the second, to prevent instabilities from developing. Such addition and deletion is not as easy in the mass-particle based methods like SPH which suffer from the so-called tensile instability problem and distorted point clouds. Thus,

[1] `https://www.meshfree.eu/`

meshfree GFDMs are more adaptive (in terms of the spatial discretization) than many other meshfree methods. The fact that point clouds can easily be modified locally is also a major advantage of meshfree GFDMs over mesh-based methods, where remeshing would often require a global re-computation. Details about this addition and deletion of points in meshfree GFDMs are given in Section 2.4.

- Since meshfree GFDMs are based on Taylor expansions (or polynomials, as explained later in this chapter), an order of accuracy at the discrete level can be easily prescribed, which is not the case for many meshfree methods such as SPH.
- Meshfree GFDMs provide a framework to naturally incorporate boundary conditions, without any extra effort, which has been a major issue in many particle-based meshfree methods such as SPH. Details about enforcing boundary conditions in meshfree GFDMs are given in Sections 2.5.4 and 2.6.2.
- In RBF generated finite differences (RBF-FD), the desired function derivatives depend on the derivative of the radial function being used as a basis. The choice of the basis function (and the related shape parameters) play a big role in overall accuracy and stability of the method [28, 31, 79]. Similarly, derivatives in SPH also depend on the derivatives of the chosen kernel function, and thus the choice of the kernel function once again plays a major role. In contrast to these other strong form meshfree methods, derivatives in meshfree GFDMs are taken as linear combinations of function values in a support domain, and do not directly depend on the derivatives of any kernel function. The weighting function only plays a role in norm minimization. It does not affect the approximation order and it can be argued that its shape does not significantly affect accuracy and stability of the method[3].

2.3 Notation

For meshfree GFDMs, the computational domain Ω, with boundary $\partial\Omega$, is discretized using a cloud of N numerical points with positions $\vec{x}_i$, $i = 1, \ldots, N$. This includes points both in the interior and on the boundary of the domain. The points are usually irregularly spaced. Each numerical point carries the necessary numerical data of the problem. Each point i has a set of neighbouring points S_i which contains $n = n(i)$ points, including itself. The neighbourhood or support S_i is determined by spatial proximity

$$S_i = \{\vec{x}_j : \|\vec{x}_j - \vec{x}_i\| \leq \beta h\}, \tag{2.3}$$

where $h = h(\vec{x}, t)$ is the radius of the support, referred to as smoothing length or interaction radius; and $\beta \leq 1$ is a positive constant. The spatial distribution of

[2]Points do not have a mass prescribed *directly*. However, in many applications, each point would have a prescribed density. Each point is associated with a volume for post-processing reasons, and thus, the density and volume combination indirectly specify a mass.

[3]The weight function has to provide a reasonable weight to the neighbouring points. It must be noted that a weight function that fails to do so (for example, a very narrow Gaussian that will effectively reduce the size of the neighbourhood) could lead to poor conditioning of the local least squares matrix.

points is described by three parameters: h, r_{min} and r_{max}. It is ensured that no two points are closer than $r_{min}h$ and there exists at least one point in every possible sphere of radius $r_{max}h$ in the computational domain. Thus, the smoothing length h also determines the spatial discretization size. r_{min} and r_{max} usually have values of approximately 0.2 and 0.45 respectively. β affects the size of neighbourhoods, and a discussion on its value is done in Chapter 4. For all the simulation results presented in this thesis, each fully interior point[4] has a neighbourhood that contains about $45 - 50$ points in 3D, and $15 - 20$ points in 2D. These numbers are slightly lesser at and near the domain boundaries. We note that many meshfree methods use smaller neighbourhoods. However, these often impose strict conditions on the regularity of the point distributions to maintain stability of the simulation.

For a central point i, the subscript ij is used to denote a quantity defined with relation to neighbouring point j. For example, the weight W of point $j \in S_i$ while performing the approximations at point i is denoted by W_{ij}. The subscript ij can have two different interpretations, depending on context, which are used interchangeably. The value of j in the subscript ij could be the point number based on a global indexing of all points $j \in S_i \subset (1, 2, \ldots, N)$. Alternatively, it could also be based on a local indexing of all points in the support domain $j \in (1, 2, \ldots, n(i))$. Further, the same interpretations also hold for function values. u_j can be used to denote the value of the function u at point j, $j \in [1, \ldots, N]$ (globally indexed), while $u_1, u_2, \ldots, u_n$ can also be used to denote the function values in the neighbourhood of some central point i.

2.4 Adding and Removing Points

As noted earlier, the ease of adding and deleting points locally during the simulation of a fluid in the Lagrangian framework poses an important advantage of meshfree GFDMs over other meshfree methods.

Points are added in the point cloud to ensure that every possible sphere of radius $r_{max}h$ has at least one point in it. The first aspect of this is the identification of 'holes' or spheres of radius $r_{max}h$ which contain no points. A possible way to do the same locally is to consider the local Delaunay tessellation of points in each support domain. Any triangle (in 2D) or tetrahedron (in 3D) with circumradius larger than $r_{max}h$ represents a hole, at the center of which a point needs to be added. After the addition of the new point, all field properties carried by each point need to be approximated at this new location. A possible way to do the same is to use the smoothing operators defined in the coming sections.

Points are deleted in the point cloud to ensure that no two points are closer than $r_{min}h$. These points can be identified by a distance calculation in each neighbourhood. Rather than deleting one of the two points, an alternative is to merge the two at a central location. This would then require interpolation of all field properties at this new location. This could be done either as done in the point addition case explained above, or it could be based solely on the two points being merged.

Further details about the addition and deletion/merging of points in meshfree GFDMs can be found in, for example, Drumm *et al.* [24].

[4] A fully interior point here refers to an interior point with no boundary neighbours.

"Correct" point cloud management in these aspects is essential for the stability of a wide range of problems. Considerations to this end in meshfree GFDMs have been done by Seibold [96], among other authors. Similar work has also been done in the context of other meshfree methods by Iske [44], among other authors. However, despite these contributions, existing literature on meshfree methods does not fully address accuracy issues with respect to point cloud management, and the required interpolation therein. These considerations of accuracy in Lagrangian meshfree GFDMs are not looked into in this thesis.

2.5 Classic GFDM Differential Operators

As the name suggests, in meshfree GFDMs, numerical derivatives are computed with a generalized finite difference approach. For a function u defined at each numerical point $i = 1, 2, \ldots, N$, its derivatives are approximated as

$$\partial^* u(\vec{x}_i) \approx \tilde{\partial}^*_i u = \sum_{j \in S_i} c^*_{ij} u_j \,, \tag{2.4}$$

where $^* = x, y, xx, \Delta, etc.$ represents the differential operator being approximated, ∂^* represents the continuous *-derivative, and $\tilde{\partial}^*_i$ represents the discrete derivative at point i. For each point i, the stencil coefficients c^*_{ij} are found using a weighted least squares approach. Two equivalent formulations can be used for the same and are explained below. In both formulations, and in the direct GFDM operators presented later, the operators can be found locally at each point by solving a small linear system, independently of the operators at the rest of the point cloud.

2.5.1 Taylor Expansions

For a point i, consider Taylor expansions around it at each neighbouring point $j \in S_i$

$$e_{ij} + u(\vec{x}_j) = u(\vec{x}_i) + \nabla u \cdot (\vec{x}_j - \vec{x}_i) + \frac{1}{2}(\vec{x}_j - \vec{x}_i)^T D (\vec{x}_j - \vec{x}_i) \,. \tag{2.5}$$

The unknown coefficients of ∇u and D are computed by a weighted least squares method, by minimizing

$$\min \, J_i = \sum_{j \in S_i} W^2_{ij} e^2_{ij} \,, \tag{2.6}$$

where W is a weighting function used to make sure that the points closer to the central point i have a larger impact than the points farther away. We note that the square of the weighting function is used only for notational convenience, the benefit of which will be made evident later. The weighting function is usually taken as a Gaussian distribution

$$W_{ij} = \exp\left(-\alpha_W \frac{\|\vec{x}_j - \vec{x}_i\|^2}{h_i^2 + h_j^2} \right) \,, \tag{2.7}$$

where α_W is a positive constant usually taken in the range of $(2, 8)$. Note that the weighting function is only defined on the local support S_i consisting of $n(i)$ points.

For the sake of brevity, we present only the case of one spatial dimension. Eq. (2.5) leads to the following system which is solved at each point $i = 1, \ldots, N$

$$\underbrace{\begin{pmatrix} e_{i1} \\ \vdots \\ e_{in} \end{pmatrix}}_{\vec{E}_i} = \underbrace{\begin{pmatrix} \delta x_{i1} & \frac{1}{2}\delta x_{i1}^2 \\ \vdots & \vdots \\ \delta x_{in} & \frac{1}{2}\delta x_{in}^2 \end{pmatrix}}_{M_i} \underbrace{\begin{pmatrix} (u_x)_i \\ (u_{xx})_i \end{pmatrix}}_{\vec{a}_i} - \underbrace{\begin{pmatrix} u_1 - u_i \\ \vdots \\ u_n - u_i \end{pmatrix}}_{\vec{b}_i} . \tag{2.8}$$

where $\delta x_{ij} = x_j - x_i$. Or, in short form $\vec{E}_i = M_i\vec{a}_i - \vec{b}_i$. The minimization Eq. (2.6) can be rewritten as

$$\min J_i = \vec{E}_i^T W_i^2 \vec{E}_i \,, \tag{2.9}$$

$$= (M_i\vec{a}_i - \vec{b}_i)^T W_i^2 (M_i\vec{a}_i - \vec{b}_i) \,, \tag{2.10}$$

where W_i is a diagonal matrix with entries $W_{i1}, \ldots, W_{in}$. A formal minimization leads to

$$\vec{a}_i = [(M_i^T W_i^2 M_i)^{-1} M_i^T W_i^2]\vec{b}_i \,. \tag{2.11}$$

Appendix A.3 shows a proof of the same. This leads to the differential operator stencils

$$(u_x)_i = \sum_{j \in S_i} c_{ij}^x (u_j - u_i) \,, \tag{2.12}$$

$$(u_{xx})_i = \sum_{j \in S_i} c_{ij}^{xx} (u_j - u_i) \,, \tag{2.13}$$

where c_{ij}^x and c_{ij}^{xx} represent the values in the first and second row respectively of the matrix $[(M_i^T W_i^2 M_i)^{-1} M_i^T W_i^2]$ in Eq. (2.11). A more efficient method to compute the stencil coefficients is shown in Appendix A.2. These stencil coefficients are then used to obtain the spatial discretization of the PDE being solved. For example, if we consider the PDE

$$au + bu_x + cu_{xx} = d \,. \tag{2.14}$$

The derivative approximations Eq. (2.12), Eq. (2.13) are substituted into the PDE to obtain

$$au_i + b \sum_{j \in S_i} c_{ij}^x (u_j - u_i) + c \sum_{j \in S_i} c_{ij}^{xx} (u_j - u_i) = d, \qquad i = 1, \ldots, N \,, \tag{2.15}$$

which forms a large sparse implicit system which is solved with an iterative method.

Alternatively, Eq. (2.8) can be rewritten as

$$\begin{pmatrix} e_{i1} \\ \vdots \\ e_{in} \end{pmatrix} = \begin{pmatrix} 1 & \delta x_{i1} & \frac{1}{2}\delta x_{i1}^2 \\ \vdots & \vdots & \vdots \\ 1 & \delta x_{in} & \frac{1}{2}\delta x_{in}^2 \end{pmatrix} \begin{pmatrix} (u_0)_i \\ (u_x)_i \\ (u_{xx})_i \end{pmatrix} - \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} . \tag{2.16}$$

Carrying out a minimisation procedure as above leads to the derivative stencils

$$(u_x)_i = \sum_{j \in S_i} c_{ij}^x u_j \,, \tag{2.17}$$

$$(u_{xx})_i = \sum_{j \in S_i} c_{ij}^{xx} u_j \,. \tag{2.18}$$

In addition, the minimization also leads to stencils for function smoothing

$$(u_0)_i = \sum_{j \in S_i} c^0_{ij} u_j \,. \tag{2.19}$$

For reasons that will be made clear later, we refer to such function smoothing stencils as function approximation stencils. Further $(u_0)_i$ is referred to as u_i in shorthand. The formulation in Eq. (2.16) and the stencils that follow are the ones we will use throughout this thesis.

2.5.2 Polynomial Method

An alternate, but equivalent, way to arrive at the stencil coefficients in Eq. (2.17) – Eq. (2.19) is to ensure that the derivatives of monomials $m \in \mathcal{M}$, up to the order of accuracy desired (usually 2), are exactly reproduced.

$$\sum_{j \in S_i} c^*_{ij} m_j = \partial^*_i m \,, \qquad \forall m \in \mathcal{M} \,, \tag{2.20}$$

$$\min \, J_i = \sum_{j \in S_i} \left(\frac{c^*_{ij}}{W_{ij}} \right)^2 \,. \tag{2.21}$$

where $^* = x, y, xx, \Delta, etc.$. An efficient method to compute the differential operators stencils in this formulation is presented in Appendix A.2.

2.5.3 Equivalence of the Two Formulations

To illustrate the equivalence of the two formulations mentioned above, we start with the Taylor expansions around point i at the location of point $j \in S_i$. In one spatial dimension,

$$e_{ij} + u_j = u_i + \delta x_{ij} (u_x)_i + \frac{1}{2} \delta x^2_{ij} (u_{xx})_i \,. \tag{2.22}$$

Multiplying Eq. (2.22) with c^*_{ij}, and summing over all $j \in S_i$, we get

$$\sum_{j \in S_i} c^*_{ij} u_j \approx \sum_{j \in S_i} c^*_{ij} u_i + \sum_{j \in S_i} c^*_{ij} \delta x_{ij} (u_x)_i + \sum_{j \in S_i} c^*_{ij} \frac{1}{2} \delta x^2_{ij} (u_{xx})_i \,, \tag{2.23}$$

$$= \left(\sum_{j \in S_i} c^*_{ij} \right) u_i + \left(\sum_{j \in S_i} c^*_{ij} \delta x_{ij} \right) (u_x)_i + \left(\sum_{j \in S_i} c^*_{ij} \frac{1}{2} \delta x^2_{ij} \right) (u_{xx})_i \,. \tag{2.24}$$

Note that the error term of $\mathcal{O}(h^3)$ has been dropped for convenience. Thus, for the first derivative, $* = x$, comparing the definition Eq. (2.4) with Eq. (2.24) leads to

$$\sum_{j \in S_i} c^x_{ij} = 0 \,, \tag{2.25}$$

$$\sum_{j \in S_i} c^x_{ij} \delta x_{ij} = 1 \,, \tag{2.26}$$

$$\sum_{j \in S_i} c^x_{ij} \delta x^2_{ij} = 0 \,. \tag{2.27}$$

Similarly for the second derivative, we get

$$\sum_{j\in S_i} c_{ij}^{xx} = 0\,, \tag{2.28}$$

$$\sum_{j\in S_i} c_{ij}^{xx}\delta x_{ij} = 0\,, \tag{2.29}$$

$$\sum_{j\in S_i} c_{ij}^{xx}\delta x_{ij}^2 = 2\,. \tag{2.30}$$

And for the function approximation stencil, we get

$$\sum_{j\in S_i} c_{ij}^{0} = 1\,, \tag{2.31}$$

$$\sum_{j\in S_i} c_{ij}^{0}\delta x_{ij} = 0\,, \tag{2.32}$$

$$\sum_{j\in S_i} c_{ij}^{0}\delta x_{ij}^2 = 0\,. \tag{2.33}$$

The three above systems are the same as Eq. (2.20) and thus the stencils in the two formulations satisfy the same consistency conditions. Further, if the same weighting function is used in Eq. (2.6) and Eq. (2.21), the weighted minimizations also lead to the same stencils in both formulations, upto numerical round off errors. A proof of this is shown in Appendix A.3.

In terms of modifying the classical meshfree GFDM to different ends, both formulations mentioned above come with their own advantages and disadvantages. This forms a major part of the present thesis. An existing extension of the Taylor expansion formulation is presented in Section 2.6. Those ideas are then extended to solve a wider range of problems in Section 2.6.1, and then later used to design improved fluid flow solvers in Chapter 5. Extensions to the polynomial formulation are discussed in Chapter 4 with the aim of introducing an approximate notion of conservation in meshfree GFDMs, and in Section 2.5.5.

2.5.4 Boundary Conditions

The ease of enforcement of boundary conditions is one of the biggest advantages of meshfree GFDMs over other meshfree methods. Consider a 2D PDE in 1 variable

$$au + bu_x + cu_y + du_{xx} + eu_{yy} + fu_{xy} = g\,. \tag{2.34}$$

An implicit discretization of this PDE, as described earlier, would lead to the following sparse linear system

$$au_i + b\sum_{j\in S_i} c_{ij}^{x}u_j + c\sum_{j\in S_i} c_{ij}^{y}u_j + d\sum_{j\in S_i} c_{ij}^{xx}u_j + e\sum_{j\in S_i} c_{ij}^{yy}u_j + f\sum_{j\in S_i} c_{ij}^{xy}u_j = g\,, \quad i = 1, 2, \ldots, N\,. \tag{2.35}$$

Similar to traditional finite differences, boundary conditions in the classical meshfree GFDMs are enforced by simply replacing the relevant rows in Eq. (2.35) by the discretized boundary condition. For Dirichlet boundary conditions $u = g_D$ at a boundary

point i, the discretized boundary condition is given by simply adding the following row in the sparse system

$$u_i = g_D \,. \tag{2.36}$$

For a Neumann boundary condition $\vec{n} \cdot \nabla u = g_N$, the following row is added

$$\sum_{j \in S_i} c_{ij}^n u_j = g_N \,, \tag{2.37}$$

where c_{ij}^n can be defined in one of two ways. The first way is to get the stencil for the derivative in the normal direction by using the stencils for the derivatives (in $2D$) in the x and y directions: $c_{ij}^n = n_i^x c_{ij}^x + n_i^y c_{ij}^y$, where $\vec{n}_i = (n_i^x, n_i^y)$. Alternatively, it is sometimes desirable to compute derivative stencils in the normal direction directly by a minimization procedure similar to that explained earlier. In the polynomial formulation, this would require the following minimization

$$\sum_{j \in S_i} c_{ij}^n m_j = \vec{n} \cdot \nabla m \,, \qquad \forall m \in \mathcal{M} \,, \tag{2.38}$$

$$\min \, J_i = \sum_{j \in S_i} \left(\frac{c_{ij}^n}{W_{ij}} \right)^2 \,. \tag{2.39}$$

The advantage of this is that a diagonal dominance procedure, as explained in the next section, can be carried out to improve the convergence of iterative solvers for the sparse linear system. If the stencils for the derivatives in the normal direction are computed using this method, the regular gradient stencils need to be modified to ensure the consistency $c_{ij}^n = n_i^x c_{ij}^x + n_i^y c_{ij}^y$. The procedure for the same is shown in Appendix A.5. For all numerical examples in this thesis, we use this method for each Neumann boundary condition in the classical GFDM framework.

An important boundary condition in fluid flow applications is the one imposed on the free surface. We do this by equating the stresses at the free surface in the normal and tangential direction(s). In 2D, this is given by

$$\vec{t}^{\,T} \cdot \mathbf{S} \cdot \vec{n} = 0 \,, \tag{2.40}$$

$$\vec{n}^{\,T} \cdot \mathbf{S} \cdot \vec{n} = p - p_0 - \sigma\kappa \,, \tag{2.41}$$

where $\vec{t}$ is a unit vector in the tangential direction to the free surface, p is the pressure, p_0 is the atmospheric pressure, $\mathbf{S} = \mathbf{S}(\vec{v})$ is the stress tensor for velocity $\vec{v}$, σ is the surface tension, and κ is the local free surface curvature which is determined geometrically. This can be simplified to obtain

$$(2n^x t^x)\frac{\partial u}{\partial x} + (n^x t^y + n^y t^x)\frac{\partial u}{\partial y} + (n^x t^y + n^y t^x)\frac{\partial v}{\partial x} + (2n^y t^y)\frac{\partial v}{\partial y} = 0 \,, \tag{2.42}$$

$$(2(n^x)^2)\frac{\partial u}{\partial x} + (2n^x n^y)\frac{\partial u}{\partial y} + (2n^x n^y)\frac{\partial v}{\partial x} + (2(n^y)^2)\frac{\partial v}{\partial y} = p - p_0 - \sigma\kappa \,. \tag{2.43}$$

This can be discretized as done in Eq. (2.35). A similar procedure is done for the stress evaluation in slip boundary conditions, and for the 3D case.

Note that the use of ghost nodes outside the domain boundary or any similar procedure is *not* needed in the enforcement of boundary conditions.

2.5.5 Diagonal Dominance

It is often desirable to have the central stencil value c^*_{ii} to be much larger in absolute value than the others $c^*_{ij}, i \neq j \in S_i$. For the Laplace operator, this significantly improves stability for Poisson problems and heat equations. More specifically, positivity is desired in the Laplace stencil: the central stencil value should be negative, and the neighbouring values should be positive. Seibold [97] provides a detailed account of the need of positivity and the resultant M-character of the matrices in linear systems. He goes on to show how to achieve the same by selecting neighbouring points by a minimization approach, rather than the circular proximity-based neighbourhoods used here. In this section, we present a method to improve stability for Poisson problems while maintaining circular neighbourhoods.

An advantage of writing the differential operators in the polynomial formulation is that it can easily be extended to add a control over the central stencil value. In addition to the monomial test functions, a Kronecker-delta function is added to the consistency conditions. The Laplace operator can then be determined as

$$\sum_{j \in S_i} c^\Delta_{ij} m_j = \partial^\Delta_i m \,, \qquad \forall m \in \mathcal{M} \,, \tag{2.44}$$

$$c^\Delta_{ii} = A_c \,, \tag{2.45}$$

$$\min\, J_i = \sum_{j \in S_i} \left(\frac{c^\Delta_{ij}}{W_{ij}} \right)^2 \,, \tag{2.46}$$

where $A_c < 0$ is the desired central stencil value. The optimal choice of A_c would be situation specific, and thus explicitly setting the central stencil value is not a desirable way to achieve positivity or diagonal dominance. However, this can be used to get a more general method to achieve the same. The key point to note is that the entire stencil changes as A_c changes. Using this, one way to achieve most possible diagonal dominance (for a given neighbourhood configuration) is to minimize the functional

$$g^\Delta = \frac{\sum_{j \in S_i} \left(c^\Delta_{ij} \right)^2}{\left(c^\Delta_{ii} \right)^2} = \frac{\langle \vec{c}^\Delta_i, \vec{c}^\Delta_i \rangle}{\left(c^\Delta_{ii} \right)^2} \,, \tag{2.47}$$

where $\vec{c}^\Delta_i = (c^\Delta_{i1}, \cdots, c^\Delta_{in})$ is the vector formed by the stencil coefficients at each neighbouring point of i, and $\langle \cdot, \cdot \rangle$ represents the scalar product. Since the minimization is done with respect to a varying central stencil value, we wish to set

$$\frac{\partial g^\Delta}{\partial c^\Delta_{ii}} = 0 \,. \tag{2.48}$$

Analytically, it can be seen that

$$\frac{\partial g^\Delta}{\partial c^\Delta_{ii}} = \frac{2}{\left(c^\Delta_{ii} \right)^2} \langle \vec{c}^\Delta_i, \frac{\partial \vec{c}^\Delta_i}{\partial c^\Delta_{ii}} \rangle - \frac{2}{\left(c^\Delta_{ii} \right)^3} \langle \vec{c}^\Delta_i, \vec{c}^\Delta_i \rangle \,. \tag{2.49}$$

Thus, Eq. (2.48) leads to

$$\langle \vec{c}^\Delta_i, \vec{d}^\Delta_i \rangle c^\Delta_{ii} - \langle \vec{c}^\Delta_i, \vec{c}^\Delta_i \rangle = 0 \,, \tag{2.50}$$

where

$$\vec{d}_i^{\,\Delta} = \frac{\partial \vec{c}_i^{\,\Delta}}{\partial c_{ii}^{\Delta}} \,. \tag{2.51}$$

It can be proven that $\vec{d}_i^{\,\Delta}$ as defined above is the same as that defined by the following minimization

$$\sum_{j \in S_i} d_{ij}^{\Delta} m_j = 0 \,, \qquad \forall m \in \mathcal{M} \,, \tag{2.52}$$

$$d_{ii}^{\Delta} = 1 \,, \tag{2.53}$$

$$\min \, J_i = \sum_{j \in S_i} \left(\frac{d_{ij}^{\Delta}}{W_{ij}} \right)^2 \,. \tag{2.54}$$

The proof of this relies on the formulation of derivative stencils computed by the QR-decomposition as shown in Appendix A and A.2. The actual proof is given in Appendix A.4.

Now, using $\vec{d}_i^{\,\Delta}$, the Laplacian stencil is split into two parts, one of which is used to satisfy the monomial consistency conditions, and the other is used to change the central stencil values.

$$\vec{c}_i^{\,\Delta} = \vec{\sigma}_i^{\,\Delta} + \alpha^{\Delta} \vec{d}_i^{\,\Delta} \tag{2.55}$$

where $\alpha^{\Delta} \in \mathbb{R}$ is determined by the minimization of the functional g^{Δ}, as in Eq. (2.50). $\vec{\sigma}_i^{\,\Delta}$ is an approximate guess for the Laplacian stencil. It satisfies the monomial consistency conditions, while $\vec{d}_i^{\,\Delta}$ as given in Eq. (2.52) – Eq. (2.54) lies in their null space. $\vec{\sigma}_i^{\,\Delta}$ is given by

$$\sum_{j \in S_i} \sigma_{ij}^{\Delta} m_j = \partial_i^{\Delta} m \,, \qquad \forall m \in \mathcal{M} \,, \tag{2.56}$$

$$\sigma_{ii}^{\Delta} = \tilde{A}_c \,, \tag{2.57}$$

$$\min \, J_i = \sum_{j \in S_i} \left(\frac{\sigma_{ij}^{\Delta}}{W_{ij}} \right)^2 \,. \tag{2.58}$$

where $\tilde{A}_c \notin \{0, 1\}$ is some fixed central stencil value for $\vec{\sigma}_i^{\,\Delta}$. Clearly, the Laplace stencil defined by Eq. (2.55) satisfies the monomial consistency conditions for every value of α^{Δ}.

Now, plugging Eq. (2.55) into Eq. (2.50) leads to

$$\alpha^{\Delta} = \frac{\langle \vec{\sigma}_i^{\,\Delta}, \vec{d}_i^{\,\Delta} \rangle \sigma_{ii}^{\Delta} - \langle \vec{\sigma}_i^{\,\Delta}, \vec{\sigma}_i^{\,\Delta} \rangle}{\langle \vec{\sigma}_i^{\,\Delta}, \vec{d}_i^{\,\Delta} \rangle - \langle \vec{d}_i^{\,\Delta}, \vec{d}_i^{\,\Delta} \rangle \sigma_{ii}^{\Delta}} \,. \tag{2.59}$$

$\vec{\sigma}_i^{\,\Delta}$ and $\vec{d}_i^{\,\Delta}$ can be computed as explained above, and thus Eq. (2.59) gives the Laplacian stencil according to Eq. (2.55).

We note that $\vec{\sigma}_i^{\,\Delta}$ and $\vec{d}_i^{\,\Delta}$ can be computed by one minimization with different right hand side vectors as shown in Appendix A.2, and thus, performing this procedure does not increase the computation time significantly.

We further note that this procedure does *not* guarantee diagonal dominance of the matrices in the resultant linear systems or positivity of the Laplace stencil. In fact,

positivity is not always possible on circular proximity-based neighbourhoods. However, empirically this method has shown to significantly improve stability in Poisson problems and heat equations. The usefulness of the outlined procedure to obtain most possible diagonal dominance is illustrated with a simple diffusion example. Consider the heat equation on the domain $[0,1] \times [0,1]$

$$\frac{\partial \phi}{\partial t} = \Delta \phi \,. \tag{2.60}$$

Initial conditions are taken as

$$\phi(\vec{x}, 0) = 5 \sin(\pi x) \sin(\pi y) \,. \tag{2.61}$$

Dirichlet 0 boundary conditions are applied on all boundaries. The resulting solutions with classical GFDM differential operators without any imposed diagonal dominance, and those with approximate diagonal dominance as explained in this section are shown in Figure 2.1 at various times. The figure illustrates the instabilities that often develop when diagonal dominance is not used. Such instabilities have often been the reason that many meshfree communities avoid meshfree GFDMs. However, as shown here, these instabilities can be easily avoided. In the remainder of this thesis, this outlined procedure for most possible diagonal dominance is always used when the Laplace operator is considered in the classical GFDM framework.

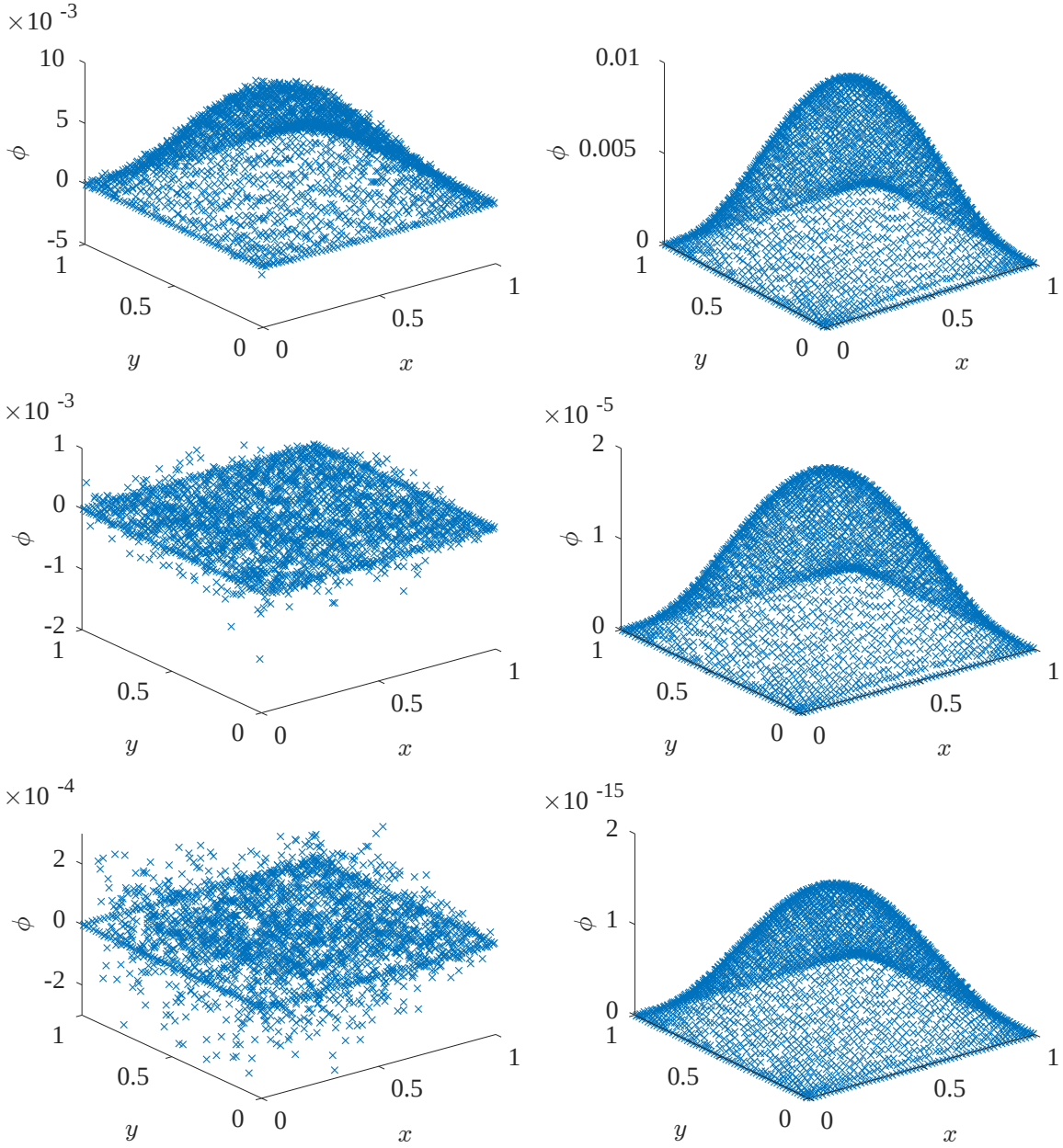


Figure 2.1: Heat equation results for $h = 0.05$, $N = 2521$ and $\Delta t = 0.01$. Without diagonal dominance (left column) and with diagonal dominance (right column). $t = 0.35$ (first row), $t = 0.7$ (middle row) and $t = 2$ (bottom row).

2.6 Direct GFDM Differential Operators

The function approximation stencil Eq. (2.19) leads to the question : 'would it be possible to arrive at function approximation stencils that also incorporate a minimization of errors in the underlying PDEs being solved?'. This would lead to a situation in which computing a function that satisfies the function approximation stencil at each point would directly give a solution to the PDEs being considered. This question was answered by Tiwari and Kuhnert [111]. Here, we refer to their method as the direct GFDM. In our earlier work, the same has also been referred to as Tiwari's approach. This method extends the Taylor expansion formulation of GFDM differential operators. Derivative and function approximation stencils in this approach are given by

$$\partial^* u(\vec{x}_i) \approx \tilde{\partial}^*_i u = \sum_{j \in S_i} \alpha^*_{ij} u_j + \sum_{r=1}^{p} \zeta^*_{ir} g_r \,, \tag{2.62}$$

where p is the number of PDEs in the system being solved, g_r are dependent on each PDE, and α^*_{ij} and ζ^*_{ir} are the coefficients computed in the local minimization. While the stencil coefficients in classical GFDM differential operators were dependent only on the local geometry of the point cloud, the stencil coefficients in this approach depend on both the local geometry and the PDE being discretized. The process of determining the unknown coefficients is explained below.

In this method, stencil coefficients are found that include the PDE being solved as a local constraint. Then a function is found that satisfies the conditions on all the stencils computed. The least squares procedure for minimizing the errors includes not just the Taylor expansions, but also the error in solving the PDE itself. Consider the PDE Eq. (2.14) used in the previous section. The system Eq. (2.16) and Eq. (2.6) gets extended to

$$\begin{pmatrix} e_{i1} \\ \vdots \\ e_{in} \\ e_{\text{PDE}} \end{pmatrix} = \begin{pmatrix} 1 & \delta x_{i1} & \frac{1}{2}\delta x_{i1}^2 \\ \vdots & \vdots & \vdots \\ 1 & \delta x_{in} & \frac{1}{2}\delta x_{in}^2 \\ a & b & c \end{pmatrix} \begin{pmatrix} u_i \\ (u_x)_i \\ (u_{xx})_i \end{pmatrix} - \begin{pmatrix} u_1 \\ \vdots \\ u_n \\ d \end{pmatrix} , \tag{2.63}$$

$$\min J_i = \sum_{j \in S_i} W^2_{ij} e^2_{ij} + W^2_{\text{PDE}} e^2_{\text{PDE}} \,, \tag{2.64}$$

which is solved at each point $i = 1, \ldots, N$. The stencils to the function approximation of u are the only ones of interest. Proceeding in the same way as done in Section 2.5 leads to a system similar to that obtained in Eq. (2.11). The first row of this system gives the function approximation stencils[5]

$$u_i = \sum_{j \in S_i} \alpha^0_{ij} u_j + \zeta^0_i d \,. \tag{2.65}$$

We then find a function u that satisfies all these stencils by solving a large sparse implicit system with an iterative method.

$$(1 - \alpha^0_{ii}) u_i - \sum_{\substack{j \in S_i \\ j \neq i}} \alpha^0_{ij} u_j = \zeta^0_i d \,. \qquad i = 1, \ldots, N. \tag{2.66}$$

[5]Once again, the $\mathcal{O}(h^3)$ terms have been dropped for convenience.

We refer to this method as the *direct* GFDM because functions are computed directly with function approximation stencils, as in Eq. (2.66), without the need for the derivative stencils.

It is empirically observed that this results in a ('close' to) diagonally dominant system, and that no extra procedure needs to be performed for stability as was necessitated in the classical GFDM case. One of the advantages of this direct approach over the classical GFDM one is that it can be used to solve algebraically over-determined, but well-posed, systems. i.e. systems with more PDEs than variables. An important use of this is to solve PDE systems on boundary points along with the imposed boundary conditions, which is not possible by the classical GFDM. Tiwari and Kuhnert [111] used this framework for pressure-Poisson equations in Navier–Stokes solvers. There, an over-determined system was solved on boundary points, consisting of both the boundary conditions and the pressure-Poisson equation itself. The same idea was later extended for imposing free surface boundary conditions [112], for compressible flows [56], and recently for heat transfer boundary conditions by Reséndiz and Saucedo [91], among other applications. A similar idea of adding the PDE on boundaries was also done in context of the meshfree Radial Basis Functions (RBF), by using an additional set of nodes adjacent to the boundary [29], which was shown to increase accuracy over standard RBF. Using the direct GFDM, a similar increase in accuracy is obtained in meshfree GFDMs and can be done without the addition of extra nodes.

However, other than solving a PDE along with the boundary condition at boundary points, to the best of our knowledge, this framework has not been used to solve over-determined problems (where an over-determined system is solved across all points in the computational domain). A basic comparison in one spatial dimension between the classical GFDM and the direct approach of this section has been done by Illiev and Tiwari [43]. However, a more detailed comparison on multi-dimensional problems has not been done. Further, the advantage of the direct framework for non-trivial simulation domains and over-determined problems has not been explored. This lack of a detailed comparison between the two formulations has resulted in them being occasionally misunderstood to be more similar than they are. One of the goals of this thesis is to fill these gaps. Further in this chapter, we show the versatility of this direct method and also present its advantages over classical GFDMs with a series of numerical examples. In Chapter 5, we show how this framework can be used to devise new algorithms which reduce some problems with existing meshfree GFDM solvers for the incompressible Navier–Stokes equations by solving an over-determined problem across the entire computational domain.

2.6.1 Over-Determined Problems

This framework can be extended to solve over-determined problems[6] and problems with multiple variables as shown below. Consider the following one-dimensional system with two unknowns u and v, and p linear PDEs, which are assumed to form a well

posed system. .

$$a_r u + b_r u_x + c_r u_{xx} + d_r v + e_r v_x + f_r v_{xx} = g_r\,, \qquad r = 1, \cdots, p\,. \tag{2.67}$$

Locally, errors in Taylor expansions in both variables are minimized along with the errors in each PDE. The system Eq. (2.63) and Eq. (2.64) gets extended to

$$\underbrace{\begin{pmatrix} e^u_{i1} \\ \vdots \\ e^u_{in} \\ e^v_{i1} \\ \vdots \\ e^v_{in} \\ e_{\text{PDE},1} \\ \vdots \\ e_{\text{PDE},p} \end{pmatrix}}_{\vec{E}_i} = \underbrace{\begin{pmatrix} 1 & \delta x_{i1} & \frac{1}{2}\delta x_{i1}^2 & & & \\ \vdots & \vdots & \vdots & & & \\ 1 & \delta x_{in} & \frac{1}{2}\delta x_{in}^2 & & & \\ & & & 1 & \delta x_{i1} & \frac{1}{2}\delta x_{i1}^2 \\ & & & \vdots & \vdots & \vdots \\ & & & 1 & \delta x_{in} & \frac{1}{2}\delta x_{in}^2 \\ a_1 & b_1 & c_1 & d_1 & e_1 & f_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_p & b_p & c_p & d_p & e_p & f_p \end{pmatrix}}_{M_i} \underbrace{\begin{pmatrix} u_i \\ (u_x)_i \\ (u_{xx})_i \\ v_i \\ (v_x)_i \\ (v_{xx})_i \end{pmatrix}}_{\vec{a}_i} - \underbrace{\begin{pmatrix} u_1 \\ \vdots \\ u_n \\ v_1 \\ \vdots \\ v_n \\ g_1 \\ \vdots \\ g_p \end{pmatrix}}_{\vec{b}_i}, \tag{2.68}$$

$$\min\, J_i = \sum_{j \in S_i} W_{ij}^2 (e_{ij}^u)^2 + \sum_{j \in S_i} W_{ij}^2 (e_{ij}^v)^2 + \sum_{k=1}^{p} W_{\text{PDE},k}^2 (e_{\text{PDE},k})^2\,, \tag{2.69}$$

where the superscript in the errors e_{ij} represent the variable for which the Taylor expansion is being performed. The weights for the Taylor expansion error terms are taken as the same for each variable, and according to a Gaussian kernel as described earlier in Eq. (2.7). Weights for the PDE errors can be chosen flexibly. Throughout this thesis, we use $W_{\text{PDE},k}^2 = 2$, which is double that of the central value of the Gaussian weighting kernel $W_{ii}^2 = 1$. The question about optimal weights for the same is an interesting problem, but is not addressed in this thesis. Extension to higher spatial dimensions are straightforward, and an example of the same is provided in Chapter 5.

The above minimization leads to the unknowns $\vec{a}_i = [(M_i^T W_i^2 M_i)^{-1} M_i^T W_i^2]\vec{b}_i$. Once again, only the function approximation stencils are of interest to us, i.e. the rows corresponding to u and v (the first and fourth rows of the system). The resultant function approximation stencils obtained can be written as

$$u_i = \sum_{j \in S_i} \alpha_{ij}^u u_j + \sum_{j \in S_i} \beta_{ij}^u v_j + \sum_{r=1}^{p} \zeta_{ir}^u g_r\,, \tag{2.70}$$

$$v_i = \sum_{j \in S_i} \alpha_{ij}^v u_j + \sum_{j \in S_i} \beta_{ij}^v v_j + \sum_{r=1}^{p} \zeta_{ir}^v g_r\,, \tag{2.71}$$

where the coefficients α, β and ζ represent the values in the relevant rows of the matrix $[(M_i^T W_i^2 M_i)^{-1} M_i^T W_i^2]$. A numerically more efficient method to obtain these stencils, as compared to computing this matrix product and inverse, is shown in Appendix A.3.1.

[6] The term 'over-determined problem' is used to refer to a system with more PDEs than variables. For more details on the same, see Appendix B.

We note that the resultant sparse linear systems can be solved by both explicit and implicit methods. If explicit integration methods or iterative procedures like Gauss-Seidel methods are being used, the system Eq. (2.70) and Eq. (2.71) can be solved as

$$u_i^{[s+1]} = \sum_{j \in S_i} \alpha_{ij}^u u_j^{[s]} + \sum_{j \in S_i} \beta_{ij}^u v_j^{[s]} + \sum_{r=1}^{p} \zeta_{ir}^u g_r^{[s]} \,, \tag{2.72}$$

$$v_i^{[s+1]} = \sum_{j \in S_i} \alpha_{ij}^v u_j^{[s]} + \sum_{j \in S_i} \beta_{ij}^v v_j^{[s]} + \sum_{r=1}^{p} \zeta_{ir}^v g_r^{[s]} \,, \tag{2.73}$$

where the bracketed superscripts s and $s+1$ denote the iteration number. On the other hand, if implicit integration methods are being used, the system Eq. (2.70) and Eq. (2.71) can be solved as

$$(1 - \alpha_{ii}^u) u_i^{[s+1]} - \sum_{\substack{j \in S_i \\ j \neq i}} \alpha_{ij}^u u_j^{[s+1]} - \sum_{j \in S_i} \beta_{ij}^u v_j^{[s+1]} = \sum_{r=1}^{p} \zeta_{ir}^u g_r^{[s]} \,, \tag{2.74}$$

$$- \sum_{j \in S_i} \alpha_{ij}^v u_j^{[s+1]} + (1 - \beta_{ii}^v) v_i^{[s+1]} - \sum_{\substack{j \in S_i \\ j \neq i}} \beta_{ij}^v v_j^{[s+1]} = \sum_{r=1}^{p} \zeta_{ir}^v g_r^{[s]} \,. \tag{2.75}$$

As mentioned earlier, the system Eq. (2.74) and Eq. (2.75) is emperically observed to provide well conditioned linear systems that can easily be solved with standard iterative solvers.

2.6.2 Boundary Conditions

Similar to the classical meshfree GFDM case explained in Section 2.5.4, enforcement of boundary conditions in the direct GFDM framework is also quite straightforward. Consider the 2D PDE in 1 variable used earlier

$$au + bu_x + cu_y + du_{xx} + eu_{yy} + fu_{xy} = g \,. \tag{2.76}$$

An implicit discretization of this PDE, as described earlier in Eq. (2.63) – Eq. (2.66), would lead to the following sparse linear system

$$(1 - \alpha_{ii}) u_i - \sum_{\substack{j \in S_i \\ j \neq i}} \alpha_{ij} u_j = \zeta_i g \,. \qquad i = 1, \ldots, N. \tag{2.77}$$

Similar to the classical GFDM case, here the boundary conditions are enforced by replacing each relevant function approximation row in Eq. (2.77) by one for the boundary condition. For a Dirichlet boundary condition $u = g_D$, the first way to do the same, as done in existing literature, is to simply replace the relevant row in the sparse system with

$$u_i = g_D \,, \tag{2.78}$$

which was also the same procedure followed in the classical GFDM case. An alternative way to enforce dirichlet boundary conditions is by adding the respective boundary

condition in the least squares system, similar to that done in Eq. (2.68). Thus, the following system has to be solved

$$\begin{pmatrix} e_{i1} \\ \vdots \\ e_{in} \\ e_{\text{PDE}} \\ e_{\text{BC}} \end{pmatrix} = \begin{pmatrix} 1 & \delta x_{i1} & \delta y_{i1} & \frac{1}{2}\delta x_{i1}^2 & \frac{1}{2}\delta y_{i1}^2 & \delta x_{i1}\delta y_{i1} \\ & & \vdots & \vdots & & \\ 1 & \delta x_{in} & \delta y_{in} & \frac{1}{2}\delta x_{in}^2 & \frac{1}{2}\delta y_{in}^2 & \delta x_{in}\delta y_{in} \\ a & b & c & d & e & f \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_i \\ (u_x)_i \\ (u_y)_i \\ (u_{xx})_i \\ (u_{yy})_i \\ (u_{xy})_i \end{pmatrix} - \begin{pmatrix} u_1 \\ \vdots \\ u_n \\ g \\ g_D \end{pmatrix}, \tag{2.79}$$

where the last row in the system is for the Dirichlet boundary condition, and the one above that is for the PDE itself. This system is solved with a minimization procedure as explained earlier. Clearly, this method only enforces the Dirichlet boundary condition in a least squares sense. Enforcing the PDE on Dirichlet boundaries, as done above, can be useful in several circumstances. For example, for enforcing the velocity divergence-free condition at no-slip walls for incompressible fluid flow simulations.

The ease of switching between the two ways to enforce Dirichlet boundary conditions also illustrates the fact that the two GFDM formulations presented here can be mixed very easily. Some points in the domain could be solved with one formulation, with the remaining points in the other.

Neumann boundary conditions $\vec{n} \cdot \nabla u = g_N$ can be enforced either in the way done in the classical GFDM case in Section 2.5.4, or could be done directly in a least squares sense by solving the following system

$$\begin{pmatrix} e_{i1} \\ \vdots \\ e_{in} \\ e_{\text{PDE}} \\ e_{\text{BC}} \end{pmatrix} = \begin{pmatrix} 1 & \delta x_{i1} & \delta y_{i1} & \frac{1}{2}\delta x_{i1}^2 & \frac{1}{2}\delta y_{i1}^2 & \delta x_{i1}\delta y_{i1} \\ & & \vdots & \vdots & & \\ 1 & \delta x_{in} & \delta y_{in} & \frac{1}{2}\delta x_{in}^2 & \frac{1}{2}\delta y_{in}^2 & \delta x_{in}\delta y_{in} \\ a & b & c & d & e & f \\ 0 & n_i^x & n_i^y & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_i \\ (u_x)_i \\ (u_y)_i \\ (u_{xx})_i \\ (u_{yy})_i \\ (u_{xy})_i \end{pmatrix} - \begin{pmatrix} u_1 \\ \vdots \\ u_n \\ g \\ g_N \end{pmatrix}, \tag{2.80}$$

where the addition of the PDE is optional. Free surface and other boundary conditions can be imposed in a similar way by adding the respective PDE(s) into the least squares system from which the function approximation stencil is obtained. Throughout this thesis, unless mentioned otherwise, we always use the least squares implementation of boundary conditions in this framework.

Of course, this least squares implementation of boundary conditions forms a less strict notion of enforcement of boundary conditions. An interesting similarity to note is that less strict notions of boundary conditions have also been becoming increasingly popular in meshed methods in the last decade, in the form of weak-form enforcement of boundary conditions [6, 92].

2.6.3 Consistency Conditions

We consider the 1D PDE used earlier in Eq. (2.14). Derivative stencils are computed by solving the minimization problem of Eq. (2.63) and Eq. (2.64). In the direct GFDM

framework, the resultant function approximation and derivative stencils can be given by

$$u_i = \sum_{j \in S_i} \alpha_{ij}^0 u_j + \zeta_i^0 d \,, \tag{2.81}$$

$$(u_x)_i = \sum_{j \in S_i} \alpha_{ij}^x u_j + \zeta_i^x d \,, \tag{2.82}$$

$$(u_{xx})_i = \sum_{j \in S_i} \alpha_{ij}^{xx} u_j + \zeta_i^{xx} d \,. \tag{2.83}$$

Note that in this direct method, we only use the function approximation stencils of Eq. (2.81). The derivative stencils of Eq. (2.82) and Eq. (2.83) are never actually used. However, the following procedure involving the derivative stencils gives an important insight into this method.

We proceed in a way similar to that done in Section 2.5.3. Multiplying the Taylor expansions with α_{ij}^* and summing over all $j \in S_i$ leads to

$$\sum_{j \in S_i} \alpha_{ij}^* u_j = \left(\sum_{j \in S_i} \alpha_{ij}^* \right) u_i + \left(\sum_{j \in S_i} \alpha_{ij}^* \delta x_{ij} \right) (u_x)_i + \left(\sum_{j \in S_i} \alpha_{ij}^* \frac{1}{2} \delta x_{ij}^2 \right) (u_{xx})_i \,, \tag{2.84}$$

with the error terms neglected. Multiplying the PDE, Eq. (2.14) with ζ_i^* and adding the result to Eq. (2.84) leads to

$$\begin{aligned} &\sum_{j \in S_i} \alpha_{ij}^* u_j + \zeta_i^* d = \\ &\left(\sum_{j \in S_i} \alpha_{ij}^* + \zeta_i^* a \right) u_i + \left(\sum_{j \in S_i} \alpha_{ij}^* \delta x_{ij} + \zeta_i^* b \right) (u_x)_i + \left(\sum_{j \in S_i} \alpha_{ij}^* \frac{1}{2} \delta x_{ij}^2 + \zeta_i^* c \right) (u_{xx})_i \,. \end{aligned} \tag{2.85}$$

Comparing this with the stencils Eq. (2.81) – Eq. (2.83) leads to the following

$$\sum_{j \in S_i} \alpha_{ij}^0 + \zeta_i^0 a = 1 \,, \tag{2.86}$$

$$\sum_{j \in S_i} \alpha_{ij}^0 \delta x_{ij} + \zeta_i^0 b = 0 \,, \tag{2.87}$$

$$\sum_{j \in S_i} \alpha_{ij}^0 \frac{1}{2} \delta x_{ij}^2 + \zeta_i^0 c = 0 \,. \tag{2.88}$$

Similarly for the first derivative, we get

$$\sum_{j \in S_i} \alpha_{ij}^x + \zeta_i^x a = 0 \,, \tag{2.89}$$

$$\sum_{j \in S_i} \alpha_{ij}^x \delta x_{ij} + \zeta_i^x b = 1 \,, \tag{2.90}$$

$$\sum_{j \in S_i} \alpha_{ij}^x \frac{1}{2} \delta x_{ij}^2 + \zeta_i^x c = 0 \,. \tag{2.91}$$

Similarly for the second derivative, we get

$$\sum_{j\in S_i} \alpha_{ij}^{xx} + \zeta_i^{xx} a = 0\,, \tag{2.92}$$

$$\sum_{j\in S_i} \alpha_{ij}^{xx} \delta x_{ij} + \zeta_i^{xx} b = 0\,, \tag{2.93}$$

$$\sum_{j\in S_i} \alpha_{ij}^{xx} \frac{1}{2} \delta x_{ij}^2 + \zeta_i^{xx} c = 1\,. \tag{2.94}$$

Now, from Eq. (2.82), it is obvious that the derivative of a linear field x at point i is given by $\sum_{j\in S_i} \alpha_{ij}^x x_j + \zeta_i^x d$. Using the result of Eq. (2.90) and then Eq. (2.89) on this leads to

$$\tilde{\partial}_i^x x = \sum_{j\in S_i} \alpha_{ij}^x x_j + \zeta_i^x d\,, \tag{2.95}$$

$$= \sum_{j\in S_i} \alpha_{ij}^x \delta x_{ij} + \sum_{j\in S_i} \alpha_{ij}^x x_i + \zeta_i^x d\,, \tag{2.96}$$

$$= 1 - \zeta_i^x b + \left(\sum_{j\in S_i} \alpha_{ij}^x \right) x_i + \zeta_i^x d\,, \tag{2.97}$$

$$= 1 - \zeta_i^x b - \zeta_i^x a x_i + \zeta_i^x d\,. \tag{2.98}$$

Accuracy of gradient reconstruction thus *seems* to depend on the PDE being considered, and the resultant stencil values. Thus, it *appears* that even for the second order accurate Taylor expansions considered here, first and second order monomials need not be differentiated exactly.

It must be emphasized here that the stencils are computed with respect to a particular system of PDEs. It is thus not entirely reasonable to expect them to have general approximation properties up to a certain consistency order. *Supposed* errors according to Eq. (2.98) are not actually representative of the introduced error in the solutions in the direct GFDM framework.

When derivatives of known functions need to be computed directly, these derivative stencils are never actually used for the reasons explained above. In fact, numerically, they are never even computed. In all numerical examples later in this chapter and in Chapter 5, derivatives of known functions are computed using the classical GFDM differential operators. Thus, in such situations, both the function approximation stencil satisfying the PDE, and classical GFDM differential operator stencils need to be computed. This results in a higher clock time requirement to solve the local least squares systems. However, as the time comparisons in Chapter 5 suggest, this increased local cost is often offset by better conditioning of the global sparse linear systems, which results in faster convergence of the iterative solvers.

An interesting comparison between the different GFDM methods can be obtained by comparing the terms being minimized. For a 1 variable case, the minimizations

can be written as

$$\text{Classical GFDM: min } J_i = \sum_{j \in S_i} W_{ij}^2 e_{ij}^2 \,, \tag{2.99}$$

$$\text{Direct GFDM: min } J_i = \sum_{j \in S_i} W_{ij}^2 e_{ij}^2 + \sum_{k=1}^{p} W_{\text{PDE},k}^2 (e_{\text{PDE},k})^2 \,, \tag{2.100}$$

where p is the number of PDEs being solved. Comparing the functional $\sum_{j \in S_i} W_{ij}^2 e_{ij}^2$, which represents the errors in the Taylor expansions, for both cases (after the solution has been computed) would be a possible way to compare the basic approximations in the two methods. An error comparison in this regard is shown in Section 2.7, and later in Section 5.4.4. These numerical results suggest that the additional error introduced in the Taylor expansions in the direct GFDM framework is small, and often worth the price to obtain improved accuracy in some other aspect.

2.7 Numerical Comparisons

The first two examples in this chapter were carried out in MATLAB R2015b on an Intel Core i7-4770 CPU rated at 3.40GHz, while the last one was done in Fortran and was run on an Intel XeonE5-2670 CPU rated at 2.60GHz.

2.7.1 Laplace Problem on a Simple Domain

We begin by solving a basic Laplace equation on a rectangular domain $[-20, 20] \times [-2, 2]$. The domain is discretized with an irregularly distributed point cloud.

$$\Delta \phi = 0 \,, \tag{2.101}$$

$$\phi = \phi_l \,, \qquad \vec{x} \in \partial\Omega_{left} \,, \tag{2.102}$$

$$\phi = \phi_r \,, \qquad \vec{x} \in \partial\Omega_{right} \,, \tag{2.103}$$

$$\vec{n} \cdot \nabla \phi = 0 \,, \qquad \vec{x} \in \partial\Omega_{wall} \,, \tag{2.104}$$

with $\phi_l > \phi_r$. $\partial\Omega_{left}$ and $\partial\Omega_{right}$ are the boundaries at $x = -20$ and $x = 20$ respectively; and $\partial\Omega_{wall}$ consists of the boundaries at $y = -2$ and $y = 2$. For $\phi_l = 1$ and $\phi_r = 0$, the obtained analytical solution is

$$\phi_e = -\frac{1}{40} x + \frac{1}{2} \,. \tag{2.105}$$

The error in the numerical solution, ϕ, is measured as

$$\epsilon_\infty = \frac{\|\phi - \phi_e\|_\infty}{\|\phi_e\|_\infty} \,. \tag{2.106}$$

In the classical GFDM, we solve the Laplace equation on interior points and the respective boundary condition on boundary points. In the direct approach, we solve only the Laplace equation on interior points, and both the Laplace equation and the respective boundary condition on boundary points.

We consider a coarse point cloud of $h = 2.0$ and $N = 333$ points, which is shown in Figure 2.2. The errors obtained for classical and direct cases are $\epsilon_\infty = 2.1 \times 10^{-14}$ and $\epsilon_\infty = 3.0 \times 10^{-14}$ respectively. Thus, both methods accurately capture the solution, even for a very coarse spatial discretization. This also illustrates the stability of the direct GFDM without the need of an additional procdure to attain diagonal dominance as is done in the classical GFDM.

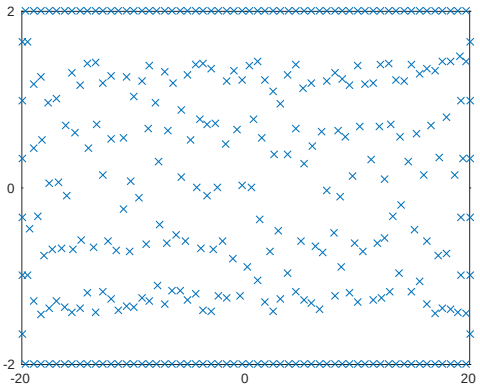


Figure 2.2: Point cloud for the simple domain of the Laplace Problem.

We now do the comparison mentioned in Section 2.6.3. After the implicit system arising in each method has been solved for the solution, the errors in Taylor expansions are compared for the computed solutions.

$$J_i = \sum_{j \in S_i} W_{ij}^2 e_{ij}^2 \,. \tag{2.107}$$

The mean value of this error over all points for the domain considered is $J_{\text{mean}} = 6.987 \times 10^{-4}$ for the classical GFDM case, and $J_{\text{mean}} = 6.995 \times 10^{-4}$ for the direct GFDM case, which suggests that the extra term in the minimization does not significantly affect the errors in the Taylor expansions in the direct GFDM.

We further extend this example to a case where the analytical solution is not differentiated exactly by the numerical differential operators. While the linear exact solution case considered above only showed that both methods produce results with close to machine precision, the following case is more appropriate to compare the approximation errors. Consider the Laplace equation $\Delta\phi = 0$ on the rectangular domain mentioned above with the exact solution

$$\phi_e = \exp(y)\sin(x)\,, \tag{2.108}$$

with Dirichlet boundary conditions on all boundaries. Errors in the numerical solution are measured as in Eq. (2.106) and as follows

$$\epsilon_2 = \frac{\|\phi - \phi_e\|_2}{\|\phi_e\|_2}\,. \tag{2.109}$$

The convergence of the two errors are shown in Figure 2.3 which illustrates that both methods produce similar results and convergence rates. The direct GFDM case produces marginally more accurate results because the Laplace equation is also solved on boundary points in addition to the Dirichlet boundary condition.

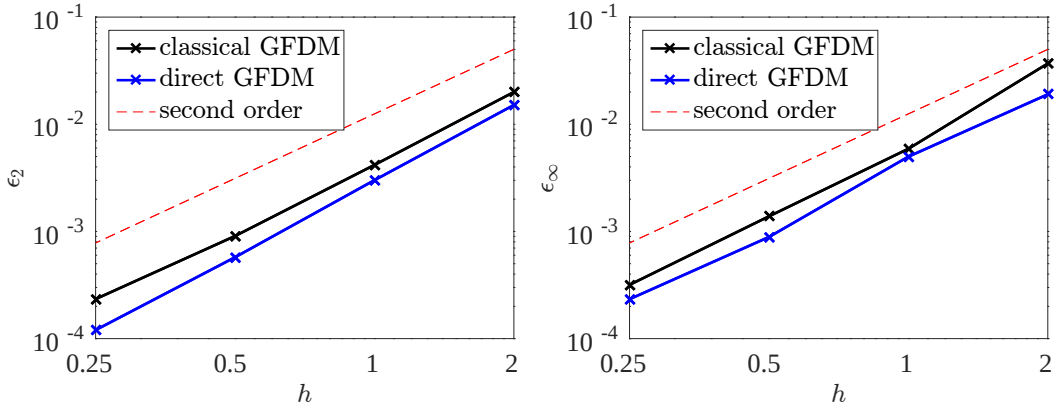


Figure 2.3: Errors in the laplace equation on a rectangular domain with exact solution $\phi_e = \exp(y)\sin(x)$: ϵ_2 (left) and ϵ_∞ (right).

2.7.2 Laplace Problem on a Non-Trivial Domain

The difference between the two aforementioned methods becomes more pronounced for more complex simulation domains. We now solve the basic Laplace equation on a domain with two equal rectangular sections connected by a narrow channel, as shown in Figure 2.4, with the same overall dimensions as the previous example. The dimensions of the narrow channel are $[-0.5, 0.5] \times [-0.2, 0.2]$.

Figure 2.4: Non-trivial domain of Laplace problem.

In this case, the boundaries $\partial\Omega_{left}$ and $\partial\Omega_{right}$ are the left-most and right-most vertical boundaries in Figure 2.4 respectively, and $\partial\Omega_{wall}$ is the remainder of the boundary. The solution can be studied analytically by considering the Lapalce equation as the steady state of a diffusion equation $\phi_t = \Delta\phi$ with the same boundary conditions. For the steady state, the flux should be constant throughout the domain

$$A\frac{\partial\phi}{\partial x} = \text{constant}, \tag{2.110}$$

where $A = A(x)$ is the cross-sectional area, which is the vertical height in the two dimensional case of Figure 2.4. Further, the total gradient in the horizontal x direction should be the same as the total change in ϕ across the two ends, from left to right

$$\int_\Omega \frac{\partial\phi}{\partial x}dx = \phi_l - \phi_r. \tag{2.111}$$

Assuming the gradient of ϕ in the x direction is constant in each of the three regions of the domain, the exact solution, ϕ_e, can be obtained using Eq. (2.110) and Eq. (2.111), and is shown in Figure 2.5.

Once again, in the classical GFDM case, we solve the Laplace equation on interior points and the respective boundary condition on boundary points. In the direct GFDM approach, we solve only the Laplace equation on interior points, and both the Laplace equation and the respective boundary condition on boundary points. Figures 2.6 – 2.8 show the change of the result ϕ in the x direction, for both methods with $\phi_l = 1$, $\phi_r = 0$. The y direction, in which the results are constant, is perpendicular to the plane of the paper. The smoothing length h is taken to be constant throughout the domain. Figures 2.6 and 2.7 illustrate that the classical GFDM produces incorrect results for coarse point clouds. This is due to an insufficient number of points in the narrow channel. On the other hand, the direct approach gives much better results in these cases as the Laplace equation is also solved on the boundary points. Both methods produce similar results for fine point clouds, as shown in Figure 2.8. Clearly, to obtain similar accuracy, the classical GFDM requires more points than the direct GFDM, which results in higher clock times.

Further, we compare errors in Taylor expansions as explained in the previous example. This is done for the $h = 0.1$ case shown in Figure 2.8, since the classical GFDM produces incorrect results for the coarser point clouds considered. In contrast to the previous section, the mean value of this error is calculated separately for interior and boundary points. The difference between the two methods is minimal for interior points, $J_{\text{mean}} = 1.102 \times 10^{-6}$ for the classical GFDM case, and $J_{\text{mean}} = 1.113 \times 10^{-6}$ for the direct GFDM case. On the other hand, a slightly larger difference is noticed for boundary points, $J_{\text{mean}} = 9.036 \times 10^{-6}$ for the classical GFDM case, and $J_{\text{mean}} = 9.315 \times 10^{-6}$ for the direct GFDM case. This once again illustrates that the errors in the Taylor expansions are not significantly increased by the additional term(s) in the local least squares minimization. This same comparison is also done for a more complex over-determined system in Section 5.4.4, which gives similar results.

This example illustrates the point that solving the governing PDE on boundary points along with the relevant boundary conditions can become essential when there are very few interior points in a region of the domain. Apart from the academic examples considered here, such situations also often occur practically in fully Lagrangian simulations of fluid flows. If a free surface is present, it is possible that the movement of the fluid results in a small number of points breaking away from the main point cloud, resulting in very few interior points in that region. A similar situation occurs when a thin layer of fluid develops during a simulation. In both these cases, the direct GFDM produces more accurate results than the classical GFDM due to the addition of the PDE in the local least squares system at boundary points.

2.7.3 General "Elliptic" Over-Determined Problems

In Section 2.6.1, we showed how the direct GFDM can be used to solve over-determined problems. An explanation of the types of problems being referred to here is given in Appendix B. In this section, we give an example of the same. Consider the domain of

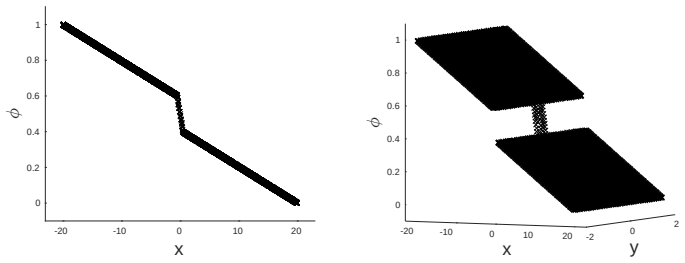


Figure 2.5: Laplace equation: exact solution from two viewpoints.

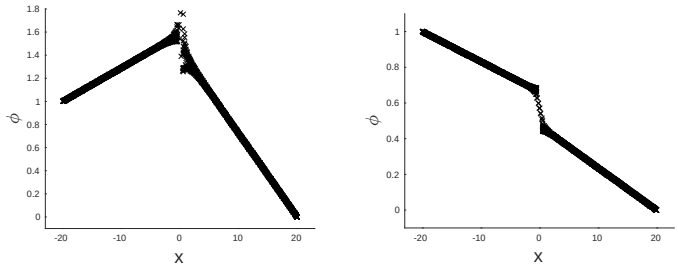


Figure 2.6: Laplace equation results, $h = 0.7$, $N = 2\,248$. Classical GFDM (left): $\epsilon_\infty = 1.4343$. Direct GFDM (right): $\epsilon_\infty = 0.1032$.

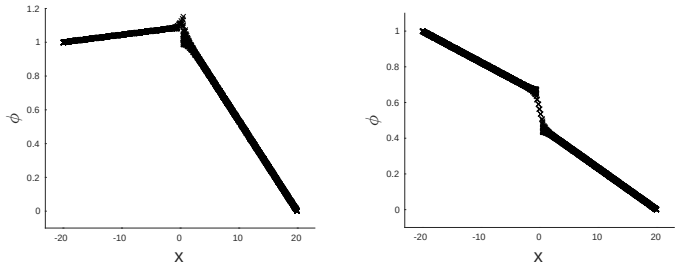


Figure 2.7: Laplace equation results, $h = 0.6$, $N = 2\,960$. Classical GFDM (left): $\epsilon_\infty = 0.7557$. Direct GFDM (right): $\epsilon_\infty = 0.0982$.

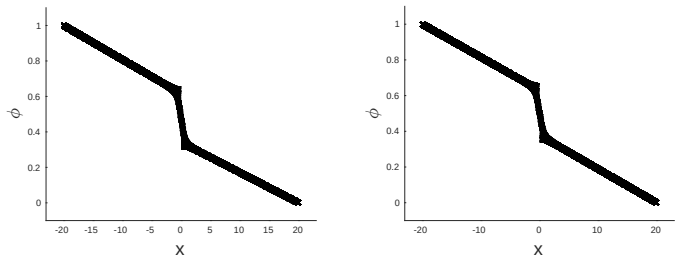


Figure 2.8: Laplace equation results, $h = 0.1$, $N = 91\,693$. Classical GFDM (left): $\epsilon_\infty = 0.08$. Direct GFDM (right): $\epsilon_\infty = 0.053$.

$[0,1] \times [0,1]$ and the following first order system of PDEs in unknowns T and ϕ

$$-T + \phi_x = x\,, \tag{2.112}$$
$$T_x = -1\,, \tag{2.113}$$
$$\phi_y = 0\,. \tag{2.114}$$

A discussion on the ellipticity of similar first order systems has been done by Krupchyk *et al.* [54]. In the direct GFDM, this system (with appropriate boundary conditions) can be solved by adding the three equations to the least squares systems. If this system is to be solved by the classical GFDM, where a square system[7] is preferable, two options arise. The first is to consider a weighted linear combination of two of the equations in the system to reduce the number of equations. The second strategy would be to observe that any solution of the above first order system also satisfies the independent Laplace equations

$$\Delta T = 0\,, \tag{2.115}$$
$$\Delta \phi = 0\,. \tag{2.116}$$

Solving this Laplace system numerically has the disadvantage that a solution to the Laplace system need not always be a solution to the first order system above. An example of this is when the following boundary conditions are used

$$\vec{n}\cdot\nabla T = 1\,, \qquad \phi = 0\,, \qquad \text{if } x = 0\,, \tag{2.117}$$
$$-\vec{n}\cdot\nabla T = 1\,, \quad \vec{n}\cdot\nabla\phi = 1\,, \qquad \text{if } x = 1\,, \tag{2.118}$$
$$\vec{n}\cdot\nabla T = 0\,, \quad \vec{n}\cdot\nabla\phi = 0\,, \qquad \text{on remaining boundaries}\,, \tag{2.119}$$

where $\vec{n}$ is the outward pointing unit normal. The first order system with these boundary conditions has the unique solution $T = 1 - x$ and $\phi = x$. Using the direct GFDM to solve the first order systems leads to very accurate results. For a coarse point cloud with $h = 0.1$ and $N = 709$ points, the relative errors obtained are $\epsilon_\infty = 2.74\times10^{-8}$ and $\epsilon_\infty = 8.03\times10^{-8}$ for T and ϕ respectively. On the other hand, the Laplace system with these boundary conditions does not have a unique solution for T. Of course, the boundary conditions can be modified for the Laplace system to include, for example, $T = 1$ at $x = 0$ to ensure that the two systems are equivalent. However, the required modification is not always trivial to ascertain. Moreover, even if the two systems were to be equivalent theoretically (putting aside the difference in required regularity of the solution), it could still sometimes be beneficial to solve the first order over-determined system numerically. A major reason for this is that in meshfree GFDMs, and several other meshfree methods too, the first and second derivatives are not consistent with each other. In the sense that $\Delta \neq \nabla\cdot\nabla$ at the discrete level. Thus, solving the Laplace system of Eq. (2.115) and Eq. (2.116) introduces an extra error in the system Eq. (2.112) – Eq. (2.114) due to this inconsistency. Resulting inaccuracies due to such errors in the specific case of fluid flow are explained in Chapter 5.

[7] By square system, we mean a system with the same number of PDEs and variables.

Chapter 3

Conservation for Meshfree GFDMs

Consider a conservation law

$$\frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{J} = 0\,, \tag{3.1}$$

with $\mathbf{J} = \mathbf{J}(\phi)$. For sufficiently smooth ϕ and $\mathbf{J}$, integrating Eq. (3.1) over the entire domain, and an application of the divergence theorem leads to the integral form of the conservation law,

$$\frac{d}{dt}\int_{\Omega} \phi \, dV = -\int_{\partial\Omega} \vec{n} \cdot \mathbf{J} \, dA\,. \tag{3.2}$$

Physically, Eq. (3.2) can be stated as: the rate of change of energy within the domain should be the same as the energy flux across its boundary, when no energy source or sink is present.

Finite volume methods (FVMs) [26] use a local balance on each discretization cell or control volume. FVMs solve Eq. (3.1) by directly enforcing a local, discrete version of Eq. (3.2). Meshfree GFDMs, on the other hand, usually directly solve the strong form of Eq. (3.1). The lack of a discrete divergence theorem causes the absence of a discrete form of Eq. (3.2). This combined with the local nature of the differential operators, Eq. (2.4), results in conservation not being ensured at a discrete level. Since this problem is inherent due to the purely local nature of approximations, it also affects most other meshfree methods.

Conservative and mimetic[1] properties of finite difference methods on different mesh structures has been a topic of extensive study (see, for example, [4, 15, 61, 102]). These include both uniform and non-uniform mesh spacing. However, most of these methods use structured, orthogonal grids; though these ideas have even been generalized to finite differences on polyhedral meshes [62]. A key point to note is that all these methods rely on the underlying grid structure to obtain conservation and mimetic properties in numerical differential operators. Occasionally, such finite difference methods have also been referred to as GFDMs. We emphasize the distinction between such mesh-based so-called GFDMs and meshfree GFDMs. In contrast to the mesh-based methods, meshfree GFDMs use arbitrarily spaced point clouds without any underlying grid defining the connectivity. The absence of an underlying grid means that ideas behind mesh-based conservative finite differences that rely on the grid can not be generalized easily to meshfree GFDMs. Conservation for meshfree GFDMs has been an elusive goal, and it does not feature widely in meshfree literature.

[1] Mimetic properties of a numerical method refer to those which mimic or imitate some quality of the continuum.

In this chapter, we present the need for conservation, and show the requirements for conservation in meshfree GFDMs. We further present the method of Chiu *et al.* [58, 59] who introduced a procedure to get conservative first derivatives in a meshfree GFDM setting. We then present a novel generalization of that work that removes the requirement of symmetry on stencil coefficients. Further, we show how the procedure used in this generalization can also be applied to obtain conservative higher derivatives, which is not possible for the method of Chiu *et al.*. We then explain why both these methods are very inefficient for most practical problems. In Chapter 4, we present a new efficient method to introduce approximate conservation in meshfree GFDMs. In both this chapter and the next, we focus on conservation with regard to the classical meshfree GFDM. However, the work can also be extended to the direct GFDM framework presented in Section 2.6.

3.1 Discrete Divergence Theorem

3.1.1 Conservative First Derivatives

Consider the divergence theorem

$$\int_\Omega \nabla \cdot \vec{g}\, dV = \int_{\partial\Omega} \vec{n} \cdot \vec{g}\, dA\,, \tag{3.3}$$

for sufficiently smooth $\vec{g}$ and $\partial\Omega$. $\vec{n} = (n^x, n^y, n^z)$ is the outward pointing unit normal. Setting $\vec{g}$ to be $(f,0,0)$, $(0,f,0)$ and $(0,0,f)$ alternatingly, a scalar version of Eq. (3.3) can be obtained as

$$\int_\Omega \partial^k f\, dV = \int_{\partial\Omega} f n^k\, dA\,. \tag{3.4}$$

Throughout this thesis, the superscript $k = x, y, z$ denotes the spatial dimension corresponding to which the derivative is being taken. Note that while the superscript $*$ for the differential operators used earlier represents any differential operator, the superscript k is used to only denote first derivatives.

We wish to find discrete differential operators of the form Eq. (2.4) satisfying the following discrete version of Eq. (3.4)

$$\sum_{i=1}^{N} V_i \tilde{\partial}_i^k f = \sum_{i\in\partial\Omega} f_i n_i^k\, A_i\,, \tag{3.5}$$

where V_i is a volume associated with point i, A_i is the portion of area of the boundary associated with boundary point i, and n_i^k is the k component of the outward pointing unit normal at boundary point i.

In addition to obtaining global conservation, a further advantage of differential operators satisfying a discrete divergence theorem is that it results in some more general mimetic properties. We now show that such a conservative solution satisfies a discrete version of integration by parts

$$\int_\Omega g\partial^k f\, d\Omega = \int_{\partial\Omega} f g\, n^k\, dA - \int_\Omega f\partial^k g\, d\Omega\,. \tag{3.6}$$

Theorem 3.1.1 (Summation by parts)**.** *If the discrete divergence theorem for the first derivatives, Eq.* (3.5)*, holds, then the following discrete version of Eq.* (3.6) *also holds*

$$\sum_{i=1}^{N} V_i g_i \tilde{\partial}_i^k f = \sum_{i \in \partial\Omega} f_i g_i n_i^k \, A_i - \sum_{i=1}^{N} V_i f_i \tilde{\partial}_i^k g \,. \tag{3.7}$$

Proof. By expanding the term $\tilde{\partial}_i^*[(f - f_i)(g - g_i)]$, it is easy to see that the following "product rule" holds.

$$\tilde{\partial}_i^* fg = f_i \tilde{\partial}_i^* g + g_i \tilde{\partial}_i^* f + \tilde{\partial}_i^*[(f - f_i)(g - g_i)] \,. \tag{3.8}$$

Now, starting from Eq. (3.8), and applying the discrete divergence theorem Eq. (3.5) to $(f - f_i)(g - g_i)$ and then fg, we get

$$\tilde{\partial}_i^k fg - f_i \tilde{\partial}_i^k g - g_i \tilde{\partial}_i^k f = \tilde{\partial}_i^k[(f - f_i)(g - g_i)] \,, \tag{3.9}$$

$$\sum_{i=1}^{N} V_i \left(\tilde{\partial}_i^k fg - f_i \tilde{\partial}_i^k g - g_i \tilde{\partial}_i^k f \right) = \sum_{i=1}^{N} V_i \tilde{\partial}_i^k[(f - f_i)(g - g_i)] \,, \tag{3.10}$$

$$= \sum_{i \in \partial\Omega} (f_i - f_i)(g_i - g_i) n_i^k \, A_i \,, \tag{3.11}$$

$$= 0 \,, \tag{3.12}$$

$$\sum_{i=1}^{N} V_i \left(f_i \tilde{\partial}_i^k g + g_i \tilde{\partial}_i^k f \right) = \sum_{i=1}^{N} V_i \tilde{\partial}_i^k fg \,, \tag{3.13}$$

$$= \sum_{i \in \partial\Omega} f_i g_i n_i^k \, A_i \,. \tag{3.14}$$

which leads to Eq. (3.7). □

Note that setting $g \equiv 1$ leads to the special case of the discrete divergence theorem, Eq. (3.5).

3.1.2 Conservative Laplace Operators

In a similar manner to that done above, the conservative criteria for higher order derivatives can also be derived. Setting $\vec{g} = \nabla\phi$ in Eq. (3.3) for some scalar valued function ϕ, we get

$$\int_\Omega \Delta\phi \, dV = \int_{\partial\Omega} \vec{n} \cdot \nabla\phi \, dA \,. \tag{3.15}$$

We wish to find a discrete Laplace operator satisfying the following discrete version of Eq. (3.15)

$$\sum_{i=1}^{N} V_i \tilde{\partial}_i^{\Delta} \phi = \sum_{i \in \partial\Omega} \vec{n}_i \cdot \tilde{\partial}_i^{\nabla} \phi \, A_i \,, \tag{3.16}$$

where $\tilde{\partial}_i^{\nabla} = \left(\tilde{\partial}_i^x, \, \tilde{\partial}_i^y, \, \tilde{\partial}_i^z \right)^T$. Like in first derivatives case, it can be shown that satisfying this conservation criteria leads to additional mimetic properties of the discrete Laplace operator.

3.2 Globally Defined Conservative Differential Operators

3.2.1 Symmetric Conservative Differential Operators

A first attempt at conservation for first derivatives in meshfree GFDMs was made by Chiu *et al.* [58, 59]. They enforce a volume weighted symmetry condition on the differential operator stencil coefficients

$$V_i c_{ij}^k = -V_j c_{ji}^k \,, \quad \forall i \neq j \,, \tag{3.17}$$

in addition to an explicit control on the central stencil value

$$c_{ii}^k = \begin{cases} 0, & \text{if } i \notin \partial\Omega \,, \\ \frac{n_i^k A_i}{2V_i}, & \text{if } j \in \partial\Omega \,. \end{cases} \tag{3.18}$$

They go on to show that Eq. (3.17) and Eq. (3.18) lead to a discrete version of the divergence theorem, similar to Eq. (3.5). We show a different proof of the same in the next section. Chiu *et al.* further establish that a necessary condition for the existence of conservative differential operator stencils satisfying Eq. (3.17) and Eq. (3.18), in addition to the usual consistency conditions of Eq. (2.20), is that the monomial test functions should satisfy the discrete divergence theorem.

$$\sum_{i=1}^{N} V_i \tilde{\partial}_i^k m = \sum_{i \in \partial\Omega} m_i n_i^k \, A_i \,, \qquad \forall m \in \mathcal{M} \,. \tag{3.19}$$

While this condition was not proved to be sufficient, they state that empirically it was observed to be sufficient. Eq. (3.17), Eq. (3.18) and the monomial consistency conditions lead to the following global system for the differential operators

$$\sum_{j \in S_i} c_{ij}^k m_j = \partial_i^k m \,, \qquad \forall m \in \mathcal{M} \,, \forall i \,, \tag{3.20}$$

$$V_i c_{ij}^k = -V_j c_{ji}^k \,, \quad \forall i \neq j \,, \tag{3.21}$$

$$c_{ii}^k = \begin{cases} 0, & \text{if } i \notin \partial\Omega \,, \\ \frac{n_i^k A_i}{2V_i}, & \text{if } j \in \partial\Omega \,, \end{cases} \tag{3.22}$$

which is an under-determined system solved with a norm minimization procedure. While Eq. (3.20) and Eq. (3.22) are local and only depend on the support of each point i, the symmetry condition Eq. (3.21) couples all the local systems together. Thus, a very large sparse system needs to be solved to obtain the differential operators. On the other hand, in classical GFDMs, only small local systems need to be solved at each point. Thus, this method is very inefficient and can not be used for large point clouds or for moving point clouds in a Lagrangian framework where the differential operators need to be recomputed at every time step. We note that the use of global approximations, similar to this one, to achieve the goal of conservation in a meshfree method, have also been suggested in the unpublished work of Diyankov [21].

A further disadvantage of this method is that the symmetric assumption in Eq. (3.17) is not always a fair one. While it makes sense for symmetric rectilinear grids, the same can not be said for arbitrary point locations. Neighbourhoods S_i and S_j usually have different spatial distribution of points, and could even contain a different number of points. In such scenarios, the symmetric coefficients assumption has no basis. This problem is further compounded for applications in which the spatial density of points is not constant, or when one sided stencils are desired.

A further drawback of this framework is that it can not be generalized to higher derivatives. Extending this procedure to the discrete Laplace operator would require setting the central stencil value $c_{ii}^{\Delta} = 0$ for all points i which do not have any boundary neighbour. This leads to an unstable operator, as discussed in Section 2.5.5.

3.2.2 Non-Symmetric Conservative Differential Operators

In this section, we present a method to generalize the conservative differential operators of Chiu *et al.* [58, 59] presented in the previous section. The symmetry requirement of Eq. (3.17) is avoided by directly enforcing the discrete divergence theorem Eq. (3.5). Substituting the differential operator definitions Eq. (2.4) in Eq. (3.5) leads to

$$\sum_{i=1}^{N} \sum_{j \in S_i} V_i c_{ij}^k f_j = \sum_{i \in \partial\Omega} f_i n_i^k A_i \,. \tag{3.23}$$

To simplify notation, we introduce

$$\bar{c}_{ij} = \begin{cases} c_{ij}, & \text{if } j \in S_i \,, \\ 0, & \text{if } j \notin S_i \,. \end{cases} \tag{3.24}$$

Using this, Eq. (3.23) can be rewritten as

$$\sum_{i=1}^{N} \sum_{j=1}^{N} V_i \bar{c}_{ij}^k f_j = \sum_{i \in \partial\Omega} f_i n_i^k A_i \,. \tag{3.25}$$

Considering a basis for f in Eq. (3.25) consisting of Kronecker delta functions leads to

$$\sum_{i=1}^{N} V_i \bar{c}_{ij}^k = \begin{cases} 0, & \text{if } j \notin \partial\Omega \,, \\ n_j^k A_j, & \text{if } j \in \partial\Omega \,, \end{cases} \tag{3.26}$$

which should hold for each $j = 1, 2, \ldots, N$. Thus, the problem of enforcing a discrete divergence theorem for the first derivatives can be reduced to enforcing the "column sums" of Eq. (3.26).

Theorem 3.2.1 (Generalization of symmetric conservative differential operators)**.** *The symmetric conservative differential operators, Eq.* (3.17) *and Eq.* (3.18)*, of Chiu et al. form a specific case of Eq.* (3.26)*.*

Proof. Starting from the LHS of Eq. (3.26) and using Eq. (3.17)

$$\sum_{i=1}^{N} V_i \bar{c}_{ij}^k = V_j \bar{c}_{jj}^k - \sum_{\substack{i=1 \\ i \neq j}}^{N} V_j \bar{c}_{ji}^k \,, \tag{3.27}$$

$$= 2V_j \bar{c}_{jj}^k - V_j \sum_{i=1}^{N} \bar{c}_{ji}^k \,, \tag{3.28}$$

$$= 2V_j \bar{c}_{jj}^k - V_j \sum_{i \in S_j} c_{ji}^k \,, \tag{3.29}$$

$$= 2V_j \bar{c}_{jj}^k \,. \tag{3.30}$$

Now, using Eq. (3.18) directly leads to the RHS of Eq. (3.26). □

Thus, Eq. (3.26) generalizes the symmetric conservative differential operators, and removes the symmetry restriction. These non-symmetric conservative differential operators thus satisfy

$$\sum_{j \in S_i} c_{ij}^k m_j = \partial_i^k m \,, \qquad \forall m \in \mathcal{M} \,, \forall i \,, \tag{3.31}$$

$$\sum_{j=1}^{N} V_j \bar{c}_{ji}^k = \begin{cases} 0, & \text{if } i \notin \partial\Omega \,, \\ n_i^k A_i, & \text{if } i \in \partial\Omega \,. \end{cases} \tag{3.32}$$

The subscript ij in Eq. (3.26) has been swapped to ji to make the following evident. For a central point i, Eq. (3.31) is the usual monomial consistency condition or the "row sums" condition which elaborates how each neighbour $j \in S_i$ affects the differential operator stencil $\vec{c}_i^k$. On the other hand, Eq. (3.32) is the conservation or "column sums" condition which elaborates how the central point i affects the stencils of each of its neighbours $\vec{c}_j^k \;\; \forall j \in S_i, \;\; j \neq i$. Thus, similar to the symmetric differential operators of the previous section, these operators also need to be computed by solving a global system. Like before, Eq. (3.31) and Eq. (3.32) represent an under-determined system, which needs to be solved with a global norm minimization procedure, an example of which is explained in Appendix A.6.

Since these differential operators satisfy the discrete divergence theorem Eq. (3.5) for an arbitrary function, clearly they should also satisfy the same for the monomial test functions. Thus, a necessary condition for the existence of such conservative differential operators is the same as that shown in the previous section in Eq. (3.19). Besides the similarities of the global norm minimization and the necessary condition, this generalization has several advantages over the symmetric conservative differential operators of the previous section.

- This method removes the unrealistic symmetry requirement on the differential operators.
- Conservative first derivative operators found in this method are empirically observed to have lesser norms than the symmetric method, which results in improved stability.

- This procedure results in a sparse linear system which has a singly bordered block diagonal structure. This structure can be exploited to obtain more efficient solvers, which is not possible in the symmetric procedure. More details about the same are given in Appendix A.6.
- Unlike the symmetric method, the procedure laid out in this generalization can also be used to obtain conservative higher derivatives. Taking Kronecker delta basis functions as before, Eq. (3.16) gives

$$\sum_{i=1}^{N} V_i c_{ij}^{\Delta} = \sum_{i \in \partial\Omega} c_{ij}^{n} A_i \,, \quad \forall j \,, \tag{3.33}$$

  where $c_{ij}^{n} = c_{ij}^{x} n_i^x + c_{ij}^{y} n_i^y + c_{ij}^{z} n_i^z$. Note that the RHS of Eq. (3.33) is 0 for all points j which have no boundary neighbours. Since this method does not directly enforce a constraint on the central stencil value, diagonal dominance can be achieved in a way similar to that explained in Section 2.5.5.

The necessary condition for the existence of a solution in both these methods, Eq. (3.19), imposes a condition on the geometric attributes of the point cloud: the volume V_i associated with each point, the area A_i associated with each boundary point, and the outward pointing unit normal $\vec{n}_i$ for each boundary point. Chiu *et al.* enforce these conditions by solving a global optimisation problem. In the next chapter, we show that the same can be enforced locally in many cases, without the need to solve an additional global optimisation problem for the geometric attributes of the point cloud.

However, the major drawback in both these methods remains the need to solve for the differential operators globally. The extent of the same can be put into perspective by the following. Consider a point cloud with N points, with n_{mean} being the average number of neighbours per point. Let the number of test functions in the monomial set $\mathcal{M}$ be $|\mathcal{M}|$. The largest portion of time taken in meshfree GFDMs is for solving the large implicit system. For one variable, this consists of N unknowns, while the sparse system has a maximum of Nn_{mean} non-zero values. In addition to this, small local systems of size approximately $|\mathcal{M}| \times n_{\text{mean}}$ are being solved at each point. For the globally defined differential operators, these small local systems are replaced with one large global system. In the non-symmetric case presented above, this represents a system of Nn_{mean} unknowns with a maximum of $(|\mathcal{M}| + 1)\, Nn_{\text{mean}}$ non-zero values in the sparse linear system. In 3D, typical values used are $n_{\text{mean}} \approx 50$ and $|\mathcal{M}| = 10$. Thus, the global system to solve the differential operators has 50 times as many unknowns, and storing the linear system takes 11 times the amount of memory of that needed in the usual implicit system to obtain the solution. Clearly, these methods are unfeasible for large point clouds, or moving point clouds for which the differential operators need to be recomputed at every time step.

Since exact conservation with meshfree GFDMs remains extremely hard to achieve in an efficient way, this motivates the need to introduce less strict notions of conservation. Rather than trying to achieve a general discrete divergence theorem for every possible discrete field, in the next chapter, we introduce a method that tries to achieve the same only for the discrete fields of interest in the problem. i.e. We make sure that,

for example, the velocity field satisfies Eq. (3.5). This is done by conserving numerical fluxes across locally defined control cells.

Chapter 4

A Flux Conserving Meshfree Method

Lack of conservation has been the biggest drawback in meshfree GFDMs. In the last chapter, we showed that getting exact conservation is very inefficient for most point clouds, and we motivated the need to introduce less strict notions of conservation for meshfree GFDMs. In this chapter, we present a novel modification of classical meshfree GFDMs to include local balances which produce an approximate conservation of numerical fluxes. This is done by enforcing a local discrete divergence theorem within the usual moving least squares framework. This results in an approximate discrete form of global conservation, Eq. (3.2). Unlike Finite Volume Methods, the local balances are based on locally defined control cells, rather than a globally defined mesh. We present the application of this method to an advection diffusion equation and the incompressible Navier–Stokes equations. Our simulations show that the introduction of this flux conservation significantly reduces the errors in conservation in meshfree GFDMs.

4.1 Introduction

As explained in Chapter 2, in meshfree GFDMs numerical differential operators are found by solving an under-determined system. These systems are closed with a weighted norm minimization criterion. However, additional properties could also be enforced on the derivative stencils before the minimization procedure is carried out. In this chapter, we use this idea to enforce approximate conservation.

Despite the name, most meshfree methods are not completely devoid of meshes. While meshfree methods lack the use of a *predefined* mesh for domain discretization, many meshfree methods use an easily generable mesh such that the solution is not too heavily dependent on the quality of that mesh (see [64] for details). Several methods, which solve partial differential equations in their weak formulation, use a so-called background mesh for numerical integration of the weak-forms. The element-free Galerkin method (EFG) [9] uses a global background mesh, while the meshless local Petrov-Galerkin method (MLPG) [5] requires only local background cells. Meshfree particle methods, such as SPH, use a predefinition of mass particles, which often requires some kind of a mesh. One of the methods to solve the problem of particle distortion in SPH is by regularization, which requires a re-meshing [114]. Several methods in the family of point interpolation methods (PIM) [67] use tessellations or background cells in so-called T-schemes for the selection of support domains. Point-based meshfree GFDMs are often referred to as "truly meshfree". However, they too

use some notion of a mesh. They often use local triangulations for accurate post-processing operations or for the identification of points on the free surface of fluid flow simulations. In each of these cases, the meshes used are a lot easier to generate and do not impose the restrictions that meshes in classical meshed methods do. A question that then arises is that if such meshes are being defined anyway, why not use them to improve the solution quality in some way?

Finite Volume Methods (FVMs) have the feature of local conservation of numerical fluxes from one control volume to its neighbouring one [26]. In this chapter, we aim to use local control cells, in a manner similar to FVMs, in a meshfree framework to introduce an approximate conservation of numerical fluxes. A local control cell at a point is formed by the Delaunay tessellation of points in its support domain. The differential operators are computed using an MLS approach which guarantees usual monomial consistency coupled with a conservation of numerical fluxes for specific fields. For modeling fluid flow, we use a fully Lagrangian framework. Since point clouds can be modified locally, and the control cells are also defined locally, this method does not face the problem of cell distortion present in Lagrangian moving mesh methods. It must be noted that the idea of generalizing control volumes used by FVMs in the context of meshfree methods is not a novel one. The Finite-Volume Particle Method (FVPM) [37, 38] generalizes control volumes to include those which need not be disjoint. In FVPM, grid generation is replaced by an expensive integration of partition of unity functions. The Meshless-Finite Volume Method (MFVM) [39] generalizes Voronoi-based moving mesh methods. MFVM uses a volume partitioning that amounts to a Voronoi tessellation with edges smoothed. Unlike both the FVPM and MFVM, the method presented here is a strong-form method.

4.2 A Meshfree Control Cell

For each point i, we consider the Delaunay tessellation of the $n(i)$ points in its support domain S_i. The Voronoi diagram forms the dual graph to the Delaunay tessellation. Among the tessellations, the Voronoi cell containing point i is the only one of interest, and could be used as the control cell over which the flux balance is carried out. The Voronoi dual to the Delaunay tessellation is formed by connecting the circumcenters of the Delaunay simplexes (triangles in 2D or tetrahedrons in 3D). Instead of a traditional Voronoi dual, we use the central cell from a centroidal dual as the local control cell over which flux balances are carried out. This centroidal dual is formed by connecting the centroids of simplexes, instead of their circumcenters. This poses several advantages over the Voronoi dual. These advantages, along with the details about the construction of the cell numerically are detailed in Appendix C. While a centroidal dual is used to construct the local control cells, we use the term 'Voronoi cell' for notational simplicity. Figure 4.1 shows such a cell within the support domain. Each point i is associated with a volume V_i, taken to be the volume of this cell. V_i is also used for accurate post-processing. The local tessellations are also used in the computation of geometric parameters of the point cloud, such as the identification of free surface points in flows with open boundaries.

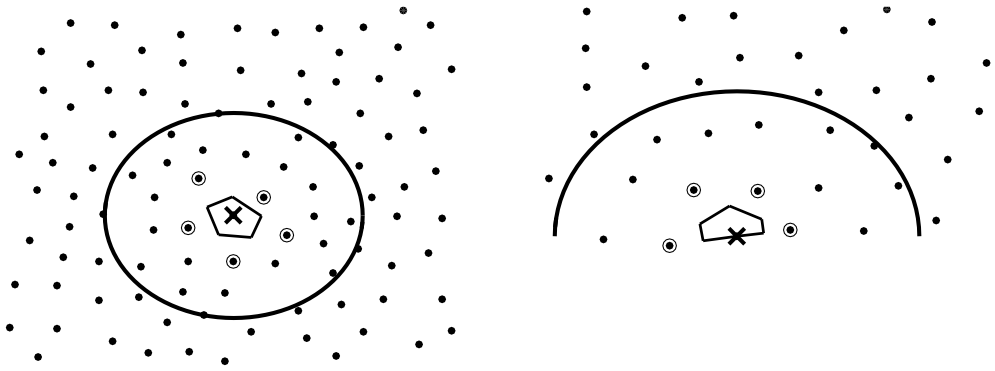


Figure 4.1: Control cell within a local support. Center point is marked with a cross. Nearest neighbours $l \in I_i$ are marked with circles. Interior point (left) and boundary point (right).

Using the local tessellation for the support S_i, we obtain a set of nearest neighbours I_i, in the Delaunay sense. For $l \in I_i$, we let $\widetilde{il}$ denote the dual edge (in 2D) or face (in 3D) of the Delaunay edge il. Further, we let A_{il} denote the area of $\widetilde{il}$ and $\vec{n}_{il}$ denote the unit normal of $\widetilde{il}$, pointing away from i. Exact details about the construction of $\widetilde{il}$ numerically, and the computation of A_{il} and $\vec{n}_{il}$ are given in Appendix C. For boundary points, the geometric area A_i and outward pointing unit normal $\vec{n}_i$ are used to truncate and close the semi-infinite Voronoi cell, as shown in Figure 4.1. To simplify notation, we define the closure $\bar{I}_i$ of I_i to include the point i itself, if i is a boundary point. Further, we let $\vec{n}_{ii} = \vec{n}_i$ and $A_{ii} = A_i$.

We note that since the Voronoi cells are defined locally on the support domain of each point, and not globally, they need not stitch together to form a global tiling of the computational domain. The uniqueness property of Delaunay tessellations suggests that the Voronoi cells should stitch together perfectly. However, that only holds for sufficiently large support domains. For small support domains, it is possible that an insufficient number of points in each neighbourhood leads to unsymmetric Voronoi cells. Under such situations, symmetry of the interface values, A_{ij} and A_{ji}; $\vec{n}_{ij}$ and $\vec{n}_{ji}$, is violated. Figure 4.2 shows such an example of adjacent non-symmetric cells. We measure the extent of stitching together to form a consistent global mesh by the error

$$\epsilon_{mesh} = \frac{\sum_{i=1}^{N} \sum_{l \in I_i} \|\vec{n}_{il} A_{il} + \vec{n}_{li} A_{li}\|}{2 \sum_{i=1}^{N} V_i} \,. \tag{4.1}$$

If $i \notin I_l$, we set $\vec{n}_{li} = \vec{0}$ and $A_{li} = 0$. If the cells are based on a global tessellation, $\vec{n}_{il} A_{il} = -\vec{n}_{li} A_{li} \, \forall i, l$, which leads to $\epsilon_{mesh} = 0$. Table 4.1 shows the extent of stitching together of locally defined Voronoi cells for different support sizes β (see Eq. (2.3)), and the corresponding average number of points in each support domain, on a sample 3D point cloud. Table 4.1 illustrates that for large enough support domains, which correspond to high values of β, the intersection of local Voronoi cells is minimum. Throughout this thesis, we use support sizes in the range of $\beta \in [0.75, 1]$. This results in approximately $45 - 55$ points in the support domain of each point in three dimensions, and about 20 points in two dimensions.

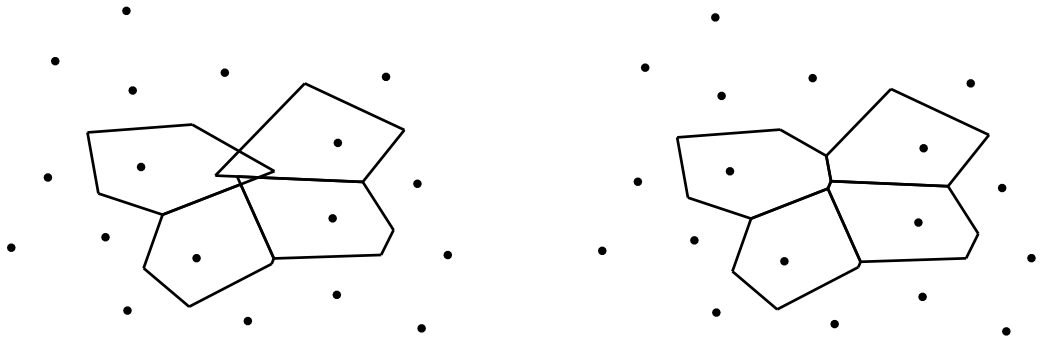


Figure 4.2: A 2D example of non-symmetric locally defined Voronoi cells (left), and the same point configuration with symmetric locally defined Voronoi cells (right). The non-symmetric cases occur when the support domains are not sufficiently large. The size of the support domains considered are $\beta = 0.6$, mean$(n_i) = 11$ (left) and $\beta = 0.7$, mean$(n_i) = 18$ (right).

Table 4.1: Errors in formation of a global tiling by stitching together locally obtained Voronoi cells (in 3D). The domain considered is a unit sphere with $h = 0.5$, $N = 1400$.

β	ϵ_{mesh}	mean(n_i)
0.5	3.31	16
0.55	1.08	21
0.6	0.33	27
0.65	9×10^{-3}	35
0.7	6×10^{-15}	41

Several meshfree methods, such as EFG [9] and PFEM [84], use a globally defined background mesh for a variety of purposes. Such a global background mesh could be used for the control cells, but defining cells based on local tessellations on a small number of points is significantly faster, especially when done in parallel. The concept of shapeless meshfree volumes for each point have also been proposed by several authors [58, 59, 48, 49], but their use presents several problems in the present context. Most importantly, the proposed meshless volumes are not closed, which, as we shall show later, is necessary for the construction of our flux conserving differential operators.

4.3 Flux Conserving Differential Operators

For the definition of our new numerical first derivative operators, we start by writing a local version of Eq. (3.4). This discrete, local version of the scalar form of the divergence theorem on the control cells can be written as follows

$$V_i \tilde{\partial}_i^k f = \sum_{l \in \bar{I}_i} F_{il}(f)\,, \tag{4.2}$$

where the superscript $k = x, y, z$ denotes the spatial dimension, $F_{il}(f)$ is a numerical flux function which depends on $\vec{n}_{il}$, A_{il}, f_i and f_l, and possibly their derivatives. One

possibility for F, to achieve a local discrete divergence theorem, can be

$$F_{il}(f) = \begin{cases} f_i n_i^k A_i & \text{if } i \in \partial\Omega \text{ and } l = i\,, \\ f_{il} n_{il}^k A_{il} & \text{elsewhere}\,, \end{cases} \tag{4.3}$$

with $f_{il} = \frac{f_i + f_l}{2}$. Further, $\vec{n}_{il}$ and A_{il} are determined locally based on the control cell as mentioned in Section 4.2. Throughout this chapter, we consider only simple symmetric flux functions similar to Eq. (4.3). However, more sophisticated flux functions can also be used and can also be coupled with flux functions in Eq. (2.4). The use of flux functions in the definition of the differential operators, Eq. (2.4), have been considered by Chiu *et al.* [58, 59]; and those specific to the Finite Pointset Method have been studied by Seifarth [98, 99].

For time-dependent problems, stencil coefficients for numerical differential operators are found such that they satisfy Eq. (4.2) for conservation of fluxes at the previous time level, in addition to the monomial consistency conditions, Eq. (2.20). Thus, if a field ϕ needs to be conserved, when proceeding from time step t^n to t^{n+1}, the discrete first-derivatives are found by the following minimization

$$\sum_{j \in S_i} c_{ij}^k m_j = \partial_i^k m\,, \qquad \forall m \in \mathcal{M}\,, \tag{4.4}$$

$$\sum_{j \in S_i} c_{ij}^k \phi_j^{(n)} = \frac{1}{V_i} \sum_{l \in \bar{I}_i} F_{il}(\phi^{(n)})\,, \tag{4.5}$$

$$\min\, J_i = \sum_{j \in S_i} \left(\frac{c_{ij}^k}{W_{ij}} \right)^2\,, \tag{4.6}$$

where the bracketed superscript denotes the time-level and $k = x, y, z$. We emphasize once again that the superscript k in Eq. (4.4) – Eq. (4.6) is used to denote only the first order derivatives in different directions, x, y, z, whereas the superscript * in Eq. (2.4) denotes any arbitrary differential operator. Eq. (4.4) – Eq. (4.6) constitute an enhancement of the classical GFDM differential operators by the conservation condition Eq. (4.5). If the monomials $m \in \mathcal{M}$ are taken up to second order, Eq. (4.4) represents 10 conditions in 3D and 6 conditions in 2D. Whereas the number of unknown c_{ij}^k values (for a particular k) is the same as the number of points in each support domain, which is typically around 50 and 20, in 3D and 2D respectively, for the support sizes being used. Thus, there is plenty of numerical freedom to impose the additional flux conservation condition, Eq. (4.5), without affecting the accuracy of gradient reconstructions, Eq. (4.4). Clearly, the stencil coefficients c_{ij}^k can be found locally at each point i, without the need for the global systems used by the previous conservative meshfree GFDMs presented in the previous chapter.

For each point $i = 1, 2, \ldots, N$, Eq. (4.4) represents the usual monomial consistency conditions as described by Eq. (2.20), which is dependent on all points in the support domain S_i. On the other hand, Eq. (4.5) represents the conservation of numerical fluxes at the previous time-level, where the fluxes are defined solely on the locally defined control cell, which depends only on the nearest neighbours $\bar{I}_i$. This difference is illustrated in Figure 4.1.

In explicit time-integration schemes where spatial derivatives are computed only at the previous time level, the differential operators defined by Eq. (4.4) – Eq. (4.6) degenerate to ones based solely on the locally defined control cell, making the scheme resemble Voronoi-based FVMs. When the derivative stencils are being used for explicit time-integration, they would only be applied to $\phi^{(n)}$. As a result, the spatial derivatives can be completely described by Eq. (4.5), which makes Eq. (4.4) irrelevant. Thus, the inclusion of Eq. (4.5) is only done with implicit time-integration schemes, which are used throughout this thesis.

The additional time taken due to the addition of Eq. (4.5) is not significant. As mentioned earlier, tessellations are usually performed in classical GFDMs for prescribing volumes to points, areas to boundary points and the detection of free surfaces. Thus, no extra tessellations need to be performed to determine the local control cells. Secondly, the size of the systems being solved is changed by a very small amount. For example, in 3D, for second order-accurate differential operators, classical GFDMs solve a system of 10 equations and about 50 unknowns at each point. The addition of Eq. (4.5) changes that to a system of 11 equations and 50 unknowns. Further, when the differential operators are defined locally as done in this chapter, the largest portion of time of meshfree GFDM simulations is taken by the iterative solvers for the large sparse linear systems obtained by the discretization of the PDEs. The computation of the differential operators takes up a much smaller portion of the simulation time. Thus, the addition of the flux conservation constraint, Eq. (4.5), has a minimal effect on the overall time taken by the entire simulation. This is shown in the numerical examples in Section 4.5.3.

Henceforth, for the sake of brevity, we denote this variant of classical GFDMs, in which the differential operators conserves fluxes of specific numerical fields, as FC-GFDM, where the FC stands for flux conserving. We note that Eq. (4.5) does not ensure the conservation of numerical fluxes of the field determined at level $n+1$ with respect to the given differential operator stencils. Thus, even with respect to the field ϕ, the use of differential operators defined by Eq. (4.4) – Eq. (4.6) would only form an approximately conservative method.

It should be ensured that the choice of the flux function F is done such that Eq. (4.4) and Eq. (4.5) are consistent with each other when ϕ in the neighbourhood of i is linearly dependent on the monomials $m \in \mathcal{M}$. For F defined as in Eq. (4.3), this is ensured by the fact that the Voronoi cell is closed and not self-intersecting by definition. This guarantees that geometric fluxes are conserved. For 0 order, the cell is closed,

$$\sum_{l \in \bar{I}_i} \vec{n}_{il} A_{il} = 0\,. \tag{4.7}$$

Similarly, first order geometric conservation ensures

$$\sum_{l \in \bar{I}_i} n_{il}^k x_{il}^{k'} A_{il} = \begin{cases} 0 & \text{if } k \neq k'\,, \\ V_i & \text{if } k = k'\,, \end{cases} \tag{4.8}$$

where $k, k' = x, y, z$ denote the spatial dimension, $\vec{x}_{il}$ is the geometric center of the edge or face $\widetilde{il}$ for $l \neq i$, given by $\vec{x}_{il} = \frac{\vec{x}_i + \vec{x}_l}{2}$; and $\vec{x}_{ii} = \vec{x}_i$. Eq. (4.8) is also used to determine the volume of the cell. We note that the geometric flux conservation

described above is the same as the necessary conditions for the existence of the globally defined conservative operators defined in the previous chapter in Eq. (3.19).

4.3.1 Higher Order Derivatives

The same procedure used above can be extended to compute numerical differential operators for higher order derivatives. This is illustrated with a diffusion operator. Consider the diffusion operator D such that $D\phi = \nabla \cdot (\alpha \nabla \phi)$. The divergence theorem applied to the vector field $\alpha \nabla \phi$ leads to

$$\int_{\Omega} \nabla \cdot (\alpha \nabla \phi)\, dV = \int_{\partial\Omega} \vec{n} \cdot (\alpha \nabla \phi)\, dA\,. \tag{4.9}$$

A discrete local version of Eq. (4.9) on the control cells can be written as

$$V_i \tilde{\partial}_i^D \phi = \sum_{l \in \bar{I}_i} G_{il}(\phi, \alpha)\,, \tag{4.10}$$

where G is a numerical flux function. Throughout this chapter, G is taken to be symmetric, similar to Eq. (4.3).

$$G_{il}(\phi, \alpha) = \begin{cases} \alpha_i \vec{n}_i \cdot \tilde{\partial}_i^{\nabla} \phi A_i & \text{if } i \in \partial\Omega \text{ and } l = i\,, \\ \alpha_{il} \vec{n}_{il} \cdot \tilde{\partial}_{il}^{\nabla} \phi\, A_{il} & \text{elsewhere}\,, \end{cases} \tag{4.11}$$

where $\alpha_{il} = \frac{\alpha_i + \alpha_l}{2}$, $\tilde{\partial}_i^{\nabla} = (\tilde{\partial}_i^x; \tilde{\partial}_i^y; \tilde{\partial}_i^z)^T$ in 3D, and $\vec{n}_{il} \cdot \tilde{\partial}_{il}^{\nabla}$ is an approximation of the first order derivative in the direction of $\vec{n}_{il}$ and is given by

$$\vec{n}_{il} \cdot \tilde{\partial}_{il}^{\nabla} \phi = \frac{\phi_l - \phi_i}{\|\vec{x}_l - \vec{x}_i\|}\,. \tag{4.12}$$

Once again, this method can easily be extended to use more sophisticated flux functions and less diffusive approximations than Eq. (4.12). Using Eq. (4.10), when proceeding from time step t^n to t^{n+1}, the discrete diffusion operator is found by the following minimization

$$\sum_{j \in S_i} c_{ij}^D m_j = \partial_i^D m\,, \qquad \forall m \in \mathcal{M}\,, \tag{4.13}$$

$$\sum_{j \in S_i} c_{ij}^D \phi_j^{(n)} = \frac{1}{V_i} \sum_{l \in \bar{I}_i} G_{il}(\phi^{(n)}, \alpha)\,, \tag{4.14}$$

$$\min\, J_i = \sum_{j \in S_i} \left(\frac{c_{ij}^D}{W_{ij}} \right)^2\,. \tag{4.15}$$

The FC-GFDM differential operators can be used on both fixed and moving point clouds. However, their use on fixed point clouds has the disadvantage of needing differential operators to be recomputed at every time step. On the other hand, since

the traditional GFDM differential operators depend only on point locations, they can be computed in a single pre-processing step. This disadvantage is not present for moving point clouds, since changing point locations means that differential operators always need to be recomputed. We thus restrict the numerical study in the following sections to moving point clouds and Lagrangian frame of references.

While using the ideas developed in this chapter in a Lagrangian framework, we reconstruct the local dual cells at each time step. Thus, the control cells are *not* advected with the Lagrangian point cloud movement. An important point to note that using irregularly spaced point clouds and a Lagrangian framework, which requires the addition and deletion of points and interpolation for the same (see Section 2.4), can lower experimental convergence orders, and can cause deviations from expected convergence trends.

We further note that if a fixed Eulerian frame of reference is being used, it might be worth the cost to solve the global system for the non-symmetric conservative differential operators introduced in Section 3.2.2 in the pre-processing step.

4.4 Numerical Results: Advection Diffusion Equation

Consider the advection-diffusion equation in the Lagrangian formulation on a fixed domain $\Omega \subset \mathbb{R}^2$

$$\frac{D\vec{x}}{Dt} = \vec{v}\,, \tag{4.16}$$

$$\frac{D\phi}{Dt} = \nabla \cdot (\alpha \nabla \phi)\,, \tag{4.17}$$

where ϕ is the concentration or temperature, α is the diffusivity, and $\vec{v}$ is the advection velocity which is assumed to be divergence-free. Energy conservation, Eq. (3.2), leads to

$$\frac{d}{dt}\int_\Omega \phi \, dV = \int_{\partial\Omega} \vec{n} \cdot (\alpha \nabla \phi - \vec{v}\phi) \; dA\,, \tag{4.18}$$

For the FC-GFDM, the numerical diffusion operator is computed as done in Eq. (4.13) – Eq. (4.15). The spatial discretization of Eq. (4.17) is obtained as

$$\frac{D\phi_i}{Dt} = \tilde{\partial}_i^D \phi = \sum_{j \in S_i} c_{ij}^D \phi_j\,. \tag{4.19}$$

In the first step, movement of the point cloud is done in accordance with Eq. (4.16),

$$\vec{x}_i^{(n+1)} = \vec{x}_i^{(n)} + \vec{v}_i^{(n)}\Delta t + \frac{\vec{v}_i^{(n)} - \vec{v}_i^{(n-1)}}{\Delta t}(\Delta t)^2\,, \tag{4.20}$$

for each point $i = 1, 2, \ldots, N$. An in-depth discussion about this movement process is done in Chapter 6. The explicit time-integration for the movement of points results in a CFL-like condition on the time step size [66, Section 4.4.9]

$$\Delta t = C_{\Delta t} \left(\frac{h}{\|\vec{v}\|} \right)_{min}\,. \tag{4.21}$$

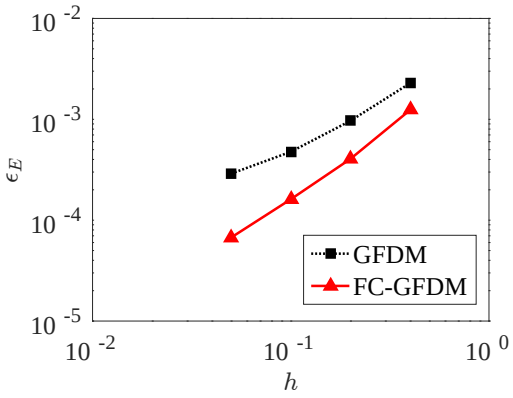


Figure 4.3: Convergence of error for the Advection Diffusion equation.

For the advection diffusion test case being considered, we use a coefficient of $C_{\Delta t} = 0.01$. The movement is followed by an implicit Euler time-integration of Eq. (4.17)

$$\frac{\phi_i^{(n+1)} - \phi_i^{(n)}}{\Delta t} = \sum_{j \in S_i} c_{ij}^D \phi_j^{(n+1)} \,, \tag{4.22}$$

The implicit system resulting from Eq. (4.22) can be solved using an iterative solver. In the results below, we use the BiCGSTAB solver [117]. Homogeneous Neumann boundary conditions are applied for ϕ. Given these boundary conditions, error in energy conservation can be measured by

$$\epsilon_E = \frac{\left| \int_\Omega \phi^{(end)} \, dV - \int_\Omega \phi^{(0)} \, dV + \int_0^{t_{end}} \left[\int_{\partial\Omega} \vec{n} \cdot \vec{v} \phi \, dA \right] \, dt \right|}{\int_\Omega \phi^{(0)} \, dV} \,, \tag{4.23}$$

where the bracketed superscript *end* denotes the values at the end of simulation, $t = t_{end}$, and the bracketed superscript 0 denotes the initial condition. The computational domain Ω is taken to be $[-2, 2] \times [-2, 2]$, and simulation is done till $t_{end} = 2\pi$. A uniform diffusion coefficient $\alpha = 0.4$ and advection field $\vec{v} = (-y, x)$ are used. Initial conditions are taken to be

$$\phi(\vec{x}, 0) = \begin{cases} 500 \,, & \text{if } \|\vec{x} - (1, 0)\|^2 < 0.1 \,, \\ 0 \,, & \text{elsewhere} \,. \end{cases} \tag{4.24}$$

The convergence of the error in energy conservation with a varying smoothing length is shown in Figure 4.3. The figure illustrates that the new flux conserving method produces much smaller errors. Both methods take similar simulation times. For an error ϵ and consecutive smoothing lengths h_1 and h_2, the rate of convergence of the solution with changing smoothing length is measured as

$$r = \frac{\log\left(\frac{\epsilon(h_2)}{\epsilon(h_1)}\right)}{\log\left(\frac{h_2}{h_1}\right)} \,. \tag{4.25}$$

The errors in energy conservation and the numerical orders of convergence of the errors are tabulated in Table 4.2.

Table 4.2: Errors and experimental convergence rates for the advection diffusion test case. h is the smoothing length, N is the number of points in the entire domain at the initial state, ϵ_E is the error in energy conservation, and r is the order of convergence of ϵ_E.

		GFDM		FC-GFDM	
h	N	ϵ_E	r	ϵ_E	r
0.4	752	2.28×10^{-3}	–	1.24×10^{-3}	–
0.2	2 778	9.66×10^{-4}	1.24	4.05×10^{-4}	1.61
0.1	10 565	4.76×10^{-4}	1.02	1.61×10^{-4}	1.33
0.05	36 224	2.89×10^{-4}	0.72	6.73×10^{-5}	1.26

4.5 Numerical Results: Incompressible Navier–Stokes Equations

Consider fluid flow modeled by the incompressible Navier–Stokes equations in Lagrangian form

$$\frac{D\vec{x}}{Dt} = \vec{v}\,, \tag{4.26}$$

$$\nabla \cdot \vec{v} = 0\,, \tag{4.27}$$

$$\frac{D\vec{v}}{Dt} = \frac{\eta}{\rho}\Delta\vec{v} - \frac{1}{\rho}\nabla p + \vec{g}\,, \tag{4.28}$$

where $\vec{v}$ is the fluid velocity, ρ is the density, η is the dynamic viscosity and $\vec{g}$ includes both gravitational acceleration and body forces. We consider conservation with respect to the velocity field, which leads to

$$\int_\Omega \nabla \cdot \vec{v}\, dV = \int_{\partial\Omega} \vec{n} \cdot \vec{v}\, dA\,, \tag{4.29}$$

$$\int_\Omega \nabla \cdot (\vec{v} \otimes \vec{v})\, dV = \int_{\partial\Omega} \vec{n} \cdot (\vec{v} \otimes \vec{v})\, dA\,, \tag{4.30}$$

$$\int_\Omega \Delta\vec{v}\, dV = \int_{\partial\Omega} \vec{n} \cdot \nabla\vec{v}\, dA\,. \tag{4.31}$$

For the FC-GFDM, the numerical Laplace operator is computed in a way similar to the numerical diffusion operator in Section 4.3.1.

$$\sum_{j\in S_i} c_{ij}^{\Delta} m_j = \partial_i^{\Delta} m\,, \qquad \forall m \in \mathcal{M}\,, \tag{4.32}$$

$$\sum_{j\in S_i} c_{ij}^{\Delta} v_j^{k,(n)} = \frac{1}{V_i} \sum_{l\in \bar{I}_i} G_{il}(v^{k,(n)}, 1)\,, \qquad k = x, y, z\,, \tag{4.33}$$

$$\min\, J_i = \sum_{j\in S_i} \left(\frac{c_{ij}^{\Delta}}{W_{ij}} \right)^2\,, \tag{4.34}$$

where the flux function G is as defined earlier and $v_j^{k,(n)}$ denotes the k-component of the velocity at point j and time-level n. The numerical first derivative approximations

are computed as

$$\sum_{j \in S_i} c_{ij}^k m_j = \partial_i^k m \,, \qquad \forall m \in \mathcal{M} \,, \tag{4.35}$$

$$\sum_{j \in S_i} c_{ij}^k v_j^{k,(n)} = \frac{1}{V_i} \sum_{l \in \bar{I}_i} F_{il}(v^{k,(n)}) \,, \tag{4.36}$$

$$\sum_{j \in S_i} c_{ij}^k v_j^{k,(n)} v_j^{k',(n)} = \frac{1}{V_i} \sum_{l \in \bar{I}_i} F_{il}(v^{k,(n)} v^{k',(n)}) \,, \qquad k' = x, y, z \,, \tag{4.37}$$

$$\min \, J_i = \sum_{j \in S_i} \left(\frac{c_{ij}^k}{W_{ij}} \right)^2 \,, \tag{4.38}$$

where the flux function F is as defined earlier. While Eq. (4.36) adds one extra constraint to the discrete differential operator definitions, Eq. (4.37) adds two or three, depending on the spatial dimension.

Using the above defined numerical differential operators, Eq. (4.26) - Eq. (4.28) are solved with a meshfree projection method, similar to that of Chorin [17]. In the first step, the point cloud is moved by solving Eq. (4.26) according to Eq. (4.20). The projection method begins with the computation of an intermediate velocity $\vec{v}^*$ by solving the conservation of momentum equation, Eq. (4.28),

$$\frac{\vec{v}^* - \vec{v}^{(n)}}{\Delta t} = \frac{\eta}{\rho} \Delta \vec{v}^* - \frac{1}{\rho} \nabla p^* + \vec{g} \,, \tag{4.39}$$

where p^* is a pressure guess found by updating the hydrostatic pressure, which is independent of the velocity [57], and depends only on gravitational acceleration and body forces. $\vec{v}^*$ is then corrected by projecting it to a divergence-free space with the help of a correction pressure,

$$\vec{v}^{(n+1)} = \vec{v}^* - \frac{\Delta t}{\rho} \nabla p_{corr} \,. \tag{4.40}$$

p_{corr} is found by applying the divergence operator to Eq. (4.40) to obtain the pressure-Poisson equation

$$\frac{\Delta t}{\rho} \Delta p_{corr} = \nabla \cdot \vec{v}^* \,. \tag{4.41}$$

The final pressure is then updated

$$p^{(n+1)} = p^* + p_{corr} \,. \tag{4.42}$$

Such meshfree projection methods have been widely used [47, 57, 110, 112]. Improvements in accuracy in such schemes for the incompressible Navier–Stokes equations are considered in Chapter 5. The spatial discretization of Eq. (4.39) – Eq. (4.41) are done using the differential operators defined earlier in this section, Eq. (4.32) – Eq. (4.34), and Eq. (4.35) – Eq. (4.38). The implicit systems obtained from Eq. (4.39) and Eq. (4.41) are solved using the BiCGSTAB iterative solver [117] without the use of any preconditioner.

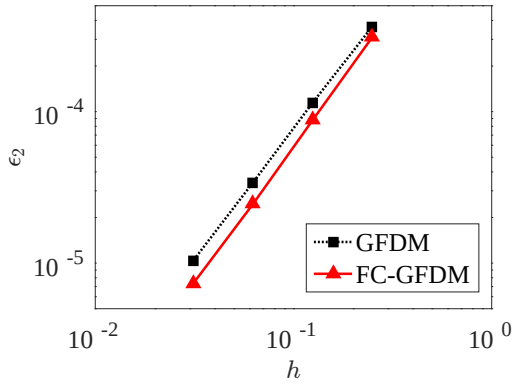


Figure 4.4: Errors for the decaying shear flow test case.

4.5.1 Decaying Shear Flow

For a validation case, we consider the case of decaying shear flow as reported in [35]. For unit density, $\rho = 1kg/m^3$, the analytical solution is given by

$$u_{\text{exact}} = 1\,, \tag{4.43}$$
$$v_{\text{exact}} = \cos(x-t)\exp(-\eta t)\,, \tag{4.44}$$
$$p_{\text{exact}} = p(t)\,, \tag{4.45}$$
$$\vec{g} = 0\,, \tag{4.46}$$

where $\vec{v}_{\text{exact}} = (u_{\text{exact}}, v_{\text{exact}})$ is the analytical solution for the velocity, and p_{exact} is the analytical solution for the pressure. The pressure is taken as $p(t) = 3.0{+}0.01\sin(20\pi t)$, as done in [35]. Initial and boundary conditions are set in accordance with the analytical solution. The computational domain is taken to be $[0,1]\times[0,1]$. The time step is determined according to Eq. (4.21) with a small coefficient of $C_{\Delta t} = 0.005$. The relative errors of the numerical solution are measured as

$$\epsilon_2 = \left[\frac{\sum \|\vec{v}_{\text{num}} - \vec{v}_{\text{exact}}\|^2}{\sum \|\vec{v}_{\text{exact}}\|^2}\right]^{\frac{1}{2}}\,, \tag{4.47}$$

where $\vec{v}_{\text{num}}$ is the numerically obtained velocity. For $\eta = 1Pa\,s$, The errors in the solutions at $t = 1s$ with and without the addition of the flux conservation condition are shown in Figure 4.4. The figure illustrates that both methods show very similar results, with FC-GFDM being slightly more accurate. The errors are also tabulated, along with numerical convergence orders according to Eq. (4.25), in Table 4.3. Experimental orders of convergence are slightly higher for the new FC-GFDM.

For "simple" domains and Dirichlet boundary conditions on all boundaries, the FC-GFDM shows only minor improvements over the classical GFDM. However, as the results in the coming sections show, FC-GFDM makes significant improvements for more complex simulation domains and boundary conditions.

Table 4.3: Errors and experimental convergence rates for the decaying shear flow test case. h is the smoothing length, N is the number of points in the entire domain at the initial state, ϵ_2 is the error in energy conservation, and r is the order of convergence of ϵ_2.

		GFDM		FC-GFDM	
h	N	ϵ_2	r	ϵ_2	r
0.25	161	3.65×10^{-4}	–	3.11×10^{-4}	–
0.125	493	1.15×10^{-4}	1.67	8.80×10^{-5}	1.82
0.0625	1 704	3.41×10^{-5}	1.75	2.46×10^{-5}	1.84
0.03125	6 293	1.04×10^{-5}	1.71	7.36×10^{-6}	1.74

4.5.2 Flow Past a Square Cylinder

We consider the flow past a square cylinder in two spatial dimensions as illustrated in Figure 4.5, which has been a widely studied problem (see, for example, [93, 100]).

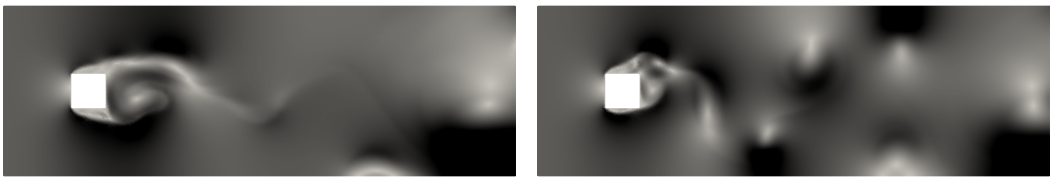


Figure 4.5: Flow past a square cylinder. Velocity profile in an inverted color scale for $Re = 10000$. $t = 23s$ (left) and $t = 34s$ (right).

The parameters of the simulations are $\rho = 1kg/m^3$ and $t_{end} = 50s$. The viscosity is given by

$$\eta = \frac{\|\vec{v}_{in}\| L}{Re}\,, \tag{4.48}$$

where $\vec{v}_{in} = (2, 0)m/s$ is the constant inflow velocity which is also used as the initial condition, $L = 30m$ is the length of the channel and Re is the Reynolds number of the flow. The height of the channel is $10m$. Homogeneous Neumann boundary conditions for the velocity are used at the outflow and no-slip conditions on the walls. The pressure is kept constant at atmospheric pressure at the outflow and Neumann boundary conditions are considered elsewhere for the pressure. A varying time step is used according to Eq. (4.21) with $C_{\Delta t} = 0.3$, which results in larger time steps than those considered in the earlier examples. The error in mass conservation is measured as the difference between the total volume of fluid flowing in and that flowing out, throughout the entire simulation, and is given by

$$\epsilon_{mass} = \left| \frac{\int_0^{t_{end}} \left[\int_{\partial\Omega_{in}} \vec{n} \cdot \vec{v}\, dA \right] dt + \int_0^{t_{end}} \left[\int_{\partial\Omega_{out}} \vec{n} \cdot \vec{v}\, dA \right] dt}{\int_0^{t_{end}} \left[\int_{\partial\Omega_{in}} \vec{n} \cdot \vec{v}\, dA \right] dt} \right| . \tag{4.49}$$

The error in a global discrete divergence theorem at the new time level is given by

$$\epsilon_{DDT} = \sum_{k=x,y,z} \frac{\left| \int_\Omega \partial^k v^{k,(n+1)}\, dV - \int_{\partial\Omega} n^k v^{k,(n+1)}\, dA \right|}{\int_\Omega dV} . \tag{4.50}$$

Table 4.4: h vs. N for flow past square cylinder.

Smoothing Length h	Total number of initial Points N
0.6	3 611
0.5	5 090
0.4	7 822
0.3	13 661

For a Reynolds number of $Re = 10^4$, the convergence with respect to the smoothing length h of ϵ_{DDT} averaged over all time steps and ϵ_{mass} for the two methods are shown in Figure 4.6. The errors in the global discrete divergence theorem for the new FC-GFDM are lower by one order of magnitude when compared to that in the case of GFDM. Correspondingly, the errors in mass conservation are also lower by one order of magnitude in the case of FC-GFDM. The total number of points in the computational domain at $t = 0s$ for the different smoothing lengths considered are shown in Table 4.4.

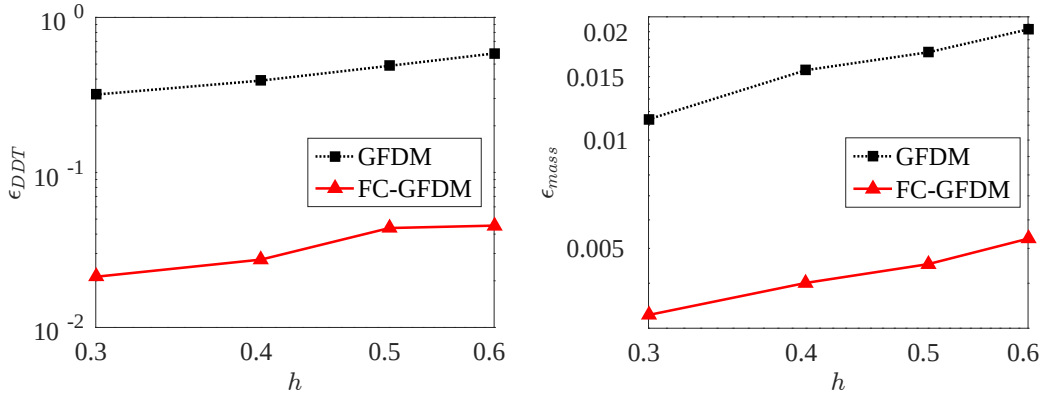


Figure 4.6: Flow past square cylinder: convergence of errors with smoothing length h on log-log plots. Error in discrete divergence theorem (left) and error in mass conservation (right). The total number of points in the computational domain for the different smoothing lengths considered are shown in Table 4.4.

A similar trend is also observed for lower Reynolds numbers of 10^3 and 500, but the errors are smaller for smaller Reynolds flow. The time evolution of the errors for $h = 0.4$ is shown in Figure 4.7. This also includes a measure for the mean velocity divergence in the domain

$$D(\vec{v}) = \frac{\int_\Omega |\nabla \cdot \vec{v}| \, dV}{\int_\Omega dV} \,. \tag{4.51}$$

Figure 4.7 illustrates that the FC-GFDM does not significantly affect the velocity divergence values, and thus, the improvement in mass conservation is solely due to smaller errors in a global discrete divergence theorem.

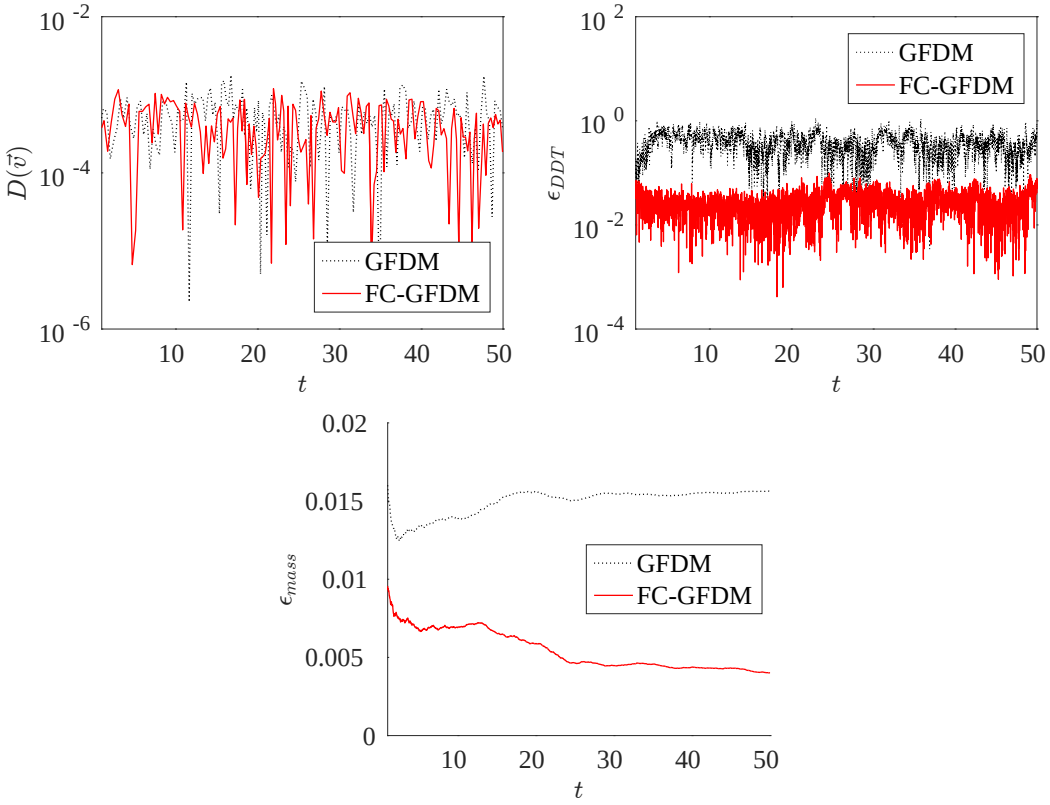


Figure 4.7: Flow past square cylinder $Re = 10000$, $h = 0.4$: Time evolution of mean velocity divergence (top left), error in discrete divergence theorem (top right) and error in mass conservation (bottom). 1 in every 40 time steps is plotted in the divergence figure, whereas all time steps are plotted in the other two figures.

4.5.3 3D Obstructed Channel Flow

The example considered in the previous section is extended to three dimensions, with multiple obstructions in the channel including both convex and concave ones. The domain considered is shown in Figure 4.8. For the simulations, we consider $\rho = 10^3 kg/m^3$, $\eta = 10^5 Pa\,s$, and $t_{end} = 1s$. The velocity at the inflow, on the left of the tube, is kept constant at $\vec{v}_{in} = (0, 0, 2)m/s$. The remaining boundary conditions are set up exactly as done in the 2D case mentioned earlier. The length of the channel is $6m$ which results in a Reynolds number in the order of magnitude of 10^{-1}. Measurement of errors are also done as in the previous section.

The convergence of ϵ_{DDT} averaged over all time steps and ϵ_{mass} for the two methods are shown with respect to a constant time step Δt in Figure 4.9. A smoothing length of $h = 0.3$ is used, which corresponds to an initial number of points $N = 20\,043$ in the entire domain. The same convergence with respect to the smoothing length h is shown in Figure 4.10. A varying time step is used according to Eq. (4.21) with $C_{\Delta t} = 0.3$. Table 4.5 shows the total number of initial points in the entire domain for the different smoothing lengths considered. Figures 4.9 and 4.10 illustrate that for a fixed h and Δt, the results produced by the new FC-GFDM exhibit a much smaller

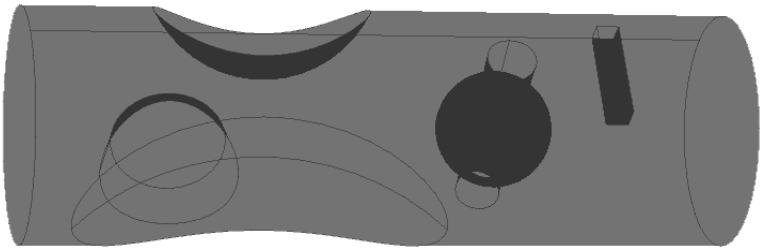


Figure 4.8: 3D Channel with multiple obstructions. Fluid inflow is on the left, and outflow on the right

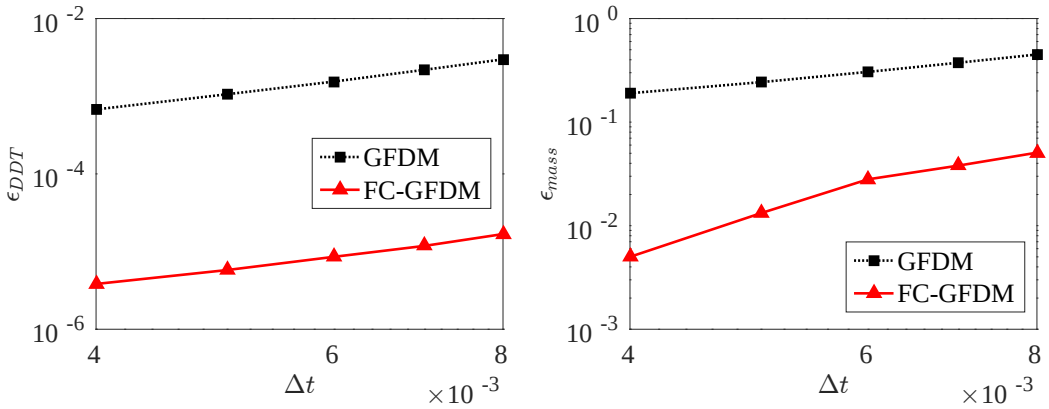


Figure 4.9: 3D obstructed channel: convergence of solution with time step Δt. Error in discrete divergence theorem (left) and error in mass conservation (right).

error in a global divergence theorem. That, in turn, results in significantly smaller errors in mass conservation. Similar to the 2D case presented earlier, the errors in FC-GFDM are lower by an order of magnitude.

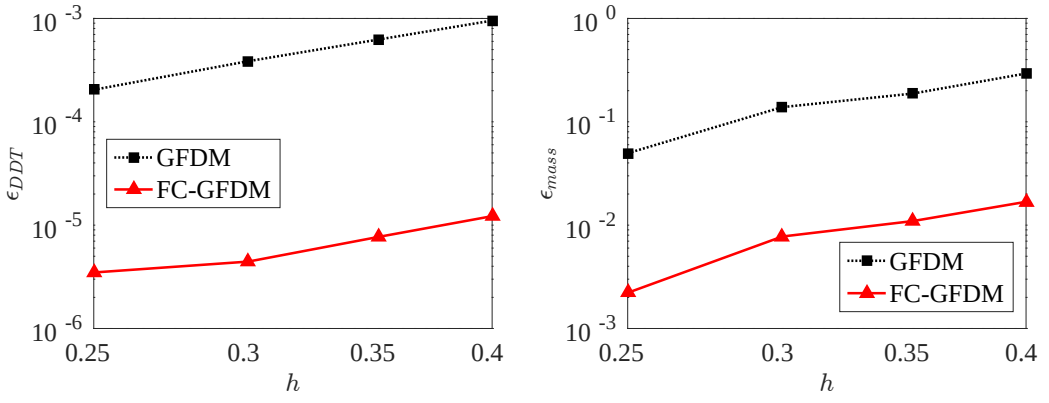


Figure 4.10: 3D obstructed channel: convergence of solution with smoothing length h. Error in discrete divergence theorem (left) and error in mass conservation (right). The total number of points in the computational domain for the different smoothing lengths considered are shown in Table 4.5.

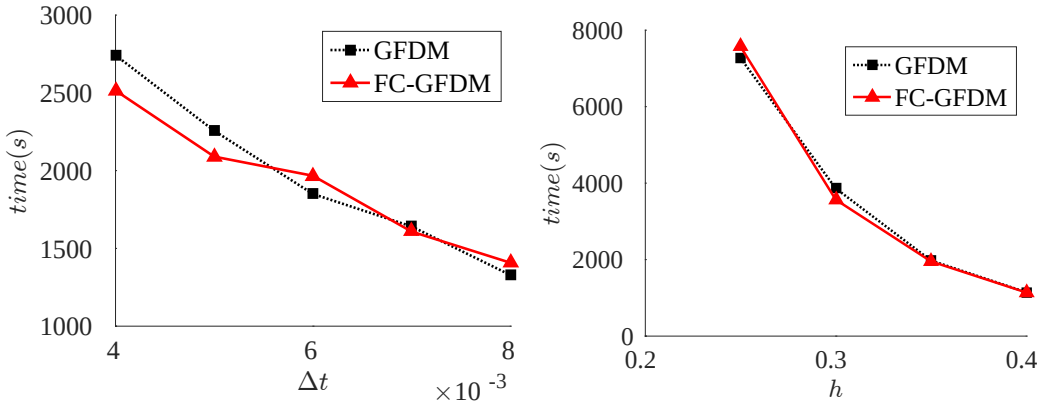


Figure 4.11: 3D obstructed channel: clock time.

Table 4.5: h vs. N for 3D channel.

Smoothing Length h	Total number of initial Points N
0.4	9 707
0.35	13 680
0.3	20 043
0.25	30 796

The time taken for the simulation of both methods are shown in Figure 4.11. The values of time taken shown in the figure represent the total time taken by each simulation, including the setup of the initial point cloud, the computation of differential operators and the solving of the large sparse linear systems at each time step, and the post-processing integrations. The simulations were carried out using Fortran and were run serially on an Intel XeonE5-2670 CPU rated at 2.60GHz. Time comparisons were done under the exact same conditions. With the exception of the differential operators, both methods use the same subroutines; while the differential operators of both GFDM and FC-GFDM use the same implementation of a QR-decomposition. Figure 4.11 illustrates that, for the same h and Δt, both methods take a similar amount of time. For almost no additional computational effort, the FC-GFDM produces much smaller errors in mass conservation. Further, Figures 4.9 – 4.11 show that the FC-GFDM takes significantly lesser time to achieve a certain tolerance of mass conservation. The addition of the flux conservation condition results in a slightly larger computation time for the differential operators of the FC-GFDM compared to those of the classical GFDM. However, as stated earlier, the effect of this on the overall simulation time is not significant. In fact, occasionally, the overall simulation time can be slightly lesser for the FC-GFDM, as illustrated in Figure 4.11. This can be explained by a possible faster convergence of the large sparse linear systems in the FC-GFDM case.

4.5.4 Sloshing

To illustrate the use of the new FC-GFDM differential operators on moving domains and problems with free surfaces, we consider the three dimensional sloshing of water in a rectangular box as shown in Figure 4.12. The dimensions of the box are $1.2m \times$

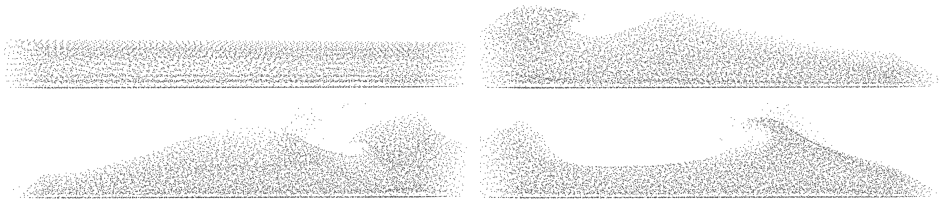


Figure 4.12: Sloshing. $t = 0s$ (top left), $t = 0.75s$ (top right), $t = 2.0s$ (bottom left) and $t = 3.2s$ (bottom right).

$0.6m \times 0.2m$. At the initial state, water occupies a cuboidal shape of dimensions $1.2m \times 0.12m \times 0.2m$.

The initial state is taken to be at rest. Slip boundary conditions are used at the walls for the velocity. Free surface boundary conditions are applied at the free boundaries, as described in Section 2.5.4, without surface tension. Neumann boundary conditions are used for the pressure at the walls. A varying time step is used according to Eq. (4.21) with $C_{\Delta t} = 0.3$. The movement of the box is represented in the gravitational and body forces term by setting $\vec{g} = (4\cos(7t), -10, 0)$. The simulation parameters are set as $\rho = 10^3 kg/m^3$, $\eta = 10^{-3} Pa\,s$, and $t_{end} = 4s$. This results in a Reynolds number of the order of magnitude of 10^7. The error in mass conservation is measured by the change in total volume occupied by all points, since the density ρ is fixed and constant throughout the domain

$$\epsilon_V = \frac{|\int_{\Omega_0} dV - \int_{\Omega} dV|}{\int_{\Omega_0} dV}, \tag{4.52}$$

where Ω_0 is the initial domain and Ω is the domain at the time when the error is measured. For an initial number of points $N = 3584$, the evolution of the error in a global divergence theorem and an error in mass conservation is shown in Figure 4.13. Similar to the earlier cases, FC-GFDM shows significantly smaller errors in a global divergence theorem. That translates to smaller errors in mass conservation which is measured using the change in total volume.

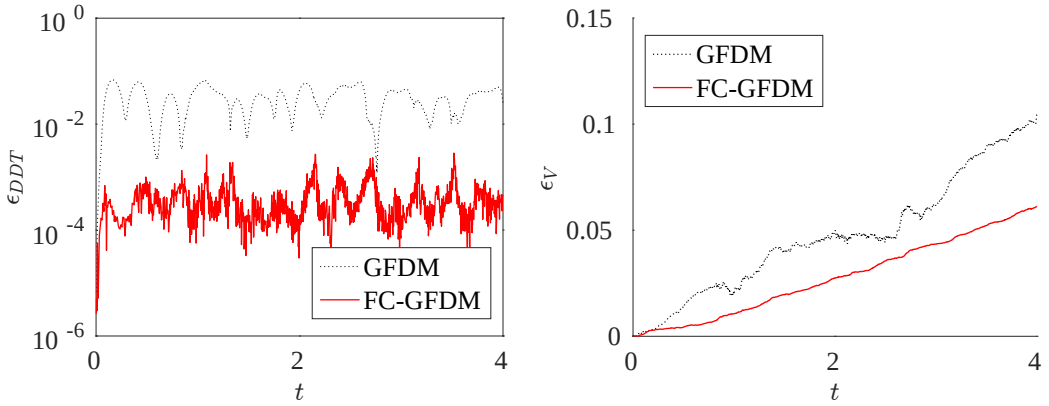


Figure 4.13: 3D Sloshing: Time evolution of error in divergence theorem (left) and error in mass/volume conservation (right).

4.6 Conclusion

We presented a novel method that combines classical moving least squares approaches to meshfree differential operators and finite-volume like flux conservation over local control cells. Implicit time-integration schemes are used to discretize the PDEs, coupled with a conservation of numerical fluxes for specific fields at the previous time-level. The locally defined control cells are easy to create automatically, and do not impose any further restrictions on the quality of the point cloud. Thus, they do not introduce the drawbacks of globally defined meshes used in mesh-based methods such as finite elements and finite volumes.

Our simulations show that the flux conserving differential operators significantly improve conservation properties of meshfree GFDMs. For the same space and time discretization, classical GFDM and the new FC-GFDM take similar simulation times, but the FC-GFDM produces smaller conservation errors.

This method can easily be extended to incorporate sophisticated flux functions. A drawback of the method is that each numerical field that needs to be conserved has to be considered individually, which can become cumbersome for large systems of PDEs. The problem of getting general conservation, in an efficient manner, via a true discrete divergence theorem for meshfree GFDMs remains open.

From a larger perspective, the ideas presented in this chapter also illustrate the potential of adding additional constraints to GFDM stencils. While this chapter presented the possibility to extend the polynomial consistency formulation of GFDM differential operators (Section 2.5.2), the next chapter presents a way to extend the direct GFDM of Section 2.6 which is based on the Taylor expansion formulation (Section 2.5.1).

Chapter 5

Meshfree GFDM and Accuracy for the Incompressible Navier–Stokes Equations

Meshfree solution schemes for the incompressible Navier–Stokes equations are usually based on algorithms commonly used in finite volume methods, such as projection methods, SIMPLE and PISO algorithms. However, drawbacks of these algorithms that are specific to meshfree methods have often been overlooked. In this chapter, we study the drawbacks of conventionally used meshfree GFDM schemes for Lagrangian incompressible Navier–Stokes equations. This includes both operator splitting schemes and monolithic schemes. The major drawback of most of these schemes is inaccurate local approximations to the mass conservation condition. Further, we propose a new modification of a commonly used monolithic scheme that overcomes these problems and shows a better approximation for the velocity divergence condition. We then perform a numerical comparison which shows the new monolithic scheme to be more accurate than existing schemes.

5.1 Introduction

Several mesh-based algorithms to solve the incompressible Navier–Stokes equations have been extended to meshfree methods. These include operator splitting methods such as the projection method [17], SIMPLE [88] and PISO [46]. However, drawbacks of these algorithms that are specific to meshfree methods have often been overlooked. One important example of such a drawback is that the discrete Laplace operator is not the same as the discrete divergence of the discrete gradient operator. This results in inaccurate approximations to the mass conservation equation. While this problem has been solved in the context of Finite Volume Methods (FVMs) with staggered grids, this problem persists in meshfree methods and introduces errors in most operator splitting algorithms.

In this chapter, we present a few commonly used meshfree solvers for the incompressible Navier–Stokes equations and study the drawbacks of these algorithms specific to meshfree GFDMs. We then propose a new monolithic solver which attempts to overcome these drawbacks. This is done by solving an over-determined problem with the direct GFDM framework presented in Section 2.6. The mass and momentum conservation equations are solved together, and simultaneously with a pressure-Poisson

equation which is needed to improve stability. All the methods considered are spatially second order accurate and use first order time integrations. However, the new monolithic method proposed here is numerically shown to be much more accurate than existing methods. This is as a consequence of the new method providing better local approximations to the mass conservation condition.

In the last chapter, we presented a method to improve global conservative properties in meshfree GFDMs, with an application to the incompressible Navier–Stokes equations. That was done by introducing an approximate discrete divergence theorem, without changing the local accuracy of the conservation equations. The example in Section 4.5.2 illustrated that improvement in global conservation properties came without improving local accuracy. In contrast, in this chapter, we present a method to improve the local accuracy of the mass conservation equation.

5.2 Existing Meshfree GFDM Algorithms for the Incompressible Navier–Stokes Equations

We consider the incompressible Navier–Stokes equations in Lagrangian form.

$$\frac{D\vec{x}}{Dt} = \vec{v}\,, \tag{5.1}$$

$$\nabla \cdot \vec{v} = 0\,, \tag{5.2}$$

$$\frac{D\vec{v}}{Dt} = \frac{\eta}{\rho}\Delta\vec{v} - \frac{1}{\rho}\nabla p + \vec{g}\,, \tag{5.3}$$

where $\vec{v}$ is the fluid velocity, p is the pressure, ρ is the density, η is the dynamic viscosity and $\vec{g}$ includes both gravitational acceleration and body forces. All the algorithms considered below start with an update of point locations by solving Eq. (5.1) according to

$$\vec{x}_i^{(n+1)} = \vec{x}_i^{(n)} + \vec{v}_i^{(n)}\Delta t + \frac{\vec{v}_i^{(n)} - \vec{v}_i^{(n-1)}}{\Delta t}(\Delta t)^2\,, \tag{5.4}$$

for each point $i = 1, \ldots, N$, where the bracketed superscript refers to the time level. An in-depth discussion about this movement process is done in Chapter 6. Following the movement of points, mass conservation Eq. (5.2) and momentum conservation Eq. (5.3) are solved according to one the methods mentioned below.

Two broad classes of algorithms are used to solve the incompressible Navier–Stokes equations. Namely, operator splitting methods, and monolithic methods. In operator splitting methods, also referred to as fractional step methods (FSM), partitioned methods, or pressure-segregation methods, the momentum conservation and mass conservation equations are solved in two separate steps. In monolithic methods, also referred to as coupled velocity-pressure methods, the mass conservation and momentum conservation equations are solved together in one large system. In the remainder of this section, we present existing meshfree methods in both these classes and study their drawbacks. In the next section, we present a new monolithic meshfree scheme which helps reduce these problems in existing solvers.

5.2.1 Operator Splitting Methods

Projection methods are a type of operator splitting methods, based on the Helmholtz-Hodge decomposition [11], which have been widely used for approximating incompressible and weakly compressible fluid flows in Finite Volume Methods [1, 13]. These methods have also been commonly used in meshfree contexts (see, for example, [112, 119]). They consist of two steps. In the first step, an intermediate velocity is computed by solving the momentum conservation equation. This intermediate velocity is then projected to a divergence-free field with the help of a correction pressure to obtain the final velocity. The intermediate velocity $\vec{v}^*$ is obtained by solving

$$\frac{\vec{v}^* - \vec{v}^{(n)}}{\Delta t} = \frac{\eta}{\rho}\Delta\vec{v}^* - \frac{1}{\rho}\nabla p^* + \vec{g}\,, \tag{5.5}$$

where p^* is a pressure guess. In the original projection method of Chorin [17], the pressure guess is taken to be zero. Non-zero values of this pressure guess have been shown to produce more accurate results [13]. Throughout this thesis, we take the pressure guess to be the pressure at the previous time step $p^* = p^{(n)}$. The step of computation of the intermediate velocity is often done explicitly, by replacing the $\Delta\vec{v}^*$ term in Eq. (5.5) with $\Delta\vec{v}^{(n)}$ [112]. However, the implicit way used in Eq. (5.5) produces more accurate, and often more stable, results. Further, the explicit treatment of the intermediate velocity would result in a CFL-condition limiting the time step by $\Delta t < \tilde{C}_{\Delta t} h^2$. i.e. the time step size is controlled by h^2 if Eq. (5.5) were to be done explicitly, while the Lagrangian movement process only controls the time step size with h.

In the second step, the velocity is corrected by projecting it to a divergence free space by

$$\vec{v}^{(n+1)} = \vec{v}^* - \frac{\Delta t}{\rho}\nabla p_{corr}\,. \tag{5.6}$$

The pressure correction p_{corr} is computed by a pressure-Poisson equation obtained by applying the divergence operator to Eq. (5.6) and setting $\nabla \cdot \vec{v}^{(n+1)} = 0$

$$\frac{\Delta t}{\rho}\Delta p_{corr} = \nabla \cdot \vec{v}^*\,. \tag{5.7}$$

Finally the pressure is updated by

$$p^{(n+1)} = p^* + p_{corr}\,. \tag{5.8}$$

Different variations of such projection methods including higher order time integration have been studied, for example, by Brown *et al.* [13]. A variety of boundary conditions have been used in these methods. A discussion on boundary conditions for projection methods in the mesh-based context can be found in Denaro [74]. In meshfree projection methods, a common approach is to obtain the boundary condition by projecting the underlying equation solved at interior points on the outward facing unit normal. Such approaches and the required stabilization can be found in Fang and Parriaux [27] and Boroomand *et al.* [12]. In this thesis, we use boundary conditions dependent on the physics of the underlying problem as done in Seibold [96]. Further, for the spatial

discretization of Eq. (5.5) and Eq. (5.7) we use the classical meshfree GFDM presented in Section 2.5, as done, for example, by Drumm *et al.* [24]. We now consider a few drawbacks of such projection methods in the meshfree context.

Consistency of numerical differential operators:

While applying the divergence operator to Eq. (5.6) to obtain Eq. (5.7), an assumption is made that the Laplace operator is the same as the divergence of the gradient operator. Specifically,

$$\nabla \cdot \nabla p_{corr} = \Delta p_{corr} \,. \tag{5.9}$$

While this is certainly true for continuous operators, it does not hold for discrete differential operators. This leads to an error in the approximation of the numerical divergence of the new velocity. Thus, mass conservation is violated at the local level. Unlike the truncation error due to the order of spatial approximation, this error does not converge to zero with a decreasing spatial discretization.

In the context of Finite Volume Methods, this problem has been known for several decades. One proposed solution was to replace the classical discrete Laplace operator with wider stencils by taking the convolution of the divergence operator and the gradient operators, i.e. setting $\Delta := \nabla \cdot \nabla$ at the discrete level [7]. Another, more widely used approach to overcome this problem is to use staggered grids in which the pressure and velocity fields are defined at different locations [36]. The staggered grid approach can not be generalized to meshfree methods since all properties, both scalars or vectors, are prescribed on the same nodes. Using the first approach of setting the numerical Laplace operator to be equal to the numerical divergence of the numerical gradient causes several issues in the meshfree context. Firstly, this would lead to different support sizes for the first and second derivatives. The second derivatives would be defined on a support double the size of that of the first derivative, which more than doubles the number of points in each support domain. Moreover, there is no control over the center stencil values, which makes diagonal dominance hard to achieve (see Section 2.5.5). As a result, convergence of the large linear systems can be troublesome. Thus, this problem still persists in meshfree projection methods. Not only for meshfree GFDMs, but also in various other meshfree approximation methods including SPH [19, 119].

Other operator splitting algorithms such as the PISO algorithm [46], SIMPLE algorithms [88] and their derivatives also rely on pressure-Poisson equations. The difference being that the Poisson equations are derived by applying the divergence operator to the momentum conservation equation. Their meshfree equivalents possess the same drawback of $\Delta \neq \nabla \cdot \nabla$ at the discrete level, which leads to the same inaccuracies in the approximation of the mass conservation condition.

Compressible boundary layer:

When using the classical GFDM approach of Section 2.5 the pressure-Poisson equation Eq. (5.7) is solved at interior points with appropriate boundary conditions on the boundary points. The divergence of the velocity at boundary points depends on the pressure-Poisson equation at boundary points which is not solved at all. This results

in the formation of a numerical boundary layer of compressible fluid, with non-zero divergence of velocity, during the simulation of incompressible fluids. This problem has been alleviated by the direct GFDM presented in Section 2.6, by solving an over-determined system at boundary points, which considers both the relevant boundary conditions and the pressure-Poisson equation. This problem of numerical boundary layers is also avoided by several solvers that solve the pressure-Poisson system before the implicit velocity system [122]. More recently, alternate solutions to this same problem were also considered by Idelsohn and Oñate [42], but specific to free surface boundaries. However, the earlier issue of $\Delta \neq \nabla \cdot \nabla$ at the discrete level is present even at the boundaries, and is often a larger source of error than the compressible boundary layer.

Poor accuracy at low Reynolds flow:

Projection methods usually suffer from low accuracy, and often instability, for fluid flows at low Reynolds numbers ($Re \ll 1$). This problem is reduced, but not overcome, by the implicit nature of Eq. (5.5). As a result, projection methods do not provide good approximations for fluids with very high viscosity such as molten glass [77]. This can be seen upon a closer look at the projection scheme. Adding Eq. (5.5) and Eq. (5.6), and then using Eq. (5.8) to gather the pressure terms, we get

$$\frac{\vec{v}^{(n+1)} - \vec{v}^{(n)}}{\Delta t} = \frac{\eta}{\rho}\Delta\vec{v}^{*} - \frac{1}{\rho}\nabla p^{(n+1)} + \vec{g}\,. \tag{5.10}$$

On the other hand, a first order discretization of the momentum equation, which is desired here, would result in a $\Delta\vec{v}^{(n+1)}$ term on the RHS. The additional error term in this regard, $\frac{\eta}{\rho}\|\vec{v}^{*} - \vec{v}^{(n+1)}\|$ scales with $\frac{1}{Re}$.

5.2.2 Monolithic Methods

Coupled velocity-pressure solvers usually have the advantage of being more stable, especially for larger time step sizes, but often come at the disadvantage of ill-conditioned systems. To avoid the problem of ill-conditioned systems, Kuhnert [57] developed a penalty formulation based on the classical GFDM approach of Section 2.5. In two spatial dimensions, for $\vec{v} = (u, v)$ and $\vec{g} = (g_x, g_y)$, it can be written in matrix form as

$$\begin{pmatrix} I - \frac{\Delta t}{\rho}\eta\mathbf{C}^{\Delta} & & \frac{\Delta t}{\rho}\mathbf{C}^{x} \\ & I - \frac{\Delta t}{\rho}\eta\mathbf{C}^{\Delta} & \frac{\Delta t}{\rho}\mathbf{C}^{y} \\ \mathbf{C}^{x} & \mathbf{C}^{y} & -A\frac{\Delta t}{\rho}\mathbf{C}^{\Delta} \end{pmatrix} \begin{pmatrix} \vec{U}^{\,(n+1)} \\ \vec{V}^{\,(n+1)} \\ \vec{P}_{corr} \end{pmatrix} = \begin{pmatrix} \vec{U}^{\,(n)} - \frac{\Delta t}{\rho}\mathbf{C}^{x}\vec{P}^{*} + \Delta t\vec{G}_x \\ \vec{V}^{\,(n)} - \frac{\Delta t}{\rho}\mathbf{C}^{y}\vec{P}^{*} + \Delta t\vec{G}_y \\ 0 \end{pmatrix}, \tag{5.11}$$

where $\mathbf{C}^{x}$ is the matrix formed by the stencil coefficients for the numerical differential operators for the x derivative, c_{ij}^{x} (see Eq. (2.11) and Eq. (2.17)). Similarly for $\mathbf{C}^{y}$ and $\mathbf{C}^{\Delta}$. The vector $\vec{U}^{\,(n+1)}$ is formed by $u^{(n+1)}$ at all points in the computational domain, $\vec{U}^{\,(n+1)} = (u_1^{(n+1)}, \ldots, u_N^{(n+1)})^T$ and similarly for the other upper case vectors. Thus, the first two blocks of rows in Eq. (5.11) represents the momentum conservation

equation at all points. The last block of rows in Eq. (5.11) is a penalty formulation for the conservation of mass equation

$$\nabla \cdot \vec{v}^{(n+1)} - A\frac{\Delta t}{\rho}\Delta p_{corr} = 0\,. \tag{5.12}$$

Rows corresponding to boundary points in Eq. (5.11) are replaced with the appropriate problem specific boundary conditions. The ideal scenario would be to set $A = 0$, to get an exact conservation of mass. However, that leads to an ill-conditioned system. Further, typical iterative solvers do not converge for such a system due to a complete lack of diagonal dominance in the last block of rows in Eq. (5.11). While specialized solvers for saddle-point problems [10, 75], could possibly be used to obtain solutions to these sparse linear systems, simulations are still usually unstable when $A = 0$. Thus, non-zero values of A need to be used. Kuhnert [57] takes this parameter to be in the range of $A \in (0, 0.3)$. Lower values of A are preferred for greater accuracy, however, higher values are needed for better conditioning and faster convergence of the resulting linear system. We note that using $A = 1$ would result in an implicit and coupled projection method, with an additional velocity correction step, Eq. (5.6), required. The final pressure is given as done before $p^{(n+1)} = p^* + p_{corr}$.

This penalty approach has been shown to be more stable than the meshfree projection method [47, 57], especially at low Reynolds flows. However, setting $A \neq 0$ in Eq. (5.12) leads to an artificial compressibility that is numerically observed to be similar to that introduced by the inconsistency between the Δ and the $\nabla \cdot \nabla$ discrete operators in operator splitting methods. Further, this approach has the same issue of a compressible boundary layer as that in classical operator splitting methods explained earlier.

This penalty approach coupled solver and the meshfree projection method have both been widely used and have shown to be robust methods with a wide variety of applications. Under the name of the Finite Pointset Method (FPM), they have also been used as the numerical basis of two commercially used meshfree simulation tools: NOGRID [77] and the meshfree module of VPS-PAMCRASH [113].

Other approaches to coupled solvers include solving the momentum conservation as in the first two blocks of rows in Eq. (5.11), with a pressure-Poisson equation replacing the third block of rows in Eq. (5.11). Like most fractional step methods, this solves the mass conservation indirectly, and introduces the same error as mentioned earlier of $\Delta \neq \nabla \cdot \nabla$ at the discrete level. In mesh-based contexts, especially for Finite Element Methods, the equivalent systems of Eq. (5.11) with $A = 0$ are often solved by algebraically decomposing the system which is essentially equivalent to a fractional step method [104]; or with the help of stability conditions [2] which introduce errors similar to that in the method described in this section.

5.3 A New Monolithic Solver

We wish to solve the momentum and mass conservation equations as

$$\frac{\vec{v}^{(n+1)} - \vec{v}^{(n)}}{\Delta t} = \frac{\eta}{\rho}\Delta\vec{v}^{(n+1)} - \frac{1}{\rho}\nabla p^* - \frac{1}{\rho}\nabla p_{corr} + \vec{g}\,, \tag{5.13}$$

$$\nabla \cdot \vec{v}^{(n+1)} = 0\,. \tag{5.14}$$

We emphasize that the desire is to solve the mass conservation directly as in Eq. (5.14) and not indirectly via a pressure-Poisson equation. Directly solving Eq. (5.14) ensures that errors arising from the inconsistency of the discrete Laplace operator and the discrete divergence of the discrete gradient do not affect the mass conservation condition.

Using the classical GFDM approach of Section 2.5, Eq. (5.13) and Eq. (5.14) lead to Eq. (5.11) with $A = 0$. As mentioned earlier, this results in an ill-conditioned system that is very hard to solve with typical iterative procedures. This problem of ill-conditioned systems can be avoided by using the direct GFDM presented in Section 2.6. For the same, we start by rewriting Eq. (5.13) and Eq. (5.14), in two spatial dimensions, to obtain

$$e_1 + u^{(n+1)} - \frac{\eta\Delta t}{\rho}\Delta u^{(n+1)} + \frac{\Delta t}{\rho}\partial^x p_{corr} = u^{(n)} - \frac{\Delta t}{\rho}\partial^x p^* + \Delta t g_x\,, \tag{5.15}$$

$$e_2 + v^{(n+1)} - \frac{\eta\Delta t}{\rho}\Delta v^{(n+1)} + \frac{\Delta t}{\rho}\partial^y p_{corr} = v^{(n)} - \frac{\Delta t}{\rho}\partial^y p^* + \Delta t g_y\,, \tag{5.16}$$

$$e_3 + \nabla \cdot \vec{v}^{(n+1)} = 0\,, \tag{5.17}$$

where e_1, e_2 and e_3 are the errors in the discretizations of the respective PDEs. A quadratic minimization of e_1, e_2 and e_3, along with the errors from the Taylor expansions in all three variables, u, v and p_{corr} can be done, similar to that done in Section 2.6. This results in a system of equations similar to Eq. (2.66) and provides better conditioned systems than Eq. (5.11) with $A = 0$. However, resulting simulations are mostly unstable, as the simulation blows up within a couple of time steps. Thus, such an approach of using the direct GFDM framework of discretization to coupled solvers has not been used successfully in the past.

A possible explanation of the instability is the lack of information about the pressure correction p_{corr}. To correct this and to introduce a further coupling condition between the velocity and pressure, we make use of the fact that this direct GFDM framework can be used to solve algebraically over-determined problems, as presented in Section 2.6.1. The system of Eq. (5.15) – Eq. (5.17) are augmented with a pressure-Poisson equation to form an over-determined system. The Poisson equation is obtained in a manner similar to that done by SIMPLE and PISO methods, by applying the divergence operator to the conservation of momentum equation Eq. (5.3). However, a slight variation is used as follows. Taking the divergence of the continuous momentum equation gives

$$\nabla \cdot \left[\frac{D\vec{v}}{Dt}\right] = \nabla \cdot \left[\frac{\eta}{\rho}\Delta\vec{v} - \frac{1}{\rho}\nabla p + \vec{g}\right]\,, \tag{5.18}$$

$$= \frac{\eta}{\rho}\Delta\nabla\cdot\vec{v} - \frac{1}{\rho}\Delta p + \nabla\cdot\vec{g}\,. \tag{5.19}$$

To discretize the LHS, we do the following

$$\begin{aligned}\nabla\cdot\left(\frac{D\vec{v}}{Dt}\right) &= \nabla\cdot\left(\frac{D\vec{v}}{Dt}\right) - \frac{D}{Dt}\left(\nabla\cdot\vec{v}\right) + \frac{D}{Dt}\left(\nabla\cdot\vec{v}\right),\\ &= \left[\nabla\cdot\left(\frac{\partial\vec{v}}{\partial t} + \left(\vec{v}\cdot\nabla\right)\vec{v}\right)\right] - \left[\frac{\partial}{\partial t}\left(\nabla\cdot\vec{v}\right) + \left(\vec{v}\cdot\nabla\right)\left(\nabla\cdot\vec{v}\right)\right] + \frac{D}{Dt}\left(\nabla\cdot\vec{v}\right),\\ &= \nabla\cdot\left[\left(\vec{v}\cdot\nabla\right)\vec{v}\right] - \left(\vec{v}\cdot\nabla\right)\left(\nabla\cdot\vec{v}\right) + \frac{D}{Dt}\left(\nabla\cdot\vec{v}\right). \qquad (5.20)\end{aligned}$$

Using this, Eq. (5.19) is discretized as follows

$$\begin{aligned}\nabla\cdot\left[\left(\vec{v}^{(n)}\cdot\nabla\right)\vec{v}^{(n+1)}\right] - \left(\vec{v}^{(n+1)}\cdot\nabla\right)\left(\nabla\cdot\vec{v}^{(n+1)}\right) + \frac{\left(\nabla\cdot\vec{v}\right)^{(n+1)} - \left(\nabla\cdot\vec{v}\right)^{(n)}}{\Delta t}\\ = \frac{\eta}{\rho}\Delta\nabla\cdot\vec{v}^{(n+1)} - \frac{1}{\rho}\Delta p^{(n+1)} + \nabla\cdot\vec{g}\,. \qquad (5.21)\end{aligned}$$

Splitting the pressure, setting $\nabla\cdot\vec{v}^{(n+1)} = 0$ and rearranging leads to the pressure-Poisson equation which will be used

$$e_4 + \Delta p_{corr} + \rho\nabla\cdot\left[\left(\vec{v}^{(n)}\cdot\nabla\right)\vec{v}^{(n+1)}\right] = \frac{\rho}{\Delta t}\left(\nabla\cdot\vec{v}\right)^{(n)} - \Delta p^{*} + \rho\nabla\cdot\vec{g}\,. \qquad (5.22)$$

Note that $\left(\nabla\cdot\vec{v}\right)^{(n)}$ is not the discrete divergence operator applied to $\vec{v}^{(n)}$. It is the divergence operator of the previous time step applied to $\vec{v}^{(n)}$. Since the numerical divergence operator of the previous time step is not available on the present point cloud, the value of $\left(\nabla\cdot\vec{v}\right)^{(n)}$ needs to be stored. The formulation of the pressure Poisson equation used here has the advantage that it penalizes incorrect values of velocity divergence from the previous time level.

In the overall scheme, Eq. (5.15) – Eq. (5.17) and Eq. (5.22) are solved at all interior points. Note that the mass conservation condition is solved directly in Eq. (5.17) and indirectly in Eq. (5.22). The introduction of the pressure-Poisson equation brings in the problem of $\Delta \neq \nabla\cdot\nabla$ at the discrete level, as mentioned earlier. However, directly solving for the conservation of mass in Eq. (5.17) ensures that these errors do not affect the numerical accuracy of the zero divergence of velocity condition.

At boundary points, the mass conservation condition Eq. (5.17) is solved in addition to the relevant boundary conditions, once again leading to more PDEs than variables. The errors in each of these PDEs or boundary conditions are minimized simultaneously with the errors in the Taylor expansions for each velocity component and the pressure.

$$\min\, J_i = \sum_{j\in S_i} W_{ij}^2 (e_{ij}^u)^2 + \sum_{j\in S_i} W_{ij}^2 (e_{ij}^v)^2 + \sum_{j\in S_i} W_{ij}^2 (e_{ij}^p)^2 + \sum_{k=1}^{4} W_{\mathrm{PDE},k}^2 (e_k)^2\,, \qquad (5.23)$$

where e_{ij}^u, e_{ij}^v and e_{ij}^p are the errors in the Taylor expansions around point i of u, v and p_{corr} respectively. For all simulations to follow, W_{ij} are taken as in Eq. (2.7), and $W_{\mathrm{PDE},k}^2 = 2$. Thus, the following least squares system is solved at each interior point to obtain the function approximation stencil coefficients : $\vec{E_i} = M_i\vec{a}_i - \vec{b_i}$ with

$$\vec{E_i} = \left(e_{i1}^u, \dots\, e_{in}^u, e_{i1}^v, \dots\, e_{in}^v, e_{i1}^p, \dots\, e_{in}^p, e_1, \dots, e_4\right)^T, \qquad (5.24)$$

$$\vec{a}_i = \left(u, u_x, u_y, u_{xx}, u_{yy}, u_{xy}, v, v_x, v_y, v_{xx}, v_{yy}, v_{xy}, p, p_x, p_y, p_{xx}, p_{yy}, p_{xy}\right)^T, \qquad (5.25)$$

$$\vec{b_i} = \left(u_1, \dots\, u_n, v_1, \dots\, v_n, p_1, \dots\, p_n, r_1, \dots\, r_4\right)^T, \qquad (5.26)$$

where p is used as shorthand for p_{corr}, and r_k are the right hand sides of equations Eq. (5.15) – Eq. (5.17) and Eq. (5.22), and each value of $\vec{a}_i$ is at point i. Further,

$$M_i = \begin{pmatrix} M_T & & \\ & M_T & \\ & & M_T \\ M_{P1} & M_{P2} & M_{P3} \end{pmatrix}, \tag{5.27}$$

with M_T being the parts coming from the Taylor expansions

$$M_T = \begin{pmatrix} 1 & \delta x_{i1} & \delta y_{i1} & \frac{1}{2}\delta x_{i1}^2 & \frac{1}{2}\delta y_{i1}^2 & \delta x_{i1}\delta y_{i1} \\ & & \vdots & \vdots & & \\ 1 & \delta x_{in} & \delta y_{in} & \frac{1}{2}\delta x_{in}^2 & \frac{1}{2}\delta y_{in}^2 & \delta x_{in}\delta y_{in} \end{pmatrix}, \tag{5.28}$$

and the part coming from the PDEs is given by

$$\left(\begin{array}{c|c|c} M_{P1} & M_{P2} & M_{P3} \end{array} \right) =$$
$$\left(\begin{array}{cccccc|cccccc|cccccc} 1 & 0 & 0 & -\frac{\eta\Delta t}{\rho} & -\frac{\eta\Delta t}{\rho} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\Delta t}{\rho} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -\frac{\eta\Delta t}{\rho} & -\frac{\eta\Delta t}{\rho} & 0 & 0 & 0 & \frac{\Delta t}{\rho} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \rho u_x^{(n)} & \rho v_x^{(n)} & 0 & 0 & 0 & 0 & \rho u_y^{(n)} & \rho v_y^{(n)} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array} \right). \tag{5.29}$$

The first two rows in Eq. (5.29) represent the momentum conservation in x and y directions respectively, the third row represents the mass conservation equation, and the last row represents the pressure-Poisson equation. Similar systems are obtained for boundary points, with the relevant rows in Eq. (5.29) replaced with the boundary conditions, as explained in Section 2.6.2. As done earlier, a formal minimization leads to $\vec{a}_i = [(M_i^T W_i^2 M_i)^{-1} M_i^T W_i^2]\vec{b}_i$. Only the function approximation stencils are of interest to us, i.e. the rows of u, v and p_{corr}. They can be written as

$$u_i = \sum_{j\in S_i} \alpha_{ij}^u u_j + \sum_{j\in S_i} \beta_{ij}^u v_j + \sum_{j\in S_i} \gamma_{ij}^u (p_{corr})_j + \sum_{k=1}^{4} \zeta_{ik}^u r_k \,, \tag{5.30}$$

$$v_i = \sum_{j\in S_i} \alpha_{ij}^v u_j + \sum_{j\in S_i} \beta_{ij}^v v_j + \sum_{j\in S_i} \gamma_{ij}^v (p_{corr})_j + \sum_{k=1}^{4} \zeta_{ik}^v r_k \,, \tag{5.31}$$

$$(p_{corr})_i = \sum_{j\in S_i} \alpha_{ij}^p u_j + \sum_{j\in S_i} \beta_{ij}^p v_j + \sum_{j\in S_i} \gamma_{ij}^p (p_{corr})_j + \sum_{k=1}^{4} \zeta_{ik}^p r_k \,, \tag{5.32}$$

for $i = 1, \ldots, N$. The coefficients α, β, γ and ζ represent the values in the relevant row of the matrix $[(M_i^T W_i^2 M_i)^{-1} M_i^T W_i^2]$. These can be rearranged to obtain the sparse

linear system

$$(1-\alpha_{ii}^{u})u_i - \sum_{\substack{j\in S_i\\ j\neq i}} \alpha_{ij}^{u}u_j - \sum_{j\in S_i} \beta_{ij}^{u}v_j - \sum_{j\in S_i} \gamma_{ij}^{u}(p_{corr})_j = \sum_{k=1}^{4} \zeta_{ik}^{u} r_k \,, \tag{5.33}$$

$$-\sum_{j\in S_i} \alpha_{ij}^{v}u_j + (1-\beta_{ii}^{v})v_i - \sum_{\substack{j\in S_i\\ j\neq i}} \beta_{ij}^{v}v_j - \sum_{j\in S_i} \gamma_{ij}^{v}(p_{corr})_j = \sum_{k=1}^{4} \zeta_{ik}^{v} r_k \,, \tag{5.34}$$

$$-\sum_{j\in S_i} \alpha_{ij}^{p}u_j - \sum_{j\in S_i} \beta_{ij}^{p}v_j + (1-\gamma_{ii}^{p})(p_{corr})_i - \sum_{\substack{j\in S_i\\ j\neq i}} \gamma_{ij}^{p}(p_{corr})_j = \sum_{k=1}^{4} \zeta_{ik}^{p} r_k \,. \tag{5.35}$$

This resulting sparse linear system is solved using an iterative solver. Eq. (5.33) – Eq. (5.35) produce a diagonally dominant system. In Eq. (5.33), for example, the magnitude of the diagonal values of $1-\alpha_{ii}^{u}$ are significantly larger than that of the off-diagonal values of α_{ij}^{u} $j \neq i$, β_{ij}^{u} and γ_{ij}^{u}. Thus, the resulting linear system converges well with typical iterative procedures.

A short comparison between the meshfree projection method presented in Section 5.2.1, the penalty approach coupled solver of Section 5.2.2 and the new coupled solver presented in this section is listed below.

- All three methods have the same theoretical convergence rate with the spatial discretization as all use Taylor expansions up to the same order of accuracy. All three methods also use similar first order temporal discretizations. Higher order methods have been studied extensively, especially for mesh-based fractional step methods (for example, [120]). Such higher order approximations could be applied to all three methods considered here.

- All three methods are "approximate" methods as opposed to "exact" methods, in the sense that they only solve the mass conservation equation up to the truncation error. However, the new method does not contain the additional sources of error present in the other two methods. The projection method solves the mass conservation indirectly which leads to errors due to a lack of consistency between the first and second order derivatives. The penalty approach coupled solver attempts to solve the mass conservation directly, but introduces an artificial compressibility to improve conditioning of the resulting system. On the other hand, the new coupled solver solves the mass conservation equation directly and without introducing an artificial compressibility, and thus provides a much better approximation to the mass conservation equation than both other methods.

- Unlike the penalty approach coupled solver, the new coupled solver avoids the saddle point structure[1] of monolithic systems without introducing any extra errors. The resulting system from the new coupled solver of Eq. (5.33) – Eq. (5.35) can be written in matrix form as

$$\begin{pmatrix} I-\boldsymbol{\alpha}^{u} & -\boldsymbol{\beta}^{u} & -\boldsymbol{\gamma}^{u} \\ -\boldsymbol{\alpha}^{v} & I-\boldsymbol{\beta}^{v} & -\boldsymbol{\gamma}^{v} \\ -\boldsymbol{\alpha}^{p} & -\boldsymbol{\beta}^{p} & I-\boldsymbol{\gamma}^{p} \end{pmatrix} \begin{pmatrix} \vec{U}^{(n+1)} \\ \vec{V}^{(n+1)} \\ \vec{P}_{corr} \end{pmatrix} = \begin{pmatrix} \vec{R}_1 \\ \vec{R}_2 \\ \vec{R}_3 \end{pmatrix} , \tag{5.36}$$

where $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are matrices formed from the coefficients in Eq. (5.33) – Eq. (5.35) and $\vec{R}_1$, $\vec{R}_2$ and $\vec{R}_3$ are vectors formed from the right hand sides of Eq. (5.33) – Eq. (5.35) respectively. On the other hand, the ideal case of traditional monolithic solvers would be Eq. (5.11) with $A = 0$, which results in

$$\begin{pmatrix} I - \frac{\Delta t}{\rho}\eta\mathbf{C}^{\Delta} & & \frac{\Delta t}{\rho}\mathbf{C}^{x} \\ & I - \frac{\Delta t}{\rho}\eta\mathbf{C}^{\Delta} & \frac{\Delta t}{\rho}\mathbf{C}^{y} \\ \mathbf{C}^{x} & \mathbf{C}^{y} & \end{pmatrix} \begin{pmatrix} \vec{U}^{(n+1)} \\ \vec{V}^{(n+1)} \\ \vec{P}_{corr} \end{pmatrix} = \begin{pmatrix} \vec{R}_4 \\ \vec{R}_5 \\ \vec{R}_6 \end{pmatrix}, \tag{5.37}$$

where $\vec{R}_4$, $\vec{R}_5$ and $\vec{R}_6$ are vectors formed from the right hand side of Eq. (5.11). The 0 diagonal block in the last block of rows makes Eq. (5.37) hard to solve with typical iterative procedures. This problem is avoided altogether in the new coupled solver in Eq. (5.36), as the sparse linear system is constructed with the function approximation stencils, and not the derivative operators. On the other hand, the penalty approach coupled solver introduces and artificial compressibility (last block of rows in Eq. (5.11)) to overcome this issue.

- Both coupled solvers solve one large implicit linear system while the projection method solves two smaller implicit linear systems. In both coupled solvers, the linear systems are of the same size, but the system is denser in the new coupled solver. While the sparsity pattern of each block of rows in Eq. (5.36) and Eq. (5.11) are identical, as they are dependent only on the support domains for each point, several zero blocks are present in Eq. (5.11), while all blocks are non-zero in Eq. (5.36). This is no longer true for cases of spatially varying viscosity η, which are very common in fluid flow applications, for which all blocks are non-zero in the systems arising from both the coupled solvers (due the presence of the extra $\nabla\eta \cdot \nabla\vec{v}$ terms).

- While the penalty approach coupled solver and the projection method based on the classical GFDM result in a compressible boundary layer, this is not present in the new coupled solver presented here, due to the addition of the mass balance equation on boundary points.

- Numerically it is observed that the new coupled solver has stability comparable to the penalty approach coupled solver, and thus, much better than the projection method which has a higher tendency of the solution to blow up, especially for low Reynolds flow.

- The larger size of the implicit linear systems means that both coupled solvers have higher memory requirements than the projection method.

- The fastest simulation times between the three methods vary on a case by case basis, but the overall simulation time is similar for all three methods.

[1]Here, we only refer to the structure of the resulting sparse linear system that needs to be solved.

5.4 Numerical Results

The explicit time-integration for the movement of points according to Eq. (5.4) results in a CFL-like condition on the time step size [66, Section 4.4.9]

$$\Delta t = C_{\Delta t} \left(\frac{h}{\|\vec{v}\|} \right)_{min} . \tag{5.38}$$

Thus, a varying time step size according to Eq. (5.38) is used in all performed simulations. To determine the numerical order of accuracy of the three methods, a small time step is used in the first test case below by using a small value of $C_{\Delta t}$. The remaining test cases use a much larger time step which results in lower experimental convergence rates. In the comparison of simulation times, the total time of the simulation refers to the total clock time (in seconds) of the simulation, including the initial point cloud setup and the post-processing error calculations. To ensure that these comparisons are realistic, all three methods were implemented by the same programmer and use identical memory management. Further, for the sake of consistency and a fair comparison, the imposed boundary conditions are the same in all three schemes. All simulations were carried out in Fortran and were run serially on an Intel XeonE5-2670 CPU rated at 2.60GHz. All sparse linear systems are solved using the BiCGSTAB iterative solver [117] without the use of any preconditioner. For the penalty approach coupled solver, the penalty coefficient is taken to be $A = 0.1$ in all simulations, since lower values result in much larger simulation times due to poor convergence of the linear systems.

In each of the examples to follow, for an error ϵ, the numerical rate of convergence of the solution with changing smoothing length is measured as

$$r = \frac{\log\left(\frac{\epsilon(h_2)}{\epsilon(h_1)}\right)}{\log\left(\frac{h_2}{h_1}\right)}, \tag{5.39}$$

where h_1 and h_2 are consecutive smoothing lengths considered. Further, we, once again, note that irregularly space point clouds, and the addition and deletion of points in the Lagrangian framework and the required interpolation therein (see Section 2.4) can lower convergence orders, and can cause slight deviations from expected convergence trends.

In all the figures below, the projection method presented in Section 5.2.1 is referred to by 'Projection', the coupled solver with the penalty formulation presented in Section 5.2.2 is referred to by 'Coupled: Penalty' and the new coupled solver which directly solves an algebraically over-determined system, as done in Section 5.3, is referred to as 'Coupled: New'.

5.4.1 Taylor-Green Vortices

As a validation case, we consider the two-dimensional decaying vortices referred to as the Taylor-Green vortices on $[0, 2\pi] \times [0, 2\pi]$. The analytical solution is given by

$$u_a = \sin(x)\cos(y)\exp(-4\pi t/Re)\,, \tag{5.40}$$
$$v_a = -\cos(x)\sin(y)\exp(-4\pi t/Re)\,, \tag{5.41}$$
$$p_a = \frac{\rho}{4}(\cos(2x)+\cos(2y))\exp(-8\pi t/Re)\,, \tag{5.42}$$
$$\vec{g} = 0\,, \tag{5.43}$$

where $\vec{v}_a = (u_a, v_a)$ is the analytical solution for the velocity, p_a is the analytical solution for the pressure, and $Re = \frac{\rho U L}{\eta}$ is the Reynolds number, with the characteristic velocity $U = 1$ and the characteristic length $L = 2\pi$. Error in the numerical solution $\vec{v}$ is measured by

$$\epsilon_2 = \left[\frac{\sum_{i=1}^{N} \|\vec{v}_i - \vec{v}_{\mathrm{a}}(\vec{x}_i)\|^2 V_i}{\sum_{i=1}^{N} \|\vec{v}_{\mathrm{a}}(\vec{x}_i)\|^2 V_i}\right]^{\frac{1}{2}}\,, \tag{5.44}$$

where V_i is a post-processing volume associated with point i (see Sections 4.1 and 4.2).

The initial condition for the velocity is taken in accordance with the exact solution, and is shown in Figure 5.1. Dirichlet boundary conditions are used on all boundaries. The simulations are done up to an ending time of $t_{end} = 1s$, for $\eta = 1Pa\,s$ and $\rho = 1kg/m^3$, which gives $Re = 2\pi$. A small time step is used according to Eq. (5.38) with $C_{\Delta t} = 0.005$. The convergence of the errors with the smoothing length h are shown in Figure 5.2 and are tabulated in Table 5.1. All three methods match the analytical solution well. All three methods exhibit a similar convergence rate, which matches the theoretical expectation. Errors in the new coupled solver are smaller than the other two methods, while the errors in the projection method are the largest, but are only slightly larger than those in the penalty approach coupled solver. Total simulation times for each case are also shown in Table 5.1. All three methods take approximately the same time, with the new coupled solver being the slowest and the projection method being the fastest. Similar results were obtained for larger Reynolds numbers.

Table 5.1: Errors, convergence orders and simulation times for the Taylor-Green vortices test case. h is the smoothing length, N is the number of points in the entire domain at the initial state, ϵ_2 is the relative error, r is the order of convergence of ϵ_2, and t is the simulation time in seconds.

		Projection			Coupled: Penalty			Coupled: New		
h	N	ϵ_2	r	t	ϵ_2	r	t	ϵ_2	r	t
1	293	3.1×10^{-2}	–	6	2.4×10^{-2}	–	7	1.2×10^{-2}	–	9
1/2	1 047	9.7×10^{-3}	1.67	28	6.5×10^{-3}	1.88	29	3.1×10^{-3}	1.95	32
1/4	3 856	3.2×10^{-3}	1.59	195	2.5×10^{-3}	1.37	202	1.0×10^{-3}	1.63	207
1/8	14 878	1.1×10^{-3}	1.54	1934	8.2×10^{-4}	1.60	2010	3.3×10^{-4}	1.59	2031

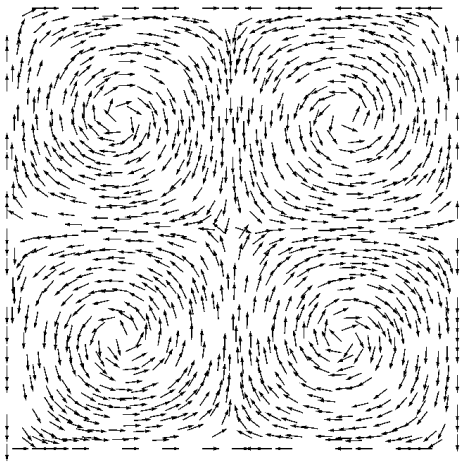


Figure 5.1: Initial condition for Taylor Green vortices. Arrow lengths are constant and are not scaled by velocity magnitudes.

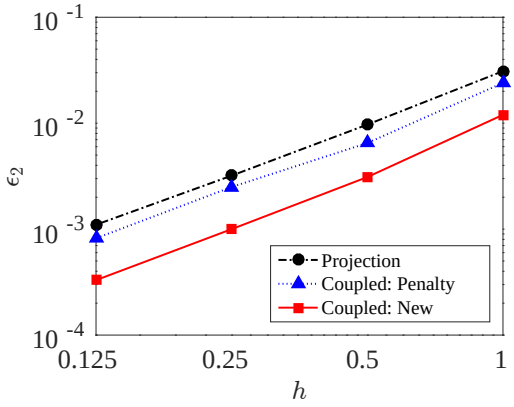


Figure 5.2: Convergence of error for Taylor-Green vortices.

5.4.2 Flow Through a Bifurcated Tube

We consider flow of a fluid through the bifurcated tube shown in Figure 5.3. The length of the tube is $60m$, the width is $4m$ in the thick region and $1m$ in the thin region. Simulation parameters are set as $t_{end} = 1s$, $\rho = 10^3 kg/m^3$ and $\eta = 2Pa\,s$. The velocity at the inflow, on the left of the tube, is kept constant at $\vec{v}_{in} = (2m/s, 0)$. This results in a Reynolds number of the order of 10^3. A varying time step is used according to Eq. (5.38) with $C_{\Delta t} = 0.05$. Homogeneous Neumann boundary conditions for the velocity are used at the outflow and no-slip conditions on the walls. The pressure is kept constant at atmospheric pressure at the outflow and homogeneous Neumann boundary conditions are considered elsewhere for the pressure. The error in mass conservation is measured as the difference between the total volume of fluid flowing in and that flowing out, throughout the entire simulation. The mass conservation error

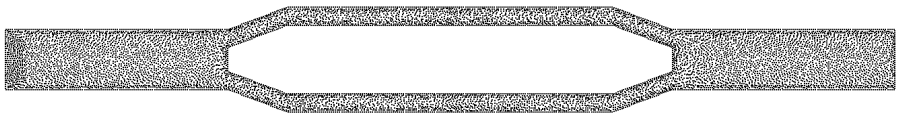


Figure 5.3: 2D Bifurcated tube. Fluid inflow is on the left, and outflow is on the right.

is measured as done in the previous chapter and is given by

$$\epsilon_{mass} = \left| \frac{\int_0^{t_{end}} \left[\int_{\partial\Omega_{in}} \vec{n} \cdot \vec{v}\, dA \right] dt + \int_0^{t_{end}} \left[\int_{\partial\Omega_{out}} \vec{n} \cdot \vec{v}\, dA \right] dt}{\int_0^{t_{end}} \left[\int_{\partial\Omega_{in}} \vec{n} \cdot \vec{v}\, dA \right] dt} \right| , \tag{5.45}$$

where $\vec{n}$ is the outward pointing unit normal and $\partial\Omega_{in}$ and $\partial\Omega_{out}$ are the inflow and outflow boundaries respectively. Note that ϵ_{mass} measures the errors during transient states too, and not just the errors in the steady state solution. A measure for the velocity divergence throughout the domain is taken as the integral of the divergence of velocity scaled by the total volume

$$D(\vec{v}) = \frac{\int_\Omega |\nabla \cdot \vec{v}| dV}{\int_\Omega dV} , \tag{5.46}$$

which is also as done in the previous chapter. This can be interpreted as the average value of local error in the mass conservation equation. Note that there is no direct correlation between the measures of divergence and mass conservation, $D(\vec{v})$ and ϵ_{mass}, because the absolute value of velocity divergence is taken in $D(\vec{v})$. The presence of a numerical source and a sink of equal magnitudes for $\nabla \cdot \vec{v}$ would cancel out while measuring the mass conservation, but they would add up while measuring $D(\vec{v})$. Figure 5.4 shows the convergence of the velocity divergence averaged over all time steps, and the convergence of the error in mass conservation with respect to changing smoothing length h. The same are also tabulated in Table 5.2. The errors follow a similar pattern to those in the Taylor-Green vortices test case. The new coupled solver produces significantly smaller errors than the other two methods in both the average velocity divergence and mass conservation. The difference between the projection method and the coupled penalty approach is quite small. We note that convergence orders are observed to be much smaller than in the earlier test case due to the use of larger time steps, Neumann boundary conditions and a non-standard domain geometry.

Simulation times for the three methods are also shown in Table 5.2. While the new coupled solver is the slowest by a large margin on the coarsest point cloud, it is the fastest on the finest point cloud. This could be due to slower convergence of linear systems in the other two methods.

To illustrate that the difference between the new coupled solver presented in this chapter and the penalty formulation coupled solver goes beyond the addition of the zero divergence equation at boundary points, we split the average divergence of velocity, Eq. (5.46), along the interior and boundary points $D(\vec{v}) = D_{int}(\vec{v}) + D_{bnd}(\vec{v})$,

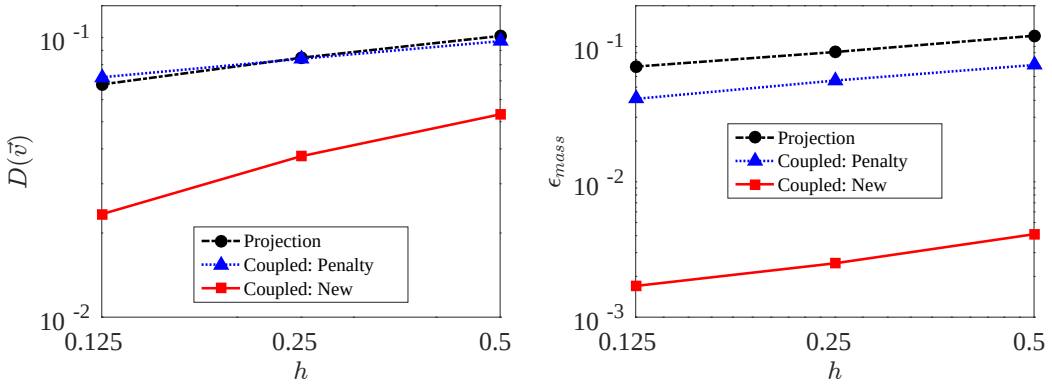


Figure 5.4: Bifurcated tube: average divergence of velocity in the simulation domain (left) and error in mass conservation (right).

Table 5.2: Errors, convergence orders and simulation times for the bifurcated tube test case. h is the smoothing length, N is the number of points in the entire domain at the initial state, ϵ_{mass} is the relative error in mass conservation, r is the order of convergence of ϵ_{mass}, and t is the simulation time in seconds.

		Projection			Coupled:Penalty			Coupled:New		
h	N	ϵ_{mass}	r	t	ϵ_{mass}	r	t	ϵ_{mass}	r	t
1/2	5 805	1.2×10^{-1}	–	556	7.3×10^{-2}	–	718	4.1×10^{-3}	–	927
1/4	21 294	9.1×10^{-2}	0.40	7378	5.6×10^{-2}	0.38	8752	2.5×10^{-3}	0.71	6056
1/8	81 125	7.1×10^{-2}	0.36	76850	4.1×10^{-2}	0.45	76936	1.7×10^{-3}	0.56	53477

with

$$D_{int}(\vec{v}) = \frac{\int_{\Omega \setminus \partial\Omega} |\nabla \cdot \vec{v}| dV}{\int_{\Omega} dV}; \quad D_{bnd}(\vec{v}) = \frac{\int_{\partial\Omega} |\nabla \cdot \vec{v}| dV}{\int_{\Omega} dV}, \tag{5.47}$$

where $\Omega \setminus \partial\Omega$ represents only the interior points. Figure 5.5 shows $D_{int}(\vec{v})$ and $D_{bnd}(\vec{v})$ averaged over all time steps. It illustrates that the new coupled solver improves the accuracy of the mass conservation condition across both interior and boundary points.

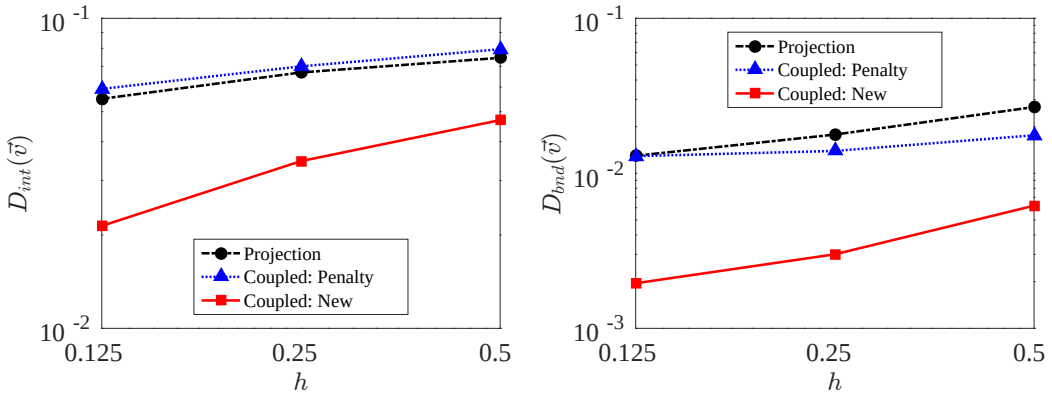


Figure 5.5: Velocity divergence on interior points (left) and boundary points (right) for the bifurcated tube test case.

5.4.3 Sloshing

The most common area of application of Lagrangian meshfree methods is for flows with moving free surfaces. We consider the sloshing of a fluid contained in a constantly moving rectangular box as shown in Figure 5.6. The dimensions of the initial state of the fluid are $1.2m \times 0.12m$, and that of the box containing the fluid are $1.2m \times 0.6m$.

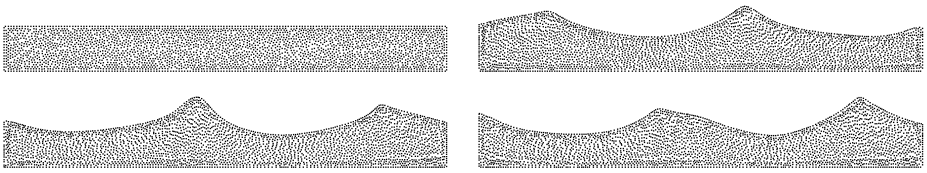


Figure 5.6: Sloshing at times $t = 0s$ (top left), $t = 1.31s$ (top right), $t = 1.63s$ (bottom left), and $t = 2.23s$ (bottom right).

The initial state is taken to be at rest. Slip boundary conditions are used at the walls for the velocity. The free surface boundary conditions are given as described in Section 2.5.4, without surface tension. Homogeneous Neumann boundary conditions are used for the pressure at the walls. The movement of the box is represented in the gravitational and body forces term by setting $\vec{g} = (2\cos(10t), -10)$. The simulation parameters are set as $t_{end} = 3s$, $\rho = 10^3 kg/m^3$, and $\eta = 0.1Pa\,s$, which leads to a Reynolds number of the order of 10^4. A varying time step is used according to Eq. (5.38) with $C_{\Delta t} = 0.3$. As done in the previous chapter, the error in mass conservation is measured by the change in total volume occupied by all points, since the density ρ is fixed and constant throughout the domain

$$\epsilon_V = \frac{|\int_{\Omega_0} dV - \int_{\Omega_{end}} dV|}{\int_{\Omega_0} dV}, \tag{5.48}$$

where Ω_0 is the initial domain and Ω_{end} is the domain at t_{end}. The average velocity divergence is measured as done earlier in Eq. (5.46). The convergence of volume conservation error and average velocity divergence $D(\vec{v})$ with respect to a changing smoothing length h for all three methods are shown in Figure 5.7 and are tabulated in Table 5.3. The results follow a similar trend to that in the earlier two test cases. The new coupled solver shows the highest accuracy in both velocity divergence and mass conservation while the projection method shows the least accuracy in both. However, the difference between the new method and the two older methods is not as large as in the previous test case. The sharp increase in accuracy between $h = 0.06$ and $h = 0.03$ is due to a bad approximation of the free surface for the coarsest point cloud. For the finer point clouds, a small convergence rate is observed once again due to the boundary conditions used and the use of large time steps.

Simulation times for the three methods are also present in Table 5.3. All three methods take almost the same simulation time, with the new coupled solver being the slowest for 3 out of the 4 point clouds considered.

We note that more turbulent sloshing problems than the one considered here produce larger errors in volume conservation, but for such problems, the largest source of error

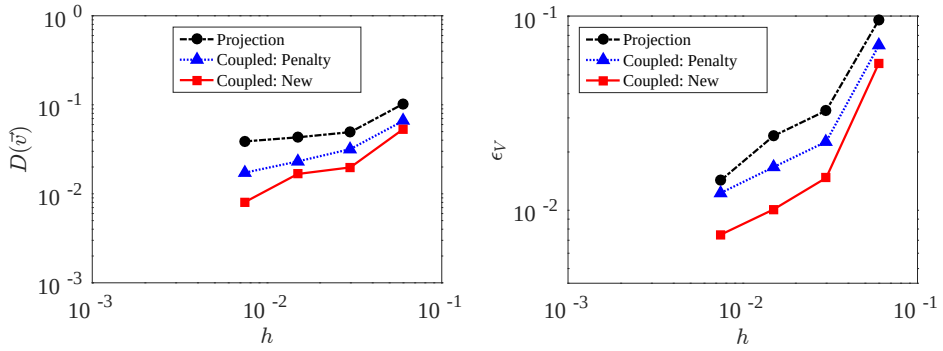


Figure 5.7: Sloshing: convergence of solution with smoothing length h. Divergence of velocity field (left) and error in mass conservation (right).

Table 5.3: Errors, convergence orders and simulation times for the sloshing test case. h is the smoothing length, N is the number of points in the entire domain at the initial state, ϵ_V is the error in volume conservation, r is the order of convergence of ϵ_V, and t is the simulation time in seconds.

		Projection			Coupled:Penalty			Coupled:New		
h	N	ϵ_V	r	t	ϵ_V	r	t	ϵ_V	r	t
0.06	382	9.59×10^{-2}	–	10	7.09×10^{-2}	–	12	5.73×10^{-2}	–	15
0.03	1 361	3.27×10^{-2}	1.56	51	2.26×10^{-2}	1.65	59	1.47×10^{-2}	1.96	68
0.015	5 119	2.42×10^{-2}	0.43	723	1.67×10^{-2}	0.44	602	1.01×10^{-2}	0.54	746
0.0075	19 704	1.43×10^{-2}	0.76	7871	1.23×10^{-2}	0.44	6685	0.75×10^{-2}	0.43	7268

is often in the management of the point cloud and not in the approximation of the PDEs.

5.4.4 Errors in Taylor Expansions

In the classical meshfree GFDM approach of Section 2.5, the errors in Taylor expansions are minimized directly. For the direct GFDM approach, the addition of the PDE error terms to the functional minimization in Eq. (2.64) and Eq. (5.23) could result in larger approximation errors in the Taylor expansions.

For $\vec{v} = (u, v)$, we examine this difference numerically by looking at the errors in the Taylor expansions for u. Formally, the functional being compared is $J_i = \sum_{j \in S_i} W_{ij} (e^u_{ij})^2$ for each point i. After the velocity at the new time-level is computed, the obtained velocity is checked for errors in the Taylor expansions. Thus, these errors include not just the truncation error in the discretization, but also the error due to the tolerance of the sparse iterative solver. This comparison is done using the Taylor-Green vortices test case considered earlier in Section 5.4.1, and for velocities after the completion of the first time step of the simulation. These errors are tabulated in Table 5.4 for the classical GFDM (with the projection method) and the direct framework used for the new coupled solver. The errors are larger in the new coupled solver, but not significantly. This agrees with the results in Section 2.7 that the added approximation error in the direct GFDM is minimal.

Table 5.4: Errors in Taylor expansions measured by the functional $J_i = \sum_{j \in S_i} W_{ij} (e^u_{ij})^2$. The table shows the mean value of J_i across all interior points at the end of the first time step for the Taylor-Green vortices test case.

h	Classical GFDM: Projection	Direct GFDM: New Coupled Solver
1.0	0.2976	0.3057
0.5	0.09579	0.09709
0.25	0.03255	0.03345
0.125	0.01153	0.01470

5.5 Conclusion

We presented a new monolithic algorithm for the incompressible Navier–Stokes equations, solved using a meshfree Generalized Finite Difference Method (GFDM). While existing algorithms either solve the mass conservation indirectly via a pressure-Poisson equation or introduce an artificial compressibility, in the new method presented here, the mass conservation is solved directly without the introduction of any artificial compressibility. This results in improved local approximations for the mass conservation equation for both interior and boundary points, which results in better accuracy overall. In this method, the momentum and mass conservation equations are solved together, and simultaneously with a pressure-Poisson equation. The addition of this Poisson equation is essential for stabilizing the scheme and results in an overdetermined system of PDEs. Accuracy is further improved at boundary points by solving the mass conservation equation in addition to the usual boundary conditions.

This new scheme was compared with two existing methods: one fractional step method and one monolithic method. All three methods have the same spatial and temporal order of accuracy. Numerical comparisons were done for different Reynolds flows, and the new method was shown to produce more accurate results. Comparisons were done for the benchmarking example of Taylor-Green vortices, and also for non-conventional examples with different domains, boundary conditions and free surfaces. In each case, the new monolithic method showed better approximations to the velocity zero-divergence condition, which resulted in more accurate results overall.

The new coupled solver exhibits stability similar to that of the penalty-approach monolithic method, and thus, much better than the projection method. This comes as the cost of a higher memory requirement to store the sparse linear systems. The new coupled solver uses a modified GFDM framework, as a result of which it avoids the saddle point structure of linear systems arising from conventional monolithic methods. The simulation times for the three methods compared were similar, and the fastest method varied on a case by case basis. Thus, the improved accuracy in the new coupled solver does not come at the cost of higher computational times. While numerical examples in this chapter were only carried out in 2 spatial dimensions, the algorithm extends easily to more realistic problems in $3D$ [2].

[2] We note that since the completion of this thesis, the new monolithic solver presented in this chapter has been extended to three spatial dimensions, and similar results were obtained.

An interesting point of study not considered here is the impact of the ratio of weights in the minimization of the functional in Eq. (5.23). Higher weights could be used, for example, for the minimization of the error in the mass conservation equation.

From a larger perspective, the ideas presented in this chapter also illustrate the potential of using the direct GFDM framework presented in Section 2.6 to solve over-determined problems. And how the same can be used to determine solutions of systems of PDEs.

Chapter 6

Point Cloud Movement in Lagrangian Meshfree Methods

In Lagrangian meshfree methods, the point cloud which forms the underlying spatial discretization moves with the flow velocity. In this chapter, we consider different numerical methods of performing this movement of points or particles. This movement is most commonly done by a first order method, which assumes the velocity to be constant within a time step. We show that this method is very inaccurate and that it introduces volume and mass conservation errors. We further propose new methods for the same which prescribe an additional ODE system that describes the characteristic velocity. Movement is then performed along this characteristic velocity. The first new way of moving points is an extension of mesh-based streamline tracing ideas to meshfree methods. In the second way, movement is done based on the difference in approximated streamlines between two time levels, which approximates the pathlines in unsteady flow. Numerical comparisons show these methods to be vastly superior to the conventionally used first order method.

6.1 Introduction

A moving Lagrangian framework is commonly used while modeling fluid flow. It often provides better approximations than the fixed Eulerian framework for flows with open free surfaces and multiphase flows with moving interfaces. The Lagrangian framework has the further advantage of avoiding the non-linear advection term, and often provides a more accurate depiction of transport phenomena. However, this comes at the cost of generally having a more restrictive time step size control, and having the need to take special care for several aspects of conservation.

In mesh-based methods, moving the mesh causes the additional disadvantage of mesh distortion. To avoid this distortion and the need to remesh, a large class of semi-Lagrangian and Arbitrary Lagrangian-Eulerian (ALE) methods have been developed (see, for example, [23, 106]). To improve conservation properties of mesh-based Lagrangian methods, the so-called ideas of trace-back and volume adjustments for mass conservation are often used [3, 45]. The ideas include adjusting the volume of individual elements or cells based on their traced-back entities, and constructing upstream vertices and cells based on their corresponding downstream ones.

Meshfree methods provide a more natural fit to Lagrangian frameworks than mesh-based methods. They use the numerical basis of a set of arbitrarily distributed nodes

without any underlying mesh to connect them. These nodes could either be mass carrying particles or numerical points. Movement of this set of nodes, referred to as a point cloud, in a Lagrangian framework could also lead to distortion. However, point cloud distortion is easier to fix, as point clouds can easily be adapted locally, especially in meshfree methods that use numerical approximation points instead of mass carrying particles (see Section 2.4).

To improve conservation properties, trace-back ideas used in mesh-based methods have also been generalized to Lagrangian meshfree methods [44]. These methods involve adjusting the volume or mass of particles or the physical properties of approximation points appropriately *after* their locations have been updated. In this chapter, we consider the process of updating the point locations itself. We try to improve conservation properties by addressing the question of the optimum process of point movement, instead of adjusting physical quantities after movement is performed.

For incompressible flow, inaccurate movement of points or particles results in errors in volume conservation. This results in the introduction of numerical compressibility. This problem is especially relevant for applications with open free surfaces. To solve this and similar problems, various 'artificial displacement' ideas have been introduced, especially in the context of the meshfree Smoothed Particle Hydrodynamics (SPH). These involve performing an extra movement step in addition to the Lagrangian movement (or, equivalently, the addition of an extra term in the usual Lagrangian movement step). This additional movement is artificial in the sense that it is not based solely on the fluid velocity, as is the nature of the Lagrangian framework; rather it is based on improving different aspects of the numerical solution, such as to conserve total incompressibility of the simulated fluid [86] or to prevent local clustering of particles [50, 101]. Both physical and non-physical arguments have been used for the same. These methods are referred to under various names, such as particle shifting, artificial particle displacement, corrective displacement and particle regularization. In contrast to such methods, the methods presented in this chapter improve the main Lagrangian movement step without the introduction of any additional artificial movement.

We begin the chapter by showing that the most widely used first order approach for movement of points is extremely inaccurate, and propose two new methods for the same. The first new method is based on generalizing mesh-based streamline tracing. In this method, streamline velocities are approximated by an ODE system, and points are moved along these approximated streamlines. This method proves to give very good approximations for quasi-stationary flow problems. A further new method is developed which considers movement according to the change of these approximated streamlines between consecutive time levels. Numerical simulations show that this method gives much better results for rapidly changing flow profiles.

To illustrate the use of these methods of point cloud movement, we use the classical meshfree GFDM framework of Section 2.5. However the same could also be applied to other meshfree methods. Since the methods discussed here can be applied to both numerical points in approximation point based meshfree methods and to particles in mass carrying particle based meshfree methods, the words 'point' and 'particle' are used interchangeably.

In Section 6.2 we introduce the notation used, and present the first order method which is commonly used to move point clouds in Lagrangian meshfree methods. In

Section 6.3, we talk about mesh-based streamline tracing, and discuss about why direct extension of these ideas to meshfree methods is challenging. We then propose new methods for point cloud movement in Section 6.4. Numerical results on the application of different movement methods are shown for a Lagrangian advection equation in Section 6.5, and for the Lagrangian Navier–Stokes equations in Section 6.6. The chapter is then concluded with a short discussion on the work in Section 6.7.

6.2 Point Cloud Movement - First Order Method

We consider the time-integration of point locations while proceeding from time level t^n to t^{n+1}, with $\Delta t = t^{n+1} - t^n$ being the time step size. Times in between the two time levels are referred to by $t^n + \tau$. Point cloud movement is done by integrating the equation

$$\frac{D\vec{x}}{Dt} = \vec{v}\,. \tag{6.1}$$

Bracketed superscirpts are used to refer to the time level. Thus, point locations at t^n and t^{n+1} are referred to by $\vec{x}^{(n)}$ and $\vec{x}^{(n+1)}$ respectively; and velocities by $\vec{v}^{(n)}$ and $\vec{v}^{(n+1)}$ respectively. The closed-form movement is referred to as $\Delta\vec{x} = \vec{x}^{(n+1)} - \vec{x}^{(n)}$. Throughout this discussion, we only consider methods which decouple the movement step from the remaining PDEs being solved. Further, within a time step, we consider the movement to be done before solving the remaining PDEs. Thus, $\vec{v}^{(n+1)}$ is unknown during movement. If the movement process would be done after the solution of remaining PDEs is computed, $\vec{v}^{(n-1)}$ and $\vec{v}^{(n)}$ would be replaced with $\vec{v}^{(n)}$ and $\vec{v}^{(n+1)}$ respectively, throughout the chapter.

6.2.1 First Order Movement

The most commonly used approach for moving point clouds is by assuming the velocity to be constant throughout the time step.

$$\Delta\vec{x} = \vec{v}^{(n)}\Delta t\,, \tag{6.2}$$

which is performed at each numerical point or particle. Several variations of this form of movement are used in different meshfree methods. The most common is for each point to move with its own velocity $\Delta\vec{x}_i = \vec{v}_i^{(n)}\Delta t\,,\ \forall i$. Some variants of SPH use movement with the average velocity in the neighbourhood of each particle [78], while others perform two first order movements per time step, one based on an 'intermediate' velocity, and one based on a 'final' velocity [87]. Some meshfree methods use the average velocity of the current and the previous time step [103], and even refer to the same as a second order method. We club all of these methods under first order methods, since they assume the velocity to be constant throughout the time step.

This method provides a very inaccurate approximation of moving points. The most significant errors come in capturing rotational parts of the flow correctly. This can be illustrated by the simple case of a rotating disc. Each point on the boundary of the disc has an instantaneous velocity in the tangential direction, as illustrated in

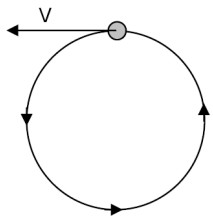


Figure 6.1: In flows with a rotational component, the most commonly used first order movement causes points to move along the tangential velocity.

Figure 6.1. Moving boundary points along this direction will result in the disc to constantly increase in volume.

Despite the significant inaccuracies due to this method of movement, it is still used extensively today in almost all Lagrangian meshfree methods (see, for example, recent work in [73, 87, 103, 118]).

6.3 Mesh-Based Particle and Streamline Tracing

Particle tracing is a well established method for visualizations for fluid flow simulations [51]. A common way to do this is to compute tangent curves by assuming a linearly varying velocity field [20, 82]. This amounts to computing analytical solutions to streamlines based on the underlying mesh. Such streamline tracing algorithms have also been extended to irregular geometries and more complex meshes [52].

These ideas of streamline tracing in visualization have also been extended to the movement of particles in Lagrangian particle methods. One such example is the Particle Finite Element Method (PFEM) which uses moving Lagrangian particles with an underlying mesh. Like the aforementioned methods, vector fields are assumed to be piecewise linear on the background mesh. Based on these, piecewise integration of particle motion is performed by identifying the locations where particles cross simplex boundaries, like that done in [51]. As a result, a closed-form solution to the movement is obtained [41, 80]. In addition to improving accuracy of movement, this method was shown to reduce the restriction on the time step size arising from the explicit nature of movement. Since most meshfree methods lack a global underlying mesh, such streamline tracing methods do not easily generalize to meshfree methods.

An alternative to analytical streamline tracing on the mesh is to approximate the streamlines numerically. This is done by splitting the time step into multiple sub-steps only for the purpose of determining the movement [41]. In each sub-step, each particle is moved with a first order method. Then, at the resulting temporary location at the end of a sub-step, a new velocity needs to be approximated for that temporary location which is then used for the movement in the next sub-step. This velocity approximation is also done based on the underlying mesh. Such a method has several drawbacks in a meshfree setting. Firstly, in the absence of a background mesh, the interpolation of velocities at the intermediate locations can prove to be very costly. Further, errors due to the interpolation at temporary locations can be quite significant [14]. Moreover,

since each sub-step uses first order movement of points, any improvement to the first order method can be coupled with this sub-stepping method.

Numerical integration based on high-order numerical methods such as Runge–Kutta methods have also been used for the purpose of particle tracking in mesh-based fluid flow problems. These methods have also been used for particle movement in molecular dynamics communities [53]. However, these methods have been shown to be less accurate than those based on streamline tracing [72]. Moreover, such higher order methods prove to be challenging in the context of many meshfree methods. These higher order methods often require estimating mid-point slopes. For the movement process, this amounts to estimating velocities in the middle of a time step (at locations where no approximation node is present). This is quite expensive in a meshfree framework, as explained above. Other higher order methods require information about velocities at more than two time-levels which, as we shall show later, is often not available.

Due to these difficulties, the first order method, Eq. (6.2), is used almost exclusively for the purpose of updating particle locations. In this chapter, we present methods to improve the accuracy of movement of Lagrangian points in meshfree methods. We begin by presenting a simple second order method and explaining why further higher order methods are not feasible. Then we extend streamline tracing to meshless contexts, and lastly, we combine the second order and streamline tracing methods to obtain a new method that gives better results for non-steady flows.

6.4 Improved Methods for Point Cloud Movement

6.4.1 Second Order

To avoid the inaccurate movement of the first order method, second order methods have been used by Kuhnert *et al.* [47, 57]. Such second order methods have also been used in ALE frameworks [40] and in a select few SPH codes [18]. Instead of assuming the velocity to be constant between two time levels, the velocity derivative is assumed to be constant. The movement can be considered to be done over a characteristic velocity $\vec{v}^c$, which satisfies the system

$$\frac{D\vec{v}^c}{Dt}(\tau) = \frac{\vec{v}^{(n)} - \vec{v}^{(n-1)}}{\Delta t}\,, \qquad 0 < \tau < \Delta t \tag{6.3}$$

$$\vec{v}^c(0) = \vec{v}^{(n)}\,. \tag{6.4}$$

Note that this system is solved for each point. The particle displacements are then found by integrating along the characteristic velocity

$$\frac{D\vec{x}}{Dt} = \vec{v}^c\,, \tag{6.5}$$

to obtain

$$\Delta\vec{x} = \vec{v}^{(n)}\Delta t + \frac{1}{2}\frac{\vec{v}^{(n)} - \vec{v}^{(n-1)}}{\Delta t}(\Delta t)^2\,. \tag{6.6}$$

We note that in this method, and henceforth, the variation of the characteristic velocity between two time steps is only considered for the sake of movement of points,

and not for the computation of the new velocity, $\vec{v}^{(n+1)}$, which is done based on the relevant PDE being solved.

Higher order time integration methods, such as those used in Discrete Element Method (DEM) simulations [53], are usually not possible, especially for approximation point-based meshfree methods. To prevent the distortion of point clouds, local adaptation is usually done. This involves adding points in locations containing 'holes' and removing points in locations containing clusters of points (see Section 2.4). All physical properties at the current time level are approximated at every new point created. However, approximating velocities from multiple time levels before the current one is extremely inaccurate. Thus, a newly created point would not posses $\vec{v}^{(n-2)}, \vec{v}^{(n-3)}, \ldots$, which are needed in many higher order time integration methods. Thus, these higher order methods can not be used in many meshfree settings.

6.4.2 Meshfree Movement Along the Streamlines

In fluid flow, streamlines describe the flow direction at a fixed instance in time. Thus, while going from time t^n to t^{n+1}, moving points along the streamline would entail moving them along the velocity field at t^n. Meshed methods approximate this streamline velocity by performing linear interpolations on the underlying mesh. In contrast, we approximate the same by prescribing an ODE system which can be solved analytically.

To perform movement along the velocity streamlines, we assume that each point is being advected based on the velocity gradient at its original location. Streamline velocities are computed based solely on the convective acceleration $(\vec{v} \cdot \nabla)\,\vec{v}$. Since the streamlines are taken at time t^n, the convective term can be taken as $(\nabla \vec{v}^{(n)})\,\vec{v}$, written as a matrix-vector product. Thus, the characteristic velocity between the time levels, along which movement is performed, can be taken by the initial value problem

$$\frac{D\vec{v}^c}{Dt}(\tau) = (\nabla \vec{v}^{(n)})\,\vec{v}^c\,, \qquad 0 < \tau < \Delta t \tag{6.7}$$

$$\vec{v}^c(0) = \vec{v}^{(n)}\,. \tag{6.8}$$

Eq. (6.7) can be interpreted as the material derivative of the velocity, with the partial time derivative set to 0. The above assumption of setting $\nabla\vec{v} \approx \nabla\vec{v}^{(n)}$ ensures that the resultant ODE system can be solved analytically to obtain[1]

$$\vec{v}^c(\tau) = \exp(\nabla \vec{v}^{(n)}\,\tau)\,\vec{v}^{(n)} \tag{6.9}$$

$$= \left[\sum_{k=0}^{\infty} \frac{1}{k!} \left(\nabla \vec{v}^{(n)}\right)^k \tau^k\right] \vec{v}^{(n)}\,. \tag{6.10}$$

Integrating the movement equation, Eq. (6.5), leads to the following closed form

$$\Delta\vec{x} = \left[\sum_{k=0}^{\infty} \frac{1}{(k+1)!} \left(\nabla \vec{v}^{(n)}\right)^k (\Delta t)^{k+1}\right] \vec{v}^{(n)}\,, \tag{6.11}$$

[1]Matrix exponential convention is used here. $\left(\nabla \vec{v}^{(n)}\right)^0$ is defined to be the identity matrix of appropriate size, irrespective of the value of $\nabla \vec{v}^{(n)}$.

where the velocity gradient is approximated numerically for each point based on the velocities of its neighbouring points. Note that in the case when the velocity is constant in space, $\nabla\vec{v}^{(n)} = 0$ and Eq. (6.11) reduces to first order movement as given in Eq. (6.2).

While this method accurately captures steady flow, a significant disadvantage is that it assumes the flow to be quasi-stationary. Ignoring the change of the velocity with time results in this method not providing very good approximations when there is a rapid change in the velocity profile.

6.4.3 Movement According to the Change in Streamlines

To overcome the disadvantage of the quasi-stationary flow assumption in the movement along the streamline method, we now present a method that is a generalization of the streamline method and the second order method. Rather than moving only along the streamline at the present time level, the difference of the streamlines of the present and the previous time levels is considered. For non-steady flows, this provides an approximation of the pathlines of the flow. The approximated streamline velocity at the present time level t^n and previous time level t^{n-1} are referred to as $\vec{v}^s$ and $\vec{v}^{s_0}$ respectively. As done earlier, the streamline velocities are computed as

$$\begin{cases} \dfrac{D\vec{v}^s}{Dt}(\tau) = (\nabla\vec{v}^{(n)})\vec{v}^s\,, & 0 < \tau < \Delta t \\ \vec{v}^s(0) = \vec{v}^{(n)}\,. \end{cases} \tag{6.12}$$

$$\begin{cases} \dfrac{D\vec{v}^{s_0}}{Dt}(\tau) = (\nabla\vec{v}^{(n-1)})\vec{v}^{s_0}\,, & 0 < \tau < \Delta t \\ \vec{v}^{s_0}(0) = \vec{v}^{(n-1)}\,. \end{cases} \tag{6.13}$$

Which leads to the streamline velocities

$$\vec{v}^s(\tau) = \left[\sum_{k=0}^{\infty} \frac{1}{k!}\left(\nabla\vec{v}^{(n)}\right)^k \tau^k\right]\vec{v}^{(n)}\,, \tag{6.14}$$

$$\vec{v}^{s_0}(\tau) = \left[\sum_{k=0}^{\infty} \frac{1}{k!}\left(\nabla\vec{v}^{(n-1)}\right)^k \tau^k\right]\vec{v}^{(n-1)}\,. \tag{6.15}$$

Note that $\vec{v}^s$ is defined at time $t^n + \tau$, while $\vec{v}^{s_0}$ is defined at time $t^{n-1} + \tau$. The second order method assumed a characteristic velocity which had a constant derivative throughout the time step, based on the difference of velocities between the time levels. In this method, we assume a characteristic velocity such that the velocity derivative is based on the difference of the approximated streamline velocities.

$$\frac{D\vec{v}^c}{Dt}(\tau) = \frac{\vec{v}^s - \vec{v}^{s_0}}{\Delta t}\,, \qquad 0 < \tau < \Delta t \tag{6.16}$$

$$\vec{v}^c(0) = \vec{v}^{(n)}\,. \tag{6.17}$$

Integrating this leads to the characteristic velocity

$$\vec{v}^c(\tau) = \vec{v}^{(n)} + \frac{1}{\Delta t}\left[\sum_{k=0}^{\infty} \frac{\tau^{k+1}}{(k+1)!}\left(\left(\nabla\vec{v}^{(n)}\right)^k \vec{v}^{(n)} - \left(\nabla\vec{v}^{(n-1)}\right)^k \vec{v}^{(n-1)}\right)\right]. \tag{6.18}$$

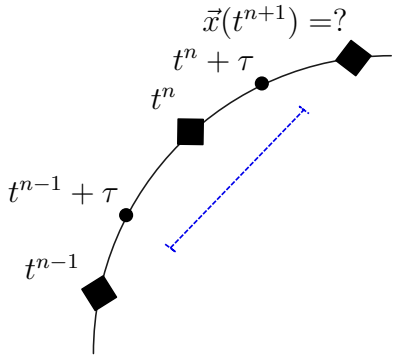


Figure 6.2: Moving with the change of streamlines: The actual locations of the point being considered is represented by squares. The smaller circles represent instantaneous dummy locations between the time levels. The instantaneous derivative of the characteristic velocity is taken to be the difference in approximated (streamline) velocities at the dummy locations (the difference along the blue line). The movement is finally done along the resultant characteristic velocity.

Integrating again to obtain the displacement,

$$\Delta\vec{x} = \vec{v}^{(n)}\Delta t + \frac{1}{\Delta t}\left[\sum_{k=0}^{\infty}\frac{(\Delta t)^{k+2}}{(k+2)!}\left(\left(\nabla\vec{v}^{(n)}\right)^k\vec{v}^{(n)} - \left(\nabla\vec{v}^{(n-1)}\right)^k\vec{v}^{(n-1)}\right)\right]. \tag{6.19}$$

This process is represented in Figure 6.2. Unlike the streamline method, this method takes into consideration the change in the velocity field between time levels, and thus provides more accurate results for rapidly changing flows. This comes at the cost of the need to store $\nabla\vec{v}^{(n-1)}$. Note that in the case when the velocity is constant in space, $\nabla\vec{v}^{(n)} = 0$ and Eq. (6.19) reduces to second order movement as given in Eq. (6.6).

In the numerical results presented later, the infinite summations in Eq. (6.11) and Eq. (6.19) are truncated after the first 5 terms, which introduces a small error. Since each of the time-integration methods considered are explicit in nature, the difference in simulation times between the methods is not significant. The results are split into two parts. The first is a pure transport problem and the second incompressible flow according to Navier–Stokes equations.

The explicit nature of each of the methods presented for point cloud movement results in a CFL condition for stability given by $\|\Delta\vec{x}\| < C_{\Delta t}h$. Obtaining a practically usable condition out of this for, say, Eq. (6.19) would be a highly non-trivial task. Thus, only a rough approximation is used by ignoring higher order Δt terms. This would lead to the same approximate stability condition on the time step size for all 4 methods

$$\Delta t < C_{\Delta t}\frac{h}{\|\vec{v}\|_{\max}}. \tag{6.20}$$

As stated in earlier chapters, the majority of simulation time is spent in setting up and solving the large sparse linear systems arising from the implicit discretizations of

the governing equations. On the other hand, all the point cloud movement methods are explicit, and each point can be moved independently. Thus, the movement process does not does consume a significant amount of computational time, and the total simulation time does not vary greatly between the methods.

We, once again, emphasize that the point cloud movement methods introduced in this chapter can be used in all Lagrangian meshfree methods, irrespective of how the other governing equations are discretized, and irrespective of how the numerical derivatives are approximated.

6.5 Numerical Results: Advection Equation

We apply the different movement methods to a pure transport problem.

$$\frac{D\vec{x}}{Dt} = \vec{v}\,, \tag{6.21}$$

$$\frac{D\phi}{Dt} = 0\,, \tag{6.22}$$

where $\vec{v}$ is the advection velocity, and ϕ is the physical quantity being transported.

While both examples considered below use 'simple' prescribed velocity fields, they are used to illustrate the impact of the point movement in Lagrangian transport in meshfree methods. In both cases, the prescribed velocity fields are such that the domain is undergoing rigid body motion. Thus, relative point positions do not change, and there is no deformation of the point cloud. Thus, there is no need to perform point additions or deletions to improve point cloud quality during the simulation, and errors in the needed interpolation do not affect the simulation. This, coupled with the fact that Eq. (6.22) can be integrated trivially, ensures that the only source of error is the movement itself.

6.5.1 Rotation

We consider a circular disc of unit radius rotating about its center. The velocity field is given by

$$\vec{v} = (-y, x)\,. \tag{6.23}$$

The discretized domain contains $N = 222$ points. Since only rigid body rotation is being performed, the area of the disc should be preserved. However, numerically, the disc expands due to movement of points along the tangential velocity at the boundaries, as shown in Figure 6.1. The error in the numerical result is measured as

$$\epsilon_{dia} = |d_{num} - d|\,, \tag{6.24}$$

where d_{num} is the numerical diameter of the disc at the end of two complete rotations, and d is the theoretical diameter of the disc. The error with different time steps for the different movement methods is shown in Figure 6.3. The first order method produces the most inaccurate results. The remaining three methods produce similar results

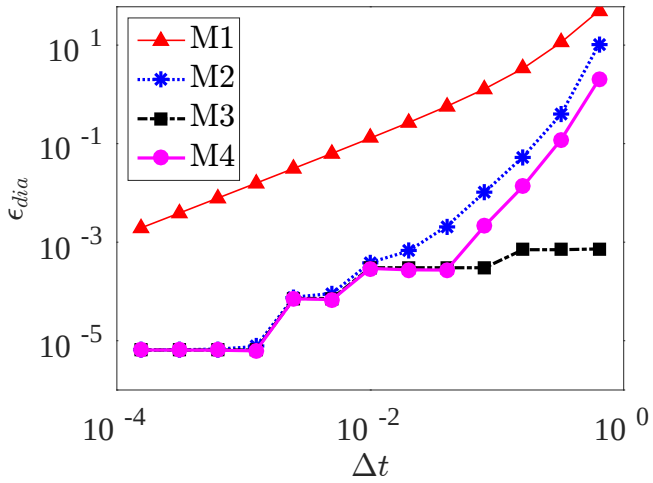


Figure 6.3: Errors for the rotating disc example. M1 stands for the first order method of movement, M2 the second order method, M3 the movement along the streamline, and M4 the movement according to the change of streamlines.

for small time steps, while the movement along the streamline produces the most accurate results for large time steps. For large time steps, the movement according to the change of streamlines is more accurate than the second order method. Even for the smallest time step considered, the first order method is worse than the other three methods by two orders of magnitude.

6.5.2 Transport Along a Lissajous Curve

We consider the same disc as in the previous example. The disc is being transported along a velocity field

$$\vec{v} = (15\cos(5t + \frac{\pi}{2}), 4\cos(4t))\,. \tag{6.25}$$

For a disc with center starting at $(0, 0)$, the exact trajectory of the center of the disc is given by the Lissajous curve

$$\vec{x}_{exact} = (3\sin(5t + \frac{\pi}{2}) - 3, \sin(4t))\,. \tag{6.26}$$

The spatially constant velocity means that $\nabla\vec{v} \equiv 0$, which results in Eq. (6.11) reducing to Eq. (6.2); and Eq. (6.19) reducing to Eq. (6.6), as explained earlier. Thus, in this case, the movement along the streamline and the first order method are the same; and the movement according to the change of the streamlines and the second order method are the same. Such cases with spatially constant velocity in a neighbourhood are relevant for flow regimes in the 'far field' of the domain where the velocity does not vary much. The error in the numerical solution is measured by

$$\epsilon_{\vec{x}} = \|\vec{x}_{num} - \vec{x}_{exact}\|\,, \tag{6.27}$$

where $\vec{x}_{num}$ is the numerical center of the disc. For $\Delta t = 0.05$, the evolution of errors for the different methods of movement are shown in Figure 6.4. The second order and change of streamline method produce more accurate results than the first order and streamline method.

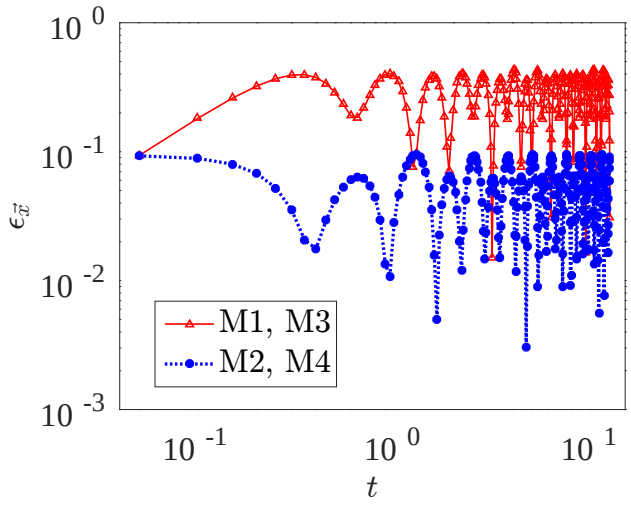


Figure 6.4: Errors in transport along Lissajous curves. M1 stands for the first order method of movement, M2 the second order method, M3 the movement along the streamline, and M4 the movement according to the change of streamlines.

6.6 Numerical Results: Incompressible Navier–Stokes Equations

Consider the incompressible Navier–Stokes equations in Lagrangian form

$$\frac{D\vec{x}}{Dt} = \vec{v}\,, \tag{6.28}$$

$$\nabla \cdot \vec{v} = 0\,, \tag{6.29}$$

$$\frac{D\vec{v}}{Dt} = \frac{\eta}{\rho}\Delta\vec{v} - \frac{1}{\rho}\nabla p + \vec{g}\,, \tag{6.30}$$

where $\vec{v}$ is the fluid velocity, p is the pressure, ρ is the density, η is the dynamic viscosity and $\vec{g}$ includes both gravitational acceleration and body forces. The numerical scheme consists of movement of the point cloud according to one of the methods mentioned earlier, which is followed by a meshfree projection method, as explained in Section 5.2.1. In addition, a $k-\epsilon$ turbulence model is also used. The boundary conditions used, including the ones at the free surface, are as explained in Section 2.5.4. While an inaccurate update of point locations represents one source of error in meshfree solution schemes to the incompressible Navier–Stokes equations, other sources of errors in the same have been discussed in earlier chapters.

The majority of simulation time is spent in setting up and solving the large sparse linear systems arising from the implicit momentum and pressure Poisson equations. Further, all the point cloud movement methods are explicit, and each point can be moved independently. Thus, the movement process does not does consume a significant amount of computational time, and the total simulation time does not vary greatly between the methods.

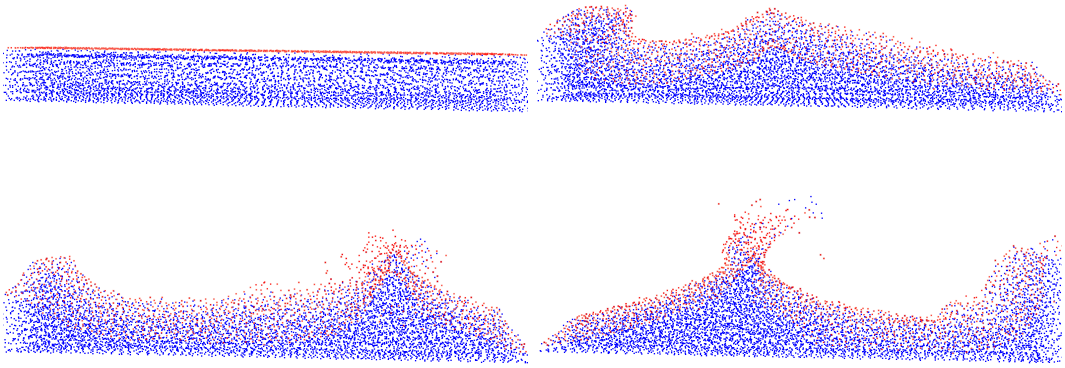


Figure 6.5: Sloshing. $t = 0s$ (top left), $t = 0.72s$ (top right), $t = 1.44s$ (bottom left) and $t = 1.92s$ (bottom right). The red color indicates the free surface.

We, once again, emphasize that the point cloud movement methods introduced in this chapter can be used in all Lagrangian meshfree methods, irrespective of how the mass and momentum equations are discretized, and irrespective of how the numerical derivatives are approximated.

Inaccurate movement of the points on the free surface can result in a volume loss or gain, similar to that illustrated in the rotating disc example considered earlier. For incompressible flow with density ρ fixed and constant throughout the domain, this error in volume conservation also represents an error in mass conservation. This error can be measured as done in earlier chapters, by the relative change in the total volume occupied by all points in the computational domain.

$$\epsilon_V = \frac{|\int_{\Omega_0} dV - \int_{\Omega_{end}} dV|}{\int_{\Omega_0} dV}, \tag{6.31}$$

where Ω_0 is the initial domain and Ω_{end} is the domain at t_{end}.

6.6.1 Sloshing

We consider the test case of sloshing of water contained in a constantly moving rectangular box. This results in a non-stationary free surface as shown in Fig 6.5. The initial state is taken to be at rest. The dimensions of the box are $1.2m \times 0.6m \times 0.2m$. At the initial state, water occupies a cuboidal shape of dimensions $1.2m \times 0.12m \times 0.2m$. Slip boundary conditions are used at the walls for the velocity. Neumann boundary conditions are used for the pressure at the walls. The movement of the box is represented in the gravitational and body forces term by setting $\vec{g} = (4\cos(7t), -10, -5)$. The simulation parameters are set as $t_{end} = 3s$, $\rho = 10^3 kg/m^3$ and $\eta = 10^{-4} Pa\,s$, which results in a Reynolds number of the order of 10^7.

For a constant smoothing length of $h = 0.06$, which corresponds to an initial number of points $N = 12010$, Figure 6.6 shows the error in volume conservation for different

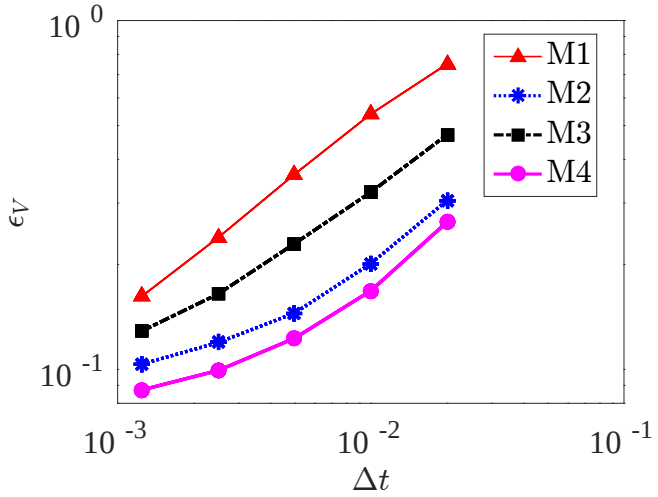


Figure 6.6: Volume conservation errors for the 3D sloshing test case. M1 stands for the first order method of movement, M2 the second order method, M3 the movement along the streamline, and M4 the movement according to the change of streamlines.

time steps. For rapidly changing flows like that in this example, the movement along the streamline method is no longer the most accurate, which agrees with the theoretical expectation since this method does not take into account the velocity change with time. In comparison, the second order movement produces more accurate results. Further, point movement according to the change in streamline method produces slightly better results than the second order method.

We note that in the less extreme sloshing case considered in Section 5.4.3, the approximations of the differential operators and the numerical scheme contribute the largest sources of errors. On the other hand, in the heavier and more extreme sloshing example considered here, the point cloud management, including movement and addition/deletion are the largest sources of errors.

To see the effect of the movement methods, it would also be important to observe what role they play on the discretization of the other governing equations. For the same, the evolution of the mean value of velocity divergence (as defined in earlier chapters in Eq. (4.51) and Eq. (5.46)) is shown in Figure 6.7 for the case of $\Delta t = 0.005$. The figure illustrates that the different methods of movement do not significantly affect the divergence-free nature of the scheme. Further, the evolution of the total pressure at two different locations is shown in Figure 6.8, for the same case of $\Delta t = 0.005$. Here, zero pressure occurs when no fluid is present at that location. While minor differences in the pressure profile are observed due to different point locations, no major difference in the pressure profile is there since the discretization of the mass and momentum equations is the same for each of the movement methods.

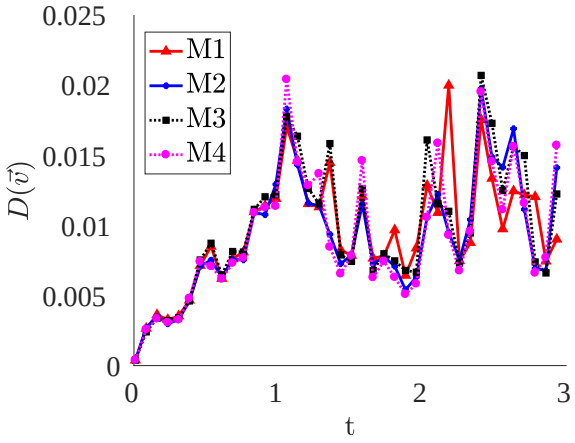


Figure 6.7: Evolution of total divergence of velocity for the sloshing test case. $\Delta t = 0.005$. Plotting interval is 15 time steps. M1 stands for the first order method of movement, M2 the second order method, M3 the movement along the streamline, and M4 the movement according to the change of streamlines.

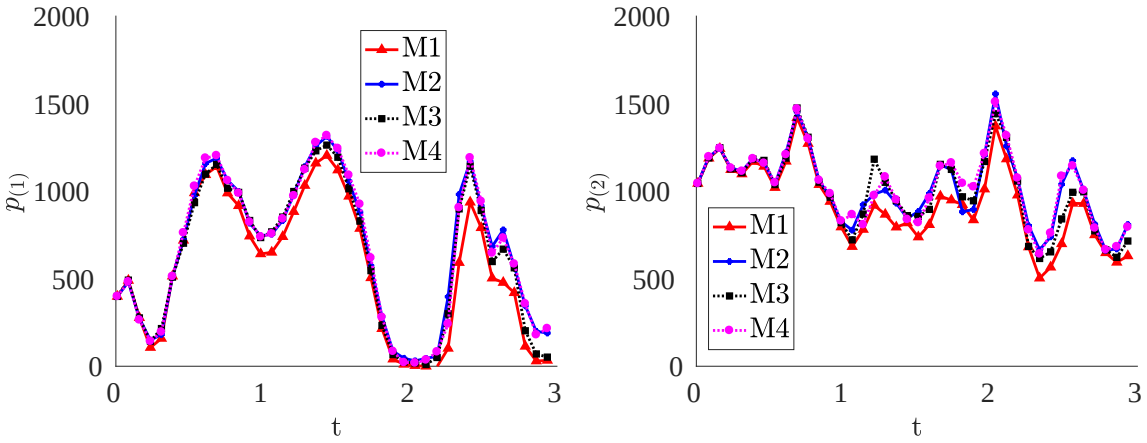


Figure 6.8: Evolution of total pressure for the sloshing test case at two different locations. $\vec{x} = (0, 0.05, 0.1)$ (left) and $\vec{x} = (0.5, 0.05, 0)$ (right). $\Delta t = 0.005$. Plotting interval is 15 time steps. M1 stands for the first order method of movement, M2 the second order method, M3 the movement along the streamline, and M4 the movement according to the change of streamlines.

6.6.2 Tank Filling

We consider the filling of an initially empty cuboidal tank from a inlet near the bottom, as shown in Figure 6.9. For a fixed inflow velocity, the theoretical total volume of the fluid inside the tank is known, and the numerical volume occupied by the point cloud can be compared with it.

This test case captures several challenges of fully Lagrangian meshfree methods. The impact of the high velocity jet on the domain walls can cause points to cluster near the location of impact. Incorrect movement of points near the impact further

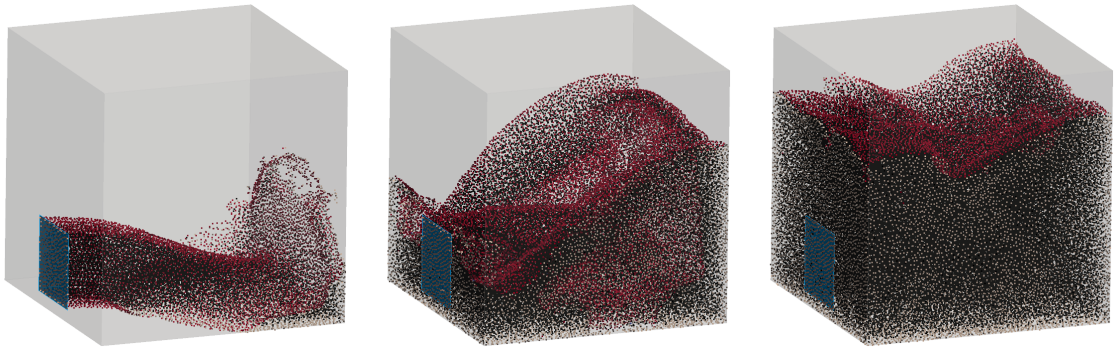


Figure 6.9: Tank filling. $t = 0.36s$ (left), $t = 1.32s$ (center), $t = 3.0s$ (right). Black points represent interior points, red points represent points on the free surface. The wall boundary points in white are hidden, except near the edges. The blue region on the left of the box marks the inflow.

enhances this problem. Such clustering is a source of tensile instability in meshfree methods that use mass carrying particles, such as SPH. While mass conservation is seemingly trivial in these particle based meshfree methods, volume conservation is an issue that affects all meshfree Lagrangian methods. In all meshfree methods, the total geometric volume occupied by the point cloud or the particle cloud represents the physical volume occupied by the simulated fluid. Since incompressible flow is being considered, this volume should be conserved. However, this is not the case numerically, and thus, an artificial compressibility is introduced. Further, incorrect movement at and near the wall boundaries can lead to points escaping the simulation domain. Depending on the meshfree method being used, these escaped points are either projected back to the wall, or are deleted. Both these cases are another source of error in conservation.

The dimensions of the tank considered in the simulations are $1m \times 1m \times 1m$, and that of the inlet are $0.3m \times 0.3m$. A constant velocity inflow of $\vec{v}_{in} = (3, 0, 0)m/s$ is used at the inlet. Slip conditions are used on the walls. The inflowing fluid has properties of $\rho = 10^3 kg/m^3$ and $\eta = 10^{-3} Pa\,s$ which results in a Reynolds number of the order of 10^6. The evolution of the numerical volume for simulations with all four movement methods considered are shown in Figure 6.10 for a large time step of $\Delta t = 0.6 \times 10^{-2}$, and a small time step of $\Delta t = 0.6 \times 10^{-3}$. In both cases, the general trend is the same as that in the sloshing test case. The movement according to the change in streamlines is the most accurate, followed by the second order method; while the first order method is the least accurate. For the larger time step considered, the errors in the streamline and first order methods are significantly larger than the other two methods, while the difference is not as big for the smaller time step. The error in the first order method with the large time step is too huge for this combination to be used in any practical simulations. The relative errors in volume at $t = 3s$ for each of the methods, and for both time steps used is also tabulated in Table 6.1.

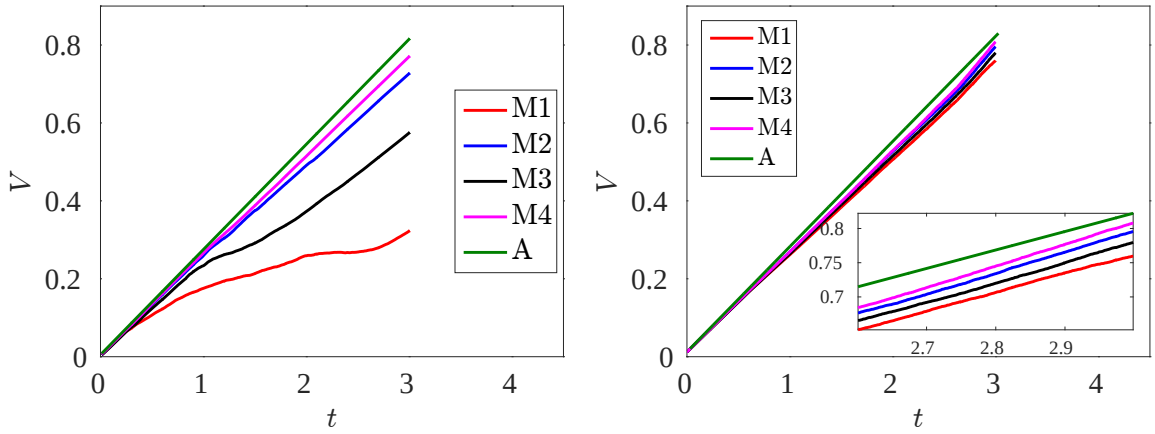


Figure 6.10: Total numerical volume for the tank filling test case. For a large time step of $\Delta t = 0.6 \times 10^{-2}$ (left), and a smaller one of $\Delta t = 0.6 \times 10^{-3}$ (right). M1 stands for the first order method of movement, M2 the second order method, M3 the movement along the streamline, and M4 the movement according to the change of streamlines. A represents the analytical volume.

Table 6.1: Tank filling test case: Relative errors in numerical volume when compared with the analytical value, for different methods of movement at $t = 3s$. The evolution of these errors is shown in Figure 6.10. M1 stands for the first order method of movement, M2 the second order method, M3 the movement along the streamline, and M4 the movement according to the change of streamlines.

	$\Delta t = 0.6 \times 10^{-2}$	$\Delta t = 0.6 \times 10^{-3}$
M1	60.4%	8.4%
M2	10.8%	4.1%
M3	29.5%	6.0%
M4	5.5%	2.6%

6.7 Conclusion

We presented different methods of moving point clouds for fully Lagrangian meshfree methods. The most often used first-order method was shown to produce very inaccurate results. We presented two different methods to improve accuracy of movement. In the first method, each point is moved along approximated streamlines in the neighbourhood of that point. In the second method, the change of streamlines around that point from the previous time step to the current time step is considered, and movement is done based on that. Each of these methods provide a better way to track particle motion, and thus they improve the accuracy of the Lagrangian movement step.

The numerical results match the theoretical expectations. For quasi-stationary flow, moving points along the velocity streamlines produces better results than the other

methods. On the other hand, for situations with rapidly changing flow profiles, movement according to the change of streamlines showed the best results.

A key point of this work is to emphasize that the accuracy and conservative properties of fully Lagrangian meshfree methods can be improved without the need of resorting to artificial, non-physical corrections of particle locations or the physical quantities carried by the particles.

Chapter 7

Conclusion and Outlook

The main goal of this PhD thesis was to explore the potential of meshfree Generalized Finite Difference Methods, and make a first step towards reducing some of their drawbacks. This was achieved in various ways.

Firstly, different meshfree GFDMs were compared. The possibility of extending a particular variant of meshfree GFDMs to solve over-determined problems was presented, with various numerical examples of the same. Existing numerical schemes to simulate fluid flow modeled by the incompressible Navier–Stokes equations were studied, and their drawbacks were analyzed. A new meshfree GFDM scheme for the same was presented, which reduced many of these drawbacks. This solved an over-determined system and numerical examples showed the resultant increase in local accuracy when compared to conventional methods.

Further, the requirements for global conservation in meshfree GFDMs was presented, and a new method to obtain approximate conservation was developed. This was done by introducing the idea of adding a flux balance on locally defined control cells. A constraint of conservation of quantities in the approximation space was added. Numerical simulations were presented which showed improved global conservation properties.

Lastly, improvements in the discretization of the Lagrangian nature of various meshfree methods were investigated. A method to numerically approximate flow streamlines in a meshfree setting was developed. This was used to further develop a new method which numerically approximates the flow pathlines for unsteady flow. The former method was shown to produce very good results for steady flow, while the latter was shown to produce good results for unsteady flow profiles.

Put together, various aspects of meshfree GFDM fluid flow solvers were improved. The three main parts being: improved local accuracy in the PDE solvers, improved global conservation, and improved accuracy in the the Lagrangian discretization.

Implementation and numerical investigations of the new methods formed a significant part of the work done in the duration of the thesis.

All in all, different ways to modify and extend conventional meshfree GFDM approximations were presented in this thesis. While these ideas were only used towards improving fluid flow solvers, they showed a lot of promise and could be used in various other applications. This opens up several possibilities to extend the work of this thesis.

The method to solve over-determined systems can be used to solve a wide variety of problems, to improve solutions in some aspect. Direct extensions of the work presented here in this regard could be applied to the field of magneto hydrodynamics (MHD)

where many of the challenges are similar in nature to the ones in conventional hydrodynamics, which were solved here. The effect of the weighting parameter in this method poses another interesting point of study.

The method of imposing flux conservation on locally defined cells within the usual moving least squares framework can also be extended to solve a large class of problems. Of course, a wide range of different flux functions can be used in this framework. Moreover, the idea behind this method could be extended in a straightforward manner to add different additional constraints to the differential operator definitions.

A further interesting point of study would be to determine how the local flux conservation constraint could also be added in the direct GFDM framework, which would couple two of the major parts of this thesis.

Appendix A

Numerical Differential Operators

Consider the classical GFDM differential operators as defined by the polynomial method in Eq. (2.20) and Eq. (2.21)

$$\sum_{j\in S_i} c^*_{ij} m_j = \partial^*_i m\,, \qquad \forall m \in \mathcal{M}\,, \tag{A.1}$$

$$\min\ J_i = \frac{1}{2}\sum_{j\in S_i}\left(\frac{c^*_{ij}}{W_{ij}}\right)^2\,. \tag{A.2}$$

where $^* = x, y, xx, \Delta, etc.$, and $\mathcal{M}$ is the set of all monomials upto order 2. Note the factor $\frac{1}{2}$ in the minimization for notational convenience. This minimization can be rewritten in matrix vector form as

$$M_i^T \vec{c}^*_i = \vec{b}^*\,, \tag{A.3}$$

$$\min\ J_i = \frac{1}{2}\|W_i^{-1}\vec{c}^*_i\|^2\,, \tag{A.4}$$

where W_i is a diagonal matrix containing the weights, $\vec{c}^*_i$ is a vector containing the stencil coefficients in the neighbourhood of i, $\vec{c}^*_i = (c^*_{i1}, \ldots, c^*_{in})^T$. M_i contains the monomial test functions up to order 2. In two spatial dimensions

$$M_i^T = \begin{pmatrix} \cdots & 1 & \cdots \\ \cdots & \delta x_{ij} & \cdots \\ \cdots & \delta y_{ij} & \cdots \\ \cdots & \frac{1}{2}\delta x_{ij}^2 & \cdots \\ \cdots & \frac{1}{2}\delta y_{ij}^2 & \cdots \\ \cdots & \delta x_{ij}\delta y_{ij} & \cdots \end{pmatrix}\,, \tag{A.5}$$

where $\delta x_{ij} = x_j - x_i$ and $\delta y_{ij} = y_j - y_i$. And the RHS vectors are given by

$$\vec{b}^0 = (1,0,0,0,0,0)^T\,, \tag{A.6}$$

$$\vec{b}^x = (0,1,0,0,0,0)^T\,, \tag{A.7}$$

$$\vec{b}^y = (0,0,1,0,0,0)^T\,, \tag{A.8}$$

$$\vec{b}^\Delta = (0,0,0,1,1,0)^T\,. \tag{A.9}$$

Since the number of neighbours are taken to be more than the number of test functions, Eq. (A.3) forms an under-determined system.

A.1 Lagrange Multipliers

A possible way to determine the stencil coefficients is to use the method of Lagrange multipliers, as done by Seibold [96]. Since

$$\|W_i^{-1}\vec{c}_i^*\|^2 = (\vec{c}_i^*)^T W_i^{-T} W_i^{-1} \vec{c}_i^* \,, \tag{A.10}$$

the Lagrange function for the minimization problem Eq. (A.3) and Eq. (A.4) can be given by

$$\mathcal{L}(\vec{c}_i^*, \vec{\lambda}) = \frac{1}{2} (\vec{c}_i^*)^T W_i^{-T} W_i^{-1} \vec{c}_i^* + \vec{\lambda}^T \left(M_i^T \vec{c}_i^* - \vec{b}^* \right) , \tag{A.11}$$

where $\vec{\lambda}$ is the vector of Lagrange multipliers. The optimal solution for $\vec{c}_i^*$ should satisfy

$$\frac{\partial \mathcal{L}(\vec{c}_i^*, \vec{\lambda})}{\partial \vec{c}_i^*} = 0 \,. \tag{A.12}$$

Using the fact that W_i is a diagonal matrix, and thus $W_i^{-T} = W_i^{-1}$, the Lagrange multipliers for the optimal solution satisfy

$$W_i^{-2} \vec{c}_i^* + \left(\vec{\lambda}^T M_i^T \right)^T = 0 \,. \tag{A.13}$$

This leads to

$$W_i^{-2} \vec{c}_i^* + M_i \vec{\lambda} = 0 \,, \tag{A.14}$$

$$\vec{c}_i^* + W_i^2 M_i \vec{\lambda} = 0 \,, \tag{A.15}$$

Substituting this value of $\vec{c}_i^*$ in Eq. (A.3), we get

$$-M_i^T W_i^2 M_i \vec{\lambda} = \vec{b}^* \tag{A.16}$$

$$\vec{\lambda} = -\left(M_i^T W_i^2 M_i \right)^{-1} \vec{b}^* \,. \tag{A.17}$$

Now, Eq. (A.15) and Eq. (A.17) lead to

$$\vec{c}_i^* = W_i^2 M_i \left(M_i^T W_i^2 M_i \right)^{-1} \vec{b}^* \,. \tag{A.18}$$

However, the computation of the matrix product $M_i^T W_i W_i M_i$ and its inverse can be costly, and thus this approach is usually not desirable.

A.2 QR Decomposition

A more efficient method than the method of Lagrange multipliers to compute the same stencil coefficients is to use a QR-decomposition. For this, Eq. (A.3) is rewritten as

$$M_i^T W_i \left(W_i^{-1} \vec{c}_i^* \right) = \vec{b}^* \,. \tag{A.19}$$

First, a QR-decomposition of $\left(M_i^T W_i\right)^T$ is performed.

$$W_i M_i = Q_i R_i \,, \tag{A.20}$$

$$= \begin{pmatrix} \hat{Q}_i & \hat{\hat{Q}}_i \end{pmatrix} \begin{pmatrix} \hat{R}_i \\ 0 \end{pmatrix} , \tag{A.21}$$

$$= \hat{Q}_i \hat{R}_i \,, \tag{A.22}$$

where $\hat{Q}_i \hat{R}_i$ forms the so-called reduced or thin QR-decomposition. Using this decomposition, Eq. (A.19) leads to

$$R_i^T Q_i^T \left(W_i^{-1} \vec{c}_i^{\,*}\right) = \vec{b}^{\,*} \,. \tag{A.23}$$

Here, the minimum norm solution of $\left(W_i^{-1} \vec{c}_i^{\,*}\right)$ is desired. Now, since Q_i is a unitary transformation

$$\|Q_i^T W_i^{-1} \vec{c}_i^{\,*}\| = \|W_i^{-1} \vec{c}_i^{\,*}\| \,. \tag{A.24}$$

Thus, the minimum norm solution of $Q_i^T W_i^{-1} \vec{c}_i^{\,*}$ is desired, which is given by

$$Q_i^T W_i^{-1} \vec{c}_i^{\,*} = \begin{pmatrix} \hat{R}_i^{-T} \\ 0 \end{pmatrix} \vec{b}^{\,*} \,. \tag{A.25}$$

This leads to

$$W_i^{-1} \vec{c}_i^{\,*} = Q_i \begin{pmatrix} \hat{R}_i^{-T} \\ 0 \end{pmatrix} \vec{b}^{\,*} \,, \tag{A.26}$$

$$= \begin{pmatrix} \hat{Q}_i & \hat{\hat{Q}}_i \end{pmatrix} \begin{pmatrix} \hat{R}_i^{-T} \\ 0 \end{pmatrix} \vec{b}^{\,*} \,, \tag{A.27}$$

$$= \hat{Q}_i \hat{R}_i^{-T} \vec{b}^{\,*} \,, \tag{A.28}$$

$$\vec{c}_i^{\,*} = W_i \hat{Q}_i \hat{R}_i^{-T} \vec{b}^{\,*} \,, \tag{A.29}$$

which is used to compute the differential operator stencils. Note that multiple GFDM stencils can be computed by performing just one decomposition. For example,

$$\left[\vec{c}_i^{\,0}, \vec{c}_i^{\,x}, \vec{c}_i^{\,y}, \vec{c}_i^{\,\Delta}\right] = W_i \hat{Q}_i \hat{R}_i^{-T} \left[\vec{b}^{\,0}, \vec{b}^{\,x}, \vec{b}^{\,y}, \vec{b}^{\,\Delta}\right] \,. \tag{A.30}$$

One of the advantages of this approach is that it avoids performing matrix multiplication $M_i^T W_i W_i M_i$ and the explicit computation of its inverse. In the next section, we show that such an approach can also be used to compute stencils in the direct GFDM approach. Here, the local linear systems are larger (for example, see Eq. (5.24) – Eq. (5.29)), and the advantage of avoiding the computation of the matrix multiplication and its inverse is thus, larger.

A.3 Taylor Expansion Method

Above, we presented the efficient computation of the differential operator stencil in classical GFDM written in the polynomial formulation. Now, we do the same for

the Taylor expansion formulation and show that they are equivalent. Desired is the solution to the minimization problem in Section 2.5.1. In 2D, the problem can be stated as

$$\underbrace{\begin{pmatrix} e_{i1} \\ \vdots \\ e_{in} \end{pmatrix}}_{\vec{E}_i} = \underbrace{\begin{pmatrix} 1 & \delta x_{i1} & \delta y_{i1} & \frac{1}{2}\delta x_{i1}^2 & \frac{1}{2}\delta y_{i1}^2 & \delta x_{i1}\delta y_{i1} \\ \vdots & & \vdots & & & \vdots \\ 1 & \delta x_{in} & \delta y_{in} & \frac{1}{2}\delta x_{in}^2 & \frac{1}{2}\delta y_{in}^2 & \delta x_{in}\delta y_{in} \end{pmatrix}}_{M_i} \underbrace{\begin{pmatrix} u_i \\ (u_x)_i \\ (u_y)_i \\ (u_{xx})_i \\ (u_{yy})_i \\ (u_{xy})_i \end{pmatrix}}_{\vec{a}_i} - \underbrace{\begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}}_{\vec{b}_i}, \tag{A.31}$$

$$\min\, J_i = (M_i\vec{a}_i - \vec{b}_i)^T W_i^2 (M_i\vec{a}_i - \vec{b}_i)\,, \tag{A.32}$$

Note that in an abuse of notation, the RHS vector $\vec{b}_i$ used here, and the one used in the earlier sections $\vec{b}^*$ are not the same. The solution to this minimization can be determined explicitly as follows. The functional J_i is expanded to obtain

$$J_i = \vec{a}_i^T M_i^T W_i^2 M_i \vec{a}_i - \vec{a}_i^T M_i^T W_i^2 \vec{b}_i - \vec{b}_i^T W_i^2 M_i \vec{a}_i + \vec{b}_i^T W_i^2 \vec{b}_i\,. \tag{A.33}$$

Using this, we perform the minimzation

$$\frac{\partial J_i}{\partial \vec{a}_i} = 0\,, \tag{A.34}$$

to obtain

$$2M_i^T W_i^2 M_i \vec{a}_i - M_i^T W_i^2 \vec{b}_i - \left(\vec{b}_i^T W_i^2 M_i\right)^T + 0 = 0\,, \tag{A.35}$$

which gives the solution

$$\vec{a}_i = \left[\left(M_i^T W_i^2 M_i\right)^{-1} M_i^T W_i^2\right] \vec{b}_i\,. \tag{A.36}$$

It is important to note here that the vector $\vec{a}_i$ containing the derivatives u_i, $(u_x)_i$, etc. are not the unknowns. The stencils to compute those derivatives are the unknowns being found by this minimization. i.e. The rows of the matrix $\left[\left(M_i^T W_i^2 M_i\right)^{-1} M_i^T W_i^2\right]$ are of importance to us. We recall the definition the classical GFDM derivatives

$$\tilde{\partial}_i^* u = \sum_{j \in S_i} c_{ij}^* u_j\,. \tag{A.37}$$

Comparing this with Eq. (A.36), we get

$$\begin{pmatrix} \cdots & c_{ij}^0 & \cdots \\ \cdots & c_{ij}^x & \cdots \\ \cdots & c_{ij}^y & \cdots \\ \cdots & c_{ij}^{xx} & \cdots \\ \cdots & c_{ij}^{yy} & \cdots \\ \cdots & c_{ij}^{xy} & \cdots \end{pmatrix} = \left[\left(M_i^T W_i^2 M_i\right)^{-1} M_i^T W_i^2\right]. \tag{A.38}$$

The transpose of the above system can be rewritten to get

$$\left[\vec{c}_i^{\,0}, \vec{c}_i^{\,x}, \vec{c}_i^{\,y}, \vec{c}_i^{\,xx}, \vec{c}_i^{\,yy}, \vec{c}_i^{\,xy}\right] = W_i^2 M_i \left(M_i^T W_i^2 M_i\right)^{-1} I\,, \tag{A.39}$$

where I is an identity matrix of appropriate size. Clearly, this is the same as Eq. (A.18). Note that the following is assumed to get said equivalence for the Laplace stencil $\vec{c}_i^{\,\Delta} = \vec{c}_i^{\,xx} + \vec{c}_i^{\,yy}$. Thus, the Taylor expansion approach and the polynomial approach are equivalent (upto numerical round off errors). It is important to note that this equivalence only holds if the weights used in the minimization are defined as done above. We emphasize this by reiterating that the minimization in the Taylor expansion formulation was written as

$$\min\, J_i = \sum_{j\in S_i} W_{ij}^2 e_{ij}^2\,, \tag{A.40}$$

while that in the polynomial formulation was written as

$$\min\, J_i = \sum_{j\in S_i} \left(\frac{c_{ij}^*}{W_{ij}}\right)^2\,. \tag{A.41}$$

If the weights in the minimizations are written in any other way, as is very often done to simplify notation, including in our earlier work, appropriate care needs to be taken to modify the weight functions to get the same equivalence. We note that in meshfree literature, the minimimization in the Taylor expansion formulation is also commonly written as $\sum \widetilde{W}_{ij} e_{ij}^2$, without the square of the weighting function; whereas that in the polynomial formulation is also commonly written as $\sum \widehat{W}_{ij} c_{ij}^2$.

A.3.1 Direct GFDM

This equivalence can be exploited to obtain a more efficient method to compute the direct GFDM differential operators described in Section 2.6. Similar to the case of classical GFDM explained above, the process of explicitly computing the matrix product followed by the inverse of the local systems can be avoided by performing a QR-decomposition. Consider the 1D over-determined system mentioned in Section 2.6.1

$$a_r u + b_r u_x + c_r u_{xx} + d_r v + e_r v_x + f_r v_{xx} = g_r\,, \qquad r = 1, \cdots, p\,. \tag{A.42}$$

We recall that the function approximation stencils are given by

$$u_i = \sum_{j\in S_i} \alpha_{ij}^u u_j + \sum_{j\in S_i} \beta_{ij}^u v_j + \sum_{r=1}^{p} \zeta_{ir}^u g_r\,, \tag{A.43}$$

$$v_i = \sum_{j\in S_i} \alpha_{ij}^v u_j + \sum_{j\in S_i} \beta_{ij}^v v_j + \sum_{r=1}^{p} \zeta_{ir}^v g_r\,. \tag{A.44}$$

The local least squares systems to obtain these function approximation stencils can be written as

$$\begin{pmatrix} 1 & \cdots & 1 & & & & a_r & \cdots & a_p \\ \delta x_{i1} & \cdots & \delta x_{in} & & & & b_r & \cdots & b_p \\ \frac{1}{2}\delta x_{i1}^2 & \cdots & \frac{1}{2}\delta x_{in}^2 & & & & c_r & \cdots & c_p \\ & & & 1 & \cdots & 1 & d_r & \cdots & d_p \\ & & & \delta x_{i1} & \cdots & \delta x_{in} & e_r & \cdots & e_p \\ & & & \frac{1}{2}\delta x_{i1}^2 & \cdots & \frac{1}{2}\delta x_{in}^2 & f_r & \cdots & f_p \end{pmatrix} \begin{pmatrix} \alpha_{i1}^u & \alpha_{i1}^v \\ \vdots & \vdots \\ \alpha_{in}^u & \alpha_{in}^v \\ \beta_{i1}^u & \beta_{i1}^v \\ \vdots & \vdots \\ \beta_{in}^u & \beta_{in}^v \\ \zeta_{i1}^u & \zeta_{i1}^v \\ \vdots & \vdots \\ \zeta_{ip}^u & \zeta_{ip}^v \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}. \tag{A.45}$$

Note that unlike Eq. (A.39), only the function approximation stencils are desired, and thus, only the relevant columns of the identity matrix are used. This system is solved with a QR-decomposition in a weighted least norm sense as done in Section A.2. This procedure can similarly be extended to higher dimensions and more variables, such as that done in Section 5.3, to obtain stencils in a more efficient way for general problems.

A.4 Diagonal Dominance Proof

As noted in Section 2.5.5, it is often beneficial to control the central stencil value c_{ii}^{Δ}. Eq. (2.44) – Eq. (2.46) can be written in matrix-vector form as

$$\begin{pmatrix} \cdots & \delta_{ij} & \cdots \\ \cdots & 1 & \cdots \\ \cdots & \delta x_{ij} & \cdots \\ \cdots & \delta y_{ij} & \cdots \\ \cdots & \frac{1}{2}\delta x_{ij}^2 & \cdots \\ \cdots & \frac{1}{2}\delta y_{ij}^2 & \cdots \\ \cdots & \delta x_{ij}\delta y_{ij} & \cdots \end{pmatrix} \begin{pmatrix} \vdots \\ c_{ij}^{\Delta} \\ \vdots \end{pmatrix} = \begin{pmatrix} c_{ii}^{\Delta} \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \tag{A.46}$$

where δ_{ij} is the Kronecker delta function. The solution of this system can be written as

$$\vec{c}_i^{\Delta} = W_i \hat{Q}_i \hat{R}_i^{-T} \tilde{b}^{\Delta}, \tag{A.47}$$

where $\tilde{b}^{\Delta}$ is the RHS of Eq. (A.46). Since the QR-decomposition is of the matrix in the LHS of Eq. (A.46), it does not depend on the central stencil value. Thus,

$$\frac{\partial \vec{c}_i^{\Delta}}{\partial c_{ii}^{\Delta}} = \frac{\partial}{\partial c_{ii}^{\Delta}} \left(W_i \hat{Q}_i \hat{R}_i^{-T} \tilde{b}^{\Delta} \right), \tag{A.48}$$

$$= W_i \hat{Q}_i \hat{R}_i^{-T} \frac{\partial}{\partial c_{ii}^{\Delta}} \left(\tilde{b}^{\Delta} \right). \tag{A.49}$$

Now,

$$\frac{\partial}{\partial c_{ii}^{\Delta}}\left(\tilde{b}^{\Delta}\right) = \frac{\partial}{\partial c_{ii}^{\Delta}} \begin{pmatrix} c_{ii}^{\Delta} \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} . \tag{A.50}$$

Thus, $\vec{d}_i^{\Delta} = \frac{\partial \vec{c}_i^{\Delta}}{\partial c_{ii}^{\Delta}}$ is given by the minimum weighted norm solution of

$$\begin{pmatrix} \cdots & \delta_{ij} & \cdots \\ \cdots & 1 & \cdots \\ \cdots & \delta x_{ij} & \cdots \\ \cdots & \delta y_{ij} & \cdots \\ \cdots & \frac{1}{2}\delta x_{ij}^2 & \cdots \\ \cdots & \frac{1}{2}\delta y_{ij}^2 & \cdots \\ \cdots & \delta x_{ij}\delta y_{ij} & \cdots \end{pmatrix} \begin{pmatrix} \vdots \\ d_{ij}^{\Delta} \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} , \tag{A.51}$$

which is equivalent to Eq. (2.52) – Eq. (2.54). This completes the proof for 2D. Extension of the same to 3D is straightforward.

A.5 Consistent Normal Operators

For the numerical differential operators, one of the mimetic properties desired is consistency between the normal direction differential operator and the usual gradient operators. Specifically, we want

$$\vec{c}_i^{\,n} = \sum_{k=x,y,z} n_i^k \vec{c}_i^{\,k} , \qquad \forall i \in \partial\Omega , \tag{A.52}$$

where $\vec{n}_i = (n_i^x, n_i^y, n_i^z)$ is the outward pointing unit normal at point i.

As mentioned in Section 2.5.4, the stencils for the derivative in the normal direction are commonly computed using a diagonal dominance procedure, similar to that done for Laplace stencils. In such situations, the desired consistency, Eq. (A.52), is usually violated. To retain this consistency, the following procedure is carried out.

Let $\vec{c}_i^{\,n}$ be the stencil for the normal operator computed using the diagonal dominance procedure of Eq. (2.38) and Eq. (2.39). Further, let $\vec{c}_i^{\,k_{\text{old}}}$, $k = x, y, z$ be the usual first derivative operators computed by Eq. (2.20) and Eq. (2.21). The first derivative operators are modified to $\vec{c}_i^{\,k}$ as follows

$$\vec{c}_i^{\,k} = \vec{c}_i^{\,k_{\text{old}}} + \vec{\alpha} n_i^k , \qquad k = x, y, z . \tag{A.53}$$

Plugging Eq. (A.53) into Eq. (A.52), we get the following, which we wish to enforce

$$\vec{c}_i^{\,n} = \sum_{k=x,y,z} n_i^k \left(\vec{c}_i^{\,k_{\text{old}}} + \vec{\alpha} n_i^k \right) , \tag{A.54}$$

$$= \sum_{k=x,y,z} n_i^k \vec{c}_i^{\,k_{\text{old}}} + \sum_{k=x,y,z} \vec{\alpha} \left(n_i^k \right)^2 , \tag{A.55}$$

$$= \sum_{k=x,y,z} n_i^k \vec{c}_i^{\,k_{\text{old}}} + \vec{\alpha} , \tag{A.56}$$

$$\vec{\alpha} = \vec{c}_i^{\,n} - \sum_{k=x,y,z} n_i^k \vec{c}_i^{\,k_{\text{old}}} . \tag{A.57}$$

This is substituted back into Eq. (A.53) to obtain the new gradient operators at boundary points. By construction, the new gradient operators are consistent with the normal operator in the sense of Eq. (A.52). Further, they also satisfy the monomial consistency conditions since

$$M_i^T \vec{c}_i^{\,k} = M_i^T \vec{c}_i^{\,k_{\text{old}}} - M_i^T \vec{\alpha} n_i^k , \tag{A.58}$$

and

$$M_i^T \vec{\alpha} = M_i^T \vec{c}_i^{\,n} - \sum_{k=x,y,z} n_i^k M_i^T \vec{c}_i^{\,k_{\text{old}}} , \tag{A.59}$$

$$= 0 . \tag{A.60}$$

Eq. (A.60) is evident from the monomial consistency conditions themselves. In $2D$, M_i^T is given by Eq. (A.5) and the RHS vectors $\vec{b}^* = M_i^T \vec{c}_i^*$ are given by

$$\vec{b}^{\,x} = (0, 1,\ 0,\ 0, 0, 0)^T , \tag{A.61}$$

$$\vec{b}^{\,y} = (0, 0,\ 1,\ 0, 0, 0)^T , \tag{A.62}$$

$$\vec{b}^{\,n} = (0, n_i^x, n_i^y, 0, 0, 0)^T . \tag{A.63}$$

Eq. (A.60) follows directly.

A.6 Non-Symmetric Global Conservative Differential Operators

Using the notation defined earlier in this appendix, Eq. (3.31) – Eq. (3.32) can be rewritten as

$$M_i^T \vec{c}_i^{\,k} = b^k , \quad i = 1, \ldots, N , \tag{A.64}$$

$$\sum_{i=1}^{N} V_i \psi_i \vec{c}_i^{\,k} = \tilde{B}^{\,k} , \tag{A.65}$$

where $\tilde{B}^{\,k}$ contains the appropriate geometric values of Eq. (3.26); and $\psi_i \in \mathbb{R}^{N \times n(i)}$ is defined such that $\vec{\tilde{c}}_i^{\,k} = \psi_i \vec{c}_i^{\,k}$, with $\vec{\tilde{c}}_i^{\,k}$ defined as in Eq. (3.24). Thus, ψ_i contains exactly $n(i)$ non-zero entries, each of them being 1. $\vec{c}_i^{\,k} \in \mathbb{R}^{n(i)}$ contains the neighbour

stencils with neighbours indexed as $(1, \ldots, n(i))$. ψ_i 'stuffs' zeros in $\vec{c}_i^k$ to take it to the globally indexed $\vec{\bar{c}}_i^k \in \mathbb{R}^N$. The above system can be further rewritten as

$$\underbrace{\begin{pmatrix} M_1^T & & \\ & \ddots & \\ & & M_N^T \\ V_1\psi_1 & \ldots & V_N\psi_N \end{pmatrix}}_{\mathcal{K}^T} \underbrace{\begin{pmatrix} \vec{c}_1^k \\ \vdots \\ \vec{c}_N^k \end{pmatrix}}_{\tilde{\mathbf{c}}^k} = \underbrace{\begin{pmatrix} b^k \\ \vdots \\ b^k \\ \tilde{B}^k \end{pmatrix}}_{\mathcal{B}}, \tag{A.66}$$

The linear system Eq. (A.66) is solved by minimizing the weighted $\|\mathcal{W}^{-1}\tilde{\mathbf{c}}^k\|$, as done before, where

$$\mathcal{W} = \begin{pmatrix} W_1 & & \\ & \ddots & \\ & & W_N \end{pmatrix}. \tag{A.67}$$

Clearly,

$$\|\mathcal{W}^{-1}\tilde{\mathbf{c}}^k\|^2 = \sum_{i=1}^{N} \|W_i^{-1}\vec{c}_i^k\|^2 . \tag{A.68}$$

Appendix B

Over-Determined Problems

Throughout this thesis, a significant amount of attention has been paid to over-determined problems. The main point to note here is that in this thesis, the term 'over-determined problem' is used to refer to specific problem formulations which contain more PDEs than variables. Of course, such systems could be and often are manipulated, either algebraically or through a series of differentiations, to obtain a (somewhat) equivalent system which is no longer over-determined. An example of this is given in Section 2.7.3. But what is of interest here is that in the formulation being solved numerically, there are more PDEs than variables. As shown in Sections 2.7.2 and 2.7.3 , and in Chapter 5, solving such over-determined problem formulations numerically can have several advantages.

We note that the mere fact that a system of differential equations is over-determined has little relation with the question of whether the system is well-posed [115]. It is quite possible for a system to have more algebraically independent PDEs than unknowns and still have an infinite number of solutions. A simple example of the same is the unbounded div-curl system in $\mathbb{R}^3$

$$\nabla \times \vec{\psi} = \vec{f}\,, \tag{B.1}$$

$$\nabla \cdot \vec{\psi} = g\,. \tag{B.2}$$

For any solution $\vec{\psi}_0$ of this system, and a harmonic $\vec{\psi}_1$ satisfying $\Delta\vec{\psi}_1 = 0$, $\vec{\psi}_0 + \vec{\psi}_1$ is also a solution of this system. This system has 4 PDEs, and only 3 unknowns, but has infinitely many solutions. Similarly, if Eq. (B.1) were to be considered independently, it would result in a system with 3 PDEs and 3 variables, but infinitely many solutions. Such situations can also occur on bounded domains. For example, imposing a Dirichlet boundary condition on a certain part of the domain boundary is not always enough to make the solution to $\nabla \times \vec{\psi} = \vec{f}$ unique. In such situations, additional closure condition(s) are needed to get a well-posed system. This results in the need to solve an over-determined system numerically. Section 2.7.3 lays out a few more situations where numerically solving an over-determined system can be beneficial.

It must be noted that in the context of systems of differential equations, the terms over, under and properly/exactly determined systems are also used to refer to the solvability of the system of equations [115]. This is used in the sense that an under-determined problem means one that has infinite solutions, and exactly determined problem has a unique solution, while a over-determined system has no solutions. This terminology is not used in this thesis.

Appendix C

Local Centroidal Dual

In Chapter 4, the idea of using locally defined control cells to add additional constraints to the definition of classical GFDM differential operators was presented. These local control cells are also used to assign a geometric volume to each point. Control cells are formed by considering the Delaunay tessellation of all points in the neighbourhood of the central point. Such a local tessellation in 2D is shown in Figure C.1. The simplexes (triangles or tetrahedrons) of the tessellation incident on the central point are the only ones of interest. These are marked in Figure C.1 in blue. A dual graph to these simplexes is used as the desired control cell. We note that in the local tessellations, simplexes with very large circumradius (larger than $3h$) are ignored. This is important at boundaries where a simplex may be formed between multiple boundary points on the outside of the domain. A 2D example of this is shown in Figure C.2.

The Voronoi dual to the Delaunay tessellation consists of edges (in 2D) or faces (in 3D) formed by connecting the circumcenters of the Delaunay simplexes. The use of the Voronoi cell containing the central point as the desired local control cell poses several problems. Most importantly, the circumcenter of a simplex need not always lie within it. This can lead to illogical volume definitions. At boundary points, it is possible that the resultant control cell can extend beyond the simulation domain, which would result in errors in estimating the total volume occupied by the point cloud. An example of this is illustrated in Figure C.3, for the same point configuration and local triangulation as the boundary point case considered in Figure C.1. Ensuring that each circumcenter lies within its simplex imposes a quality criterion on the point cloud to produce 'good' simplexes. We empirically observe that this criterion is significantly stricter than the ones traditionally used in Lagrangian meshfree GFDMs, which are described in Section 2.4, especially at boundary points.

A possible solution to avoid this problem is to use a centroidal dual to the Delaunay tessellation. This is formed by connecting the centroids (barycenters) of each simplex, rather than their circumcenters. By definition, the centroid lies at the mean position of all the nodes of the simplex, and thus, always lies within the simplex. The difference between the Voronoi and the centroidal duals is illustrated in Figure C.3. The figure also illustrates two possible ways to construct the centroidal dual. It can either be formed by connecting the centroids to each other directly, or by connecting them to the mid-point of each Delaunay edge. We use the latter approach to form the control cell used in Chapter 4, due to its ease of generalization to 3D.

At boundary points, the dual needs to be truncated to form a closed cell. This process is illustrated by the green lines in Figure C.3. Those green lines are also used

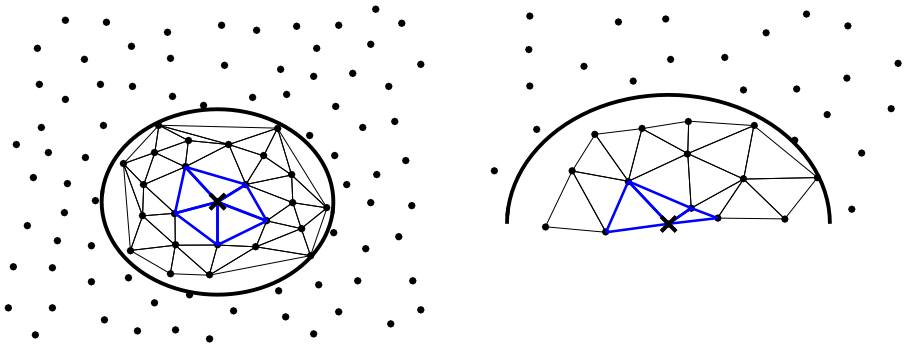


Figure C.1: Local Delaunay triangulation within the support domain. For an interior point (left) and a boundary point (right). The central point is marked with a cross, the support domain is indicated with the circle. The blue triangles are the ones incident on the center point. A dual graph to these is used as the local control cell.

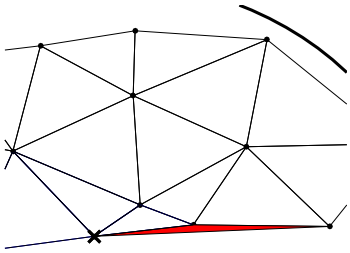


Figure C.2: A triangle being ignored for the boundary point configuration used in Figure C.1. The ignored triangle is marked in red.

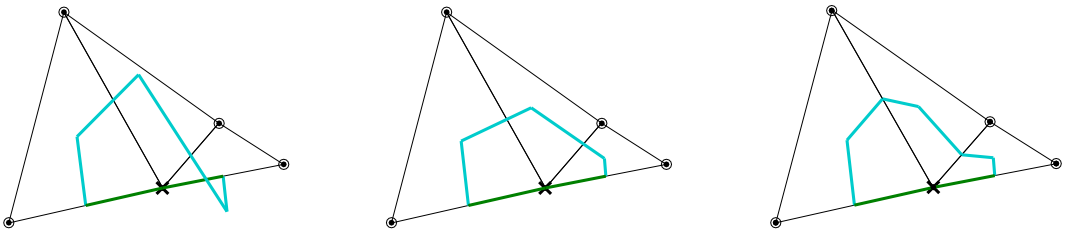


Figure C.3: Voronoi dual (left) and centroidal dual (center and right) to the Delaunay triangles marked in blue in the boundary point configuration in Figure C.1. The light blue lines are part of the semi-infinite duals, which are truncated by the green lines to form a closed cell. The center figure shows the centroidal dual formed by connecting the centroids directly. The right figure shows the one formed by connecting the centroids to the edge midpoints, which is the procedure used for the local control cell. *Remark:* In this and several figures to follow, the scaling of the two dimensions are not the same. As a result, perpendicular lines may appear to not be so in the figure.

to prescribe the geometric area A_i and normal $\vec{n}_i$ associated with the central boundary point. This can lead to one of two problems with the use of the Voronoi dual. If A_i and $\vec{n}_i$ are defined such that they accurately represent the local geometry of the domain boundary, the resultant control cell could be self-intersecting, as shown in Figure C.3. On the other hand, if the control cell is closed such that it is not self-intersecting, the resultant A_i and $\vec{n}_i$ do not accurately capture the local geometry of the domain boundary. This problem no longer persists when the centroidal dual is used.

The use of a centroidal dual has the further advantage that the volume of each simplex is split evenly between all its nodes[1], while in the traditional Voronoi dual, the volume split is uneven.

Details regarding the numerical construction of the centroidal dual are given below.

C.1 Construction of the Centroidal Dual

For a point i, we start with an established local Delaunay tessellation $\mathbb{T}_i$. The set of simplexes incident on point i is denoted by $\mathcal{T}_i$, and the set of all neighbours connected to i with a Delaunay edge is denoted by I_i.

The local control cell is given by the union of a dual to each edge $il, l \in I_i$. This dual $\widetilde{il}$ is formed by connecting the centroids of each simplex incident on the edge il to the center of the edge il. It must be noted that a centroidal dual formed in this way is not a dual to the Delaunay tessellation in a formal graph theoretic sense, but only loosely resembles a dual graph.

C.1.1 2D

- If the Delaunay edge il has two triangles incident on it, the dual $\widetilde{il}$ consists of two edges formed by connecting the centroid of each of these triangles to the mid-point of the edge il. This is shown in Figure C.4 (left).

- If the Delaunay edge il has only one triangle incident on it, the dual $\widetilde{il}$ consists of the single edge formed by connecting the centroid of this triangle to the mid-point of the edge il, as shown in Figure C.4 (center). This case commonly occurs for edges connecting two boundary points.

- For boundary points, the dual is closed by connecting the mid-points of each Delaunay edge which has only one triangle incident on it with the center point itself. This is shown in Figure C.4 (right).

The different parts (edges in 2D) of the dual $\widetilde{il}$ are indexed as ilk, for a varying k. $\vec{n}_{ilk}$ and A_{ilk} are used to denote the unit normal to the k-part of the dual, pointing away from i, and the area (length in 2D) of that part respectively. Then the normal $\vec{n}_{il}$ and area A_{il} for the dual $\widetilde{il}$, which is used to define the fluxes across the dual region,

[1] When the cells are defined based on local tessellations, as done in this thesis, this only holds true if the local cells stitch together to form a global mesh.

are defined as

$$\vec{n}_{il} A_{il} = \sum_k \vec{n}_{ilk} A_{ilk} \,, \tag{C.1}$$

$$A_{il} = \sum_k A_{ilk} \,. \tag{C.2}$$

A similar sum is carried out over the boundary points to define the boundary normal and area. The volume associated with each point is taken to be the volume (area in 2D) of the entire cell. Note that the normal $\vec{n}_{il}$ is *not* made of unit length to ensure that the normal and area definitions are consistent with the cell being closed:

$$\sum_{l \in I_i} \vec{n}_{il} A_{il} = 0 \,, \qquad \text{if } i \notin \partial\Omega \,, \tag{C.3}$$

$$\vec{n_i} A_i + \sum_{l \in I_i} \vec{n}_{il} A_{il} = 0 \,, \qquad \text{if } i \in \partial\Omega \,. \tag{C.4}$$

We note that in the case of a Voronoi dual, the normal $\vec{n}_{il}$ is in the direction of the edge il. This is generally not true for the centroidal dual used here.

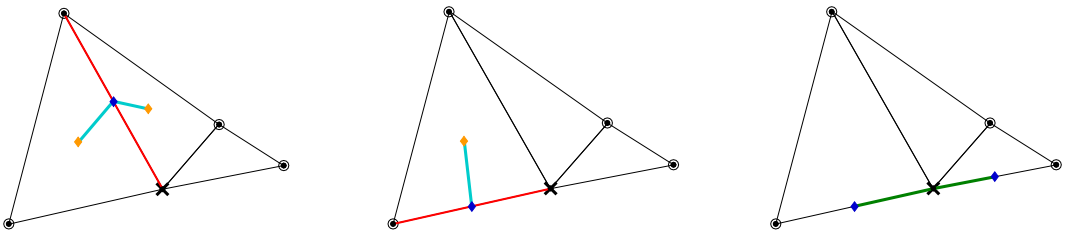


Figure C.4: Construction of the centroidal dual. The left and center figures show the dual to a Delaunay edge. The Delaunay edge is marked in red, while its dual is marked in light blue. The centroid of each triangle incident on the edge is marked with an orange diamond, while the center of the edge itself is marked with a dark blue diamond. The left figure shows the case of two triangles incident on the edge, while the center shows the case of one triangle. The right figure shows the closure of the cell by the boundary half-edges.

C.1.2 3D

The dual in 3D is formed by generalizing the 2D procedure mentioned above. The dual $\widetilde{il}$ to each Delaunay edge il is formed by connecting the centroids of all the tetrahedrons incident on il. Since all these centroids need not lie in a plane, the centroids need to be connected with the mid-point of the edge il, in a way similar to that done in the 2D case above. In 2D, $\widetilde{il}$ consisted of multiple edges sharing a common vertex. Similarly, in 3D, $\widetilde{il}$ consists of multiple triangles sharing a common vertex, with several pairs of triangles sharing a common edge. Each triangular part of $\widetilde{il}$ corresponds to a pair of tetrahedra incident on il which share a common face. The centroids of each of such a

pair of tetrahedra are connected to each other, and to the mid-point of the edge il to form the dual triangle. The union of all such triangles gives $\widetilde{il}$. For boundary points, the cells are closed in a manner similar to that done in 2D. Normals and areas are also given in a way similar to 2D.

$$\vec{n}_{il} A_{il} = \sum_k \vec{n}_{ilk} A_{ilk} \,, \tag{C.5}$$

$$A_{il} = \sum_k A_{ilk} \,, \tag{C.6}$$

where the summation index k is over all triangular parts of $\widetilde{il}$. Once again, $\vec{n}_{il}$ is *not* made of unit length. It must be noted that taking such a centroidal dual no longer guarantees that the local cells will be convex.

Bibliography

[1] A. S. Almgren, J. B. Bell, and W. G. Szymczak. A numerical method for the incompressible navier-stokes equations based on an approximate projection. *SIAM Journal on Scientific Computing*, 17(2):358–369, 1996.

[2] A. R. E. Antunes, P. R. M. Lyra, R. B. Willmersdorf, and S. M. A. Bastos. An implicit monolithic formulation based on finite element formulation for incompressible navier–stokes equations. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 37(1):199–210, 2015.

[3] T. Arbogast and C.-S. Huang. A fully conservative eulerian-lagrangian method for a convection-diffusion problem in a solenoidal field. *J. Comput. Phys.*, 229(9):3415–3427, May 2010.

[4] J. Arteaga-Arispe and J. M. Guevara-Jordan. A conservative finite difference scheme for static diffusion equation. *Divulgaciones Matemáticas*, 16(1):39–54, 2008.

[5] N. S. Atluri and T. Zhu. A new meshless local petrov-galerkin (mlpg) approach in computational mechanics. *Computational Mechanics*, 22(2):117–127, 1998.

[6] Y. Bazilevs and T. Hughes. Weak imposition of dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36(1):12 – 26, 2007. Challenges and Advances in Flow Simulation and Modeling.

[7] J. B. Bell, P. Colella, and L. H. Howell. An efficient second-order projection method for viscous incompressible flow. In *Proceedings of the Tenth AIAA Computational Fluid Dynamics Conference, AIAA*, volume 360, 1991.

[8] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139(1–4):3–47, 1996.

[9] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37(2):229–256, 1994.

[10] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[11] H. Bhatia, G. Norgard, V. Pascucci, and P.-T. Bremer. The helmholtz-hodge decomposition-a survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1386–1404, Aug. 2013.

[12] B. Boroomand, A. A. Tabatabaei, and E. Oñate. Simple modifications for stabilization of the finite point method. *International Journal for Numerical Methods in Engineering*, 63(3):351–379, 2005.

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

[14] P. G. Buning. Sources of error in the graphical analysis of cfd results. *Journal of Scientific Computing*, 3(2):149–164, 1988.

[15] J. E. Castillo and R. D. Grone. A matrix analysis approach to higher-order approximations for divergence and gradients satisfying a global conservation law. *SIAM Journal on Matrix Analysis and Applications*, 25(1):128–142, 2003.

[16] J. Chen and J. Beraun. A generalized smoothed particle hydrodynamics method for nonlinear dynamic problems. *Computer Methods in Applied Mechanics and Engineering*, 190(1-2):225 – 239, 2000.

[17] A. J. Chorin. Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762, 1968.

[18] A. Crespo, J. Domínguez, B. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. García-Feal. DualSPHysics: Open-source parallel CFD solver based on smoothed particle hydrodynamics (SPH). *Computer Physics Communications*, 187:204 – 216, 2015.

[19] S. J. Cummins and M. Rudman. An sph projection method. *Journal of Computational Physics*, 152(2):584 – 607, 1999.

[20] D. Diachin and J. Herzog. Analytic streamline calculations on linear tetrahedra. In *13th Computational Fluid Dynamics Conference*, page 1975, 1997.

[21] O. Diyankov. Uncertain grid method for numerical solution of PDE. `http://www.neuroksoftware.com/resources/ugm_final.pdf`.

[22] J. M. Domínguez, A. J. C. Crespo, M. Gómez-Gesteira, and J. C. Marongiu. Neighbour lists in smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 67(12):2026–2042, 2011.

[23] J. Donea, S. Giuliani, and J. Halleux. An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33(1):689 – 723, 1982.

[24] C. Drumm, S. Tiwari, J. Kuhnert, and H.-J. Bart. Finite pointset method for simulation of the liquid - liquid flow field in an extractor. *Computers & Chemical Engineering*, 32(12):2946 – 2957, 2008.

[25] V. Dyachenko. The free point method for problems of continuous media. *Computer Methods in Applied Mechanics and Engineering*, 2(3):265 – 277, 1973.

[26] R. Eymard, T. Gallouët, and R. Herbin. Finite volume methods. In *Solution of Equation in $\mathbb{R}^n$ (Part 3), Techniques of Scientific Computing (Part 3)*, volume 7 of *Handbook of Numerical Analysis*, pages 713 – 1018. Elsevier, Amsterdam, 2000.

[27] J. Fang and A. Parriaux. A regularized lagrangian finite point method for the simulation of incompressible viscous flows. *Journal of Computational Physics*, 227(20):8894 – 8908, 2008.

[28] G. E. Fasshauer and J. G. Zhang. On choosing "optimal" shape parameters for RBF approximation. *Numerical Algorithms*, 45(1):345–368, 2007.

[29] A. Fedoseyev, M. Friedman, and E. Kansa. Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary. *Computers & Mathematics with Applications*, 43(3):439 – 455, 2002.

[30] N. Flyer, G. B. Wright, and B. Fornberg. Radial basis function-generated finite differences: A mesh-free method for computational geosciences. In W. Freeden, M. Z. Nashed, and T. Sonar, editors, *Handbook of Geomathematics*, pages 1–30, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[31] B. Fornberg and C. Piret. On choosing a radial basis function and a shape parameter when solving a convective PDE on a sphere. *Journal of Computational Physics*, 227(5):2758 – 2780, 2008.

[32] L. Gavete, M. Gavete, and J. Benito. Improvements of generalized finite difference method and comparison with other meshless method. *Applied Mathematical Modelling*, 27(10):831 – 847, 2003.

[33] A. Ghosh and S. Deshpande. Least squares kinetic upwind method for inviscid compressible flows. *AIAA paper*, 1735:1995, 1995.

[34] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375, 1977.

[35] Z. Guo, B. Shi, and N. Wang. Lattice BGK model for incompressible Navier–Stokes equation. *Journal of Computational Physics*, 165(1):288 – 306, 2000.

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

[37] D. Hietel, M. Junk, J. Kuhnert, and S. Tiwari. Meshless methods for conservation laws. In G. Warnecke, editor, *Analysis and Numerics for Conservation Laws*, pages 339–362, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[38] D. Hietel, K. Steiner, and J. Struckmeier. A finite-volume particle method for compressible flows. *Mathematical Models and Methods in Applied Sciences*, 10(09):1363–1382, 2000.

[39] P. F. Hopkins. A new class of accurate, mesh-free hydrodynamic simulation methods. *Monthly Notices of the Royal Astronomical Society*, 450(1):53–110, 2015.

[40] F. Hu, T. Matsunaga, T. Tamai, and S. Koshizuka. An ALE particle method using upwind interpolation. *Computers & Fluids*, 145:21 – 36, 2017.

[41] S. Idelsohn, N. Nigro, A. Limache, and E. Oñate. Large time-step explicit integration method for solving problems with dominant convection. *Computer Methods in Applied Mechanics and Engineering*, 217-220:168 – 185, 2012.

[42] S. R. Idelsohn and E. Oñate. The challenge of mass conservation in the solution of free-surface flows with the fractional-step method: Problems and solutions. *International Journal for Numerical Methods in Biomedical Engineering*, 26(10):1313–1330, 2010.

[43] O. Iliev and S. Tiwari. A generalized (meshfree) finite difference discretization for elliptic interface problems. In *Revised Papers from the 5th International Conference on Numerical Methods and Applications*, NMA '02, pages 488–497, London, UK, 2003. Springer-Verlag.

[44] A. Iske. On the construction of mass conservative and meshless adaptive particle advection methods. In V. M. A. Leitao, C. J. S. Alves, and C. Armando Duarte, editors, *Advances in Meshfree Techniques*, pages 169–186, Dordrecht, 2007. Springer Netherlands.

[45] A. Iske and M. Käser. Conservative semi-lagrangian advection on adaptive unstructured meshes. *Numerical Methods for Partial Differential Equations*, 20(3):388–411, 2004.

[46] R. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40 – 65, 1986.

[47] A. Jefferies, J. Kuhnert, L. Aschenbrenner, and U. Giffhorn. Finite pointset method for the simulation of a vehicle travelling through a body of water. In M. Griebel and A. M. Schweitzer, editors, *Meshfree Methods for Partial Differential Equations VII*, pages 205–221, Cham, 2015. Springer International Publishing.

[48] A. Katz and A. Jameson. Meshless scheme based on alignment constraints. *AIAA journal*, 48(11):2501–2511, 2010.

[49] A. J. Katz. *Meshless methods for computational fluid dynamics*. PhD thesis, Stanford University, Stanford, 2009.

[50] A. Khayyer, H. Gotoh, and Y. Shimizu. Comparative study on accuracy and conservation properties of two particle regularization schemes and proposal of an optimized particle shifting scheme in ISPH context. *Journal of Computational Physics*, 332:236 – 256, 2017.

[51] P. Kipfer, F. Reck, and G. Greiner. Local exact particle tracing on unstructured grids. *Computer Graphics Forum*, 22(2):133–142, 2003.

[52] R. A. Klausen, A. F. Rasmussen, and A. F. Stephansen. Velocity interpolation and streamline tracing on irregular geometries. *Computational Geosciences*, 16(2):261–276, 2012.

[53] H. Kruggel-Emden, M. Sturm, S. Wirtz, and V. Scherer. Selection of an appropriate time integration scheme for the discrete element method (DEM). *Computers & Chemical Engineering*, 32(10):2263 – 2279, 2008.

[54] K. Krupchyk, W. M. Seiler, and J. Tuomela. Overdetermined elliptic systems. *Foundations of Computational Mathematics*, 6(3):309–351, 2006.

[55] J. Kuhnert. *General Smoothed Particle Hydrodynamics*. PhD thesis, Kaiserslautern University, Kaiserslautern, 1999.

[56] J. Kuhnert. An upwind finite pointset method (FPM) for compressible euler and navier-stokes equations. In M. Griebel and M. A. Schweitzer, editors, *Meshfree Methods for Partial Differential Equations*, pages 239–249, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[57] J. Kuhnert. Meshfree numerical scheme for time dependent problems in fluid and continuum mechanics. In S. Sundar, editor, *Advances in PDE Modeling and Computation*, pages 119–136, New Delhi, 2014. Anne Books.

[58] E. Kwan-yu Chiu, Q. Wang, R. Hu, and A. Jameson. A conservative meshfree scheme and generalized framework for conservation laws. *SIAM Journal on Scientific Computing*, 34(6):A2896–A2916, 2012.

[59] E. Kwan-yu Chiu, Q. Wang, and A. Jameson. A conservative meshless scheme: General order formulation and application to euler equations. In *Proceedings of the 49th AIAA Aerospace Sciences Meeting*, pages 651–669, Orlando, 2011.

[60] S. Li and W. K. Liu. Meshfree and particle methods and their applications. *Applied Mechanics Review*, 55:1–34, 2002.

[61] K. Lipnikov, G. Manzini, and M. Shashkov. Mimetic finite difference method. *Journal of Computational Physics*, 257:1163–1227, 2014.

[62] K. Lipnikov, M. Shashkov, and D. Svyatskiy. The mimetic finite difference discretization of diffusion problem on unstructured polyhedral meshes. *Journal of Computational Physics*, 211(2):473 – 491, 2006.

[63] T. Liszka and J. Orkisz. Special issue-computational methods in nonlinear mechanics the finite difference method at arbitrary irregular grids and its application in applied mechanics. *Computers and Structures*, 11(1):83 – 95, 1980.

[64] G.-R. Liu. *Meshfree methods: moving beyond the finite element method*. Taylor & Francis, Boca Raton, FL, USA, 2009.

[65] G.-R. Liu and Y.-T. Gu. *An introduction to meshfree methods and their programming*. Springer Science & Business Media, 2005.

[66] G.-R. Liu and M. B. Liu. *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific, Singapore, 2003.

[67] G.-R. Liu and G.-Y. Zhang. *Smoothed point interpolation methods: G space theory and weakened weak forms*. World Scientific, Singapore, 2013.

[68] M. Liu, J. Shao, and J. Chang. On the treatment of solid boundary in smoothed particle hydrodynamics. *Science China Technological Sciences*, 55(1):244–254, 2011.

[69] M. Liu, W. Xie, and G. Liu. Modeling incompressible flows using a finite particle method. *Applied Mathematical Modelling*, 29(12):1252 – 1270, 2005.

[70] W. K. Liu, S. Jun, and Y. F. Zhang. Reproducing kernel particle methods. *International Journal for Numerical Methods in Fluids*, 20(8-9):1081–1106, 1995.

[71] R. Löhner and E. Oñate. A general advancing front technique for filling space with arbitrary objects. *International Journal for Numerical Methods in Engineering*, 61(12):1977–1991, 2004.

[72] A. Lopes and K. Brodlie. Accuracy in 3d particle tracing. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization: Algorithms, Applications and Numerics*, pages 329–341, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[73] M. Luo, C. G. Koh, W. Bai, and M. Gao. A particle method for two-phase flows with compressible air pocket. *International Journal for Numerical Methods in Engineering*, 108:695–721, Nov. 2016.

[74] F. Maria Denaro. On the application of the helmholtz–hodge decomposition in projection methods for incompressible flows with general boundary conditions. *International Journal for Numerical Methods in Fluids*, 43(1):43–69, 2003.

[75] B. Metsch. *Algebraic Multigrid (AMG) for Saddle Point Systems*. PhD thesis, Institut für Numerische Simulation, Universität Bonn, Bonn, July 2013.

[76] S. Milewski. Meshless finite difference method with higher order approximation—applications in mechanics. *Archives of Computational Methods in Engineering*, 19(1):1–49, 2012.

[77] A. Möller and J. Kuhnert. Simulation of the glass flow inside a floating process / Simulation de l'écoulement du verre dans le procédé float. *Revue Verre*, 13(5):28–30, 2007.

[78] J. Monaghan. On the problem of penetration in particle methods. *Journal of Computational Physics*, 82(1):1 – 15, 1989.

[79] M. Mongillo. Choosing basis functions and shape parameters for radial basis function methods. *SIAM Undergraduate Research Online*, 4:190–209, 2011.

[80] P. Nadukandi. Numerically stable formulas for a particle-based explicit exponential integrator. *Computational Mechanics*, 55(5):903–920, 2015.

[81] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, 10(5):307–318, 1992.

[82] G. M. Nielson and I.-H. Jung. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):360–372, Oct 1999.

[83] E. Oñate, S. Idelsohn, O. C. Zienkiewicz, and R. L. Taylor. A finite point method in computational mechanics. applications to convective transport and fluid flow. *International Journal for Numerical Methods in Engineering*, 39(22):3839–3866, 1996.

[84] E. Oñate, S. R. Idelsohn, F. Del Pin, and R. Aubry. The particle finite element method-an overview. *International Journal of Computational Methods*, 01(02):267–307, 2004.

[85] J. Onderik and R. Ďurikovič. Efficient neighbor search for particle-based fluids. *Journal of the Applied Mathematics, Statistics and Informatics (JAMSI)*, 4(1):29–43, 2008.

[86] G. Pahar and A. Dhar. A robust volume conservative divergence-free ISPH framework for free-surface flow problems. *Advances in Water Resources*, 96:423 – 437, 2016.

[87] G. Pahar and A. Dhar. Robust boundary treatment for open-channel flows in divergence-free incompressible SPH. *Journal of Hydrology*, 546:464 – 475, 2017.

[88] S. Patankar and D. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787 – 1806, 1972.

[89] C. Praveen and S. M. Deshpande. Kinetic meshless method for compressible flows. *International Journal for Numerical Methods in Fluids*, 55(11):1059–1089, 2007.

[90] P. Randles and L. Libersky. Smoothed particle hydrodynamics: Some recent improvements and applications. *Computer Methods in Applied Mechanics and Engineering*, 139(1-4):375 – 408, 1996.

[91] E. O. Reséndiz-Flores and F. R. Saucedo-Zendejo. Two-dimensional numerical simulation of heat transfer with moving heat source in welding using the finite pointset method. *International Journal of Heat and Mass Transfer*, 90:239 – 245, 2015.

[92] I. S. Strub and A. M. Bayen. Weak formulation of boundary conditions for scalar conservation laws: an application to highway traffic modelling. *International Journal of Robust and Nonlinear Control*, 16(16):733–748, 2006.

[93] A. Saha, G. Biswas, and K. Muralidhar. Three-dimensional study of flow past a square cylinder at low reynolds numbers. *International Journal of Heat and Fluid Flow*, 24(1):54 – 66, 2003.

[94] B. Šarler and R. Vertnik. Meshfree explicit local radial basis function collocation method for diffusion problems. *Computers & Mathematics with Applications*, 51(8):1269 – 1282, 2006.

[95] M. Schweitzer. Meshfree and generalized finite element methods. *Institute for Numerical Simulation, University of Bonn, Habilitation*, 2008.

[96] B. Seibold. *M-Matrices in Meshless Finite Difference Methods*. PhD thesis, Kaiserslautern University, 2006.

[97] B. Seibold. Minimal positive stencils in meshfree finite difference methods for the poisson equation. *Computer Methods in Applied Mechanics and Engineering*, 198(3-4):592 – 601, 2008.

[98] T. Seifarth. Gitterfreie lösungsschemen zur numerischen lösung von transportvorgängen. Master's thesis, University of Kassel, Kassel, 2014.

[99] T. Seifarth. *Numerische Algortihmen für gitterfreie Methoden zur Lösung von Transportproblemen*. PhD thesis, University of Kassel, Kassel, 2017.

[100] S. Sen, S. Mittal, and G. Biswas. Flow past a square cylinder at low reynolds numbers. *International Journal for Numerical Methods in Fluids*, 67(9):1160–1174, 2011.

[101] M. S. Shadloo, A. Zainali, and M. Yildiz. Simulation of single mode rayleigh–taylor instability by sph method. *Computational Mechanics*, 51(5):699–715, 2013.

[102] M. Shashkov. *Conservative finite-difference methods on general grids*, volume 6. CRC press, Boca Raton,FL, 1995.

[103] A. Skillen, S. Lind, P. K. Stansby, and B. D. Rogers. Incompressible smoothed particle hydrodynamics (SPH) with reduced temporal noise and generalised fickian smoothing applied to body-water slam and efficient wave-body interaction. *Computer Methods in Applied Mechanics and Engineering*, 265:163 – 173, 2013.

[104] F. S. Sousa, C. M. Oishi, and G. C. Buscaglia. Spurious transients of projection methods in microflow simulations. *Computer Methods in Applied Mechanics and Engineering*, 285:659 – 693, 2015.

[105] D. Sridar and N. Balakrishnan. An upwind finite difference scheme for meshless solvers. *Journal of Computational Physics*, 189(1):1 – 29, 2003.

[106] A. Staniforth and J. Côté. Semi-lagrangian integration schemes for atmospheric models-a review. *Monthly weather review*, 119(9):2206–2223, 1991.

[107] P. Suchde and J. Kuhnert. Point cloud movement for fully lagrangian meshfree methods. *Journal of Computational and Applied Mathematics*, 340:89 – 100, 2018.

[108] P. Suchde, J. Kuhnert, S. Schröder, and A. Klar. A flux conserving meshfree method for conservation laws. *International Journal for Numerical Methods in Engineering*, 112(3):238–256, 2017.

[109] P. Suchde, J. Kuhnert, and S. Tiwari. On meshfree GFDM solvers for the incompressible Navier–Stokes equations. *Computers & Fluids*, 165:1 – 12, 2018.

[110] S. Tiwari, A. Klar, and S. Hardt. Numerical simulation of wetting phenomena by a meshfree particle method. *Journal of Computational and Applied Mathematics*, 292:469 – 485, 2016.

[111] S. Tiwari and J. Kuhnert. Finite pointset method based on the projection method for simulations of the incompressible navier-stokes equations. In M. Griebel and M. A. Schweitzer, editors, *Meshfree Methods for Partial Differential Equations*, pages 373–387. Springer, 2002.

[112] S. Tiwari and J. Kuhnert. A meshfree method for incompressible fluid flows with incorporated surface tension. *Revue Européenne des Éléments Finis*, 11(7-8):965–987, 2002.

[113] A. Tramecon and J. Kuhnert. Simulation of advanced folded airbags with VPS-PAMCRASH/FPM: Development and validation of turbulent flow numerical simulation techniques applied to curtain bag deployments. In *SAE Technical Paper*, Warrendale, PA, USA, 2013. SAE International.

[114] J. Tu, G. H. Yeoh, and C. Liu. *Computational fluid dynamics: a practical approach*. Butterworth-Heinemann, Newton, MA, USA, 2012.

[115] J. Tuomela. General systems of PDEs: from overdetermined systems to involutive systems. `http://www.i3m.univ-montp2.fr/CANUM2003/COMMUNICATIONS/CONF_PLEN/jukka.tuomela.pdf`, 2003.

[116] E. Uhlmann, R. Gerstenberger, and J. Kuhnert. Cutting simulation with the meshfree finite pointset method. *Procedia CIRP*, 8:391 – 396, 2013.

[117] H. A. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[118] D. Violeau and B. D. Rogers. Smoothed particle hydrodynamics (SPH) for free-surface flows: past, present and future. *Journal of Hydraulic Research*, 54(1):1–26, 2016.

[119] R. Xu, P. Stansby, and D. Laurence. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. *Journal of Computational Physics*, 228(18):6703 – 6725, 2009.

[120] Q. Zhang. A fourth-order approximate projection method for the incompressible navier - stokes equations on locally-refined periodic domains. *Applied Numerical Mathematics*, 77:16 – 30, 2014.

[121] T. Zhang, Y.-F. Ren, C.-M. Fan, and P.-W. Li. Simulation of two-dimensional sloshing phenomenon by generalized finite difference method. *Engineering Analysis with Boundary Elements*, 63:82 – 91, 2016.

[122] D. Zhou, B. Seibold, D. Shirokoff, P. Chidyagwai, and R. R. Rosales. Meshfree finite differences for vector poisson and pressure poisson equations with electric boundary conditions. In M. Griebel and M. A. Schweitzer, editors, *Meshfree Methods for Partial Differential Equations VII*, pages 223–246, Cham, 2015. Springer International Publishing.

Academic CV

10/2014 -01/2018	Student at University of Kaiserslautern, Germany *PhD in Mathemtics*
10/2013 -01/2018	Stay at Fraunhofer Institute for Industrial Mathematics ITWM, Kaiserslautern, Germany
08/2012 -06/2013	Research Assistant at TIFR-CAM, Bangalore, India
08/2008 - 07/2013	Student at BITS-Pilani, India *Master of Sciences in Mathematics* *Bachelor of Engineering in Mechanical Engineering*
06/2008	Graduated High School at RN Podar, Mumbai, India

Akademischer Lebenslauf

10/2014 -01/2018	Student Technische Universität Kaiserslautern, Deutschland *Doktorand am Fachbereich Mathemtik*
10/2013 -01/2018	Aufenthalt Fraunhofer-Institut für Techno und Wirtschafts-mathematik ITWM, Kaiserslautern, Deutschland
08/2012 -06/2013	Wissenschaftlicher Assistent TIFR-CAM, Bangalore, Indien
08/2008 - 07/2013	Student BITS-Pilani, Pilani, Indien *Master of Sciences in Mathematics* *Bachelor of Engineering in Mechanical Engineering*
06/2008	Schulabschluss RN Podar, Mumbai, Indien

The goal of this PhD is to improve various aspects of meshfree Generalized Finite Difference Methods (GFDMs). In this thesis, different meshfree GFDMs are compared, and their potential to solve over-determined problems is presented. A new method is presented that introduces conservation of fluxes in a meshfree setting, which reduces the problem of lack of conservation that has plagued meshfree methods. Special attention is paid on the application of meshfree GFDMs to simulate fluid flow modeled by the incompressible Navier − Stokes equations. A new meshfree GFDM scheme for the same is presented which improves local accuracy, and shows better approximations to the mass conservation condition. Further, different aspects of meshfree Lagrangian frameworks are studied, and new methods to improve accuracy in the Lagrangian movement process are also presented.

ISBN 978-3-8396-1325-2

9 783839 613252

FRAUNHOFER VERLAG