A CONSERVATIVE MESHLESS FRAMEWORK FOR
CONSERVATION LAWS WITH APPLICATIONS IN
COMPUTATIONAL FLUID DYNAMICS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

(Edmond) Kwan Yu Chiu
April 2011
This dissertation is online at: http://purl.stanford.edu/fn845yv3048

2011 by Kwan Yu Chiu. All Rights Reserved.
Re-distributed by Stanford University under license with the author.
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Antony Jameson, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Gianluca Iaccarino

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Robert MacCormack

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.
Abstract

In recent decades, advancement in computing technology and hardware has made numerical simulations more applicable than ever before to large and complex problems. Computational simulations have also become an integral part of the design process of engineering systems. However, mesh generation, which is essential to the discretization process for most existing numerical methods, has often remained the bottleneck of the simulation process, especially when domain boundaries are characterized by non-trivial geometry.

As a result, many have developed meshless algorithms to circumvent mesh generation. These algorithms have been applied with various levels of success to a wide variety of problems in computational mechanics. Unfortunately, because of their loose connectivity requirements, meshless methods face a different set of challenges, the most fundamental of which is the lack of formal conservation at the discrete level. This lack of conservation can lead to unpredictable errors in numerical solutions, especially in the presence of discontinuities, e.g. shock waves that commonly occur in compressible flow applications.

To address this important issue, this thesis focuses on the formulation and implementation of a novel conservative meshless scheme and its applications in computational fluid dynamics (CFD).

The scheme is formulated based on obtaining derivative approximations using function values and generated coefficients satisfying a set of reciprocity and polynomial consistency conditions. These conditions lead to a finite-volume-like meshless scheme that satisfies discrete local conservation properties, the first scheme of such nature documented in the literature. To generate the required coefficients, a global
linear minimum-norm or quadratic programming problem is formulated by expressing the consistency conditions as an underdetermined linear system of constraints. Two algorithms are designed to solve the global problem, resulting in coefficients that, in addition to satisfying the necessary constraints, also minimize an upper bound of a representation of the global discretization error. Linear advection test problems confirm convergence of the algorithm.

A generalization of the derivative approximation is introduced to allow the use of arbitrary consistent interface values in the derivative operator while maintaining discrete conservation. In the context of solving conservation laws, this creates a flexible framework within which a wide variety of numerical flux schemes, such as those previously developed for finite volume discretization, can be used. This also means that the new meshless framework can be adopted at minimal costs in processes using traditional numerical tools. The practicality of this new framework is demonstrated by solving classic compressible flow problems using, without modifications, a piece of software designed for finite volume discretization. The meshless numerical results compare well with those obtained using meshed finite volume discretizations and other meshless schemes, highlighting the validity of the new framework and its potential to be applied to problems of greater complexity and scale.
I feel very fortunate that this thesis has benefited from the contribution of so many people in various ways. First and foremost, my adviser, Professor Antony Jameson, has always offered unparalleled support and mentorship. From the beginning, he encouraged me to pursue research that I am truly passionate about. Dedicated to and passionate about CFD himself, he backed up his encouragement by providing me with ample freedom and excellent resources. With his rich experience in the field, he also gave me valuable advice that helped advance my research. It is an honor to have had an advisor who treats the benefits and needs of his students as priorities.

Professor Qiqi Wang at the Massachusetts Institute of Technology has also been a key collaborator and mentor. With his breadth and depth of knowledge, his suggestions and ideas not only helped with the understanding of the meshless scheme at the important early stages of development, but also helped steer the current research to target a large class of problems, magnifying the impact of the work in this thesis.

In addition, the resources Professor Michael Saunders provided on optimization were instrumental in improving the speed and practicality of the current work. The interests and suggestions from Professors Bob MacCormack, Professor Gianluca Iaccarino and Professor Sanjiva Lele all helped push the work in this thesis further.

A key inspiration for me to research in meshless algorithms came from a precious opportunity to visit the BMW Sauber F1 Team in 2008. I am extremely grateful to Mr. Torbjörn Larsson and Mr. Willem Toet for giving me the privilege to work with a brilliant and welcoming team of engineers in the Computational Fluid Dynamics Group and Aerodynamics Department. In particular, it was a pleasure to work closely with Dr. Francesco Del Citto on preprocessing methodology. The experience and
knowledge I gained would also help shape the development path of the conservative meshless framework.

Although not directly involved in my thesis research, others have also contributed to this thesis in significant ways. My family, whom I can always count on, have always been very supportive of my decision to pursue a career in engineering and motorsports. Fellow students in the Aerospace Computing Laboratory must not go unmentioned. They contributed to a helpful and pleasant environment for all of us in our quests to help advance science and engineering. A word of thanks also goes to my roommates and friends with whom I have tremendously enjoyed exploring and experiencing lives as graduate students.
# Contents

Abstract \hspace{1cm} v

Acknowledgments \hspace{1cm} vii

1 Introduction \hspace{1cm} 1

1.1 The Challenge of Domain Discretization \hspace{1cm} 1

1.2 Definition and Survey of Meshless Methods \hspace{1cm} 2

1.2.1 Meshless Methods Based on Strong Forms \hspace{1cm} 3

1.2.2 Meshless Methods Based on Weak Forms \hspace{1cm} 4

1.2.3 Hybrid Meshless Methods \hspace{1cm} 4

1.3 Challenges Faced by Meshless Methods \hspace{1cm} 5

1.4 A Preview of the Current Thesis \hspace{1cm} 7

2 A Meshless Differentiation Operator \hspace{1cm} 9

2.1 Formulation \hspace{1cm} 9

2.2 Global Conservation and Mimetic Properties \hspace{1cm} 10

2.3 Local Conservation \hspace{1cm} 13

2.4 Existence of Meshless Coefficients — Necessary Conditions \hspace{1cm} 16

3 Constraints in Matrix Form \hspace{1cm} 19

3.1 Elementary Definitions from Graph Theory \hspace{1cm} 19

3.2 Matrix Expressions \hspace{1cm} 20

3.3 Equivalent Expressions \hspace{1cm} 24

3.3.1 Delta Form \hspace{1cm} 24
3.3.2 Point-Centered Form ............................................. 25

4 Operator Construction .............................................. 27
   4.1 Segregated Approach .......................................... 28
      4.1.1 Requirement on Number of Local Neighbors ............... 31
   4.2 Coupled Approach .............................................. 32
      4.2.1 Requirement on Number of Local Neighbors ............... 33
   4.3 Choice of Objective Function for an Optimal Global Error Bound .. 34

5 Numerical Results .................................................. 39
   5.1 Coefficient Generation .......................................... 40
      5.1.1 Sample 1D Domain ......................................... 40
      5.1.2 Sample 2D Domain ......................................... 50
   5.2 Advection Equation .............................................. 53
      5.2.1 Advection Equation with Prescribed Inflow ............... 53
      5.2.2 Stability at the Semi-Discrete Level ....................... 54
      5.2.3 Time Stepping and Discrete Stability ...................... 54
      5.2.4 1D Results .................................................. 55
      5.2.5 1D Results with Periodic Boundary Condition ............. 59
      5.2.6 2D Results .................................................. 61

6 A Generalized Meshless Framework ................................ 67
   6.1 Geometric Interpretation of the Meshless Coefficients .......... 68
   6.2 Advection Revisited ............................................ 69

7 Application to Euler Equations .................................. 75
   7.1 The Euler Equations ............................................ 76
   7.2 Stabilization and High Resolution Schemes in Meshless Framework .. 77
      7.2.1 Matrix Diffusion ........................................... 78
      7.2.2 Scalar Diffusion ........................................... 79
      7.2.3 Upwind Splitting Methods ................................ 80
      7.2.4 Improved Resolution ....................................... 81
7.3 Spatial Discretization in Meshless Framework ........................................ 82
    7.3.1 Boundary Conditions ................................................................. 83
7.4 Time Discretization ................................................................. 85
7.5 Convergence Acceleration .......................................................... 87
    7.5.1 Local Time Stepping ................................................................. 87
    7.5.2 Residual Averaging ................................................................. 87
    7.5.3 Enthalpy Damping ................................................................. 88
7.6 Meshless Coefficients Generation .................................................. 88
7.7 Compressible Flow Test Cases ...................................................... 89

8 Conclusion and Future Work ......................................................... 99

A Equivalence of Matrix Constraints ............................................... 101
    A.1 Expression of Matrix Constraints in Terms of Incidence Matrices ... 101
    A.2 From “Opposite” Form to Delta Form ........................................... 103
    A.3 From Delta Form to Point-centered Form ...................................... 104

B Existing Schemes Satisfying C-1 and C-2 ...................................... 107

Bibliography ......................................................................................... 109
# List of Tables

3.1 Dimension of polynomial bases ............................................ 21
4.1 Number of local neighbors required for the segregated algorithm. . . . 31
4.2 Number of local neighbors required for the coupled algorithm. . . . 34
5.1 1D meshless coefficient generation with segregated algorithm . . . . 40
5.2 1D meshless coefficient generation with coupled algorithm . . . . . 41
5.3 Numerical error for 1D advection, segregated coefficients . . . . . 58
5.4 Numerical error for 1D advection, coupled coefficients . . . . . . . 58
5.5 Numerical error for periodic 1D advection, segregated coefficients . . 59
5.6 Numerical error for periodic 1D advection, coupled coefficients . . . 59
7.1 Cylinder inviscid drag convergence ........................................ 91
7.2 Inviscid drag convergence for subsonic flow over airfoils ............... 91
7.3 Lift and drag coefficients, NACA 0012, $M = 0.8, \alpha = 1.25^\circ$ .... 92
7.4 Lift and drag coefficients, NACA 0012, $M = 0.85, \alpha = 1.0^\circ$ ....... 93
7.5 Lift and drag coefficients, RAE 2822, $M = 0.75, \alpha = 3.0^\circ$ ......... 93
7.6 Lift and drag coefficients, KORN airfoil, $M = 0.75, \alpha = 0.0^\circ$ .... 93
# List of Figures

3.1 Sample 1D connectivity ................................................. 23

5.1 1D leading error term of meshless discretization, $n_p = 50$ .......... 42
5.2 1D meshless coefficients $m_i$, $n_p = 50$ .......................... 43
5.3 1D meshless coefficients $a_{ij}$ viewed from a solution point, $n_p = 50$ .......................... 44
5.4 1D meshless coefficients $a_{ij}$, $n_p = 50$ .......................... 45
5.5 1D meshless coefficients $a_{ij}$, $n_p = 100$ .......................... 47
5.6 1D meshless coefficients $a_{ij}$, $n_p = 200$ .......................... 48
5.7 1D meshless coefficients $a_{ij}$, $n_p = 400$ .......................... 49
5.8 Connectivity for sample 2D domain .................................. 51
5.9 Virtual volumes and virtual face area magnitudes, coupled algorithm 52
5.10 Solutions to 1D advection equation .................................. 57
5.11 Point convergence for 1D advection equation ......................... 58
5.12 Solutions to 1D advection equation, periodic boundary conditions .................................. 60
5.13 Point convergence for periodic 1D advection equation ............... 61
5.14 Solutions to 2D advection equation, central-type scheme .......... 63
5.15 Solutions to 2D advection equation, central-type scheme (2) ...... 64
5.16 Point convergence for 2D advection equation, central-type scheme 65

6.1 Solutions to the 2D advection equation, upwind scheme ............ 71
6.2 Solutions to the 2D advection equation, upwind scheme (2) ........ 72
6.3 Point convergence of 2D advection equation, upwind scheme ....... 73

7.1 Stencil for edge-based reconstruction ................................. 82
7.2 Ghost points at domain boundary ........................................ 84
7.3 Virtual volumes and face magnitudes around RAE 2822 .......... 90
7.4 Subsonic drag convergence for flow over airfoils .................. 92
7.5 Flow over NACA 0012, $M = 0.80$, $\alpha = 1.25^\circ$ ............... 94
7.6 Flow over NACA 0012, $M = 0.85$, $\alpha = 1.0^\circ$ ............... 95
7.7 Flow over RAE 2822, $M = 0.75$, $\alpha = 3.0^\circ$ ............... 96
7.8 Flow over KORN airfoil, $M = 0.75$, $\alpha = 0.0^\circ$ ............... 97

B.1 Sample triangular mesh section for nodal finite volume ............. 107
Chapter 1

Introduction

1.1 The Challenge of Domain Discretization

In recent decades, advancements in computing technology and hardware have enabled scientists and engineers to use numerical simulations to tackle problems with complexity and scale larger than ever before. The use of computational simulations has become an integral part of the design process of engineering systems.

In a typical simulation, systems of ordinary or partial differential equations (PDEs) governing different physical phenomena in the engineering system of interest are discretized into systems of algebraic equations, which are then solved with numerical algorithms. The finite difference method (FDM), finite volume method (FVM) and finite element method (FEM) are popular classes of such numerical algorithms. Traditionally, an important part of this discretization and solution process is mesh generation. To obtain a mesh, the domain of interest is decomposed into constituent geometrical units in which the variables of the differential equations are calculated. Depending on the numerical techniques used, these constituents are well known as points, volumes or elements in meshes, and must be connected in predefined manners that provide all the necessary information for obtaining the set of algebraic equations suitable for the numerical scheme of choice.

It has long been known that mesh generation limits the overall accuracy, robustness and speed of numerical simulation processes. Two issues constantly surface. The
CHAPTER 1. INTRODUCTION

quality of numerical solutions often depend significantly on meshes. Generation of meshes can also require large amounts of time and labor for any engineering systems with reasonable geometric complexity. Over the years, many researchers have devoted time to improve the mesh generation process. Various (although sometimes ad hoc) mesh quality measures aimed at guaranteeing convergence and accuracy of numerical solutions have been defined based on geometry and experience obtained from trial and error. Mesh generation techniques, such as the Delauney triangulation\[1\], overset mesh\[2\], and adaptive mesh refinement\[3\], have been continuously improved. Newer algorithms and ideas including automated parallel mesh generation\[4\],\[5\] and strand mesh\[6\] have also been invented. Despite these excellent efforts, however, the very improvements and breakthroughs in numerical algorithms and computing hardware that have propelled large-scale scientific computing have also exposed the weaknesses in the mesh generation processes to date. To some degree, all meshing algorithms still have difficulties addressing one of the two issues mentioned above. Known techniques that lead to high-quality meshes require much and frequent human intervention, while methods that allow more automation still lack the desired robustness to handle domains defined by complex geometry. As a result, mesh generation frequently remains the most labor-intensive and time-consuming step in a computational simulation and creates a bottleneck that limits the overall speed of the numerical simulation process.

These problems and limitations have raised interests in and the need for alternative domain discretization procedures that bypass the use of a mesh — meshless algorithms.

1.2 Definition and Survey of Meshless Methods

To attempt to circumvent the tedious task of mesh generation, many have introduced meshless methods for numerical solution of PDEs. Meshless methods have different names such as gridless, gridfree, meshfree, and so on. Regardless of their names, they all work around the need of a mesh in some sense. In meshless methods, the governing PDEs are discretized into the resulting set of algebraic equations without the need to geometrically form a mesh, but by using a global cloud of points in the entire
1.2. DEFINITION AND SURVEY OF MESHLESS METHODS

domain, and a local point cloud formed by a subset of global points around each of
the global points. Both the global and local point clouds are formed with connectivity
that is less strict than those required in a mesh. Meshless methods can be classified
according to different criteria. Here, an overview of the most well-known meshless
methods classified by their underlying formulation procedure from the differential
equations will be presented.

1.2.1 Meshless Methods Based on Strong Forms

This class of methods is based on the discretization of the strong form of the governing
equations using various approximation techniques. A variety of methods fall into this
category. They mostly differ in the way the unknown functions are treated.

The Smooth Particle Hydrodynamics (SPH) framework, first introduced by Mon-
aghan[7], uses an integral approximation of a function. In SPH, solution points can
move around in a Lagrangian approach. Designed for astrophysical applications, SPH
is suitable for problems with an unknown domain size. The Finite Point Method
(FPM), first developed by Oñate et al.[8], and later used by Löhner et al.[9], uses
polynomial basis functions to approximate the unknown function. FPM was used
on both incompressible and compressible flow problems with success. Batina[10] had
also previously arrived at a similar formulation using polynomial bases with least
squares and used it to simulate inviscid and viscous compressible flow. In the case
when the polynomial approximation function goes through the solution points, the
formulation reduces to a Taylor series formulation, which was also used by many to
develop meshless schemes. The Least Squares Kinetic Upwind Method (LSKUM) by
Deshpande et al.[11] was one of such schemes. LSKUM was later used or modified
by Ghosh and Despande[12], Ramesh and Despande[13], Anandhanarayanan and Na-
garathinam[14], Harish and Pavanakumar[15], Srinarayana et al.[16] and Kumar et
al.[17] to simulate compressible flow problems and more complex problems including
store separation. Similar Taylor-based methods were developed and studied by
Praveen[18,19], Praveen and Balakrishnan[20], Jaisnakar et al.[21], Morinshi[22], and
Balasubramanyam and Rao[23,24]. More recently, Katz and Jameson[25,26] developed another Taylor-based method and used it for solving compressible flow problems and handling communications between different grids.

Another interpolation technique is the use of radial basis functions (RBFs)[27] whose values are based only on the radius from a center point in space. Kansa[28,29] first applied RBFs to fluid dynamics applications. Other work by Hon and Schaback[30], Power and Barraco[31], Li et al.[32], Sharan et al.[33], Sarler and Vertnik[34], Wendland[35], Divo and Kassab[36,37], Chinchapatnam et al.[38], Shu et al.[39], and Tota and Wang[40] have also involved RBFs in incompressible flow, compressible flow, and heat transfer problems.

1.2.2 Meshless Methods Based on Weak Forms

As one might expect, this class of meshless methods is based on the weak form (e.g. the variational form) of the governing equations. The first of such methods was the Diffuse Element (DE) method introduced by Nayroles et al.[41]. Belytschko et al. soon followed and extended DEM to the Element Free Galerkin (EFG) method. Because of the underlying weak form in their formulations, DE and EFG both require some background grid for computing the required integrals in the weak form. Later Alturi and Zhu[42] introduced the Meshless Local Petrov Galerkin (MLPG) method based on a local variational formulation. MLPG reduces the need for the background mesh to a local one, improving on that of DEM and EFG. MLPG was used on incompressible flow problems by Lin an Atluri[43]. Separate from the work above, Liu et al.[44] also introduced the Reproducing Kernal Particle Method (RKPM) based on SPH. Meshless method based on the more general partition-of-unity framework have also been developed by Duarte and Oden[45] and Melenk and Babuska[46].

1.2.3 Hybrid Meshless Methods

Another context in which meshless methods have demonstrated success is their use in a hybrid setting, e.g. handling parts of the domain in a simulation. For example, Belytschko et al.[47] coupled a finite element method with EFG. Kirshman and Liu[48],
Koh et al.[49] and Luo et al.[50] coupled meshless schemes near geometry surfaces with Cartesian grid methods in the rest of the domain. Kamatsuchi[51] performed large-scale viscous calculations using a meshless method in a near-wall subgrid that supplemented sub-divided Cartesian meshes through immersed boundary methods. Katz et al.[52] also used a meshless method as an interface between different meshes.

1.3 Challenges Faced by Meshless Methods

The work mentioned above demonstrates the great potential in applying meshless methods to tackle complex problems in scientific computing. However, the local nature of the discretization in meshless schemes also means that meshless methods face their own different set of challenges. It would be unfair to compare mesh generation with meshless algorithms without also highlighting these challenges.

The most fundamental challenge is the lack of formal conservation. Conservation is simply that, given a set of conservation laws (differential equations) governing certain physical variables (mass, momentum, energy, etc.), the integrated rate of change in any selected volume in a domain must be equal to the net influx at the boundary of the volume. Finite difference, finite volume and finite element methods can easily be made to satisfy conservation discretely. This is mainly due to the symmetry and consistency built into common fluxes at interfaces by using geometric quantities such as face areas and volumes (or their equivalents) in these mesh-based discretization procedures.

However, the local nature of meshless discretization procedures means that the symmetry and geometric consistency resulting from geometric information in mesh-based schemes are not available. Thus, as far as the work found in the literature surveyed for this thesis has shown, mesh-free schemes do not preserve conservation at the discrete level except in very limited situations (such as with a uniform point distribution, which defeats the purpose of meshless schemes). This lack of conservation has often been a source of doubt of the accuracy and robustness of meshless methods. Indeed, non-conservation can lead to unpredictable errors, especially when sharp discontinuities exist in the solution. The difficulty in quantifying the effects of
non-conservation on accuracy and stability of algorithms really does not come into the defense of meshless methods in any way. As a result, this lack of conservation sometimes also takes the blame for the lack of robustness observed in certain meshless methods, especially in those based on the strong form, which assume some smoothness in the solutions of the governing equations. This is a rather unfortunate fact because strong-form-based methods provide the most flexibility in discretization.

In addition, almost all meshless algorithms have higher computational costs when compared to their mesh-based counterparts again because of the lack of symmetry in the discretization procedure. In meshless methods based on strong forms, this means that fluxes are computed on a per-solution-point basis, instead of the common approach in meshed schemes to compute fluxes at some interface per each pair of connected solution points and symmetrically distribute them to each solution point in the pair, on either side of the interface. As a result, many strong-form-based meshless methods cost 2-3 times as much as finite difference or finite volume methods. This asymmetry also posts steeper memory requirements for meshless schemes. Katz and Jameson[26] recently formulated their Taylor-series-based meshless algorithm into a form analogous to finite volume that not only reduced the cost of their method to that of a finite volume method but also required less memory than finite volume methods. However, their scheme is still formally non-conservative.

In weak form-based meshless methods, the lack of symmetry manifests itself in the computation of the inner product of basis functions at different solution point, as the local discretization at each point may result in basis functions of different shapes and magnitudes. The extra overhead in these methods are even more prohibitive. Costs in excess of 5-10 times of that of finite element methods have been reported.

Other than the inherent challenges that arise from the lack of symmetry in the local discretization, meshless methods also suffer from other problems that hinder their development. One of such problems that is very real is that, even though meshless methods do not require rigid connectivity in the form of a grid, they still require initial point distributions and ways to generate point clouds from these distributions. More often than not, meshless methods are susceptible to point distribution as much as mesh-based methods. In methods based on weak forms, the size and shape of the
support domain for obtaining local integrals further complicate the issue. Researchers have applied a wide variety of techniques to generate the point distributions and limited connectivity required for meshless methods. To form point clouds, Oñate et al. [8] used a quadrant search technique, while Löhner et al. [9] used an octree search followed by local Delauney triangulation. Both methods further prune clouds to avoid ill-conditioned local least squares matrices. Chiu and Jameson [53] used a quality function based on both orthogonality and proximity to obtain local point clouds of good quality. However, most studies have not addressed the issue of the sensitivity of the accuracy of the algorithms with respect to change in global point distributions. The search for a good algorithm for obtaining optimal point distributions is very much a wide-open research subject. This highlights that, even though meshless connectivity has much less stringent reciprocity requirements, meshless methods face other challenges in preprocessing, particularly in ensuring the quality of point clouds.

On top of all the challenges at the algorithm and solution level, visualization of meshless solutions can also be an issue as many features in most popular post-processing software are designed to work with connectivity associated with meshes.

1.4 A Preview of the Current Thesis

The goal of this thesis is to develop a new meshless method that addresses some of the challenges above, particularly the challenge of achieving discrete conservation that has eluded researchers for over two decades. To that effect, the new meshless scheme should, i) observe formal conservation at the discrete level, ii) possess enough generality for solving general conservation laws, iii) have efficiency as good as those of some existing mesh-based schemes, and iv) harness existing numerical algorithms, both from the meshless and meshed worlds, and even other general algorithms in linear algebra, as much as possible. The first two criteria are simply the design criteria for a conservative meshless scheme. The last two criteria would allow the new method to be attractive from a cost standpoint. In particular, the final criterion would facilitate the implementation of the new method, or even its integration into existing mesh-based solvers, making the new method quickly and easily accessible to the scientific
community.

The first and most important criterion of discrete conservation will be addressed in chapter 2, in which the formulation of an algebraic meshless derivative operator based on local function values and a set of chosen coefficients is presented. It is proven that the new method is both locally and globally conservative at the discrete level. Because of the novel nature of this conservative formulation, much effort should be spent on understanding the requirements and techniques to obtain the unknown coefficients. Chapter 3 contributes to part of that by including some manipulation and various equivalent forms of the constraints in matrix form, preparing the reader for discussions on generation of the meshless coefficients.

Chapter 4 contains discussions of some necessary conditions based on the algebraic and matrix form of the constraints. There, two different algorithms designed for generating the meshless coefficients are also presented. The generation algorithms are first tested on sample domains of interest. To gain further understanding, the resulting meshless coefficients are then used to solve the classic advection equation. These results are shown in chapter 5. Many algorithms are first characterized and tested on the advection equation. Therefore, the tests in chapter 5 serves as a great benchmark for the current meshless scheme. Also, the availability of exact solutions to the advection equation allows one to gain a significant amount of insight into the coefficient generation process. The criteria iii) and iv) are addressed in chapter 6, where an important and desirable extension of the meshless algorithm is presented. This extension allows one to incorporate numerical schemes derived for finite volume discretization with minimal, if not completely free of, modifications. The feasibility and validity of this framework is confirmed by revisiting the test cases involving the advection equation. This extension is the cornerstone for the current framework to be applied to much more general problems. In chapter 7, one can see this generalized framework at work in several classic benchmark problems in computational fluid dynamics involving inviscid subsonic and transonic flow over airfoils. There, the superb agreement of the meshless results with finite volume results demonstrates the practicality and potential of the framework.

Finally, conclusions, implications and future work are discussed in chapter 8.
Chapter 2

A Meshless Differentiation Operator

2.1 Formulation

Given a domain of interest $\Omega$, one can discretize the domain using a global cloud of $n_p$ solution points $\{\vec{x}_i\}$, $i = 1, \ldots, n_p$. Consider a point cloud containing points both on the boundary ($i \in S_B$) and in the interior of the domain ($i \notin S_B$), where $S_B$ is the set of boundary points in the discretization. At each solution point $\vec{x}_i \in \Omega$, define the local point cloud $s_i$ to be the set of $n_{nbr_i}$ neighboring points, excluding point $i$ itself.

Given an unknown function $u$, the discrete first derivative in the $k$-th spatial direction of an unknown scalar $u$, denoted by $\delta^k u(\vec{x}_i)$, is defined by

$$m_i \delta^k u(\vec{x}_i) = a_{i}^{k} u(\vec{x}_i) + \sum_{j \in s_i} a_{ij}^{k} u(\vec{x}_j),$$

where $\vec{x}_i$ is the coordinates of the $i$-th point, and $a_{i}^{k}$ and $m_i$ represent the meshless coefficients to be determined. For consistent dimensions of the equation, one can consider $m_i$ to represent a virtual volume associated with each point, while the coefficients $a_{ij}^{k}$ for the point pair $(i, j)$ have corresponding dimensions of area (this characterization is justified in section 2.3).

The meshless coefficients $a_{ij}^{k}$ and $m_i$ satisfy the following two conditions:
CHAPTER 2. A MESHLESS DIFFERENTIATION OPERATOR

C-1. Reciprocity and boundary conditions:

\[ a_{ij}^k = -a_{ji}^k, \quad i \neq j \quad (j \in s_i \Leftrightarrow i \in s_j) \]
\[ a_{ii}^k = 0, \quad i \notin S_B \]
\[ a_{ii}^k = \frac{1}{2} n_i^k, \quad i \in S_B, \]

where \( n_i^k \) is the \( k \)-th component of the outward-facing, area-weighted boundary normal of a boundary point.

C-2. Consistency of order \( L \):

\[ a_{ii}^k \phi(\vec{x}_i) + \sum_{j \in s_i} a_{ij}^k \phi(\vec{x}_j) = m_i \partial^k \phi(\vec{x}_i) \]

for all multivariate polynomials \( \phi \) of total order up to \( L \), where \( \partial^k \) is the continuous first derivative operator in the \( k \)-th spatial coordinate.

The remainder of this chapter will be devoted to presenting the properties of the meshless operator defined above. Unless otherwise specified, the presented analysis holds for three dimensions in space and automatically applies to lower dimensional cases.

2.2 Global Conservation and Mimetic Properties

The scientific community has always cast doubt on meshless methods because of their lack of formal conservation. The feasibility of a conservative meshless algorithm is also one of the biggest unanswered questions in the field. In this section, it will be demonstrated that if a discrete operator satisfies the conditions in section 2.1, then it also has desirable discrete properties corresponding to some classic properties of the continuous first derivative operator \( \partial^k \), namely:

- Conservation

\[ \int_{\Omega} \partial^k u dx = \int_{\partial \Omega} u n^k ds \]
• Integration by parts
\[
\int_{\Omega} v \partial^k u \, dx = \int_{\partial \Omega} v n^k \, ds - \int_{\Omega} u \partial^k v \, dx
\]
(where \( u \) and \( v \) are two scalars defined in the domain of interest)
• Energy conservation
\[
\int_{\Omega} u \partial^k u \, dx = \int_{\partial \Omega} \frac{1}{2} u^2 n^k \, ds
\]

**Theorem 2.2.1** (Discrete conservation). If \( m_i \) and \( a^k_{ij} \) satisfy C-1 and C-2, then

\[
\sum_{i=1}^{np} m_i \delta^k u_i = \sum_{i \in S_B} u_i n^k_i ,
\]  
where \( S_B \) is the collection of all boundary points.

**Proof.** We can write the discrete first derivative as

\[
m_i \partial^k u_i \approx m_i \delta^k u_i = \sum_{j=1}^{np} a^k_{ij} u_j ,
\]
where \( \tilde{a}^k_{ii} = a^k_{ii}, \tilde{a}^k_{ij} = a^k_{ij} \) if \( j \in s_i \), and \( \tilde{a}^k_{ij} = 0 \) otherwise. Let \( \phi \equiv 1 \) in C-2, and use \( a^k_{ij} + a^k_{ji} = 0 \) from C-1, we get

\[
\tilde{a}^k_{ij} = -\tilde{a}^k_{ji} \quad (\text{only if } i \neq j)
\]

\[
\sum_{j=1}^{np} \tilde{a}^k_{ji} = a^k_{ii} ,
\]
and also

\[
\sum_{j=1}^{np} \tilde{a}^k_{ji} = 2a^k_{ii} .
\]
Incorporating equation (2.3) into the left hand side of (2.2) and using (2.4), we have

\[ \sum_{i=1}^{n_p} m_i \delta^k u_i = \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} \tilde{a}_{ij}^k u_j = \sum_{i=1}^{n_p} \left( \sum_{j=1}^{n_p} \tilde{a}_{ij}^k \right) u_j = \sum_{i=1}^{n_p} 2a_{ij}^k u_j \\
= \sum_{i \in S_B} n_i^k u_i , \quad (2.7) \]

where we changed the dummy index from \( j \) to \( i \) in the last step.

**Theorem 2.2.2** (Summation by parts). If \( m_i \) and \( a_{ij}^k \) satisfy C-1 and C-2, then

\[ \sum_{i=1}^{n_p} m_i v_i \delta^k u_i + \sum_{i=1}^{n_p} m_i u_i \delta^k v_i = \sum_{i \in S_B} v_i u_i n_i^k . \quad (2.8) \]

**Proof.** Substituting equation (2.3) into the left hand side of (2.8), we have

\[ \sum_{i=1}^{n_p} m_i v_i \delta^k u_i + \sum_{i=1}^{n_p} m_i u_i \delta^k v_i \\
= \sum_{i=1}^{n_p} v_i \sum_{j=1}^{n_p} \tilde{a}_{ij}^k u_j + \sum_{i=1}^{n_p} u_i \sum_{j=1}^{n_p} \tilde{a}_{ij}^k v_j \\
= \sum_{i=1}^{n_p} v_i \sum_{j=1}^{n_p} \tilde{a}_{ij}^k u_j + \sum_{i=1}^{n_p} u_i \left( \sum_{j=1}^{n_p} \tilde{a}_{ij}^k v_j - \sum_{j \neq i} \tilde{a}_{ij}^k v_j \right) + 2 \sum_{i=1}^{n_p} a_{ii}^k v_i u_i \\
= \sum_{i=1}^{n_p} v_i u_i n_i^k , \quad (2.9) \]

where we exchanged the dummy indices \( i \) and \( j \) in the second term on the second-to-last line.

**Corollary 2.2.3** (Discrete energy conservation). If \( m_i \) and \( a_{ij}^k \) satisfy C-1 and C-2, then

\[ \sum_{i=1}^{n_p} m_i u_i \delta^k u_i = \sum_{i \in S_B} \frac{u_i^2}{2} n_i^k . \quad (2.10) \]
Proof. Equation (2.10) is obtained by setting \( v = u \) in (2.8).

Thus, right from the start, we have obtained the first breakthrough with a globally conservative meshless operator. This is very important as the new scheme is primarily designed for numerically solving conservation laws.

## 2.3 Local Conservation

Local discrete conservation is important in computing solutions with discontinuities. It is known that the location and speed of such discontinuities may be incorrect when a non-conservative scheme is used. With the establishment of discrete global conservation, one can then naturally ask whether this discrete global conservation actually resulted from telescoping terms from some discrete local conservation, i.e. whether conservation is satisfied discretely for an arbitrary, connected part of the discretized domain. The proof in this section demonstrates that the current meshless operator does indeed preserve local conservation at the discrete level, i.e. it satisfies the discrete analog of

\[
\int_{\omega_i} \partial^k u dx = \int_{\partial \omega_i} u n^k ds
\]

at each solution point \( i \), where \( \omega_i \) is some control volume defined around the point \( i \).

**Theorem 2.3.1** (Local discrete conservation). If \( m_i \) and \( a_{ij}^k \) satisfy C-1 and C-2, then, defining \( n_{ij}^k = 2a_{ij}^k \) to be some virtual face area vectors for interior point pairs, the following conditions hold:

1. At each point \( i \),

\[
m_i \delta^k u_i = \begin{cases} 
\sum_{j \in s_i} u_{f_j} n_{f_j}^k + u_i n_i^k & i \in S_B \\
\sum_{j \in s_i} u_{f_j} n_{f_j}^k & i \notin S_B
\end{cases}
\]  \hspace{1cm} (2.11)

where \( u_{f_j} \) represents an interface value of \( u \).
2. In addition, each $m_i$ represents a virtual volume associated with point $i$ that is enclosed by some virtual faces, with face areas denoted by $\vec{n}_{f_j}$, between point $i$ and point $j$, with corresponding interface function values $u_{f_j}$ (plus boundary faces if $i \in S_B$).

To facilitate the proof, we introduce the following corollary.

**Corollary 2.3.2** (Local discrete geometric conservation law). If $m_i$ and $a_{ij}^k$ satisfy C-1 and C-2, and the vector valued multivariate function $\vec{\phi}$ satisfies the divergence free condition

$$\nabla \cdot \vec{\phi} = \sum_{k=1}^{m} \partial^k \phi^k = 0,$$

then the following condition holds:

$$\vec{\phi}_i \cdot \vec{n}_i + \sum_{j \in s_i} \vec{\phi}_{f_j} \cdot \vec{n}_{f_j} = 0 \quad i \in S_B$$

$$\sum_{j \in s_i} \vec{\phi}_{f_j} \cdot \vec{n}_{f_j} = 0 \quad i \notin S_B. \quad (2.12)$$

The proof of theorem 2.3.1 and corollary 2.3.2 will be presented together. As one will see, equation (2.11) directly leads to (2.12), which in turn leads to condition 2 above that is required to complete the proof of Theorem 2.3.1.

**Proof.** Applying C-2 to $\phi \equiv 1$ leads to $a_{ii}^k + \sum_{j \in s_i} a_{ij}^k = 0$. Multiplying this by $u_i$ and adding the result to the definition of the first derivative operator (2.1), we have

$$m_i \delta^k u_i = a_{ii}^k u_i + \sum_{j \in s_i} a_{ij}^k u_j + a_{ii}^k u_i + \sum_{j \in s_i} a_{ij}^k u_i$$

$$= 2a_{ii}^k u_i + \sum_{j \in s_i} a_{ij}^k (u_i + u_j)$$

$$= 2a_{ii}^k u_i + \sum_{j \in s_i} n_{ij}^k \frac{(u_i + u_j)}{2}. \quad (2.13)$$

For interior points, $a_{ii}^k = 0$. For boundary points, $a_{ii}^k = \frac{1}{2} n_{ii}^k$. If we let $u_{f_j} = \frac{1}{2} (u_i + u_j)$, then we obtain equation (2.11).
To prove Corollary 2.3.2, let $u = \phi$ be a polynomial of total order $L$. Consistency of order $L$ gives

$$m_i \partial^k u_i = \begin{cases} 
\sum_{j \in s_i} u_{f_j} n_{f_j}^k + u_i n_i^k & i \in S_B \\
\sum_{j \in s_i} u_{f_j} n_{f_j}^k & i \notin S_B
\end{cases}$$

If $\bar{\phi}$ is divergence free, summing over equation (2.14) applied to each component of $\bar{\phi}$ results in equation (2.12).

To complete the proof of Theorem 2.3.1, it remains to show that coefficients $m_i$ and $a_{ij}$ are consistent and do not lead to numerical sources. This can be seen through the following two properties:

1. Let $u = x^k$ (recall $k$ denotes spatial dimension). Equation (2.11) becomes

$$\sum_{j \in s_i} n_{f_j} \frac{x_{f_j} + x_i}{2} = m_i$$

for an interior point $i$, and

$$\sum_{j \in s_i} n_{f_j} \frac{x_{f_j} + x_i}{2} + n_i x_i = m_i$$

for a boundary point.

2. Corollary 2.3.2 applied to $\phi^k \equiv 1$ and $\bar{\phi}^k \equiv 0$ for $\bar{k} \neq k$ ($k$ and $\bar{k}$ denotes spatial indices) yields

$$\sum_{j \in s_i} n_{f_j}^k = 0$$

for an interior point $i$ and

$$\sum_{j \in s_i} n_{f_j}^k + n_i^k = 0$$

for a boundary point.

These conditions guarantee that a virtual volume of size $m_i$ is fully enclosed by its interfaces, which include boundary faces if appropriate. Thus, no numerical sources arise from inconsistent definition of virtual faces and volumes. The scheme is locally conservative.

Theorem 2.3.1 justifies the geometric interpretation of the coefficients $a_{ij}^k$ and $m_i$ as analogs of face areas and volumes. In section 6.1, one will see that, with this geometric interpretation, the meshless operator can be generalized to a conservative framework that can handle a wide variety of problems. In particular, the highly desirable local conservation property also suggests that the meshless operator and its variants could be used with appropriate numerical algorithms to deal with discontinuities in numerical solutions. A collection of transonic flow test problem results that confirm this claim can be found in section 7.
2.4 Existence of Meshless Coefficients — Necessary Conditions

The desirable conservation properties of the current meshless operator carry promise. However, a question that has not been addressed so far is how one can obtain the set of coefficients \( a_{ij}^k \) and \( m_i \) that satisfy conditions C-1 and C-2. This section highlights the potential challenge in generating the values of the coefficients by bringing into light some extra global properties of the meshless derivative operator and coefficients. These properties are the global analog of the local geometric conservation law (corollary 2.3.2) and the related equation (2.14). They are also the discrete version of the famous divergence theorem. As one will see, these properties provide important insight into what is necessary to obtain the coefficients that satisfy C-1 and C-2.

**Theorem 2.4.1 (Discrete divergence theorem).** If \( m_i \) and \( a_{ij}^k \) satisfy C-1 and C-2, then the following condition holds for all multivariate polynomials \( \phi \) of total order \( 2L \):

\[
\sum_{i \in S_B} \phi(\vec{x}_i) n_i^k = \sum_{i=1}^{n_p} m_i \partial^k \phi(\vec{x}_i). 
\]  

(2.15)

**Proof.** It is sufficient to prove (2.15) for all multivariate monomials \( \phi \) of order less than or equal to \( 2L \). Let \( \phi = \phi_1 \phi_2 \), where both \( \phi_1 \) and \( \phi_2 \) are monomials or order less than or equal to \( L \) and thus satisfy condition C-2:

\[
a_{ii}^k \phi_1(\vec{x}_i) + \sum_{j \in \mathcal{S}_i} a_{ij}^k \phi_1(\vec{x}_j) = m_i \partial^k \phi_1(\vec{x}_i) \quad (2.16)
\]

\[
a_{ii}^k \phi_2(\vec{x}_i) + \sum_{j \in \mathcal{S}_i} a_{ij}^k \phi_2(\vec{x}_j) = m_i \partial^k \phi_2(\vec{x}_i). \quad (2.17)
\]

Multiplying equation (2.16) by \( \phi_2(\vec{x}_i) \), and (2.17) by \( \phi_1(\vec{x}_i) \) and adding the results, we have

\[
2a_{ii}^k \phi_1(\vec{x}_i) \phi_2(\vec{x}_i) + \sum_{j \in \mathcal{S}_i} a_{ij}^k (\phi_1(\vec{x}_i) \phi_2(\vec{x}_j) + \phi_1(\vec{x}_j) \phi_2(\vec{x}_i)) = m_i \partial^k (\phi_1(\vec{x}_i) \phi_2(\vec{x}_i)).
\]
Since $a_{ij}^k + a_{ji}^k = 0$ in condition C-1, we know that
\[
\sum_{(i,j) \in E} a_{ij}^k (\phi_1(\vec{x}_i)\phi_2(\vec{x}_j) + \phi_1(\vec{x}_j)\phi_2(\vec{x}_i)) = 0,
\]
where $E$ is the set of connected edges in the domain. Summing equation (2.4) over $i = 1, \ldots, n_p$, we have
\[
\sum_{i=1}^{n_p} 2a_{ii}^k \phi_1(\vec{x}_i)\phi_2(\vec{x}_i) = \sum_{i=1}^{n_p} m_i \partial^k (\phi_1(\vec{x}_i)\phi_2(\vec{x}_i)).
\]
Finally, using the definition of $a_{ii}^k$ from condition C-1, we get equation (2.15) for $\phi = \phi_1\phi_2$.

At first glance, the discrete divergence theorem seems like a benign property naturally resulting from the reciprocity of the coefficients $a_{ij}^k$, which were crucial for establishing local conservation. However, the discrete divergence theorem actually signifies something further: In the process of obtaining equation (2.15), all the terms involving $a_{ij}^k$ in the nodal constraints were cancelled after the summation of a linear combination of the constraints. This means that the linear constraints on $a_{ij}^k$ represented by conditions C-1 and C-2 are linearly dependent. In other words, for a set of $a_{ij}^k$'s to satisfy conditions C-1 and C-2, $m_i$ must satisfy equation (2.15).

In a similar fashion, the following corollary shows that equation (2.15) as linear constraints for $m_i$ are also dependent.

**Corollary 2.4.2** (Discrete geometric conservation law). If $m_i$ and $a_{ij}^k$ satisfy C-1 and C-2, and the vector valued multivariate polynomials $\vec{\phi}$ of order $2L$ satisfies the divergence free condition
\[
\nabla \cdot \vec{\phi} = \sum_{k=1}^{HL} \partial^k \phi^k = 0,
\]
then the following condition holds:
\[
\sum_{i \in S_B} \vec{\phi}(\vec{x}_i) \cdot \vec{n}_i = 0.
\]
Proof. Equation (2.18) is obtained by summing equation (2.15) over $k = I, II, III$ and using the divergence free condition.

Thus, to have any hope of obtaining a consistent set of $a^k_{ij}$’s and $m_i$’s, or just a set of consistent $m_i$’s that allow subsequent solution of $a^k_{ij}$’s, the boundary face areas $n_i^k$ must be chosen appropriately according to equation (2.18).

Fortunately, equation (2.18) represents a set of linearly independent constraints and can be solved in various ways. One can refer to chapter 4 for the strategies employed to solve equation (2.18) in a step of two different algorithms that compute meshless coefficients satisfying all the constraints mentioned so far.
Chapter 3

Constraints in Matrix Form

In chapter 2, the meshless operator and the constraints on its coefficients were presented in algebraic form. It was mentioned that conditions C-1 and C-2 can be expressed as a linear system of constraints and written as a matrix-vector system. In this chapter, one can find the expression of these constraints as functions of simple matrices with the help of some simple concepts from graph theory. The resulting matrix form plays an important role in the understanding of the properties of the system.

3.1 Elementary Definitions from Graph Theory

Graph theory has long been applied in various engineering applications such as circuits and network-related problems. This section reviews selected concepts in graph theory that are relevant to subsequent analysis of the meshless operator.

Definition 3.1.1 (Graph). A graph $G = (V, E)$ consists of a set of $n_v$ vertices $V = v_1, v_2, \ldots, v_{n_v}$ and a set of $n_e$ edges $E = e_1, e_2, \ldots, e_{n_e}$. Each edge connects a pair of vertices. The order of the vertices in the pair is irrelevant.

Definition 3.1.2 (Simple graph). A graph that does not have self-loops (an edge that connects a vertex to itself) or parallel edges (multiple edges connecting the same vertex pair) is a simple graph.
Definition 3.1.3 (Directed graph). A directed or an oriented graph \( G = (V, E) \) also consists of a set of \( n_v \) vertices \( V = v_1, v_2, \ldots, v_{n_v} \) and a set of edges \( E = e_1, e_2, \ldots, e_{n_e} \), except now each edge connecting vertices \( (v_i, v_j) \) is directed from \( v_i \) to \( v_j \).

Definition 3.1.4 (Incidence). In an undirected graph, an edge \( e \) is incident with a vertex \( v \) if \( v \) is an end point of \( e \). In a directed graph, an edge \( e \) directed from \( v_i \) to \( v_j \) is incident out of a vertex \( v_i \) and incident into \( v_j \).

Definition 3.1.5 (Degree). In any graph, the degree of a vertex is the number of edges incident with a vertex. In a directed graph, this is further divided into the in-degree and the out-degree which corresponds to the number of edges incident into and out of the vertex, respectively.

Mathematically, one can see that the solution points and connectivity is a graph object with solution points as graph vertices and connectivity edges as graph edges.

The following representation of graphs by matrices is also helpful for the analysis in the coming sections.

Definition 3.1.6 (Incidence matrix). The (unoriented) incidence matrix \( U \) of a graph is the \( n_v \times n_e \) matrix in which \( U_{ig} = 1 \) if the edge \( g \) is incident with \( i \) and \( U_{ig} = 0 \) otherwise.

Definition 3.1.7 (Directed incidence matrix). The directed incidence matrix \( B \) of a graph is the \( n_v \times n_e \) matrix in which \( B_{ig} = 1 \) if the edge \( g \) is incident out of \( i \), \( B_{ig} = -1 \) if the edge \( g \) is incident into \( i \), and \( B_{ig} = 0 \) otherwise.

With the above definitions, the matrix form of the constraints will be presented next.

3.2 Matrix Expressions

Let \( \mathbf{a}_p = \begin{bmatrix} \mathbf{a}_p^T, \mathbf{a}_p^H, \mathbf{a}_p^{III}^T \end{bmatrix}^T \) denote the global \( (n_p n_d) \times 1 \) vector holding values of \( a_{ii}^k \).

Similarly, let \( \mathbf{a}_f = \begin{bmatrix} \mathbf{a}_f^T, \mathbf{a}_f^H, \mathbf{a}_f^{III}^T \end{bmatrix}^T \) be the \( (n_f n_d) \times 1 \) vector of edge coefficients \( a_{ij}^k \), where only one of \( a_{ij}^k \) or \( a_{ji}^k \) is stored (the subscripts “p” and “f” denote “point” and
“face” respectively). Let \( m \) be the global \( n_p \times 1 \) vector holding values of \( m_i \). Then, the constraints on the meshless coefficients can be expressed as

\[
C \begin{bmatrix}
a_p \\
a_f \\
m
\end{bmatrix} = 0.
\]

(3.1)

\( C \) is a matrix of size \( n_p n_b n_f n_d \times (n_d(n_b + n_f) + n_p) \), where

\[
n_p = \text{number of points} \\
n_f = \text{number of virtual faces (meshless connectivity edges)} \\
n_b = \text{number of boundary points} \\
n_d = \text{number of spatial dimensions} \\
n_\phi = \text{total number of polynomial constraints per point}
\]

\[
= \sum_{r=0}^{L} \binom{n_d}{r} = \sum_{r=0}^{L} \frac{(r + 1)^{(n_d - 1)}}{(n_d - 1)!}
\]

and the overbar “\( \overline{\cdot} \)” notation denotes the rising sequential product

\[
x^{\overline{n}} = x(x + 1) \cdots (x + n).
\]

Table 3.1 lists \( n_\phi \), the number of linearly independent polynomials in the basis, for various total spatial dimensions and polynomial consistencies.

<table>
<thead>
<tr>
<th>( n_d )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>6</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L )</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>10</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Dimension of polynomial bases
CHAPTER 3. CONSTRAINTS IN MATRIX FORM

The constraint matrix $C$ in equation (3.1) can be partitioned as

$$ C = \begin{bmatrix} C_p & C_f & D \end{bmatrix}. $$

(3.2)

In three dimensions,

$$ C_p = \begin{bmatrix} \Phi_p & 0 & 0 \\ 0 & \Phi_p & 0 \\ 0 & 0 & \Phi_p \end{bmatrix}, \quad C_f = \begin{bmatrix} \Phi_{opp} & 0 & 0 \\ 0 & \Phi_{opp} & 0 \\ 0 & 0 & \Phi_{opp} \end{bmatrix}, \quad D = \begin{bmatrix} D' \\ D'' \\ D''' \end{bmatrix}, \quad (3.3)$$

and

$$ \Phi_p = \begin{bmatrix} (\Phi_1)_p \\ (\Phi_2)_p \\ \vdots \\ (\Phi_n)_p \end{bmatrix}, \quad \Phi_{opp} = \begin{bmatrix} (\Phi_{1,opp}) \\ (\Phi_{2,opp}) \\ \vdots \\ (\Phi_{n,opp}) \end{bmatrix}, \quad D^k = \begin{bmatrix} (\partial^k \Phi_1)_p \\ (\partial^k \Phi_2)_p \\ \vdots \\ (\partial^k \Phi_n)_p \end{bmatrix}. \quad (3.4)$$

In (3.4), $(\Phi_q)_p$ is the $n_p \times n_p$ diagonal matrix of nodal values of the $q$-th polynomial in the basis. $(\partial^k \Phi_q)_p$ is the $n_p \times n_p$ diagonal matrix of nodal values of the first derivative (in the $k$-th spatial dimension) of the $q$-th basis polynomial. $(\Phi_q)_{opp}$ represents the $n_p \times n_f$ sparse rectangular matrix in which the $(i, \hat{i}j)$-th entry contains the value of the $q$-th basis polynomial at $j$ if edge $\hat{i}j$ is directed from point $i$ to $j$, or the negative of the value of the $q$-th basis polynomial at $j$ if edge $\hat{i}j$ is directed from point $j$ to $i$, or zero if edge $\hat{i}j$ does not connect to point $i$ (“opp” stands for “opposite” side of the edge).

Given the tedious notation, the above formulation is best illustrated with examples. The following two examples contain some actual matrices corresponding to those in equations (3.3) and (3.4).

**Example 3.2.1** (1D, $L=2$). Here, the polynomial basis is $\{1, x, x^2\}$. Thus

$$ C_p = \Phi_p = \begin{bmatrix} I_p \\ X_p \\ X^2_p \end{bmatrix}, \quad C_f = \Phi_{opp} = \begin{bmatrix} B \\ X_{opp} \\ X^2_{opp} \end{bmatrix}, \quad D = \begin{bmatrix} 0_p \\ -I_p \\ -2X^2_p \end{bmatrix}. \quad (3.5)$$
where $0_p$ is the $n_p \times n_p$ zero matrix, $I_p$ is the $n_p \times n_p$ identity matrix, $B$ is the oriented incidence matrix of the connectivity graph, and $X_p = \text{diag}\{x_i\}$ is the $n_p \times n_p$ diagonal matrix containing the nodal $x$-coordinate values of the solution points, and so on. The system of constraints can then be written as

$$
\begin{bmatrix}
I_p & B & 0_p \\
X_p & X_{opp} & -I_p \\
X_p^2 & X_{opp}^2 & -2X_p^2
\end{bmatrix}
\begin{bmatrix}
a_p \\
a_f \\
m
\end{bmatrix} = 0.
$$

(3.6)

For example, in the 1D connectivity in figure 3.1, we have

Figure 3.1: Sample 1D connectivity

$$
B = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 \\
-1 & 1 & 0 & 0 & -1 \\
0 & -1 & 1 & -1 & 0 \\
0 & 0 & -1 & 0 & 1
\end{bmatrix}
$$

$$
X_{opp} = \begin{bmatrix}
x_2 & 0 & 0 & x_3 & 0 \\
-x_1 & x_3 & 0 & 0 & -x_4 \\
0 & -x_2 & x_4 & -x_1 & 0 \\
0 & 0 & -x_3 & 0 & x_2
\end{bmatrix}
$$

Example 3.2.2 (2D, L=1). In this case, the polynomial basis is $\{1, x, y\}$. Thus,

$$
\Phi_p = \begin{bmatrix}
I_p \\
X_p \\
Y_p
\end{bmatrix}, \quad \Phi_{opp} = \begin{bmatrix}
B \\
X_{opp} \\
Y_{opp}
\end{bmatrix}, \quad D^I = \begin{bmatrix}
0_p \\
-I_p \\
0_p
\end{bmatrix}, \quad D^{II} = \begin{bmatrix}
0_p \\
0_p \\
-I_p
\end{bmatrix}.
$$

(3.7)
The system of constraints can then be written as

\[
\begin{bmatrix}
I_p & B & 0_p \\
X_p & X_{opp} & -I_p \\
Y_p & Y_{opp} & 0_p \\
I_p & B & 0_p \\
X_p & X_{opp} & 0_p \\
Y_p & Y_{opp} & -I_p
\end{bmatrix}
\begin{bmatrix}
a_p \\
a_f \\
m
\end{bmatrix} = 0. 
\tag{3.8}
\]

### 3.3 Equivalent Expressions

Consider a row from any matrix. It is well known that adding to it a linear combination of other rows of the same matrix does not change the rank of the matrix, and both the original and new matrix represent the same linear system provided that the right-hand side is modified using the same linear combination.

This section contains some equivalent systems obtained by manipulating the consistency constraint equations in the manner mentioned above. In some cases, the modified forms not only provide further insight into the properties of the system, but also lead to practical advantages during the generation of the coefficients.

#### 3.3.1 Delta Form

Given the consistency constraints from condition C-2

\[
a_{ii}^k \phi_{q_i} + \sum_{j \in s_i} a_{ij}^k \phi_{q_j} = m_i \partial^k \phi_{q_i}
\]

for \(q = 1, \ldots, n_\phi\) polynomials of order up to \(L\), one can multiply the constant consistency condition

\[
a_{ii}^k + \sum_{j \in s_i} a_{ij}^k = 0
\]

by \(\phi_{q_i}\) and subtract the result from the constraints corresponding to the \(q\)-th polynomial in the basis (except that of the constraint corresponding to the constant polynomial as that would result in self-cancel). This results in the “delta” form of
3.3. EQUIVALENT EXPRESSIONS

the constraints:

\[ \sum_{j \in s_i} a_{ij}^k (\phi_{q_j}^k - \phi_{q_i}^k) = m_i (\partial_{\phi}^k \phi^q_i) \]

for \( q = 2, \ldots, n_\phi \).

In matrix terms, this is the same as

\[
\begin{bmatrix}
\bar{C}_p & \bar{C}_f & D
\end{bmatrix}
\begin{bmatrix}
a_p \\
a_f \\
m
\end{bmatrix} = 0 ,
\]

(3.9)

where

\[
\bar{C}_p = \begin{bmatrix}
\bar{\Phi}_p & 0 & 0 \\
0 & \bar{\Phi}_p & 0 \\
0 & 0 & \bar{\Phi}_p
\end{bmatrix}, \quad
\bar{C}_f = \begin{bmatrix}
\bar{\Phi}_f & 0 & 0 \\
0 & \bar{\Phi}_f & 0 \\
0 & 0 & \bar{\Phi}_f
\end{bmatrix},
\]

(3.10)

and

\[
\bar{\Phi}_p = \begin{bmatrix}
I_p \\
0_p \\
\vdots \\
0_p
\end{bmatrix}, \quad
\bar{\Phi}_f = \begin{bmatrix}
B \\
U\Delta_{\phi^2} \\
\vdots \\
U\Delta_{\phi^n} f
\end{bmatrix}.
\]

From the zero blocks in \( \bar{\Phi}_p \), one can observe that this formulation leads to more sparsity in the constraint matrix, an advantage when the system is being solved for the coefficients. In appendix A, one can also see the matrix operations that demonstrate the equivalence of the constraints in delta form and the original matrix form.

3.3.2 Point-Centered Form

The point-centered form is obtained by observing that one can write the consistency constraints at each solution point using a polynomial basis containing powers of the polynomials \( \bar{x} - \bar{x}_i \) instead of \( \bar{x} \). The constraint for constant consistency remains the same as

\[ a_{ii}^k + \sum_{j \in s_i} a_{ij}^k = 0 . \]
The constraints for the rest of the polynomials in the basis look like
\[ a_{ii}^k \tilde{\phi}_{qi} + \sum_{j \in s_i} a_{ij}^k \tilde{\phi}_{qj} = m_i (\partial_i^k \tilde{\phi}_q) . \]

However, using the powers \((\vec{x} - \vec{x}_i)\) means that the derivative term evaluated at the point \(i\) is nonzero only when \(\tilde{\phi}_q\) is a polynomial of total order exactly one and also of order one in the \(k\)-th spatial direction (i.e. \(\tilde{\phi}_q \equiv x - x_i \) when \(k = I\), \(\tilde{\phi}_q \equiv y - y_i \) when \(k = II\), etc.).

In matrix form, the constraints corresponding to the new basis are represented by
\[
\begin{bmatrix}
\tilde{C}_p & \tilde{C}_f & \tilde{D}
\end{bmatrix}
\begin{bmatrix}
a_p \\
a_f \\
m
\end{bmatrix}
= 0 ,
\] (3.12)

where
\[
\tilde{C}_p = \begin{bmatrix}
\tilde{\Phi}_p & 0 & 0 \\
0 & \tilde{\Phi}_p & 0 \\
0 & 0 & \tilde{\Phi}_p
\end{bmatrix},
\tilde{C}_f = \begin{bmatrix}
\tilde{\Phi}_f & 0 & 0 \\
0 & \tilde{\Phi}_f & 0 \\
0 & 0 & \tilde{\Phi}_f
\end{bmatrix},
\tilde{D} = \begin{bmatrix}
\tilde{D}^I \\
\tilde{D}^{II} \\
\tilde{D}^{III}
\end{bmatrix},
\] (3.13)

and
\[
\tilde{\Phi}_p = \Phi_p = \begin{bmatrix}
I_p \\
0_p \\
\vdots \\
0_p
\end{bmatrix},
\tilde{\Phi}_f = \begin{bmatrix}
B \\
(\tilde{\Phi}_2)_{opp} \\
\vdots \\
(\tilde{\Phi}_{n_\phi})_{opp}
\end{bmatrix},
\tilde{D}^k = \begin{bmatrix}
0_p \\
\vdots \\
0_p \\
I_p \\
0_p \\
\vdots \\
0_p
\end{bmatrix}_{n_\phi},
\] (3.14)

where \(\hat{q}\) denotes the index such that \(\tilde{\phi}_{\hat{q}}\) is the linear polynomial in the \(k\)-th spatial dimension. One can see that writing the constraints in this form further improves the sparsity of the constraint matrix. An example of the matrix operations that transform the original constraints to the point-centered ones can be found in appendix A.
Chapter 4

Operator Construction

In section 2.4, the linear dependence of C-1 and C-2 was shown to result in a set of necessary compatibility conditions. In this chapter, two algorithms for generating coefficients that satisfy C-1 and C-2 and these implied constraints will be presented.

While it is not proven that the compatibility conditions are sufficient for the existence of coefficients $a_{ij}^k$ and $m_i$ satisfying C-1 and C-2, there are existing finite volume schemes that do satisfy these conditions. These schemes actually can be expressed in the same algebraic form as the current meshless scheme (see appendix B for details). Numerical experiments in chapter 5 that use either of the algorithms in this chapter also show that the singular system of linear constraints is compatible (i.e., there exists infinitely many solutions) after these compatibility conditions are satisfied.

Many numerical simulations involve the solution of conservation laws in a domain enclosed by some discrete representation of the boundary geometry. Thus, both algorithms begin with a set of corrected boundary normals $n_i^k$ that satisfy corollary 2.4.2 (or equation (2.18)), which is a necessary condition for the existence of compatible coefficients.
4.1 Segregated Approach

In this approach, $m_i$'s that satisfy equation (2.15) are first generated. Then, $a^k_{ij}$'s satisfying C-1 and C-2 are computed. The algorithm is as follows:

1. Calculate initial estimates of $\bar{n}_i$ at each boundary point based on the geometry of the domain boundary.

2. Project the estimates of $\bar{n}_i$'s onto the linear subspace that satisfies (2.18).

3. With initial estimates of $m_i = 0$, project the $m_i$'s into the linear manifold that satisfies (2.15).

4. Solve a constrained least squares problem for the $\bar{a}_{ij}$'s to enforce C-1 and C-2, while minimizing $\sum_{(i,j) \in E} \|\bar{a}_{ij}\|^2_2$, where $E$ is the set of all neighborhood pairs $\{(i,j) \mid j \in s_i\}$.

Let us now go through steps 2 to 4 in detail:

In step 2, let $\mathbf{n}$ be a column vector that contains the $\bar{n}_i$'s for all boundary points $i \in S_B$. The geometric conservation law (2.18) can be written in matrix form as

$$G^T \mathbf{n} = 0. \tag{4.1}$$

The number of columns of the matrix $G$ is equal to the number of linearly independent vector valued, divergence-free multivariate polynomials of maximum order $2L$. Each column of $G$ contains the values of one of these polynomials at all boundary points. When $L$ is small, $G$ is a thin matrix.

To ensure that the total volume enclosed by the boundary faces does not change during the projection process, the additional constraint

$$\frac{1}{n_d} \sum_{i \in S_B} \bar{x}_i \cdot \bar{n}_i = m_0 \tag{4.2}$$

is enforced for each closed boundary of the domain, where $n_d$ is the number of spatial dimensions and $m_0 = \frac{1}{n_d} \sum_{i \in S_B} \bar{x}_i \cdot \bar{n}_{i0}$, where $\bar{n}_{i0}$ is the initial estimate of $\bar{n}_i$. 
4.1. SEGREGATED APPROACH

Let \( n_0 \) be the initial estimate of \( n \) (by concatenating the components of \( \tilde{n}_{i_0} \)'s). The vector \( n \) can be obtained by

\[
R^T y = g - G^T n_0 \\
\Delta n = Qy \\
n = n_0 + \Delta n ,
\]

(4.3)

where \( g = (0, \ldots, 0, m_0)^T \) and \( QR = G \) is the (thin) QR decomposition of \( G \). The projected \( n \) satisfies the linear equation (4.1), which is equivalent to the geometric conservation law (2.18).

For step 3, let us denote \( m \) as the column vector that contains the \( m_i \)'s for all points and write equation (2.15) in matrix form as

\[
D^T m = E^T n .
\]

(4.4)

The matrices \( D \) and \( E \) have the same number of columns, which is equal to the number of linearly independent multivariate polynomials of maximum order \( 2L \). Each column of \( D \) contains the divergence of one of these polynomials at all points. The corresponding column of \( E \) contains the values of the polynomial at all boundary points. When \( L \) is small, both \( D \) and \( E \) are thin matrices.

In order to compute \( m \) that satisfies equation (4.4), thin singular value decomposition (SVD) is performed on \( D \). Equation (4.4) becomes

\[
SU^T m = V^T E^T n ,
\]

where \( D = USV^T \) is the SVD of \( D \). As suggested by the geometric conservation law (2.18), the matrix \( D \) does not have full column rank (recall that this rank deficiency is related to the divergence-free vector-valued multivariate polynomials, and the number of zero singular values of \( D \) is equal to the number of linearly independent divergence-free vector polynomials of order up to \( 2L \)). Moreover, if \( n \) satisfies (4.1), the rows of \( V^T E^T n \) corresponding to the zero singular values are 0’s. Let \( b_1 \) be the rows of \( V^T E^T n \) corresponding to the nonzero singular values, \( U_1 \) be the columns of \( U \)
corresponding to the nonzero singular values, and \( S_1 \) be the square submatrix of \( S \) corresponding to the nonzero singular values. Equation (4.4) becomes

\[
S_1 U_1^T \mathbf{m} = b_1,
\]

which can be satisfied by

\[
\mathbf{m} = U_1 S_1^{-1} b_1.
\]

In addition, just like in finite volume schemes, the \( m_i \)'s are required to be positive, which is not guaranteed by SVD. In the event that some of the \( m_i \)'s are non-positive, an optimization procedure is invoked to minimize \( \| \mathbf{m} \|_2^2 \) subject to equation (4.4) and the positivity of \( m_i \). The positivity constraint is enforced by \( m_i > m_{\min} \), where \( m_{\min} \) is a user-selected parameter, typically on the order of \( \sqrt{\epsilon_{\text{mac}}} \) (\( \epsilon_{\text{mac}} \) is the machine zero) to avoid the virtual volume at any location to come arbitrarily close to zero. The resulting system is a quadratic program, so one can carry out this part of the algorithm using any solvers capable of handling quadratic programs or general convex optimization problems, such as CVX[54, 55], CVXOPT[56] and PDCO[57].

For step 4, let us denote \( \mathbf{a}^k \) as a column vector that contains the \( a_{ij}^k \)'s for all neighborhood point pairs. For each point pair \( (i, j) \), either \( a_{ij}^k \) or \( a_{ji}^k \) is stored, so the reciprocity condition \( a_{ij}^k = -a_{ji}^k \) is automatically satisfied. The constraints C-2 are written in the linear form

\[
\Phi_{\text{opp}} \mathbf{a}^k = \mathbf{d}^k,
\]

where \( \Phi_{\text{opp}} \mathbf{a}^k \) contains the terms \( \sum_{j \in s_i} a_{ij}^k \phi(\vec{x}_j) \), and \( \mathbf{d}^k \) contains both the \( a_{ii}^k \phi(\vec{x}_i) \) and \( m_i \partial_k \phi(\vec{x}_i) \) terms.

The system of constraints (4.5) for the least squares problem can be quite large. A number of tools are again available for solving this problem. For the work in this thesis, the minimum-norm \( \mathbf{a}^k \) that satisfies (4.5) was obtained using the Krylov iterative method LSQR[58], which handles matrices of arbitrary ranks and dimensions. Note that although the system is singular due to the discrete divergence theorem (2.15), it has a compatible right hand side constructed by choosing \( m_i \)'s and \( \vec{n}_i \)'s that satisfy equation (4.4). In section 4.3, one will see an analysis that justifies the use of
the minimum norm as the criterion for selecting the desirable solution among those satisfying the linear constraints.

### 4.1.1 Requirement on Number of Local Neighbors

In addition to the necessary conditions in section 2.4, enough degrees of freedom must exist relative to the number of constraints in the linear system (4.5) in order for step 4 to yield any solutions. For the system in each spatial dimension, there are \( n_f \) unknowns and \( n_\phi n_p \) constraints. Therefore, the additional necessary condition for computing the coefficients \( a_{ij}^k \) is

\[
n_f \geq n_\phi n_p .
\]

(4.6)

Given the reciprocity condition in C-1, we can see that the total number of edges is related to the number of neighbors in the local point cloud by

\[
n_f = \frac{1}{2} \sum_{i=1}^{n_p} (n_{nbr})_i = \frac{1}{2} \overline{n_{nbr}} n_p ,
\]

(4.7)

where \( \overline{n_{nbr}} \) is the average number of neighbors in the local point clouds. Therefore, we can express the necessary condition in terms of the number of local neighbors, i.e.

\[
\overline{n_{nbr}} \geq 2n_\phi .
\]

(4.8)

Table 4.1 shows the minimum average number of local neighbors for various spatial dimensions \( n_d \) and polynomial consistency \( L \).

<table>
<thead>
<tr>
<th>( n_d )</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_{nbr} )</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>( L )</td>
<td>2</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 4.1: Number of local neighbors required for the segregated algorithm.
4.2 Coupled Approach

In this approach, $m_i$’s and $a_{ij}^k$’s that satisfy C-1 and C-2 (and hence equation (2.15)) are simultaneously computed. The coupled algorithm is as follows:

1. Based on the geometry of the domain boundary, calculate estimates of $\vec{n}_i$’s for all boundary points (same as in the segregated approach).

2. Project the estimates of $\vec{n}_i$’s onto the linear subspace that satisfies (2.18) (also same as in the segregated approach).

3. Solve a quadratic program\footnote{For simple small problems, alternative algorithms such as SVD can be used. For instance, see section 5.1.1.} for the $\vec{a}_{ij}$’s and $m_i$’s to enforce C-1 and C-2 while minimizing

$$\sum_{(i,j) \in E} \|\vec{a}_{ij}\|_2^2,$$

where $E$ is the set of all neighborhood pairs $\{(i, j) \mid j \in s_i\}$.

In step 3, the constraints from C-2 are

$$\Phi_{opp} a_f^k + D^k m = \tilde{d}^k,$$\hspace{1cm} (4.9)

where, as before, $a_f$ is the column vector containing the $\vec{a}_{ij}$’s for all point pairs, and $m$ is the vector containing the $m_i$’s for all points. $\Phi_{opp} a_f^k$ contains the terms $\sum_{j \in s_i} a_{ij}^k \phi(\vec{x}_j)$, but now $D^k m$ contains the $m_i \partial^k \phi(\vec{x}_i)$ terms, and $\tilde{d}^k = -C_p a_p^k$ contains boundary terms of the type $a_{ii}^k \phi(\vec{x}_i)$.

Expressed in the matrix notation from chapter 3, equation (4.9) is

$$\begin{bmatrix} C_f & D \end{bmatrix} u = \tilde{d},$$\hspace{1cm} (4.10)
where

\[ C_f = \begin{bmatrix} \Phi_{opp} & 0 & 0 \\ 0 & \Phi_{opp} & 0 \\ 0 & 0 & \Phi_{opp} \end{bmatrix}, \quad D = \begin{bmatrix} D_I \\ D_{II} \\ D_{III} \end{bmatrix}, \quad u = \begin{bmatrix} a_f \\ m \end{bmatrix}, \quad \tilde{d} = \begin{bmatrix} \tilde{d}_I \\ \tilde{d}_{II} \\ \tilde{d}_{III} \end{bmatrix}. \]

One can replace \( C_f \) and \( D \) in system (4.10) by their counterparts in one of the equivalent forms in chapter 3. Any equivalent form of system (4.10), along with the constraints \( m_i > m_{\min} \), then again can be solved using QP or convex optimization tools. Right preconditioning was applied by scaling the columns of \( \Phi_{opp} \) by \( \| \Delta x_{ij} \|_2 \) when enforcing the constraints (and scaling the objective function accordingly).

Note that, although the constraint matrix does not have full row rank because of the discrete divergence theorem (2.15), no problems arise from infeasibility during the solution procedure when the right-hand side is constructed using a set of \( \tilde{n}_i \)'s satisfying equation (4.1). In section 4.3, one will see the justification of the use of the norm of \( a_f \) as the objective function for the optimization problem.

### 4.2.1 Requirement on Number of Local Neighbors

Similar to the case in the segregated approach, one also needs enough unknowns for solutions to the linear system (4.10) to exist. System (4.10) has \( n_f n_d + n_p \) unknowns and \( n_{\phi} n_p n_d \) constraints. After some simple algebra, the additional necessary condition for computing the coefficients \( a_{ij}^k \) is now

\[ n_f \geq n_p \left( n_{\phi} - \frac{1}{n_d} \right). \quad \text{(4.11)} \]

The condition on the local number of neighbors is now\(^2\)

\[ n_{nbr} \geq 2 \left( n_{\phi} - \frac{1}{n_d} \right). \quad \text{(4.12)} \]

\(^2\)The rank deficiency actually relaxes this constraint. However, the effect, given the small dimensions of polynomial bases relative to the number of points, is insignificant.
Table 4.2 shows the minimum average number of local neighbors for various spatial dimensions $n_d$ and polynomial consistency $L$ for the coupled algorithm. One can see that, by coupling $m$ and $a_f$, the number of neighbors required to achieve a given polynomial consistency is decreased.

<table>
<thead>
<tr>
<th>$L$</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>19</td>
<td>39</td>
</tr>
</tbody>
</table>

Table 4.2: Number of local neighbors required for the coupled algorithm.

### 4.3 Choice of Objective Function for an Optimal Global Error Bound

In the solution of the quadratic programming problem for either coefficient generation algorithms, the objective function was simply stated to be $\|a_f\|_2^2$. It makes every sense to question whether this choice of the objective function is a good one. Here, we shall go through an analysis that motivates this choice.

Consider the meshless operator in 1D. Expanding the values of $u_j$ at all the points in the local cloud of $i$ using Taylor’s series, one obtains

$$m_i \delta^x u_i = \sum_j a_{ij} u_j$$

$$= \sum_j a_{ij} (u_i + \Delta x_{ij} \partial^x u_i + \frac{1}{2} \Delta x_{ij}^2 \partial^{xx} u_i + \frac{1}{6} \Delta x_{ij}^3 \partial^{xxx} u_i + \cdots)$$

$$= m_i \partial^x u_i + \sum_j a_{ij} (\frac{1}{2} \Delta x_{ij}^2 \partial^{xx} u_i + \frac{1}{6} \Delta x_{ij}^3 \partial^{xxx} u_i + \cdots). \quad (4.13)$$
4.3. AN OPTIMAL GLOBAL ERROR BOUND

For $L$-th order consistency, one can approximate the error in the derivative approximation by

$$
\int_{\Omega} (\partial^x u - \delta^x u)^2 d\Omega \approx \sum_i m_i (\partial^x u - \delta^x u)^2
$$

$$
= \sum_i \frac{1}{m_i} \left[ \sum_j a_{ij} \left( \frac{1}{(L+1)!} \Delta x_{ij}^{(L+1)} \partial^{(L+1)} x u_i 
\right.
+ \frac{1}{(L+2)!} \Delta x_{ij}^{(L+2)} \partial^{(L+2)} x u_i + \cdots \right)^2
$$

$$
= \sum_i \frac{1}{m_i} \left[ \sum_j a_{ij} \sum_{r=L+1}^{\infty} \frac{1}{r!} \Delta x_{ij}^{(r)} (\partial^{(r)} x u_i) + \cdots \right]^2.
$$

(4.14)

Equation (4.14) can be represented in matrix form as

$$
\int_{\Omega} (\partial^x u - \delta^x u)^2 d\Omega \approx \sum_i m_i (\partial^x u - \delta^x u)^2
$$

$$
= \mathbf{a}^T \mathbf{R}_L^T \mathbf{M}^{-1} \mathbf{R}_L \mathbf{a}
$$

$$
= \| \mathbf{M}^{-\frac{1}{2}} \mathbf{R}_L \mathbf{a} \|_2^2,
$$

(4.15)

where $\mathbf{M}$ is the diagonal matrix of $m_i$’s, $\mathbf{R}_L$ has entries

$$
R_{L,i,j} = \sum_{r=L+1}^{\infty} \frac{\Delta x_{ij}^{(r)}}{r!} (\partial^{(r)} x u)_i,
$$

and $i,j$ is the global edge index of the meshless edge $(i, j)$.

Clearly, the matrix $\mathbf{R}_L^T \mathbf{M}^{-1} \mathbf{R}_L$ is symmetric positive semi-definite. One can decompose it into

$$
\mathbf{R}_L^T \mathbf{M}^{-1} \mathbf{R}_L = \mathbf{Q}^T \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} \mathbf{Q},
$$

(4.16)

where $\mathbf{Q}^T$ is the $n_p \times n_f$ matrix of normalized eigenvectors and $\Lambda$ is the diagonal matrix of non-zero (positive) eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_{n_r}\}$, where $n_r$ is the rank of $\mathbf{R}_L^T \mathbf{M}^{-1} \mathbf{R}_L$. 
One can then write

\[
\|M^{-\frac{1}{2}}R_La\|_2^2 = \|\Lambda^{\frac{1}{2}}Qa\|_2^2 \\
\leq \|\Lambda^{\frac{1}{2}}\|_2^2\|Qa\|_2^2 \\
= \max_i \{\lambda_i\}\|a\|_2^2.
\tag{4.17}
\]

Therefore, minimizing the 2-norm of the coefficient vector is equivalent to minimizing a bound on the global error of the derivative approximation, given a set of virtual volumes \(m_i\) on a point distribution and connectivity. For multiple spatial dimensions, one can write down a similar expression for each of the spatial derivatives. Thus, the segregated algorithm optimizes this upper bound of the approximation error in each spatial dimension based on a given \(m\). The choice of objective function as \(\|a_f\|_2^2\) is optimal in this sense. Alternatively, one can also write down a similar expression in terms of \(a_f\) (i.e. all components of the virtual face areas) by adding together the error expressions in all spatial dimensions. This expression suggests that the coupled algorithm minimizes the total error in all the \(n_d\) spatial derivatives, or the error of the divergence operator. Since the solution of conservation laws inherently requires the divergence operator, the coupled algorithm is also optimizing the error bound based on a given connectivity while adjusting the value of \(m\) to reach the optimum.

Taking advantage of the matrix form, we can further characterize the bound on the maximum eigenvalue \(\lambda_{\text{max}} = \max_i \{\lambda_i\}\). Again, the following analysis will be written in 1D, but one can obtain very similar results in higher dimensions with minimal algebra.

Using Taylor’s theorem, one can bound the magnitude of the entries in \(R_L\) using the Cauchy’s estimate of the remainder

\[
|R_{Li,j}| \leq C_{Lij} \frac{\Delta x_{ij}^{L+1}}{(L+1)!},
\tag{4.18}
\]

where \(C_{Lij}\) is an edge-wise constant such that \(|u^{(L+1)x}| \leq C_L \forall x \in (x_i, x_j)\). Using
4.3. AN OPTIMAL GLOBAL ERROR BOUND

this estimate, we have

\[ \lambda_{\text{max}} = \| M^{-\frac{1}{2}} R_L \|_2^2 \]
\[ \leq \| M^{-\frac{1}{2}} \|_2^2 \| R_L \|_2^2 \]
\[ \leq \| M^{-\frac{1}{2}} \|_2^2 \| R_L \|_F^2 \]
\[ \leq \| M^{-\frac{1}{2}} \|_2^2 \sum_{i,j} \left( C_{L,ij} \frac{\Delta x_{ij}^{L+1}}{(L + 1)!} \right)^2 \]
\[ \leq \frac{1}{\min_i \{ m_i \}} \max_{i,j} \{ \Delta x_{ij}^{2(L+1)} \} \max_{i,j} \{ C_{L,ij}^2 \} \frac{2n_f}{(L + 1)!^2}, \quad (4.19) \]

where \( 2n_f \) is simply the number of entries in the summation on the second-to-last line.

Equation (4.19) suggests that two strategies may help decrease the approximation error: using as few edges as possible and assigning uniform volumes to all solution points. Minimizing the number of connectivity edges minimizes \( n_f \). With connectivity formed by selecting points among closest neighbors, this also leads to a lower value of \( \max_{i,j} \{ \Delta x_{ij}^{2(L+1)} \} \), helping decrease the upper bound on \( \lambda_{\text{max}} \). This argument seems to suggest that the coupled algorithm has an advantage over the segregated one as it allows one to satisfy consistency constraints of a given order with fewer edges. On the contrary, choosing uniform \( m_i \)'s that minimize \( \frac{1}{\min_i \{ m_i \}} \) seems to suggest that using the segregated algorithm with uniform volumes would produce good results.

Therefore, it is not obvious whether such strategies would lead to desirable operators, not to mention that \( m \) and \( a_f \) depend not only on each other, but also on the connectivity and hence \( n_f \). In the next chapter, one will see that results from experiments suggest the coupled algorithm leads to better results — it produces coefficients that lead to more accurate numerical solutions to conservation laws — by decreasing \( \| a_f \|_2 \) at the expense of slightly increasing the minimum value of \( m_i \). Specific details are discussed in section 5.2.
Chapter 5

Numerical Results

Although the meshless scheme presented in the previous chapters is intended to tackle complex multi-dimensional problems, much insight about the characteristics of the scheme can be gained by performing analysis and solving model problems in 1D. In the context of the current meshless discretization, the meaning of “numerical results” is twofold. The meshless coefficients themselves are important numerical results to be studied. In addition, the numerical solutions to conservation laws obtained using different sets of meshless coefficients should be investigated. This chapter presents both kinds of numerical results from test cases involving relatively simple discretized domains and model problems. To begin, section 5.1 contains results from coefficient generation on sample domains. Section 5.1.1 presents 1D results and highlights the differences in the results obtained using the segregated algorithm and the coupled algorithm. Section 5.1.2 features a domain in 2D with connectivity that would cause difficulties for traditional mesh generators. Then, details of the advection model problem are presented in section 5.2. The formulation and time discretization of the advection equation are presented (in a general, multi-dimensional form), followed by the numerical results from solving the 1D and 2D advection equation with different boundary conditions on the sample domains from section 5.1. From these results, one can see the convergent behavior of the meshless scheme and the difference in quality of the numerical solutions obtained using coefficients from the two different algorithms.
5.1 Coefficient Generation

5.1.1 Sample 1D Domain

The domain of interest is defined to be \([0, 1]\). In the following test cases, this domain is discretized with uniformly distributed points indexed from left to right.

During connectivity generation, each point \(i\) is connected to its 4 nearest neighbors. Both the segregated algorithm and the coupled algorithm were used to generate the meshless coefficients \(m_i\) and \(a_{ij}\) that satisfy linear consistency \((L = 1)\). The boundary normals were set to \(\mp 1\) at the respective ends of the domain, satisfying the discrete divergence theorem and geometric conservation law. For the segregated algorithm, the volumes were assigned to be uniform \((\frac{1}{n_p})\) across all points, satisfying the discrete divergence theorem. SVD was used to solve for the values of \(a_{ij}\), minimizing the norm of the unknown vector subject to this choice of \(m_i\)'s. In the coupled approach, the coefficients \(a_{ij}\) and \(m_i\) are solved for together. Because the constraints matrices were small, SVD was also used to obtain the results, minimizing the combined magnitude of the vector containing all the \(a_{ij}\)'s and \(m_i\)'s. Further comments on the effect of minimizing this alternative norm instead of \(\|a\|_2\) will be made later.

Table 5.1 shows the quantities associated with the coefficient generation for \(L = 1\) using the segregated algorithm. One can see that the linear consistency constraints are satisfied at all the solution points. \(\xi_L\) is the vector of the leading term of the pointwise error obtained after expanding the definition of the meshless operator using

<table>
<thead>
<tr>
<th>(n_p)</th>
<th>(e_\infty)</th>
<th>(|a_f|_2)</th>
<th>(|m|_2)</th>
<th>(|u|_2)</th>
<th>(|\xi_L|_\infty)</th>
<th>(|\xi_L|_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2.9340e-14</td>
<td>65.60</td>
<td>0.1414</td>
<td>65.60</td>
<td>9.2461e-2</td>
<td>1.6859e-1</td>
</tr>
<tr>
<td>100</td>
<td>1.0481e-13</td>
<td>194.94</td>
<td>0.1</td>
<td>194.94</td>
<td>4.8158e-2</td>
<td>1.0533e-1</td>
</tr>
<tr>
<td>200</td>
<td>3.9080e-13</td>
<td>564.38</td>
<td>0.07071</td>
<td>564.38</td>
<td>2.4545e-2</td>
<td>6.8397e-2</td>
</tr>
<tr>
<td>400</td>
<td>1.5790e-12</td>
<td>1614.64</td>
<td>0.05</td>
<td>1614.64</td>
<td>1.2387e-2</td>
<td>4.5942e-2</td>
</tr>
<tr>
<td>800</td>
<td>4.6185e-12</td>
<td>4592.84</td>
<td>0.03536</td>
<td>4592.84</td>
<td>6.2218e-3</td>
<td>3.1574e-3</td>
</tr>
</tbody>
</table>

Table 5.1: Results of coefficient generation in 1D using the segregated algorithm.
5.1. COEFFICIENT GENERATION

Taylor series. More specifically,

\[ \xi_{Li} = \sum_{j \in s_i} \frac{1}{(L+1)!} a_{ij} \Delta x^{(L+1)}_{ij} \]

The last two columns of table 5.1 serve as indicators of the expected errors for the discretization. In this case, the scheme is expected to be first-order accurate, precisely what one would expect from satisfying linear consistency.

Similarly, table 5.2 shows the quantities associated with the coefficient generation for \( L = 1 \) using the coupled algorithm. One can see that the linear consistency constraints are satisfied even better — to the order of machine zero — in all cases. Moreover, by allowing the virtual volumes to vary, the coupled algorithm decreased the norm of \( a_f \) by three orders of magnitude. From the expression of the discretization error in section 4.3, one may expect this set of coefficients to give superior accuracy compared to that from the coefficients in the segregated case. This hypothesis is further favored by the factor-of-three decrease in the leading coefficients in the higher order terms of the Taylor’s expansion, even though the scheme is still only guaranteed to be first-order accurate. Figure 5.1 shows the nodal value of \( \xi_i \) calculated using both sets of coefficients for the case with \( n_p = 400 \). One can see that the coupled algorithm produced a set of coefficients that led to much smaller leading error terms everywhere in the domain. In section 5.2.4, one will see that the accuracy of numerical solutions to model test problems varies according to the coefficients used, as suggested by the claims above.

A few figures are included to allow further visualization of the results. The virtual

<table>
<thead>
<tr>
<th>( n_p )</th>
<th>( e_\infty )</th>
<th>( | a_f |_2 )</th>
<th>( | m |_2 )</th>
<th>( | u |_2 )</th>
<th>( | \xi_L |_\infty )</th>
<th>( | \xi_L |_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1.1380e-15</td>
<td>1.4952</td>
<td>0.1452</td>
<td>1.5022</td>
<td>3.2595e-2</td>
<td>5.8755e-2</td>
</tr>
<tr>
<td>100</td>
<td>1.6376e-15</td>
<td>2.1762</td>
<td>0.1013</td>
<td>2.1785</td>
<td>1.6149e-2</td>
<td>2.9121e-2</td>
</tr>
<tr>
<td>200</td>
<td>2.1649e-15</td>
<td>3.1202</td>
<td>0.0712</td>
<td>3.1210</td>
<td>8.0361e-3</td>
<td>1.4492e-2</td>
</tr>
<tr>
<td>400</td>
<td>2.5535e-15</td>
<td>4.4425</td>
<td>0.0502</td>
<td>4.4428</td>
<td>4.0082e-3</td>
<td>7.2285e-3</td>
</tr>
<tr>
<td>800</td>
<td>3.7470e-15</td>
<td>6.3036</td>
<td>0.0354</td>
<td>6.3037</td>
<td>2.0016e-3</td>
<td>3.6098e-3</td>
</tr>
</tbody>
</table>

Table 5.2: Results of coefficient generation in 1D using the coupled algorithm.
Figure 5.1: Representative error term $|\frac{1}{m_i} \sum_j a_{ij} \Delta x_{ij}^2|$ for each set of coefficients. Upper: segregated coefficients with uniform volumes. Lower: coupled coefficients.
5.1. COEFFICIENT GENERATION

volumes computed using the coupled algorithm for \( n_p = 50 \) are shown in figure 5.2. Figure 5.3 shows the coefficients from the view of a solution point close to the middle of the domain (with a sign convention that all edges start from the solution point of interest). Finally, figure 5.4 features global views of the coefficients \( a_{ij} \) and \( a_{ji} \). There, the plots show the adjacency matrix with entries weighed by the value of \( a_{ij} \)'s. A dot at the \((i, j)\)-th entry indicates that points \( i \) and \( j \) are connected by an edge. The color of the dot indicates the value of the coefficient \( a_{ij} \). The \( a_{ji} \)'s are colored using the negative of the corresponding \( a_{ij} \)'s.

Figure 5.2: Meshless coefficients \( m_i \) in 1D domain computed using the coupled algorithm, \( n_p = 50 \).

Two observations must be emphasized. First of all, the minimum-norm solution possesses some intuitive symmetry. Figure 5.2 shows that the coefficients \( m_i \) are symmetric about the midpoint of the domain. In figures 5.3 and 5.4, one can also see that the linear consistency conditions lead to values of \( a_{ij} \) with an anti-symmetry that is typical of discrete first derivative operators on uniform point distributions. Moreover, figure 5.4 shows that the same type of connectivity edges all had practically the same value of \( a_{ij} \). These trends were also observed for the cases with higher point
Figure 5.3: Visualization of meshless coefficients $a_{ij}$ in 1D domain, from the view of a solution point, $n_p = 50$. Upper: segregated coefficients with uniform volumes. Lower: coupled coefficients. Dashed lines connect each coefficient marker to the two points that defines the connectivity edge associated with the coefficient.
5.1. COEFFICIENT GENERATION

Figure 5.4: Visualization of meshless coefficients $a_{ij}$ in 1D domain, $n_p = 50$. Upper: segregated coefficients with uniform volumes. Lower: coupled coefficients.
densities, as shown in figures 5.5–5.7. Even though it is likely that the symmetry was a by-product of the specification of a minimum-norm solution satisfying linear consistency, it is still quite surprising as no such symmetry conditions were specified in any part of the coefficient generation process.

The second observation is the large difference in magnitude of the coefficients $a_{ij}$ obtained by the two algorithms. This was previously highlighted in tables 5.1 and 5.2, but can now be seen in more detail. From figure 5.4, one can tell that each coefficient generated by the segregated algorithm has a larger magnitude than its counterpart from the coupled algorithm. The symmetry (or anti-symmetry) mentioned above helps explain this phenomenon. Specifically, in figure 5.3, one can see that, with fixed values of $m_i$, the segregated algorithm could only arrive at the anti-symmetry inherent to the linear consistency constraints by producing $a_{ij}$’s with large magnitudes, such that the correct estimates of the basis polynomial derivatives are obtained with some cancellation between the contributions of the nearest neighbors and the more distant ones. Furthermore, the values of $a_{ij}$ were very sensitive to those of $m_i$, evident from the small difference in the $m_i$’s between the two cases. In both cases, the values of $m_i$ remained very close to the uniform average of 0.02 in most of the domain.
Figure 5.5: Visualization of meshless coefficients $a_{ij}$ in 1D domain, $n_p = 100$. Upper: segregated coefficients with uniform volumes. Lower: coupled coefficients.
Figure 5.6: Visualization of meshless coefficients $a_{ij}$ in 1D domain, $n_p = 200$. Upper: segregated coefficients with uniform volumes. Lower: coupled coefficients.
Figure 5.7: Visualization of meshless coefficients $a_{ij}$ in 1D domain, $n_p = 400$. Upper: segregated coefficients with uniform volumes. Lower: coupled coefficients.
5.1.2 Sample 2D Domain

The domain under consideration is $\Omega = \{ \vec{x} \mid 1 < \|\vec{x}\|_2 < 5 \}$. Four point clouds with 165, 563, 2060 and 7896 points, respectively, were formed. Each point cloud is the union of a Cartesian grid in the interior and evenly spaced points on the boundary. To generate the neighborhood set $s_i$ for each point $i$, each local cloud $s_i$ was first set to be the 8 points nearest to point $i$, and then set to $s_i \leftarrow s_i \cup \{ j \mid i \in s_j \}$ to enforce the reciprocity condition. Although the point clouds look relatively simple at first glance, it is actually difficult to generate a traditional finite volume mesh using the current connectivity involving at least 8 neighbors per point. Figure 5.8 shows the point cloud and neighborhood sets for $n_p = 165$.

Construction of $n^k_i$, $m_i$ and $a^{k}_{ij}$

In this example, the coupled algorithm was used to generate coefficients with polynomial order $L = 1$. At each boundary point $i$, an initial estimate of $n^k_i$ was computed by averaging the normals of the two adjacent boundary faces. Using the algorithm in section 4.2, convex optimization was used to generate the remaining coefficients after the boundary face estimates had been corrected for compatibility. The objective function was $\|a\|^2_2$, which is equivalent to replacing the error matrix $R_L^T M^{-1} R_L$ in equation (4.15) (except now in 2D) by the identity matrix.

In this study, the optimization was carried out in MATLAB using CVX [54,55]. For the largest point cloud, coefficients satisfying all the pointwise consistency constraints to order $1 \times 10^{-6}$ or better were obtained in about a minute on a workstation with Intel Xeon 5160 processors in a quad-core configuration. For larger-scale problems, one can utilize other optimization software libraries [59-61]. Since the optimization problem is a quadratic program, it is also possible for a QP-specific solver to yield better efficiency.

Figure 5.9 shows scatter plots of the virtual volumes $m_i$ and virtual face area magnitudes $\|\vec{a}_{ij}\|_2$ for $n_p = 165$. The semi-toroidal shape formed by the coefficient magnitudes in figure 5.9 shows that the enforced consistency constraints were enough to make the magnitudes of $\vec{a}_{ij}$’s vary with local point spacing and connectivity. The
Figure 5.8: The point cloud and the neighborhood pairs with 165 points (the coarsest point cloud). Open circles: interior points. Filled circles: boundary points. A line connecting two points indicates that they are mutual neighbors.
Figure 5.9: Virtual volumes and virtual face area magnitudes computed using the coupled algorithm. Left: virtual volumes. Right: virtual face areas. $n_p = 165$. 
virtual faces are the smallest near the boundaries, where the shortest connectivity edges are located. The size of the faces increases smoothly from these small boundary values to the largest values on the interior of the annulus, where the connectivity edges are longer and more uniform. The virtual volumes $m_i$ also adjust similarly and accordingly. Furthermore, this behavior was observed in all current test cases and also in other unreported test problems. It is very encouraging that the solutions generated by the presented algorithms are both unique (because of the minimum-norm problem) and physically sensible.

5.2 Advection Equation

The advection problem is a classic model problem for assessing the accuracy and stability of numerical schemes. In this thesis, test problems involving the 1D and 2D advection equation are featured in sections 5.2.4 and 5.2.6 respectively. Before the presentation of numerical results, sections 5.2.1–5.2.3 contain the details of the discretization used in both the 1D and 2D test problems. For generality, the discretization will be presented in 3D.

5.2.1 Advection Equation with Prescribed Inflow

Consider the equation

$$\frac{\partial u}{\partial t} + \vec{V} \cdot \nabla u = 0$$

with boundary condition $u(x, t) = u_B(x, t)$ at the inlet part of the boundary characterized by $\{x \in \partial \Omega \mid \vec{n} \cdot \vec{V} < 0\}$. The advection velocity $\vec{V} = (V_I, V_{II}, V_{III})$ is constant. This equation is discretized as

$$\frac{du_i}{dt} + V^k \delta^k u_i = \begin{cases} 0 & i \notin s_B \\ u_B - u_i [V^n_i]_n & i \in s_B \\ \frac{m_i}{m_i} [V^n_i]_n & i \in s_B \end{cases}$$

\[5.2\]

Superscript $k$ follows Einstein notation.
where $V_i^n = n_i^k V^k$ and $[V_i^n]_- = -\min(0, V_i^n)$ denotes the negative part of $V_i^n$. The penalty term in the discretization (5.2) is consistent with the continuous boundary condition.

### 5.2.2 Stability at the Semi-Discrete Level

The stability of the above discretization can be proven using corollary 2.2.3. Multiplying equation (5.2) by $m_i u_i$, summing the result over $i$, and using equation (2.10), we get

$$\frac{de}{dt} = -\sum_{i \in s_B} \left(-\frac{u_i^2}{2} u_i^n - (u_i^2 - u_i u_B)[V_i^n]_-\right),$$

where the energy is defined by

$$e = \sum_{i=1}^{np} \frac{1}{2} m_i u_i^2.$$

By splitting $V_i^n$ into positive and negative parts so that $V_i^n = [V_i^n]_+ - [V_i^n]_-$, we then get

$$\frac{de}{dt} = \frac{1}{2} \sum_{i \in s_B} \left(-u_i^2 [V_i^n]_+ - (u_i - u_B)^2 [V_i^n]_- + u_B^2 [V_i^n]_-\right).$$

The first two terms, which represent respectively the energy convected out of the domain and the penalty term on the boundary, cannot increase the total energy. The third term corresponds to the energy convected into the domain and depends only on the boundary condition. Therefore, the energy cannot increase exponentially, i.e. the semi-discrete scheme (5.2) is stable.

### 5.2.3 Time Stepping and Discrete Stability

The Crank-Nicolson (C-N) scheme in time was used to obtain the results in the next few sections. It is well-known that C-N is unconditionally stable when used with central difference schemes on the advection equation. Here, it will be shown that C-N is also unconditionally stable when applied to the linear advection equation with the current meshless spatial discretization.
The fully discrete scheme can be written as

\[
\frac{u_i^{(t+1)} - u_i^{(t)}}{\Delta t} + V^k \delta^k u_i^{(t+\frac{1}{2})} = \begin{cases} 
0 & i \notin s_B \\
\frac{u_B^{(t+\frac{1}{2})} - u_i^{(t+\frac{1}{2})}}{m_i} [V^n_i]_- & i \in s_B
\end{cases},
\]  

(5.3)

where

\[
u_i^{(t+\frac{1}{2})} = \frac{1}{2} (u_i^{(t)} + u_i^{(t+1)}) .
\]

Just like in the semi-discrete case, multiplying equation (5.3) by \( m_i u_i^{(t+\frac{1}{2})} \) and summing over \( i \) gives

\[
\frac{e^{(t+1)} - e^{(t)}}{\Delta t} = - \sum_{i \in s_B} \left( \frac{(u_i^{(t+\frac{1}{2})})^2}{2} V^n_i - \left( (u_i^{(t+\frac{1}{2})})^2 - u_i^{(t+\frac{1}{2})} u_B^{(t+\frac{1}{2})} [V^n_i]_- \right) \right)
\]

and

\[
\frac{e^{(t+1)} - e^{(t)}}{\Delta t} = \frac{1}{2} \sum_{i \in s_B} \left( - (u_i^{(t+\frac{1}{2})})^2 [V^n_i]_+ - (u_i^{(t+\frac{1}{2})} - u_B^{(t+\frac{1}{2})})^2 [V^n_i]_- + (u_i^{(t+\frac{1}{2})})^2 [V^n_i]_- \right)
\]

where the energy is now defined at each time step as

\[
e^{(t)} = \sum_{i=1}^{n_p} \frac{1}{2} m_i (u_i^{(t)})^2 .
\]

The arguments carry over from the semi-discrete case, showing that the fully discrete scheme is stable.

### 5.2.4 1D Results

Here, numerical results from solving the advection equation in the 1D domain discretized in section 5.1.1 are presented. The one-dimensional advection velocity is
taken to be unity. The respective initial and boundary conditions of the time-
dependent problem are

\[ u(x, 0) = 0 \]
\[ u(0, t) = \sin 2\pi t. \]

The boundary condition was enforced using the penalty term as shown in equation (5.2). To advance the solution in time, the Crank-Nicolson scheme mentioned in section 5.2.3 was used with a time step of \( \Delta t = 0.001 \). Figure 5.10 plots the numerical solutions at \( t = 2 \), obtained using coefficients computed by both coefficient generation algorithms with 400 solution points in the domain, against the exact solution \( u_{\text{exact}}(x, 2) = -\sin(2\pi x) \). Visually, the coupled coefficients lead to better numerical accuracy, although both sets of coefficients satisfy conservation and linear consistency.

For quantitative comparison of the results, tables 5.3 and 5.4 show the errors of the numerical solutions at \( t = 2 \), computed using coefficients from the segregated and coupled algorithms respectively, with various point densities in the domain. Figure 5.11 plots the error convergence in both the infinity and \( L_2 \) norms. One can see that solutions from both sets of coefficients converge to the exact solution with increasing point density. However, the coupled coefficients lead to less numerical error at all times, in addition to producing a higher order of convergence. The segregated coefficients show first-order convergence, while the coupled coefficients allow convergence that is almost second-order on a discretization that is formally first-order accurate.
Figure 5.10: Solutions to the 1D advection equation, $t = 2$. Upper: segregated coefficients with uniform volumes. Lower: coupled coefficients. Red: exact solution. Black: numerical solution.
<table>
<thead>
<tr>
<th>$n_p$</th>
<th>$|e|_{\infty}$</th>
<th>$|e|_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>8.266e-1</td>
<td>4.352e-1</td>
</tr>
<tr>
<td>200</td>
<td>2.261e-1</td>
<td>1.146e-1</td>
</tr>
<tr>
<td>400</td>
<td>1.325e-1</td>
<td>6.230e-2</td>
</tr>
<tr>
<td>800</td>
<td>7.331e-2</td>
<td>3.097e-2</td>
</tr>
</tbody>
</table>

Table 5.3: Numerical error for 1D advection equation using segregated coefficients.

<table>
<thead>
<tr>
<th>$n_p$</th>
<th>$|e|_{\infty}$</th>
<th>$|e|_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>5.364e-2</td>
<td>1.496e-2</td>
</tr>
<tr>
<td>200</td>
<td>2.024e-2</td>
<td>4.124e-3</td>
</tr>
<tr>
<td>400</td>
<td>8.849e-3</td>
<td>1.320e-3</td>
</tr>
<tr>
<td>800</td>
<td>4.267e-3</td>
<td>4.485e-4</td>
</tr>
</tbody>
</table>

Table 5.4: Numerical error for 1D advection equation using coupled coefficients.

Figure 5.11: Variation of error in solution to the 1D advection equation, computed using the meshless algorithm, with respect to number of points.
5.2. ADVECTION EQUATION

5.2.5 1D Results with Periodic Boundary Condition

To investigate the accuracy of the discretization in the absence of effects from boundary conditions, the advection equation (5.1) was solved with the same set of parameters, but now with the initial solution

\[ u(x, 0) = \sin(2\pi x) \]

and periodic boundary conditions. Figure 5.12 plots the numerical solutions at \( t = 2 \) with 400 points in the domain. The exact solution is \( u_{\text{exact}}(x, 2) = \sin(2\pi x) \). Once again, coefficients from using uniform weights and the segregated approach resulted in larger numerical errors (here seen as dispersion of the wave) than the coefficients from the coupled approach. Tables 5.5 and 5.6 show the errors of the numerical solutions at \( t = 2 \) for various number of solution points.

<table>
<thead>
<tr>
<th>( n_p )</th>
<th>( |e|_\infty )</th>
<th>( |e|_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>9.310e-1</td>
<td>6.160e-1</td>
</tr>
<tr>
<td>200</td>
<td>4.331e-1</td>
<td>3.006e-1</td>
</tr>
<tr>
<td>400</td>
<td>2.377e-1</td>
<td>1.484e-1</td>
</tr>
<tr>
<td>800</td>
<td>1.135e-1</td>
<td>7.362e-2</td>
</tr>
</tbody>
</table>

Table 5.5: Numerical error for periodic 1D advection equation using segregated coefficients.

<table>
<thead>
<tr>
<th>( n_p )</th>
<th>( |e|_\infty )</th>
<th>( |e|_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>6.083e-2</td>
<td>2.455e-2</td>
</tr>
<tr>
<td>200</td>
<td>1.347e-2</td>
<td>5.572e-3</td>
</tr>
<tr>
<td>400</td>
<td>3.286e-3</td>
<td>1.371e-3</td>
</tr>
<tr>
<td>800</td>
<td>7.240e-4</td>
<td>3.594e-4</td>
</tr>
</tbody>
</table>

Table 5.6: Numerical error for periodic 1D advection equation using coupled coefficients.

Comparing these results with those in tables 5.3 and 5.4, one can see that, as the inflow boundary conditions were removed, the numerical error in the solutions
Figure 5.12: Solutions to the periodic 1D advection equation at $t = 2$. Upper: segregated coefficients with uniform volumes. Lower: coupled coefficients. Red: exact solution. Black: numerical solution.
5.2. ADVECTION EQUATION

Figure 5.13: Variation of error in solution to the periodic 1D advection equation, computed using the meshless algorithm, with respect to number of points.

Computed using the segregated coefficients increased at all point densities, while the accuracy of the solutions computed using the coupled coefficients actually improved on finer point distributions. The convergence plots in figure 5.13 best highlight the difference. With the coupled coefficients, the convergence was second order, as opposed to first order with the segregated coefficients.

5.2.6 2D Results

For the 2D test cases, the advection equation (5.1) with advection velocity \((V^1, V^2) = (-\sqrt{2}/2, \sqrt{2}/2)\) was solved in the domain described in section 5.1.2. The initial and boundary conditions are

\[
\begin{align*}
    u(x, t) &= \frac{\sqrt{2}}{20} (x + y) + 1 , & t = 0 \\
    u(x, t) &= \frac{\sqrt{2}}{20} (x + y) + \cos[t - 5 - V^k x^k]_+ , & x \in \partial \Omega , \; n^k V^k < 0 .
\end{align*}
\]
The exact solution to this initial-boundary-value problem is

\[ u(x, t) = \frac{\sqrt{2}}{20} (x + y) + \cos(t - 5 - V^k x^k) + . \]

As in the 1D cases, the numerical solutions were computed using the discretization (5.2). To assess the convergence of the scheme, the \( L_\infty \) and \( L_2 \) norms of the error in the numerical solutions against the analytic solution were also computed.

Figure 5.14 shows the numerical solution computed by the central flux scheme (5.2) at \( t = 100 \), again obtained using Crank-Nicolson time stepping. One can see that the resolution of the solution improves with increasing point cloud density as expected. From figure 5.16, one can also see that the scheme is roughly second-order accurate, as one would expect from similar finite volume schemes.

With these promising results, the next chapter of this thesis will focus on how one can extend the meshless scheme to a general, practical and efficient framework for tackling a wide range of problems.
Figure 5.14: Solutions to the 2D advection equation at $t = 100$ using the central flux scheme (5.2). Point clouds of size $n_p = 165, 563$. 
Figure 5.15: Solutions to the 2D advection equation at $t = 100$ using the central flux scheme (5.2). Point clouds of size $n_p = 2060, 7896$. 
Figure 5.16: Numerical error of the central scheme (5.2) as a function of point cloud size $n_p$. 
Chapter 6

A Generalized Meshless Framework

A major reason that finite volume discretization on meshes receives such popularity is because of its ability to be combined with a wide range of numerical methods, such as upwinding, artificial dissipation, flux limiters, etc., giving it the flexibility to tackle a wide range of problems in computational mechanics.

The results in the previous chapters have demonstrated that the current conservative meshless discretization is convergent on classic model problems. To show that this discretization has the potential to operate at a level of flexibility as high as that of finite volume discretization, an extension of the meshless discretization that allows the use of general flux formulae at virtual interfaces will be presented here. Examples from solving the 2D advection equation will be shown and compared to the results previously obtained in section 5.2.6. In chapter 7, the power of this generalization will be further demonstrated by the numerical solution of the Euler equations using some well established techniques originated from finite volume discretizations.
6.1 Geometric Interpretation of the Meshless Coefficients

To see that the current formulation actually leads to a more general conservative meshless framework, recall from theorem 2.3.1 that the reciprocity condition in C-1 ($a_{ij}^k = -a_{ji}^k$) guarantees that if the $a_{ij}^k$'s represent components of (half of) the face normal vectors, these faces then form parts of the enclosures of some consistent volumes represented by $m_i$ and $m_j$. Moreover, equation (2.13) shows that the consistency conditions C-2 allow the discrete derivative to be represented by the weighted sum of the symmetric interface values $u_f = \frac{u_i + u_j}{2}$.

Thus, the global conservation properties in section 2.2 and the local properties in section 2.3 resulting from the geometric interpretation of the meshless coefficients can be viewed as results of the existence of common virtual face areas at virtual interfaces and symmetric interface fluxes. Therefore, discrete conservation can actually be achieved regardless of the interface flux formulation, provided that the flux is symmetric. More precisely, instead of equation (2.13), one can formulate the derivative operator in a more general form:

$$m_i \delta_f^k u_i = 2a_{ii}^k u_i + \sum_{j \in s_i} 2a_{ij}^k F_{ij},$$

where the symmetric interface flux $F_{ij}$ can be a function of $u_i, u_j$, the derivative of $u$ at the $i$-th and $j$-th points, and so on. The coefficients $n_i^k$, $m_i$ and $a_{ij}^k$ representing boundary normals, virtual cell volumes and interface normals can still be generated by one of the algorithms in chapter 4. However, instead of using the central average flux $u_f = \frac{u_i + u_j}{2}$, one can now apply more sophisticated interface fluxes while preserving conservation.

An example of the generalized derivative operator is an upwind scheme, whose
6.2. ADVECTION REVISITED

interface flux $F_{ij}$ is defined by

$$
F_{ij} = \begin{cases} 
  u_i + \frac{x_j^k - x_i^k}{2} \delta_i^k u_i, & a_{ij}^k u^k < 0 \\
  u_j + \frac{x_i^k - x_j^k}{2} \delta_j^k u_j, & a_{ij}^k u^k > 0 \\
\end{cases} ,
$$

(6.2)

where $\delta_i^k$ is a first-order accurate reconstruction of the derivative in the $k$-th spatial dimension. Section 6.2 contains results from solving the 2D advection equation using this scheme. There, $\delta_i^k$ is constructed as

$$
\delta_i^k u_i = \sum_{j \in s_i} a_{ij}^k (u_j - u_i) ,
$$

where $a_{ij}^k$ are chosen by solving

$$
\min \sum_{j \in s_i} (a_{1ij}^k)^2 \quad \text{s.t.} \quad \sum_{j \in s_i} a_{1ij}^k (x_j^k - x_i^k) = d_{kk'}
$$

for each $i$ and $k$, where $k' = I, II, III$ and $d$ represents the Kronecker delta. The second-order accuracy of this particular upwind flux $F_{ij}$ leads to a first-order accurate approximation of the derivative in equation (6.1). Higher order fluxes are certainly possible with the availability of many existing reconstruction schemes developed for finite volume methods.

6.2 Advection Revisited

To show the ability of the generalized framework to accommodate more general numerical flux functions, equation (5.1) was discretized with the upwind derivative $\delta_i^k$ discussed in section 6.1:

$$
\frac{du_i}{dt} + V^k \delta_f^k u_i = \begin{cases} 
  0 & i \notin s_B \\
  \frac{u_B - u_i}{m_i} [V_i^n]^+ & i \in s_B \\
\end{cases}.
$$

(6.3)
Although this scheme does not conserve kinetic energy like the central flux scheme (5.2), it does maintain global and local conservation properties.

Numerical solutions were obtained in exactly the same manner as in section 5.2.6. Figures 6.1–6.3 show numerical solutions at $t = 100$ and the numerical error of the upwind scheme (6.3). Compared with results in section 5.2.6, aside from additional smoothness in the solution profile, the solutions computed using both schemes are very similar, especially as the point density increases. Point convergence study also shows that the asymmetric scheme was roughly second-order accurate. Most importantly, these results confirm the potential and flexibility of the current conservative meshless framework to harness well proven schemes developed for conservation laws. This greatly reduces the overhead for integrating this new framework into existing solution procedures, making it a very attractive option.
Figure 6.1: Solutions to the 2D advection equation at $t = 100$ using the upwind scheme (6.3). Point clouds of size $n_p = 165, 563$. 
Figure 6.2: Solutions to the 2D advection equation at $t = 100$ using the upwind scheme (6.3). Point clouds of size $n_p = 2060, 7896$. 
Figure 6.3: Numerical error of the upwind scheme (6.3) as a function of the point cloud size $n_p$. 

$N$  

$10^{-3}$ 

$10^{-2}$ 

$10^{-1}$ 

$10^0$ 

$10^1$ 

$10^2$ 

$10^3$ 

$10^4$ 

$10^5$
Chapter 7

Application to Euler Equations

So far, the conservative meshless framework has been shown to perform very well in linear test problems with relatively smooth solutions.

A major motivation for developing a conservative meshless framework is to handle discontinuities that arise in general nonlinear coupled systems of conservation laws. Therefore, it is imperative that the framework be tested on such systems.

The Euler equations for compressible flow make a perfect candidate for such tests. Engineers still routinely use numerical solutions to the Euler equations to predict aerodynamic quantities such as lift, drag, and heat transfer, in the transonic and supersonic regimes in which shock waves frequently form on or near aerodynamic surfaces. This is because the Euler equations are very accurate when the effects of viscosity are confined to within thin boundary layers even though the equations do not model viscous effects. The engineering systems involved can feature highly complex geometries (e.g. the space shuttle). Fast, robust, and automated procedures for solving the Euler equations are constantly in demand.

Furthermore, before certain numerical strategies became standard practices, researchers spent significant amounts of time on finding stable meshed discretizations for the Euler equations, especially in the presence of shock waves. Therefore, a great variety of discretization strategies and numerical results are available for comparison, making it a classic system of conservation laws, not to mention that new CFD algorithms are often validated on the Euler equations, adding to the tally of available
benchmarks.

As for the contents in this chapter, section 7.1 briefly outlines the formulation of the Euler equations. Sections 7.2–7.4 explain the discretization procedure used. The details about how the generalized framework was used to build a stable conservative meshless discretization for the Euler equations through leveraging finite volume flux schemes is explained, followed by discussions on the boundary conditions, time discretization and convergence acceleration techniques used in the numerical experiments. The final sections in the chapter feature numerical results. As before, the numerical results are split into two parts. Section 7.6 presents results for coefficient generation. Section 7.7 features the numerical solutions obtained by using the generalized meshless framework to compute subsonic and transonic flow solutions around 2D airfoils.

### 7.1 The Euler Equations

In 2D, the Euler equations are

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} = 0$$

(7.1)

with

\[
\mathbf{w} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u h \end{pmatrix}, \quad \mathbf{g} = \begin{pmatrix} \rho v \\ \rho v u \\ \rho v^2 + p \\ \rho v h \end{pmatrix},
\]

where

$$h = e + \frac{p}{\rho},$$

and \( \rho, u, v, P, E, \) and \( H \) are, respectively, the density, \( x \)- and \( y \)-components of the flow velocity, pressure, total energy and total enthalpy. The total energy is related to other flow quantities by the ideal gas law, which can be expressed as

$$e = \frac{p}{(\gamma - 1)\rho} + \frac{1}{2}(u^2 + v^2),$$
where \( \gamma \) is the ratio of specific heats. For convenience and consistency with the notation used in other chapters, the notation \( \mathbf{F} = (F^I, F^II) \equiv (\mathbf{f}, \mathbf{g}) \) is introduced. The Euler equations are accurate in flow regimes in which viscous effects are insignificant outside boundary layers.

### 7.2 Stabilization and High Resolution Schemes in Meshless Framework

The Euler equations are a hyperbolic system of conservation laws modeling the transport of a system of superpositioned waves. Many researchers have spent effort on developing stable, accurate and cheap schemes for such conservation laws. In particular, they devoted time to create schemes that handle the possible discontinuities resulting from nonlinear fluxes. When such discontinuities are present, it is desirable for schemes to capture the discontinuities without spurious oscillations. Total variational diminishing (TVD)[62] and local extremum diminishing (LED)[63] principles are classic efforts for this purpose and are widely applied in finite volume discretizations.

While there are various ways to understand stabilization techniques (such as through characteristic decomposition and domain of dependence analysis), the key to these stabilization techniques is essentially to add the appropriate amount of artificial diffusion adequate for stability while preserving accuracy to a desired degree. For the class of schemes that are second-order accurate on smooth meshes, a large class of schemes is covered by the general definition

\[
\tilde{f}_{ij} = \frac{1}{2} (f_{ni} + f_{nj}) - d_{ij},
\]

where an interface flux \( \tilde{f}_{ij} \) between point \( i \) and point \( j \) consists of the central average of the area-weighted interface normal fluxes at points on both sides of the interface, and the modificative diffusive flux \( d_{ij} \). Recall that, in the meshless context, the coefficients \( a_{ij}^k \) can be treated as interface areas. One can therefore define the interface-normal
flux to be
\[ f_n = \sum_k 2 a_{ij}^k F^k. \]
A few examples of such fluxes that also satisfy the LED principle are now discussed.

### 7.2.1 Matrix Diffusion

Matrix diffusion closely relates to the decomposition of the conservative variables into characteristic waves[64]. The main idea behind such schemes is that, given the characteristic decomposition of the waves, one can diagonalize an approximate Jacobian \( A_{ij} \) satisfying
\[ A_{ij}(w_j - w_i) = f_{nj} - f_{ni} \]
into
\[ A_{ij} = T\Lambda T^{-1}, \]
where \( \Lambda \) contains the eigenvalues representing the speeds of individual waves, and \( T \) is the matrix representing the transformation from conservative to characteristic variables. The columns of \( T \) are eigenvectors of \( A_{ij} \). With this diagonalization, the quantity
\[ d_{ij} = |A_{ij}|(w_j - w_i), \]
where \( |A_{ij}| = T|\Lambda|T^{-1} \), produces an upwind scheme with different dissipation added to each characteristic wave to achieve stability. One such dissipation is the one resulting from a linearization by Roe, where, given two states of the conservative variables, e.g. at points \( i \) and \( j \), the interface Jacobian \( A_{ij} \) is evaluated using specially averaged variables
\[
 u = \frac{\sqrt{\rho_j} u_j + \sqrt{\rho_i} u_i}{\sqrt{\rho_i} + \sqrt{\rho_j}}, \quad v = \frac{\sqrt{\rho_j} v_j + \sqrt{\rho_i} v_i}{\sqrt{\rho_i} + \sqrt{\rho_j}}, \quad H = \frac{\sqrt{\rho_j} H_j + \sqrt{\rho_i} H_i}{\sqrt{\rho_i} + \sqrt{\rho_j}},
\]
7.2. MESHLESS STABILIZATION / HIGH RESOLUTION SCHEMES

In 2D, this results in a Jacobian matrix of the form

\[
A = \begin{bmatrix}
0 & n^I & n^H & 0 \\
n^I(\gamma - 1)\frac{q^2}{2} - uu_n & u_n - (\gamma - 2)n^Iu & n^H u - (\gamma - 1)n^I v & n^I(\gamma - 1) \\
n^H(\gamma - 1)\frac{q^2}{2} - vu_n & n^H v - (\gamma - 1)n^H u & u_n - (\gamma - 2)n^H v & n^H(\gamma - 1) \\
u_n \left(\gamma - 1\right)\frac{q^2}{2} - H & n^H H - (\gamma - 1)uu_n & n^H H - (\gamma - 1)vu_n & \gamma u_n
\end{bmatrix}
\]

for the direction normal to the interface, with

\[
T = \begin{bmatrix}
1 & 0 & 1 & 1 \\
u & cn^H & u + cn^I & u - cn^I \\
v & -cn^I & v + cn^H & v - cn^H \\
\frac{q^2}{2} & c(n^H u - n^I v) & H + cu_n & H - cu_n
\end{bmatrix} \Lambda = \begin{bmatrix}
u_n & 0 & 0 & 0 \\
0 & u_n & 0 & 0 \\
0 & 0 & u_n + c & 0 \\
0 & 0 & 0 & u_n - c
\end{bmatrix},
\]

where \(n^I, n^H\) are the components of the unit interface normals, \(u_n = un^I + vn^H\) is the velocity component normal to the interface, \(q^2 = u^2 + v^2\) is the squared magnitude of the velocity, and \(H = \frac{c^2}{\gamma - 1} + \frac{q^2}{2}\) is the enthalpy evaluated using interface quantities. This type of dissipation is exact for a linear problem and admits shocks over one interior point (in 1D). However, this superior accuracy comes at the cost of evaluating the characteristic relationships with each calculation of the flux (even when written in an alternative form that avoids some of the matrix multiplications).

7.2.2 Scalar Diffusion

An alternative to matrix diffusion is scalar diffusion. Instead of adding different amounts of dissipation to each characteristic wave, one ensures upwinding and stability by adding dissipation of the form

\[
d_{ij} = \lambda_{\text{max}}(w_j - w_i)
\]

according to the wave speed of the fastest traveling wave, i.e.

\[
\lambda_{\text{max}} = \max\{|u_n|, |u_n + c|, |u_n - c|\}.
\]
As one might expect, the ease to implement scalar diffusion comes at the expense of accuracy. Scalar diffusion cannot admit shocks over one interior point, even in 1D.

### 7.2.3 Upwind Splitting Methods

An intermediate class of methods considers splitting the convective portion of the inviscid flux from the pressure component. The advection upwind splitting method (AUSM)[65] and convective upwind split pressure (CUSP)[66] schemes are two widely used schemes in this class.

In particular, a version of CUSP is well suited to steady inviscid flows because it admits solutions with constant total enthalpy. In this H-CUSP scheme, the diffusive flux is

\[ d_{ij} = \alpha^* c (w_{hj} - w_{hi}) + \beta (f_{nj} - f_{ni}) , \]

where \( c \) denotes the speed of sound at the interface, and \( w_h = (\rho, \rho u, \rho v, \rho H)^T \) are the conservative variables with energy replaced by enthalpy. \( \alpha^* \) and \( \beta \) denote computed coefficients[66] based on the average state in the form of

\[ \alpha^* c = \alpha c - \beta q , \]

where, for ideal gases,

\[
\begin{align*}
\alpha &= |M| \\
\beta &= \begin{cases} 
\max \left(0, \frac{u_n + \lambda^+}{u_n + \lambda^-} \right) & 0 \leq M \leq 1 \\
-\max \left(0, \frac{u_n + \lambda^-}{u_n + \lambda^+} \right) & -1 \leq M \leq 0 \\
\text{sign}(M) & M \geq 1 
\end{cases} \\
\lambda^\pm &= \frac{\gamma + 1}{2\gamma} \pm \sqrt{\left(\frac{\gamma + 1}{2\gamma} u_n\right)^2 + \frac{c^2 - q^2}{\gamma}} ,
\end{align*}
\]

where \( M \) is the local Mach number, \( \lambda \) is an eigenvalue of the flux Jacobian of the altered conservative variables[66], and the rest of the notation follows those in the Roe linearization in section 7.2.1.
7.2.4 Improved Resolution

It is known that the basic formulation of artificial dissipation that yields LED schemes is only first-order accurate. To improve accuracy, one can replace the left and right states by respective reconstructed left and right states at the interface of interest. Reconstruction that maintains stability is another deep subject of research in itself, with various researchers such as van Leer, van Albada, Chakravarthy, etc. having contributed to the literature by designing different limiters[67–70].

Here, reconstruction of the solutions using the symmetric limited positive (SLIP) scheme by Jameson[66] is presented. In the context of CUSP, the diffusive flux becomes

\[ d_{ij} = \alpha^* c(w_{hR} - w_{hL}) + \beta(f_{nR} - f_{nL}) \]

with the left and right states

\[ w_L = w_i + \frac{1}{4}s(\Delta w^+, \Delta w^-)\Delta w^- \]
\[ w_R = w_j - \frac{1}{4}s(\Delta w^+, \Delta w^-)\Delta w^+ , \]

where \( s \) is a limited average operator acting as a slope limiter in the vicinity of discontinuities, and \( \Delta w^+ \) and \( \Delta w^- \) are changes in conservative variables at \( i \) and \( j \) away from the interface. In the unstructured setting, this change is approximated by[25]

\[ \Delta w^+ = (w_j + l_j \cdot \vec{w} w_j) - w_j = w^+ - w_j \]
\[ \Delta w^- = w_i - (w_i + l_i \cdot \vec{w} w_i) = w_i - w^- , \]  \hspace{1cm} (7.3)

where the gradients of \( w \) are computed using the meshless coefficients, and \( l_i \) and \( l_j \) are the vectors from \( \tilde{i} \) and \( \tilde{j} \) to the pseudo points as shown in figure 7.1.

The limited average \( s \) is defined by

\[ s(\Delta w_R, \Delta w_L) = 1 - \left| \frac{\Delta w_R - \Delta w_L}{|\Delta w_R| + |\Delta w_L| + \varepsilon} \right|^r , \]  \hspace{1cm} (7.4)

where \( r \) is a positive number and \( \varepsilon \) is a threshold to avoid division by zero. In equation (7.4), one can see the limiting functionality of \( s \): when \( \Delta w_R \) and \( \Delta w_L \) have opposite
signs, which is the case for the existence of a local extremum, \( s \) evaluates to zero. In that case, the reconstructed states are simply the values at solution points \( i \) and \( j \) on either side of the interface, leading to a locally LED scheme with first order accuracy. In regions containing smooth solutions, \( s \) approaches one. The scheme then recovers the higher-order accuracy from the fully reconstructed states.

### 7.3 Spatial Discretization in Meshless Framework

Given the definitions in previous sections, the flux divergence is computed by

\[
m_i(\delta_f \cdot \mathbf{F})_i = \sum_{j \in s_i} \tilde{f}_{ij}, \tag{7.5}
\]

where \( \tilde{f}_{ij} \) is defined by equation (7.2) with the diffusive flux computed using CUSP with SLIP reconstruction, as mentioned in section 7.2.4. One can show that equation (7.5) is equivalent to

\[
m_i(\delta_f \cdot \mathbf{F})_i = \sum_{k} \sum_{j \in s_i} 2a_{ij} \tilde{F}_{kij}^k, \tag{7.6}
\]

where \( \tilde{F}_{kij}^k \) contains the relevant component of artificial dissipation \( d_{ij} \). Equation (7.6) is in the form of equation (6.1), the generalized form of the meshless scheme. One can see that the generalization is exactly what enables the incorporation of the convective
flux and the stabilizing flux in the same discretization framework.

In the test cases presented in the remainder of this chapter, the limiter parameter in SLIP reconstruction in equation (7.4) is \( r = 3 \).

### 7.3.1 Boundary Conditions

Often given less emphasis than they deserve, boundary conditions play an important role in obtaining good numerical results. In this section, we shall see that the current meshless framework again accommodates a number of different options for enforcing suitable boundary conditions. The boundary conditions for impermeable walls and circulation-corrected inflow/outflow boundaries relevant to the steady-state problems investigated (see section 7.7) are presented here.

**Impermeable Walls**

The exact boundary conditions for fixed, solid walls for the Euler equations is the flow tangency condition \( \vec{u} \cdot \hat{n} = 0 \), where \( \vec{u} \) denotes the velocity vector (as opposed to the unknown in the meshless operator).

Since part of the global point cloud coincides with the domain boundary, one can enforce

\[
\vec{u}_i \cdot \hat{n}_i = 0 , \quad i \in S_B
\]

and correct the pressure at point \( i \) by the normal momentum equation

\[
\frac{\partial p}{\partial n} = -\frac{\rho q}{r} ,
\]

where \( r \) is the radius of curvature of the wall. Another option is of course to include only the pressure flux associate with the boundary face \( \hat{n}_i \) when computing the fluxes.

For the work in this thesis, the boundary conditions were enforced through introducing ghost points to demonstrate the flexibility of the meshless framework. In this treatment, each interior neighbor point \( j \) in the local cloud of a boundary point \( i \) is first reflected about the tangent plane at \( i \) to obtain the ghost point \( g_j \), as shown in figure 7.2. The neighbors of \( i \) that lie on the boundary are also duplicated. When
enforcing the flow tangency conditions, one can simply set the ghost velocity of the reflected ghost points by negating the normal component of the velocity, i.e.

$$\vec{u}_{gj} = (\vec{u}_j \cdot \hat{t}_i)\hat{t}_i - (\vec{u}_j \cdot \hat{n}_i)\hat{n}_i,$$

and copying the velocity of the duplicated boundary ghost points.

This procedure actually allows a boundary point to be treated exactly like an interior point. The meshless coefficients $a_{kgj}^k$ for flux computations at the ghost points are given by

$$\vec{a}_{kgj} = \vec{a}_{ij} - 2(\vec{a}_{ij} \cdot \hat{n}_i)\hat{n}_i.$$

Similar expressions exist for the duplicated boundary points. The virtual volumes $m_i$ at the boundary points are also doubled ($\tilde{m}_i = 2m_i$). One can show that these modifications preserve the consistency of the discretization. In this case, the modified $\tilde{m}_i$’s at the boundary points no longer obviously correspond to some virtual volumes. This highlights that the meshless framework can still differ from its finite volume counterpart.
7.4. TIME DISCRETIZATION

Inflow/Outflow Boundary

At the farfield boundary, characteristic relations can be used to enforce boundary conditions based on waves traveling into and out of the domain. Incoming and outgoing boundary-normal Riemann invariants\cite{71,72} are defined by

\[
R_{\infty i} = \vec{u}_{\infty} \cdot \hat{n}_i - \frac{2c_\infty}{\gamma - 1},
\]

\[
R_{ei} = \vec{u}_e \cdot \hat{n}_i - \frac{2c_e}{\gamma - 1},
\]

where the subscripts “\(e\)” and “\(\infty\)” denote extrapolated quantities and freestream quantities respectively. These two Riemann invariants are averaged to obtain the normal velocity at the ghost point \(\vec{u}_{gj} \cdot \hat{n}_i\).

The tangential velocity depends on the direction of the boundary-normal flow at the boundary:

\[
\vec{u}_{gj} \cdot \hat{\hat{t}_i} = \begin{cases} 
\vec{u}_e \cdot \hat{\hat{t}_i} & \vec{u}_{gj} \cdot \hat{n}_i \geq 0 \\
(\vec{u}_f)_{gj} \cdot \hat{\hat{t}_i} & \vec{u}_{gj} \cdot \hat{n}_i < 0 
\end{cases}
\]

(7.7)

In the above, \(\vec{u}_f\) is the circulation-corrected far-field velocity\cite{72,73} given by

\[
(\vec{u}_f)_{gj} = \vec{u}_\infty + \frac{\Gamma}{r_{gj}(1 - M_\infty^2(\sin \theta_{gj} \cos \alpha - \cos \theta_{gj} \sin \alpha)^2)} \hat{x}_{gj},
\]

where \(M_\infty\) and \(\alpha\) are the free-stream Mach number and angle of attack, respectively, \(\Gamma\) is the circulation around the object of interest, and \(r_{gj}\) and \(\theta_{gj}\) are the polar coordinates of the point \(g_j\) measured from an assumed center of rotation, typically selected to be the quarter-chord points for airfoils.

Obtaining entropy in a way similar to equation (7.7), one can compute all other necessary variables through enforcing constant total enthalpy in the domain.

7.4 Time Discretization

In the current test cases, the Runge-Kutta scheme previously used by Jameson\cite{74} and others was used for advancing solutions towards their steady-state values. Designed
to handle larger time steps than many other Runge-Kutta schemes while maintaining the same level of complexity (number of stages), this scheme treats the convective and diffusive parts of the residual separately.

Let us write the Euler equations in the semi-discrete form

$$\frac{\partial w_i}{\partial t} + R_i = 0 , \quad (7.8)$$

where the residual

$$R_i = \frac{1}{m_i} (Q_i - D_i) , \quad (7.9)$$

is divided into the convective part $Q_i$ and the diffusive part $D_i$. For the Euler equations,

$$Q_i = \sum_{j \in s_i} a_{ij} (f_{nj} + f_{ni}) \quad D_i = \sum_k \sum_{j \in s_i} d_{ij} , \quad (7.10)$$

For a scheme with $q$ stages, we have

$$w^{(n+1,0)} = w^n$$

$$\vdots$$

$$w^{(n+1,s)} = w^n - \alpha_s \Delta t \left( Q^{(s-1)} + D^{(s-1)} \right)$$

$$\vdots$$

$$w^{(n+1)} = w^{(n+1,q)} ,$$

where

$$Q^{(0)} = Q(w^n), \quad D^{(0)} = D(w^n)$$

$$\vdots$$

$$Q^{(s)} = Q(w^{(n+1,s)}), \quad D^{(s)} = \beta_s D(w^{(n+1,s)}) + (1 - \beta_s) D^{(s-1)} ,$$

and $\alpha_s$ and $\beta_s$ are stage coefficients that can be chosen to tune the properties of the scheme. A 5-stage scheme with which Mavriplis[75] and many others have reported
success was used. For this scheme,
\[
\vec{\alpha} = \left(\frac{1}{4}, \frac{1}{6}, \frac{3}{8}, \frac{1}{2}, 1\right) \quad \vec{\beta} = (1, 0, 0.56, 0, 0.44) .
\]

## 7.5 Convergence Acceleration

In steady-state problems, one can often trade temporal accuracy for an increase in convergence speed. Here, we discuss a few convergence acceleration techniques proven successful when applied to the Euler equations.

### 7.5.1 Local Time Stepping

When one uses a uniform time step for the entire domain, the limitation on the time step often results from the spectral radii of the inviscid flux Jacobian associated with the solution points with the smallest virtual volumes. In the meshless scheme, one can approximate the spectral radius at a point \(i\) by
\[
\lambda_i \approx \sum_{j \in s_i} (2\vec{u}_{ij} \cdot \vec{a}_{ij} + \|2\vec{a}_{ij}\|_2 c_{ij}) ,
\]
where \(\vec{u}_{ij}\) and \(c_{ij}\) are the velocity and the speed of sound at the virtual interface respectively. Without the limit of time accuracy, one can naturally replace the global time step
\[
\Delta t = \min_i \left\{ \text{CFL} \frac{m_i}{\lambda_i} \right\}
\]
by the local version
\[
\Delta t_i = \text{CFL} \frac{m_i}{\lambda_i} .
\]

### 7.5.2 Residual Averaging

Empirically, smoothing the residual is a very effective technique for accelerating convergence. After each global iteration, the residual \(R\) at each solution point is implicitly
smoothed by approximately solving

\[ R_i = R_i + \epsilon \nabla^2 R_i \]

using two Jacobi iterations, where the \( \nabla^2 \) represents a combinatorial discrete Laplacian operator, and \( \epsilon \) is a user-input parameter between 0 and 1. Experimentally, one can often double the CFL for time marching when residual smoothing is implemented in the above form.

### 7.5.3 Enthalpy Damping

In enthalpy damping\cite{76}, one makes use of the fact that the total enthalpy \( H_t \) is constant throughout the domain. As long as a numerical scheme admits solutions with constant stagnation enthalpy, one can add source terms in terms of the difference between \( H_t \) and the farfield stagnation enthalpy \( H_{t\infty} \) to accelerate convergence of numerical solutions to ones with constant stagnation enthalpy. The terms \( \alpha \rho(H_t - H_{t\infty}) \), \( \alpha \rho \bar{u}(H_t - H_{t\infty}) \), and \( \alpha(H_t - H_{t\infty}) \) are respectively added to the mass, momentum, and energy equations, where \( \alpha \) is a user-chosen parameter.

### 7.6 Meshless Coefficients Generation

As the conservative meshless framework provides a stable discretization (in the LED sense), the set of meshless coefficients is the last key piece of essential information one would need for solving the selected compressible flow test problems.

The meshless coefficients for airfoil geometries for the test cases were obtained using the coupled approach described in section 4.2. Instead of CVX, Saunder’s PDCO\cite{57} was used. Even for the finest point distributions on the order of 8000 points, it took less than twenty seconds for a workstation with Intel Xeon 5160 processors in a quad-core configuration to obtain meshless coefficients that satisfy the pointwise consistency constraints to order \( 1 \times 10^{-6} \) or better.

Figure 7.3 shows the virtual volumes and face magnitudes of point clouds around the RAE 2822 airfoil used in some of the test cases. One can see that, just like
in section 5.1.2, the meshless coefficients automatically adjusted to the local point spacing. The coefficients for all other geometries showed a similar behavior.

7.7 Compressible Flow Test Cases

This section presents numerical solutions to the Euler equations obtained using the generalized conservative meshless framework from chapter 6 with the stabilization schemes, boundary treatments, time-marching algorithms, and other techniques described earlier in this chapter. The generalized framework allows the conservative meshless framework to be implemented in the same way as finite volume discretizations, namely using point locations and reciprocal coefficients at interfaces (as opposed to some finite volume implementations that compute face normals and other necessary quantities from a set of dual or face vertices and face edges). As a result, a piece of finite volume software was used without any modifications when switching between finite volume and meshless discretizations.

A large portion of the test problems are classic subsonic and transonic benchmark aerodynamics problems listed in AGARD documents [77,78] and investigated by many researchers using various numerical algorithms. It is very natural to validate the current meshless framework using these problems. In particular, Katz [79] and Katz and Jameson [80] used some of the same test problems for comparing a few strong-form-based meshless schemes against a finite volume discretization on structured grids. Besides being compared to their finite volume counterparts, the current results were also compared against those obtained using other meshless algorithms presented in Katz [79].

Theoretically, inviscid isentropic flow around objects has zero drag. A few subsonic problems were first investigated to confirm that the computed drag would indeed converge to zero with increasingly refined point distributions.

The first case features flow over a cylinder at $M = 0.3$. Table 7.1 lists the drag convergence results. Because the cylinder created large disturbances to the incoming uniform flowfield, the drag did not quite reach zero with 128 surface points. However, one can see that the scheme shows at least second-order accuracy even though it is
Figure 7.3: Virtual volumes and face magnitudes around RAE 2822
only formally first-order accurate.

\[
\begin{array}{|c|c|c|}
\hline
n_s & c_d & \text{CM} \\ \hline
32 & 0.1459 & 0.1429 \\ \hline
64 & 0.0273 & 0.0256 \\ \hline
128 & 0.0002 & 0.0048 \\ \hline
\end{array}
\]

Table 7.1: Drag convergence for unit cylinder, \( M = 0.3, \alpha = 0^\circ \)

Table 7.2 lists drag convergence results for the lifting test cases of flow over RAE2822 and KORN airfoil. Since the point distributions were not regular, the table also lists the total number of solution points in the domain. From figure 7.4, one can see the order of convergence lies between two and three, even though the scheme is only formally first-order accurate. This superconvergence demonstrated here is certainly interesting and encouraging, as it does not occur when many other meshless methods are applied to nonlinear problems.

\[
\begin{array}{|c|c|c|}
\hline
n_s & n_p & c_d \text{ CM} & \text{FLO82} \\ \hline
40 & 456 & 0.0056 \\ \hline
80 & 1579 & 0.0013 \\ \hline
160 & 5732 & 0.0002 \\ \hline
\end{array}
\]

Table 7.2: Inviscid drag convergence for subsonic flow over airfoils

Now, we shall see selected results from transonic test cases. Tables 7.3 through 7.6 list the lift and drag coefficients computed with the current scheme along with those computed using finite volume discretizations and meshless methods listed in Katz[79]. In the tables, “CM” represents the current results obtained using our conservative meshless framework. “FV-S” represents structured finite volume results obtained using Jameson’s FLO82. “FV-U” represents unstructured finite volume results obtained using the same software used for computing the conservative meshless results. Other acronyms are consistent with those in Katz[79]: “MV” denotes the
alignment-based “meshless volume scheme” by Katz[26], “TLS” denotes a Taylor-series-based meshless scheme. “PLS” denotes a polynomial-based meshless scheme, similar to the finite point method. “RBF” denotes a meshless method based on radial basis functions.

<table>
<thead>
<tr>
<th>Method</th>
<th>$c_l$</th>
<th>diff. (%)</th>
<th>$c_d$</th>
<th>diff. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV-S</td>
<td>0.3737</td>
<td>-</td>
<td>0.0237</td>
<td>-</td>
</tr>
<tr>
<td>FV-U</td>
<td>0.3639</td>
<td>2.6</td>
<td>0.0221</td>
<td>6.8</td>
</tr>
<tr>
<td>CM</td>
<td>0.3615</td>
<td>3.3</td>
<td>0.0226</td>
<td>4.6</td>
</tr>
<tr>
<td>MV</td>
<td>0.3713</td>
<td>0.6</td>
<td>0.0229</td>
<td>3.4</td>
</tr>
<tr>
<td>TLS</td>
<td>0.3616</td>
<td>3.2</td>
<td>0.0231</td>
<td>2.5</td>
</tr>
<tr>
<td>PLS</td>
<td>0.3684</td>
<td>1.4</td>
<td>0.0242</td>
<td>2.1</td>
</tr>
<tr>
<td>RBF</td>
<td>0.3986</td>
<td>6.7</td>
<td>0.0269</td>
<td>13.5</td>
</tr>
</tbody>
</table>

Table 7.3: Lift and drag coefficients, NACA 0012, $M = 0.8$, $\alpha = 1.25^\circ$.

As one can see, the conservative meshless scheme compares extremely well with both structured and unstructured finite volume methods and other meshless techniques. Figures 7.5 through 7.8 show the surface pressure coefficient plot (overplotted on FLO82 results) and glyph plot of the Mach numbers around the airfoils. The surface pressure coefficients show that shocks are captured sharply. Also, the shock-free results from the KORN airfoil case indicate the high accuracy of the simulations. These results show the excellent potential of the current scheme to be extended for
7.7. COMPRESSIBLE FLOW TEST CASES

<table>
<thead>
<tr>
<th></th>
<th>$c_l$</th>
<th>diff. (%)</th>
<th>$c_d$</th>
<th>diff. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV-S</td>
<td>0.3891</td>
<td>-</td>
<td>0.0582</td>
<td>-</td>
</tr>
<tr>
<td>FV-U</td>
<td>0.3904</td>
<td>0.3</td>
<td>0.0559</td>
<td>4.0</td>
</tr>
<tr>
<td>CM</td>
<td>0.3895</td>
<td>0.1</td>
<td>0.0568</td>
<td>2.4</td>
</tr>
<tr>
<td>MV</td>
<td>0.3923</td>
<td>0.8</td>
<td>0.0572</td>
<td>1.7</td>
</tr>
<tr>
<td>TLS</td>
<td>0.3830</td>
<td>1.6</td>
<td>0.0565</td>
<td>2.9</td>
</tr>
<tr>
<td>PLS</td>
<td>0.3883</td>
<td>0.2</td>
<td>0.0593</td>
<td>1.9</td>
</tr>
<tr>
<td>RBF</td>
<td>0.3343</td>
<td>14.1</td>
<td>0.0570</td>
<td>2.1</td>
</tr>
</tbody>
</table>

Table 7.4: Lift and drag coefficients, NACA 0012, $M = 0.85$, $\alpha = 1.0^{\circ}$.

<table>
<thead>
<tr>
<th></th>
<th>$c_l$</th>
<th>diff. (%)</th>
<th>$c_d$</th>
<th>diff. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV-S</td>
<td>1.1481</td>
<td>-</td>
<td>0.0486</td>
<td>-</td>
</tr>
<tr>
<td>FV-U</td>
<td>1.1552</td>
<td>0.6</td>
<td>0.0475</td>
<td>2.3</td>
</tr>
<tr>
<td>CM</td>
<td>1.1589</td>
<td>0.9</td>
<td>0.0479</td>
<td>1.4</td>
</tr>
<tr>
<td>MV</td>
<td>1.1338</td>
<td>1.2</td>
<td>0.0474</td>
<td>2.5</td>
</tr>
<tr>
<td>TLS</td>
<td>1.1319</td>
<td>1.4</td>
<td>0.0481</td>
<td>1.0</td>
</tr>
<tr>
<td>PLS</td>
<td>1.1347</td>
<td>1.2</td>
<td>0.0491</td>
<td>1.0</td>
</tr>
<tr>
<td>RBF</td>
<td>1.1744</td>
<td>2.3</td>
<td>0.0545</td>
<td>12.1</td>
</tr>
</tbody>
</table>

Table 7.5: Lift and drag coefficients, RAE 2822, $M = 0.75$, $\alpha = 3.0^{\circ}$.

<table>
<thead>
<tr>
<th></th>
<th>$c_l$</th>
<th>diff. (%)</th>
<th>$c_d$</th>
<th>diff. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV</td>
<td>0.6308</td>
<td>-</td>
<td>0.0000</td>
<td>-</td>
</tr>
<tr>
<td>FV-U</td>
<td>0.6379</td>
<td>1.1</td>
<td>0.0002</td>
<td>-</td>
</tr>
<tr>
<td>CM</td>
<td>0.6372</td>
<td>1.0</td>
<td>0.0003</td>
<td>-</td>
</tr>
<tr>
<td>MV</td>
<td>0.6353</td>
<td>0.9</td>
<td>0.0000</td>
<td>-</td>
</tr>
<tr>
<td>TLS</td>
<td>0.6178</td>
<td>2.1</td>
<td>0.0009</td>
<td>-</td>
</tr>
<tr>
<td>PLS</td>
<td>0.6164</td>
<td>2.3</td>
<td>0.0011</td>
<td>-</td>
</tr>
<tr>
<td>RBF</td>
<td>0.6200</td>
<td>1.7</td>
<td>0.0010</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.6: Lift and drag coefficients, KORN airfoil, $M = 0.75$, $\alpha = 0.0^{\circ}$.

wider applications.
Figure 7.5: Flow over NACA 0012, $M = 0.80$, $\alpha = 1.25^\circ$
Figure 7.6: Flow over NACA 0012, $M = 0.85$, $\alpha = 1.0^\circ$
Figure 7.7: Flow over RAE 2822, $M = 0.75$, $\alpha = 3.0^\circ$
7.7. COMPRESSIBLE FLOW TEST CASES

Figure 7.8: Flow over KORN airfoil, $M = 0.75$, $\alpha = 0.0^\circ$
Chapter 8

Conclusion and Future Work

In the introduction, it was mentioned that the work in this thesis had been aimed at developing a new meshless method that would achieve discrete conservation, a challenge that had eluded researchers for over two decades. It is very exciting that the conservative meshless scheme in this thesis represents the first documented meshless scheme in the literature that lives up to that challenge.

In addition to discrete conservation, discrete scalar and energy conservation properties were proven. The convergence of the scheme was also demonstrated in a series of model problems, in which superconvergence was observed, just like when popular meshed-based discretizations were used. In addition, the coefficients for the meshless scheme can be generated by the solution of a convex optimization problem. This not only means that the coefficients can be obtained efficiently, but also allows the coefficient generation procedure to benefit from continuous improvements in the field of convex optimization itself.

A generalized framework that both preserves numerical conservation and allows incorporation of flux schemes originally designed for meshed discretizations, without sacrificing any efficiency, was also introduced using the reciprocity of the meshless coefficients. The success of this philosophy of harnessing existing numerical schemes was shown by the excellent results obtained by solving the Euler equations for compressible flow using this framework with stabilizing artificial dissipation originally designed for finite volume discretizations.
Thus, it was demonstrated that the work in this thesis not only addressed the lack of conservation suffered by meshless schemes, but did so with a viable alternative that is as general and efficient as traditional meshed algorithms. The potential to apply the new framework to a wide variety of problems is particularly promising and exciting.

For meshless algorithms to completely replace mesh generation as an efficient, or even standard, pre-processing procedure for computational analysis, they must adequately address the challenges of point generation and connectivity selection. This thesis did not address these issues as it develops the meshless framework with a given set of points and connectivity. Fortunately, although not included in this thesis, ongoing work in these two key areas has already yielded encouraging results. It remains hopeful that a meshless framework can further improve and achieve the level of robustness for routine analysis.

Improvements in areas other than point generation are also being identified and pursued. As a first step towards handling problems of higher complexity, extensions of the current method to three dimensions is being implemented. For problems of larger scale, distributed optimization and domain decomposition techniques are being investigated as potential methods to facilitate efficient generation of the meshless coefficients.

All in all, the work in this thesis serves as a solid foundation for tackling complex, real-world problems using a conservative meshless algorithm. The extensions mentioned above should open doors to new opportunities for fast and robust numerical analysis in computational mechanics for practical systems in engineering.
Appendix A

Equivalence of Matrix Constraints

It was mentioned in chapter 3 that the consistency constraints can be expressed in various different equivalent forms. Here, the matrix operations that allow one to obtain one form of the constraints from another will be shown.

A.1 Expression of Matrix Constraints in Terms of Incidence Matrices

One can actually express the matrix $\Phi_{opp}$ in terms of the connectivity graph incidence matrices mentioned in section 3.1. First, notice

$$g_j = g_i + (g_j - g_i)$$
$$-g_i = -g_j + (g_j - g_i)$$

for any function $g$. In matrix form, this translates to

$$G_{opp} = G_pB + U\Delta_{G_f} ,$$  \hspace{1cm} (A.1)
where $\Delta_{G_f}$ is the $n_f \times n_f$ diagonal matrix with terms $g_j - g_i$. For instance, in the 1D example 3.2.1,

$$X_{opp} = \begin{bmatrix} x_2 & 0 & 0 & x_3 & 0 \\ -x_1 & x_3 & 0 & 0 & -x_4 \\ 0 & -x_2 & x_4 & -x_1 & 0 \\ 0 & 0 & -x_3 & 0 & x_2 \end{bmatrix} = \begin{bmatrix} x_1 + (x_2 - x_1) & 0 & 0 & x_1 + x_{31} & 0 \\ -x_2 + (x_2 - x_1) & x_2 + x_{32} & 0 & 0 & -x_2 + x_{24} \\ 0 & -x_3 + x_{32} & x_3 + x_{43} & -x_3 + x_{31} & 0 \\ 0 & 0 & -x_4 + x_{43} & 0 & x_4 + x_{24} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} (x_2 - x_1) \\ (x_3 - x_2) \\ (x_4 - x_3) \\ (x_3 - x_1) \end{bmatrix} = X_pB + U\Delta_{X_f},$$

where $x_{ij} = x_j - x_i$. Thus, one can again rewrite the system of constraints for this example as

$$\begin{bmatrix} I_p & B & 0_p \\ X_p & X_pB + U\Delta_{X_f} & -I_p \\ X_p^2 & X_p^2B + U\Delta_{X^2_f} & -2X_p^2 \end{bmatrix} \begin{bmatrix} a_p \\ a_f \\ m \end{bmatrix} = 0.$$
In general, the matrix $\Phi_{opp}$ can now be written as

$$\Phi_{opp} = \Phi_p B + \begin{bmatrix} U\Delta\Phi_{1f} \\ U\Delta\Phi_{2f} \\ \vdots \\ U\Delta\Phi_{n\phi f} \end{bmatrix},$$

(A.3)

which means

$$\begin{bmatrix} \Phi_p & \Phi_{opp} \end{bmatrix} = \begin{bmatrix} \Phi_p I_p & \Phi_p B \\ 0_p & U\Delta\Phi_{1f} \\ 0_p & U\Delta\Phi_{2f} \\ \vdots & \vdots \\ 0_p & U\Delta\Phi_{n\phi f} \end{bmatrix} + \begin{bmatrix} \Phi_p I_p & \Phi_p B \\ 0_p & U\Delta\Phi_{1f} \\ 0_p & U\Delta\Phi_{2f} \\ \vdots & \vdots \\ 0_p & U\Delta\Phi_{n\phi f} \end{bmatrix}.$$

(A.4)

### A.2 From “Opposite” Form to Delta Form

Once the original constraints are expressed in terms of the incidence matrices, it becomes very obvious that the delta form can be obtained from the original form through simple matrix operations:

$$\begin{bmatrix} \Phi_p & \Phi_f & D^k \end{bmatrix} = \begin{bmatrix} I_p & B & 0_p \\ 0_p & U\Delta\Phi_{2f} & (\partial^k\Phi_2)_p \\ \vdots & \vdots & \vdots \\ 0_p & U\Delta\Phi_{n\phi f} & (\partial^k\Phi_{n\phi})_f \end{bmatrix} = \begin{bmatrix} I_p & B & 0_p \\ (\Phi_2)_p & (\Phi_2)_p B + U\Delta\Phi_{2f} & -(\partial^k\Phi_2)_p \\ \vdots & \vdots & \vdots \\ (\Phi_{n\phi})_p & (\Phi_{n\phi})_p B + U\Delta\Phi_{n\phi f} & -(\partial^k\Phi_{n\phi})_f \end{bmatrix} - \begin{bmatrix} \Phi_p I_p & \Phi_p B & 0_p \end{bmatrix}$$

$$= \begin{bmatrix} I_p & -\Phi_2 - I_p \\ \vdots & \vdots \\ -\Phi_{n\phi} - I_p \end{bmatrix} \begin{bmatrix} I_p & B & 0_p \\ (\Phi_2)_p & (\Phi_2)_p B + U\Delta\Phi_{2f} & -(\partial^k\Phi_2)_p \\ \vdots & \vdots & \vdots \\ (\Phi_{n\phi})_p & (\Phi_{n\phi})_p B + U\Delta\Phi_{n\phi f} & -(\partial^k\Phi_{n\phi})_f \end{bmatrix}$$
\[
\begin{bmatrix}
I_p \\
-(\Phi_2)_p I_p \\
\vdots \\
-(\Phi_{n_0})_p I_p \\
\end{bmatrix} = 
\begin{bmatrix}
\Phi_p & \Phi_{opp} & D^k \\
\end{bmatrix}.
\]

In other words, the delta form matrix constraints are obtained by subtracting from the original constraints the constant consistency condition weighed by the value of the basis polynomial at each solution point, exactly the same procedure presented in algebraic form in section 3.3.1.

### A.3 From Delta Form to Point-centered Form

Similarly, the delta form constraints can be manipulated to obtain the point-centered form. The algebra is more complicated, so the simple 1D example 3.2.1 will again be used for illustration.

Recall that the system of constraints in example 3.2.1 (1D, \(L = 2\)) is

\[
\begin{bmatrix}
I_p \\
X_p \\
X_p^2 \\
\end{bmatrix} 
\begin{bmatrix}
B \\
X_p B + U\Delta_X f \\
X_p^2 B + U\Delta_{X^2} f \\
\end{bmatrix} 
\begin{bmatrix}
0_p \\
-I_p \\
-2X_p^2 \\
\end{bmatrix} 
\begin{bmatrix}
a_p \\
a_f \\
m \\
\end{bmatrix} = \mathbf{0},
\]

which, in delta form, becomes

\[
\begin{bmatrix}
\Phi_p & \Phi_f & D \\
\end{bmatrix} 
\begin{bmatrix}
a_p \\
a_f \\
m \\
\end{bmatrix} = 
\begin{bmatrix}
I_p & B & 0_p \\
0_p & U\Delta_X f & -I_p \\
0_p & U\Delta_{X^2} f & -2X_p^2 \\
\end{bmatrix} 
\begin{bmatrix}
a_p \\
a_f \\
m \\
\end{bmatrix} = \mathbf{0}.
\]

Now, the point-centered form of constraints for this example is

\[
\begin{bmatrix}
\Phi_p & \Phi_f & D \\
\end{bmatrix} 
\begin{bmatrix}
a_p \\
a_f \\
m \\
\end{bmatrix} = 
\begin{bmatrix}
I_p & B & 0_p \\
0_p & U(\Delta_X) f & I_p \\
0_p & B(\Delta_X)^2 & 0_p \\
\end{bmatrix} 
\begin{bmatrix}
a_p \\
a_f \\
m \\
\end{bmatrix} = \mathbf{0}.
\]
By expanding \((x_j - x_i)^2\) into
\[
(x_j - x_i)^2 = x_i^2 - 2x_i x_j + x_j^2
= 2x_i^2 + (x_j^2 - x_i^2) - 2x_i x_j,
\]
one can write
\[
B(\Delta_X)^2 f = 2X_p^2 B + U(\Delta_{(X^2)}) f - 2X_p X_{opp}
= 2X_p^2 B + U(\Delta_{(X^2)}) f - 2X_p (X_p B + U(\Delta_X) f)
= U(\Delta_{(X^2)}) f - 2X_p U(\Delta_X) f.
\]
Therefore,
\[
\tilde{\Phi}_f = \begin{bmatrix}
    B \\
    U(\Delta_X)_f \\
    B(\Delta_X)_f^2
\end{bmatrix}
= \begin{bmatrix}
    I_p \\
    I_p \\
    -2X_p & I_p
\end{bmatrix}
\begin{bmatrix}
    B \\
    U(\Delta_X)_f \\
    U(\Delta_{X^2})_f
\end{bmatrix}
= P \tilde{\Phi}_f,
\]
i.e. rows of \(\tilde{\Phi}_f\) can be expressed as linear combinations of rows of \(\tilde{\Phi}_f\) from the delta form. The matrix
\[
P = \begin{bmatrix}
    I_p \\
    I_p \\
    -2X_p & I_p
\end{bmatrix}
\]
represents the contribution of rows of \(\tilde{\Phi}_f\) in the linear combination. Because of the
linearity of the derivative operator,

$$\tilde{D} = \begin{bmatrix} 0_p \\ -I_p \\ 0_p \end{bmatrix} = \begin{bmatrix} I_p \\ I_p \\ -2X_p \end{bmatrix} \begin{bmatrix} 0_p \\ -I_p \\ -2X_p \end{bmatrix} = PD .$$

Thus, the point-centered constraint matrix for this 1D example can be expressed as

$$\begin{bmatrix} \tilde{\Phi}_p & \tilde{\Phi}_f \end{bmatrix} \tilde{D} = \begin{bmatrix} I_p & B & 0_p \\ 0_p & U(\Delta X)_f & -I_p \\ 0_p & B(\Delta X)^2_f & 0_p \end{bmatrix} = \begin{bmatrix} I_p & B & 0_p \\ 0_p & U(\Delta X)_f & -I_p \\ 0_p & U(\Delta X^2)_f & -2X_p \end{bmatrix} = P \begin{bmatrix} \tilde{\Phi}_p & \tilde{\Phi}_f & \tilde{D} \end{bmatrix} ,$$

demonstrating the equivalence of the point-centered constraints and the other two forms of constraints. The exact same procedure can be applied to show the equivalence of constraints in examples involving higher spatial dimensions and polynomial orders.
Appendix B

Existing Schemes Satisfying C-1 and C-2

To further motivate that the current meshless scheme is a natural mesh-free analog of conservative finite volume schemes, an example of an existing finite volume scheme satisfying C-1 and C-2 with $L = 1$ is presented here.

Consider a nodal finite volume scheme on the median dual of a regular triangular mesh, a representative portion of which is denoted in figure B.1. In each triangle,

\[
S \frac{\partial u}{\partial x} = \int u \, dy \simeq \sum_k u_j \Delta y_j .
\]  

Figure B.1: Section of regular triangular mesh for nodal finite volume scheme with primal nodes (filled) and median-dual nodes (hollow)

Gauss theorem gives

\[
S \frac{\partial u}{\partial x} = \int u \, dy \simeq \sum_k u_j \Delta y_j .
\]  

(B.1)
This result is exact for polynomials of order one. For a conservation law

\[
\frac{\partial \phi}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0 ,
\]

the following approximation of the integral form

\[
\left( \sum_j S \right) \frac{\partial \phi}{\partial t} \bigg|_0 = -\frac{1}{2} \sum_k \{(f_{j+1} + f_j)(y_{j+1} - y_j) - (g_{j+1} + g_j)(x_{j+1} - x_j)\}
\]

\[
= -\frac{1}{2} \sum_k \{(f_j)(y_{j+1} - y_{j-1}) - (g_j)(x_{j+1} - x_{j-1})\}
\]

is also exact for a polynomial of order one with \( f = c_1 \phi, \ g = c_2 \phi \), i.e. if the governing equations are homogeneous of order one. This is indeed the case for several conservation laws, including the advection equation and Euler Equations for inviscid, compressible flows. Jameson and Mavripilis [81] have previously successfully obtained numerical solutions to the Euler Equations using this scheme.

However, it also turns out that some commonly-used schemes do not satisfy C-1 and C-2 with \( L \leq 1 \). Without diving into details, we point out that finite difference or finite volume schemes constructed in a fashion similar to the one above does not satisfy C-2 for \( L = 1 \) when used on a structured, but non-uniform, quadrilateral mesh.
Bibliography


[59] OBOE, https://projects.coin-or.org/OBOE.


