A Model Integrated Meshless Solver (mims) For Fluid Flow And Heat Transfer

Salvadore Gerace
University of Central Florida

A MODEL INTEGRATED MESHLESS SOLVER (MIMS) FOR FLUID FLOW AND HEAT TRANSFER

by

SALVADORE A. GERACE
B.S. University of Central Florida, 2006
M.S. University of Central Florida, 2007

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Mechanical, Materials, and Aerospace Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2010

Major Professor: Alain J. Kassab

ABSTRACT

Numerical methods for solving partial differential equations are commonplace in the engineering community and their popularity can be attributed to the rapid performance improvement of modern workstations and desktop computers. The ubiquity of computer technology has allowed all areas of engineering to have access to detailed thermal, stress, and fluid flow analysis packages capable of performing complex studies of current and future designs. The rapid pace of computer development, however, has begun to outstrip efforts to reduce analysis overhead. As such, most commercially available software packages are now limited by the human effort required to prepare, develop, and initialize the necessary computational models. Primarily due to the mesh-based analysis methods utilized in these software packages, the dependence on model preparation greatly limits the accessibility of these analysis tools. In response, the so-called meshless or mesh-free methods have seen considerable interest as they promise to greatly reduce the necessary human interaction during model setup. However, despite the success of these methods in areas demanding high degrees of model adaptability (such as crack growth, multi-phase flow, and solid friction), meshless methods have yet to gain notoriety as a viable alternative to more traditional solution approaches in general solution domains. Although this may be due (at least in part) to the relative youth of the techniques, another potential cause is the lack of focus on developing robust methodologies. The failure to approach development from a practical perspective has prevented researchers from obtaining commercially relevant meshless methodologies which reach the full potential of the approach.

The primary goal of this research is to present a novel meshless approach called MIMS (Model Integrated Meshless Solver) which establishes the method as a generalized solution technique capable of competing with more traditional PDE methodologies (such as the finite element and finite volume methods). This was accomplished by developing a robust meshless technique as well as a comprehensive model generation procedure. By closely integrating the model generation process into the overall solution methodology, the presented techniques are able to fully exploit the strengths of the meshless approach to achieve levels of automation, stability, and accuracy currently unseen in the area of engineering analysis. Specifically, MIMS implements a blended meshless solution approach which utilizes a variety of shape functions to obtain a stable and accurate iteration process. This solution approach is then integrated with a newly developed, highly adaptive model generation process which employs a quaternary triangular surface discretization for the boundary, a binary-subdivision discretization for the interior, and a unique shadow layer discretization for near-boundary regions. Together, these discretization techniques are able to achieve directionally independent, automatic refinement of the underlying model, allowing the method to generate accurate solutions without need for intermediate human involvement. In addition, by coupling the model generation with the solution process, the presented method is able to address the issue of ill-constructed geometric input (small features, poorly formed faces, etc.) to provide an intuitive, yet powerful approach to solving modern engineering analysis problems.

To my wife Liz, without whom I would be lost.

ACKNOWLEDGMENTS

I would like to provide special thanks to...

Alain Kassab and Eduardo Divo for always having time to answer my questions and for showing the utmost confidence in my abilities.

&

Kevin Erhart for providing support and encouragement throughout the entire research effort and for putting up with my never ending requests for comments and criticisms throughout the writing process.

&

My family for never letting me give up and for helping me get to where I am today.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

When attempting to solve a set of partial differential equations, as is commonly encountered in engineering disciplines, an engineer will typically approach the problem from either an analytical or numerical perspective. Despite analytical techniques being capable of providing exact solutions to the differential equations, their reliance on coordinate transformations and variable substitution makes them ill-suited for real world problems with complex geometries. In addition, many of the more complex differential equations have no known closed form solution, even for the simplest of geometries. As such, it is common practice in engineering to solve the underlying equations via numerical techniques whereby the solution domain is discretized and the governing equations and boundary conditions are satisfied (to some level of accuracy) over each discrete region of the domain. Although there are many numerical techniques capable of solving engineering partial differential equations, the most commonly used rely on a structured connectivity between nodes and are generally classified as mesh-based techniques (for their use of a structured connectivity mesh). Finite difference, finite element, and finite volume methods may all be broadly categorized as mesh-based techniques as they all require some sort of defined connectivity (or arrangement) of the underlying computational nodes. Although mesh-based methods have proven themselves capable for a wide range of problem domains, there is still the undesirable responsibility of having to define a connectivity within the solution interior. Despite efforts to automate the mesh generation process, a considerable amount of time and human effort is still spent preparing and meshing the computational model when presented with a problem consisting of complex geometry.

In an attempt to eliminate the need for the underlying nodal connectivity, some researchers have turned to the area of meshless and mesh-reduction methods. These methods, which seek to replace the underlying structured connectivity with an unstructured interpolation scheme, have shown considerable promise in many application areas. However, they have failed, as of yet, to provide competition to more conventional mesh-based approaches (such as finite element and finite volume methods) when applied to real-world, industrially relevant applications. This may largely be attributed to the relative youth of the field (finite element and finite volume methods were both developed in the 1960s, while modern meshless methods have only been around since the 1980s), however, it may also be caused by a lack of proper focus. Specifically, many researchers focus on generating new meshless techniques, while failing to address the underlying cause for concern in model discretization. This failure to address the underlying issues of mesh generation has resulted in a general lack of practicality of the field and has largely relegated meshless methods to academic endeavors and specialized application domains.

Despite the inability of meshless methods to thus far compete against more traditional mesh-based techniques, there has been considerable advances within the field. Modern meshless implementations are generally at least as efficient (in terms of memory and computational expense) as unstructured mesh-based techniques and therefore have reached the solution potential of more traditional approaches. It is for this reason that the focus of this research effort will be presenting a novel meshless methodology and solution approach which builds upon current meshless research to obtain an industrially relevant process. At the core of this research is a tailored, collocation-based meshless method which has been developed to be both robust as well as accurate under a variety of nodal configurations. This is accomplished through the use of a unique blend of meshless shape functions, as well as a novel generalized finite difference scheme which directly integrates into the meshless solution process. Another key innovation allowing the method to reach its overall goals is the development of a point distribution process specifically designed to take advantage of the liberties granted by the developed meshless method. By tightly integrating the model generation and solution process, the method is able to specifically address the underlying issues which make mesh generation such a time consuming and tedious process. In addition, this integration allows for a highly adaptive and robust system capable of efficiently evolving both the solution, as well as the underlying discretization with no human interaction.

In order to appreciate the innovation and contributions of this research, Chapter 2 will begin by presenting a general overview of the history of meshless methods. This historical presentation will focus on major developments which contributed to the growth and popularity of meshless methods within the engineering community. In addition, it will present the leading formulation techniques in an attempt to demonstrate the wide variety of solution methodologies within the field. Following a presentation of the history of meshless methods, Chapter 3 will describe the specific meshless methodology that will be employed within the context of this research effort. The purpose of describing the methodology is to present a rationale for meshless methods (as compared to more traditional mesh-based techniques) and to address common misconceptions regarding mesh-based versus meshless techniques. Chapter 3 will also provide a general outline for the specific implementation details of the developed method and provide justification for each component within the context of the general solution technique. Once the meshless methodology has been described, Chapter 4 will present the various techniques for generating the necessary meshless shape functions for solving engineering problems. As is the case with most solution techniques utilizing collocation, the process by which the necessary shape function are constructed largely defines the effectiveness of the approach. In this respect, Chapter 4 will describe the core meshless method processes utilized within this method. Following presentation of the various shape construction techniques, Chapter 5 will provide specific meshless implementation details that are often overlooked in academic research papers. The topics addressed in this chapter will provide several solutions specifically related to robust meshless implementations, and as such, represent a key contribution of this effort. The second component of the method is the model generation procedures, which will be described in Chapter 6. As the model generation process has been designed to be most appropriate for meshless point distributions, a thorough presentation of both the rationale as well as implementation details will be provided. Following this, a brief discussion of the adaptive refinement capabilities of the method will be discussed in Chapter 7. Once the entire meshless procedure has been described, Chapter 8 will present additional details pertinent to specific application domains (governing equations) and provide the remaining components necessary to complete the current work. Chapter 9 will then follow by providing simple verification tests which demonstrate the fundamental accuracy of the described techniques. Following the verification examples, Chapter 10 will present a selection of case studies, representing a wide range of problem characteristics, in order to demonstrate the capabilities of the meshless method. Unlike Chapter 9, which focuses primarily on accuracy, Chapter 10 will focus on illustrating the advantages of the described meshless implementation with respect to currently available engineering analysis software. Finally, Chapter 11 will provide a summary of the research effort, highlighting the overall contributions to the engineering community and providing conclusions that can be made through this research.

CHAPTER 2
HISTORY OF MESHLESS METHODS

In general, the term *meshless method* may refer to any technique used to solve partial differential equations whereby little or no underlying connectivity is required between solution nodes. Unfortunately, this broad definition encompasses many different solution techniques, oftentimes with little in common other than the fact that they do not require an underlying nodal connectivity. In general, however, most meshless methods may be classified into three board categories: spectral methods, local influence methods, and radial basis function methods. This chapter will serve as a brief history of meshless method development and will illustrate the progression from early techniques to the modern methods used as a foundation for this research. In addition, it will demonstrate the connections between the various methods and classify their development in an attempt to serve as a categorical review of the existing body of knowledge.

2.1 Spectral Methods

Spectral methods represent a class of solution techniques by which orthogonal functions such as Legendre or Chebyshev polynomials are globally interpolated at a set of structured points in order to solve boundary-value problems for partial differential equations. Primarily contributed to the work of Orszag [1] and Kreiss and Oliger [2], the use of globally interpolating functions allows for highly accurate results with a minimal number of solution nodes, a property often referred to as *spectral convergence* [3, 4]. In fact, spectral methods have demonstrated exponential convergence with respect to the nodal spacing [5], a very desirable trait when attempting to solve large scale problems. Although historically based on Fourier transform methods, spectral methods can theoretically utilize any set of orthogonal functions to globally represent the solution over the entire domain. Unfortunately, the technique that gives the method its spectral convergence (global interpolation) also presents problems when the solution may not be represented as a globally smooth function (as is the case when discontinuities are present) or when the problem domain becomes complex (which oftentimes presents issues with over-fitting due to the required number of nodes) [6]. To address these issues, most recent efforts in spectral methods have departed from the classical idea of global interpolation in favor of a more practical multidomain approach whereby small subdomains are constructed to reduce the number of degrees of freedom for a particular problem [7, 8]. By breaking the problem into smaller regions, this approach is able to address the issues of attempting to represent the field globally while at the same time maintaining the spectral convergence property (albeit over each subregion, as opposed to the entire domain). Patera's work on the *spectral element method* (SEM) [9], a variational formulation similar to finite elements, led to considerable advancements in the general applicability of spectral methods to complex geometries [10, 11, 12]. Most recent efforts in the field of "classical" spectral elements range somewhere between minimal multidomain spectral methods, where the minimum number of subdomains are utilized to solve the problem, and the more discrete spectral element methods, where the entire domain is subdivided in much the same way as finite elements.

Despite exhibiting very desirable properties, “classical” spectral methods remained dependent on a structured grid, dictated by the underlying orthogonal function approximations. In an attempt to eliminate this dependency, the so-called *pseudo-spectral methods* were developed [4, 13] whereby the same Legendre or Chebyshev polynomials used in “classical” spectral methods were collocated at data centers, instead of using a more traditional method of moments, least squares, or Galerkin formulation. The use of collocation produced a method which eliminated much of the need for underlying connectivity, and allowed for pseudo-spectral methods to gain recognition as a meshless method. The psuedo-spectral methods enjoyed success as the technique matured, primarily in the areas of incompressible fluid flow [14, 15, 16] and wave propagation [17, 18, 19] where the underlying field solutions are smooth and continuous.

2.2 Local Influence Methods

The second subset of meshless development may be broadly categorized as local influence methods, or methods which define nodal influence on a local scale, as opposed to the global scale used in spectral methods; *smooth particle hydrodynamics* (SPH) [20, 21] is generally acknowledged as the first of these methods. By representing the underlying fields as a set of N small volumes (particles), SPH provides many advantages over alternative techniques when modeling phenomenon with very large geometric deformations. Though originally applied to astrophysics [22, 23] (which has historically proven difficult to model due to the diffuse nature of astrological bodies such as stars and dust clouds), phenomena such as free surface flow [24, 25] and dynamic response of elasto-plastic materials (including fracture) [26, 27, 28] are also ideal candidates for this method. Despite being one of the first techniques to bear the meshless moniker, SPH has historically had issues demonstrating acceptable accuracy [29] unless a large number of point masses are utilized [30]. Regardless of accuracy concerns, SPH has been successfully applied to many solution domains including multi-phase flows [31, 32], geophysical flows [33, 34], and solid friction [35], though special care must be taken to ensure accurate results [36, 37, 38].

Inspired in part by the success of SPH, Nayroles et al. developed the *diffuse element method* (DEM) [39] as an alternative local influence meshless method. Based on finite element inspired Galerkin formulations, DEM utilized moving least squares (MLS) [40, 41] approximations to produce an "element" capable of evaluating field derivatives without any underlying structure (by utilizing local influence regions). However, despite being a major stepping stone in meshless methods due to its use of MLS approximations, direct DEM implementations have been limited primarily to specialized problem domains [42] and coupled implementations with traditional finite element methods [43]. The lack of continued research in this area is partially due to the generalization of the method by Belytschko and colleagues [29, 44] into the *element free Galerkin* (EFG) method. Despite being more expensive than SPH, EFG proved to be more adept at solving complex problems such as crack growth [45] that had historically presented difficulties to finite element methods due to the necessary problem geometry. In addition, through the work of Duarte and Oden [46, 47] and Melenk and Babuska [48], the EFG method and its related generalizations (such as *Hp-Cloud methods* and *Partition of Unity Finite Element methods*) were proven to have good convergence qualities over a wide range of problem domains. The work done within the Partition of Unity paradigm [49] (which includes extended finite element methods and Partition of Unity Finite

Element methods), established a solid mathematical foundation for this class of meshless methods. In addition, by formally including discontinuities into the underlying fields [50], methods based on the Partition of Unity paradigm have been able to solve many classically "difficult" problems for traditional finite elements including crack growth [51] and hole and inclusion formation [49].

The third group of techniques that fall under the classification of local influence methods are works based on the notion of *generalized finite differencing*, originally proposed by Liszka and Orkisz [52, 53]. The basic concept of generalized finite differencing is to approximate the underlying field at a set of data nodes using a smooth function (in many cases these are monomial basis functions, resulting in a formulation similar to MLS), and to obtain derivatives by applying finite differencing equations over this smoothed approximating field [54, 55, 56]. One of the major benefits of generalized finite differencing is that it can draw upon the wealth of existing knowledge in the area of traditional finite differencing (such as boundary condition application, upwinding, stability concerns, etc.) and, as such, are usually much less complex than alternative meshless methods based on Galerkin formulations [57]. Another major benefit of these methods is that traditional finite differencing formulations may be directly applied to areas of local structure within the domain, resulting in significantly reduced computational effort as both preprocessing (constructing the underlying smooth approximating function) and iteration times are decreased (due to the smaller stencils required by traditional finite differencing formulations) [58]. Unfortunately, because finite differencing is such a ubiquitous technique, many works utilizing generalized finite differencing are not classified as meshless methods making it difficult to correctly identify all derivative works within this field. Regardless, there have been several notable applications of this method within the areas of viscous flow [58, 59], metal-forming [60], and optics [61, 62, 63].

2.3 Radial Basis Function Methods

Radial basis function (RBF) methods are the third branch of meshless techniques and are largely contributed to the work of Kansa in the early 1990s [64, 65]. At the core of these methods is a class of interpolating basis functions referred to as radial basis functions which are radially symmetric about a point of influence and exhibit specific properties with regards to smoothness and continuity [66]. Perhaps most notable of the RBF interpolators is the Hardy Multiquadric [67], which was originally developed in the early 1970s as a means of reproducing irregular topographic surfaces from sparse data sets. Since then, RBF interpolations have been applied to many different interpolation tasks with considerable success [68, 69, 70, 71].

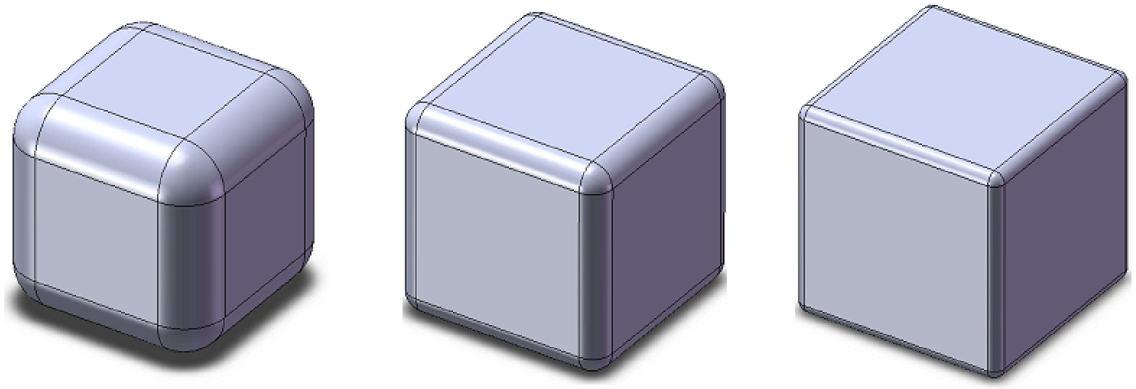
Within the area of meshless methods, Kansa utilized RBF interpolations to represent the underlying solution field such that partial derivatives could be obtained at any arbitrary point within the domain. By collocating the governing equations at a specified number of nodes, the RBF interpolators could be directly differentiated and unknown field quantities could be determined through an explicit or implicit updating scheme. Although early implementations of the RBF collocation method utilized global interpolations [72, 73], it was soon clear that the RBF interpolations exhibited troublesome ill-conditioning problems when applied over large data sets [74]. These issues seemed to stem from the so-called "uncertainty relation" formally described by Schaback [75]. Stevens et al. [76] summarized the uncertainty relation succinctly as suggesting that for RBF interpolations “better [moment matrix] conditioning is actually associated with worse accuracy, and worse [moment matrix] conditioning is associated with improved accuracy”. This result is somewhat counter-intuitive and has resulted in many techniques which attempt to address the uncertainty relation [77, 78]. Most notably of these techniques is to apply domain decomposition in order to subdivide the problem into more manageable solution domains [79, 80, 81, 82]. Although these domain decomposition methods are successful at mitigating the problems identified by the uncertainty relation, they introduce additional concerns such as how to handle the introduced domain interfaces and how to automate the process of decomposing the domain. Regardless of this fact, global RBF collocation methods have been successfully applied to many problem domains including convection-diffusion problems [83], large scale heat transfer [84], and thermoelasticity [85].

Despite the successes of global RBF collocation methods, there was a growing concern that the uncertainty relation would prevent the method from achieving widespread use as problem sizes increased. In response to this, three research groups independently developed varying implementations of local RBF collocation techniques [86, 87, 88]. These techniques forgo global interpolation in favor of small, compact, overlapping local influence regions (commonly referred to as *support domains* or *topologies*) which are constructed for each computational node. By reducing the interpolating space in this manner, the authors were able to achieve more stable solutions as compared to global techniques, while still utilizing the accuracy of the underlying RBF interpolations. The local RBF collocation method quickly gained popularity with applications in all areas of engineering analysis including general diffusion [89], metal casting with phase changes [90, 91, 92], fluid flow and convective heat transfer [93, 94, 95, 96], and computational mechanics [97]. In addition, the adaptability of collocation techniques have allowed researchers to develop novel extensions to the standard local RBF collocation techniques. For example, Stevens et al. [76, 98] developed a method of imposing the governing equations throughout the domain through use of a local Hermitian interpolation scheme. This scheme allows for a means of directly satisfying both the governing equation and boundary conditions at each collocation point through the basis functions themselves and results in increased accuracy of the underlying partial derivatives (however, there is still the unanswered question as to how to linearize the differential operators in this technique for general problem domains). Another notable source of development has been the work performed by Šarler and Vertnik [99] by which local, direct differentiation of the radial basis functions has yielded impressive results in the area of incompressible turbulent flows [100]. Most recently, work has been performed to apply the method to compressible turbulent flows [101] as well as flows in the high-mach number regime [102] by utilizing the *advection upstream splitting method* (AUSM) [103].

CHAPTER 3
MESHLESS METHODOLOGY

When approaching meshless methods from a practical point of view, it should be clear that there are many different techniques which may be utilized to obtain a capable method. However, the goal of this research is not simply to develop a capable method, but to develop a general solution methodology able to compete against more traditional methods such as finite element and finite volume. To accomplish this, it is important to first identify the common characteristics of current methods and where they fall short of satisfying the overall goal of this research. Certainly, as the name implies, the primary goal of all meshless methods is to remove the need for an underlying connectivity mesh during the solution process. However, this goal actually seeks to address two distinct problems with mesh-based techniques: (a) generating the necessary meshes is a time consuming process involving considerable human interaction, and (b) solution quality can be highly dependent on the quality of the mesh. Fully investigating both of these issues is critical because it is common to mistake the implications or misinterpret the underlying causes, leading to misinformation regarding the benefits of meshless methods. This chapter will begin by exploring these two issues in detail and illustrate why they are such a concern to the engineering community. Following this, the capabilities of current meshless methods will be investigated with respect to the overall goals of the solution technique. Finally, this chapter will present the specific meshless methodology utilized throughout the remainder of this research, and detail how it addresses the shortcomings of current methods.

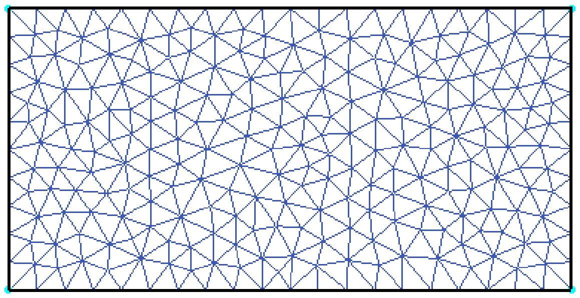
First, with regards to generating the underlying mesh, there is a common misconception that this argument refers to the actual mesh generation process, in which case most experienced FEM or FVM users would respond by indicating that modern unstructured mesh generation routines can largely automate this process. Although this is true, many people overlook the fact that most real-world computational models begin as solid models which may consist of poorly formed faces and edges, small features that may have little effect on the solution (such as fillets in the case of heat transfer), and other areas which can cause problems for automatic mesh generation software. Take, for example, the three variations of a simple cube shown in Figure 3.1; despite all three models representing similar parts, the one with relatively small fillets (shown in Figure 3.1c) will require a mesh with smaller elements in order to properly represent the geometry while still maintaining acceptable mesh quality (in terms of edge and volume aspect ratios). For this simple model the increased node count may be acceptable, however, for larger parts and assemblies which may have many of these fine features, it quickly becomes prohibitive (in terms of computational cost and memory) to simply decrease the mesh size to accommodate these components. As such, more often than not, a human must first manually clean the original solid model by removing unnecessary or ill-constructed features to prepare it for the mesh generation process. This effort is generally the largest contributor to the human involvement during mesh generation, not the mesh generation itself. This, coupled with the fact that this process must be repeated whenever there is a change to the underlying solid model, is the primary reason why the mesh generation process is such a burden on engineering design.



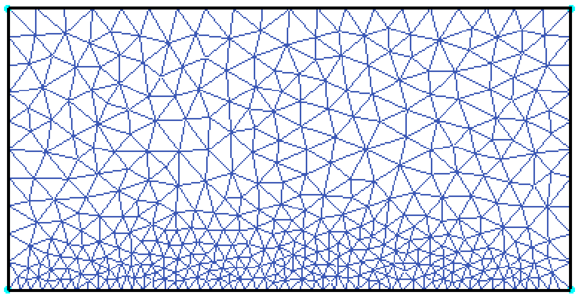
(a) Large Fillet (b) Medium Fillet (c) Small Fillet

Figure 3.1: Simple Cube with Fillet

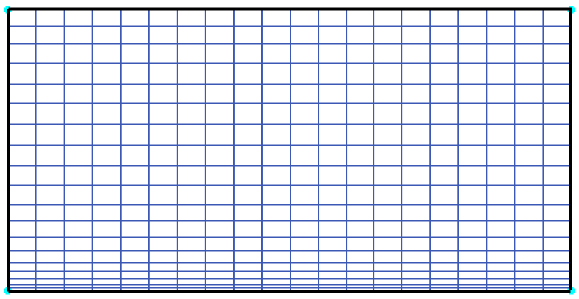
The second issue has to do with the fact that the solution quality of methods such as finite elements and finite volumes can be highly dependent on the quality of the underlying mesh. As there has been a considerable amount of research done on the effects of mesh quality on solutions [104, 105], there can be no denying that this fact is true. However, this does not necessarily imply that this is always a concern; in fact, for most models, current automated mesh generation algorithms are capable of producing satisfactory meshes (in terms of numerical error due to skewed elements) without any human interaction. However, there are two major limitations which still make this issue a concern in real-world problems. First, as stated previously, in order to produce an acceptable mesh for geometry with complex features, the mesh generation software may require an unnecessarily small mesh size, something which may not always be possible within the confines of modern computer architectures. Second, and arguably more important, is the issue of being able to properly determine an appropriate mesh size for a given problem. For example, Figure 3.2 illustrates three potential meshes for a simple flow over a flat plate problem, where, in these figures, the flow is incoming from the left and the flat plate is located at the bottom of the domain. These figures demonstrate three different approaches to meshing this problem: (a) unstructured mesh (Figure 3.2a), (b) clustered unstructured mesh (Figure 3.2b), and (c) high-aspect ratio structured mesh (Figure 3.2c), and depending on the characteristics of the flow, these three meshes may give vastly different results. The primary concern with all three meshes is that none of them implicitly (i.e. automatically) considers the actual flow characteristics of the problem. The unstructured mesh shown in Figure 3.2a is arguably the easiest to generate as it can be done automatically, however, if the mesh size is too large, it may fail to properly capture the boundary layer that will form over the flat plate. At the same time, if a uniform mesh is created which is too small, then excess computational effort will be needed to solve the problem. Only an unstructured mesh which is clustered, such as the one shown in Figure 3.2b, will be able to capture the boundary layer with minimal excess computational effort. However, this type of mesh relies on the user to input an acceptable clustering factor, a process which not only takes time, but is also error prone if the user is not experienced in mesh generation and solution characteristics. Arguably the best mesh for this type of problem would be a structured mesh exhibiting high aspect ratios, as shown in Figure 3.2c, however, the generation of this type of mesh is done almost completely manually (and thus, prone to errors), and becomes exceedingly difficult as the underlying geometry becomes more complex. Together, these two limitations prevent fully-automated mesh generation from being applied to all but the simplest of geometric configurations due to the importance of producing an acceptable mesh and the inability of automated mesh generation procedures to consistently accomplish this task.



(a) Unstructured Mesh



(b) Clustered Mesh



(c) High Aspect Ratio Mesh

Figure 3.2: Example Meshes for Flat Plate

The question that should now be asked is whether or not simply removing the reliance on an underlying nodal connectivity eliminates both of these issues. Unfortunately, it would appear at first glance that the answer is no. By examining the supposed disadvantages of mesh-based techniques, now with focus on meshless methods, it is quite easy to identify why this is the case. First, with regard to the problem of human interaction during mesh generation, there is no doubt that the elimination of an underlying connectivity will remove the need for complex mesh generation procedures. However, it does not directly address the issue of disparate feature sizes. This is an important distinction to make, as it has already been demonstrated that the problem with automatic mesh generation procedures is that they are incapable of dealing with highly disparate features without reducing the mesh size to the smallest common denominator, thereby adding an excessive number of nodes throughout the bulk of the domain. Therefore, the elimination of a structured connectivity will eliminate the need for automatic mesh generation procedures, however, no meshless method property gives it an explicit ability to address these types of geometric concerns. Second, with regards to the solution being dependent on the underlying mesh quality, meshless methods are still approximating the solution over discretized space, and as such, will still be (at least somewhat) reliant on the distribution of points utilized to solve the problem. One could argue that meshless methods are able to obtain better convergence characteristics than similar mesh-based methods; however, that does not overcome the fact that if enough nodes are not located in a region of space to capture a highly complex behavior (such as a boundary layer) there will be an incurred error in that region.

If meshless methods do not explicitly alleviate the issues with mesh-based techniques, why have they garnered so much attention? Interestingly, it is not simply the elimination of the underlying mesh that solves the issues with mesh-based methods, but rather the liberties that are granted during point distribution which facilitates ease of use and solution generality. Unfortunately, this is a point often missed by researchers, and consequently, so is the fact that the underlying point distribution techniques are arguably as important as the methods themselves. As it is these so-called liberties that truly give meshless methods their advantage over mesh-based techniques, the focus of development shouldn't simply be on removing the need for an underlying connectivity mesh, as originally stated, but rather, on *removing the need for an underlying connectivity mesh and facilitating robustness with regards to the underlying point distribution.* This second point is critical because if a meshless method is overly sensitive to the underlying point distribution (which can be quite common) then it will be difficult to develop a point distribution method that can be applied to arbitrarily complex geometries. Thus, formalizing these goals results in the following requirements of the meshless method technique:

1. Must require no underlying connectivity between nodes
2. Must be robust in terms of stability and accuracy
3. Must not use excess computational resources (speed or memory)
4. Must be capable of being applied to general solution domains

The first requirement is obvious and is explicitly satisfied by all meshless methods. The second requirement is what allows the underlying point distribution method to be applied to arbitrarily complex geometries, and as such, is of critical importance. To satisfy this requirement a method must not only be stable with regard to the regularity of the point distribution, but must also be capable of representing solutions with acceptable accuracy. Although these two requirements generally go without saying, as is the case with many numerical techniques, they are oftentimes highly competing goals. The third requirement is a necessity if the method is going to be utilized in real-world problem scenarios, especially large problems on realistic geometries. Finally, the fourth requirement simply ensures that the method can handle arbitrary boundary condition types, discontinuities, and other potential solution features that would otherwise prevent it from being applied to a specific engineering field.

Together, these requirements dictate the necessary capabilities of a meshless method to be successful in addressing the primary issues of mesh-based techniques. Once these requirements have been satisfied by a method, it may be utilized in conjunction with a point distribution algorithm to develop a complete solution process. In order for the automatic point distribution method to be successful in its tasks, it requires the following capabilities:

1. Must be capable of being generated without any human interaction
2. Must be able to properly handle small geometric features
3. Must allow for adaptive, local refinement
4. Must be capable of producing and maintaining disparate aspect ratios

The first requirement of the point distribution method is obvious given the fact that the overall goal is to eliminate human input during the preprocessing stages. However, it implies that the remaining three requirements must be capable of being designed such that they can operate without any human assistance. The second requirement directly addresses the issues of mesh-based techniques, in other words, by being able to properly handle small geometric features the point distribution method will have eliminated the need for model cleanup, and thus, will have significantly reduced the human expense of running an engineering model. The third and fourth requirements address the issue presented in the flat plate example illustrated in Figure 3.2, whereby the necessary grid size is unknown without first examining the problem characteristics. By implementing a locally adaptive refinement procedure, a point distribution method will be capable of determining the appropriate spacing automatically, as the actual solution develops. In addition, by requiring that the point distribution be capable of producing and maintaining disparate aspect ratios, point distributions exhibiting similar qualities to the mesh shown in Figure 3.2c will be obtainable, which will significantly reduce the unnecessary expense of the refinement process as refinement will occur in only those directions for which it is deemed necessary.

Having formalized the requirements for both the meshless solution and point distribution technique, a meshless method to specifically satisfy these requirements will now be outlined. The first major decision was to select a collocation-based meshless approach to serve as the foundation of the solution scheme. Collocation was chosen for several reasons, the first being that it is a point-based approach, and as such, can be applied directly to a solution domain without special consideration for boundary condition application, as is often the case when utilizing a non-interpolating approximation such as moving least squares [106]. The second reason for choosing collocation was due to the fact that collocation techniques can be formulated such that their computational time and memory requirements are kept at a minimum (due to the local nature of the formulations). The final reason is that collocation allows for use of a variety of interpolation schemes in order to develop the underlying shape functions for field and derivative evaluation. Understanding that the meshless method will utilize collocation to formulate the updating scheme for the governing equations, the next step was to decide on appropriate shape functions to represent the underlying solution field and its derivatives. It is in this respect that the current method departs from current techniques in that no single interpolating method is used to construct the necessary shape functions. Instead, a blend of moving least squares, radial basis function interpolation, and virtual finite differencing is utilized to obtain a method that is both stable and accurate. This departure allows for a method which is not married to any particular interpolation scheme, and as such may take advantage of the relative strengths and weaknesses of each technique. The mathematical details of the various shape function techniques utilized in this procedure will be detailed in Chapter 4, along with a presentation of specific implementation details in Chapter 5. The second major development is the description of a point distribution method which is capable of satisfying the necessary requirements in an efficient manner. This point distribution method, based upon a quaternary triangular surface discretization, a binary-subdivision interior discretization, and an adaptive boundary layer representation (termed the *shadow layer*), represents a novel means of generating the computational nodes for any meshless solution process. Capable of directionally independent adaptive refinement, global interpolation, and surface influence screening, it represents the most complete, meshless-specific point distribution method developed to date, and will be presented in full detail in Chapter 6. The final component of the method is an h-adaptive refinement strategy allowing for efficient adaptation of the underlying geometry used throughout the solution process. By coupling the solution and model generation processes into a single adaptive procedure, the presented method allows for an entirely automated process for solving any set of partial differential equations over arbitrarily complex domains. The details of the h-adaptive refinement strategy are presented in Chapter 7. Having fully described the method, Chapter 8 will then address specific application area details (including upwinding and shape function selection) that arise when applying the described methodology to practical governing equations. The specific combination of numerical method and model generation procedure outlined in the remainder of this paper is collectively referred to as the Model Integrated Meshless Solution (MIMS) method and represents the first major attempt at producing an industrially relevant meshless method implementation capable of competing with more traditional mesh-based techniques such as finite element and finite volume methods.

CHAPTER 4
SHAPE FUNCTION CONSTRUCTION

The first step in developing a collocation based meshless method is determining the approximation techniques that will be used to represent the necessary derivatives. However, prior to examining the specific shape function techniques used in the MIMS method, it is worth presenting a brief description of collocation in order to introduce the specific nomenclature that will be used throughout the remainder of the chapter.

Regardless of the approximation technique(s) utilized, the necessary component for any point collocation method is a set of shape functions $\underline{\Phi}$ which, when multiplied by a set of known field values $\underline{u}$, provides an estimate for the field or its derivatives at a specific point in space. Once the shape functions have been developed for each computational node, the governing equations may be collocated (explicitly satisfied) at each point, leading to an update equation capable of advancing an initial solution through either an iteration or timestep procedure to a final, converged solution. For example, Figure 4.1 illustrates a one-dimensional domain containing $n + 1$, equally spaced computational nodes, over which the solution to the following governing equation is desired

$$\frac{\partial^2 u}{\partial x^2} = u''' \tag{4.1}$$

where u''' represents a uniform (constant) generation over the domain.

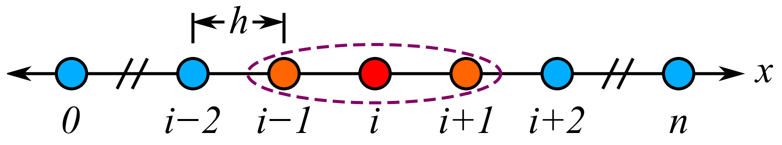


Figure 4.1: One Dimensional Topology

To solve Eq. (4.1), a method of approximating $\partial^2 u/\partial x^2$ is required, and on this simple, structured domain, standard finite difference approximations may be utilized. As the details of finite differencing will be presented in Section 4.3, for now it will be sufficient to note that at any given location, the second derivative with respect to any direction may be approximated as

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_x = \frac{u\left(x-\Delta x\right)-2u\left(x\right)+u\left(x+\Delta x\right)}{\Delta x^2} = \frac{u_{i-1}-2u_i+u_{i+1}}{h^2} \tag{4.2}$$

where u_i is the field solution at node i and h is the nodal spacing throughout the domain. Thus, Eq. (4.2) may be represented in terms of a set of shape functions as

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i = \left\{ \begin{array}{ccc} \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \end{array} \right\} \left\{ \begin{array}{c} u_{i-1} \\ u_i \\ u_{i+1} \end{array} \right\} = \underline{\Phi}_i \, \underline{u}_i \tag{4.3}$$

where u_{i-1} and u_{i+1} are the values at the previous and next positions to the i-th node, respectively. At this point, the expression for $\partial^2 u/\partial x^2$ at any interior node may be represented in terms of a set of shape functions $\underline{\Phi}$ and current nodal values $\underline{u}$. It is worth noting that the surrounding nodes which contribute to the value at a particular location (u_{i-1}, u_{i+1}, and u_i for this example) are referred to as the *support domain* or *topology* of that location and are largely the meshless equivalent of the underlying connectivity present in mesh-based techniques.

Once the necessary differential operators have been obtained, each instance of the differential operator in the governing equation may be replaced by its equivalent shape function representation. For example, substituting Eq. (4.3) into the governing equation shown in Eq. (4.1) results in the following system of equations

$$\underline{\Phi}_i \, \underline{u}_i = u''' \qquad i = 1, 2, \ldots, n-1 \tag{4.4}$$

If it is assumed that appropriate boundary conditions have been applied at nodes 0 and n, then Eq. (4.4) represents a system of equations for which the unknown nodal values $\underline{u}$ can be determined. Assembling and solving the full system of equations for the unknown values directly is referred to as an implicit technique and is commonly utilized in the case of linear partial differential equations. Alternatively, each equation provided by Eq. (4.4) may be utilized as an update rule for the central node. For example, in the case of the current governing equation, each node may be updated according to

$$u_i^{(k+1)} = \frac{u_{i-1}^{(k)} + u_{i+1}^{(k)} - h^2 u'''}{2} \tag{4.5}$$

where $u_i^{(k)}$ represents the value at the i-th node at the k-th iteration. Thus, provided an initial guess $\underline{u}^{(0)}$, Eq. (4.5) may be used to explicitly solve Eq. (4.1).

Having demonstrated the process of solving a partial differential equation via collocation, the challenge for meshless methods is determining appropriate shape functions without a defined point connectivity. To accomplish this, underlying interpolations are utilized to represent the field locally to a point of interest, and are then differentiated to obtain the necessary differential operators. As such, the remainder of this chapter will detail the various approximation/interpolation techniques utilized in the MIMS method.

4.1 Moving Least Squares (MLS)

Moving least squares (MLS), or local regression and loss, was originally developed as a curve fitting technique to be applied to noisy or piecewise data [41, 107]. MLS approximations utilize basis functions (generally polynomials) to best represent a field by minimizing the error across a local set of data points. Unlike interpolation techniques, MLS does not exhibit the Kronecker delta function property, implying that $u_{MLS}(\mathbf{x}_i) \neq u_i$ where $u_{MLS}(\mathbf{x}_i)$ is the MLS approximated value and u_i is the actual value at point $\mathbf{x}_i$. This fact allows the technique to generate smoother field approximations than similarly formulated interpolation methods, avoiding the oscillations oftentimes seen between data points due to over-fitting. The ability of MLS to smooth noisy data is what has garnered it so much attention, having been used in curve fitting [108, 109], surface reconstruction [41, 110, 111], and other applications where over-fitting of the underlying data is a concern [112]. Within the context of meshless methods, MLS was first used by Nayroles et al. [39] in their diffuse element method (DEM). Shortly after, Belytchko et al. [29, 45] modified DEM by introducing Lagrange multipliers to more efficiently apply Dirichlet boundary conditions and an underlying cell structure with which they could generate the necessary quadrature points for the formulation. This new technique, which was given the name element free Galerkin method (EFG) by its authors, gained considerable popularity and led to many of the later advancements within the field of meshless methods.

Fundamental to both DEM and EFG was the use of MLS approximations in constructing the necessary shape functions for both the field and its derivatives. To accomplish this, MLS formulations begin by representing the field as

$$u\left(\mathbf{x}\right)=\sum_{j=1}^{m}a_{j}\left(\mathbf{x}\right)p_{j}\left(\mathbf{x}\right)=\underline{p}^{\mathrm{T}}\left(\mathbf{x}\right)\underline{a}\left(\mathbf{x}\right) \tag{4.6}$$

where $u\left(\mathbf{x}\right)$ is the field value at location $\mathbf{x}$, m is the number of basis functions utilized in the approximation, $p_j\left(\mathbf{x}\right)$ is the j-th basis function, and $a_j\left(\mathbf{x}\right)$ is the coefficient for the j-th basis function. The MLS basis functions are generally chosen as the lowest order monomials able to provide an acceptable representation of the underlying field. Typically, these monomials are chosen by selecting a given depth (constant, linear, quadratic, etc.) from Pascal's pyramid of monomials [106], as shown in Figure 4.2. Once an appropriate depth is chosen, $\underline{p}\left(\mathbf{x}\right)$ is built by utilizing all monomial terms on and below this depth; for example, if a quadratic depth was chosen, the three-dimensional basis function set would consist of

$$\underline{p}^{\mathrm{T}}\left(\mathbf{x}\right)=\underline{p}^{\mathrm{T}}\left(x,y,z\right)=\left\{\begin{array}{cccccccccc}1 & x & y & z & x^2 & y^2 & z^2 & xy & yz & zx\end{array}\right\} \tag{4.7}$$

Understanding that the goal of MLS is to minimize the error at each data point within some local region, the total error over a set of n data points may be formally written as

$$J=\sum_{i=1}^{n}W\left(\mathbf{x}-\mathbf{x}_i\right)\left(\underline{p}^{\mathrm{T}}\left(\mathbf{x}_i\right)\underline{a}\left(\mathbf{x}\right)-u_i\right)^2 \tag{4.8}$$

where $\mathbf{x}_i$ is the position of the i-th node, u_i is the field value at the i-th node, and $W\left(\mathbf{x}-\mathbf{x}_i\right)$ is a weight function that indicates the relative effect of each node to the point of interest, $\mathbf{x}$. The weight function is critical because it allows the implementer to provide higher importance to nodes closer to the point of interest, as well as providing a gradual transition from one topology to the next. Although there are many common weight functions utilized in MLS

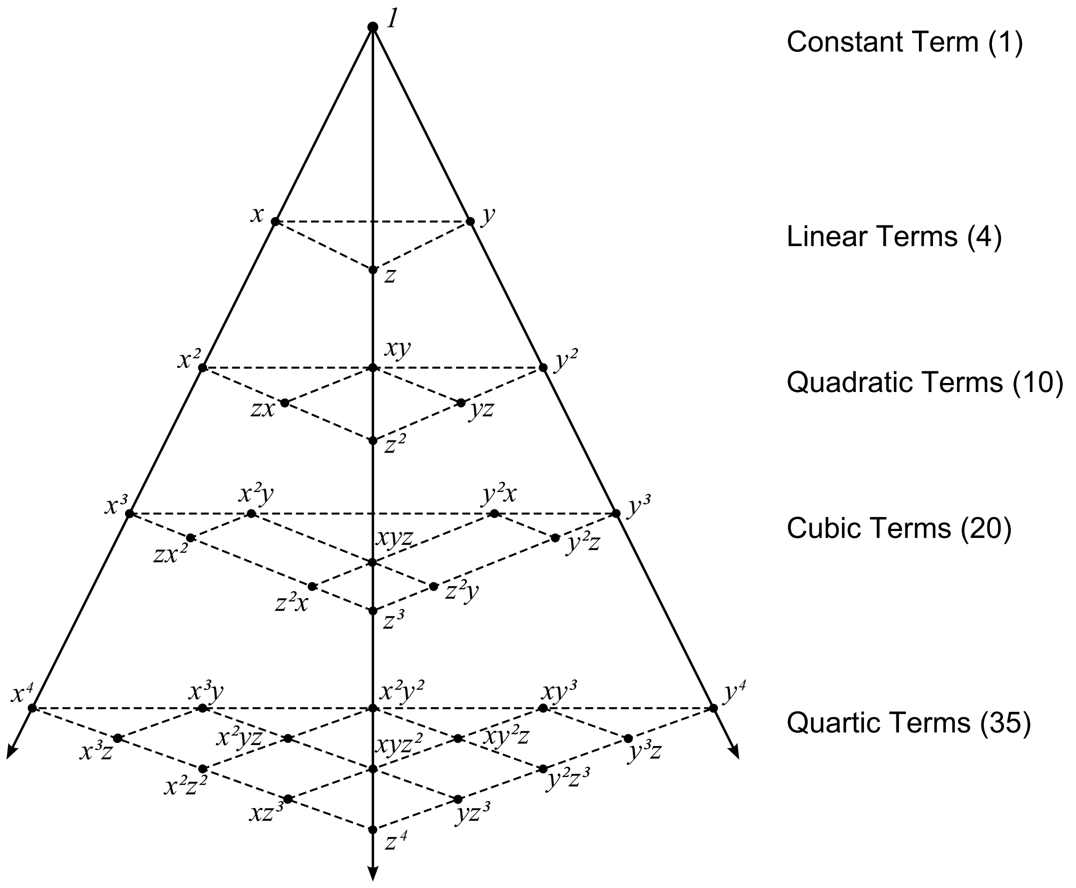


Figure 4.2: Pascal Pyramid of Monomials (Reproduced from [106])

approximations, the quartic weight function presented by Liu et al. [113] demonstrates several ideal properties for meshless implementations. First, because it is a single expression (as opposed to more complex piecewise representations), it is easily differentiable. Second, both the first and second derivatives are smooth, which aids in satisfying the compatibility condition throughout the domain. Thus, the quartic weight function $W(\mathbf{x}-\mathbf{x}_i)$ may be expressed as

$$W(\mathbf{x}-\mathbf{x}_i) = W\left(\bar{d}\right) = \begin{cases} \frac{2}{3} - \frac{9}{32}\bar{d}^2 + \frac{19}{192}\bar{d}^3 - \frac{5}{512}\bar{d}^4 & \text{for } \bar{d} \le 1 \\ 0 & \text{for } \bar{d} > 1 \end{cases} \tag{4.9}$$

where $\bar{d}$ is a scaled dimensional parameter used to ensure consistent influence in each topology and is commonly computed as

$$\bar{d} = \frac{|\mathbf{x} - \mathbf{x}_i|}{\alpha r_s} \tag{4.10}$$

with r_s equal to the radius of the topology (computed as the distance from the center node to its most extreme neighbor) and α taking on a value greater than 1. In the context of Eq. (4.9), α dictates the size of the topology for which $W \neq 0$, and as such, generally takes on values between 1 and 3.

To produce an MLS approximation at an arbitrary point $\mathbf{x}$, the error representation shown in Eq. (4.8) must be minimized with respect to the expansion coefficients $\underline{a}\,(x)$. This is accomplished by setting the derivative with respect to the expansion coefficients equal to zero as

$$\frac{\partial J}{\partial \underline{a}} = 0 \tag{4.11}$$

which results in the system of linear equations

$$\underline{\underline{A}}\,(\mathbf{x})\,\underline{a}\,(\mathbf{x}) = \underline{\underline{B}}\,(\mathbf{x})\,\underline{u} \tag{4.12}$$

where $\underline{u}$ is a vector containing the known field values for each node in the topology

$$\underline{u}^{\mathrm{T}} = \left\{ \begin{array}{ccccc} u_1 & u_2 & u_3 & \dots & u_n \end{array} \right\} \tag{4.13}$$

$\underline{\underline{A}}$ is typically referred to as the weighted moment matrix

$$\underline{\underline{A}}\,(\mathbf{x}) = \sum_{i=1}^{n} W\,(\mathbf{x} - \mathbf{x}_i)\,\underline{p}\,(\mathbf{x}_i)\,\underline{p}^{\mathrm{T}}\,(\mathbf{x}_i) \tag{4.14}$$

and $\underline{\underline{B}}$ is of the form

$$\underline{\underline{B}}(\mathbf{x}) = \begin{bmatrix} \underline{B}_1 & \underline{B}_2 & \dots & \underline{B}_n \end{bmatrix}_{n \times m} \tag{4.15}$$

$$\underline{B}_i = W(\mathbf{x} - \mathbf{x}_i)\,\underline{p}(\mathbf{x}_i) \tag{4.16}$$

Solving Eq. (4.12) for $\underline{a}(\mathbf{x})$ gives

$$\underline{a}(\mathbf{x}) = \underline{\underline{A}}^{-1}(\mathbf{x})\,\underline{\underline{B}}(\mathbf{x})\,\underline{u} \tag{4.17}$$

Substituting this result into Eq. (4.8) results in the following expression for $u(\mathbf{x})$

$$u(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_j(\mathbf{x})\left(\underline{\underline{A}}^{-1}(\mathbf{x})\,\underline{\underline{B}}(\mathbf{x})\right)_{ji} u_i \tag{4.18}$$

Realizing that the shape functions can be directly extracted from Eq. (4.18), this expression may be written as

$$u(\mathbf{x}) = \underline{\Phi}(\mathbf{x})\,\underline{u} \tag{4.19}$$

where

$$\underline{\Phi}(\mathbf{x}) = \underline{p}(\mathbf{x})\,\underline{\underline{A}}^{-1}(\mathbf{x})\,\underline{\underline{B}}(\mathbf{x}) \tag{4.20}$$

It should be noted that in order for $\underline{\underline{A}}$ to be non-singular, the number of nodes in the local topology (n) should be much larger than the number of monomial terms utilized in the approximation (m). Although this does not explicitly guarantee the existence of $\underline{\underline{A}}^{-1}$, for most practical scenarios enforcing $n \gg m$ is a sufficient condition for this to be true.

To arrive at the derivatives of $u(\mathbf{x})$, which are necessary to formulate the governing equation into an explicit form, Eq. (4.19) may be differentiated as

$$\partial u(\mathbf{x}) = \partial\underline{\Phi}(\mathbf{x})\,\underline{u} \tag{4.21}$$

where ∂ may represent any derivative operator (such as $\partial/\partial x$, $\partial^2/\partial x^2$, ∇^2, etc.). Note, however, from Eq. (4.20) that all three components of $\underline{\Phi}$ ($\underline{p}$, $\underline{\underline{A}}^{-1}$, and $\underline{\underline{B}}$) are functions of $\mathbf{x}$, and as such, care must be taken to properly differentiate each term. Complete formulations have been presented up to third order derivatives [106], however, in practice, direct differentiation of the MLS shape functions is impractical for anything beyond first derivatives [114, 115]. Thus, for the case of first derivatives, the shape functions may be differentiated as

$$\frac{\partial \Phi(\mathbf{x})}{\partial x_k} = \underline{\underline{A}}^{-1}(\mathbf{x}) \frac{\partial \underline{p}(\mathbf{x})}{\partial x_k} \underline{\underline{B}}(\mathbf{x}) + \underline{p}(\mathbf{x}) \underline{\underline{A}}^{-1} \left(\frac{\partial \underline{\underline{B}}(\mathbf{x})}{\partial x_k} - \frac{\partial \underline{\underline{A}}(\mathbf{x})}{\partial x_k} \underline{\underline{A}}^{-1}(\mathbf{x}) \underline{\underline{B}}(\mathbf{x}) \right) \tag{4.22}$$

where k may equal 1, 2, or 3, indicating differentiation with respect to x (x_1), y (x_2), or z (x_3). In addition, evaluation of Eq. (4.22) requires computation of partial derivatives of $\underline{p}$, $\underline{\underline{A}}$, and $\underline{\underline{B}}$. Realizing from Eqs. (4.14) and (4.16) that only the weight function is a function of $\mathbf{x}$ (the polynomial terms are a function of $\mathbf{x}_i$, which does not vary over the topology), differentiation of these terms is fairly straightforward, and may be obtained as

$$\frac{\partial \underline{\underline{A}}(\mathbf{x})}{\partial x_k} = \sum_{i=1}^{n} \frac{\partial W(\mathbf{x} - \mathbf{x}_i)}{\partial x_k} \underline{p}(\mathbf{x}_i) \underline{p}^{\mathrm{T}}(\mathbf{x}_i) \tag{4.23}$$

and

$$\frac{\partial \underline{\underline{B}}(\mathbf{x})}{\partial x_k} = \begin{bmatrix} \frac{\partial \underline{B}_1}{\partial x_k} & \frac{\partial \underline{B}_2}{\partial x_k} & \cdots & \frac{\partial \underline{B}_n}{\partial x_k} \end{bmatrix}_{n \times m} \tag{4.24}$$

$$\frac{\partial \underline{B}_i}{\partial x_k} = \frac{\partial W(\mathbf{x} - \mathbf{x}_i)}{\partial x_k} \underline{p}(\mathbf{x}_i) \tag{4.25}$$

It is also worth noting that by distributing the $\underline{p}(\mathbf{x}) \underline{\underline{A}}^{-1}$ term into the parenthesis within Eq. (4.22), one may avoid having to perform any matrix-matrix multiplications throughout this process.

It is worth noting at this time that although MLS approximations and their corresponding derivative approximations may be used in formulations following a weighted residuals or Galerkin formulation, they are typically mutually exclusive from collocation based approaches due the lack of Kronecker delta function property (the negative effects of this will be demonstrate later in Chapter 5). Despite this, they still serve an important role as a generalized approximation scheme that can be used where collocation is either satisfied by other means (i.e. Virtual Finite Differencing) or as a post-processing tool where collocation is not necessary. In addition, several governing equations (such as incompressible flow via pressure correction) require approximations of derivatives (oftentimes on boundaries) to impose as initial, or boundary conditions to subsequent problems. Both of these applications justify the use of MLS and as such, it is important to consider it an available tool when developing a robust meshless methodology.

4.2 Radial Basis Function with Polynomial Reproduction (RBFP)

Radial Basis Function (RBF) interpolation, like MLS, was originally developed as a general curve fitting and surface reconstruction technique [116, 68, 70, 66] and is commonly utilized within the area of medical imaging [69, 71] to construct patient geometry from sparse imaging data. Unlike MLS, RBF is an interpolation process (i.e. it satisfies the Kroneker delta property), and as such, the accuracy of RBF tends to improve as the topology size is increased. In addition, because non-polynomial basis functions are utilized (unlike other commonly used unstructured data interpolation schemes), the underlying RBF moment matrix is guaranteed to be non-singular if care is taken during selection of the so-called shape parameter. Within the context of meshless methods, RBF interpolations have been separately introduced through two distinct research paths. First, because of the non-singular nature of RBF moment matrices, point interpolation methods often utilize RBF basis functions (as an alternative to mononomials) in an attempt to increase the generality of the technique [117, 118, 119]. Second, RBF interpolations were introduced through spectral and pseudo-spectral methods [3, 4, 9, 7, 11], which are typically based on global orthogonal functions such as Legendre or Chebyshev polynomials (and require a regular nodal point distribution); though later efforts were able to generalize the concepts to be applied to irregular point distributions, thus gaining the meshless namesake [84, 120]. Regardless of the means of introduction, RBF interpolation offers advantages over MLS for some situations, and as such, is worth examining for use within the meshless collocation solution process.

Similar to previous techniques, construction of the RBF shape functions begin by representing the field as a finite series representation, multiplying a set of basis functions by a set of expansion coefficients as:

$$u\left(\mathbf{x}\right) = \sum_{i=1}^{n} \alpha_i R_i\left(\mathbf{x}\right) = \underline{R}^{\mathrm{T}}\left(\mathbf{x}\right) \underline{\alpha} \tag{4.26}$$

where $u\left(\mathbf{x}\right)$ is the field value at location $\mathbf{x}$, n is the number of nodes in the local topology at position $\mathbf{x}$, α_i is the i-th expansion coefficient, and R_i is a radial basis function based on the distance between point $\mathbf{x}$ and point $\mathbf{x}_i$. There have been many suggested radial basis functions, with some of the most commonly used shown in Table 4.1.

Table 4.1: Typical Radial Basis Functions

Name	Expression	Shape Parameters
Multiquadrics	$R_i(\mathbf{x}) = (r_i^2 + c^2)^q$	c, q
Gaussian	$R_i(\mathbf{x}) = \exp(-cr_i^2)$	c
Thin Plate Spline	$R_i(\mathbf{x}) = r_i^\eta$	η
Logarithmic	$R_i(\mathbf{x}) = r_i^\eta \log(r_i)$	η

Note that for all radial basis functions shown in Table 4.1, r_i is defined as the radial distance between location $\mathbf{x}$ and $\mathbf{x}_i$, which in three-dimensions is given by:

$$r_i = r_i(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_i| = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \tag{4.27}$$

where $\mathbf{x} = \{x \;\; y \;\; z\}^{\mathrm{T}}$ and $\mathbf{x}_i = \{x_i \;\; y_i \;\; z_i\}^{\mathrm{T}}$. Although all of the basis functions listed in Table 4.1 have been used in meshless method implementations to some extent, many agree that the family of so-called Inverse Hardy Multiquadrics (MQ) [67] (a Multiquadric with $q = -0.5$) produce the most stable and accurate results across the largest subset of problem domains [65, 64, 72, 121]. As such, the specific radial basis function employed will be the inverse Hardy MQ of the form:

$$R_i(\mathbf{x}) = \frac{1}{\sqrt{r_i^2 + c^2}} \tag{4.28}$$

where c is known as the shape parameter, and will be addressed in detail in Chapter 5.

Having described the basis function used in this interpolation technique, the next step is to enforce Eq. (4.26) at each node in the topology, which results in the following expressions:

$$\begin{aligned} u\left(\underline{x}_1\right) &= \sum_{i=1}^{n} \alpha_i R_i\left(\underline{x}_1\right) \\ u\left(\underline{x}_1\right) &= \sum_{i=1}^{n} \alpha_i R_i\left(\underline{x}_2\right) \\ &\vdots \\ u\left(\underline{x}_n\right) &= \sum_{i=1}^{n} \alpha_i R_i\left(\underline{x}_n\right) \end{aligned} \tag{4.29}$$

Each node produces a single equation which, when combined, produces the following system of equations:

$$\begin{bmatrix} R_1\left(\underline{x}_1\right) & R_2\left(\underline{x}_1\right) & \dots & R_n\left(\underline{x}_1\right) \\ R_1\left(\underline{x}_2\right) & R_2\left(\underline{x}_2\right) & \dots & R_n\left(\underline{x}_2\right) \\ \vdots & \vdots & \ddots & \vdots \\ R_1\left(\underline{x}_n\right) & R_2\left(\underline{x}_n\right) & \dots & R_n\left(\underline{x}_n\right) \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{Bmatrix} = \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{Bmatrix} \tag{4.30}$$

or simply:

$$\underline{\underline{G}}\,\underline{\alpha} = \underline{u} \tag{4.31}$$

If the inverse of the moment matrix $\underline{\underline{G}}$ exists (which is guaranteed if c is chosen appropriately), the expansion coefficients may be obtained as:

$$\underline{\alpha} = \underline{\underline{G}}^{-1}\underline{u} \tag{4.32}$$

Rather than solve for the expansion coefficients directly, the expression for $\underline{\alpha}$ obtained in Eq. (4.32) may be substituted into Eq. (4.26), leading to the following expression for $u\left(\mathbf{x}\right)$:

$$u\left(\mathbf{x}\right) = \underline{\underline{G}}^{-1}\underline{u}\,\underline{R}\left(\mathbf{x}\right) = \underline{R}^{\mathrm{T}}\left(\mathbf{x}\right)\underline{\underline{G}}^{-1}\underline{u} = \underline{\Phi}\left(\mathbf{x}\right)\underline{u} \tag{4.33}$$

where $\underline{\Phi}(\mathbf{x})$ is the shape function vector, expressed as:

$$\underline{\Phi}(\mathbf{x}) = \underline{R}^{\mathrm{T}}(\mathbf{x})\,\underline{\underline{G}}^{-1} \tag{4.34}$$

Notice from Eq. (4.34) that the RBF shape function vector at a given position depends entirely on geometric quantities, and as such, may be precomputed and stored during preprocessing.

Prior to discussing RBF derivatives, it is important to note that as formulated, the RBF interpolations will not pass standard patch tests due to their inability to reproduce constant and linear fields. Although slightly paradoxical, pure radial basis functions are better suited to more complex field solutions and as such, exhibit some stability issues when presented with constant and linear solutions. To remedy this situation, RBF interpolations are commonly augmented with polynomial terms. As several researches have demonstrated [122, 106, 123], adding polynomial terms to the RBF formulation almost universally improves the resulting interpolations with minimal added computational effort. Thus, to add polynomial terms, Eq. (4.26) is rewritten to include a set of additional polynomial basis functions as:

$$u(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i R_i(\mathbf{x}) + \sum_{j=1}^{m} \beta_j p_j(\mathbf{x}) = \underline{R}^{\mathrm{T}}(\mathbf{x})\,\underline{\alpha} + \underline{p}^{\mathrm{T}}(\mathbf{x})\,\underline{\beta} \tag{4.35}$$

where m is the number of polynomial terms added to the approximation, β_j is the j-th polynomial expansion coefficient, and $p_j(\mathbf{x})$ is the j-th polynomial basis function. In a similar manner to Eq. (4.29), Eq. (4.35) may be applied to all of the nodes in the topology, resulting in the following set of expressions:

$$u\left(\mathbf{x}_k\right) = \sum_{i=1}^{n} \alpha_i R_i\left(\mathbf{x}_k\right) + \sum_{j=1}^{m} \beta_j p_j\left(\mathbf{x}_k\right) \qquad k = 1,\, 2, \ldots,\, n \tag{4.36}$$

or in matrix form:

$$\underline{u} = \underline{\underline{G}}\,\underline{\alpha} + \underline{\underline{F}}\,\underline{\beta} \tag{4.37}$$

where G is the same matrix as in Eq. (4.31). Notice that Eq. (4.36) only provides n equations, however, the introduction of polynomial basis terms has increased the number of unknowns to $n + m$. The remaining equations are provided by requiring the polynomials to satisfy an extra requirement which guarantees a unique approximation [122]. This requirement is usually imposed as the following constraint equations:

$$\sum_{i=1}^{n} p_j\left(\mathbf{x}_i\right) \alpha_i = 0 \qquad j = 1,\, 2, \ldots,\, m \tag{4.38}$$

or in matrix form:

$$\underline{\underline{F}}^{\mathrm{T}} \underline{\alpha} = 0 \tag{4.39}$$

Combining Eqs. (4.37) and (4.39) gives:

$$\begin{bmatrix} \underline{\underline{G}} & \underline{\underline{F}} \\ \underline{\underline{F}} & \underline{\underline{0}} \end{bmatrix} \begin{Bmatrix} \underline{\alpha} \\ \underline{\beta} \end{Bmatrix} = \begin{Bmatrix} \underline{u} \\ \underline{0} \end{Bmatrix} \tag{4.40}$$

where $\underline{\underline{G}}$ is the pure RBF moment matrix and $\underline{\underline{F}}$ is the polynomial augmentation matrix given by:

$$\underline{\underline{F}} = \begin{bmatrix} p_1(\mathbf{x}_1) & p_2(\mathbf{x}_1) & \dots & p_m(\mathbf{x}_1) \\ p_1(\mathbf{x}_2) & p_2(\mathbf{x}_2) & \dots & p_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(\mathbf{x}_n) & p_2(\mathbf{x}_n) & \dots & p_m(\mathbf{x}_n) \end{bmatrix} \tag{4.41}$$

Thus Eq. (4.40) may be more succinctly represented as:

$$\underline{\underline{H}} \begin{Bmatrix} \underline{\alpha} \\ \underline{\beta} \end{Bmatrix} = \begin{Bmatrix} \underline{u} \\ \underline{0} \end{Bmatrix} \tag{4.42}$$

with $\underline{\underline{H}}$ representing the fully augmented moment matrix.

At this point, Eq. (4.42) may be used to generate the shape functions in a similar manner to that used in Eqs. (4.32)-(4.34). Alternatively, a more efficient procedure was presented by Liu [106] in which overall matrix multiplications are reduced and the pure RBF moment matrix may be decoupled from the polynomial components. Thus, starting from Eq. (4.37), the RBF expansion coefficients may be solved as:

$$\underline{\alpha} = \underline{\underline{G}}^{-1}\underline{u} - \underline{\underline{G}}^{-1}\underline{\underline{F}}\,\underline{\beta} \tag{4.43}$$

Substituting Eq. (4.43) into Eq. (4.39) gives:

$$\underline{\beta} = \underline{\underline{S_\beta}}\,\underline{u} \tag{4.44}$$

$$\underline{\underline{S_\beta}} = \left(\underline{\underline{F}}^{\mathrm{T}}\underline{\underline{G}}^{-1}\underline{\underline{F}}\right)^{-1}\underline{\underline{F}}^{\mathrm{T}}\underline{\underline{G}}^{-1} \tag{4.45}$$

where $\underline{\underline{F}}^{\mathrm{T}}\underline{\underline{G}}^{-1}\underline{\underline{F}}$ is commonly referred to as a transformed moment matrix. Also notice that $\underline{\underline{F}}^{\mathrm{T}}\underline{\underline{G}}^{-1}$ needs to be computed only once throughout this entire procedure. Substituting Eq. (4.44) back into Eq. (4.39) results in the following expression for the RBF expansion coefficients:

$$\underline{\alpha} = \underline{\underline{S_\alpha}}\,\underline{u} \tag{4.46}$$

where, after simplification:

$$\underline{\underline{S_\alpha}} = \underline{\underline{G}}^{-1} - \left(\underline{\underline{F}}^{\mathrm{T}}\underline{\underline{G}}^{-1}\right)^{\mathrm{T}}\underline{\underline{S_\beta}} \tag{4.47}$$

Finally, Eq. (4.35) may be written as

$$u(\mathbf{x}) = \left(\underline{R}^{\mathrm{T}}\underline{\underline{S_\alpha}} + \underline{p}^{\mathrm{T}}\underline{\underline{S_\beta}}\right)\underline{u} = \underline{\Phi}(\mathbf{x})\,\underline{u} \tag{4.48}$$

where the shape functions for a particular node are given by

$$\underline{\Phi}(\mathbf{x}) = \underline{R}^{\mathrm{T}}\underline{\underline{S_\alpha}} + \underline{p}^{\mathrm{T}}\underline{\underline{S_\beta}} \tag{4.49}$$

Once again, it is important to note that the shape functions provided by Eq. (4.49) are strictly geometrically dependent, and as such, may be precomputed during preprocessing stages of the algorithm.

A major benefit of the formulation shown in Eq. (4.49) is that the direct derivatives of the underlying RBF interpolators may be easily obtained as

$$\partial\underline{\Phi} = \partial\underline{R}^{\mathrm{T}}\underline{\underline{S_\alpha}} + \partial\underline{p}^{\mathrm{T}}\underline{\underline{S_\beta}} \tag{4.50}$$

where ∂ may represent any derivative operator (such as $\partial/\partial x$, $\partial^2/\partial x^2$, ∇^2, etc.) for which analytical derivatives exist for both $R_i(\mathbf{x})$ and $p_j(\mathbf{x})$.

4.3 Virtual Finite Differencing (VML)

Virtual finite differencing (VML), unlike MLS and RBF, is not an approximation or interpolation method but rather a formulation technique which utilizes interpolating shape functions to form a generalized finite difference approach. Although generalized finite difference methods have been around since the early 1980s [52, 53], their applicability has been limited due to a lack of unifying methodology. It is interesting to note that the similarities to conventional finite differencing often cause these techniques to be misclassified as a mesh-based methods, a problem compounded by the fact that many generalized finite differencing formulations are developed for partially structured regions. VML, on the other hand, utilizes the approximation/interpolation qualities of MLS and RBFP to augment traditional finite differencing without introducing any additional overhead once initial preprocessing has been performed. In this respect, VML is a useful tool for developing a robust meshless methodology, representing a truly meshless generalized finite difference approach.

As VML depends largely on traditional finite differencing formulations, it is important to first provide a brief overview of the underlying methodology employed in finite differencing methods. Finite difference approximations start from the Taylor series representation of the derivatives at a given location and truncate the series to produce approximate values that can be evaluated with acceptable error. For example, to arrive at a second order approximation of the first derivative at point x_i, as illustrated in Figure 4.3, two Taylor series may be utilized, one centered at $x + h$ and one centered at $x - h$. Thus, expanding the Taylor series

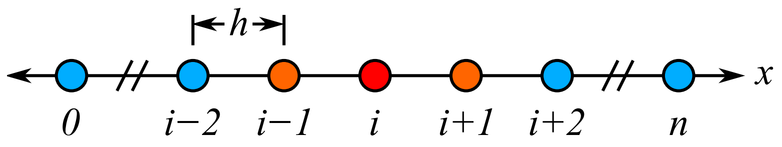


Figure 4.3: One-Dimensional Discretization

about these points results in

$$f\left(x+h\right)=f\left(x\right)+hf'\left(x\right)+\frac{1}{2!}h^2f''\left(x\right)+\frac{1}{3!}h^3f'''\left(x\right)+\frac{1}{4!}h^4f^{(4)}\left(x\right)+\cdots \tag{4.51}$$

$$f\left(x-h\right)=f\left(x\right)-hf'\left(x\right)+\frac{1}{2!}h^2f''\left(x\right)-\frac{1}{3!}h^3f'''\left(x\right)+\frac{1}{4!}h^4f^{(4)}\left(x\right)-\cdots \tag{4.52}$$

Subtracting Eq. (4.51) from Eq. (4.52):

$$f\left(x+h\right)-f\left(x-h\right)=2hf'\left(x\right)+\frac{2}{3!}h^3f'''\left(x\right)+\frac{2}{5!}h^5f^{(5)}\left(x\right)+\cdots \tag{4.53}$$

which may be solved for $f'\left(x\right)$ as

$$f'\left(x\right)=\frac{f\left(x+h\right)-f\left(x-h\right)}{2h}-\frac{2}{3!}h^2f'''\left(x\right)-\frac{2}{5!}h^4f^{(5)}\left(x\right)-\cdots \tag{4.54}$$

The result shown in Eq. (4.54) represents the Taylor series representation of the derivative at a particular location x. To arrive at the finite difference representation, the higher order derivative terms may be truncated, resulting in the following expression

$$f'\left(x\right)\approx\frac{f\left(x+h\right)-f\left(x-h\right)}{2h}+O\left(h^2\right) \tag{4.55}$$

where $O\left(h^2\right)$ indicates that the error is on the order of h^2, as indicated by the first leading term truncated from the series. Placing this equation into a more common form for discrete data sets (keeping with the convention shown in Figure 4.3), results in

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} \approx \frac{u\left(x_i+\Delta x\right)-u\left(x_i-\Delta x\right)}{2\Delta x} = \frac{u\left(x_{i+1}\right)-u\left(x_{i-1}\right)}{2\Delta x} \tag{4.56}$$

where Δx has replaced h in the formulation, and u is the value of a particular field function.

The process of truncating the Taylor series representations of the derivatives may be performed for nearly any desired derivative, and in all cases, the process is roughly the same. As such, there are many texts which fully detail the derivation process for most derivative operators [124], and, for completeness, several of the most common three-dimensional operators are provided in Eqs. (4.57)-(4.65).

$$\left.\frac{\partial u}{\partial x}\right|_{x,y,z} = \frac{u\left(x+\Delta x,y,z\right)-u\left(x-\Delta x,y,z\right)}{2\Delta x} \tag{4.57}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x,y,z} = \frac{u\left(x+\Delta x,y,z\right)-2u\left(x,y,z\right)+u\left(x-\Delta x,y,z\right)}{\Delta x^2} \tag{4.58}$$

$$\left.\frac{\partial u}{\partial y}\right|_{x,y,z} = \frac{u\left(x,y+\Delta y,z\right)-u\left(x,y-\Delta y,z\right)}{2\Delta y} \tag{4.59}$$

$$\left.\frac{\partial^2 u}{\partial y^2}\right|_{x,y,z} = \frac{u\left(x,y+\Delta y,z\right)-2u\left(x,y,z\right)+u\left(x,y-\Delta y,z\right)}{\Delta y^2} \tag{4.60}$$

$$\left.\frac{\partial u}{\partial z}\right|_{x,y,z} = \frac{u\left(x,y,z+\Delta z\right)-u\left(x,y,z-\Delta z\right)}{2\Delta z} \tag{4.61}$$

$$\left.\frac{\partial^2 u}{\partial z^2}\right|_{x,y,z} = \frac{u\left(x,y,z+\Delta z\right)-2u\left(x,y,z\right)+u\left(x,y,z-\Delta z\right)}{\Delta z^2} \tag{4.62}$$

$$\begin{aligned}\left.\frac{\partial^2 u}{\partial x\partial y}\right|_{x,y,z} &= \frac{u\left(x+\Delta x,y+\Delta y,z\right)}{4\Delta x\Delta y}-\frac{u\left(x+\Delta x,y-\Delta y,z\right)}{4\Delta x\Delta y}+\cdots\\ &\frac{u\left(x-\Delta x,y+\Delta y,z\right)}{4\Delta x\Delta y}-\frac{u\left(x-\Delta x,y-\Delta y,z\right)}{4\Delta x\Delta y}\end{aligned} \tag{4.63}$$

$$\begin{aligned}\left.\frac{\partial^2 u}{\partial x\partial z}\right|_{x,y,z} &= \frac{u\left(x+\Delta x,y,z+\Delta z\right)}{4\Delta x\Delta z}-\frac{u\left(x+\Delta x,y,z-\Delta z\right)}{4\Delta x\Delta z}+\cdots\\ &\frac{u\left(x-\Delta x,y,z+\Delta z\right)}{4\Delta x\Delta z}-\frac{u\left(x-\Delta x,y,z-\Delta z\right)}{4\Delta x\Delta z}\end{aligned} \tag{4.64}$$

$$\left.\frac{\partial^2 u}{\partial y \partial z}\right|_{x,y,z} = \frac{u\left(x, y+\Delta y, z+\Delta z\right)}{4\Delta y \Delta z} - \frac{u\left(x, y+\Delta y, z-\Delta z\right)}{4\Delta y \Delta z} + \cdots \\ \frac{u\left(x, y-\Delta y, z+\Delta z\right)}{4\Delta y \Delta z} - \frac{u\left(x, y-\Delta y, z-\Delta z\right)}{4\Delta y \Delta z} \tag{4.65}$$

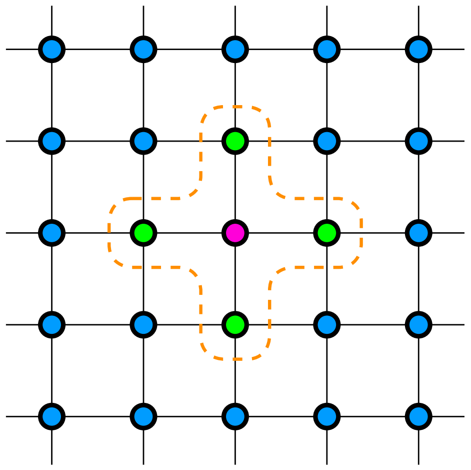
Note that all of the above listed approximations are second order accurate with respect to the spacing in the direction of the derivative. In addition, it is easily shown that Eqs. (4.57)-(4.65) may all be placed into the standard form of

$$\partial u\left(\mathbf{x}\right) = \partial \underline{\Phi}\left(\mathbf{x}\right) \underline{u} \tag{4.66}$$

where the derivative shape function vector $\partial\underline{\Phi}$ consists of the leading coefficients multiplying each respective nodal value.

There are several benefits of formulating the problem using Eqs. (4.57)-(4.65) in a traditional finite difference fashion. First, the finite difference method is one of the oldest PDE solution techniques and, as such, it has a large knowledge base pertaining to optimizations in terms of solution accuracy and speed. Second, because the formulations are derived from the Taylor series representation, they have predictable error, and many techniques have been developed to utilize this error within the solution process. Third, because of the prevalence of finite differencing methods, there have been many proposed approaches for handling convective derivatives where upwinding is necessary, a common source of concern for more traditional meshless techniques (such as diffuse MLS or pure RBFP).

Despite these benefits, there is one obvious limitation of pure finite differencing. If the underlying point distribution is regular, such as the one shown in Figure 4.4a, then finite differencing may be applied without any concerns. However, as soon as the point distribution is irregular, such as in Figure 4.4b where there is a necessary node missing at


(a) Structured

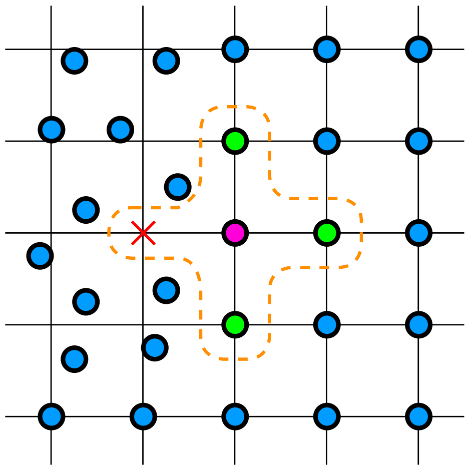

(b) Unstructured

Figure 4.4: Sample Point Distributions

the red x for the orange topology, then the derivative operators of Eqs. (4.57)-(4.65), which rely on a structured set of data, can no longer be applied. This limitation prevents pure finite differencing from being applied to arbitrary geometries since almost any real-world model will have boundary faces that are not grid aligned (and therefore, will introduce irregularities into the point distribution). However, most point distribution methods are based on Cartesian alignment and even in highly irregular geometry there are bound to be regions exhibiting local structure (such as the right side of the point distribution in Figure 4.4b). For this reason, it is beneficial to take advantage of those areas with local structure and directly apply finite differencing using Eqs. (4.57)-(4.65). Further justifying the use of pure finite differencing in areas of local structure is the fact that RBFP, when applied to a structured topology with the proper configuration, will generate the same shape functions that are obtained via Eqs. (4.57)-(4.65), though at a much higher computational expense (due to the additional operations necessary to build and invert the respective moment matrices).

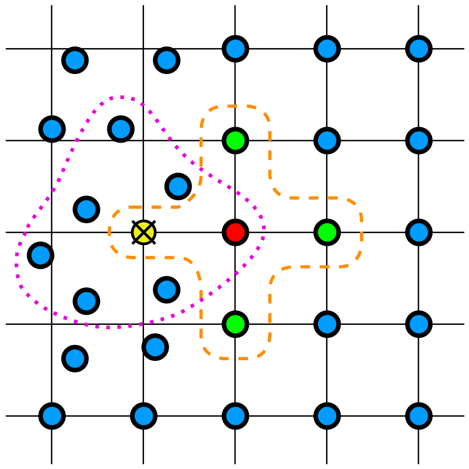


Figure 4.5: Virtual Topology

To apply the finite differencing equations shown in Eqs. (4.57)-(4.65) to an unstructured region such as the one shown in Figure 4.4b, some means of approximating the missing node values must be developed. Fortunately, such methods have already been described in the previous sections; MLS and RBFP approximations may be applied to determine the missing nodal values. Once these have been obtained, it becomes trivial to directly apply the finite difference equations to represent the underlying derivatives. Thus, the process of obtaining the shape functions via VML begins by constructing a virtual node at the required locations of missing structure and building a local topology about that position (as shown in Figure 4.5 as the yellow node and the purple dotted region, respectively). Once the virtual topology has been determined, any of the available interpolation schemes (MLS or RBFP) may be utilized to determine the value at the desired location. This interpolation will be of the form:

$$u(\mathbf{x}) = \underline{\Phi}(\mathbf{x})\,\underline{u} \tag{4.67}$$

where $u(\mathbf{x})$ represents the unknown value at the virtual node.

Once the value of the virtual node has been determined, Eqs. (4.57)-(4.65) may be directly applied over the given domain. However, rather than simply use the interpolators to determine the value at each virtual node, they may be integrated into the shape functions provided by finite differencing to produce a compact form which adds no additional overhead to the solution process. For example, to develop the second order, second derivative operator given in Eq. (4.58) for the red node (located at coordinates x, y, z) shown in Figure 4.5, the unknown quantity at the virtual node located at $(x - \Delta x, y, z)$ is required. Thus, an interpolation is used to obtain the value as:

$$u\left(x - \Delta x, y, z\right) = \underline{\Phi}\left(x - \Delta x, y, z\right)\underline{u} \tag{4.68}$$

which may be substituted into Eq. (4.58) which results in:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x,y,z} = \frac{u\left(x + \Delta x, y, z\right) - 2u\left(x, y, z\right) + \underline{\Phi}\left(x - \Delta x, y, z\right)\underline{u}}{\Delta x^2} \tag{4.69}$$

By realizing that the existing weights can be combined with the finite difference weights, and that the virtual topology nodal vectors can be appended to one another (making sure to combine duplicate nodes and their associated weights), Eq. (4.69) may be rewritten in terms of shape functions as:

$$\frac{\partial^2 u\left(\mathbf{x}\right)}{\partial x^2} = \left\{\begin{array}{ccc} \frac{1}{\Delta x^2}\underline{\Phi}\left(x - \Delta x, y, z\right) & -\frac{2}{\Delta x^2} & \frac{1}{\Delta x^2} \end{array}\right\}\left\{\begin{array}{c} \underline{u} \\ \\ u\left(x, y, z\right) \\ \\ u\left(x + \Delta x, y, z\right) \end{array}\right\} = \underline{\Phi}\,\underline{u} \tag{4.70}$$

thus resulting in a similar form which may be integrated with the direct differentiation schemes shown in the previous sections.

In summary, VML is more than a simple generalized finite difference technique, encapsulating two principles regarding the use of finite difference within the context of meshless solvers. The first principle states that, where possible, pure finite differencing should be utilized to obtain the necessary derivative operators. The second principle states that in areas with some local structure, it is beneficial to use the existing information that is present (such as the value at $u\left(x+\Delta x, y, z\right)$ in Figure 4.5) to provide greater accuracy than would be possible through direct differentiation of an approximation or interpolation based approach (such as MLS or RBFP). Another unique property of VML operators is that they may utilize any field approximation technique to obtain the value at the virtual nodes. In this regards, MLS may be chosen to produce a stable solution (due to its ability to smooth the underlying fields), or RBFP may be chosen to provide higher accuracy. It should be noted that in practice, MLS oftentimes proves to be the best approximation for use with VML principles. This has to do with the fact that RBFP has a tendency to oscillate between nodes, and while the value and its derivatives at the data center may be accurate, a numerical derivative based on intermediate values tends to contain considerable error in complex field situations. That being said, MLS, because of its capability of smoothing field oscillations, provides an ideal balance of accuracy (since the accuracy can be made to match that of the underlying finite difference approximation) and stability for this approach. Although either technique will result in a valid solution process, MLS will be chosen as the interpolation method of choice on the grounds of stability and robustness.

4.4 Chapter Summary

To summarize, the MIMS method will utilize a combination of moving least squares, radial basis function interpolation, and virtual finite differencing to produce the field and derivative operators needed throughout the solution process. It is important to acknowledge that this is a major departure from traditional thinking whereby each researcher has his or her interpolation scheme of choice (as evidenced by the various meshless histories presented in Chapter 2) and very little blending of techniques occurred. This represents a major shortcoming in the current state-of-the-art as each shape function technique has their own relative strengths and weaknesses. It makes sense, therefore, to not restrict oneself to a single shape function generation scheme and to instead selectively choose the best technique on a local, node-by-node basis. Furthermore, although moving least squares and radial basis function interpolation have both been extensively studied by existing meshless research, virtual finite differencing has been developed specifically for this method to supplement existing techniques where they lack robustness (e.g. upwinding, one sided derivatives, boundary layer tangential derivatives, etc.). Despite being a relatively new development by the author, results utilizing this technique have been published at several international conferences [101, 102, 125], with additional illustrative examples shown in Chapter 9.

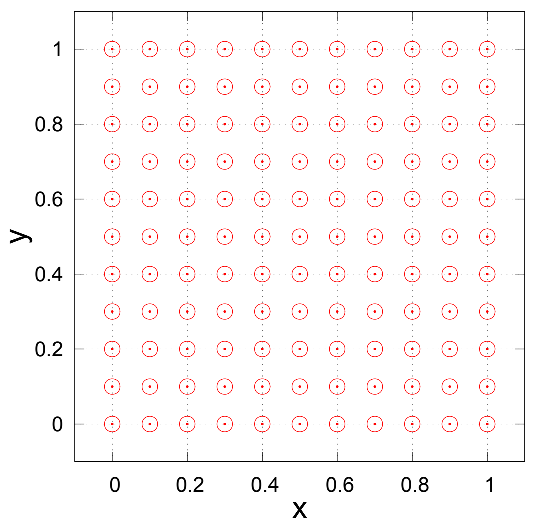
CHAPTER 5
MESHLESS IMPLEMENTATION DETAILS

The previous chapter described the meshless discretization process utilizing various shape function generation techniques, however, little was said regarding practical implementation details. Issues such as stability, optimal support domain construction, and shape parameter optimization were omitted for the sake of generality. Unfortunately, these implementation details can dramatically affect the stability and robustness of a particular meshless method implementation. Despite the importance of these issues, discussions in current literature are mostly limited to small scale problems and are generally addressed in two-dimensions only. This represents a major deficiency in the current study of practical meshless methods, and as such, the following chapter will be devoted to addressing some of the more practical concerns with implementing a robust meshless method.

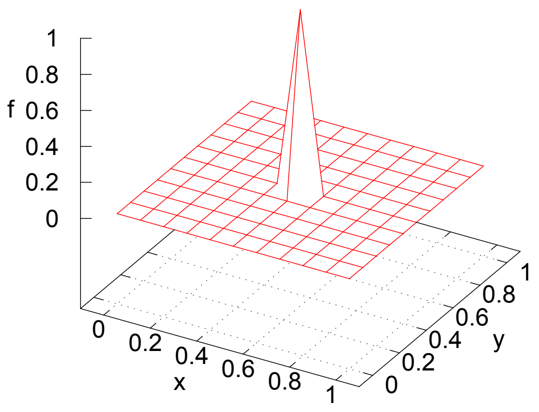
5.1 Numerical Stability Analysis

Performing an analysis of stability is a critical step in developing a robust PDE solution technique as it gives justification and guidance for selecting important quantities such as the minimum time step and maximum discretization size. Historically, stability analysis has been performed using one of three techniques, the discrete perturbation method, the von Neumann method, or the matrix method. The overall goal of these techniques is to determine the response of the discretized PDE to errors introduced during the solution process. If the errors are damped over the domain, then the process is generally considered stable. On the other hand, if the errors are amplified, then the process will have a tendency to fail to converge and may be classified as unstable. In general, the stability limit (the point at which the discretized PDE begins to amplify errors) is dependent on the underlying discretization, time step, and most importantly, the process by which the continuous PDE representing the governing equations are formulated into an algebraic system. For processes such as structured finite differencing, it is possible to analytically determine the stability limit by using one of the three stability analysis techniques previously mentioned. However, within the context of meshless methods, the situation becomes much more complex as the characteristics of the support domain become a potential source of variation in the stability limit. Furthermore, even the most simplistic meshless formulation process will result in a set of PDE update equations relying on an inverted matrix, and, as such, it becomes nearly impossible to develop analytical representations for the stability limit without imposing arbitrary bounds on the algebraic models. Even if one is able to overcome these obstacles, it is difficult to arrive at general stability statements which are applicable across all topological configurations [126, 127], and more importantly applicable to practical numerical algorithms. In spite of this, the concept of analyzing the stability of meshless methods would benefit our understanding of these techniques, and as such, it is worth exploring alternative methods of determining stability behavior.

One alternative technique is to experimentally establish stability behavior by numerically simulating the discrete perturbation method. When experimentally examining stability of a method, it is important to eliminate as many variables as possible since the goal is to determine the behavior of the underlying discretized PDE system (which should, in theory,



(a) Point Distribution



(b) Initial Condition

Figure 5.1: Stability Analysis

be independent of the problem domain). However, in meshless methods the problem domain is part of the discretization process (via the unstructured topologies), and as such, some finite simulated space must be defined. The space chosen for this study is the point distribution shown in Figure 5.1a, where 121 nodes have been evenly distributed over a square, two-dimensional domain of size 1×1, resulting in an average node spacing of $s = 0.1$.

The process of testing the stability limit via an experimental discrete perturbation method involves initializing the entire field to a uniform solution of zero, and then perturbing the central node by some error ϵ. The solution is then allowed to progress from this point while attempting to solve a particular governing equation. If, after some number of iterations the initial perturbation has dissipated, then the update process has controlled the error. On the other hand, if the initial perturbation has not dissipated after a specified number of iterations, then the update process has potentially amplified the error and could indicate instability. To simulate this process using meshless methods, the governing equation applied to the domain is the two-dimensional Laplace equation of the form

$$\nabla^2 f = 0 \tag{5.1}$$

with uniform, constant boundary conditions of

$$f = 0 \quad \text{for all } \Gamma \tag{5.2}$$

By imposing Dirichlet conditions on the boundary, the exterior nodes are able to reflect waves back into the interior of the domain, thus requiring that the update equations damp the initial perturbation, not the boundary conditions. Having defined the governing equations and boundary conditions, the shape function construction techniques of the previous chapter are used to discretize Eq. (5.1) into a system of equations of the form

$$Ax = b \tag{5.3}$$

where A consists of the shape function weights provided by the particular meshless method chosen, and b consists of all zeros (due to the boundary conditions and Laplace equation). To determine stability at this point, the system of equations is solved using a relaxed Jacobi iteration scheme, such that

$$x^{(k+1)} = (1-\omega)\, x^{(k)} + \omega D^{-1}\left(b - Rx^{(k)}\right) \tag{5.4}$$

where ω is the relaxation parameter and D and R are the diagonal and remainder component of A when decomposed as

$$A = D + R \tag{5.5}$$

Note that when $\omega < 1$, the resulting update equation is under-relaxed, while when $\omega > 1$, the resulting update equation is over-relaxed. As originally stated, the goal of performing a stability analysis is to determine the range of values which produce a stable system. Applying this to the described numerical testing process, this amounts to determining the value of ω which causes a particular PDE discretization technique to damp the solution. If the solution converges regardless of the value of ω (up to a maximum value of 2) than the solution process is unconditionally stable. If the solution diverges regardless of the value of ω, then the solution process is unconditionally unstable. Most likely, however, the solution will only converge for values of ω less than some critical threshold ω_s, which defines the stability limit for that particular configuration (a configuration indicating a particular time step and local discretization). The difficulty in determining ω_s is that for the case of both RBFP direct differentiation and Virtual Finite Differencing, there are additional parameters that must be determined *a priori* (the shape parameter c, and the virtual node spacing Δs, respectively) for each configuration. To account for these, and to determine their effects on the stability of the system, we wish to find the relationship between ω_s and the respective parameter for each meshless technique. This amounts to finding the roots of the following equation

$$\chi\left(\underline{\lambda},\omega\right) = \max\left[\lim_{k\to\infty}\left(\left(1-\omega\right)x^{(k)} + \omega D^{-1}\left(\underline{\lambda}\right)\left(b - R\left(\underline{\lambda}\right)x^{(k)}\right)\right)\right] - \epsilon \tag{5.6}$$

where $\underline{\lambda}$ is a specified set of geometric constants for the particular shape function technique, and $\max\left(\underline{v}\right)$ returns the maximum value within the $\underline{v}$ vector. Equation (5.6) is essentially fixing a set of input parameters $\underline{\lambda}$, iterating the perturbed initial field until steady state is reached ($k \to \infty$), and comparing the maximum value in the solution field to the initial perturbation value ϵ. In this manner, the value of ω which makes $\chi = 0$, can be thought of as the stability limit ω_s for a particular input parameter set $\underline{\lambda}$. Therefore, the process of determining the relationship between $\underline{\lambda}$ and ω_s can be summarized in the following steps:

1. Choose a set of input parameters $\underline{\lambda}_i$ (contains c for RBFP and Δs for VML)
2. Initialize the domain and build $D^{-1}\left(\underline{\lambda}_i\right)$ and $R\left(\underline{\lambda}_i\right)$ matrices of Eq. (5.6)
3. Solve $\chi\left(\underline{\lambda}_i, \omega_i\right) = 0$ for ω_i

In practice the limit in Eq. (5.6) can not be evaluated analytically, and thus, a set number of iterations must be performed during evaluation (for these results, $k_{max} = 1000$).

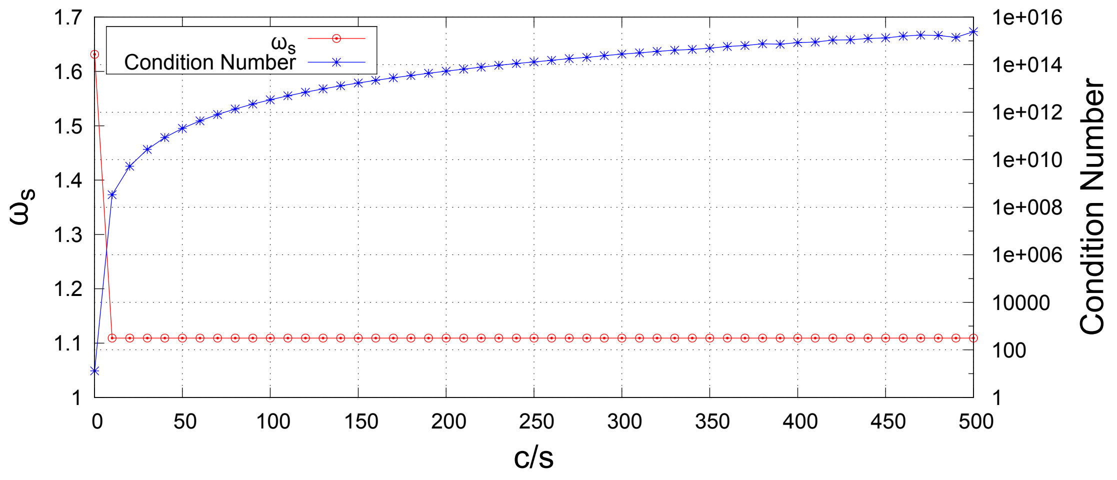
As described in the previous chapter, the three meshless shape functions that are of interest are Moving Least Squares (MLS), Radial Basis Function with Polynomial Reproduction (RBFP), and Virtual Finite Differencing with MLS approximation (VML). In the case of MLS direct differentiation, the only input parameter is the size of the support domain, and, as this is not a continuous variable, it will not be included in the input parameter vector $\underline{\lambda}_{MLS}$. For RBFP direct differentiation, the input parameters are the size of the support domain, and value of the so-called shape parameter c; once again, the size of the support domain will be handled via different test cases, and thus, $\underline{\lambda}_{RBFP} = \{c\}$. Finally, for VML with MLS approximation, the input parameters may include the size of the support domain (once again omitted) and the spacing of the virtual nodes used in the finite difference stencil Δs , thus $\underline{\lambda}_{VML} = \{\Delta s\}$. To build appropriate support domains for this simple region, a routine was developed to select the closest n nodes to the point of interest, with any equally spaced clusters being adding in unison (i.e. if $n = 5$ but the 5th and 6th nodes are equally spaced from the support center, then both nodes will be included in the topology). For a structured point distribution, there are three values of n that provide varying internal spacings (5, 9, and 13) and as such these will be the primary topology sizes that will be examined.

The first analysis was performed on MLS direct differentiation, and, as the input parameter space is null, there is no relationship to be made between ω_s and $\underline{\lambda}$. Therefore, the single result will indicate whether or not MLS was able to successfully damp the initial perturbation on a specified support domain size. After running the experiment on all three topology sizes, none of the resulting MLS systems were able to successfully damp the initial perturbation (all systems reported $\omega = 0$ as the only solution). Although this result is important, it is not altogether surprising since it has already been stated how it is a well known issue of MLS. Because MLS approximations are not interpolations (i.e. the Kronecker delta is not satisfied, $u_{MLS}(\mathbf{x}_i) \neq u_i$) they are unable to properly impose Dirichlet boundary conditions on the boundary and, as such, boundary conditions of this type are most commonly imposed via penalty functions [29, 106]. In fact, generalizing this statement, MLS approximations are mutually exclusive from collocation techniques since they are unable to explicitly satisfy nodal values at the collocation locations. Understanding this fact, it is clear that MLS direct differentiation is ill-suited to a collocation based meshless approach. Nonetheless, as has already been stated, it is still a useful tool for post-processing, as well as for obtaining quantities that are imposed as boundary conditions to other problems (such as in incompressible flow pressure-correction schemes). For this reason direct differentiation of the MLS should only be used when computing quantities that do not require the Kronecker delta property for stability.

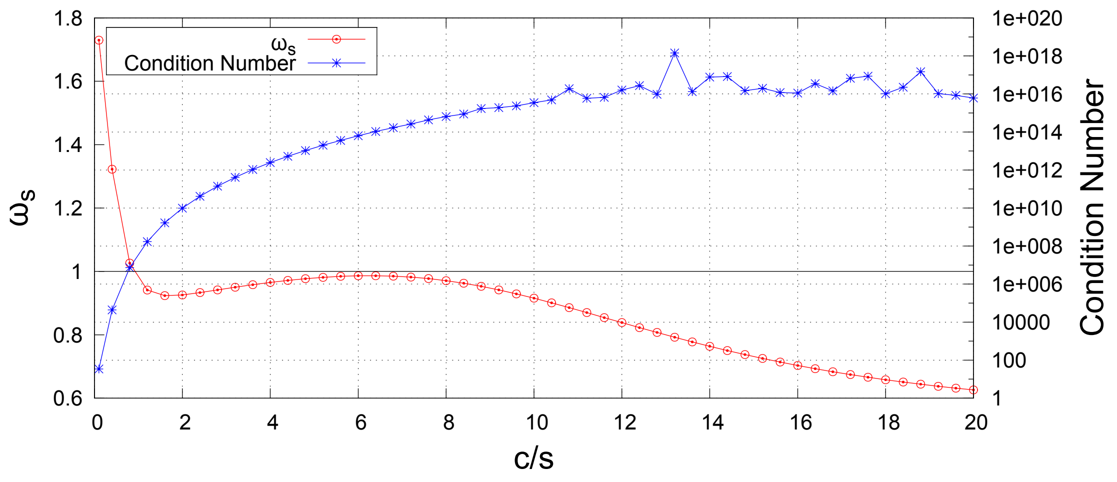
The second analysis was performed on RBF direct differentiation, and, like MLS, the results were obtained for the three topology sizes indicated previously. However, unlike

MLS, the RBF collocation matrix requires a shape parameter c to be defined for each node (for evaluation of Eq. (4.28)). Traditionally this c value has been based on the local mesh size [106] or optimized to obtain a specified condition number for the moment matrix [96], however, to the best of my knowledge, these values have been based primarily on previous experience and rule of thumb heuristics. Therefore, one of the primary goals of this analysis is to provide numerical justification for the selection of c in a practical setting. After performing the analysis, the results for the three topology sizes may be seen in Figures 5.2a, 5.2b, and 5.2c. Note that in each of these figures the dependent axis has been normalized with respect to the domain node spacing s.

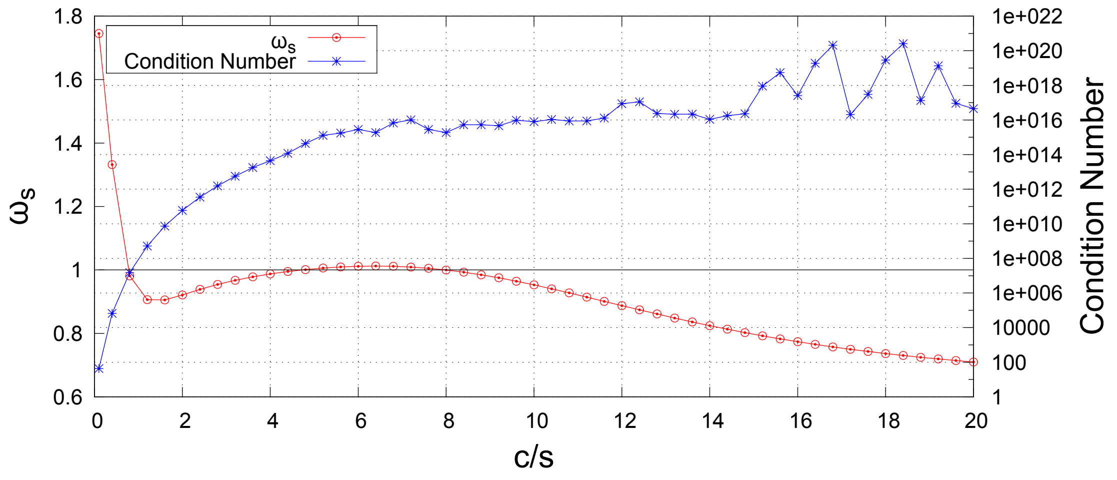
Examining these results, the first thing to note is that with a structured support domain of size 5, the RBF is unconditionally stable (up to $\omega \approx 1.11$) within the range of c values tested. Though notable, this is not unexpected as it can be shown that given 5 nodes in a structured, two-dimensional domain, the Laplace derivative of the RBF shape functions will provide the same weights (though they may be scaled, depending on c) as finite differencing over the same set of nodes. More useful are the results shown in Figures 5.2b and 5.2c, which illustrate the results for the support domains with $n = 9$ and $n = 13$, respectively. First, it is important to note that for the majority of typical c values (in the range of $2s$-$10s$), the RBF iteration process requires under-relaxation in order to damp the initial perturbation. Compare that with finite differencing which is unconditionally stable (for all $\omega \leq 1$) when applied to this governing equation, and this result becomes significant (as a personal note, this issue has been encountered several times by myself and fellow colleagues when direct RBF fails to converge under some topological configurations without appropriate



(a) Structured Domain $n = 5$



(b) Structured Domain $n = 9$



(c) Structured Domain $n = 13$

Figure 5.2: RBF Normalized Shape Parameter vs ω_s

under-relaxation). Given this knowledge, it would be ideal if it were possible to infer the best range of c values to allow for maximum stability from these results. Unfortunately, even over this small variation in problem setup, there is a significant difference in the optimal c value with respect to grid spacing. However, by plotting the condition number with respect to the stability limit (as shown in Figure 5.3), we can see that there is a clear relationship between the stability limit and the condition number of the RBF moment matrix. In fact, these results give clear justification for the use of an optimization strategy targeting a specified condition number. These results suggest an optimal range between 1×10^{11} and 1×10^{13} as ideal for stability, which is slightly higher than the typically quoted range of 1×10^{10} to 1×10^{12}. The results shown in Figure 5.3 imply that, in terms of stability, it is more appropriate to base the value of c on the condition number of the RBF moment matrix, as opposed to any locally defined average nodal spacing. This, coupled with the uncertainty relation formally described by Schaback [75], provides justification for optimizing c on a node-by-node basis in an attempt to achieve a specified range of condition numbers.

The final meshless shape function utilized in the MIMS method is the Virtual Finite Differencing scheme with an underlying MLS approximation. MLS approximations are optimal for VML because of their ability to smooth oscillations present in the field. When performing VML, the input properties include the virtual node spacing Δs, and as such, this study seeks to determine the stability limit as a function of this value. Performing the same study using VML shape functions, it can be seen from the results shown in Figure 5.4 that there is a range of optimal values (roughly between $0.8 \leq \Delta s/s \leq 1.6$) over which the system is able to damp the initial perturbation with no under-relaxation (in fact, over-relaxation

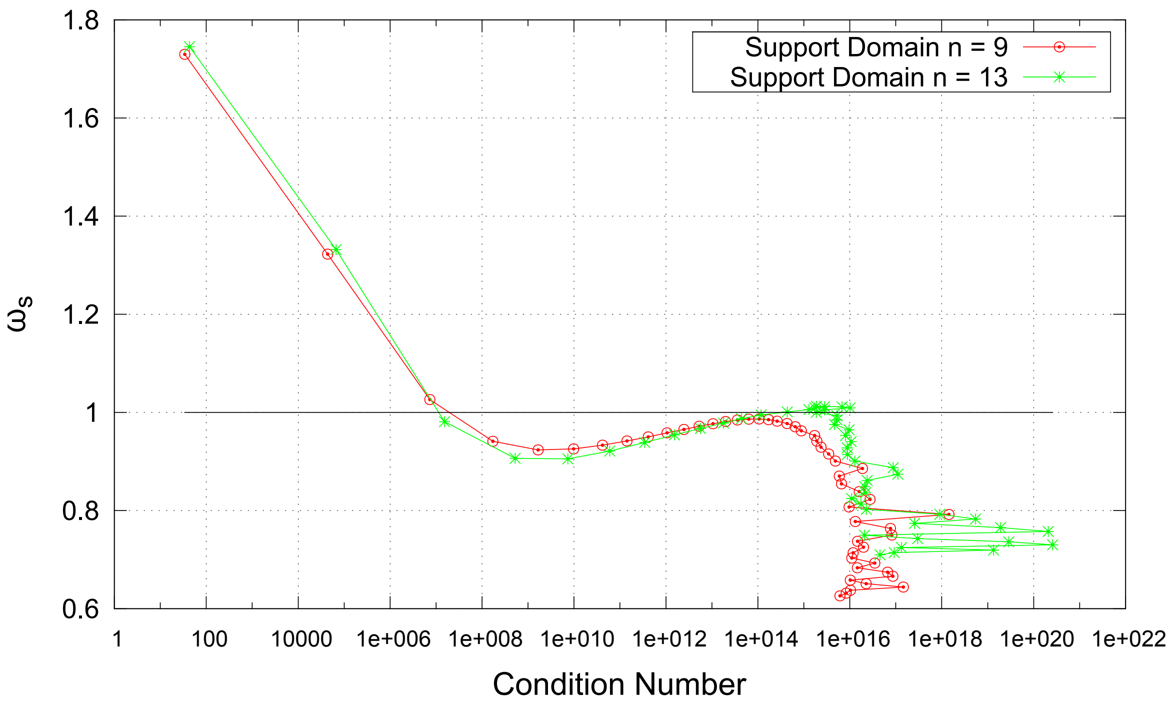


Figure 5.3: RBF Condition Number vs ω_s

values as high as $\omega = 1.5$ are possible in some cases). Fortunately, this optimal range for virtual spacing is quite easy to implement as it involves placing the virtual node between 1 and 1.2 times the average nodal spacing within the node's support domain. It should be noted that, as opposed to the preprocessing required to optimize c for RBFP, this optimal value requires little extra computational effort to obtain.

As a final study, the nodes of the structured region were moved by a random offset of $\pm 10\%$ of the nodal spacing, thus producing the point distribution shown in Figure 5.5. This was done to verify that these results extend to the case of an unstructured region, as well as to ensure that the RBFP results were not caused by the interpolation taking advantage of the structured nature of the nodes. The original experiments were reproduced for both RBFP and VML and are shown in Figures 5.6 and 5.7, respectively (along side the original

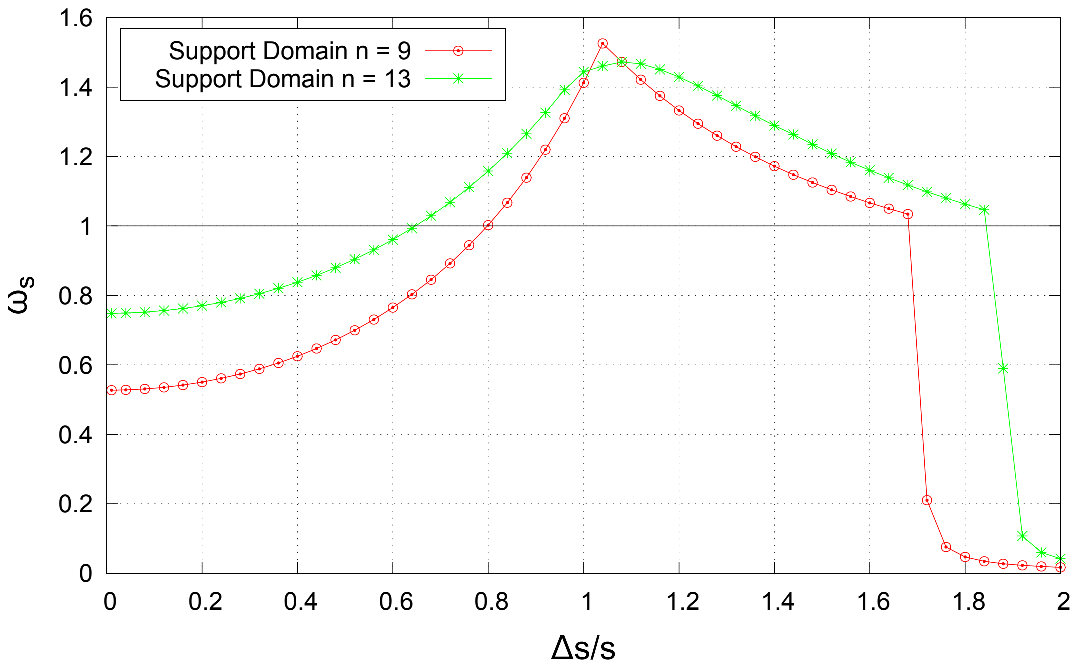


Figure 5.4: VML Normalized Virtual Spacing vs ω_s

data). In both cases, the unstructured results follow the same general trends of the previous results, validating that the process is (at least somewhat) independent of the local node distribution.

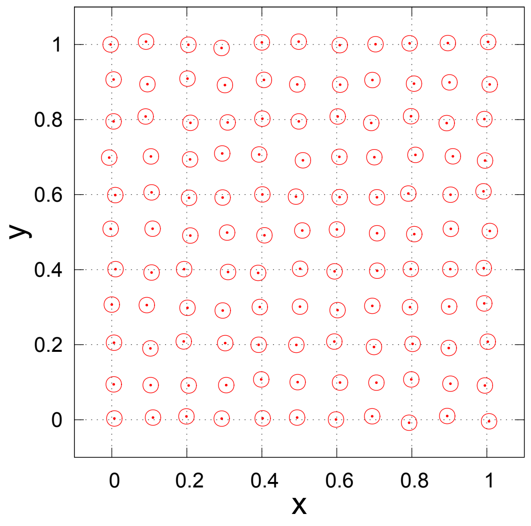


Figure 5.5: Randomized Point Distribution

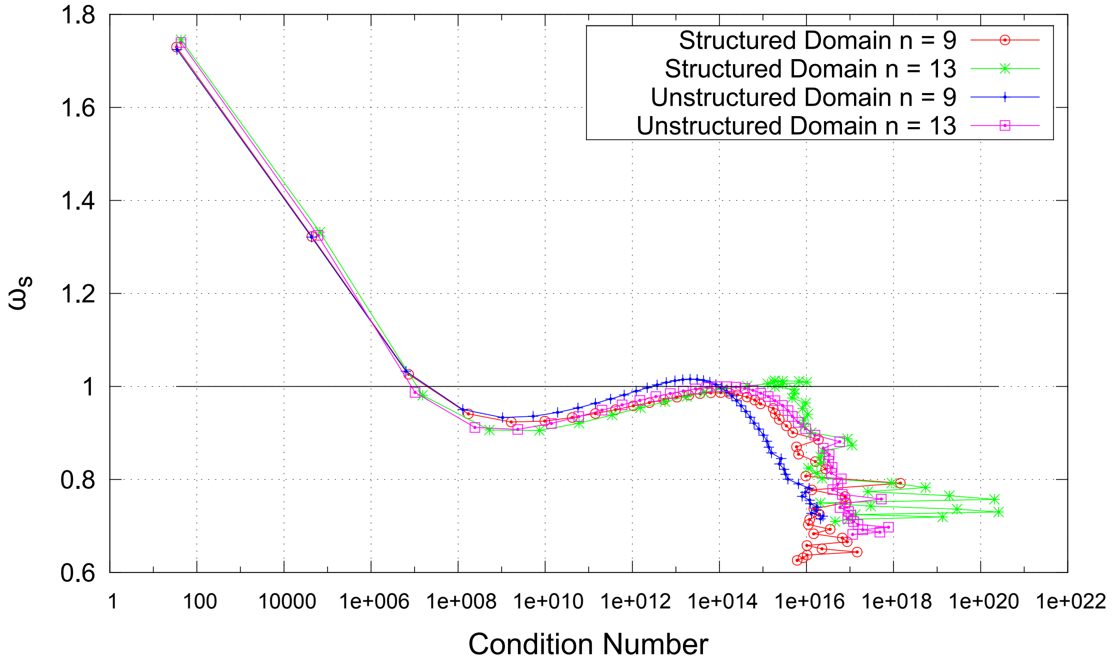


Figure 5.6: RBF Condition Number vs ω_s

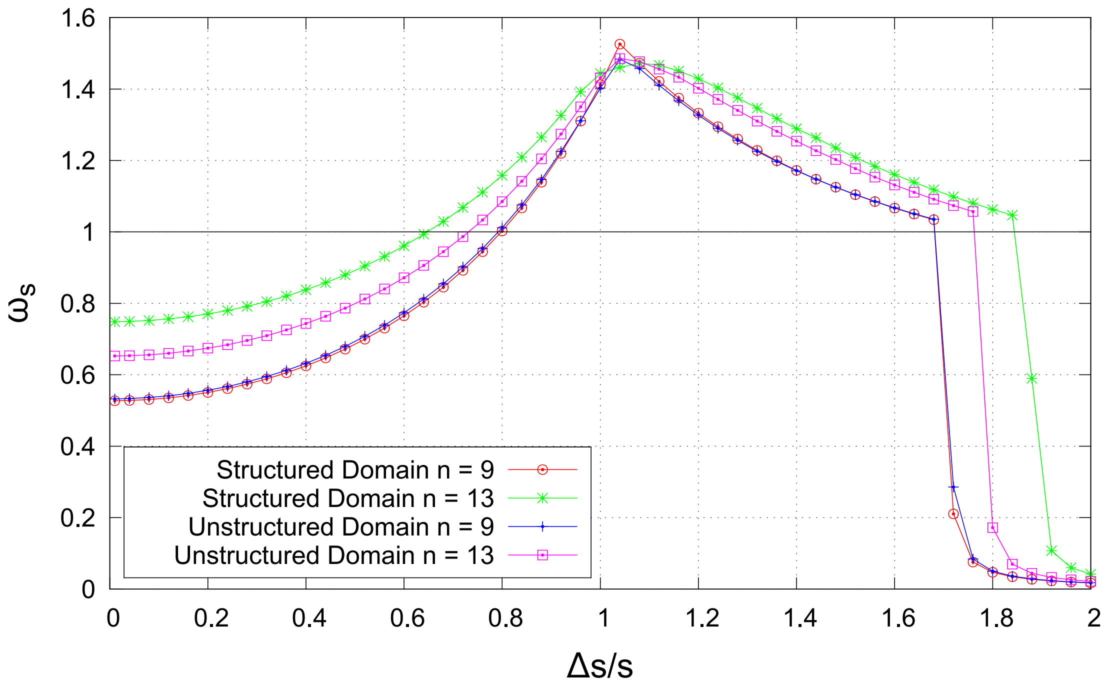


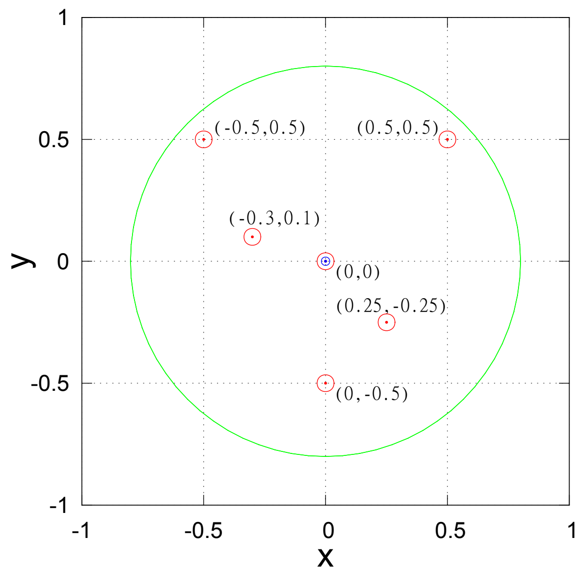
Figure 5.7: Unstructured VML Normalized Virtual Spacing vs ω_s

On a final note, the above listed experiments were all repeated over several different node counts (to vary s) and no significant difference was found between the presented data. This suggests that, like finite difference, the stability behavior of RBFP and VML shape functions are independent of grid spacing when solving the steady-state Laplace equation.

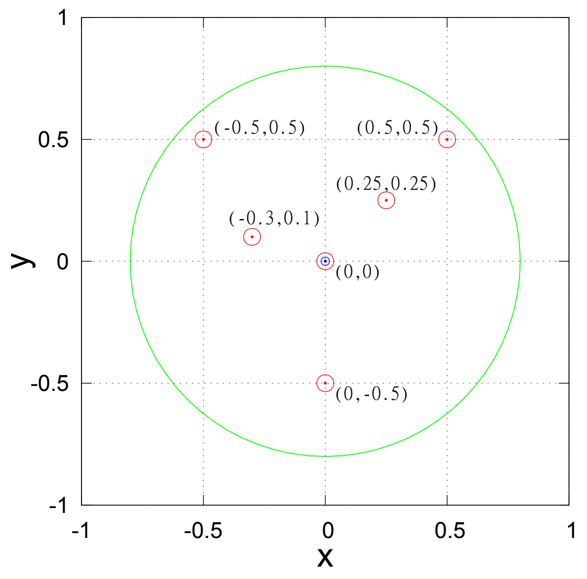
In summary, these results indicate that in order to develop stable shape functions using RBFP direct differentiation, one should optimize the value of c to obtain a local condition number between 1×10^{11} and 1×10^{13}. In addition, under-relaxation should be employed when solving the resulting system of equations in order to ensure convergence. As for VML, the virtual node spacing should be placed between 1 and 1.2 times the average nodal spacing of the local support domain in order to maximize stability. From these results it appears as though VML produces a more stable system than RBFP, and as such, under-relaxation will generally not be required for convergence.

5.2 Support Domain Construction

No topic in Meshless methods is as widely overlooked as the process of generating the necessary support domains for a given geometry and interpolation technique. While there is plenty of research describing why one interpolation technique is more accurate than another, few papers acknowledge the importance of developing proper support domains; and fewer still describe processes that can consistently determine appropriate influence regions for a distribution of unstructured nodes. Rather than describe the problem in words, it seems most appropriate to illustrate the importance of proper support domain selection through a simple, yet practical, example.



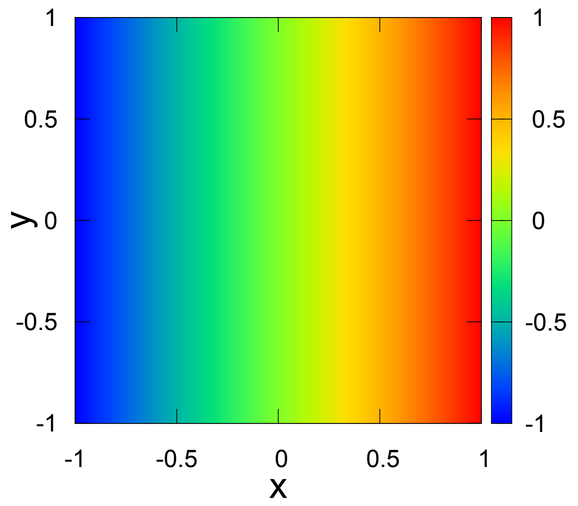
(a) Influence Domain 1



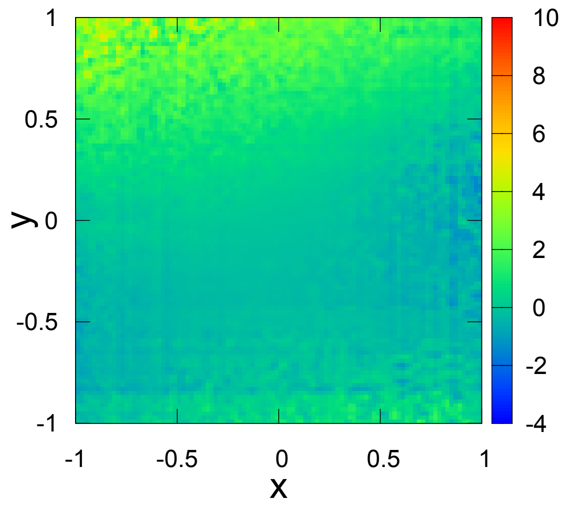
(b) Influence Domain 2

Figure 5.8: Two-Dimensional Influence Domains

Figure 5.8 presents two similar local node distributions, where the highlighted node at position $(0, 0)$ is the point at which a support domain is desired. For the sake of this example, the Virtual Finite Difference shape function technique will be utilized, with



(a) Linear Field over Influence Domain 1



(b) Linear Field over Influence Domain 2

Figure 5.9: Linear Field over Two-Dimensional Influence Domains

the underlying field approximation done via Moving Least Squares with quadratic basis functions (i.e. $\underline{p}$ includes 1, x, y, xy, x^2, and y^2). It is worth mentioning that both of these MLS support domains violate the condition that $n \gg m$, however, for the sake of this example we will overlook this detail. Thus, both support domains contain 6 nodes, with the position of only one node varying from Influence Domain 1 (Figure 5.8a) to Influence Domain 2 (Figure 5.8b). Despite this minor change in topology layout, if a linear field is applied over these two topologies, and the underlying MLS approximation is sampled across the local domain, the two resulting fields are dramatically different. Figure 5.9a illustrates that Influence Domain 1 is able to exactly capture the solution regardless of the sampled location (which is as expected from a quadratic MLS), while Influence Domain 2 (shown in Figure 5.9b) results in an unacceptably poor approximation of the underlying field. When translated to the VML stencil, the field shown in Figure 5.9a will produce accurate estimations of the derivative, while the field shown in Figure 5.9b will not. The purpose of this simple example is to demonstrate that it is not simply the size or shape of the underlying topology that determines whether or not it will produce good approximations. In order for an unstructured approximation method to produce accurate results, it must have a sufficient sampling of *directional* information, which will result in a better conditioned moment matrix, and a smoother approximation. This, therefore, justifies the statement that (for MLS approximations) an acceptable topology should satisfy the criteria $n \gg m$. However, this raises another issue; if one were to simply choose a very large support domain for each node (thereby guaranteeing enough directional information), the solution would tend to be either overly dissipative (in the case of MLS) or overly oscillatory (in the case of

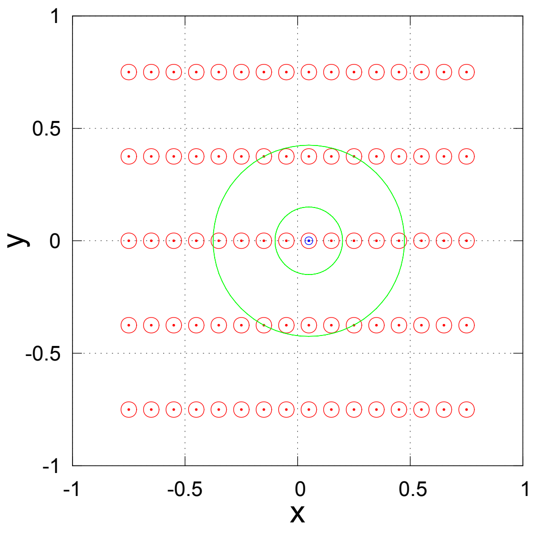


Figure 5.10: Clustered Node Distribution in 2D

RBFP). Therefore, the ideal support domain is the *smallest* collection of nodes containing enough directional information to produce an accurate approximation.

Despite the obvious importance of solution quality on obtaining an appropriate support domain for each node, most meshless implementations simply determine a node's topology based on a local radius defined by the neighboring nodes to the point of interest. Although this may be acceptable for a uniformly random point distribution, as soon as a topological structure is introduced, such as node clustering, simply selecting a support region based on a local radius metric is no longer sufficient. Take, for example, the node distribution shown in Figure 5.10; both the large and small radius influence circles (shown in green) have issues that would deter their use. The small topology will most certainly produce a singular moment matrix due to the lack of information in the y-direction, while the larger topology has the potential of causing over-smoothing (MLS) or over-fitting (RBFP) in the x-direction. Further compounding the issue is the fact that in real-world problem geometries, it is easy to encounter node distributions that are prone to one-sided or ill-posed support domain construction when complex boundaries are involved. The primary issue with a radius based approach is that it is not based on any quantifiable metric that can be definitively shown to represent the quality of the underlying support domain. Although these types of procedures may be acceptable for simple geometries, they will quickly break down when applied to real-world, adaptive point distributions, especially in three-dimensional geometries.

To address this deficiency in current meshless research, a straightforward procedure of systematically selecting the most appropriate, minimal support domain for a particular node has been developed. The process begins by acknowledging that any robust routine must attempt to satisfy the two primary characteristics of an ideal support domain. The first characteristic is that a given set of nodes must include enough *directional* information to generate an appropriate approximation of the underlying field. This characteristic directly relates to the accuracy of the proposed approach and will be the primary metric used when attempting to determine if a support domain is acceptable. The second characteristic is that given two equally sufficient sets of nodes, the one that minimizes the number of nodes is preferred both on the grounds of efficiency, as well as reducing over-smoothing (in the case of MLS) or over-fitting (in the case of RBFP) resulting from excess overlap in the resulting topological map. The difficulty in developing a systematic approach is in quantifying the two characteristics in a manner that can be used to identify how well a particular support domain will behave, or at the very least, when a particular support domain will produce an acceptable result and when it will not. The innovation comes from a behavioral analysis of the MLS moment matrix under various topological configurations. Take, for example, the moment matrices for the configurations shown in Figures 5.8a and 5.8b (generated using Eq.

(4.14) with quadratic basis functions) given by A_1 and A_2, respectively

$$A_1 = \begin{bmatrix} 6.000 & -0.050 & 0.350 & -0.093 & 0.652 & 0.822 \\ -0.050 & 0.652 & -0.093 & 0.243 & -0.011 & 0.013 \\ 0.350 & -0.093 & 0.822 & 0.013 & 0.243 & 0.110 \\ -0.093 & 0.243 & 0.013 & 0.130 & -6.606 \times 10^{-3} & -4.206 \times 10^{-3} \\ 0.652 & -0.011 & 0.243 & -6.606 \times 10^{-3} & 0.137 & 0.130 \\ 0.822 & 0.013 & 0.110 & -4.206 \times 10^{-3} & 0.130 & 0.192 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 6.000 & -0.050 & 0.850 & 0.033 & 0.652 & 0.822 \\ -0.050 & 0.652 & 0.033 & 0.275 & -0.011 & 0.013 \\ 0.850 & 0.033 & 0.822 & 0.013 & 0.275 & 0.142 \\ 0.033 & 0.275 & 0.013 & 0.130 & 1.206 \times 10^{-3} & 3.606 \times 10^{-3} \\ 0.652 & -0.011 & 0.275 & 1.206 \times 10^{-3} & 0.137 & 0.130 \\ 0.822 & 0.013 & 0.142 & 3.606 \times 10^{-3} & 0.130 & 0.192 \end{bmatrix}$$

Although is has already been shown that the first configuration (A_1) represents a more accurate distribution than the second (A_2), it may not be immediately obvious from the geometric configuration alone. To analyze why this might be the case, both moment matrices were decomposed into their singular values, and the condition numbers were determined to be $A_1 = 1.865 \times 10^3$ and $A_2 = 1.967 \times 10^{17}$. Clearly this demonstrates that the MLS moment matrix for the second nodal distribution is nearly singular, resulting in the poor approximations shown in Figure 5.9b. One might be inclined to define a quality metric based on the singularity of the underlying moment matrix by analyzing the condition number, however, there are two concerns with a method of this approach. First, computing the condition number is an expensive calculation, most commonly performed through an implementation of Singular Value Decomposition. Second, although the condition number of the matrix may indicate that there will be inaccuracies of the approximation, it doesn't provide an indication as to how to rectify the issue. Therefore, instead of examining the condition number of the matrix, we may analyze the rank of the moment matrix, and its corresponding reduced-row echelon form (computed via matrix triangulation) in order to determine which of the monomial basis functions (i.e. directions) are lacking information. Computing these values for the current example, it can be found that the rank of A_1 is 6, while the rank of A_2 is 5, indicating that it is rank deficient, and therefore highly singular. Furthermore, the reduced-row echelon form of A_2 is:

$$rref\left(A_2\right) = \begin{bmatrix} 1.000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.000 & 0 & 0 & 0 & 0.500 \\ 0 & 0 & 1.000 & 0 & 0 & -0.500 \\ 0 & 0 & 0 & 1.000 & 0 & -1.000 \\ 0 & 0 & 0 & 0 & 1.000 & 2.000 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

which, as it turns out, can give a rough indication as to the particular basis functions in the underlying MLS moment matrix which are missing directional information. Recall from Eq. (4.14) that the MLS moment matrix is defined as

$$\underline{\underline{A}}\left(\mathbf{x}\right) = \sum_{i=1}^{n} W\left(\mathbf{x}-\mathbf{x}_i\right)\underline{p}\left(\mathbf{x}_i\right)\underline{p}^{\mathrm{T}}\left(\mathbf{x}_i\right)$$

indicating that a particular element of the moment matrix is a summation of the contributions from each node with respect to two basis functions

$$A_{j,k} = \sum_{i=1}^{n} W\left(\mathbf{x} - \mathbf{x}_i\right) p_j\left(\mathbf{x}_i\right) p_k\left(\mathbf{x}_i\right) \tag{5.7}$$

where p_i and p_j are the ith and jth basis functions of the basis function set derived from Pascal's Pyramid. Now recall that the reduced-row echelon form of the matrix provides the linear dependence of one row with another, and as such, if a row consists of all zeros, it can be thought of as singular (it has no unique relationship with any other row, including itself). Similarly, if a column contains a non-zero term, then that particular matrix term is not linearly independent from the corresponding row, and therefore, could also be a source of singularity. Following this logic, we make the following suppositions regarding the reduced-row echelon form of the MLS moment matrix:

Supposition *If a diagonal term A_{ii} of a MLS moment matrix in reduced-row echelon form is zero, then the corresponding monomial basis term p_i has singular tendencies.*

Justification The justification for this statement comes from the fact that the diagonal terms of the moment matrix are, by definition, the scalar product of the ith monomial term (i.e. $x \cdot x$ or $x^2 \cdot x^2$), and as such, in order to be zero, must be not be linearly independent. This implies that there does not exist a unique solution for this particular row, and, since each other element of the row multiplies the ith monomial term, it follows that the most likely monomial missing sufficient information (and thus, having singular tendencies) is the ith term of the basis set.

Supposition *If an off-diagonal term A_{ij} of a MLS moment matrix in reduced-row echelon form is non-zero, then the corresponding monomial basis term p_j has singular tendencies.*

Justification The second supposition employs similar logic to the first; by realizing that each element within the jth column of the moment matrix multiplies a similar term (p_j), in order for term A_{ij} to be non-zero, this monomial must be contributing (in some manner) to the overall non-uniqueness of the row. Therefore, it follows that the most likely candidate missing sufficient information is the jth term of the basis set.

Although based on general inferences, from basic experimental analysis of simple nodal configurations it can be shown that, for most cases, these two suppositions prove accurate. That being said, in order to successfully utilize this information a procedure capable of generating appropriate local support domains must be developed. The solution is to utilize support domains that are ellipsoidal, instead of spherical. In this manner the elliptic domains can grow directionally, according to the information provided by the MLS moment matrix in reduced-row echelon form, to produce appropriate support domains regardless of local node distribution characteristics. Applying this logic to Figure 5.10, might result in a support domain as shown in Figure 5.11, a dramatic improvement on both of the previously offered solutions. Algorithm 5.1 provides a formalization of the support domain generation process where σ is a user-supplied growth factor, which is generally specified to be 10% of the current support domain size (in a particular direction).

At this point it is worth noting that although the listed algorithm is guaranteed to produce a support domain that produces a non-rank deficient MLS moment matrix (if one exists), it may find surprisingly small topologies if the basis order used during the optimiza-

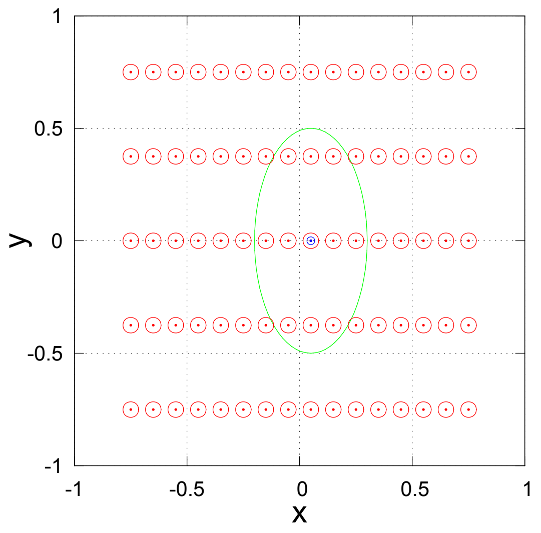


Figure 5.11: Improved Topology in Clustered Node Distribution

Algorithm 5.1 Support Domain Generation Process

```
Input: A desired support center node $n_\ell$ located at $(x_\ell, y_\ell)$.
  Let $\Omega_\ell$ be an initial support domain for $n_\ell$.
  $r_\ell \leftarrow$ rank of MLS moment matrix $A$ for support domain $\Omega_\ell$
  while $r_\ell \neq$ size of $\Omega_\ell$ do
    Mark diagonals $i$ that have singular tendencies.
    Mark columns $j$ that have singular tendencies.
    for each $i$ and $j$ with singular tendencies do
      Grow $\Omega_\ell$ in the appropriate monomial direction(s) by $\sigma$.
    end for
    $r_\ell \leftarrow$ rank of MLS moment matrix $A$ for support domain $\Omega_\ell$
  end while
Output: A size optimal support domain for $n_\ell$.
```

tion process is the same as that of the underlying approximation used during the solution process. Although these pathologically small topologies may not be completely rank deficient (as reported by a numerical algorithm), they oftentimes produce poor approximations due to the high condition number of the associated moment matrix. As a consequence, the general statement that $n \gg m$ can easily be violated via this optimization process. Therefore, there are two additional criteria that are implemented to insure that acceptable topologies will always be produced. First, rather than optimize with respect to the order basis function used during the solution process, it is more appropriate to optimize to one higher order to guarantee acceptable solutions in the lower order space. For example, if one wishes to utilize a complete second order basis function set during the solution process, then the moment matrix used in Algorithm 5.1 should contain the complete third order basis function set. The second criteria imposes a minimum size on the resulting topology such that the number of nodes must be at least double that of the number of basis functions. In practice, this value applies a sufficient criterion on n to guarantee that a support domain which is returned from the optimization process is acceptable not only in terms of the fundamental approximation, but also in terms of the resulting system of equations representing the iteration process. Although the second criteria is rarely encountered in practice (due to the increased order of the optimized moment matrix), it is important to include when developing a robust meshless implementation. When handling this case, the easiest solution is to grow the current ellipse uniformly until enough nodes have been found.

The final consideration is in regards to the applicability of this technique when constructing influence topologies which will be utilized by RBFP shape functions. As it turns out, the same topologies generated using a quadratic MLS support also produce well structured topologies for RBFP interpolations. As such, the same process may be used with confidence to generate optimal support domains for both RBPF and MLS field approximations, regardless of local point distribution characteristics.

5.3 RBF Shape Parameter Optimization

As demonstrated in Section 5.1, the value chosen for the RBF shape parameter c can have a dramatic effect on the overall stability of the discretized system. In addition, it has been repeatedly shown [75, 116, 120, 128] that as the condition number of the RBF moment matrix is increased, the smoothness and accuracy of the interpolated field is improved (a result commonly referred to as the *uncertainty* or *trade-off principle*). Given the importance of choosing an appropriate value for c, it is critical to not overlook the task of specifically optimizing its value on a node-by-node basis. Although there has been considerable research in this area as applied to general RBF interpolations [129, 130, 131] it is difficult to put many of these theoretically bounded techniques into practice (such as the Contour-Padé algorithm [132]) due to the large computational cost required for the general topological situations that arise in three-dimensional space. However, the practical numerical stability results demonstrated in Section 5.1 suggest that optimizing c to target specific a matrix condition number range will satisfy both the uncertainty principle, as well as the stability criteria previously demonstrated. To accomplish this, one must determine the value of c which solves the fitness function

$$f\left(c\right) = \kappa\left(\underline{\underline{A}}\left(c\right)\right) - \tau \tag{5.8}$$

where κ represents the condition number of a matrix, and τ is a target value for the condition number. It was shown through the numerical stability analysis that, at least for the particular test case shown, an optimal value for τ (in terms of stability) is between 1×10^{11} and 1×10^{13}, and indeed, this is the ideal way to impose this condition. This is due to the fact

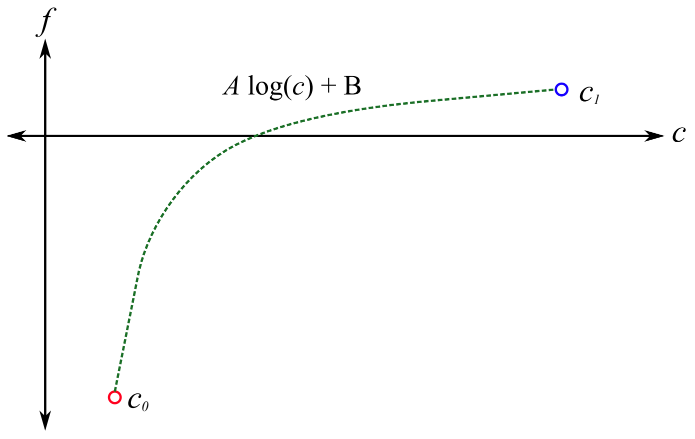


Figure 5.12: Logarithmic Curve Fit

that evaluating $f(c)$ carries a high computational cost due to the calculation of κ (usually computed via singular value decomposition). Although any numerical root finding scheme will solve Eq. (5.8), because of the specific behavior of the condition number with respect to c, it is far more efficient in practice to use a tailored optimization process.

The optimization process employed within the context of this work is a logarithmic approximation routine that takes advantage of the relationship between the condition number and c. Thus, the process starts by defining two points c_0 and c_1 which are known to bracket the root. Then, by fitting a logarithmic approximation of the form $A \log(c) + B$ through the data points (as shown in Figure 5.12) the value for the approximated root may be analytically determined. Once the approximated root is found, the actual fitness function is evaluated and the new point is set as either c_0 or c_1, such that the range $[c_0, c_1]$ brackets the root. In this respect, the process is very similar to the Bisection method, with the exception being that a logarithmic curve fit is utilized instead of linear.

In practice, this procedure converges within an acceptable range of κ in far fewer iterations than a generalized root finding process. As such, it results in considerable savings during preprocessing due to the dramatically reduced number of condition number calculations that need to be performed. Psuedocode for the proposed technique is provided in Algorithm 5.2.

Algorithm 5.2 RBF Shape Parameter Optimization Routine

```
Input: An initial guess bracket $[c_0, c_1]$ and a target range $[\tau_{min}, \tau_{max}]$
  $\tau \leftarrow \frac{\tau_{min}+\tau_{max}}{2}$
  $Q \leftarrow \frac{f(c_0)-f(c_1)}{\log(c_0)-\log(c_1)}$
  $R \leftarrow f(c_1) - Q\log(c_1)$
  $c_2 \leftarrow \exp\left(\frac{-R}{Q}\right)$
  while $\kappa\left(\underline{\underline{A}}(c_2)\right) \leq \tau_{min}$ or $\tau_{max} \leq \kappa\left(\underline{\underline{A}}(c_2)\right)$ do
    if $f(c_0) \cdot f(c_2) > 0$ then
      $c_0 \leftarrow c_2$
    else
      $c_1 \leftarrow c_2$
    end if
    $Q \leftarrow \frac{f(c_0)-f(c_1)}{\log(c_0)-\log(c_1)}$
    $R \leftarrow f(c_1) - Q\log(c_1)$
    $c_2 \leftarrow \exp\left(\frac{-R}{Q}\right)$
  end while
Output: $c_2$
```

At this time, it is worth mentioning an interesting phenomenon that occurs with RBF interpolation which is oftentimes overlooked when attempting to optimize the shape parameter. It is well established that for smooth fields, the uncertainty principle holds in that a higher value for the RBF moment matrix condition number will generally produce a more accurate interpolation. However, when the field is not smooth (as oftentimes occurs in compressible fluid flow due to shocks and expansion waves that represent physical discontinuities in the domain) this statement is no longer appropriate. Take, for example, the one-dimensional discontinuity shown in Figure 5.13 and the various RBF interpolations used to represent the field with increasing condition numbers. Notice that contrary to the commonly accepted

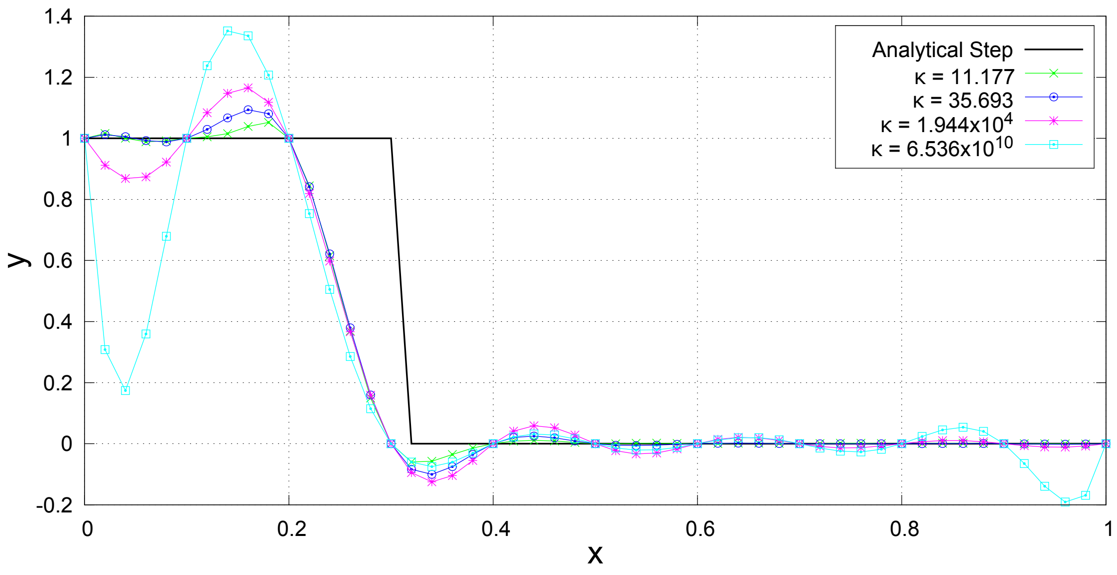


Figure 5.13: RBF Interpolation of Step Function

behavior, when representing this discontinuity the interpolation quality is significantly reduced as the condition number increases. This is easily explained by realizing that the RBF is essentially over-fitting the discontinuity, however, it is an important result as it demonstrates that care must be taken when utilizing RBF interpolations where discontinuities are present. Although not specifically examined within this work (because of the use of VML with MLS in upwinded derivatives), it may be worth exploring procedures that could optimize the shape parameter not only on topological configuration, but also on the local field characteristics. Perhaps, in this manner, a process of determining appropriate c values could be determined that could simultaneously optimize on the basis of both stability and accuracy that could effectively account for discontinuities in the field.

5.4 Chapter Summary

The purpose of this chapter has been to address common meshless implementation details that are oftentimes overlooked by researchers due to the practical nature of the concerns. That being said, the fact that these issues are rarely discussed does not diminish their importance; on the contrary, it emphasizes the current research gap which prevents the application of meshless methods to industrially relevant problems. The particular solutions presented in this chapter have attempted to address the most common concerns when developing an industrially relevant meshless implementation, and, as will be evidenced by the results shown in Chapter 10, they appear to succeed for a large set of problems. It is important to realize that even though meshless methods may succeed in eliminating the need for an underlying mesh, this does not imply that the solutions they generate are independent of the underlying point distribution. The stability and accuracy of any numerical technique will always be dependent on the discretization employed, since, at a fundamental level, one is representing a continuous system over discrete points (or volumes) in space. The goal is to reduce this dependency, and in the case of meshless methods, to facilitate procedures that allow for more intelligent (and automated) discretizations than previously possible.

CHAPTER 6
MESHLESS MODEL GENERATION

The previous two chapters have demonstrated that even in unstructured domains it is possible to obtain field and differential operators at desired locations. However, nodal placement is still of profound importance to the practicality of the interpolation techniques and despite the considerable liberties that the various meshless shape functions allow, there is clearly still some degree of requirement for properly constructed topologies in order to obtain accurate solutions. In addition, there is also the concern of solution time with respect to meshless methods, as the local support domains required by MLS and RBFP can cause the iteration procedure to slow down considerably. Finally, the major reason for developing a specific model generation procedure is to utilize the flexibility of the methods to develop a procedure of automatically discretizing any arbitrarily complex geometry, regardless of relative feature size or aspect ratios. However, it is important to never lose focus of the fact that, at least fundamentally, the meshless collocation method requires no defined structural connectivity. As such, any model generation techniques are free to take liberties in the way point distributions are generated to ensure the most accurate solutions while simultaneously reducing the computational expense wherever possible.

To accomplish these goals, an automated model generation technique has been developed which utilizes three fundamental techniques: quaternary triangular surface discretization, binary-subdivision interior discretization, and an adaptive boundary layer representation referred to as the shadow layer. Together, these three components make up the adaptive model generation procedure of the MIMS process. The following sections will describe each technique in detail and illustrate how their development and integration are critical to implementing an industrially relevant meshless method technique.

6.1 Quaternary Triangular Surface Discretization

The first stage of meshless discretization involves distributing boundary nodes on the surface of the respective solution regions. It is important to begin the discretization process on the boundary (as opposed to the interior), since, at the time of initialization, it is the only section of the domain where solution characteristics can be inferred. For instance, if a surface is designated as a no-slip wall in a fluid flow problem, it can be inferred that there will most likely be high gradients normal to the boundary and relatively low gradients tangential to the boundary. On the other hand, if a surface has a prescribed shear force in an elasticity problem, it will most likely have high gradients in the tangential direction instead. The fact that the applied boundary condition type provides some indication as to the local field characteristics is an important consideration that must be taken into account by a fully automated point distribution method. Understanding the motivation for developing the initial discretization on the boundary, the distribution process is tasked with producing boundary nodes that are (1) evenly distributed on the surface, (2) able to be selectively refined, and (3) have an underlying structure in order to facilitate computation of area-based boundary quantities (such as heat or mass flux). It is important to note that unlike mesh based solution techniques, meshless methods do not require boundary cells to have optimal area properties (such as minimal skew). In fact, the only reason to have any type of boundary connectivity is to facilitate calculation of area-based boundary quantities, and as these are generally post-processed values, the overall solution quality does not depend on cell characteristics. However, if an optimal triangular surface mesh (in terms of the triangulation quality) is provided for a given region, it would certainly be appropriate for meshless methods as a baseline discretization. Understanding this fact, and acknowledging the incredible amount of research that has been performed in the area of optimal surface meshes [133, 134, 135], the underlying data structure of the meshless surface discretization will be triangular in nature.

Having satisfied two of the three requirements of the meshless boundary nodes by using a structure based on triangular elements (evenly distributed on the surface and having an underlying structure), the final requirement (being able to selectively refine) will be satisfied by representing the base triangulation as a recursive *quaternary triangulation* structure. Quaternary triangulation is a recursive triangulation consisting of geometrically fixed splitting rules [136] and although they have been used to generate level-of-detail models for graphical applications [136, 137] as well as geophysical models for cartographic applications [138, 139] they have rarely been utilized to generate computational meshes. This is primarily due to the fact that when dealing with three-dimensional models, mesh-based techniques require a three-dimensional volumetric mesh (tetrahedrals). Therefore, a recursive surface mesh (based on triangles) provides little added benefit as it would necessarily need to be related to a volumetric mesh in order to serve as a base structure for refinement. However, with meshless methods, only a simple point distribution is needed on the boundary, and as such, the quaternary triangulation structure is perfectly suited to the task.

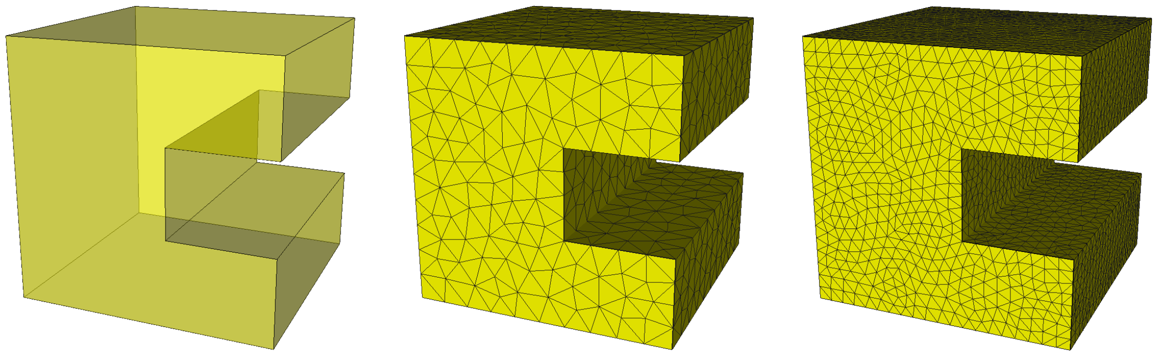

(a) Geometric Model (b) Initial Triangulation (c) Refined Triangulation

Figure 6.1: U-Block Surface Discretization

The quaternary triangular surface discretization process begins by performing a standard triangulation of the underlying surface geometry, such as the one shown in Figure 6.1b. In this figure, the geometric model of the simple U-Block shown in Figure 6.1a has been discretized using a basic surface triangulation and refinement procedure. It is important to reiterate that although high quality surface meshes (as measured using traditional metrics such as skewness, area ratios, etc.) are ideal, they are not completely necessary for the case of meshless methods. However, they will certainly distribute nodes more evenly across the boundary surfaces, and as such, the current implementation applies several two-, and three-dimensional mesh refinement strategies [140, 141, 142] when discretizing complex geometry to help ensure that the initial triangulation is of acceptable quality.

Once an initial surface triangulation has been created, the quaternary triangular structure is initialized from the existing discretization. A quaternary triangular mesh (QTM) is a surface meshing technique utilizing a recursive storage structure whereby each triangular element can store four child elements. Each child element is constructed by splitting each edge of the element at their respective midpoints and connecting the newly created vertices

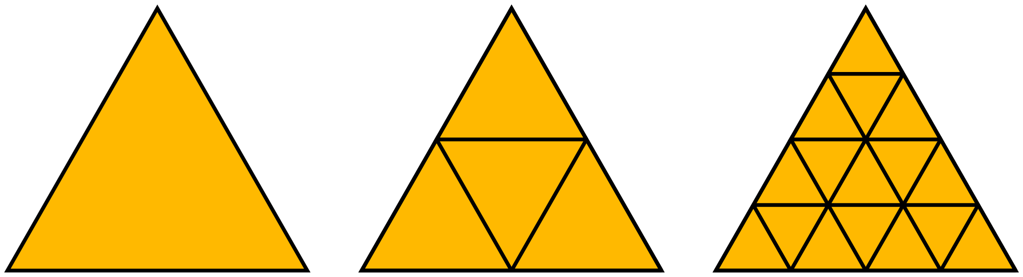
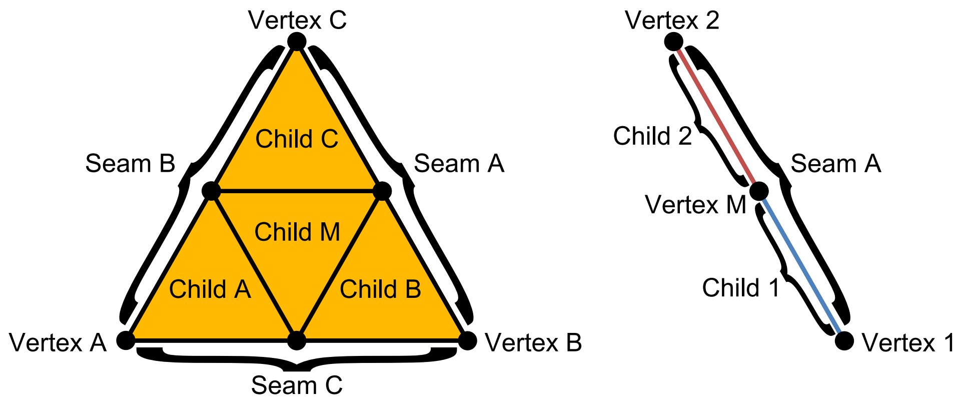

(a) Original Element (b) Level 1 Recursion (c) Level 2 Recursion

Figure 6.2: Quaternary Triangular Element

in a pairwise fashion. For example, Figure 6.2a illustrates a leaf QTM element which was created from the initial surface triangulation (it is referred to as a leaf since it has no branching children). To refine this element, the edges are split and connected to form four child elements, as seen in Figure 6.2b. It is important to note that the actual data structure is stored in a recursive manner, which allows for fast indexing as well as efficient geometric operations through the implicit level-of-detail representation. Further refinement of the original element's four children subsequently produces the discretization shown in Figure 6.2c.

To efficiently represent this structure, MIMS implements each element by storing pointers to its three vertices (Vertex A, Vertex B, and Vertex C), its three seams (Seam A, Seam B, and Seam C), and its four potential children (Child A, Child B, Child C, and Child M), as seen in Figure 6.3a. The edges of each element are referred to as seams in order to differentiate them from the physical edges of the geometric model (at which two faces intersect, causing a discontinuity in surface normals). The seam structure is also recursive, storing pointers to its two endpoints (Vertex 1 and Vertex 2), and, if it has been split, its midpoint (Vertex M) and two children (Child 1 and Child 2), as seen in Figure 6.3b. By



(a) Element (b) Seam

Figure 6.3: Quaternary Triangular Constructs

utilizing this structure, it becomes trivial to refine an existing model by simply determining which leaf elements should be refined and then forcing a split on each seam and creating the necessary children, as seen in Figure 6.1c.

There are several important characteristics of the QTM refinement process which make it ideal for meshless method surface distributions. First, due to the fixed nature of the edge splitting rule, each child of a particular element will be a similar triangle to its parent (scaled by $\frac{1}{2}$). This is very useful as it implies that every child will have the same geometric properties (skewness, edge-to-area ratio, etc.) as its corresponding root element. Thus, if the initial distribution consists of triangles of acceptable quality, then any refined derivative (at any arbitrary level) is guaranteed to consist of triangles of no-worse quality. This allows the point distribution method to perform a single optimization on the initial triangulation and not worry about further optimizations once refinement has occurred. Another important characteristic is that if computational nodes are placed exclusively at the vertices and/or centroids of the leaf elements, they can only be added to the distribution through refinement, never removed. This is due to the fact that the vertices are shared between the parent

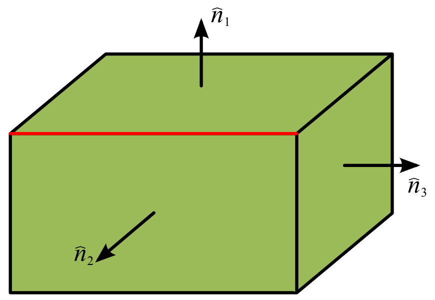
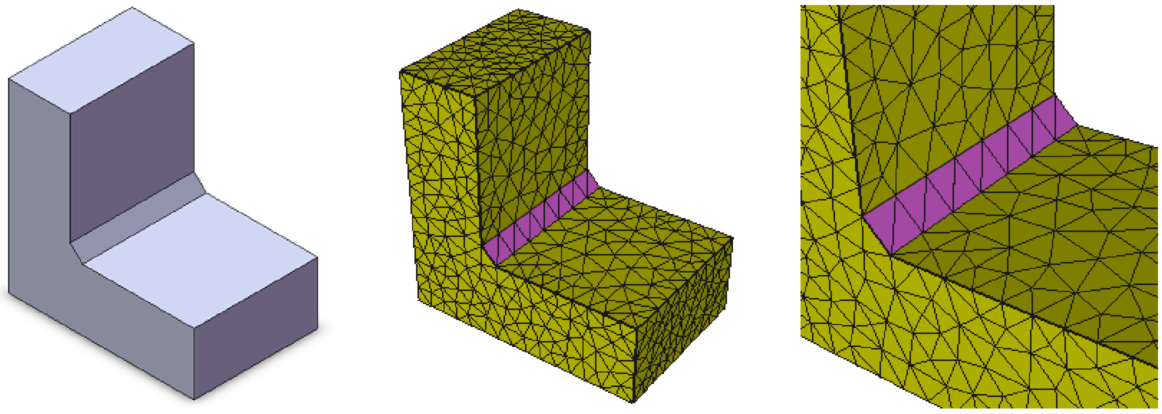


Figure 6.4: Geometric Edge

element and its children, as is the centroid, which is shared between the parent and its middle child (Child M). Thus, there is no need to keep track of complicated nodal histories when transitioning from one refinement level to the next. Lastly, because the elements produce a structured surface mesh over the domain, the underlying meshless method can utilize the direct surface values to calculate tangential derivatives on the boundary. Oftentimes this is difficult to obtain using direct differentiation of the underlying shape functions because the topologies tend to be highly one-sided, leading to excessive error due to the extrapolation that must occur. However, even on highly convex boundaries, the tangential derivatives are easily obtained from the surface elements directly, leading to more robust and accurate solution reporting.

The final component of the surface discretization process is determining how to place the computational nodes on the boundary of the domain. Given that the QTM has been constructed, the most logical location to place computational nodes is either at the vertices or centroids of the leaf elements. It has already been mentioned that either of these placement locations result in no nodes being removed during refinement, however, both locations have special considerations that should be discussed. Using the element vertices seems like the most logical choice for nodal placement, however, there is the issue of how to handle those



(a) Model (b) QTM (c) QTM Closeup

Figure 6.5: Narrow Geometric Face

vertices that lie on geometric edges where the normal vector is not easily defined, such as the indicated edge in Figure 6.4. Although, in this case, averaging $\hat{n}_1$ and $\hat{n}_2$ may seem appropriate, for the case of general face intersections, this is not always true. Therefore, to place nodes at element vertices, one solution is to utilize only vertices that lie on the interior of geometric faces. If this technique is utilized, however, there may be situations where no nodes are placed on the entire face, such as the highlighted face in the QTM discretization shown in Figure 6.5. In this situation, because the geometric face is very narrow, no vertices lie on the interior of the region, and thus, no nodes will be placed on this face. It is therefore advantageous to not only place nodes at the vertices, but also at the centroids of the elements. By doing so, not only is the issue of narrow faces addressed, but it also allows boundary nodes to further approach the edges of the geometry due to the configuration of the elements. Therefore, once the QTM is constructed and refined to an appropriate initial level, computational nodes are then distributed at all non-edge vertices as well as at each leaf element's centroid.

At this point a complete procedure for discretizing the boundary has been described, however, there is one final consideration that needs to be made in order to fully satisfy the requirements for the meshless point distribution method. Recall that one of the primary concerns with mesh-based techniques is their inability to automatically handle small geometric features. It would appear that this may be a concern for the quaternary triangular surface discretization as well, since it is based on a surface triangulation process. However, most surface triangulation routines are fully capable of automatically meshing a boundary with very small features; it just requires an excessive number of elements (due to the small element size). In mesh-based techniques, this directly relates to the computational nodes as these methods require a closed representation of the domain with no missing elements. However, in meshless methods there is no such requirement, and as such, there is no reason why nodes which are deemed too locally refined (too close to their neighbors) can't simply be omitted from the computational domain. Then, if at a later time the local boundary has reached a level of discretization that is consistent with the small feature's size, the omitted nodes may be simply reinstated. The ability to simply omit a problematic node is a prime example of a computational liberty that is afforded by utilizing a meshless method over more conventional mesh-based techniques.

6.2 Binary-Subdivision Interior Discretization

Octree and quadtree based structures have been commonplace in the field of mesh generation, not only for meshless techniques, but also for finite element and finite volume simulations as well. Their implicit connectivity and recursive structure not only allow for easy construction,

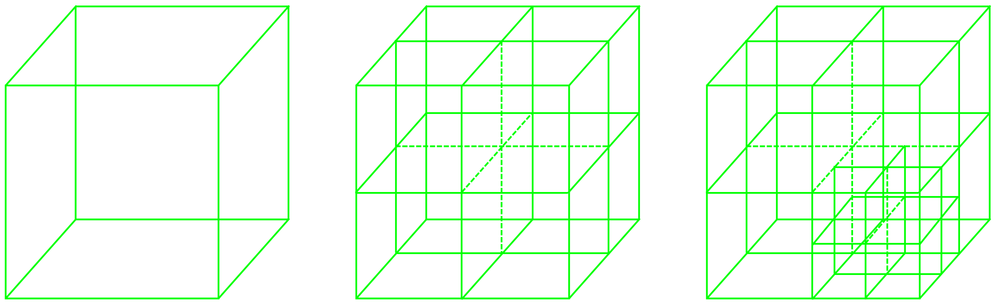

(a) Level 1 (b) Level 2 (c) Level 3

Figure 6.6: Octree Refinement

but also fast querying, a fact which makes them well suited to use in binary space partitioning trees [143]. In the purest sense, an octree (and its two-dimensional counterpart, the quadtree) is a structure based upon even subdivisions of a particular Cartesian space. In three-dimensions the Cartesian space represents a bounded volume in xyz space, while in two-dimensions the Cartesian space represents a bounded surface on some planar geometry. To illustrate this concept, Figure 6.6 shows three levels of refinement on a simple three-dimensional cube using octree splitting rules.

Notice how at each level of refinement only specific octree cells need to be refined (as can be seen going from Level 2 to 3 in Figures 6.6b and 6.6c). This property of octrees and quadtrees is commonly utilized to recursively approximate a boundary which is not properly aligned to the underlying grid. For example, Figure 6.7 shows the ability of these Cartesian subdivision techniques to capture non-aligned boundary geometry.

Collectively, Cartesian subdivision techniques offer several advantageous qualities that can be utilized in mesh generation:

1. They allow for fast traversal due to their recursive nature,

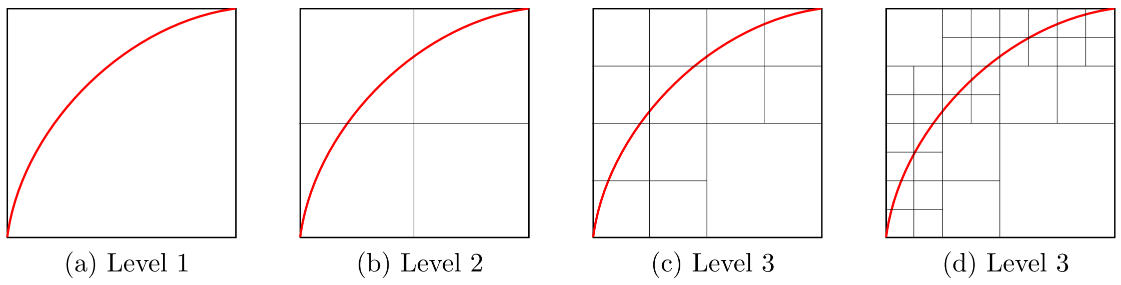
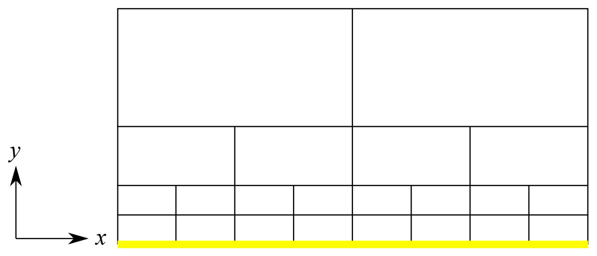


Figure 6.7: Quadtree Boundary Capturing

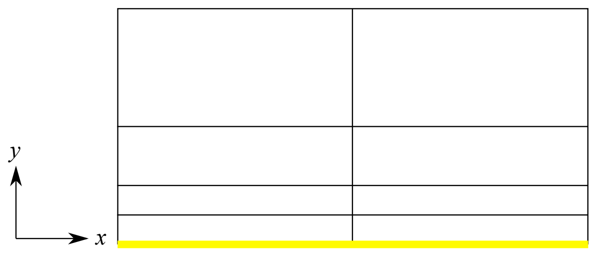
2. They minimize the number of geometric containment checks required as entire cells can be flagged as inside the domain (and thus, all children must be inside the domain as well), and

3. They provide a constant refinement factor (always dividing the space by two).

Indeed, Cartesian subdivision techniques are so popular that an entire solution methodology (Cartesian grid methods) has been developed around their construction [144, 145, 146, 147]. However, the very qualities that make these techniques so promising also present issues when utilized in the solution of complex flows. First, attempting to utilize a mesh based solely on Cartesian subdivision (as is the case with pure Cartesian grid method) exhibits the so-called stair-casing issue where irregular boundaries are present [148]. Although this is generally a major concern for pure Cartesian grid methods, this is not a concern for the MIMS method as an independent surface discretization will be used to provide accurate boundary representations. The second, and more important issue is the inability of pure octree and quadtree discretizations to produce point distributions with non-uniform aspect ratios (aspect ratios that can vary with respect to direction and location). For example, Figure 6.8a illustrates a simple horizontal boundary (bottom of figure) and the resulting quadtree grid when attempting to refine in the y direction to capture a high vertical gradient. It is clear that although the refinement certainly increased the resolution in the desired direction,



(a) Quadtree



(b) Independent

Figure 6.8: Subdivision of Boundary Layer

it also inadvertently refined in the x direction as well (a more appropriate refinement for this case would be the one shown in Figure 6.8b, which has only refined in the y direction). This occurs because of the requirement of pure Cartesian subdivision techniques to always split cells in all directions simultaneously. Thus, in order to obtain an aspect ratio in a particular region of space, the original Cartesian cell must be created with the desired final aspect ratio (as it cannot change its aspect ratio through refinement). Obviously this is a major issue as boundaries are not always oriented with respect to a single orthogonal direction (and thus, the aspect ratio would not be maintained properly throughout the recursive children), only portions of the domain may be able to support high aspect ratios, and most importantly, the user may have no idea where high aspect ratio cells are appropriate. Although there have been attempts at modifying the octree cell structure to allow for directionally independent refinement [149], most have been focused on generating meshes for finite volume methods, where a complete cell structure is required. Fortunately, since meshless methods require no such cell structure, and as such all that is required is a distribution of points, there is no need to stick to a pure octree distribution technique.

To address the issues with pure Cartesian based discretization techniques, a novel point generation strategy based on binary tree subdivisions has been developed. The primary difference between the meshless binary tree discretization technique and previous attempts to extend the octree-based methods to non-isotropic refinement is the underlying data structure. Within all octree (and quadtree) mesh generation techniques, the underlying data structure is the cell (which makes sense, as most of these techniques are coupled with finite volume solvers). However, the underlying data structure of the meshless binary tree discretization is the vertex. Moving the point of view from the cell to the vertex allows for a more dynamic and local data structure capable of non-isotropic refinement as well as highly efficient topology construction. To accomplish this, the Cartesian grid concept may be abstracted into a binary addressing system whereby each direction in the domain is divided into 2^m sections where m is usually set according to the maximum allowable value of the underlying data type (e.g. in most modern architectures the unsigned int data type is stored in 32 bits, and thus m is typically set to 30 in order to allow sufficient room for computations without overflow). This concept is illustrated for a two-dimensional domain in Figure 6.9 which has been discretized using $m = 5$. Thus, this discretization essentially provides an integer based mapping system from floating-point x, y, z coordinates to integer-based i, j, k coordinates. The coordinate transformations are given by

$$x\left(i\right) = \frac{x_{max} - x_{min}}{2^m} \times i + x_{min} \tag{6.1}$$

$$y\left(j\right) = \frac{y_{max} - y_{min}}{2^m} \times j + y_{min} \tag{6.2}$$

$$z\left(k\right) = \frac{z_{max} - z_{min}}{2^m} \times k + z_{min} \tag{6.3}$$

where x_{max}, x_{min}, y_{max}, y_{min}, z_{max}, and z_{min} are the maximum and minimum x, y, z coordinates of the domain, respectively (generally this is set slightly larger than the boundary

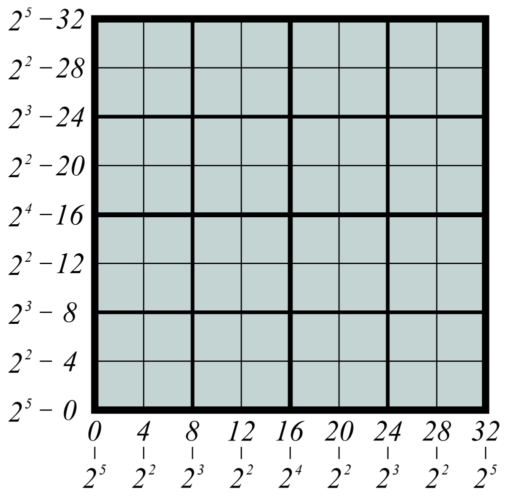


Figure 6.9: Binary Addressing

extents). The primary benefit of using an integer based system is that there is no risk of incurring any round-off error during calculations, a fact which greatly simplifies the procedure (as there is no need to account for error in floating point comparisons). In addition, because of the binary nature of the mapping, all divisions and multiplications may be performed using efficient bit shifts:

$$\frac{n}{2} = n \gg 1 \tag{6.4}$$

$$n \times 2 = n \ll 1 \tag{6.5}$$

where $\gg$ and $\ll$ represent the right and left bit shift operators, respectively.

Another important characteristic of the binary addressing system is that, through use of the i, j, k coordinates alone, the *discretization level* at which a node resides can be determined. The discretization level represents the number of binary subdivisions required to reach the root level (corresponding to a spacing of 1) from a given address and is useful during refinement operations, as will be later demonstrated. This value can be computed by simply continuously dividing each coordinate by 2 until it is not evenly divisible, at which point the number of divisions that has been performed is referred to as the discretization level of that node. Expressed mathematically, the process of determining the discretization level for an integer coordinate w may be represented by the following recursive function

$$L(w,d) = \begin{cases} d & \text{if } w \bmod 2 \neq 0 \\ L\left(\frac{w}{2}, d+1\right) & \text{otherwise} \end{cases} \tag{6.6}$$

where $x \bmod y$ is the modulo operator which returns the remainder of x divided by y. Thus, the function provided in Eq. (6.6) may be initially seeded with $d = 0$ and the return value will indicate the binary gridline on which w resides. The only exception is the case where $w = 0$, for which the return value should always be m. To illustrate the various discretization levels, Figure 6.9 provides the values for each coordinate in the form 2^d.

Having detailed the addressing system, the domain can therefore be constructed from a collection of vertices, with each vertex storing its i, j, k position in space, references to its six neighbors (negative/positive i, j, and k), and a bit flag to indicate the state of each direction (as shown in Figure 6.10). In this manner, storage is kept at a minimum (approximately 40 bytes per node on most 32-bit systems), while at the same time encapsulating the most efficient (and necessary) aspects of the octree cell distribution. In particular, storing direct references to the Cartesian neighbors allows any node, in $O(1)$ time, to identify its neighbors (if they exist), a process which is necessary for determining whether or not a node may utilize structured optimizations (i.e. finite differencing).

The process of building the initial binary-subdivision interior discretization (BSID) begins by collecting statistical data from the boundary discretization in order to reference

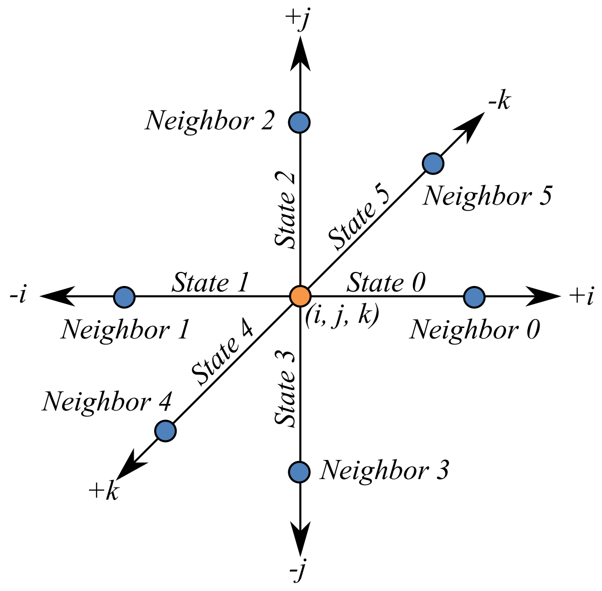


Figure 6.10: Cartesian Vertex

an appropriate initial nodal spacing. First, the average (s_{ave}), standard deviation (s_σ), and maximum (s_{max}) element seam lengths are computed globally for the boundary discretization. The minimum value is then selected from the maximum seam length (s_{max}) and average seam length plus standard deviation ($s_{ave} + s_\sigma$), which will serve as the initial starting delta. Finally, the extents of the domain are computed in xyz space, which allows for transformation between the integer addressing system and original model. Once the necessary quantities have been obtained from the boundary discretization, the next step is to seed the interior with a single node. The key to the binary-subdivision discretization is the fact that once proper connectivity has been established to any collection of nodes, the remaining geometry can be constructed using refinement, as opposed to specialized construction routines. Thus, the minimal number of nodes required to seed the interior is one (a single node in the center), plus the corresponding extent nodes which will serve as the extents of the domain. The concept of an extent node is simply a vertex which lies at the extents of ijk space, which, by definition will have one or more null neighbors. Figure 6.11 shows the initial seed

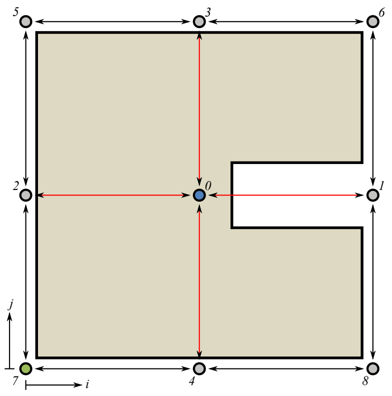


Figure 6.11: Example Initial Seed

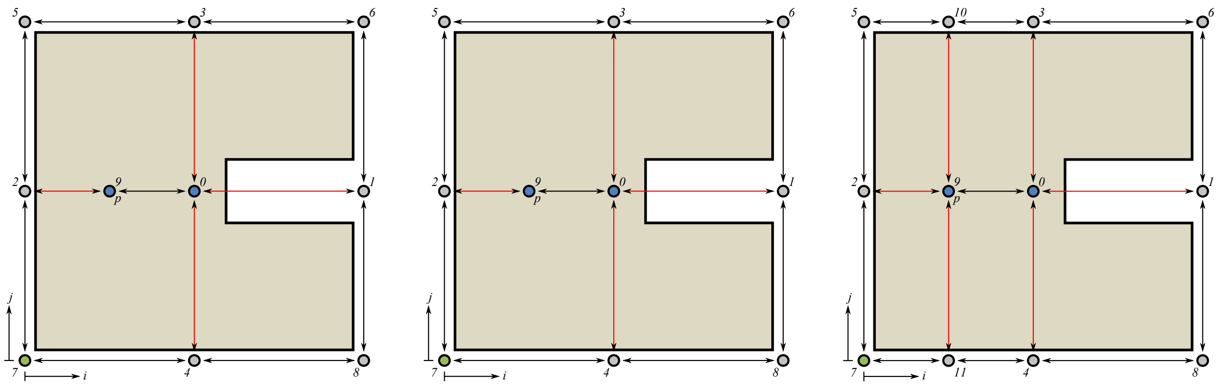
and the associated connectivity for a simple two dimensional domain (examples are shown in two-dimensions for illustrative purposes, however, the process is dimensionally independent). There are several important features to note in Figure 6.10, prior to describing the refinement process. First, extent nodes are produced at all extents of the domain, including at the origin $(0, 0)$ and at the maximum coordinates $(2^m, 2^m)$. This is done to allow for traversal along the boundary through each node without having to facilitate complicated referencing (e.g. referencing nodes 3 and 1 if node 6 was not present). Second, the node located at the origin (marked in green) has been stored separately and will serve as a starting point for several sweeping operations that are performed during refinement (as will be later described). Third, the color of the nodes indicate whether or not the node is inside (blue) or outside (gray) the domain. Lastly, the color of each neighbor reference (shown as arrows) indicates whether or not a ray drawn between two nodes intersects the boundary. In Figure 6.11, a red line indicates that the line intersects the boundary and a black line indicates that it does not. Thus, not only must the boundary representation be capable of determining if a point lies inside its bounds, but it also must be capable of determining if a line segment pierces any of its elements.

Once the initial seed has been produced for a domain, refinement can be used to generate the remaining initial discretization, and in this manner, serves as a general process which accepts any criteria as the basis of distribution. The general refinement process may be summarized as follows:

Algorithm 6.1 BSID Refinement Process

```
Input: A fully connected BSID of $\Omega$.
  Let $Q$ be a queue containing the non-extent nodes of input BSID.
  while $Q$ is not empty do
    Let $v$ be current node in $Q$
    for each neighbor $n$ of $v$ do
      if $n$ to $v$ requires refinement then
        Add node $p$ between $n$ and $v$.
        Add node $p$ to queue $Q$.
      end if
    end for
    Remove $v$ from $Q$.
  end while
Output: A refined BSID of $\Omega$.
```

Obviously, there are several important elements from Algorithm 6.1 that need to be described in further detail. First, when determining whether two nodes require refinement (such as in the case between n and v), there are several different criteria that may be used including distance between nodes, field gradients, or various conservation quantities. However, it is important to note that all these criteria take the form of an integer or floating point field on which some refinement threshold may be placed. In the case of the initial discretization, the criteria is the distance between nodes and the threshold is the initial delta as computed during the data collection phase. Second, a process must be developed to properly add a node p between two existing nodes n and v. This process is critical because it must not only produce a new node at the proper location but it is also responsible for



(a) Node Insertion (b) Establish Connectivity (c) Establish Extents

Figure 6.12: BSID Refinement Process Example

ensuring proper connectivity of the BSID once p has been inserted. The node insertion process may be summarized as follows:

Algorithm 6.2 BSID Node Insertion Process

```
Input: Two nodes, $v$ and $n$, and a direction $i$ indicating how $v$ and $n$ are related.
  Let $\delta \leftarrow L(v_i, 0)$.
  Let $s \leftarrow |v_i - n_i|$.
  while $s \leq \delta$ do
    $\delta \leftarrow \frac{\delta}{2}$
  end while
  Let $p$ reside at distance $\delta$ from $v$ in $i$ direction.
  Connect $p$ to $v$ and $n$.
  Determine interior state of $p$.
  Determine bounding state of $p$ in $i$ direction.
  Establish connectivity for $p$ (Algorithm 6.3).
  if $p$ is missing neighbors then
    Establish missing extents for $p$ (Algorithm 6.4).
  end if
```

Algorithm 6.2 illustrates the use of the discretization level to compute the proper location of the newly inserted node p. Using this value is critical to a proper discretization (as opposed to simply placing p at the midpoint between v and n) as it ensures that p always subdivides the space between v and n in a binary appropriate manner. Figure 6.12a demonstrates the insertion process between nodes $n^{\langle 0 \rangle}$ and $n^{\langle 2 \rangle}$ up to the point where full connectivity is established. Notice that at the time that node p ($n^{\langle 9 \rangle}$) has been connected to its creating neighbors, and it has also been properly identified as inside the domain and intersecting the boundary in the negative i direction. Note that because of the single directional nature, many optimizations may be made during containment and intersection checks. For example, in the case of $n^{\langle 9 \rangle}$, because $n^{\langle 0 \rangle}$ and $n^{\langle 2 \rangle}$ are known to intersect the boundary, if a single intersection check is performed between $n^{\langle 0 \rangle}$ and $n^{\langle 9 \rangle}$, it can immediately be determined that there must be an intersection between $n^{\langle 9 \rangle}$ and $n^{\langle 2 \rangle}$. This is due to the fact that there must be at least one intersection somewhere between $n^{\langle 0 \rangle}$ and $n^{\langle 2 \rangle}$, and if there is no intersection between $n^{\langle 0 \rangle}$ and $n^{\langle 9 \rangle}$, then there must be an intersection between $n^{\langle 9 \rangle}$ and $n^{\langle 2 \rangle}$. Similar optimizations can be performed in cases where no intersections are present (in these cases, no containment or intersection checks need to be performed). Although these optimizations do not directly affect the outcome of the point distribution process, they can considerably improve the computational performance of the algorithms and assist in achieving quadtree/octree-like performance. In addition, it should be noted that direction i in Algorithms 6.2, 6.3, and 6.4 refer to any coordinate direction of the ijk system. In this regards, j or k could be passed into these routines and the coordinate system would be referenced by the indicated direction.

Once the node has been inserted, connectivity must be established for the remaining directions. This is done by utilizing the origin of the discretization and utilizing a sweeping process to determine whether or not nodes exist in the expected directions. This process may be represented as:

Algorithm 6.3 BSID Establish Connectivity

```
Input: Unconnected node $p$ and insertion direction $i$.
  Let $t \leftarrow$ origin.
  repeat
    $t \leftarrow$ neighbor of $t$ in $k$ direction.
  until $t_k = p_k$
  while $t_i < p_i$ do
    $t \leftarrow$ neighbor of $t$ in $i$ direction.
  end while
  if $t_i = p_i$ then
    while $t_j \leq p_j$ do
      $t \leftarrow$ neighbor of $t$ in $j$ direction.
    end while
    Let $r \leftarrow$ neighbor of $t$ in negative $j$ direction.
  end if
  Connect $p$ to $t$ and $r$.
  Determine bounding state of $p$ in $j$ direction.
```

Applying Algorithm 6.3 to the configuration shown in Figure 6.12a produces Figure 6.12b. Notice that no connectivity has been established since there are no nodes present in the positive and negative j directions. The remaining component is to establish the missing extents for this node, which may be done according to Algorithm 6.4.

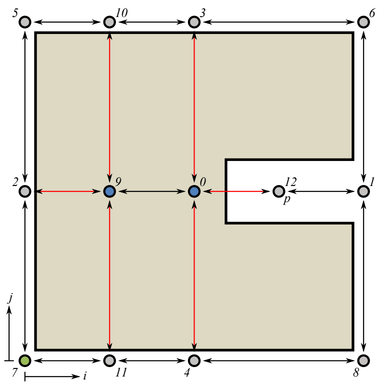
Algorithm 6.4 BSID Establish Missing Extents

```
Input: Incomplete node $p$ and insertion direction $i$.
  for each missing neighbor of $p$ in direction $h$ do
    Create extent node $e$ in $h$ direction.
    Connect $p$ to $e$.
    Fully connect $e$ to neighboring extent nodes.
    Determine bounding state of $p$ in $h$ direction.
  end for
```

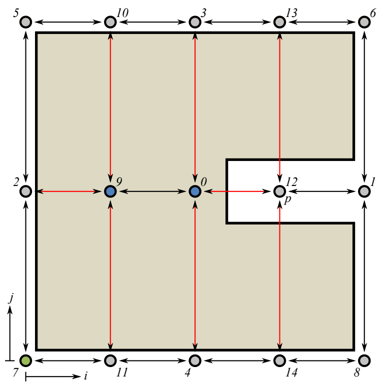
The only remaining component is fully connecting the newly inserted boundary nodes, for which a process very similar to Algorithm 6.3 may be developed. Applying this routine to the configuration shown in Figure 6.12b results in the valid configuration shown in 6.12c. It should become clear the importance of adding and fully connecting the necessary extent nodes as it provides a search path to connect newly inserted nodes to the off-direction neighbors. To demonstrate this process for full distance refinement of the domain shown in Figure 6.11, Figure 6.13 illustrates the resulting discretization after each stage of node insertion for a particular queue ordering.

The discretization process illustrated in Figure 6.13 highlights some of the more important aspects of the binary-subdivision interior distribution process. First, a principal concern of adding each node is to ensure that once insertion is complete, the point distribution is once again in a completely valid connectivity state. This is critical to the proper operation of the refinement strategy as it relies on the internal connectivity to determine where to place new nodes. Second, the connections which intersect the boundary (highlighted in red) can be shown to represent the boundary shape in much the same way as quadtrees and octrees (such as in Figure 6.7). This is important as it allows the point distribution to optimize unnecessary operations by implicitly knowing what is and is not inside the domain. As the number of points is increased, this implicit boundary representation will become more detailed, and as such, the optimizations that the interior point distribution can apply will have a larger effect. Lastly, as the point distribution process is inserting nodes in a directionally independent manner (as opposed to isotropically splitting cells), it is easy to see how it is able to dynamically adjust aspect ratios by simply refining only in select directions.

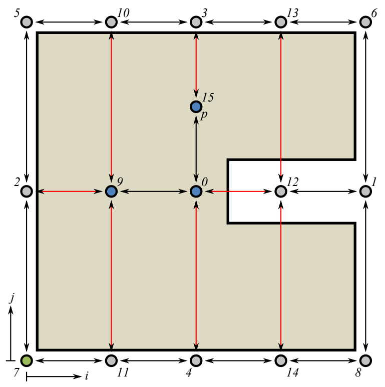
The final element of the interior discretization technique is addressing the boundary/interior interface. This process is of critical importance to the stability of the method as even though a node may be inside the domain, it may be too close to the boundary relative



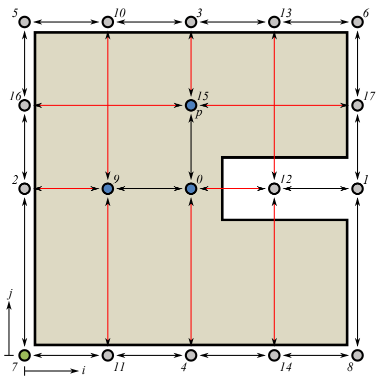
(a) $n^{\langle 12 \rangle}$ Node Insertion



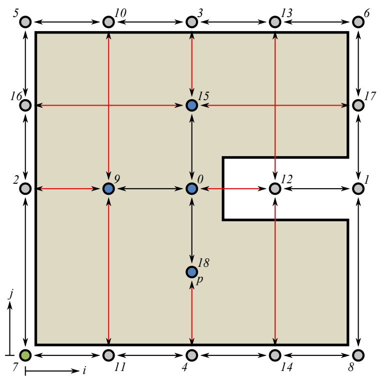
(b) $n^{\langle 12 \rangle}$ Establish Extents



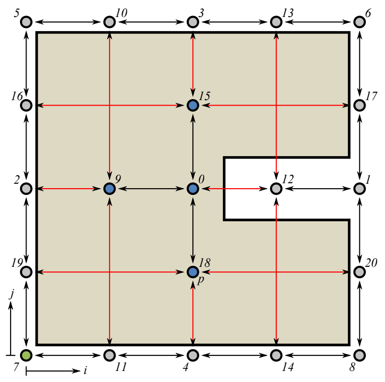
(c) $n^{\langle 15 \rangle}$ Node Insertion



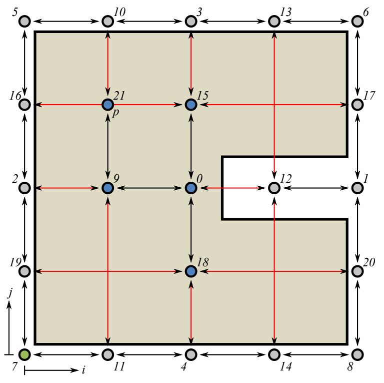
(d) $n^{\langle 15 \rangle}$ Establish Extents



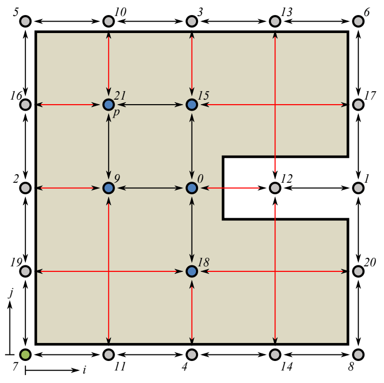
(e) $n^{\langle 18 \rangle}$ Node Insertion



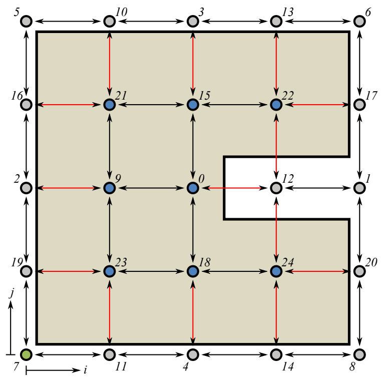
(f) $n^{\langle 18 \rangle}$ Establish Extents



(g) $n^{\langle 21 \rangle}$ Node Insertion



(h) $n^{\langle 21 \rangle}$ Establish Connectivity



(i) Final Distribution

Figure 6.13: BSID Refinement Steps

to the local element size. If this were to occur, the topology for the interior node would be very poor due to the highly one-sided nature of the local distribution. However, avoiding this issue may be accomplished by implementing a local distance threshold (based on the nearest boundary size) below which all interior nodes are simply omitted from the domain. Thus, in much the same manner as overly refined boundary nodes were removed, there is no need for special cases during point distribution; instead, an appropriate filter is simply applied to remove ill-behaved nodes prior to the solution process.

6.3 Shadow Layer Discretization

The final component of the meshless model generation process is construction of the so-called shadow layer distribution which serves as an adaptive boundary layer for problems exhibiting high gradients normal to the boundary. The shadow layer serves as an interface between the boundary and interior regions whereby boundary quantities may be accurately obtained (by normal-aligning the distribution to the boundary) and sufficiently isolated boundary information may be propagated back into the interior. Although handled slightly differently during model generation, it is important to note that nodes that reside in the shadow layer are, for all intents and purposes, interior nodes. No special boundary layer equations are solved, and the governing equations are collocated normally throughout this region of the domain. The only thing that distinguishes them from the interior (besides their orientation) is the fact that the differential operators are formulated in a rotated coordinate space to ensure alignment to the local boundary (discussed in detail in Section 8.2).

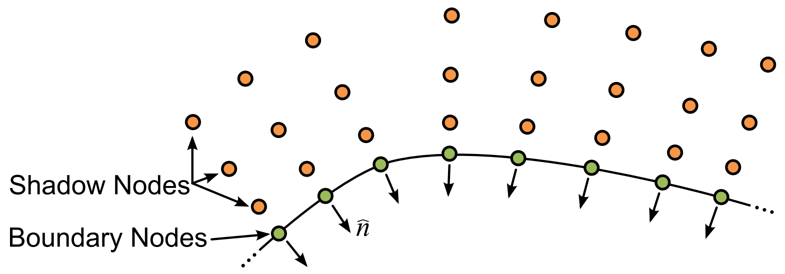


Figure 6.14: Shadow Layer

The process of generating the shadow layer begins by analyzing the boundary conditions applied to each surface of the model. This is done such that only surfaces with high normal-gradient conditions will generate a shadow layer. In particular, this technique is of primary interest where no-slip wall conditions are applied in fluid flow which are capable of producing extremely high normal gradients (boundary layers). The process of actually generating the shadow nodes is fairly straightforward and is illustrated on a simple two dimensional boundary in Figure 6.14. Notice that in this figure, the shadow layer has a depth of 3 (the depth is generally initialized at 2 and allowed to refine as the solution progresses), and each boundary node (shown in green) has a single associated column of shadow nodes. In this manner, normal derivatives may be obtained directly (both on the boundary and in the shadow layer) through finite differencing and tangential derivatives are generated in the usual meshless manner. The placement of the outer shadow layer (one farthest from boundary) is based on the distance to the nearest interior node (generally half of this value) and subsequent layers are distributed using an appropriate scaling method. By adjusting the scaling method, one can obtain faster (or slower) growth in the shadow layer, similar to how structured boundary layer meshes have an associated growth factor set by the user. However, in the case of meshless method, this scaling factor may be automatically controlled

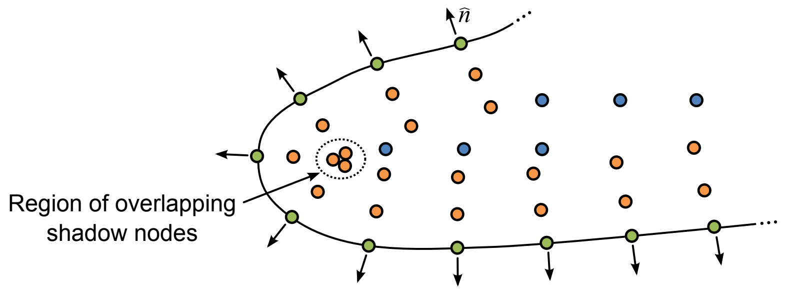


Figure 6.15: Concave Shadow Layer Region

by the field behavior (through examination of the normal to tangential gradient ratio) and updated accordingly during refinement phases.

It is important to realize that the process of adding shadow nodes can potentially introduce problems in highly concave boundary situations, as illustrated in Figure 6.15. To address this issue, a technique of collapsing shadow nodes which are very close to one another is necessary in order to eliminate instability in the underlying meshless interpolations. This process begins by creating a collapsed node at the center of mass for the offending set of shadow nodes. This collapsed node utilizes an interpolation operator which simply averages the values from the nearby lumped nodes. In this manner, any node which is not part of the collapsed set simply utilizes the collapsed node instead of the set of overlapping shadow nodes which could cause numerical instability (as in the case of RBFP interpolation). Since the overlapping nodes are, by definition, very close to one another, a simple average value provides sufficient accuracy to represent the nodes as a single point of influence. Similarly, when solving the governing equation at the offending nodes, all overlapping nodes are invisible to each other and they each individually satisfy the governing equation at their respective location in space. By lumping their effect for external influences and eliminating their effect on one another, the nodes do not need to be directly removed and the respective boundary nodes do not need to apply special finite difference equations.

By utilizing a discretization that can provide an independent boundary layer representation, the model generation procedure can develop point distributions capable of capturing high boundary gradients without having to resort to extreme interior refinement in order to correspond with non-aligned boundaries. In addition, since the shadow layer is directly aligned with the normal and tangential directions on the boundary, it becomes trivial to produce high-aspect ratio point distributions with respect to the boundary orientation.

6.4 Chapter Summary and Examples

One of the major innovations of this research is the integration of the solution process with the model generation procedure. In this regards, the liberties offered by the meshless technique are able to be fully utilized to produce a robust solution process. However, it is important to understand that each respective component of the model generation process represents a significant advancement of the state-of-the-art in model generation. The quaternary triangular surface discretization represents a new use for a data structure that, up to this point, has been utilized primarily in rendering applications. By tailoring this structure to the generation of meshless point distributions, it is able to serve as a comprehensive boundary discretization procedure able to provide a more complete description than previously possible. The binary-subdivision interior discretization technique is a unique and novel method of distributing points within a domain. Although very difficult to adapt to mesh-based techniques, the requirements of the meshless methods provide the

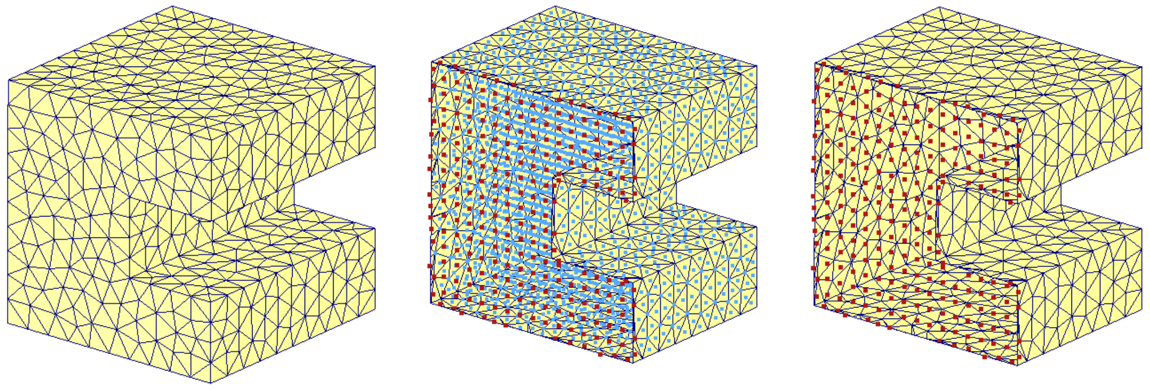

(a) Boundary Discretization (b) Interior Distribution (c) Interior Plane Highlighted

Figure 6.16: U-Block Point Distribution

foundations for its development and will hopefully serve as a new class of discretization schemes. Lastly, the shadow layer discretization illustrates a major advancement of this method with respect to current meshless techniques as it allows for direct capturing of the high gradients typically seen in boundary layer regions. This area has historically been a serious concern for all numerical physics simulators (not just meshless methods) as the discrepancy between boundary and interior node distributions occur in the regions which have the highest gradients, and are, therefore, most prone to error. By directly addressing this through the use of the shadow layer, the proposed method is able to move the distribution discrepancy away from the boundary, and thus, out of the high gradient boundary layer regions. In this respect, it is able to provide more accurate boundary layer solutions while simultaneously increasing the stability of the technique.

To demonstrate the types of point distributions obtainable using the presented model generation procedure, several initial discretizations were generated and are presented in Figures 6.16 and 6.17. In each of these examples, three figures are shown. The first set of figures (Figures 6.16a and 6.17a) present the initial quaternary triangular surface

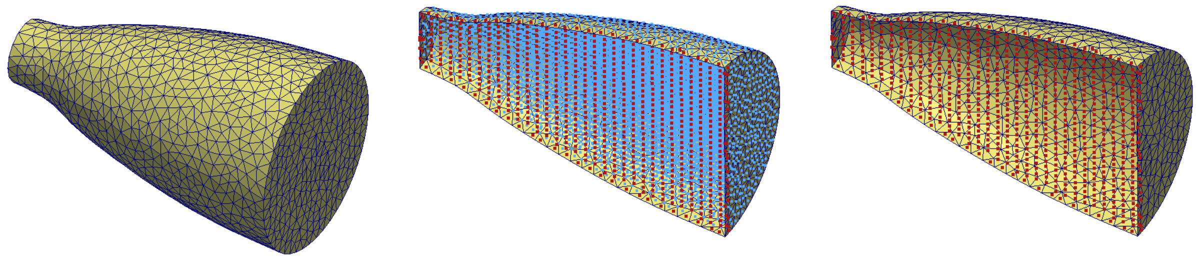

(a) Boundary Distribution (b) Interior Distribution (c) Interior Plane Highlighted

Figure 6.17: Axisymmetric Nozzle Point Distribution

discretization which serve to illustrate the underlying boundary distribution. The second set of figures (Figures 6.16b and 6.17b) have had the boundary sliced in order to expose the interior distributions present. In addition, a particular plane of interior (and boundary) nodes have been highlighted in order to bring attention to the local structure present throughout much of the domain. Finally, the last set of figures (Figures 6.16c and 6.17c) have had all nodes removed except for the highlighted plane, to aid in visualization. In both of these examples, it becomes clear that by utilizing a Cartesian based approach to generate interior nodes, a vast majority of the interior domain has local structure. This represents a tremendous advantage of this point distribution process as the solution methodology can leverage the local structure to reduce overall preprocessing and computational time. In addition, these examples illustrate the ability to produce interior discretizations that conform to the initial boundary discretization. This is important as it will allow for consistent refinement when the solution dictates that both the interior and boundary point distributions need to be improved.

CHAPTER 7
ADAPTIVE REFINEMENT

Adaptive solution refinement represents one of the major cornerstones of the MIMS method as it allows solutions to develop without prior knowledge of the underlying characteristics. Without adaptive refinement, it would be very difficult (if not impossible) to obtain a solution methodology capable of automatically solving arbitrarily complex problems without human interaction. Fortunately, the use of the shape functions outlined in Chapter 4 and the model generation techniques outlined in Chapter 6 allow the MIMS method to achieve a robust h-refinement strategy, as well as an opportunity to provide adaptive refinement on the underlying geometric model, a feature unique to this process.

7.1 Point Distribution (h) Refinement

The most common adaptive refinement strategy, point distribution refinement (often referred to as h-refinement) is performed by reducing the average nodal spacing in areas of high gradient, thereby increasing the accuracy of the underlying interpolation (or approximation) techniques. Refinement of the underlying node distribution (through adding new points, generating finer meshes, etc.) is a critical component of automated solution techniques as it allows for capturing of field characteristics which occur at length scales smaller than the original distribution. In addition, it provides a means of selectively increasing the computational expense of a solution, committing more resources to the areas which require

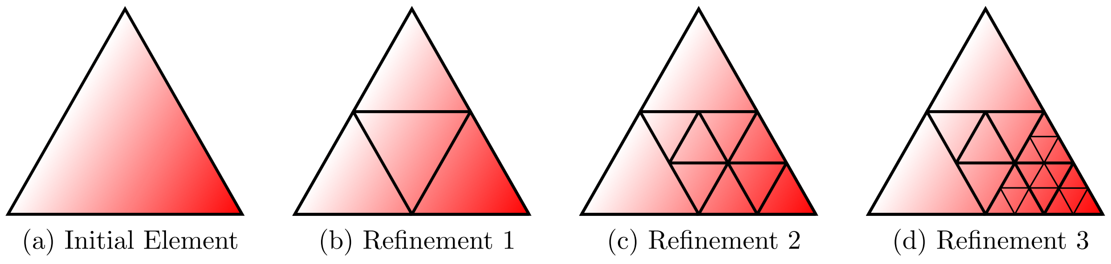


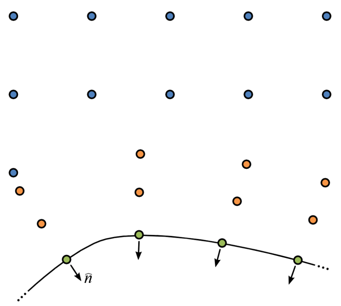
Figure 7.1: Boundary Element Refinement

it most (as opposed to simply naively refining the entire discretization). The MIMS method employs point distribution refinement via the three model generation processes presented in Chapter 6. It has already been demonstrated how the boundary, interior, and shadow layers implement adaptive refinement, however, it is important to examine how each of these mechanisms may be driven by the underlying field solution.

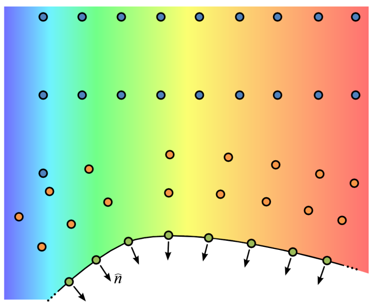
As previously described, the MIMS method utilizes a quaternary triangular surface discretization as its boundary representation and through a recursive subdivision process, is able to selectively refine the point distribution. The selective refinement process is illustrated in Figure 7.1 where an initial element (Figure 7.1a) is refined based on an underlying gradient field. Here, the red regions of the field have higher gradients, and are therefore subject to higher levels of refinement, as seen in Figure 7.1d. Understanding that the boundary discretization can perform local refinement, it is now important to identify the types of field gradients that may be identified and refined using this technique. Recall that any refinement that occurs in this manner will be adding solution nodes on the boundary, therefore, only the nodal spacing in the tangential direction will be reduced. Thus, this mode of refinement is appropriate for capturing high gradients that act tangential to the boundary.

To represent the interior, the MIMS method utilizes a binary-subdivision interior discretization capable of anisotropic refinement on both large and small scales. The utilization of a vertex-based structure allows for evaluation of refinement criteria between nodes, allowing for an implicit gradient based approach. Undeniably, this discretization technique allows for directionally independent refinement anywhere within the interior of the domain, allowing the technique to capture flow phenomenon such as wakes and bulk flow turbulence without needing to identify these locations prior to solving the problem. However, in the boundary/interior interface regions, unless the boundary is aligned with the coordinate axes, the interior discretization will be unable to perform direct refinement in the boundary normal direction. Although it certainly will be able to perform some sort of refinement in this area, because it will not be aligned with the boundary, it will be impossible to achieve high aspect ratios (since refinement will need to occur in multiple directions to account for the curvature of the boundary). Therefore, this type of refinement, although appropriate everywhere in the interior, is not well suited to refining in areas with large gradients normal to the boundary.

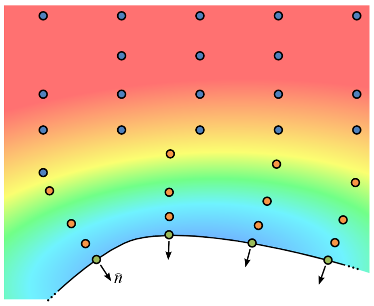
The final component of the model generation procedure is the shadow layer, which specifically addresses the inability of the interior distribution to refine normal to the boundary. Because the shadow layer is a direct normal projection of each boundary node, it is trivial to add additional fidelity in the normal direction. Furthermore, by isolating the normal direction, the shadow layer is able to refine without affecting the resolution in any other coordinate frames. An important characteristic of shadow nodes is that although their initial placement is based on the boundary condition type, if it is later found that another boundary has high normal gradients, they may be added in that region to increase the fidelity of the solution and facilitate refinement in the normal direction.



(a) Initial Distribution



(b) Tangential Gradient



(c) Normal Gradient

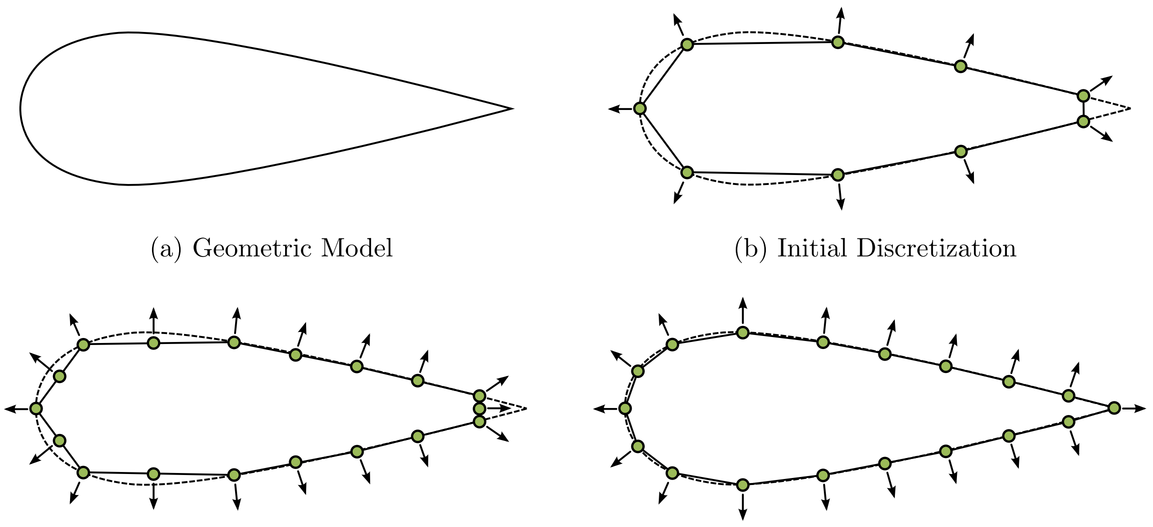
Figure 7.2: Refinement Examples

To summarize, refinement of the point distribution is achieved through the three subsystems of the model generation process, with each subsystem addressing different types of solution gradients. As a final illustration, Figure 7.2 demonstrates the results of refining an initial distribution on two different fields. Figure 7.2b presents a tangential gradient, and in this situation it is clear that the boundary is refined in the tangential direction and the interior refines to match. Figure 7.2c, on the other hand, presents a field containing gradients in the normal direction, and in this case the shadow layer has refined, along with the interior directions exhibiting the gradient. In this regards, it should be clear how the various model generation components are able to work together to selectively refine in those areas which are most affected by the underlying solution characteristics.

7.2 Geometric Model Refinement

The second type of refinement present in the MIMS method is refinement of the computational representation of the underlying geometric model. This type of refinement is oftentimes overlooked (either because it is too complex or deemed unnecessary) in traditional solution techniques despite representing a major component of the automatic refinement process. The basic concept is that as the computational discretization (whether it be mesh or point based) is refined, it should better represent the underlying geometric model. This is possible because most computational models are generated from analytical surface representations (such as IGES or STEP) and therefore have access to an underlying CAD model capable of representing the geometry at arbitrary resolutions. Unfortunately, most mesh generation software simply constructs an initial mesh on the surface of the part and never re-consults the underlying analytical model to provide a more accurate representation. This behavior is partially due to the limitations of automatic re-meshing and refinement, and partially due to the fact that most solution techniques are not directly coupled to the underlying mesh generation process (and therefore do not have access to the original geometric model). The proposed meshless method point distribution techniques, on the other hand, can be fully coupled with the analytical surface representations with minimal effort and therefore can achieve complete correlation between the computational and geometric models throughout the entire solution process.

To illustrate the importance of geometric model refinement, Figure 7.3 presents the commonly encountered problem of discretizing an airfoil. Figure 7.3a shows the original, analytical model provided to the discretization software while Figure 7.3b represents the initial discretization created. Assuming that this computational model is accurate enough to generate a basic solution, the solver now decides to refine the model. If, however, it is not capable of consulting the geometric model, the best representation that it can generate is the one shown in Figure 7.3c, which is generated by direct interpolation of the original discretization. However, if the refinement process is able to conform to the original geometry,


(a) Geometric Model (b) Initial Discretization

(c) Refinement based on Discretization (d) Refinement based on Model

Figure 7.3: Geometric Refinement Example

then a much more accurate representation is possible, as shown in Figure 7.3d. For many problems this is a minor concern, however, for problems with highly curved geometry (such as in the case with the airfoil) then the differences between Figures 7.3c and 7.3d can have a significant effect on the overall solution accuracy.

By implementing a geometric model refinement process, the presented method is able to distribute its initial point distribution without concern of limiting the solution accuracy through misrepresentation of the underlying geometry. As the solution refines, so too will the geometric representation, leading to more accurate solution behavior. The only requirement of the initial point distribution is that it be refined to a degree sufficient enough to represent the basic behavior of the underlying fields. Once this is achieved, automatic refinement can use the underlying field data to determine which areas require additional refinement, and further improvement of the geometric representation will occur when appropriate.

7.3 Refinement and Convergence Criteria

The goal of any numerical method is to obtain accurate solutions, however, due to the dependence on the underlying problem discretization it is difficult to guarantee this fact without demonstrating what is often referred to as *grid convergence*. Essentially, the concept of grid convergence is that if the grid spacing is further reduced from a given solution, the solution will not appreciably change; thus, the current result has become independent of the discretization. This is an important quality to demonstrate prior to reporting a result as final since the solution behavior can change substantially with respect to discretization before this is obtained (it is common to see results change 50% or more as the discretization is improved). Typically, when solving problems using mesh-based approaches, an engineer must solve the problem on several discretization levels in order to demonstrate that the solution is no longer varying with respect to the underlying mesh. As one can imagine, this adds considerable overhead and computation time to the solution process. Furthermore, it is difficult (and extremely time consuming) to analyze the intermediate solutions for *local grid convergence* (grid convergence on a node-by-node basis) since it is unlikely that nodes will directly align from one independent mesh to the next. Fortunately, because of the adaptive refinement processes and the coupled nature of the underlying CAD model and meshless solver, MIMS is able to integrate the grid convergence idea directly into the solution process (note that *grid*, in the case of MIMS, refers to the underlying point distribution). That being said, it is important to understand the refinement and convergence criteria used to determine the refinement sites during the refinement process, and how MIMS utilizes the previous solution levels to determine when to report a solution as final (i.e. grid converged).

Having defined the way in which MIMS utilizes the three geometric representations to produce a comprehensive refinement strategy, the actual refinement process involves a three-stage approach. First, refinement sites are selected based on their current field (gradient) value and past solution history. Next, distance-scaled gradients are calculated across the potential refinement sites (for the indicated refinement field). Finally, the sites containing the highest gradients are isolated and refined. The first stage, selecting valid refinement sites, is arguably the most important as it provides a means of automatically obtaining grid convergence. However, if gradient values alone were the criteria on which refinement was based, the same percentage of nodes would continue to be selected for refinement since they will always have the highest gradient values. As the goal of grid convergence is to have no change in solution from one discretization level to the next, the field values immediately following the previous refinement level are maintained and compared to the converged solution at the current level. If a node's value did not change by a certain percentage relative to the total field span, then it cannot cause a refinement site to be created (though it may still be part of a refinement site if a nearby node passes this threshold). Therefore, by controlling this refinement threshold, one can essentially enforce local grid convergence across the solution domain. In practice, grid convergence percentages of 1-2% with respect to the field span produce appropriately grid converged solutions within an acceptable number of refinement levels (and iterations). Once the candidate refinement sites have been determined, the distance-scaled gradients are computed for each location. To accomplish this task, one can utilize the fact that each refinement site represents a structured region within the model; interior refinement sites represent two neighboring nodes (thus, gradients may be calculated via finite differencing), shadow refinement sites represent a shadow node and its boundary/shadow column neighbor, and boundary refinement sites represent a particular QTM surface element. In this respect, no additional meshless operators are required during the gradient calculation step (a necessity if acceptable speed is going to be maintained). Once the distance-scaled gradient values have been computed for each candidate site, the values are sorted by magnitude and the top percentage threshold is chosen for refinement (in practice, a threshold which includes all sites with a gradient value in the top 20th percentile obtains a good balance of iteration and refinement speed). Once refinement has occurred, the solution is allowed to iterate until an acceptable convergence criteria has been met, at which point the refinement process is employed again. The benefit of this procedure is that the task of identifying when the solution has reached grid convergence is removed from the user; instead, the process continues indefinitely until there are no more remaining candidate refinement sites, indicating that no change has occurred from the previous refinement level, and that the solution has been successfully grid converged.

7.4 Chapter Summary

It has been shown that utilization of automatic adaptive refinement relaxes the quality requirements of the initial point distribution since any initial errors due to discretization may be later alleviated through improvements made to the local point structure. This ability, therefore, is critical to the realization of an industrially relevant solution methodology which significantly reduces human involvement. The research innovations which make this possible are the development of adaptive-friendly model generation processes, as well as the incorporation of analytical surface constrains into the boundary representation. Together, these techniques demonstrate a significant advancement to the current meshless state-of-the-art as they elevate the importance of adaptive refinement from a useful afterthought to a critical component of the method. In addition, by integrating local grid convergence criteria into the solution process the MIMS method is able to report grid converged results requiring no additional knowledge or input from the user. This represents a major advancement over current solution processes as the importance of demonstrating grid convergence is oftentimes overlooked by inexperienced users, resulting in poor solutions being inappropriately reported as final.

CHAPTER 8
SOLUTION IMPLEMENTATION DETAILS

Having fully described the components of the proposed meshless implementation in the previous four chapters, the final topics that need to be addressed are implementation details specific to the partial differential equations being solved. In particular, upwinding, selection of shape function (for particular node and derivative types), and the handling of shadow nodes in generalized coordinates, as applied to specific governing equation forms, will be discussed. Although not directly applicable to all application domains, discussing these topics is important in order to present a complete meshless solution process.

8.1 Upwinding

One area that meshless methods excels with respect to mesh-based methods is in development and implementation of appropriate upwinding procedures for convective derivatives. In conventional mesh-based methods, the existing connectivity explicitly defines the direction of information propagation throughout the domain, and as such, care must be taken to appropriately upwind convective terms in order to maintain a stable and accurate solution process. However, in meshless techniques, the freedom afforded by the underlying field approximations allows for evaluation along arbitrary directions, as well as at arbitrary distances. For example, if an upwinded first derivative in the x-direction is required at the highlighted node in Figure 8.1, one must consider both the case that the upwind direction

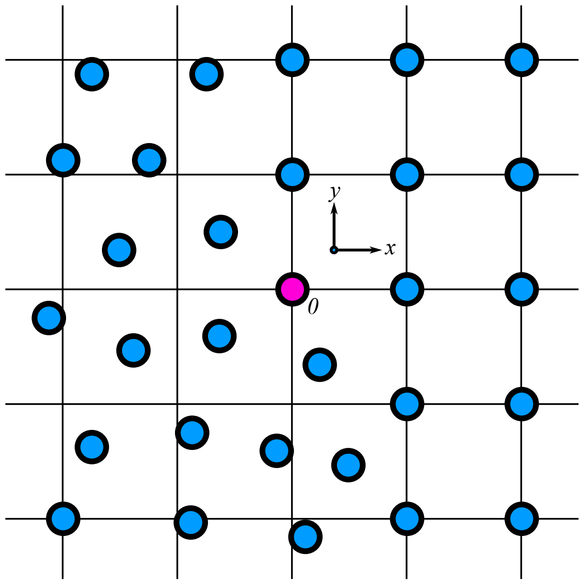


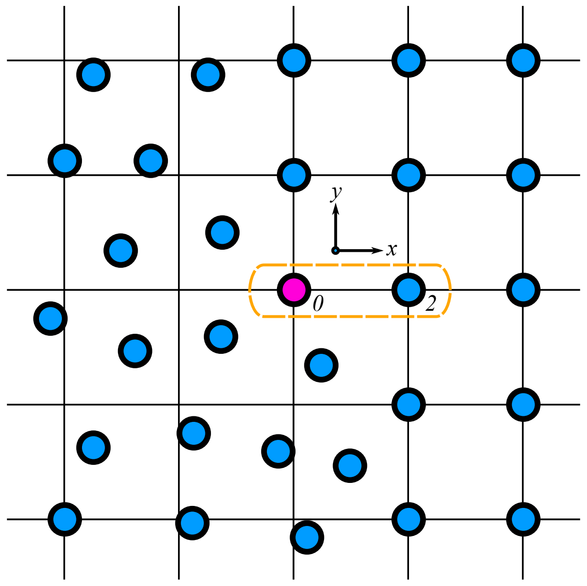
Figure 8.1: Sample Upwinding Point Distribution

is in the positive x-direction, and also the case when the upwind direction is in the negative x-direction.

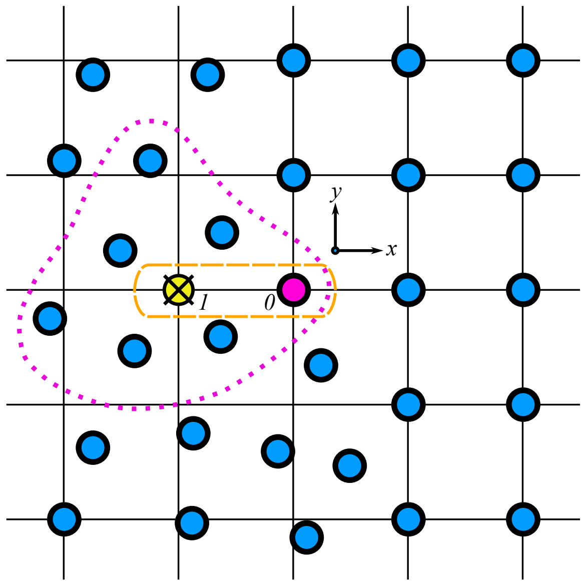
Fortunately, by employing meshless VML techniques, this is a fairly trivial process. First, because a real node exists in the positive x-direction, an appropriate upwinded operator is constructed via the topology shown in Figure 8.2a with the finite difference operator consisting of

$$\left.\frac{\partial u}{\partial x}\right|_{x^+} = \frac{u_2 - u_0}{\Delta x} \tag{8.1}$$

where u_0 and u_2 are the values at the nodes indicated by corresponding indices in Figure 8.2a. Second, because there exists no node at the required location for the backward difference operator, an appropriate meshless technique must be utilized. Employing the Virtual Finite Differencing concept, the negative x-direction upwind operator can be constructed by simply



(a) Forward Difference Topology



(b) Backward Difference Topology

Figure 8.2: Upwind Topologies

using MLS (or RBFP) to interpolate to the required location, and then finite differencing as

$$\left.\frac{\partial u}{\partial x}\right|_{x^-} = \frac{u_0 - u_1}{\Delta x} = \frac{u_0 - \underline{\Phi}\left(x - \Delta x, y, z\right)\underline{u}}{\Delta x} \tag{8.2}$$

where $\underline{\Phi}$ is the corresponding field interpolation basis functions. In this manner, any number of arbitrarily aligned upwind operators may be constructed, without requiring additional complexity in the underlying solution routines.

It should be noted that an alternate approach of generating upwinded operators using RBFP interpolation has been developed by Šarler et. al. [91, 99, 100], which involves projecting the central value into the upwind direction and evaluating at its new position, instead of the actual node location. Although this technique has demonstrated itself capable for a wide range of application domains, the projection process requires a rebuilding of the corresponding support domain whenever there is a change in the "upwind" coordinate direction (i.e. changing from negative x to positive x). In addition, because the projection is done during the solution process, the basis functions are no longer able to be combined into a single vector-vector multiplication, thereby increasing the computational expense of the routines. It is for these reasons that the VML technique will be adopted as the upwind process of choice for the current meshless implementation.

In addition to facilitating geometric upwinding, meshless methods also allow for natural implementations of more sophisticated upwinding schemes such as the Advection Upstream Splitting Method (AUSM) proposed by Liou and Steffen [103]. This method, which seeks to combine the accuracy of the Roe splitting method with the speed and simplicity of Van Leer Splitting schemes, provides a means of decomposing the convective flux vectors, E_c, F_c, and G_c, from the Navier-Stokes equations given as

$$\frac{\partial Q}{\partial t} + \frac{\partial E_c}{\partial x} + \frac{\partial F_c}{\partial y} + \frac{\partial G_c}{\partial z} = \frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y} + \frac{\partial G_v}{\partial z} \tag{8.3}$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_t \end{bmatrix} \quad E_c = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ (\rho e_t + p)\, u \end{bmatrix} \quad F_c = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v w \\ (\rho e_t + p)\, v \end{bmatrix} \quad G_c = \begin{bmatrix} \rho w \\ \rho w u \\ \rho w v \\ \rho w^2 + p \\ (\rho e_t + p)\, w \end{bmatrix}$$

$$E_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ E_v^{'} \end{bmatrix} \quad F_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ F_v^{'} \end{bmatrix} \quad G_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ G_v^{'} \end{bmatrix}$$

and $E_v^{'} = u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x$, $F_v^{'} = u\tau_{yx} + v\tau_{yy} + w\tau_{yz} - q_y$, and $G_v^{'} = u\tau_{zx} + v\tau_{zy} + w\tau_{zz} - q_z$. The reader is referred to the original paper by Liou and Steffen for full details, however, it is worth presenting a brief overview of the method in order to illustrate the implementation within the context of this work. The technique begins by recognizing that each of the convective flux vectors, E_c, F_c, and G_c are comprised of both convective and pressure terms such that any of these vectors may be decomposed as (using E_c for example):

$$E_c = u \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ (\rho e_t + p) \end{bmatrix} + \begin{bmatrix} 0 \\ p \\ 0 \\ 0 \\ 0 \end{bmatrix} = E_i^{(c)} + \begin{bmatrix} 0 \\ p \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8.4}$$

From this point, Liou and Steffen postulate that the true convective terms, $\Psi_i^{(c)}$, are convected by some "suitably defined velocity", while the pressure terms are "governed by the acoustic wave speeds" (where $\Psi_i^{(c)}$ can refer to any of the three convective flux vectors). Thus, AUSM provides a means to properly separate the components and determine the correct direction of information propagation. The "true" convective terms are manipulated by factoring u, then multiplying and dividing by the local sound speed, a, which produces:

$$E_i^{(c)} = u \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ (\rho e_t + p) \end{bmatrix} = M \begin{bmatrix} \rho a \\ \rho a u \\ \rho a v \\ \rho a w \\ a\,(\rho e_t + p) \end{bmatrix} \tag{8.5}$$

where M is the signed directional mach number, computed as $M = u/a$. It is at this point that Liou and Steffen separate the convective terms into $\pm 1/2$ components, corresponding to the interfaces between neighboring nodes:

$$E^{(c)}_{1/2} = M_{1/2} \begin{bmatrix} \rho a \\ \rho a u \\ \rho a v \\ \rho a w \\ a\left(\rho e_t + p\right) \end{bmatrix}_{L/R} \tag{8.6}$$

where $M_{1/2}$ is computed according to the following:

$$M_{1/2} = M^-_L + M^+_R \tag{8.7}$$

and

$$M^\pm = \begin{cases} \pm\frac{1}{4}\left(M \pm 1\right)^2 & \text{if } |M| \leq 1 \\ \pm\frac{1}{2}\left(M \pm |M|\right) & \text{otherwise} \end{cases} \tag{8.8}$$

Having defined the convective terms properly, the pressure terms may be split in a similar manner:

$$p_{1/2} = p^+_L + p^-_R \tag{8.9}$$

where

$$p^\pm = \begin{cases} \frac{p}{2}\left(1 \pm M\right) & \text{if } |M| \leq 1 \\ \frac{p}{2}\left(M \pm |M|\right)/M & \text{otherwise} \end{cases} \tag{8.10}$$

Finally, the convective derivative at node i is found by central differencing the half node values:

$$\left.\frac{\partial E_c}{\partial x}\right|_i = \frac{E_{i+1/2} - E_{i-1/2}}{\Delta x} \tag{8.11}$$

Separated in this manner, each component is functionally upwinded using a "Mach-number-weighted average", as well as an appropriate pressure splitting weighting technique. Because this formulation technique employs its own blended finite difference operators, the Virtual Finite Differencing procedure is utilized to obtain necessary quantities for Eqs. (8.7)-(8.11) at the left and right neighbors of the half nodes. In this respect, the meshless field interpolations can be used to generate localized interpolators at the necessary locations, and, rather than precompute upwind operators directly for each direction, Eq. (8.11) is applied to each convective term (with $E_{i+1/2}$ and $E_{i-1/2}$ utilizing the prebuilt localized interpolators) to determine the appropriate values for the convective derivatives.

The ability to generate accurate approximations for the underlying solution field allows collocation-based meshless methods to define arbitrary directional derivatives which can be easily adapted to handle any form of point-based upwinding scheme. In this respect, it provides a natural implementation path for any solution technique designed for finite difference applications, allowing it to draw upon a wealth of previous research within this field.

8.2 Shape Function Selection

The selection of appropriate shape functions is a critical component of a well tuned meshless implementation. Although any of the generation techniques described in Chapter 4 may be used without special consideration (with the exception of direct MLS differentiation), deciding the most appropriate situations for each requires some discussion. Before presenting the specific guidelines used in this work, the advantages and disadvantages of each technique have been summarized in Table 8.1.

As the goal of this work is to produce a robust meshless implementation, focus is placed on stability, as opposed to accuracy, when selecting shape functions for each particular node. In particular, a technique should be chosen because it is both numerically stable and capable of increasing its accuracy through local refinement of the node distribution, since the adaptive procedures described in Chapter 7 will be utilized to improve the geometric discretization. Understanding this fact, it becomes clear that the most general purpose shape function technique is VML utilizing an MLS approximation. Not only does this technique allow for straightforward upwinding, but it also provides excellent stability due to the MLS smoothing properties. That being said, the ability of direct RBFP differentiation to produce accurate solutions with minimal support domains, makes them an ideal candidate for use in areas lacking sufficient information to form an appropriate MLS topology (without introducing unwanted geometric diffusion). This situation most commonly arises when computing tangent operators for boundary nodes and for normal derivative computation for both boundary nodes (when shadow nodes are not present) and of the last layer of shadow nodes (where structured information is not present). As such, the process of selecting the

Table 8.1: Comparison of Meshless Shape Functions

Shape Function	Advantages	Disadvantages
MLS Approximation	- Capable of smoothing oscillations in field - Exact for represented monomial basis functions - Insensitive to evaluation location	- Does not satisfy Kroneker delta - Requires larger topologies than RBFP
MLS Differentiation	- Smooths oscillations in gradients	- Unstable without penalty function for boundary - Low order representation restricts derivative order
RBFP Interpolation	- Requires smaller topologies than MLS	- Requires expensive shape parameter optimization - Prone to oscillations between nodes
RBFP Differentiation	- Compact support domain (minimal smearing) - High accuracy at node locations if tuned appropriately	- Accuracy suffers for non-symmetric derivatives - Oftentimes requires under-relaxation to converge
VML utilizing MLS	- Highly stable due to MLS smoothing - Capable of isoparametric differentiation (no wasted effort)	- Requires large support domains - Prone to geometric diffusion
VML utilizing RBF	- Iteration speed is improved due to smaller topologies	- Prone to oscillation and error amplification due to oscillations between nodes in RBFP - Reduced order from underlying RBFP

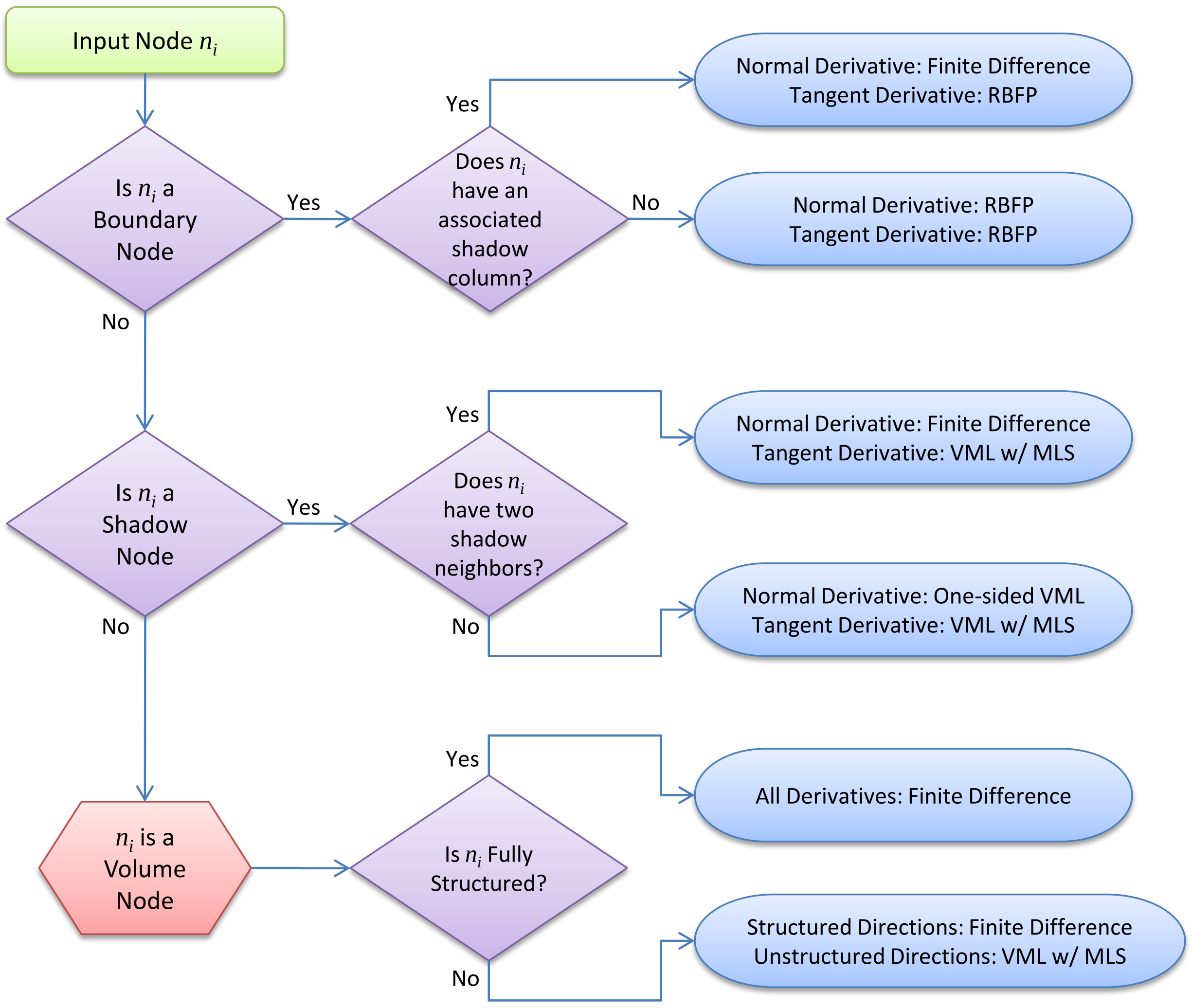

Input Node n_i
Is n_i a Boundary Node
Yes
No
Does n_i have an associated shadow column?
Yes
No
Normal Derivative: Finite Difference
Tangent Derivative: RBFP
Normal Derivative: RBFP
Tangent Derivative: RBFP
Is n_i a Shadow Node
Yes
No
Does n_i have two shadow neighbors?
Yes
No
Normal Derivative: Finite Difference
Tangent Derivative: VML w/ MLS
Normal Derivative: One-sided VML
Tangent Derivative: VML w/ MLS
n_i is a Volume Node
Is n_i Fully Structured?
Yes
No
All Derivatives: Finite Difference
Structured Directions: Finite Difference
Unstructured Directions: VML w/ MLS

Figure 8.3: Shape Function Selection Process

most appropriate shape function construction technique may be summarized by the flowchart shown in Figure 8.3.

8.3 Generalized Coordinate Systems

The final application specific consideration that needs to be made is selecting the most appropriate coordinate frame to solve the governing equations. For interior nodes generated via the Binary-Subdivision Interior Discretization described in Section 6.2, the choice is obvious: use a native Cartesian coordinate frame and utilize a meshless shape function when structured information is not present. However, for interior nodes generated in the shadow layer (shadow nodes), a decision must be made as to whether the problem should be solved in Cartesian space, or in a local, rotated space aligned with the associated boundary normal direction. Figure 8.4 demonstrates a simplified local, rotated space, for a representative boundary and associated shadow nodes. Obviously, any high normal gradients present in a boundary layer will be better captured if the differential operators are constructed such that they are normal aligned (since there is a considerable amount of structured information in that direction). Since we wish to take advantage of as much structure as possible, there are two ways to approach this problem; the first is to apply a standard coordinate transformation process whereby the Jacobian and other necessary transformation metrics are precomputed for each node and combined with the shape functions to obtain a compact form of the resulting Cartesian differential operators. The second technique is to directly work in generalized coordinate space, and thus, all operators are left in the local coordinate system of the shadow nodes and the associated metrics are handled directly in the governing equation formulation. Although either technique is acceptable, through experience it has been found that working with the governing equations in generalized form is oftentimes the most straightforward technique, since many simplifications can be made due to the fact that the local coordinate space is only a rotation of the Cartesian directions, and contains no stretching. However, the choice to rotate the operators or work in generalized coordinate space is not as important as acknowledging the structure that is present in the domain and ensuring that the developed routines are able to utilize it to its advantage.

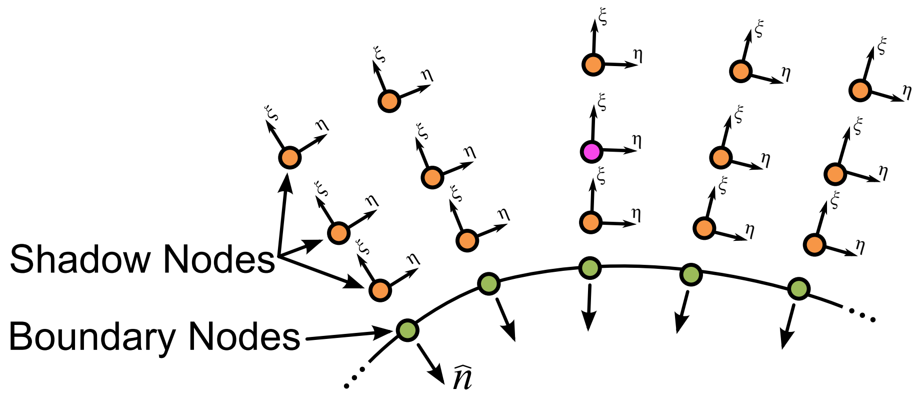


Figure 8.4: Boundary Aligned Shadow Coordinates

It is also important to note that in order to provide proper minimal support domains, the process described in Section 5.2 should construct ellipsoidal topologies that are aligned with the rotated coordinate frame of each shadow node. By aligning the topologies with the local coordinate system, more appropriately shaped influence regions can be constructed, resulting in less numerical dissipation and more accurate solutions.

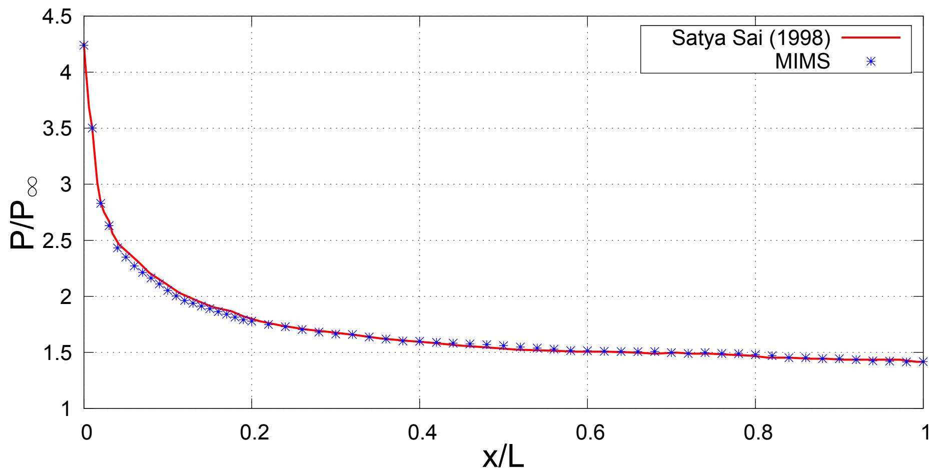
CHAPTER 9
VERIFICATION RESULTS

The presented meshless methodology is unique in that is represents a departure from previous focus, which primarily concerned itself with shape function development, and instead focuses on model generation and robustness issues specifically pertaining to meshless techniques. It is important to understand that although several critical advancements have been developed (such as the automated support domain construction and shape function selection process) and important results presented (such as the numerical stability analysis), the foundation of the methodology is built upon proven shape function generation techniques (such as RBFP and MLS). The true advancing research is the development of model generation and adaptive refinement algorithms tailored to the meshless procedures, and a realization that in order to be competitive with existing techniques, meshless methods must take advantage of the liberties afforded to them by the elimination of the structured connectivity throughout the domain. As such, it makes sense that to demonstrate the capabilities of the proposed technique, realistic problems must be completed, and compared against existing commercially available engineering analysis packages. Before presenting several challenging real-world problems, which demonstrate the full capabilities of the method, a few classical benchmark problem will be provided to verify the accuracy of the underlying solution techniques. Note that some of these results have been presented at several international conferences [101, 102] as part of this research.

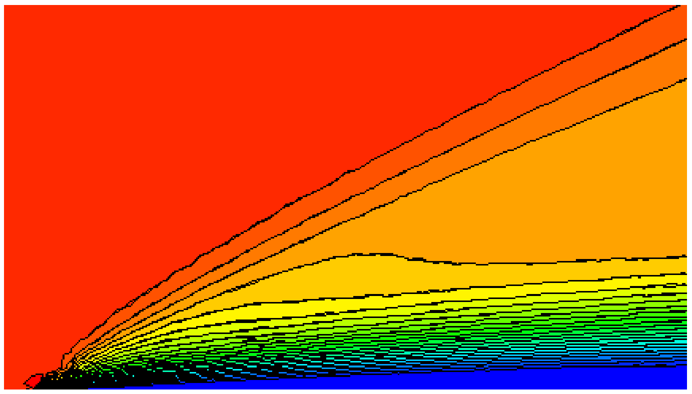
9.1 Supersonic Flat Plate

The first demonstrating test case is the classic problem of flow over a flat plate. In this example, a flat plate of unit length was modeled, with the inlet at a distance of 0.2 units from the front of the leading edge and the vertical boundary at a distance of 0.8 units from the flat plate. The plate was assumed to be infinite in length, as the outlet immediately followed the unit length plate. The inlet conditions imposed include a Reynolds number of $Re_\infty = 1000$, and a Mach number of $M_\infty = 3.0$. For this particular problem, an initial distribution was created and the solution was allowed to refine three levels, resulting in a final point distribution consisting of approximately 200,000 nodes. Marshall [150] presented results to a similar problem which were verified against data obtained by Satya Sai et al., which will serve as a comparison for the results generated by the MIMS method.

Figure 9.1a illustrates the normalized pressure distribution along the length of the flat plate, as directly compared to the Satya Sai et al. results quoted by Marshall. These results demonstrate a very good correlation between the data obtained by Satya Sai et al. and the MIMS method, even in the high pressure region at the onset of the flat plate. It is important to note that the data points shown for MIMS have been interpolated, and do not represent actual node locations. To further illustrate the obtained solution, Figure 9.1b plots the obtained Mach contours near the onset of the flat plate. This figure illustrates that the shock is being accurately captured, as well as the resulting boundary layer downstream from the beginning of the flat plate. Note that although the results generated by Satya Sai et al.



(a) Pressure Distribution along Plate



(b) Mach Contours

Figure 9.1: Supersonic Flat Plate Results

were developed in two-dimensions, the meshless implementation is fully three-dimensional (Figure 9.1b is an illustrative two-dimensional slice of the domain).

9.2 Subsonic Smooth Expanding Nozzle

The second test case deals with modeling viscous flow through a simple smooth expanding nozzle. A two-dimensional depiction of the problem geometry is given in Figure 9.2, with an understanding that this problem was constant in the z direction, having a domain thickness

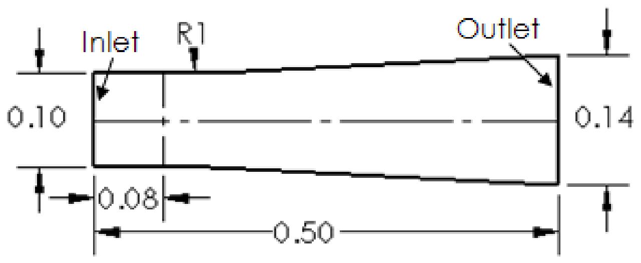
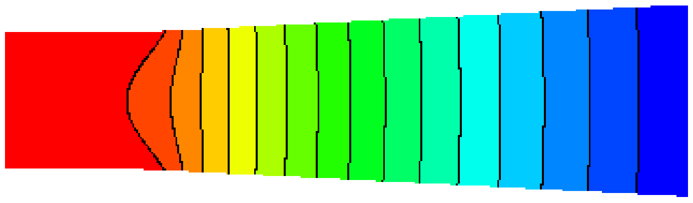


Figure 9.2: Smooth Expanding Nozzle Geometry

of 0.05m. The geometry of this problem is given in Figure 9.2 and will be used for both the case of subsonic and supersonic flow.

To impose subsonic flow conditions throughout the nozzle, an inlet Mach value of $M = 0.4$ was prescribed with a corresponding stagnation pressure and temperature of $P_0 =$ 100000Pa and $T_0 = 300$K, respectively. Additionally, all walls (other than inlet and outlet) were assumed to have no friction (slip walls). No refinement was performed on this model, in an attempt to isolate the solution process with respect to grid spacing. An initial meshless point distribution consisting of approximately 45,000 nodes was automatically generated for this geometry, and the results were compared to two-dimensional results generated by FLUENT (structured grid, approximately 70,000 two-dimensional cells). Shown in Figures 9.3a and 9.3b, are the Mach contours for the MIMS and FLUENT results, respectively, both plotted on a scale of $\Delta M = \{0.27, 0.4\}$.

Beyond the simple qualitative comparison, several pertinent quantities were also plotted along the mid line ($y = 0$, $z = 0.025$) and compared between the two techniques in order illustrate the solution quality. As evidenced by the comparison shown in Figures 9.4 and 9.5, there exists good correlation on both pressure and Mach number between the two solution


(a) MIMS Mach Contours

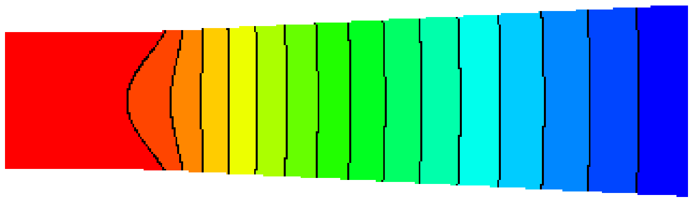

(b) FLUENT Mach Contours

Figure 9.3: Mach Contour Plots for Subsonic Smooth Expanding Nozzle

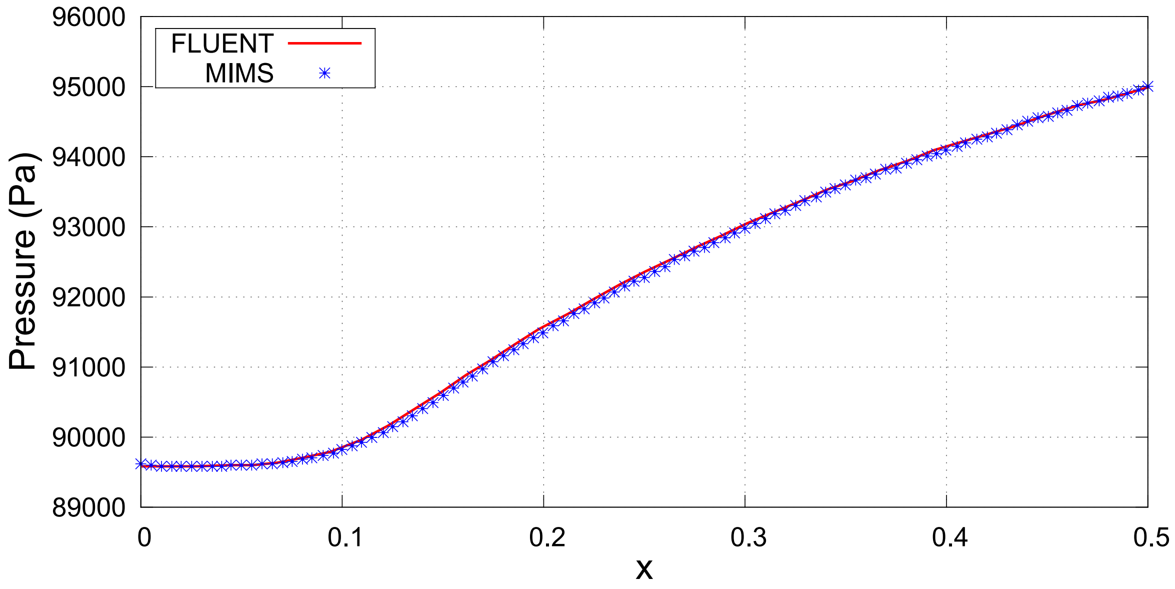


Figure 9.4: Midline Pressure Comparison for Subsonic Smooth Expanding Nozzle

techniques, with an maximum percent deviation between the two results of approximately 3% (relative to FLUENT results).

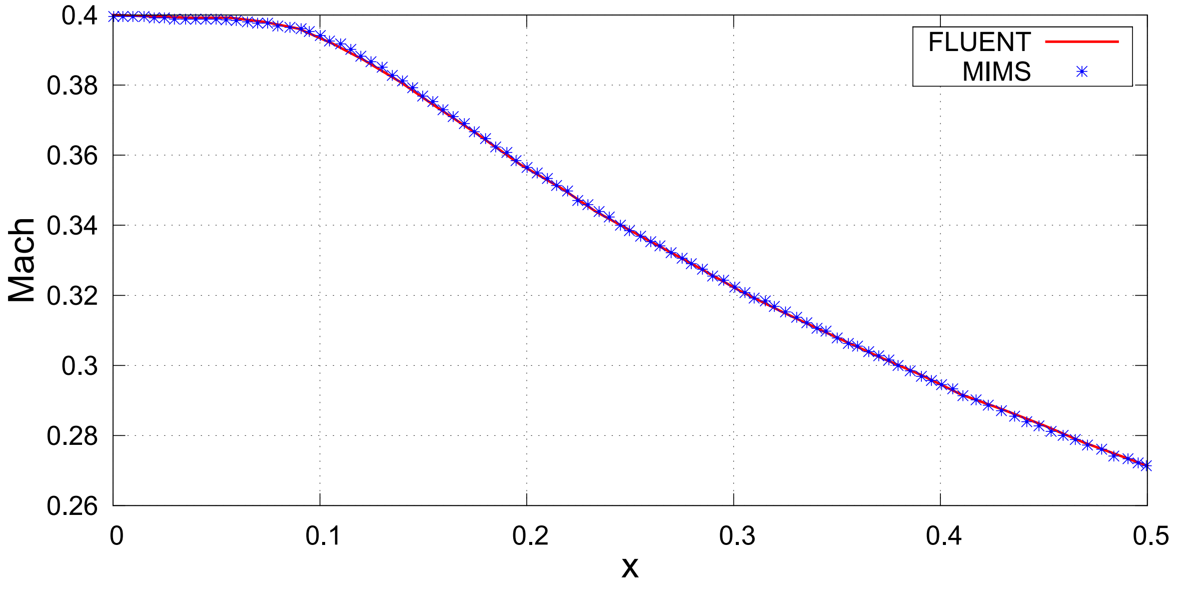


Figure 9.5: Midline Mach Comparison for Subsonic Smooth Expanding Nozzle

9.3 Supersonic Smooth Nozzle

The third test case deals once again with the smooth nozzle geometry presented in the previous section, except for this case supersonic flow was imposed, rather than subsonic. To generate a supersonic flow field, an inlet Mach number of $M = 2$ and a stagnation pressure and temperature of $P_0 = 100000$Pa and $T_0 = 300$K, respectively, were imposed. Additionally, all non-inlet and outlet walls were assumed to be friction free (slip walls).

Due to the steepness of the nozzle walls, the flow field exhibits a series of interacting compression and expansion waves within the nozzle. Although this indicates a poor nozzle design, it serves as an interesting test problem due to the multiple wave interactions which take place within the computational domain. For this particular problem, the meshless solution began with the same initial discretization of approximately 45,000 nodes and required three levels of refinement for grid convergence, resulting in a final grid consisting of approximately 160,000 nodes. Additionally, the results from this case were compared

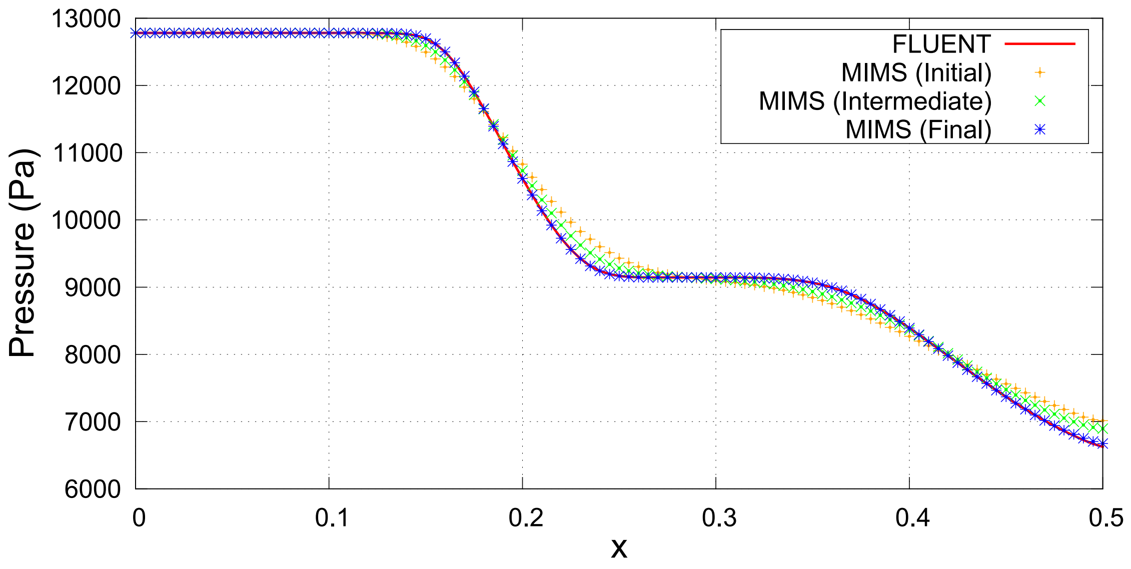


Figure 9.6: Midline Pressure Comparison

against a two dimensional solution generated by the commercial CFD package FLUENT, whose computational grid consisted of approximately 70,000 cells (which, in equivalent 3D, would correspond to more than 500,000 cells). Note that these point distributions were the result of 3 levels of refinement on both Mach and pressure gradients on the boundary and interior. To serve as a quantitative comparison, the pressure levels along a mid line ($y = 0$, $z = 0.025$) were compared to those obtained via FLUENT and are shown in Figure 9.6.

There is excellent agreement between the solutions obtained using FLUENT and the final refined point distribution solved using the meshless method techniques described herein. Additionally, this problem illustrates a major advantage of the proposed technique over other methods in that an initially poor discretization does not prevent the user from obtaining a good final solution.

9.4 Subsonic NACA-0012 Airfoil

The next case that was performed was that of a NACA-0012 airfoil at an angle of attack of $\alpha = 10\,\text{deg}$, placed in a subsonic flow with freestream conditions of $M_\infty = 0.8$ and Reynolds number of $Re_\infty = 500$. Although this particular case is highly unphysical due to the extremely low Reynolds number with corresponding Mach number, it does provide a non-turbulent flow field and is a common test case used to demonstrate proper flow characteristics. The results obtained are compared to data presented by Marshall and Ruffin [151], who in turn compares their data to Casalini and Dadone [152] with good correlation. Note that the computational geometry extends 2 chord lengths in front of the airfoil, 8 chord lengths behind the airfoil, and 6 chord lengths on the top and bottom of the airfoil. The initial MIMS point distribution consisted of approximately 160,000 nodes, and after three levels of refinement, the final point distribution consisted of just over 525,000 nodes.

Figure 9.7 begins by comparing the pressure coefficient, C_p, over the top and bottom surfaces of the airfoil with those obtained by both Casalini and Dadone, as well as Marshall and Ruffin. There exists a very good correlation between the obtained Meshless results and those presented by Marshall and Ruffin, which is understandable considering that they utilized a Cartesian based solver (NASCART-GT).

Another useful comparison is that of airfoil skin friction, C_f, obtained over both the top and bottom surfaces of the airfoil. Figure 9.8 presents the comparison of the results obtained using the new meshless method with the benchmark results previously mentioned.

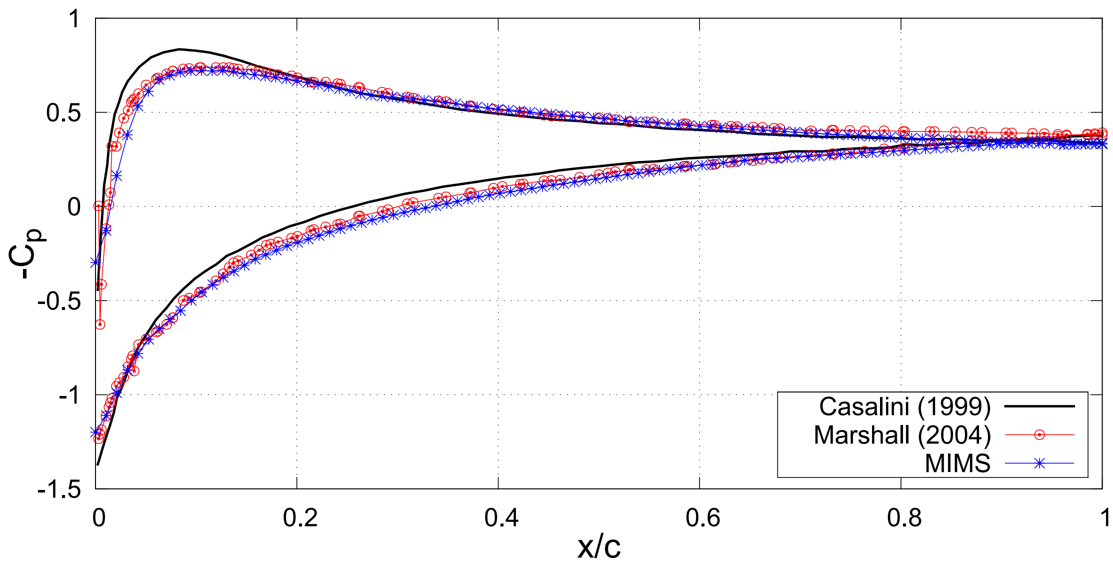


Figure 9.7: Subsonic Viscous NACA-0012 Pressure Coefficient

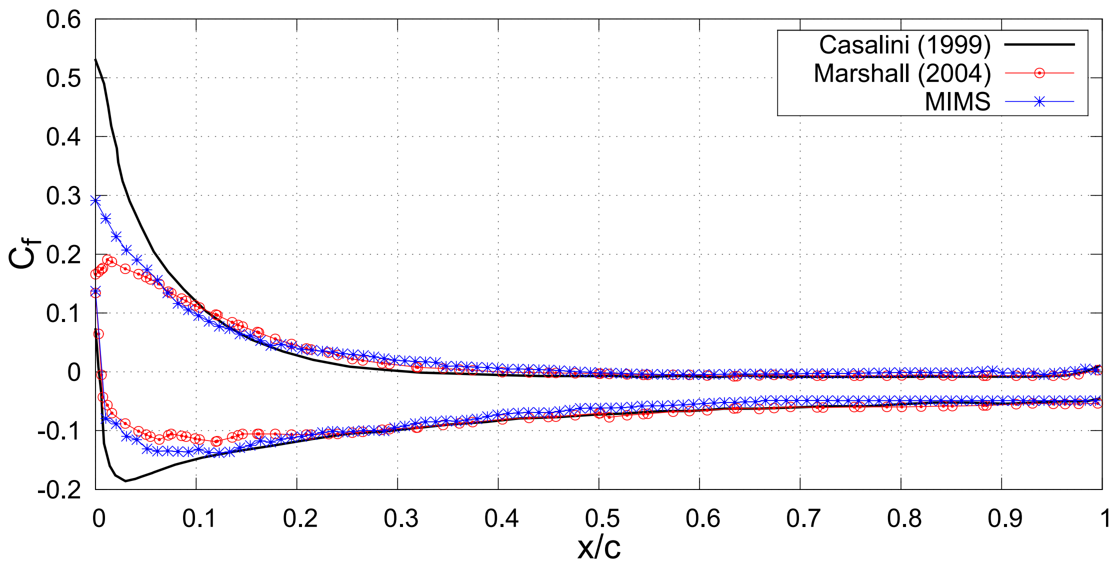


Figure 9.8: Subsonic Viscous NACA-0012 Skin Friction Coefficient

Once again, there exists very good agreement between data sets, with deviations mainly occurring in the high gradient stagnation region near the nose of the airfoil.

Finally, a contour plot illustrating the local airfoil region was generated, and is presented in Figure 9.9, which highlights the capturing of the trailing wake region, as well as the stagnation point at the nose of the airfoil.

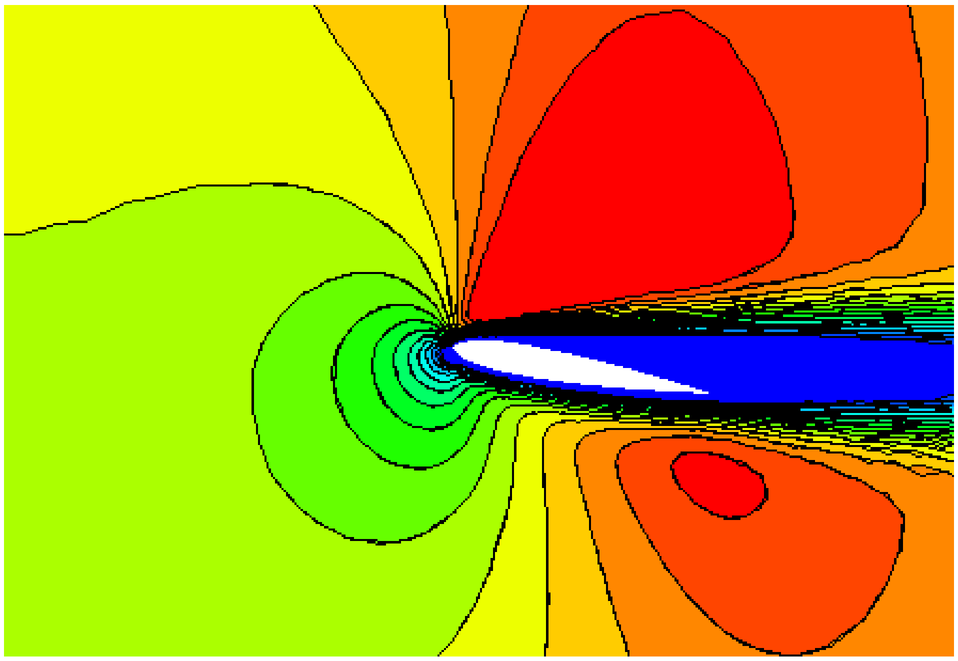


Figure 9.9: Mach Contours for Subsonic Viscous NACA-0012

9.5 Supersonic NACA-0012 Airfoil

The final test case performed, involved the same NACA-0012 geometry from the previous example, re-run under supersonic free-stream conditions of $M_\infty = 2.0$, and a Reynolds number of $Re_\infty = 1000$. As with the previous case, an initial distribution of approximately 160,000 nodes was used and allowed to refine three levels, and all geometric conditions remained the same.

Results presented by Marshall and Ruffin [151] were once again used as comparison, and Figure 9.10 shows the comparison of pressure coefficient along the upper and lower surfaces of the airfoil. There is again reasonable correlation between the two data sets, although there are some minor discrepancies throughout the chord length.

Figure 9.11 illustrates the comparison of skin friction along the top and bottom airfoil surfaces, and it is noted that there exists fairly substantial differences in values near the nose

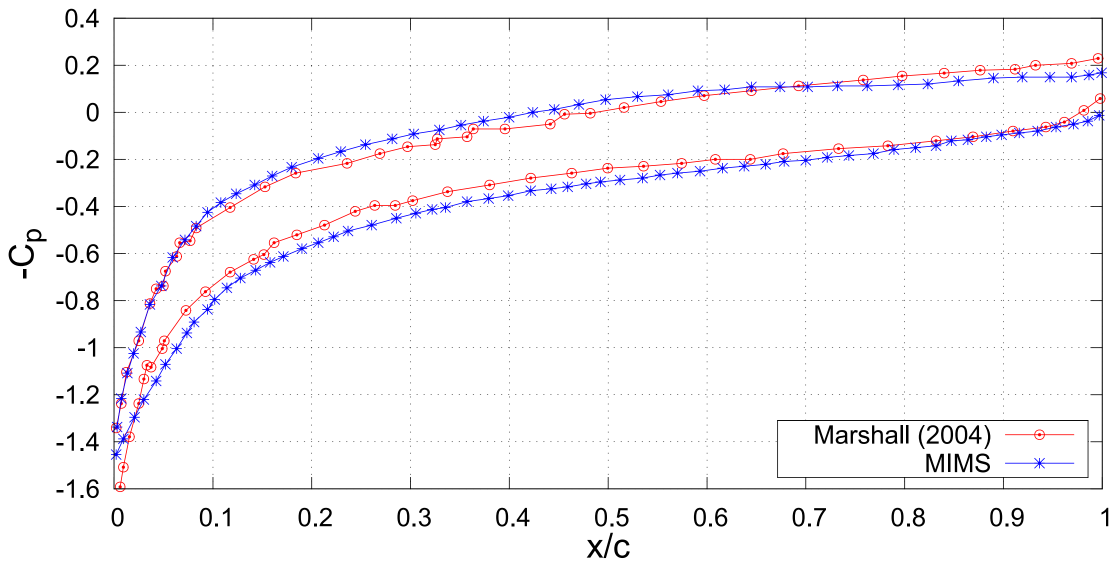


Figure 9.10: Supersonic Viscous NACA-0012 Pressure Coefficient

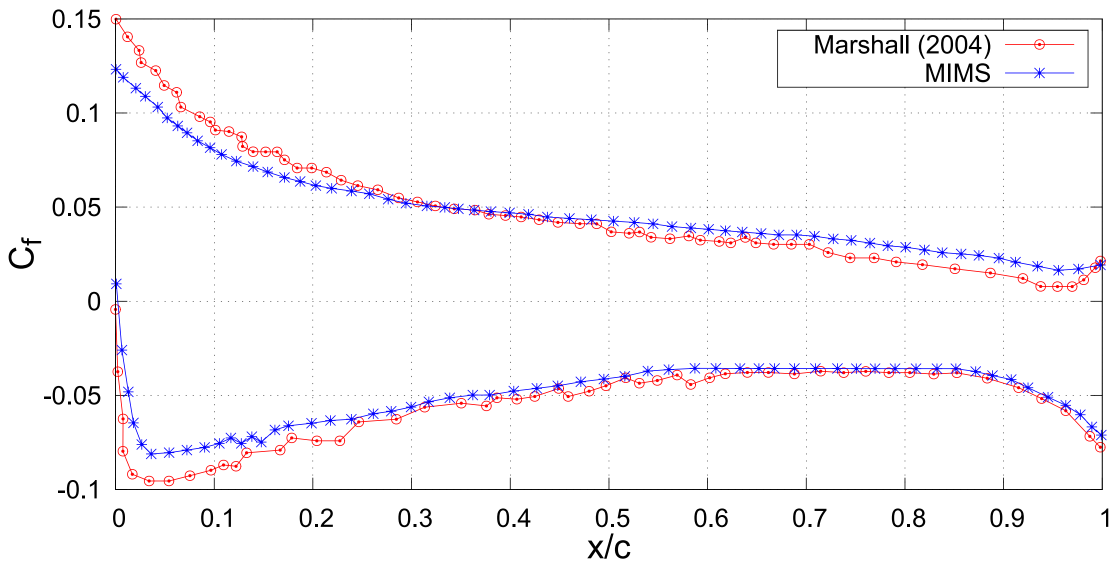


Figure 9.11: Supersonic Viscous NACA-0012 Skin Friction Coefficient

of the geometry. Finally, a plot illustrating the Mach contours is presented in Figure 9.12, illustrating the bow shock characteristics, as well as the flow behavior in the region close to the airfoil surface.

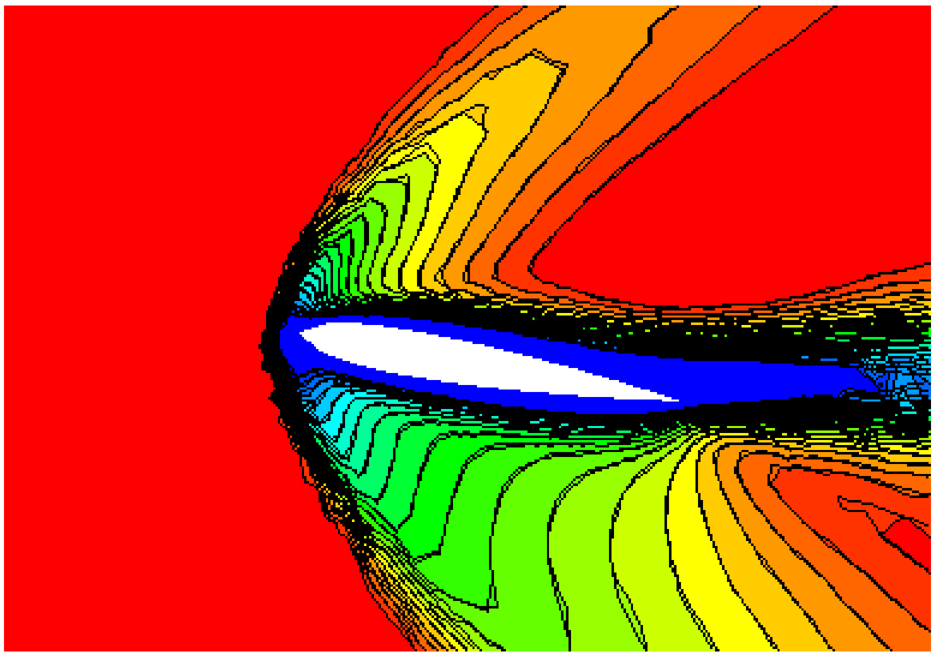


Figure 9.12: Mach Contours for Supersonic Viscous NACA-0012

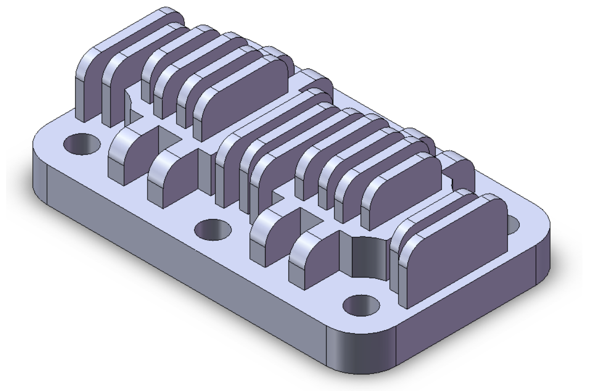
CHAPTER 10
CASE STUDIES

The previous chapter has provided several verification results that demonstrate the solution accuracy of the underlying meshless techniques with respect to several established benchmark and experimental results. However, the previous chapter did not sufficiently demonstrate the proposed advancements which make the MIMS methodology competitive on an industrial level. The primary focus of this research has been the development of the additional meshless point distribution and adaptive refinement procedures that can utilize the liberties of the solution process to obtain a robust, accurate, and easy to use engineering analysis tool. Ultimately, in order to be considered a competitor to more traditional solution techniques (such as finite element and finite volume methods), the developed methodology must not only demonstrate sufficient accuracy, but also a reduction in overall solution time (or, at least, a reduction in *human solution time*). Unfortunately, it is oftentimes difficult to quantify this value, since differently skilled users of commercially available codes will require vastly different times to complete a solution. In addition, each solution brings unique challenges to the user, and thus, is it difficult to make broad statements from a few representative problems. Nonetheless, in an attempt to demonstrate significant improvement over existing technologies, three sample problems have been developed which present common challenges encountered by engineers in real-world settings. To serve as a comparison, each problem has been solved with at least one commercially available software package by an experienced user. Rather than simply examine the final results, times and pertinent data (such as overall node counts) were recorded at all stages of the solution process (including mesh generation) in an attempt to quantify the overall benefits of the new meshless method. In this respect, these case studies present the complete solution process, from start to finish, that a well-trained engineer would undergo to arrive at an appropriate solution.

It is worth noting at this time that two commercially available analysis packages were utilized throughout these case studies, denoted COMMERCIAL 1 and COMMERCIAL 2. In addition, all meshing operations were performed via a commercially available grid generation software denoted GRID GENERATOR 1. All Microsoft Windows based solutions (MIMS and COMMERCIAL 1) were performed on a system consisting of a AMD Phenom X4 3.00GHz processor with 8GB of RAM, while all UNIX based solutions (COMMERCIAL 2 and GRID GENERATOR 1) were performed on a Xeon 3.06GHz processor with 4GB of RAM. Also note that although both systems offer multi-core/processor functionality, all timed operations were performed in single threaded mode to ensure an accurate comparison. Finally, it is worth mentioning that the author performed all necessary mesh generation and solution setup tasks for all three analysis packages (COMMERCIAL 1 & 2 as well as MIMS) in order to provide a fair comparison in skill level between each system. The author has been using GRID GENERATOR 1 to construct analysis meshes for approximately 5 years, and as such, can be considered quite knowledgeable in the area of mesh generation, representing an appropriately well-trained engineer.

10.1 Cylinder Head Heat Transfer

The goal of the first case study is to illustrate the advantages of the proposed meshless methodology over mesh-based techniques for a geometry that requires care during the mesh generation process. Although far from being considered complex, the simple two cylinder head shown in Figure 10.1 does represent a non-trivial geometry that could potentially be encountered in an industrial setting (for example, this could be a simplified representation of the head of a small piston compressor). Most notably, the geometry contains many small features in the form of heat sinks located on the top of the model which complicates the meshing process. Due to the complexity of the model, full dimensions would require considerable explanation (appropriate models may be obtained from the author upon request), though it is worth noting some important characteristics of the design. The total surface area of the model is 1217.17cm^2, the total surface area exposed to the internal cylinders is 78.54cm^2, the combined surface area of the six mounting holes is 56.55cm^2, and the surface area of the base is 186.69cm^2. In addition, the total volume is 650cm^3 and the overall dimensions are 20cm $\times$ 12cm $\times$ 6cm.


(a) Top Isometric

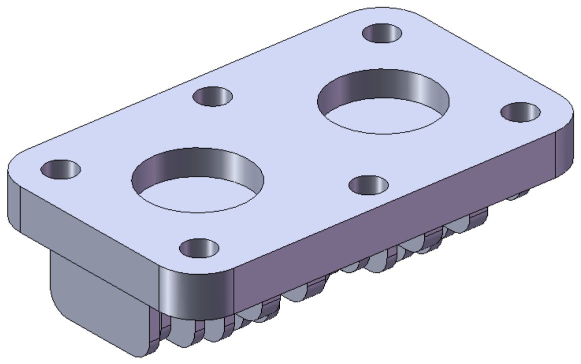

(b) Bottom Isometric

Figure 10.1: Cylinder Head

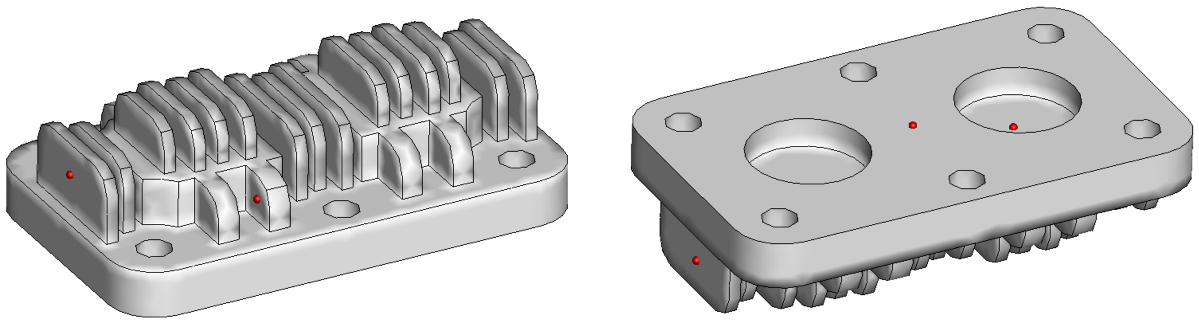
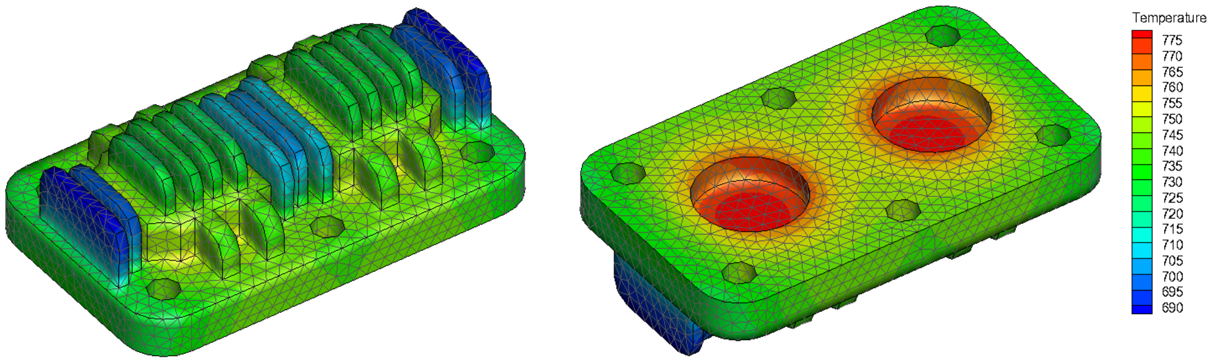

(a) Top Isometric (b) Bottom Isometric

Figure 10.2: Cylinder Head Measurement Points

To simulate the heat transfer occurring due to engine operation, the surface area exposed to the internal combustion chambers (two large cylindrical cutouts shown in Figure 10.1b) were provided a constant $100000\text{W}/\text{m}^2$ of input heat flux, the base of the model and the six mounting holes were insulated (simulating a thick, insulating cylinder head gasket), and the remaining surfaces were given convection conditions of $h = 20\text{W}/\text{m}^2 - \text{K}$ and $T_\infty = 100\text{K}$. The material selected for this model was high carbon AISI 1040 steel with material properties $\rho = 7840.0\text{kg}/\text{m}^3$, $C_p = 490.0\text{J}/\text{kg} - \text{K}$, and $k = 48.0\text{W}/\text{m} - \text{K}$. Given the geometry of the problem, several probe location were placed on the exterior of the model (to simulate accessible thermocouple measurement points). The measurement points were located on one of the exterior fins, the center of the base, the center of the combustion chamber cylinder, and the side of one of the support brackets, as denoted in Figure 10.2 as red points. These measurement points will serve as the quantitative comparison between the results generated via the MIMS method and COMMERCIAL 1 software package.

Beginning with COMMERCIAL 1, given the complexity of the geometry, it was decided by the operating engineer that an initial, uniformly distributed, unstructured tetrahedral mesh would be created to serve as a benchmark for analysis. As such, the initial mesh was

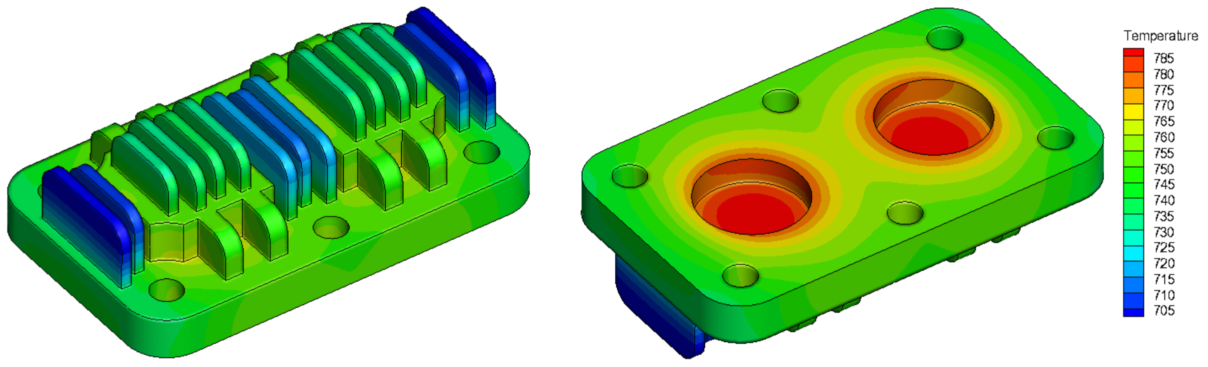


(a) Top Isometric (b) Bottom Isometric

Figure 10.3: Cylinder Head Initial Mesh and Temperature Result

created in GRID GENERATOR 1 with a discretization size of 0.65cm; a number determined through trial and error (during meshing) to produce an acceptable mesh with a minimum number of cells. The resulting mesh, which consisted of 16,108 cells (4,612 nodes) can be seen in Figure 10.3, along with a contour plot of the converged temperature field for this discretization.

Having established the initial solution, the engineer decided that due to the complexities in the geometry, it would be overly time consuming to produce a clustered mesh according to this temperature field. This was partially due to the complexity in the geometry, and partially due to the complexity of the field (in so much that it would be difficult to translate high field gradient locations to geometric edges and faces as the basis for mesh clustering). As such, a secondary mesh was produced with half the initial distribution with an average edge length of 0.325cm, resulting in a mesh with 106,548 cells (24,839 nodes). At this point the solution was once again solved to convergence. After analyzing the resulting field, it was found that the overall temperature span of the field changed from (688.4K – 780.1K) with the initial mesh to (698.1K – 789.6K) with the new, finer discretization. Understanding that



(a) Top Isometric (b) Bottom Isometric

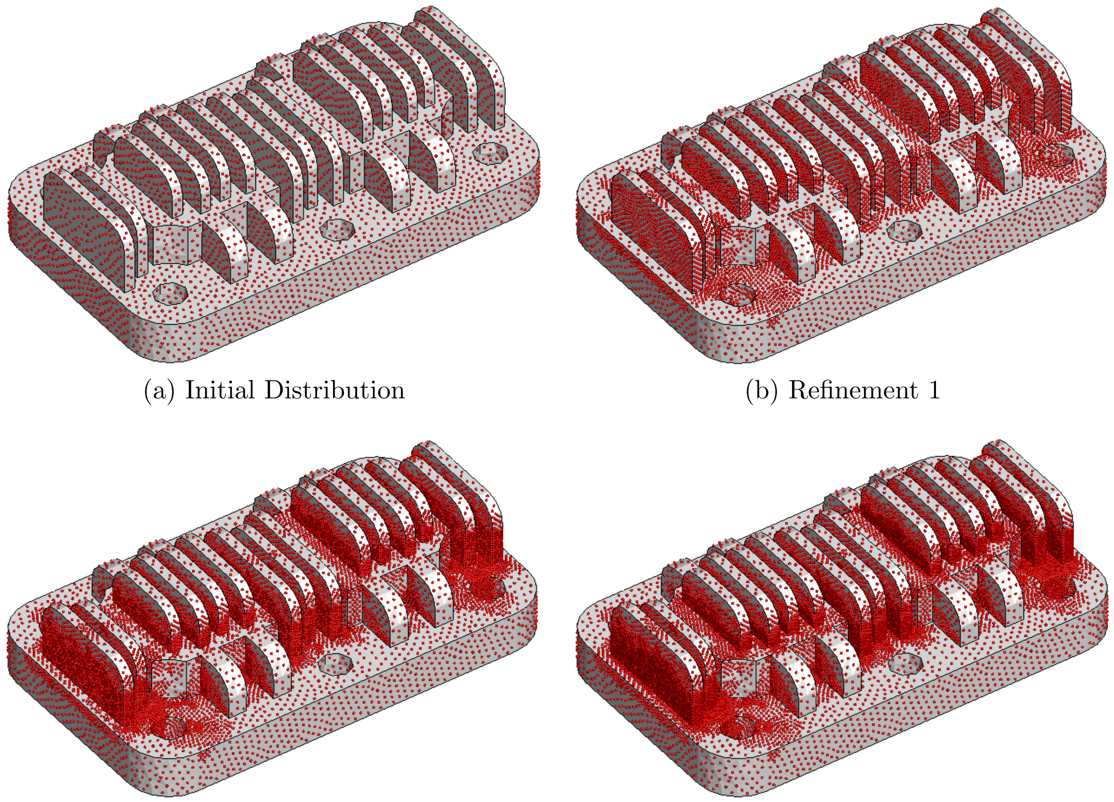
Figure 10.4: Cylinder Head Final Temperature Result

this could be a by-product of the coarse initial mesh, the engineer once again refined the distribution with a uniform average edge length of 0.20cm, resulting in a mesh consisting of 422,225 cells (88,301 nodes). This final mesh was once again run to convergence, and the resulting temperature range was found to be (699.0K – 789.8K); determining this to be an acceptable grid converged solution, the engineer reported his results as final. The final temperature field is illustrated in Figure 10.4.

It is important to note that because the commercial analysis package did not have remeshing capabilities, each time the engineer was required to construct a more refined mesh, he had to go back to the mesh generation package and remesh the existing geometry. This added additional human interaction time, and thus, increased the overall turnaround time on the analysis. In addition, because COMMERCIAL 1 was reloading the solution after each remesh, it was essentially having to start from scratch each time, which considerably increased the computation time for the final distribution.

The second approach used to solve this problem was to utilize the developed MIMS process and let the algorithms decide when and where to refine the point distribution. As with the mesh-based approach, the initial solution distribution consisted of an average surface edge length of approximately 0.65cm, however, unlike the mesh-based approach, this value was determined by the QTM model interpreter based on minimizing deviation from the underlying geometry. Once the initial solution had been found, the MIMS method continued to refine and solve the solution over three refinement levels until a local grid convergence value of 1% deviation with respect to the temperature span had been achieved. Visual representations of the boundary point distribution at each level of refinement are shown in Figure 10.5. The first thing to notice from these figures is that unlike the mesh-based process, the meshless method is able to locally refine in the areas of highest gradient (as opposed to global uniform reduction in mesh size). Thus, despite having performed three refinement levels, the final meshless node count was only 156,507 nodes, while the meshless node count at the previous refinement level was 88,082 nodes (which compares very closely with the node count in the final COMMERCIAL 1 mesh). In this respect, the final refinement level added a relative small percentage of nodes with respect to the overall volume (indicating the importance of local refinement at this level). The second important thing to note is that the geometric representation is improving as the solution is refined (notice the curvature of the mounting holes as the solution progresses). This is an important component that allows the solution process to perform initial discretizations with little regard to full approximation of the underlying geometry (it only needs to be good enough to obtain an accurate snapshot of the solution behavior).

By exposing the interior of the domain at the initial and final refinement levels, it can be seen in Figure 10.6 that the MIMS method was able to refine in small, local regions of the


(a) Initial Distribution

(b) Refinement 1

(c) Refinement 2

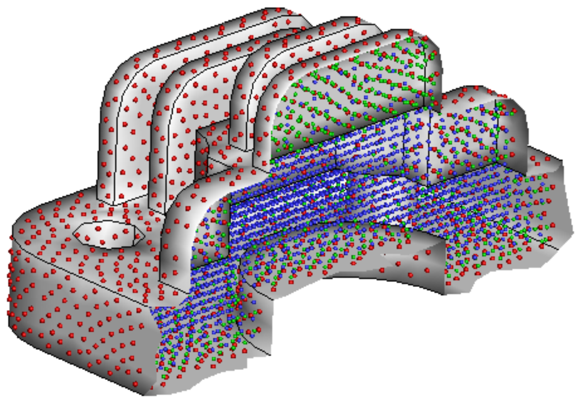
(d) Refinement 3

Figure 10.5: Meshless Cylinder Head Refinement Levels

domain without introducing instability (this figure illustrates boundary nodes as red spheres, shadow nodes as green spheres, and interior nodes as blue spheres). This is important as it demonstrates the robustness of the underlying shape functions as well as the support domain optimization process described in Section 5.2.

To serve as a quantitative comparison of the results generated using the MIMS method and COMMERCIAL 1, the values at the four probe locations were determined at each refinement level, and are listed in Table 10.1. As expected, both solution techniques provide roughly equivalent (less than 1% deviation) results at the final discretization levels.

Having demonstrated that the two solution processes are arriving at equivalent solutions, the major comparison remaining is solution time. Recall that one of the major arguments


(a) Initial Distribution

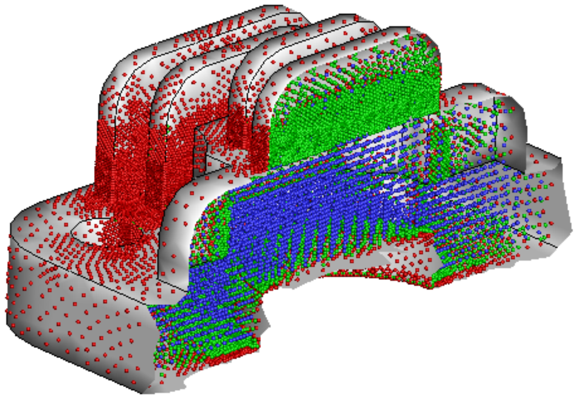

(b) Final Distribution

Figure 10.6: Meshless Cylinder Head Point Distribution

Table 10.1: Cylinder Head Probe Values

(a) MIMS

Discretization	Fin	Base	Chamber	Support
Initial Solution	720.23	758.32	714.16	714.16
Refinement 1	728.93	763.39	789.86	752.60
Refinement 2	728.85	763.39	789.86	752.62
Refinement 3	728.85	763.39	789.86	752.60

(b) COMMERCIAL 1

Discretization	Fin	Base	Chamber	Support
0.65cm	714.16	752.01	779.54	742.77
0.325cm	723.68	762.29	789.34	751.59
0.20cm	724.77	762.85	789.72	752.14

for using a meshless method is a reduction in analysis time. Thus, Table 10.2 lists a breakdown of the solution process times for both the MIMS method and the manually meshed process. Examining these times, it becomes clear that the meshless solution process requires considerably less time to complete. In fact, not only does it require only 5% of the total engineering time (which, after all, is the most expensive), it is also able to reduce the total solution time by nearly three hours. The dramatic difference in solution times may be explained by a combination of the MIMS method being able to start from a previously

Table 10.2: Cylinder Head Calculation Time Comparison

(a) MIMS

Task	Time
Problem Setup*	00:02:00
Initial Preprocessing	00:00:12
Total Solve Time	00:12:29
Total Refine Time	00:11:13
Total	**00:25:54**
***Total Engineer Time**	**00:02:00**

(b) COMMERCIAL 1

Task	Time
Mesh Setup (0.65cm)*	00:06:00
Problem Setup (0.65cm)*	00:05:00
Solve Time (0.65cm)	00:01:16
Mesh Setup (0.325cm)*	00:06:00
Problem Setup (0.325cm)*	00:05:00
Solve Time (0.325cm)	00:22:31
Mesh Setup (0.20cm)*	00:09:00
Problem Setup (0.20cm)*	00:05:00
Solve Time (0.20cm)	02:16:16
Total	**03:16:06**
***Total Engineer Time**	**00:36:00**

converged solution after each refinement, and by differences in the way the two algorithms reach a converged steady state solution.

Concluding this case study, it should be clear that both the total required time, and more importantly, the total engineering time spent solving this problem was significantly reduced when using the adaptive meshless method over the commercially available mesh-based approach. Although the final solution required more nodes for the MIMS approach, the automatic refinement procedure allows the solution to be explicitly reported as locally grid-converged; a statement that cannot be made about the alternative approach (since no formal grid convergence study was performed, which is a time consuming process and difficult to ensure in a local sense). This case study has also demonstrated the ability of the MIMS approach to handle quite complex geometry, an important consideration if it is to compete with more traditional techniques as a general solution process.

10.2 Cooling Jet Flow

The second case study represents the most complex field solution of the test cases performed, and also seeks to demonstrate the advantages of the MIMS method when problem accuracy is highly dependent on the underlying mesh quality. The problem geometry consists of a rectangular channel with three cooling jets angled to provide film cooling flow over the bottom, no-slip surface. A visual representation of the domain can be seen in Figure 10.7, with pertinent dimensions shown in Figure 10.8.

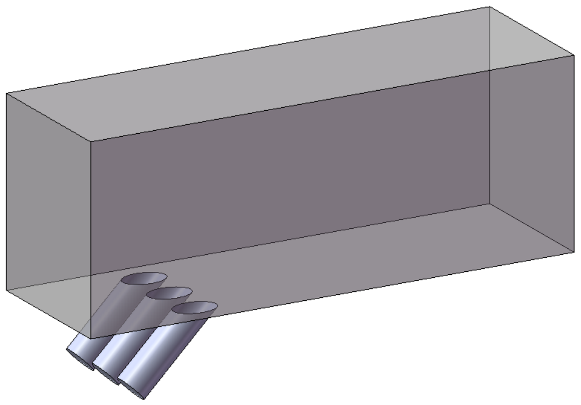


Figure 10.7: Cooling Jet Geometry

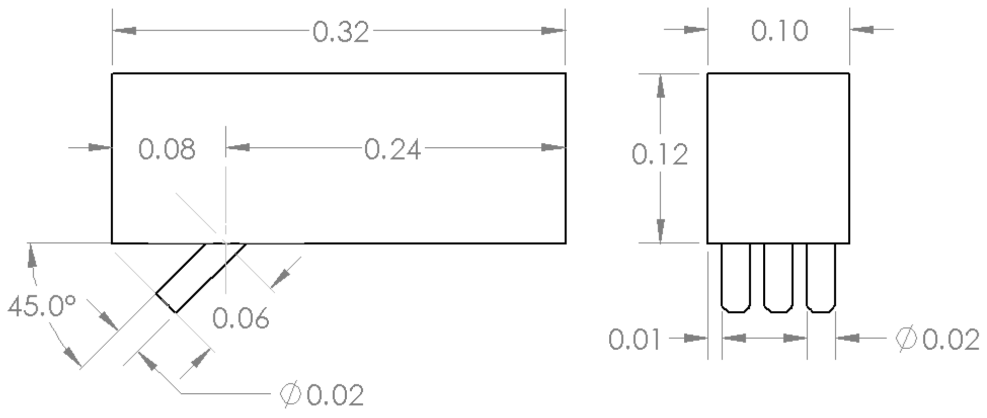


Figure 10.8: Cooling Jet Dimensions (m)

To simulate film cooling, the bulk flow inlet (located at the short end of the rectangular channel) was given conditions of $P_0 = 100,000\text{Pa}$ and $T_0 = 800\text{K}$, corresponding to an inlet Mach number of approximately $M = 0.4$. The outlet (located at the opposite end of the channel) was given an outlet pressure of $P_{out} = 70000\text{Pa}$, and the bottom surface of the channel was set as no-slip (to produce an appropriate boundary layer). The three inlet jets were given conditions of $P_0 = 150,000\text{Pa}$ and $T_0 = 500\text{K}$, resulting in a slightly increased injection speed of approximately $M = 0.68$. All other walls (including the cylindrical sides of the jet injections) were given full-slip (zero shear) boundary conditions. It is worth pointing out that the jets were angled 45 deg into the flow, to allow for a more natural transition for the incoming, cooled fluid into the boundary layer.

As this problem contains considerable complexity due to the injected flow and geometry, it is an ideal test case to demonstrate the advantages of the proposed meshless method over existing technologies. To serve as a proper illustration, two approaches were taken to solve this problem with existing software; first, a structured, highly formed mesh was generated using GRID GENERATOR 1 and solved via COMMERCIAL 2, and second, a clustered, unstructured mesh was generated (once again in GRID GENERATOR 1) and solved via COMMERCIAL 1. In this respect, both approaches to solving this problem (spend more time meshing and less time solving, or spend less time meshing and more time solving) can be examined, and the corresponding results will provide insight into the relative strengths and weaknesses of each approach when compared to the results generated via the MIMS method.

The first approach to solving this problem was to generate a structured mesh for the domain and solve using COMMERCIAL 2. Structured meshes have the advantage

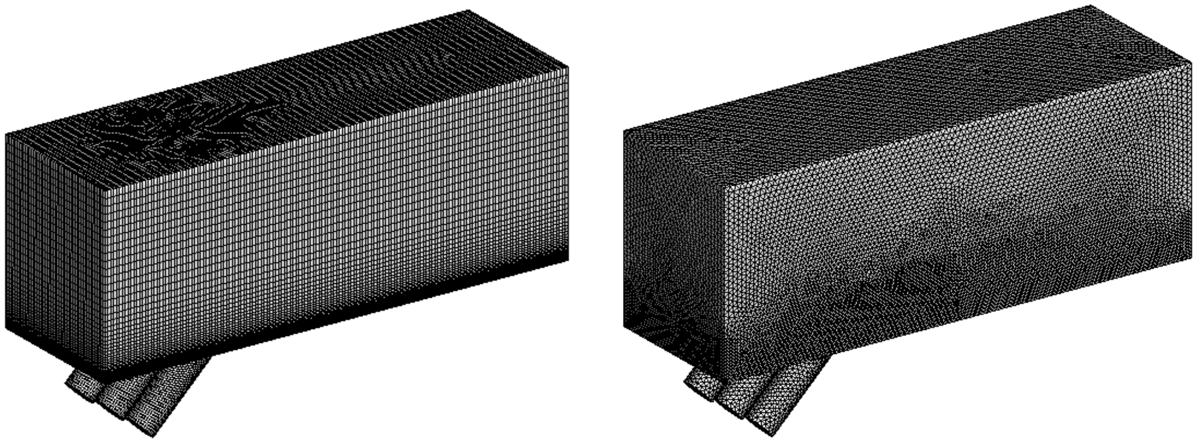

(a) Structured Mesh (COMMERCIAL 2) (b) Unstructured Mesh (COMMERCIAL 1)

Figure 10.9: Cooling Jet Meshes

of being able to maintain high aspect ratios, and thus, obtain high levels of accuracy with minimal computational effort (as illustrated in Figure 3.2c). However, they come at the cost of engineering time as they typically require a considerable effort on the part of a skilled engineer to generate an appropriate discretization for each problem. As is typical, the final mesh for this problem required roughly 2 hours and 15 minutes to produce, and resulted in 476,889 cells (499,350 nodes). It is also worth mentioning that this mesh required considerable expertise to generate (detailed knowledge of mapped-mesh generation techniques and geometric manipulation), and thus, would most likely not be obtainable by a user that was not proficient in using mesh generation software. A view of the final mesh that was supplied to COMMERCIAL 2 is shown in Figure 10.9a.

The second approach to solve this problem utilizes an unstructured mesh that is clustered near the regions where the highest gradients are expected. Unfortunately, unstructured meshes are unable to produce high aspect ratio cells without considerably sacrificing accuracy. For this reason, unstructured meshes usually require more cells to accurately solve a problem than their structured counterparts. This can be seen from Figure 10.9b, which shows the mesh generated for the cooling jet problem. While the structured mesh is able to generate a highly clustered boundary layer, the unstructured mesh is able to produce minimal clustering in this region; despite the lack of boundary layer refinement, the unstructured mesh still consists of 594,804 cells (113,593 nodes). Although the fidelity is not as good, the unstructured mesh only took 27 minutes to generate, a reduction of 80% from the time required to generate the structured mesh.

The final approach is to use the MIMS method to solve the problem and automatically refine the geometry in the most appropriate locations. In this respect, the meshless approach involves no initial mesh generation, while at the same time is able to appropriately cluster the discretization to capture the pertinent phenomenon. Because the initial distribution involves only 34,414 nodes, the solution is quickly able to reach an acceptable result which can be used as a guideline for refinement. In addition, because the process only refines where necessary, the technique is able to arrive at a comparable solution (after four refinement stages) with only 98,903 nodes, less than that of the unstructured mesh supplied to COMMERCIAL 1, and slightly less than one-fifth of the number of nodes used in the structured domain. As a comparison, the initial QTM discretization generated by the MIMS process is shown in Figure 10.10a and the final point distribution after four levels of refinement is shown in Figure 10.10b. Useful to notice is the high level of refinement that occurred at the jet entry point where the two flow streams begin to interact.

To visualize the results, all three models were plotted on the same temperature scale (360K to 780K with 22 levels) and an isosurface was generated at the $T = 600\text{K}$ value. The

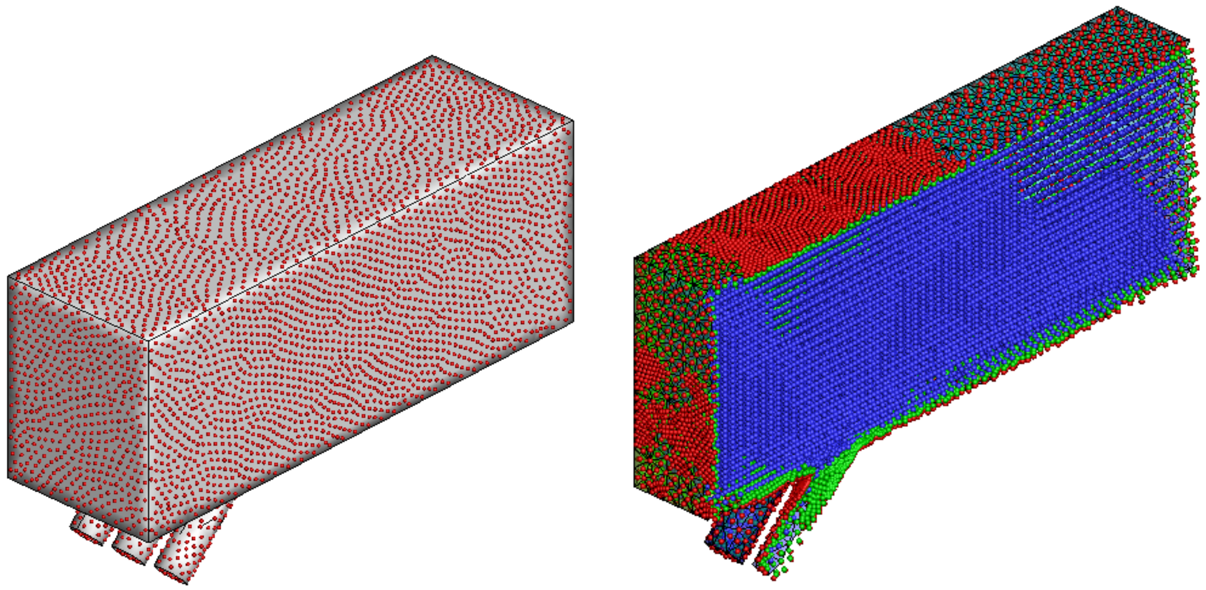
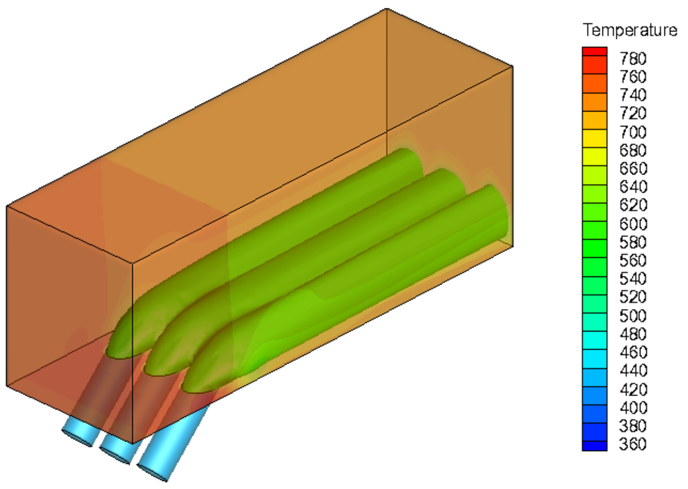
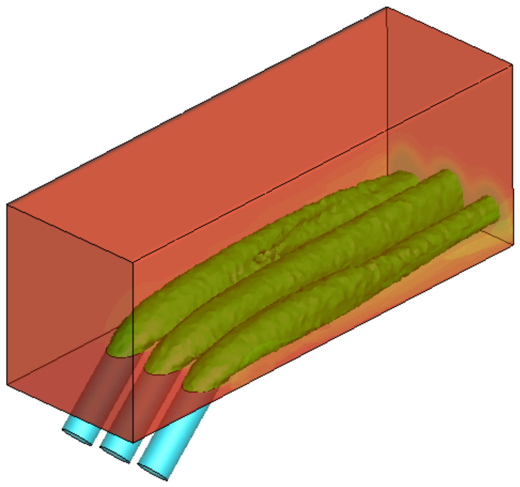

(a) Initial QTM Discretization (b) Final Point Distribution

Figure 10.10: Cooling Jet Meshless Point Distributions

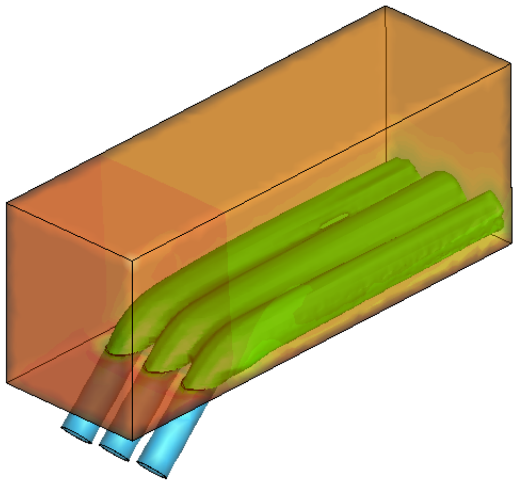
respective visualization results may be seen in Figure 10.11. Examining these figures, it becomes clear that the COMMERCIAL 1 results are considerably less smooth than either of the other two results. In fact, the results generated via MIMS appears to be in quite good agreement with the results obtained with COMMERCIAL 2 despite the considerably coarser resulting discretization. In order to provide a quantitative measurement of agreement, the values of temperature were extracted from the midline of the geometry (along the no-slip base), and are plotted in Figure 10.12. From this figure it appears that all three methods produce somewhat differing results in details, though the overall trends are in good agreement. In addition, from the onset of the cooling flow to the tail end of the domain, the MIMS results are between the COMMERCIAL 2 and COMMERCIAL 1 results, indicating that it is able to obtain accurate results despite the lack of underlying mesh.



(a) COMMERCIAL 2



(b) COMMERCIAL 1



(c) MIMS

Figure 10.11: Cooling Jet Temperature Contours ($T = 600\text{K}$ Isosurface)

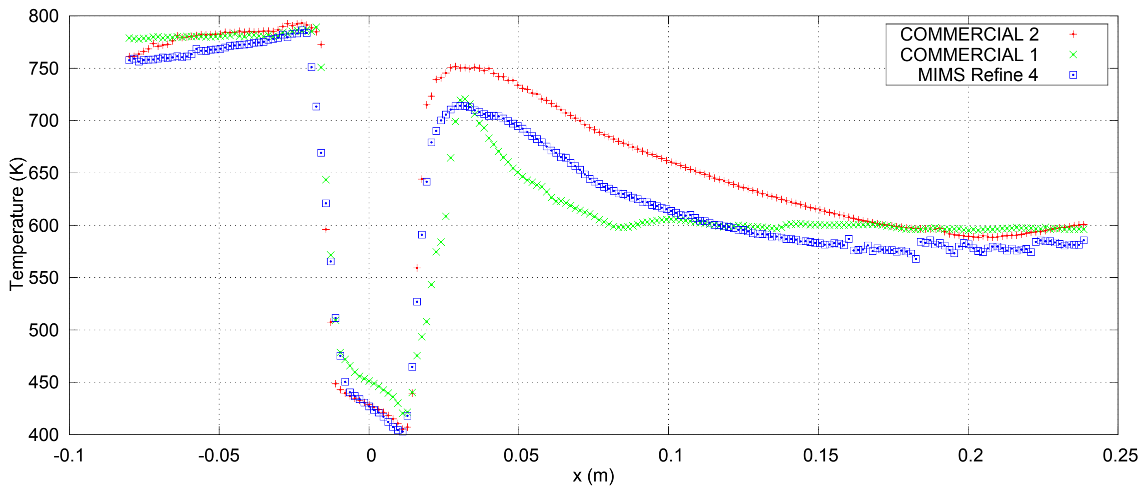


Figure 10.12: Temperature Along Cooling Jet Base

Table 10.3: Cooling Jet Calculation Time Comparison

(a) COMMERCIAL 2

Task	Time
Mesh Setup*	02:15:00
Problem Setup*	00:05:00
Solve Time	02:16:35
Total	**04:36:35**
***Total Engineer Time**	**02:20:00**

(b) COMMERCIAL 1

Task	Time
Mesh Setup*	00:27:00
Problem Setup*	00:05:00
Solve Time	03:32:02
Total	**04:04:02**
***Total Engineer Time**	**00:32:00**

(c) MIMS

Task	Time
Problem Setup*	00:05:00
Initial Preprocessing	00:02:18
Total Solve Time	00:27:31
Total Refine Time	00:29:26
Total	**01:13:15**
***Total Engineer Time**	**00:05:00**

As a final comparison for this problem, the respective solution times were recorded and are shown in Tables 10.3a-10.3c. Similar to the first case study, the MIMS process with its automatic refinement is able to complete the solution in considerably less time than either of the other two solution techniques. In addition, although the structured and unstructured mesh solutions were only 30 minutes apart, the structured mesh required almost two hours more engineering time than the unstructured. This, coupled with the fact that all three results generate comparable solutions, provides justification for the use of a MIMS approach when requiring quick analysis of components or flow fields.

Summarizing the findings of the second case study, we continue to see the MIMS method completing analysis in significantly less time without appreciably sacrificing accuracy. In addition, the fact that the meshless approach is locally grid converging the solution during refinement, gives confidence towards the results. Although one may argue that the increased fidelity afforded by the structured grid is necessary, it is difficult to justify the costs when considering that the entire MIMS process can be completed in half the time that it takes to generate the structured mesh. This makes the area of parametric design and analysis an ideal application for the MIMS method as frequent changes in model geometry can result in overwhelming engineering costs when attempting to solve each problem via a mesh-based approach.

10.3 Normal Shock Nozzle

The final test case seeks to demonstrate the accuracy and adaptability of the proposed technique when presented with sharp discontinuities in an underlying flow pressure field. To accomplish this, a nozzle presented by Hoffman [153] is solved where the cross sectional area is given as

$$S(x) = 1.398 + 0.347 \tanh(0.8x - 4) \tag{10.1}$$

and the nozzle inlet and outlet are located at $x = 0$m and $x = 7$m, respectively. The problem geometry can be seen in Figure 10.13, with the small end of the nozzle (inlet) located at $x = 0$m and the larger end (outlet) located at $x = 7$m. Assuming inlet conditions of $M = 1.5$, $P_0 = 100,000$Pa, $T_0 = 300$K, and $P = 27240.3$Pa, and an outlet pressure $P_{out} = 66809.6$Pa, there is a normal shock within the nozzle located at $x = 5$m.

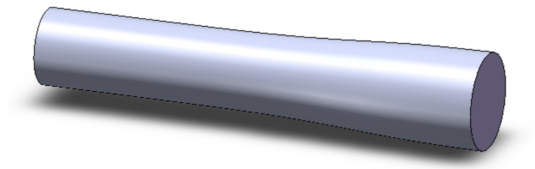
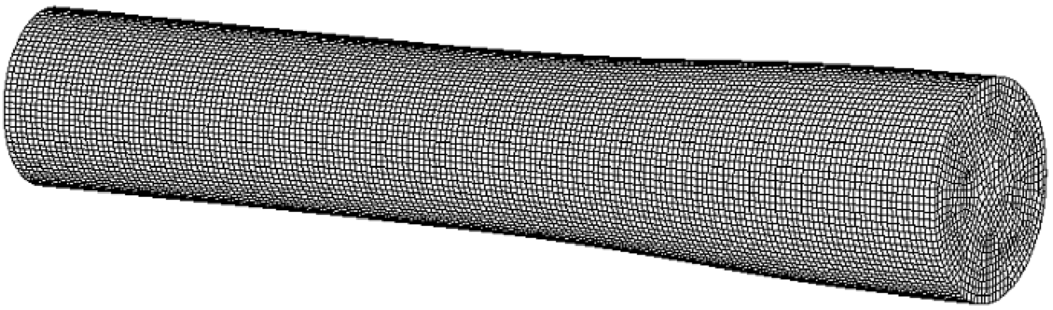


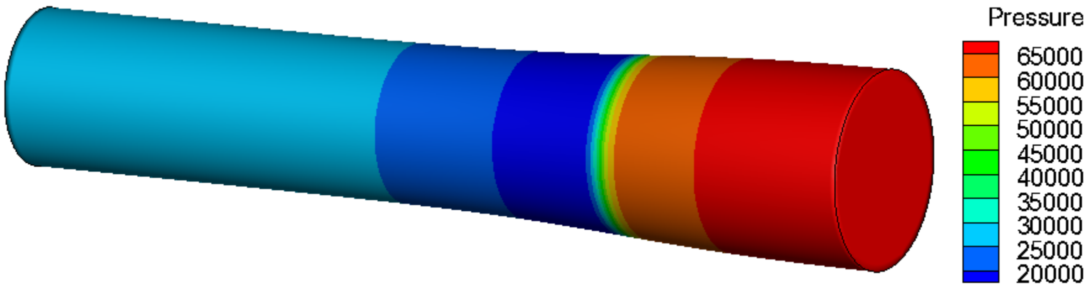
Figure 10.13: Normal Shock Nozzle Geometry

Although analytical solutions to this problem are provided by Hoffman, to provide a benchmark for arriving at the solution numerically, it was assumed that the problem geometry (in the form of an IGES file) and the appropriate boundary conditions were given to an engineer to solve using the COMMERCIAL 2 package. To simulate a real-world scenario, the operating engineer assumed no previous knowledge about the flow behavior prior to setting up the initial mesh. Therefore, to get an idea about the behavior of the solution, the engineer began by generating an initial, structured mesh consisting of 132,650 cells (141,680 nodes), at a uniform spacing of 0.02m. This mesh, and the corresponding solution can be seen in Figure 10.14. Having identified that a shock was present in the domain, the engineer then generated an appropriately clustered, structured mesh in order to capture the discontinuity as best as possible. This clustered mesh, which consisted of 415,450 cells (437,346 nodes), and the associated pressure distribution, can be seen in Figure 10.15.

To solve the problem using a MIMS approach, the domain is usually initialized from the IGES file and allowed to progress through alternating solution and refinement stages until grid convergence is achieved. However, this problem demonstrates an interesting pathological exception to the normal refinement strategy. Since the discontinuity in the field is an actual, physical compression shock (as opposed to an expansion or compression wave which has a finite width), it exists over a theoretically infinitesimal area. Therefore, when the meshless refinement strategies are utilized as previously described, the solution should be constantly improving, and as such, grid convergence should be theoretically impossible to achieve. However, despite this expectation, the solution process was reported as grid converged after

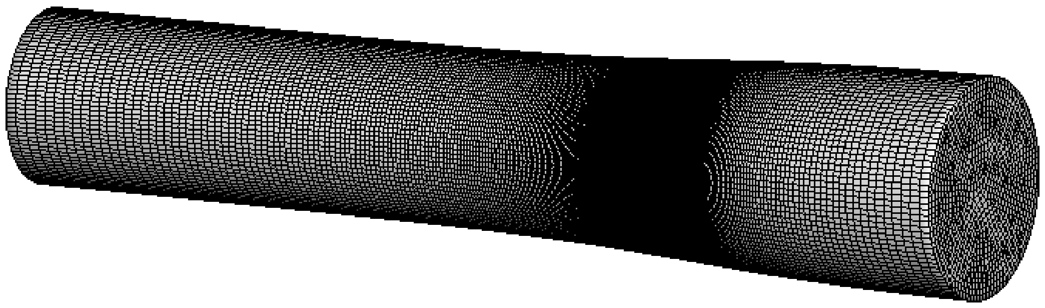


(a) Mesh

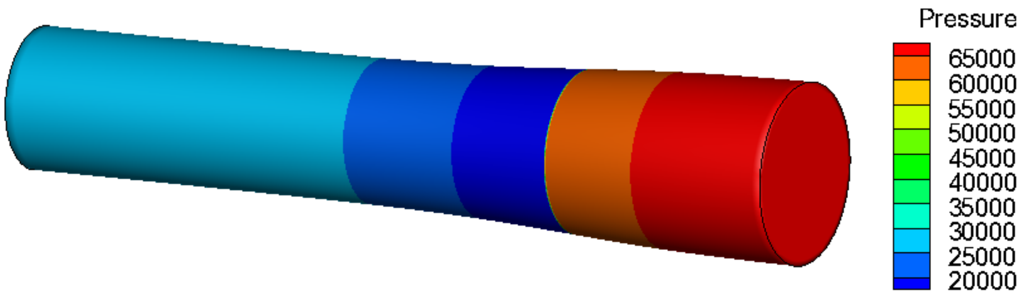


(b) Pressure Distribution

Figure 10.14: Initial Solution for Normal Shock Nozzle (COMMERCIAL 2)



(a) Mesh

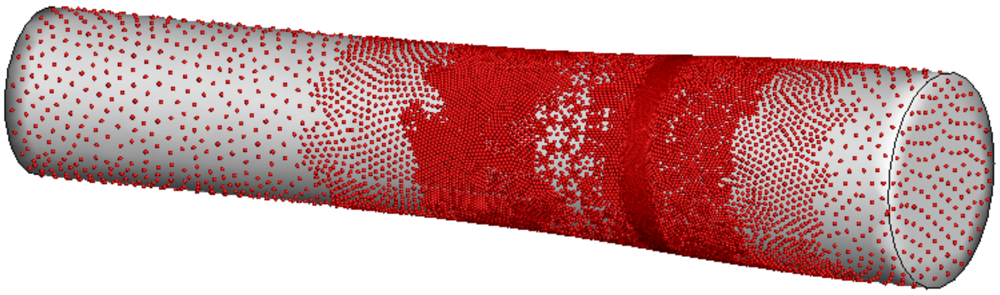


(b) Pressure Distribution

Figure 10.15: Final Solution for Normal Shock Nozzle (COMMERCIAL 2)

4 levels of refinement, with a final point distribution consisting of 88,429 nodes. After examining the pressure distribution results shown in Figure 10.16, and plotting the pressure down the nozzle midline (as shown in Figure 10.18), it becomes clear that the meshless results had indeed become grid converged, at least from the perspective that the solution did not change from one refinement level to the next; this is an undesirable result as it indicates that the grid accuracy is fixed by some upper bounds and further refinement will not improve the solution. However, what is actually occurring is that there is a discrepancy between the definition (and application) of grid convergence, since grid convergence and the associated refinement gradients are only measured between locally structured points; the size of the support domains are not considered. The support domains, do, however, define the local influence region, and as such, dictate the minimum distance over which a discontinuity can be captured. Therefore, although the meshless operators are causing the results to be smeared (much like finite difference operators smear over one grid point), they take longer for the solution to improve as the local domain is refined, and are causing the structured regions to appear as though they are grid converged.

Understanding that the lack of shock width fidelity is due to the underlying interpolation support domain size, the shock location is still captured very well, as shown in Figure 10.18. It is important to mention that Figure 10.18 was generated via a post-processed line interpolation, and as such, the point markers are shown for visualization purposes and do not indicate the locations of computational nodes.


(a) Surface Point Distribution

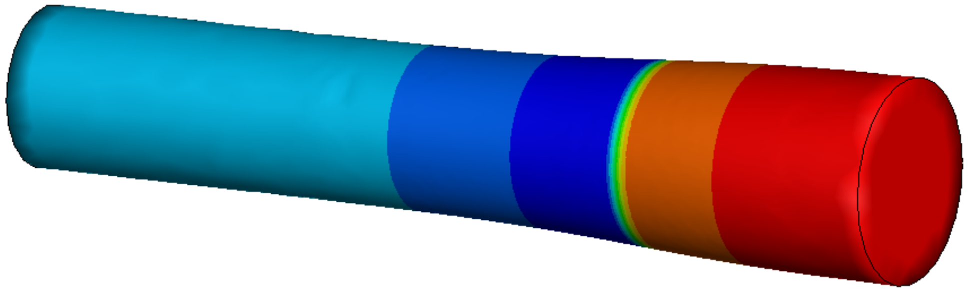

(b) Pressure Distribution

Figure 10.16: Final Solution for Normal Shock Nozzle (MIMS)

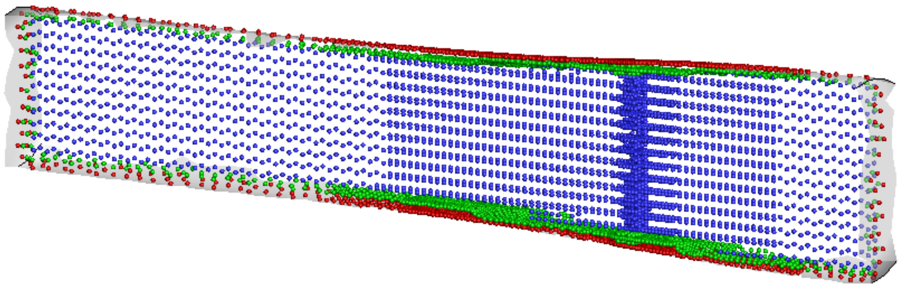


Figure 10.17: Normal Shock Nozzle Interior Point Distribution (MIMS)

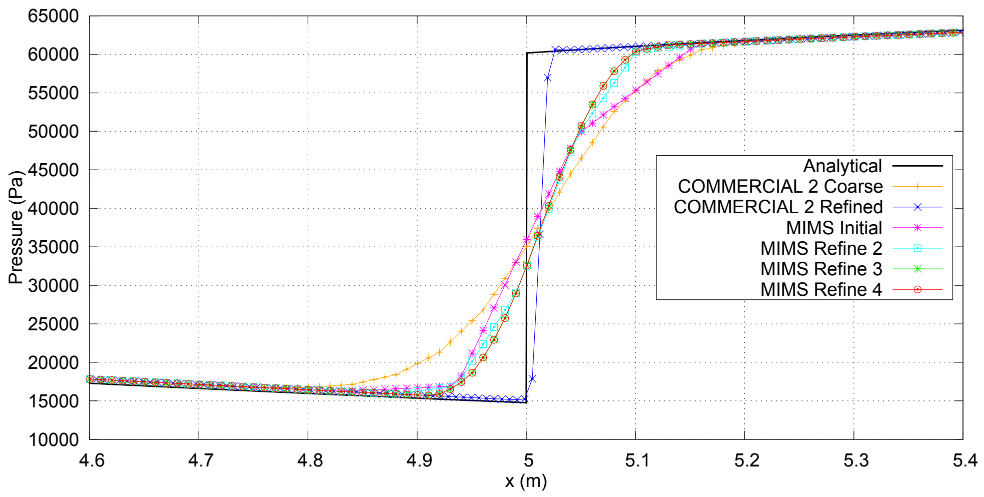


Figure 10.18: Midline Pressure (Line Markers Do Not Indicate Computational Points)

Table 10.4: Normal Shock Nozzle Calculation Time Comparison

(a) Meshless

Task	Time
Problem Setup*	00:02:00
Initial Preprocessing	00:01:12
Total Solve Time	00:10:24
Total Refine Time	00:07:41
Total	**00:22:17**
***Total Engineer Time**	**00:02:00**

(b) COMMERCIAL 2

Task	Time
Mesh Setup (0.02m)*	00:05:00
Problem Setup (0.02cm)*	00:02:00
Solve Time (0.02cm)	00:24:36
Mesh Setup (Clustered)*	00:12:00
Problem Setup (Clustered)*	00:02:00
Solve Time (Clustered)	02:12:27
Total	**02:58:03**
***Total Engineer Time**	**00:21:00**

Even though the meshless solutions failed to achieve the same levels of accuracy as the structured mesh, it is still important to compare final computation times between the two solution processes. Thus, Table 10.4 lists the total solution times for both the meshless solution and the solution generated using COMMERCIAL 2. As has been typical amongst the previous two case studies, the meshless process takes considerably less time to arrive at its converged result.

10.4 Remarks

Having demonstrated the accuracy of the MIMS methodology, this chapter sought to demonstrate the advantages over more traditional mesh-based techniques in the areas of initial model generation time and overall solution time. Although these metrics are oftentimes difficult to objectively quantify, by solving three representative case studies from start to finish with various techniques it should become clear that a robust meshless method employing adaptive refinement is able to compete with commercially available mesh-based engineering analysis software on the basis of solution time. The elimination of mesh generation time is the most notable improvement over existing techniques as this is time spent by the engineer, whose time is considerably more expensive than computer time. Although this may be appealing on its own, what is more interesting is that despite the additional preprocessing and overhead computations required for the meshless process, the MIMS method is able to advance the solution to steady state in less time than the mesh-based techniques. This can be attributed to the fact that an adaptive refinement process can most appropriately refine the model, resulting in discretizations requiring fewer computational nodes, and a more efficient coarse to fine solution transition. It is for this reason that it is important to consider solution times against real-world problem geometries and field configurations. In simple verification test settings, it is unreasonable to expect any meshless technique to compete with a solution generated on a structured mesh since the geometry and problem do not require insight on behalf of the engineer. On the contrary, when field characteristics are unknown *a priori*, the engineer must make assumptions about the flow field, and as such, is prone to either produce a mesh with too little (resulting in loss of accuracy) or too much (resulting in increased computation time) initial detail.

Another conclusion that can be drawn from these case studies is that it will be difficult for meshless methods, in general, to compete in terms of accuracy, against solutions generated over structured meshes (especially when sharp field discontinuities are present). This statement is justified by realizing that the unstructured meshless operators, in order to obtain solution robustness, will generally require a larger support domain than the equivalently formed structured mesh. As evidenced by the results for the normal shock nozzle, despite every consideration that was made during support domain construction and directional refinement, the normal shock is still smeared over the largest topology that spans the discontinuity; whereas the structured domain is able to smear the solution, at least, over one cell width. That being said, this case does not represent a failure on behalf of meshless methods, instead, it illustrates their practical utility and serves to demonstrate where analysts should defer to alternative solution techniques. For example, if a MIMS solver was utilized to generate the initial solution shown in Figure 10.14 instead of COMMERCIAL 2, 10 minutes would have been saved, resulting in 30% less time spent during the initial solution process. At that point the engineer could determine if a more detailed study should be performed, and whether the necessary time should be invested to generate a structured mesh better able to resolve the flow phenomenon. In many scenarios, exactly capturing discontinuity width will not be a primary concern; indeed, for the normal shock nozzle, simply knowing that there is a shock present and accurately determining its location will usually provide sufficient information to drive further design improvements. In this manner, MIMS-based meshless methods present themselves as an ideal general solver; they are capable of automatically resolving continuous flow fields to a grid-converged state in minimal time, while at the same time providing the user the necessary information needed to assess whether or not further detailed studies are warranted.

CHAPTER 11
CONCLUSIONS

In summary, the primary focus of this research has been on developing an industrially relevant, numerical physics solution process implementing a novel meshless method. Collectively referred to as the Model Integrated Meshless Solution method, or MIMS, this methodology incorporates both a unique meshless implementation utilizing a variety of interpolation techniques as well as a novel model generation process capable of automatically generating point distributions for arbitrarily complex geometries. It is the development and fusion of these techniques which represent the primary contribution of this research to the scientific community. To provide a graphical representation of the complete research efforts, Figure 11.1 illustrates each component of the MIMS process in flowchart form, with each component linked to the associated modules. Figure 11.1 should serve as justification for the overarching theme of this research: that to be competitive on an industrially relevant level, meshless methods must be tightly integrated with the model generation process. This is evidenced by the multitude of interconnects between the model generation modules and the solution routines, each of which represent a major advantage of the proposed method over traditional mesh-based techniques. Without these close connections between the solution and model generation processes, meshless methods become just another PDE solution technique and will have a difficult time competing with more established methodologies. However, when integrated with the model generation process, meshless methods can reach their full potential

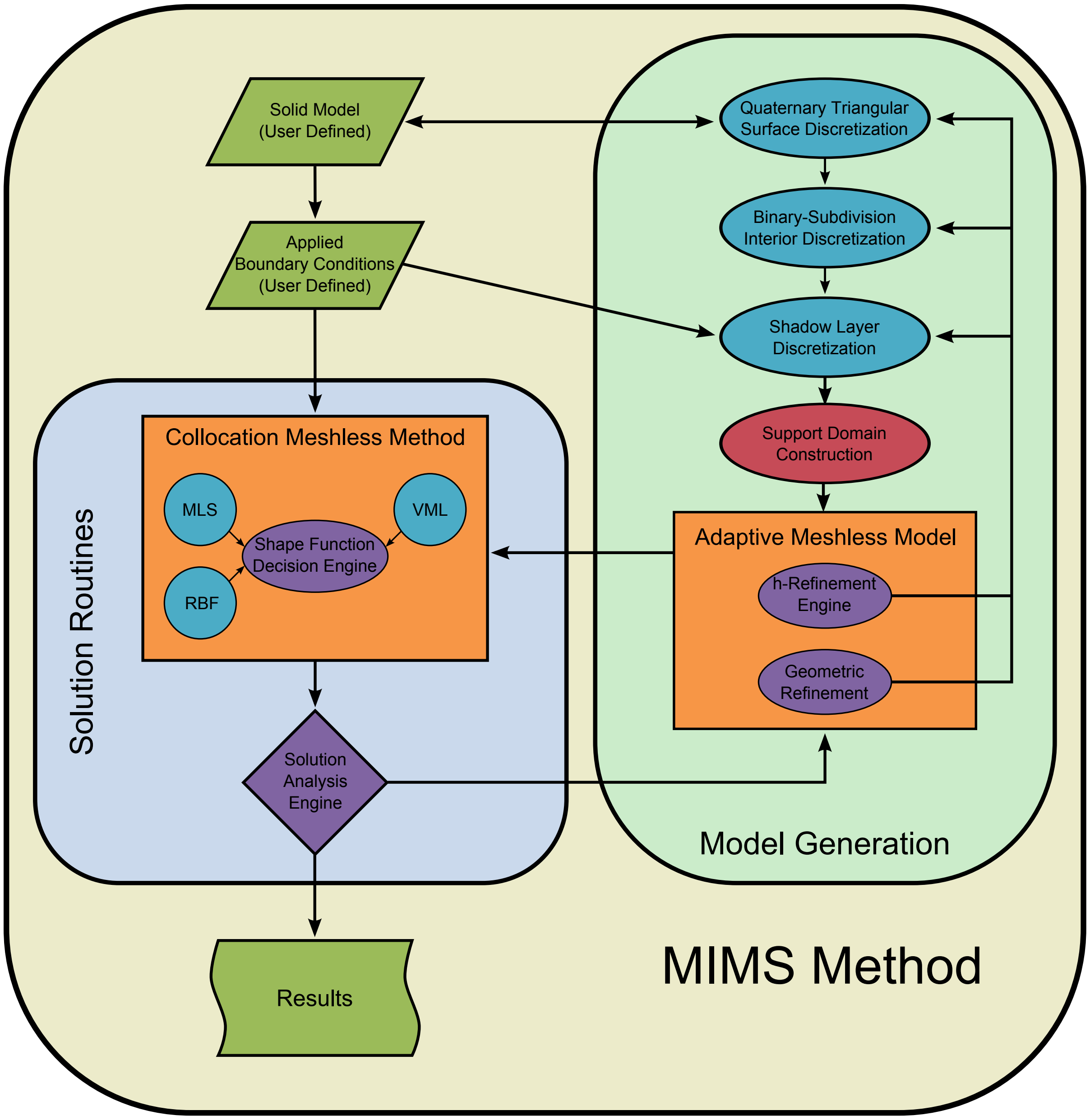

Solid Model (User Defined)
Applied Boundary Conditions (User Defined)
Quaternary Triangular Surface Discretization
Binary-Subdivision Interior Discretization
Shadow Layer Discretization
Support Domain Construction
Adaptive Meshless Model
h-Refinement Engine
Geometric Refinement
Model Generation
Solution Routines
Collocation Meshless Method
MLS
VML
RBF
Shape Function Decision Engine
Solution Analysis Engine
Results
MIMS Method

Figure 11.1: MIMS Process Flowchart

and obtain levels of automation and speed unparalleled by current engineering analysis packages.

The primary contribution of this research to the meshless community has been the development of model generation structures and the associated algorithmic developments that allow for a compact point distribution that can take advantage of the liberties granted by meshless methods. In this respect the MIMS method is able to obtain accurate solutions for arbitrarily complex geometries. In addition to this, there have been several incremental advancements that have facilitated a more complete method; these include:

- Acknowledgment and development of complete near-boundary region representation (shadow layer)
- Formulation and analysis of Virtual Finite Differencing shape function generation technique
- Development of a support domain optimization procedure capable of handling clustered point distributions with high aspect ratios
- Examination and validation of heuristic RBF shape parameter optimization routine

These advancements, coupled with the model integration and adaptive refinement procedures have facilitated development of the MIMS method. Through a set of validation problems and three thorough case studies (whereby the entire solution process was compared to currently available engineering analysis packages), this research effort has demonstrated both the validity and applicability of the MIMS methodology. These example problem have demonstrated that a properly integrated meshless approach can significantly reduce both engineering and computational times while at the same time providing a high degree of accuracy for most engineering problems. In addition, it has emphasized the ideal use of meshless techniques; for parametric and early study designs where many different geometric configurations will be analyzed and overall trends compared. In this manner, and based on the comparative times demonstrated throughout Chapter 10, the MIMS method represents an enormous potential time savings and will result in a more streamlined and thorough design process.

Although substantial improvements over existing technologies have been demonstrated, there are several areas that were beyond the scope of this work and could benefit from further research. Specifically, these include:

- Further exploration of practical numerical stability behavior and understanding the implications of elliptical support domains on interpolation behavior
- Investigate the possibility of optimizing RBF shape parameter value on local field characteristics to provide better capturing of high gradient regions
- Examination of using weight functions to control numerical smearing of the solution across high-aspect ratio regions (via directionally independent weight control)
- Further improvements to support domain construction in terms of optimizing performance and reducing excess nodal influence
- Addressing refinement as applied to unsteady solutions; currently the meshless techniques employed in the MIMS method may be generally applied to unsteady problems, however, the adaptive refinement and model generation routines would need to be modified to facilitate efficient removal of nodes during the solution process

Each of these areas of research could provide additional benefits to the MIMS method, further expanding its capabilities in the areas of heat transfer and compressible fluid flow. In addition, future efforts may be desired to apply these techniques to other areas of engineering analysis including incompressible fluid flow, elasticity, and electromagnetism.

This research has introduced a new technique for the solution of heat transfer and compressible fluid flow problems called the Model Integrated Meshless Solution (MIMS) method, providing a fundamental start into development of integrated model generation techniques tailored for meshless methods. This research has illustrated that meshless methods must be tightly integrated with the model generation process to be competitive, and only through this integration will they be capable of fully realizing their potential and be considered a competitive technique with more established solution methodologies. The results shown have demonstrated that once sufficient levels of integration have been achieved, meshless methods can significantly reduce both engineering and overall computational expense while at the same time providing a high degree of accuracy for real-world engineering problems. These qualities make the MIMS method an ideal platform to provide an intuitive, yet powerful approach to solving modern engineering analysis problems.

LIST OF REFERENCES

[1] Orszag, S.A., "Numerical Methods for the Simulation of Turbulence", *Physics of Fluids*, **12**, 250–257 (1969).

[2] Kreiss, H.O. and Oliger, J., "Comparison of accurate methods for the integration of hyperbolic equations", *Tellus*, **24**, 199–215 (1972).

[3] Gottlieb, D. and Orzag, S.A., *Numerical Analysis of Spectral Methods: Theory and Applications*, Society for Industrial and Applied Mathematics, Bristol, England (1977).

[4] Maday, Y. and Quateroni, A., "Spectral and Pseudo-Spectral Approximations of the Navier-Stokes Equations", *SIAM Journal of Numerical Analysis*, **19**(4), 761–780 (1982).

[5] Canuto, C., Hussaini, M.Y., Quarteroni, A. and Zang, T.A., *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, NY (1988).

[6] Canuto, C., Hussaini, M.Y., Quarteroni, A. and Zang, T.A., *Spectral Methods: Fundamentals in Single Domains*, Springer-Verlag, Berlin, Germany (2006).

[7] Macaraeg, M. and Street, C.L., "Improvement in Spectral Collocation Discretization Through a Multiple Domain Technique", *Applied Numerical Mathematics*, **2**, 95–108 (1986).

[8] Canuto, C., Hussaini, M.Y., Quarteroni, A. and Zang, T.A., *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics*, Springer-Verlag, New York, NY (2007).

[9] Patera, A., "A Spectral Element Method of Fluid Dynamics: Laminar Flow in a Channel Expansion", *Journal of Computational Physics*, **54**, 468–488 (1984).

[10] Orszag, S.A., "Spectral Methods for Problems in Complex Geometries", *Journal of Computational Physics*, **37**, 70–92 (1980).

[11] Hwar, C.K., Hirsch, R., Taylor, T. and Rosenberg, A.P., "A Pseudo-Spectral Matrix Element Method for Solution of Three Dimensional Incompressible Flows and its Parallel Implementation", *Journal of Computational Physics*, **83**, 260–291 (1989).

[12] Proot, M.J. and Gerritsma, M.I., "A least-squares spectral element formulation for the Stokes problem", *Journal of Scientific Computing*, **17**, 285–296 (2002).

[13] Bayliss, A., Gottlieb, D., Matkowsky, B.J. and Minkoff, M., "An adaptive pseudo-spectral method for reaction diffusion problems", *Journal of Computational Physics*, **81**(2), 421–443 (1981).

[14] Ku, H.C., Hirsh, R.S. and Taylor, T.D., "A pseudospectral method for solution of the three-dimensional incompressible Navier-Stokes equations", in "Tenth International Conference on Numerical Methods in Fluid Dynamics", vol. 264 of *Lecture Notes in Physics*, 391–397, Springer Berlin / Heidelberg (1986).

[15] Heinrichs, W., "Least Squares Spectral Collocation for the Navier Stokes Equations", *Journal of Scientific Computing*, **21**(1), 81–90 (2004).

[16] Krastev, K. and Schäfer, M., "A multigrid pseudo-spectral method for incompressible Navier-Stokes flows", *Comptes Rendus Mécanique*, **333**(1), 59–64 (2005).

[17] Fornberg, B., "The pseudospectral method: Comparisons with finite differences for the elastic wave equation", *Geophysics*, **52**(4), 483–501 (1987).

[18] Renaut, R. and Frohlich, J., "A Pseudospectral Chebychev Method for the 2D Wave Equation with Domain Stretching and Absorbing Boundary Conditions", *Journal of Computational Physics*, **124**, 324–336 (1996).

[19] Faccioli, E., Maggio, F., Paolucci, R. and Quarteroni, A., "2d and 3D elastic wave propagation by a pseudo-spectral domain decomposition method", *Journal of Seismology*, **1**(3), 237–251 (1997).

[20] Lucy, L.B., "A numerical approach to the testing of the fission hypothesis", *Astronomical Journal*, **82**, 1013–1024 (1977).

[21] Monaghan, J.J., "Why Particle Methods Work", *SIAM Journal on Scientific and Statistical Computing*, **3**(4), 422–433 (1982).

[22] Gingold, R.A. and Monaghan, J.J., "Smoothed particle hydrodynamics - Theory and application to non-spherical stars", *Monthly Notices of the Royal Astronomical Society*, **181**, 375–389 (1977).

[23] Monaghan, J.J., "An introduction to SPH", *Computer Physics Communications*, **48**(1), 89–96 (1988).

[24] Fang, J., Owens, R.G., Tacher, L. and Parriaux, A., "A numerical study of the SPH method for simulating transient viscoelastic free surface flows", *Journal of Non-Newtonian Fluid Mechanics*, **139**, 68–84 (2006).

[25] Fang, J., Parriaux, A., Rentschler, M. and Ancey, C., "Improved SPH methods for simulating free surface flows of viscous fluids", *Applied Numerical Mathematics*, **59**(2), 251–271 (2009).

[26] Libersky, L.D. and Petschek, A.G., "Smooth particle hydrodynamics with strength of materials", in H.E. Trease, M.J. Fritts and W.P. Crowley, eds., "Advances in the Free-Lagrange Method Including Contributions on Adaptive Gridding and the Smooth Particle Hydrodynamics Method", vol. 395/1991 of *Lecture Notes in Physics*, 248–257, Springer Berlin / Heidelberg, New York, NY (1991).

[27] Libersky, L.D., Petschek, A.G., Carney, T.C., Hipp, J.R. and Allahdadi, F.A., "High strain Lagrangian hydrodynamics: a three-dimensional SPH code for dynamic material response", *Journal of Computational Physics*, **109**(1), 67–75 (1993).

[28] Benz, W. and Asphaug, E., "Simulations of brittle solids using smooth particle hydrodynamics", *Computer Physics Communications*, **87**(1-2), 253–265 (1995).

[29] Belytschko, T., Lu, Y.Y. and Gu, L., "Element-Free Galerkin methods", *International Journal of Numerical Methods*, **37**, 229–256 (1994).

[30] Johnson, G.R., Petersen, E.H. and Stryk, R.A., "Incorporation of an SPH option into the EPIC code for a wide range of high velocity impact computations", *International Journal of Impact Engineering*, **14**(1-4), 385–394 (1993).

[31] Monaghan, J.J. and Kocharyan, A., "SPH simulation of multi-phase flow", *Computer Physics Communications*, **87**(1-2), 225–235 (1995).

[32] Morris, J.P., "Simulating surface tension with smoothed particle hydrodynamics", *International Journal for Numerical Methods in Fluids*, **33**(3), 333–353 (2000).

[33] Oger, L. and Savage, S.B., "Smoothed particle hydrodynamics for cohesive grains", *Computer Methods in Applied Mechanics and Engineering*, **180**(1), 169–183 (1999).

[34] Ata, R. and Soulaïmani, A., "A stabilized SPH method for inviscid shallow water flows", *International Journal for Numerical Methods in Fluids*, **47**(2), 139–159 (2004).

[35] Maveyraud, C., Benz, W., Sornette, A. and Sornette, D., "Solid friction at high sliding velocities: an explicit three-dimensional dynamical smoothed particle hydrodynamics approach", *Journal of Geophysical Research*, **104**, 28769–28788 (1999).

[36] Monaghan, J.J., "Smoothed particle hydrodynamics", *Annual review of astronomy and astrophysics*, **30**, 543–574 (1992).

[37] Liu, G.R. and Liu, M.B., *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*, World Scientific Publishing Company (2003).

[38] Monaghan, J.J., “Smoothed particle hydrodynamics”, *Reports on Progress in Physics*, **68**, 1703–1759 (2005).

[39] Nayroles, B., Touzot, G. and Villon, P., “Generalizing the finite element method: Diffuse approximation and diffuse elements”, *Computational Mechanics*, **10**(5), 307–318 (1992).

[40] Shepard, D., “A two-dimensional interpolation function for irregularly-spaced data”, in “Proceedings of the 1968 23rd ACM National Conference”, 517–524, ACM (1968).

[41] Lancaster, P. and Salkauskas, K., “Surfaces Generated by Moving Least Squares Methods”, *Mathematics of Computation*, **37**(155), 141–158 (1981).

[42] Marechal, Y., Coulomb, J.L., Meunier, G. and Touzot, G., “Use of the diffuse element method for electromagnetic field computation”, *IEEE Transactions on Magnetics*, **29**(2), 1475–1478 (1993).

[43] Modaressi, H. and Aubert, P., “A Diffuse Element-Finite Element Technique for Transient Coupled Analysis”, *International Journal for Numerical Methods in Engineering*, **39**(22), 3809–3838 (1996).

[44] Lu, Y.Y., Belytschko, T. and Gu, L., “A new implementation of the element free Galerkin method”, *Computer methods in applied mechanics and engineering*, **113**(3), 397–414 (1994).

[45] Belytschko, T., Gu, L. and Lu, Y.Y., “Fracture and crack growth by element free Galerkin methods”, *Modelling and Simulation in Materials Science and Engineering*, **2**(3), 519–534 (1994).

[46] Duarte, C.A. and Oden, J.T., “Hp clouds - a meshless method to solve boundary-value problems”, Tech. Rep. 95-05, Texas Institute for Computational and Applied Mathematics, University of Texas at Austin (1995).

[47] Duarte, C.A., “A review of some meshless methods to solve partial differential equations”, Tech. Rep. 95-06, Texas Institute for Computational and Applied Mathematics, University of Texas at Austin (1995).

[48] Melenk, J.M. and Babuska, I., “The Partition of Unity Finite Element Method: Basic Theory and Application”, *Computer Methods and Applications in Mechanics and Engineering*, **139**, 289–316 (1996).

[49] Sukumar, N., Dolbow, J., Devan, A., Yvonnet, J., Chinesta, F., Ryckelynck, D., Lorong, P., Alfaro, I., Martinez, M.A., Cueto, E. and Doblare, M., “Meshless Methods and Partition of Unity Finite Elements”, *International Journal of Forming Processes*, **8**(4), 409–427 (2005).

[50] Dolbow, J., *An extended finite element method with discontinuous enrichment for applied mechanics*, Ph.D. thesis, Northwestern University (1999).

[51] Moës, N., Dolbow, J. and Belytschko, T., "A finite element method for crack growth without remeshing", *International Journal for Numerical Methods in Engineering*, **46**, 131–150 (1999).

[52] Liszka, T. and Orkisz, J., "The finite difference method at arbitrary irregular grids and its application in applied mechanics", *Computers and Structures*, **11**, 83–95 (1980).

[53] Liszka, T. and Orkisz, J., "Finite difference method for arbitrary irregular meshes in nonlinear problems of applied mechanics", *International Journal For Numerical Methods In Engineering*, **20**, 1599–1612 (1984).

[54] Chung, K.C., "A generalized finite-difference method for heat transfer problems of irregular geometries", *Numerical Heat Transfer*, **4**, 345–357 (1981).

[55] Tworzydlo, W., "Analysis of large deformations of membrane shells by the generalized finite difference method", *Computers and Structures*, **27**(1), 39–59 (1987).

[56] Tworzydlo, W., "The fdm in arbitrary curvilinear co-ordinates - formulation, numerical approach and applications", *International Journal For Numerical Methods In Engineering*, **28**, 261–277 (1989).

[57] Benito, J.J., Urena, F. and Gavete, L., "Influence of several factors in the generalized finite difference method", *Applied Mathematical Modelling*, **25**(12), 1039–1053 (2001).

[58] Chew, C.S., Yeo, K.S. and Shu, C., "A generalized finite-difference (GFD) ALE scheme for incompressible flows around moving solid bodies on hybrid meshfree-Cartesian grids", *Journal of Computational Physics*, **218**(2), 510–548 (2006).

[59] Tang, D., Yangb, C., Kobayashi, S. and Ku, D.N., "Generalized finite difference method for 3-D viscous flow in stenotic tubes with large wall deformation and collapse", *Applied Numerical Mathematics*, **38**(1-2), 49–68 (2001).

[60] Tseng, A.A., "A generalized finite difference scheme for convection-dominated metal-forming problems", *International Journal for Numerical Methods in Engineering*, **20**(10), 1885–1900 (1983).

[61] Schulz, D., Glingener, C. and Voges, E., "Novel generalized finite-difference beam propagation method", *IEEE Journal of Quantum Electronics*, **30**(2), 1132–1140 (1994).

[62] Radic, S. and George, N., "Ultrafast pulse propagation in periodic optical media: a generalized finite-difference time-domain approach", *Optics Letters*, **19**(14), 1064–1066 (1994).

[63] Schulz, D., Glingener, C., Bludszuweit, M. and Voges, E., “Mixed Finite Element Beam Propagation Method”, *Journal of Lightwave Technology*, **16**(7), 1336–1342 (1998).

[64] Kansa, E.J., “Multiquadrics: A Scattered Data Approximation Scheme with Applications to Computational Fluid Dynamics II-Solutions to Parabolic, Hyperbolic and Elliptic Partial Differential Equations”, *Computers and Mathematics with Applications*, **19**, 147–161 (1990).

[65] Kansa, E.J., “Multiquadrics: A Scattered Data Approximation Scheme with Applications to Computational Fluid Dynamics I-Surface Approximations and Partial Derivative Estimates”, *Computers and Mathematics with Applications*, **19**, 127–145 (1990).

[66] Buhmann, M.D., *Radial Basis Functions: Theory and Implementation*, Cambridge University Press, Cambridge, MA (2003).

[67] Hardy, R.L., “Multiquadric Equations of Topography and Other Irregular Surfaces”, *Journal of Geophysical Research*, **76**, 1905–1915 (1971).

[68] Dyn, N., Levin, D. and Rippa, S., “Numerical Procedures for Surface Fitting of Scattered Data by Radial Basis Functions”, *SIAM Journal of Scientific and Statistical Computing*, **7**(2), 639–659 (1986).

[69] Carr, J.C., Fright, W.R. and Beatson, R.K., “Surface interpolation with radial basis functions for medical imaging”, *IEEE Transactions on Medical Imaging*, **16**(1), 96–107 (1997).

[70] Powell, M.J.D., “The Theory of Radial Basis Function Approximation”, in W. Light, ed., “Advances in Numerical Analysis”, vol. 2, 143–167, Oxford Science Publications (1992).

[71] Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C. and Evans, T.R., “Reconstruction and representation of 3D objects with radial basis functions”, in “SIGGRAPH ’01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques”, ACM, New York, NY (2001).

[72] Cheng, A.H.D., Golberg, M.A., Kansa, E.J. and Zammito, G., “Exponential convergence and H-c multiquadric collocation method for partial differential equations”, *Numerical Methods for Partial Differential Equations*, **19**(5), 571–594 (2003).

[73] Gerace, S., Divo, E. and Kassab, A., “A radial-basis function approach to thermoelasticity Meshless method”, in B. Gamez, D. Ojeda, G. Larrazábal and M. Cerrolaza, eds., “Simulation and Modeling in Engineering and Science: Proceedings of the VIII International Congress on Numerical Methods and Applied Science”, Sociedad Venezolana Numerical Methods in Engineering, Margarita Island, Venezuela (2006).

[74] Kansa, E.J. and Hon, Y.C., “Circumventing the Ill-Conditioning Problem with Multiquadric Radial Basis Functions: Applications to Elliptic Partial Differential Equations”, *Computers and Mathematics with Applications*, **39**, 123–137 (2000).

[75] Schaback, R., “Multivariate interpolation and approximation by translates of a basis function”, *Series In Approximations and Decompositions*, **6**(1), 491–514 (1995).

[76] Stevens, D., Power, H., Lees, M. and Morvan, H., “The use of PDE centres in the local RBF Hermitian method for 3D convective-diffusion problems”, *Journal of Computational Physics*, **228**(12), 4606–4624 (2009).

[77] Brown, D., “On Approximate Cardinal Preconditioning Methods for Solving PDEs with Radial Basis Functions”, *Engineering Analysis with Boundary Elements*, **29**(4), 343–353 (2005).

[78] Ling, L., Opfer, R. and Schaback, R., “Results on meshless collocation techniques”, *Engineering Analysis with Boundary Elements*, **30**(4), 247–253 (2006).

[79] Beatson, R.K., Light, W.A. and Billings, S., “Fast Solution of the Radial Basis Function Interpolation Equations: Domain Decomposition Methods”, *SIAM Journal on Scientific Computing*, **22**(5), 1717–1740 (2000).

[80] Ling, L. and Kansa, E.J., “Preconditioning for Radial Basis Functions with Domain Decomposition Methods”, *Mathematical and Computer Modelling*, **38**(5), 320–327 (2004).

[81] Mai-Duy, N. and Tran-Cong, T., “Mesh-free radial basis function network methods with domain decomposition for approximation of functions and numerical solution of Poisson’s equations”, *Engineering Analysis with Boundary Elements*, **26**, 133–156 (2002).

[82] Munoz-Gomez, J., Gonzalez-Casanova, P. and Rodriguez-Gomez, G., “Domain decomposition by radial basis functions for time dependent partial differential equations”, in “Proceedings of the 2nd IASTED International Conference on Advances in Computer Science and Technology”, Puerto Vallarta, Mexico (2006).

[83] Rosales, A.H. and Power, H., “Non-overlapping domain decomposition algorithm for the Hermite radial basis function meshless collocation approach: Applications to convection diffusion problems”, *Journal of Algorithms and Computational Technology*, **1**(1), 127–159 (2007).

[84] Divo, E., Mitteff, E. and Quintana, L., “A Parallel Domain Decomposition Technique for Meshless Methods Applications to Large-Scale Heat Transfer Problems”, in “ASME Summer Heat Transfer Meeting”, (2004).

[85] Gerace, S., *A Meshless Method Approach for Solving Coupled Thermoelasticity Problems*, Honors in the Major Undergraduate thesis, University of Central Florida (2006).

[86] Lee, C.K., Liu, X. and Fan, S.C., "Local multiquadric approximation for solving boundary value problems", *Computational Mechanics*, **30**(5-6), 396–409 (2003).

[87] Tolstykh, A.I. and Shirobokov, D.A., "On using radial basis functions in a finite difference mode with applications to elasticity problems", *Computational Mechanics*, **33**(1), 68–79 (2003).

[88] Shu, C., Ding, H. and Yeo, K.S., "Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations", *Computer Methods in Applied Mechanics and Engineering*, **192**(7-8), 941–954 (2003).

[89] Sarler, B. and Vertnik, R., "Meshfree explicit local radial basis function collocation method for diffusion problems", *Computers & Mathematics with Applications*, **51**(8), 1269–1282 (2006).

[90] Vertnik, R., Zaloznik, M. and Sarler, B., "Solution of transient direct-chill aluminium billet casting problem with simultaneous material and interphase moving boundaries by a meshless method", *Engineering Analysis with Boundary Elements*, **30**(10), 847–855 (2006).

[91] Vertnik, R. and Sarler, B., "Meshless local radial basis function collocation method for convective-diffusive solid-liquid phase change problems", *International Journal for Numerical Methods in Heat and Fluid Flow*, **16**(5), 617–640 (2006).

[92] Vertnik, R. and Sarler, B., "Simulation of continuous casting of steel by a meshless technique", *International Journal of Cast Metals Research*, **22**(1-4), 311–313 (2009).

[93] Divo, E. and Kassab, A., "A Meshless Method for Conjugate Heat Transfer", *Engineering Analysis*, **29**, 136–149 (2005).

[94] Divo, E. and Kassab, A., "Boundary Elements and RBF Meshless Methods Modeling in Thermo-Fluids", Tech. rep., NASA Thermal & Fluids Workshop (2005).

[95] Divo, E. and Kassab, A., "Modeling of Convective and Conjugate Heat Transfer by a Third Order Localized RBF Meshless Collocation Method", in "Proceedings of NHT-05, Eurotherm82", (2005).

[96] Divo, E. and Kassab, A., "An Efficient Localized Radial Basis Function Meshless Method for Fluid Flow and Conjugate Heat Transfer", *AMSE Journal of Heat Transfer*, **129**, 179–183 (2007).

[97] Gerace, S., *An Interactive Framework for Meshless Methods Analysis In Computational Mechanics And Thermofluids*, Master's thesis, University of Central Florida (2007).

[98] Stevens, D., Power, H. and Morvan, H., "An order-N complexity meshless algorithm for transport-type PDEs, based on local Hermitian interpolation", *Engineering Analysis with Boundary Elements*, **33**(4), 425–441 (2009).

[99] Sarler, B., *Advances in Meshfree Techniques*, vol. 5 of *Computational Methods in Applied Sciences*, chap. From Global to Local Radial Basis Function Collocation Method for Transport Phenomena, 257–282, Springer (2007).

[100] Vertnik, R. and Sarler, B., "Solution of Incompressible Turbulent Flow by a Mesh-Free Method", *Computer Modeling in Engineering and Sciences*, **44**(1), 65–95 (2009).

[101] Erhart, K., Gerace, S., Divo, E. and Kassab, A., "Turbulent compressible flow analysis with meshless methods", in "Proceedings of the Int. Conf. on Comp. Methods for Coupled Problems in Science and Engineering (COUPLED PROBLEMS 2009)", Ischia Island, Italy (2009).

[102] Erhart, K., Gerace, S., Divo, E. and Kassab, A., "An rbf interpolated generalized finite difference meshless method for compressible turbulent flows", in "Proceedings of the ASME 2009 International Mechanical Engineering Congres & Exposition", ASME, Lake Buena Vista, FL, USA (2009).

[103] Liou, M.S. and Steffen, C.J., "A New Flux Splitting Scheme", *Journal of Computational Physics*, **107**, 23–39 (1993).

[104] George, P.L., *Automatic Mesh Generation: Application to Finite Element Methods*, John Wiley & Sons, New York, NY (1991).

[105] Ladeveze, P., Pelle, J.P. and Rougeot, P.H., "Error Estimation and Mesh Optimization for Classical Finite Elements", *Engineering Computations*, **8**(1), 69–80 (1991).

[106] Liu, G.R., *Mesh Free Methods: Moving beyond the Finite Element Method*, CRC Press, Boca Raton, FL (2003).

[107] Cleveland, W.S., *Visualizing Data*, AT&T Bell Laboratories, Murray Hill, NJ (1993).

[108] Levin, D., "The Approximation Power Of Moving Least-Squares", *Mathematics Of Computation*, **67**(224), 1517–1531 (1998).

[109] Levin, D., *Geometric modeling for scientific visualization*, Springer-Verlag (2004).

[110] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D. and Silva, C.T., "Computing and Rendering Point Set Surfaces", *IEEE Transactions on Visualization and Computer Graphics*, **9**(1), 3–15 (2003).

[111] Amenta, N. and Kil, Y.J., "Defining point-set surfaces", in "Proceedings of the International Conference on Computer Graphics and Interactive Techniques", 264–270, SIGGRAPH, ACM, Los Angeles, CA (2004).

[112] Hawkins, D.M., "The Problem of Overfitting", *J. Chem. Inf. Comput. Sci.*, **44**(1), 1–12 (2004).

[113] Liu, G.R., Liu, M.B. and Lam, K.Y., "A general approach for constructing smoothing functions for meshfree methods", in "Proceedings of the Ninth International Conference on Computing in civil and Building Engineering", 431–436, Taipei, Taiwan (2002).

[114] Breitkopf, P., Rassineux, A., Touzot, G. and Villon, P., "Explicit form and efficient computation of MLS shape functions and their derivatives", *International Journal For Numerical Methods In Engineering*, **48**, 451–466 (2000).

[115] De, S. and Bathe, K.J., "Towards an efficient meshless computational technique: the method of finite spheres", *Engineering Computations*, **18**(1), 170–192 (2001).

[116] Franke, R., "Scattered data interpolation: Tests of some methods", *Mathematics of Computation*, **38**, 181–200 (1982).

[117] Wang, J.G. and Liu, G.R., "Radial point interpoloation method for elastoplastic problems", in "Proceedings of the 1st International Conference on Structural Stability and Dynamics", 703–708, Taipei, Taiwan (2000).

[118] Wang, J.G. and Liu, G.R., "Radial point interpolation method for no-yielding surface models", in "Proceedings of the First M.I.T. Conference on Computational Fluid and Solid Mechanics", 538–540, Cambridge, MA (2001).

[119] Wang, J.G. and Liu, G.R., "A point interpolation meshless method based on radial basis functions", *International Journal for Numerical Methods in Engineering*, **54**(11), 1623–1648 (2002).

[120] Fasshauer, G., "RBF Collocation Methods as Pseudo-Spectral Methods", in A. Kassab, C.A. Brebbia and E. Divo, eds., "Boundary Elements XVII", 47–57, WIT Press (2005).

[121] Divo, E. and Kassab, A., "An Efficient Localized RBF Meshless Method Applied to Fluid Flow and Conjugate Heat Transfer", in "ASME IMECE", (2005).

[122] Golberg, M.A., Chen, C.S. and Bowman, H., "Some recent results and proposals for the use of radial basis functions in the BEM", *Engineering Analysis with Boundary Elements*, **23**(4), 285–296 (1999).

[123] Kassab, A. and Divo, E., "An Efficient Localized Radial Basis Function Meshless Method for Fluid Flow and Conjugate Heat Transfer", *ASME Journal of Heat Transfer*, **129**, 179–183 (2007).

[124] Tannehill, J.C., Anderson, D.A. and Pletcher, R.H., *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation (1997).

[125] Gerace, S., Erhart, K., Divo, E. and Kassab, A., "Generalized finite difference meshless method in computational mechanics and thermofluids", in "Proceedings of the Int. Conf. on Comp. Methods for Coupled Problems in Science and Engineering (COUPLED PROBLEMS 2009)", (2009).

[126] Platte, R.B. and Driscoll, T.A., "Eigenvalue stability of radial basis function discretizations for time-dependent problems", *Computers & Mathematics with Applications*, **51**(8), 1251–1268 (2006).

[127] Fuselier, E.J., Narcowich, F.J., Ward, J.D. and Wright, G.B., "Error and stability estimates for surface-divergence free RBF interpolants on the sphere", *Mathematics of Computation*, **78**, 2157–2186 (2009).

[128] Fasshauer, G., *Meshfree Approximation Methods with MATLAB*, vol. 6 of *Interdisciplinary Mathematical Sciences*, World Scientific Publishers (2007).

[129] Foley, T., "Near optimal parameter selection for multiquadric interpolation", *Journal of Applied Science and Computation*, **1**, 54–69 (1994).

[130] Rippa, S., "An algorithm for selecting a good value for the parameter c in radial basis function interpolation", *Advances in Computational Mathematics*, **11**(2-3), 193–210 (1999).

[131] Fasshauer, G.E. and Zhang, J.G., "On choosing "optimal" shape parameters for RBF approximation", *Numerical Algorithms*, **45**(1-4), 345–368 (2007).

[132] Fornberg, B. and Wright, G., "Stable computation of multiquadric interpolants for all values of the shape parameter", *Computers & Mathematics with Applications*, **48**(5-6), 853–867 (2004).

[133] Bern, M. and Eppstein, D., *Computing in Euclidean Geometry*, chap. Mesh Generation and Optimal Triangulation, 23–90, Lecture Notes Series on Computing, World Scientific (1992).

[134] Chew, L.P., "Guaranteed-quality mesh generation for curved surfaces", in "SCG '93: Proceedings of the ninth annual symposium on Computational geometry", ACM, San Diego, California, USA (1993).

[135] Ruppert, J., "A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation", *Journal of Algorithms*, **18**(3), 548–585 (1995).

[136] De Floriani, L. and Puppo, E., "Hierarchical triangulation for multiresolution surface description", *ACM Trans. Graph.*, **14**(4), 363–411 (1995).

[137] Zhao, X., Chen, J. and Li, Z., "A QTM-based Algorithm for Generation of the Voronoi Diagram on a Sphere", in D. Richardson and P. van Oosterom, eds., "Advances in Spatial Data Handling: Proceedings from the 10th International Symposium on Spatial Data Handling", 269–285, Springer Berlin (2002).

[138] Dutton, G., "Encoding and Handling Geospatial Data with Hierarchical Triangular Meshes", in M.J. Kraak and M. Molenaar, eds., "Advances In GIS Research II: Proceedings of the Sixth International Symposium on Spatial Data Handling", 505–518, Taylor & Francis Publishing (1996).

[139] Dutton, G.H., *A Hierarchical Coordinate System for Geoprocessing and Cartography*, vol. 79 of *Lecture Notes in Earth Sciences*, chap. Computational aspects of a quaternary triangular mesh, 41–70, Springer Berlin Heidelberg (1999).

[140] Frey, P. and Borouchaki, H., "Geometric surface mesh optimization", *Computing and Visualization in Science*, **1**(3), 113–121 (1998).

[141] Wang, D., Hassan, O., Morgan, K. and Weatherill, N., "EQSM: An efficient high quality surface grid generation method based on remeshing", *Computer Methods in Applied Mechanics and Engineering*, **195**(41-43), 5621–5633 (2006).

[142] Tournois, J., Alliez, P. and Devillers, O., "Interleaving Delaunay Refinement and Optimization for 2D Triangle Mesh Generation", in M.L. Brewer and D. Marcum, eds., "Proceedings of the 16th International Meshing Roundtable", 83–101, Springer Berlin Heidelberg, Seattle, Washington, USA (2007).

[143] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, 3 edn. (2007).

[144] Berger, M.J. and LeVeque, R.J., "An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries", *AIAA-89-1930* (1989).

[145] Clarke, D.K., Salas, M.D. and Hassan, H.A., "Euler Calculations for Multielement Airfoils Using Cartesian Grids", *AIAA Journal*, **24**, 353–358 (1986).

[146] Coirier, W.J. and Powell, K.G., "An Accuracy Assessment of Cartesian Mesh Approaches for the Euler Equations", *Journal of Computational Physics*, **117**, 121–131 (1995).

[147] Ye, T., Mittal, R., Udaykumar, H. and Shyy, W., "An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries", *Journal of Computational Physics*, **156**, 209–240 (1999).

[148] Luo, H., Bauma, J.D. and Löhner, R., "A Hybrid Cartesian Grid and Gridless Method for Compressible Flows", *Journal of Computational Physics*, **214**(2), 618–632 (2006).

[149] Wang, Z.J., CPhen, R.F., Hariharan, N., Przekwas, A.J. and Grove, D., "A 2n Tree Based Automated Viscous Cartesian Grid Methodology For Feature Capturing", *AIAA-99-3300* (1999).

[150] Marshall, D.D., *Extending the Functionalities of Cartesian Grid Solvers: Viscous Effects Modeling and MPI Parallelization*, Ph.D. thesis, Georgia Institute of Technology (2002).

[151] Marshall, D.D. and Ruffin, S.M., "An Embedded Boundary Cartesian Grid Scheme for Viscous Flows using a New Viscous Wall Boundary Condition Treatment", in "42nd AIAA Aerospace Sciences Meeting and Exhibit", AIAA 2004-0581, Reno, NV (2004).

[152] Casalini, F. and Dadone, A., "Computations of viscous flows using a multigrid finite volume lambda formulation", *Engineering Computations*, **16**(7), 767 – 786 (1999).

[153] Hoffmann, K.A. and Chiang, S.T., *Computational Fluid Dynamics*, vol. 2, Engineering Education System, 4 edn. (2004).