2007

An Interactive Framework For Meshless Methods Analysis In Computational Mechanics And Thermofluids

Salvadore Anthony Gerace
University of Central Florida

AN INTERACTIVE FRAMEWORK FOR MESHLESS METHODS ANALYSIS IN COMPUTATIONAL MECHANICS AND THERMOFLUIDS

by

SALVADORE ANTHONY GERACE
B.S. University of Central Florida, 2006

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
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

Fall Term
2007

ABSTRACT

In recent history, the area of physics-based engineering simulation has seen rapid increases in both computer workstation performance as well as common model complexity, both driven largely in part by advances in memory density and availability of clusters and multi-core processors. While the increase in computation time due to model complexity has been largely offset by the increased performance of modern workstations, the increase in model setup time due to model complexity has continued to rise. As such, the major time requirement for solving an engineering model has transitioned from computation time to problem setup time. This is due to the fact that developing the required mesh for complex geometry can be an extremely complicated and time consuming task. Consequently, new solution techniques which are capable of reducing the required amount of human interaction are desirable.

The subject of this thesis is the development of a novel meshless method that promises to eliminate the need for structured meshes, and thus, the need for complicated meshing procedures. Although the savings gain due to eliminating the meshing process would be more than sufficient to warrant further study, the proposed method is also capable of reducing the computation time and memory footprint compared to similar models solved using more traditional finite element, finite difference, finite volume, or boundary element methods.

In particular, this thesis will outline the development of an interactive, meshless, physically accurate modeling environment that provides an extensible framework which can be applied to a multitude of governing equations encountered in computational mechanics and thermofluids. Additionally, through the development of tailored preprocessing routines, efficiency and accuracy of the proposed meshless algorithms can be tested in a more realistic and flexible environment. Examples are provided in the areas of elasticity, heat transfer and computational fluid dynamics.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Most real world engineering problems involve complicated multi-variable partial differential equations that are either too difficult or impractical to solve analytically. It is for this reason that numerical methods were developed, allowing solutions for these types of problems to be found with acceptable accuracy and within a reasonable time frame. The earliest developments in numerical methods were in the areas of Finite Difference, Finite Element, and Finite Volume methods, which approximate a domain as a discrete set of points, elements, or volumes with a defined connectivity. Although these methods are capable of solving most engineering problems, they require structured point distributions or meshes which take both time and skill to develop. In parallel, the method of Boundary Elements was developed which solves for the field solution by integrating only over the boundary, rather than the entire domain. The Boundary Element method eliminated the need for meshing the interior, but is limited in the scope of governing equations that it can solve. It is for this reason that new developments have begun focusing on meshless, or mesh reduction, methods in an attempt to create a generalized technique which minimizes the amount of human interaction needed to build a particular model. Additionally, due to the increase in both memory density and availability of clusters and multi-core processors, common engineering models have become more and more complex in an attempt to more accurately represent the problem physics and geometry. While the increase in computation time due to model complexity has largely been offset by the increased performance of modern workstations, the increase in model setup time due to model complexity has continued to rise. As such, within recent history the major time requirement for solving an engineering model has largely transitioned from computation time to problem setup time, further increasing the desirability of a meshless solution technique that does not require human interaction.

Having established the motivation for exploring meshless methods as a solution technique, it is important to acknowledge the previous work that has been done in this field. Understanding that meshless methods are a relative newcomer to the field of computational methods, there has still been substantial development of these techniques as they have the hope of reducing the effort devoted to model preparation [1-11]. The approach finds its origin in classical spectral or pseudo-spectral methods [12-20] that are based on global orthogonal functions such as Legendre or Chebyshev polynomials requiring a regular nodal point distribution. In contrast, meshless methods use a nodal or point distribution that is not required to be uniform or regular in their spatial distribution [21-23]. Early developments by our research group at the University of Central Florida focused primarily on applications to heat transfer [24], and involved using global interpolation schemes. After some time, the area of heat transfer was revisited, however, new emphasis was placed on localized interpolation schemes, which proved both more efficient and more accurate than their global counterparts [25-28]. Once the localized methods had been proven capable in the area of heat transfer, new applications in thermoelasticity [29-30], inverse problems [31], as well as fluid flow solutions [32] were developed and implemented mainly in two-dimensional space. As this process matured, it became clear that the next step in the evolution was to create a complete three-dimensional implementation of these routines. However, rather than simply create individual applications for each solution domain, it was decided that the best course of action was to develop a meshless framework which would be capable of being applied to a variety of governing equations.

This thesis will begin by detailing the meshless collocation techniques that will be implemented in the developed framework that may be used for solving various governing equations. Specifically, the next chapter will present two meshless collocation techniques, Localized Radial Basis collocation and Virtual RBF Finite Difference

collocation. Following the meshless collocation development, the meshless framework formulation technique will be presented, which is a method of forming any linear set of equations into a standardized form which can be solved in a general fashion (pivotal to any framework). Next, several governing equations (including steady-state heat conduction, steady-state ideal flow with heat advection-diffusion, and steady-state elasticity) will be presented and formulated into the form needed for the solution process. Prior to the verification and examples, several pre-processing concepts will be explained and detailed in a general fashion to more concretely describe some of the techniques used throughout the framework. Following this, verification problems and several examples will be demonstrated for the aforementioned governing equations. Finally, the last chapter will include conclusions and future work that could be completed within this field.

CHAPTER 2
MESHLESS COLLOCATION TECHNIQUES

Conventional numerical methods commonly used in engineering applications, including finite element, finite volume, and to some degree, boundary element methods, all introduce the idea of a defined connectivity between nodes or volumes. This connectivity is what allows the various techniques to determine the influence of any node to its neighbors. While it is true that for simple models, development of this connectivity can be largely automated, as the complexity and size of the problem increases it becomes exceedingly difficult to automate this procedure. With meshless methods, the underlying goal is to eliminate the need for a defined connectivity mesh. Instead, the influence of one node on its neighbors is defined by an interpolation technique that can be used regardless of model geometry or nodal spacing.

For all collocation methods, the underlying nodal influence can be expressed in the form of a set of weights multiplied by a set of nodal values. By correctly determining a particular set of weights, these methods are capable of approximating both the field, as well as derivative values at a particular node. For example, in the case of a simple second order central difference over the one-dimensional set of nodes shown in Figure 2.1,

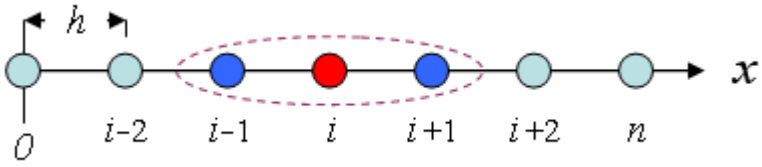

Figure 2.1. Sample Finite Difference Mesh

the approximated value using finite difference for the second derivative in the x-direction is given by Equation 2.1,

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i = \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} + O\left(h^2\right) \tag{2.1}$$

where ϕ represents the field values and h represents the horizontal distance between nodes. Therefore, for this particular case, the second derivative could be expressed as a vector-vector multiplication whereby the weight vector would consist of the coefficients to the nodal values, and the nodal values would consist of the corresponding values of ϕ. This concept can be expressed mathematically in the following expression

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i = \left\{ \begin{array}{ccc} \frac{1}{h^2} & \frac{-2}{h^2} & \frac{1}{h^2} \end{array} \right\} \left\{ \begin{array}{c} \phi_{i+1} \\ \phi_i \\ \phi_{i-1} \end{array} \right\} = \{W\}^{\mathrm{T}} \{\phi\} \tag{2.2}$$

As already stated, the notion of expressing a derivative operator as a multiplication of a set of weights, $\{W\}$, by a set of nodal values, $\{\phi\}$, is the underlying mechanism for any collocation method. The differences between methods arise in the various ways that the weight vectors are determined. In the finite difference technique, the weights are determined by truncating the Taylor series representation of the desired derivative, while in finite element and finite volume, the weights are built from interpolation functions used over a structured element (or volume). In contrast, for meshless methods, the weight vectors are built using various interpolating functions which do not require exact correlation with existing nodes. Due to this fact, there is no required connectivity for building the weights for each derivative operator, and thus, no connectivity mesh is required. In the end, however, the meshless methods still result in a vector set of weights multiplied by a vector set of nodal values to approximate the derivatives at any location in the field.

The remaining sections of this chapter will detail the two collocation techniques used in the meshless algorithms presented in this paper. The first technique, Localized Radial Basis Function (RBF) collocation, uses the concept of a radial basis function to interpolate the field about a data center and is used for radially symmetric derivatives (as

well as a general field interpolator). The second technique, Virtual RBF Finite Difference collocation, combines RBF field interpolation with standard finite difference approximations to develop expressions for non-radially symmetric and up-winded derivatives.

2.1. LOCALIZED RADIAL BASIS FUNCTION COLLOCATION

Localized Radial Basis Function collocation begins with the principle that any arbitrary domain Ω can be interpolated over by collocating about a number of points using some basis function, χ. This method (and in fact, both meshless collocation techniques implemented in this paper) breaks the overall region into smaller sub-domains, called topologies, which allow for a more efficient and accurate solution method when compared to global interpolation techniques. As a simple example, the process of breaking down a two-dimensional region into representative topologies is demonstrated in Figure 2.2.

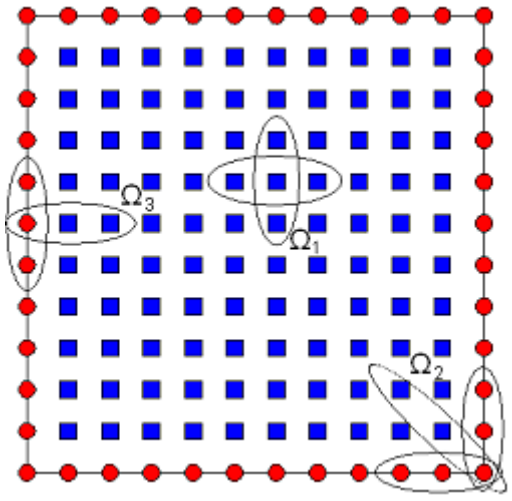

Figure 2.2. Representative Topologies for Two-Dimensional Region

The region shown is a two-dimensional representation, however, in general, the nodes can describe an n-dimensional field (where n is generally 2 or 3). The represented

field, $\phi(\boldsymbol{x})$, can be shown to be globally interpolated by multiplying the basis functions by a set of expansion coefficients as follows

$$\phi(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \chi_j(\boldsymbol{x}) + \sum_{k=1}^{NP} \alpha_{(k+N)} P_k(\boldsymbol{x}) \tag{2.3}$$

where N is the total number of points in the domain, α are the expansion coefficients for ϕ, and $\chi(\boldsymbol{x})$ are a-priori defined expansion function. Additionally, a similar expansion is performed over NP number of polynomial functions, $P_k(\boldsymbol{x})$, which must be added to the overall expansion to guarantee that constant and linear fields can be retrieved exactly. For this collocation method, the value of the function, ϕ, is known at particular locations, $\boldsymbol{x}_j$, and the basis function, $\chi(\boldsymbol{x})$, can be evaluated at these same locations. With both of these values known, the only unknown in this equation is the expansion coefficient vector α. However, this formulation assumes a global collocation, which, as already stated, is not ideal. Thus, the concept of local topologies can be used to reformulate Equation 2.3 such that instead of summing over the entire domain, the basis function can be applied only to the local topologies, Ω_i. Therefore, the interpolated field can now be expressed as

$$\phi(\boldsymbol{x}) = \sum_{j=1}^{NF} \alpha_j \chi_j(\boldsymbol{x}) + \sum_{k=1}^{NP} \alpha_{(k+NF)} P_k(\boldsymbol{x}) \tag{2.4}$$

where instead of summing over the entire region, Equation 2.4 instead is summed over the number of points in a given topology, expressed as NF. For example, looking at the two-dimensional representative topology shown in Figure 2.3, NF is equal to 5, as there are 5 nodes included in the topology for the data center (Node 1).

Figure 2.3. Representative Local Topology

A critical component to this type of meshless collocation technique is determining a suitable basis function, χ, that will accurately interpolate between data points. Much research has gone into analyzing the behavior of the most common basis functions for this type of technique [12, 21-24], and the most accurate and stable was determined to be the family of so-called Inverse Hardy Multiquadrics [33] (an Inverse Multiquadric function where $n = 1$) which follows the form

$$\chi_j(\boldsymbol{x}) = \frac{1}{\sqrt{r_j(\boldsymbol{x})^2 + c^2}} \tag{2.5}$$

where $r_j(\boldsymbol{x})$ is the Euclidean distance given by

$$r_j(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_j\| \tag{2.6}$$

and c is a free constant known as the shape parameter. This parameter governs the overall oscillatory nature of the interpolator function and from Equation 2.5 it can be seen that increasing the value of c will cause the response of the interpolation function to become flatter (fewer oscillations between nodes). This behavior is advantageous because it provides a more accurate local representation of the field, however, doing so will eventually cause the function to become so flat, and thus ill conditioned, that the system becomes singular, and will be unable to be solved. Much research has been performed on the subject of determining optimal values of c for particular basis functions [11-14, 24-25] and it was determined that the best interpolation occurs when the conditioning number of the system reaches the range of $10^{10} \sim 10^{12}$. For most simple geometry it was found that acceptable conditioning values were produced by using a value of c equal to 10 times the average minimum nodal spacing in the domain [14]. However, for more complex geometry, optimization must be performed on this parameter to obtain accurate interpolation behavior. For the purpose of the initial framework development, this optimization was performed using a simple brute force technique whereby the shape

parameter was continuously increased until the expansion system became singular and unable to be solved, at which point the shape parameter was reduced to the highest stable value determined. Though this is an inefficient method of determining the parameter with no prior knowledge, by using an initial starting point of 10 times the average minimum nodal spacing in the domain, very few attempts were needed to determine an acceptable value of c for each topology.

Having defined the basis function used in this collocation technique, the next step is to construct the weights associated with the derivative operators. To accomplish this, Equation 2.4 may be applied to all of the nodes in a given topology as follows

$$\begin{aligned}
\phi(\boldsymbol{x}_1) &= \sum_{j=1}^{NF} \alpha_j \chi_j(\boldsymbol{x}_1) + \sum_{k=1}^{NP} \alpha_{(k+NF)} P_k(\boldsymbol{x}_1) \\
\phi(\boldsymbol{x}_2) &= \sum_{j=1}^{NF} \alpha_j \chi_j(\boldsymbol{x}_2) + \sum_{k=1}^{NP} \alpha_{(k+NF)} P_k(\boldsymbol{x}_2) \\
&\vdots \\
\phi(\boldsymbol{x}_{NF}) &= \sum_{j=1}^{NF} \alpha_j \chi_j(\boldsymbol{x}_{NF}) + \sum_{k=1}^{NP} \alpha_{(k+NF)} P_k(\boldsymbol{x}_{NF})
\end{aligned} \tag{2.7}$$

Expanding the summations in Equation 2.7 produces a system of equations where the unknown values are the expansion coefficients, α

$$\begin{bmatrix}
\chi_1(\boldsymbol{x}_1) & \dots & \chi_{NF}(\boldsymbol{x}_1) & P_1(\boldsymbol{x}_1) & \dots & P_{NP}(\boldsymbol{x}_1) \\
\chi_1(\boldsymbol{x}_2) & \dots & \chi_{NF}(\boldsymbol{x}_2) & P_1(\boldsymbol{x}_2) & \dots & P_{NP}(\boldsymbol{x}_2) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\chi_1(\boldsymbol{x}_{NF}) & \dots & \chi_{NF}(\boldsymbol{x}_{NF}) & P_1(\boldsymbol{x}_{NF}) & \dots & P_{NP}(\boldsymbol{x}_{NF}) \\
P_1(\boldsymbol{x}_1) & \dots & P_1(\boldsymbol{x}_{NF}) & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
P_{NP}(\boldsymbol{x}_1) & \dots & P_{NP}(\boldsymbol{x}_{NF}) & 0 & \dots & 0
\end{bmatrix}
\begin{Bmatrix}
\alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{NF} \\ \alpha_{NF+1} \\ \vdots \\ \alpha_{NF+NP}
\end{Bmatrix}
=
\begin{Bmatrix}
\phi(\boldsymbol{x}_1) \\ \phi(\boldsymbol{x}_2) \\ \vdots \\ \phi(\boldsymbol{x}_{NF}) \\ 0 \\ \vdots \\ 0
\end{Bmatrix} \tag{2.8}$$

At this point, the expansion coefficients can be solved for, and the interpolation used as shown. However, rather than solve for the α values directly, this step can be

avoided by pre-building the collocation vectors. Thus, knowing that Equation 2.8 is of the standard form

$$[G]\{\alpha\} = \{\phi\} \tag{2.9}$$

the α vector can be solved for by multiplying both sides by the inverse of the basis matrix, $[G]$,

$$\{\alpha\} = [G]^{-1}\{\phi\} \tag{2.10}$$

Since the interest is to develop the weights representing a particular derivative at the data center, another application of Equation 2.4, this time *only* at the data center, produces

$$\phi(\boldsymbol{x_c}) = \sum_{j=1}^{NF} \alpha_j \chi_j(\boldsymbol{x_c}) + \sum_{k=1}^{NP} \alpha_{(k+N)} P_k(\boldsymbol{x_c}) \tag{2.11}$$

or, in matrix notation,

$$\phi(\boldsymbol{x_c}) = \{\chi(\boldsymbol{x_c})\}^{\mathrm{T}}\{\alpha\} \tag{2.12}$$

By introducing the expression for $\{\alpha\}$ shown in Equation 2.10 into Equation 2.12, the following interpolation function for the field ϕ at the data center can be expressed as

$$\phi(\boldsymbol{x_c}) = \{\chi(\boldsymbol{x_c})\}^{\mathrm{T}}[G]^{-1}\{\phi\} \tag{2.13}$$

At this point Equation 2.13 is capable of interpolating to any data center which lies within the given region. Interestingly enough, although in general, the data center belongs to its own topology, this does not need to be the case. Therefore, Equation 2.13 is truly a localized field interpolator and is not restricted only to the nodal positions. This concept of localized interpolation is pivotal to the development of the Virtual RBF Finite Difference technique that will be discussed in the following section.

Although a localized interpolator has now been developed, the goal is still to approximate the derivatives of the field, not the field itself. To do this, rather than apply

Equation 2.4 directly, the derivative is first taken of the field and the basis function. For example, the Laplacian operator would be constructed as

$$\nabla^2\phi(\boldsymbol{x}_c) = \sum_{j=1}^{NF}\alpha_j\nabla^2\chi_j(\boldsymbol{x_c}) + \sum_{k=1}^{NP}\alpha_{(k+N)}\nabla^2 P_k(\boldsymbol{x_c}) \tag{2.14}$$

or, in matrix notation,

$$\nabla^2\phi(\boldsymbol{x}_c) = \left\{\nabla^2\chi(\boldsymbol{x_c})\right\}^{\mathrm{T}}\{\alpha\} \tag{2.15}$$

Once again, substituting the expression for $\{\alpha\}$ shown in Equation 2.10, results in the following expression for the Laplacian derivative of the field at the data center, $\boldsymbol{x}_c$

$$\nabla^2\phi(\boldsymbol{x_c}) = \left\{\nabla^2\chi(\boldsymbol{x_c})\right\}^{\mathrm{T}}[G]^{-1}\{\phi\} \tag{2.16}$$

Realizing that the $\{\nabla^2\chi(\boldsymbol{x_c})\}$ vector and the $[G]$ matrix are entirely geometry dependant, this multiplication can be performed as a preprocessing step resulting in the following expression

$$\nabla^2\phi(\boldsymbol{x_c}) = \{\Psi_{\nabla^2}\}^{\mathrm{T}}\{\phi\} \tag{2.17}$$

which, is of the desired form of a weight vector, $\{\Psi_{\nabla^2}\}$, multiplied by a nodal value vector, $\{\phi\}$. For this particular case, $\{\Psi_{\nabla^2}\}$ represents the weights associated with the Laplacian derivative for a given topology, however, similar techniques could be used to build any derivative operator. In practice, because radial basis functions perform best when operating on radially symmetric operators, only the Laplacian derivative is developed and used in this fashion.

2.2. VIRTUAL RBF FINITE DIFFERENCE COLLOCATION

As previously described, conventional finite difference collocation techniques involve truncating the Taylor series expansion to approximate a given derivative at a specific location within a field. The finite difference formulations can therefore be

directly applied to any regular point distribution when the surrounding nodes are properly located within the bounds of the approximation. However, this technique has a limitation in that it requires a regular, defined distribution of nodes, something that is not possible for an unstructured, meshless domain. By utilizing some of the concepts of Localized Radial Basis collocation, the standard finite difference formulation can be extended to non-regular node distributions and be made into a meshless technique.

The first step to formulating the Virtual RBF Finite Difference technique is to understand the concepts behind native finite differencing. As already stated, the underlying concept is to truncate the Taylor series representation of the derivatives at a given location to produce approximate values that can be evaluated with acceptable error. Rather than derive the finite difference equations, something that can be found in many standard texts [34], equations for several of the most common three-dimensional derivative operators are listed below in Equations 2.18a-2.18i.

$$\left.\frac{\partial \phi}{\partial x}\right|_{x,y,z} = \frac{\phi(x+\Delta x, y, z) - \phi(x-\Delta x, y, z)}{2\Delta x} \tag{2.18a}$$

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x,y,z} = \frac{\phi(x+\Delta x, y, z) - 2\phi(x, y, z) + \phi(x-\Delta x, y, z)}{\Delta x^2} \tag{2.18b}$$

$$\left.\frac{\partial \phi}{\partial y}\right|_{x,y,z} = \frac{\phi(x, y+\Delta y, z) - \phi(x, y-\Delta y, z)}{2\Delta y} \tag{2.18c}$$

$$\left.\frac{\partial^2 \phi}{\partial y^2}\right|_{x,y,z} = \frac{\phi(x, y+\Delta y, z) - 2\phi(x, y, z) + \phi(x, y-\Delta y, z)}{\Delta y^2} \tag{2.18d}$$

$$\left.\frac{\partial \phi}{\partial z}\right|_{x,y,z} = \frac{\phi(x, y, z+\Delta z) - \phi(x, y, z-\Delta z)}{2\Delta z} \tag{2.18e}$$

$$\left.\frac{\partial^2 \phi}{\partial z^2}\right|_{x,y,z} = \frac{\phi(x, y, z+\Delta z) - 2\phi(x, y, z) + \phi(x, y, z-\Delta z)}{\Delta z^2} \tag{2.18f}$$

$$\left.\frac{\partial^2\phi}{\partial x\partial y}\right|_{x,y,z} = \frac{1}{2\Delta x}\left(\frac{\phi(x+\Delta x,y+\Delta y,z)-\phi(x+\Delta x,y-\Delta y,z)}{2\Delta y} - \frac{\phi(x-\Delta x,y+\Delta y,z)-\phi(x-\Delta x,y-\Delta y,z)}{2\Delta y}\right) \tag{2.18g}$$

$$\left.\frac{\partial^2\phi}{\partial x\partial z}\right|_{x,y,z} = \frac{1}{2\Delta x}\left(\frac{\phi(x+\Delta x,y,z+\Delta z)-\phi(x+\Delta x,y,z-\Delta z)}{2\Delta z} - \frac{\phi(x-\Delta x,y,z+\Delta z)-\phi(x-\Delta x,y,z-\Delta z)}{2\Delta z}\right) \tag{2.18h}$$

$$\left.\frac{\partial^2\phi}{\partial y\partial z}\right|_{x,y,z} = \frac{1}{2\Delta y}\left(\frac{\phi(x,y+\Delta y,z+\Delta z)-\phi(x,y+\Delta y,z-\Delta x)}{2\Delta z} - \frac{\phi(x,y-\Delta y,z+\Delta z)-\phi(x,y-\Delta y,z-\Delta z)}{2\Delta z}\right) \tag{2.18i}$$

Note that all of the above listed approximations are second order accurate with respect to the spacing in the direction of the derivative, i.e. $O(h^2)$.

Having identified the most common finite difference approximations, the next step is to develop the techniques needed to transform these structured equations into an unstructured form. To illustrate the overall concept, the easiest departure point is the case of a region with a known field, where the derivative at a specific location is desired. As a representative problem, Figure 2.4 shows a specific slice (constant z) of a three-dimensional region, with a known, continuous field given by $\phi(x, y, z)$.

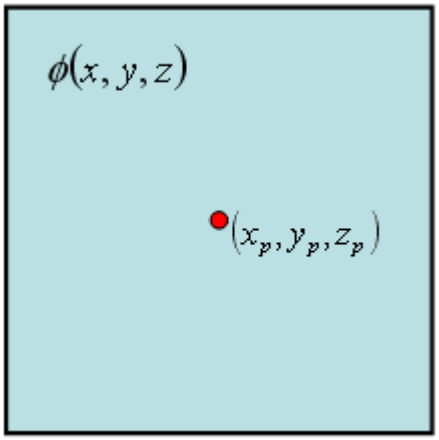

Figure 2.4. Example Field

For this example, the second derivative with respect to the x-direction ($\frac{\partial^2 \phi}{\partial x^2}$) is desired at the indicated node, located at (x_p, y_p, z_p). However, since there are initially no nodes at the locations necessary for the finite difference equation listed in Equation 2.18b, it may seem like the finite difference approximations can not be used. However, in this case the underlying field is known for all locations; thus, it is a trivial task to project "virtual" nodes at the necessary spacing, then evaluate the field using the provided function, as shown in Figure 2.5

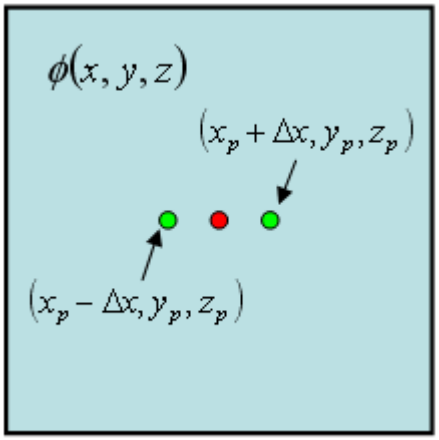

Figure 2.5. Example Field with Virtual Nodes

Here the virtual nodes are shown in green, with the data center node shown in red. Thus, having projected these nodes at the necessary locations, and determining their values, the finite difference approximation for the desired derivative can easily be applied.

Although this concept is very simple in its approach, and is quite obvious when the underlying field is known, it suffers from a major limitation. Without knowing the field everywhere within the domain (as a continuous function), the values at the virtual points can not be directly evaluated. As the overall goal in all of these techniques is to solve for this underlying field, this technique is impractical as a solution approach as described. However, by realizing that instead of using a known field equation to find the values of the virtual points, they can instead be interpolated from the surrounding data

points that exist within the field, and this concept can be applied to general solutions. To illustrate this concept, Figure 2.6 shows the same domain, data center, and virtual points that were used before, except now the field function is not known, and additional nodes have been added (at which the field is known).

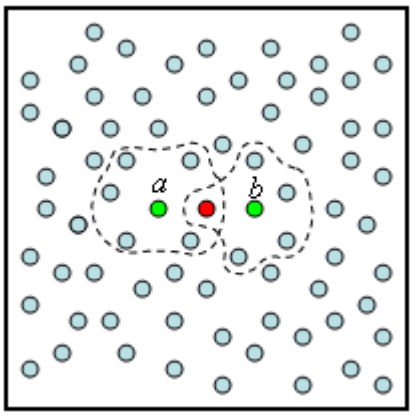

Figure 2.6. Example Distribution with Virtual Point Topologies

Therefore, in order to determine the field value at the virtual points, a topology can be built around each node (shown in Figure 2.6 as dashed lines). However, unlike the topologies constructed previously, these virtual topologies will not include the value at the data center (in this case, the virtual node). Thus, once the topologies have been chosen for each virtual point, Equation 2.13 can be applied in order to interpolate to the desired locations. It is interesting to note that all basis function components are functions of geometry only, and as such, as long as the node locations do not change, they will remain constant throughout the solution process. Therefore, a similar set of weights can be constructed for interpolating to the virtual points, and can be expressed as

$$\phi(\boldsymbol{x_v}) = \{\Psi\}^{\mathrm{T}}\{\phi\} \tag{2.19}$$

At this point, all the necessary tools have been developed for utilizing finite difference approximations over an unstructured mesh. First, the desired data center is identified, and virtual points are created at the required locations. Second, topologies are

created at these virtual points and the interpolation weight vectors, $\{\Psi\}$, are produced. Third, using the values of the field at the surrounding nodes included in the virtual topologies, the values at the virtual points can be evaluated using Equation 2.19. Lastly, having computed the value of the field at the virtual nodes for the data center, the desired finite difference approximation can be applied, and the value of the derivative obtained.

Although this process will certainly determine the desired derivative values, it is not in the form of an overall vector-vector multiplication that can be applied directly to the data center. To demonstrate how this procedure can be placed into the desired form, once again take for example the region shown in Figure 2.6, where the desired derivative is the second derivative with respect to x. Through application of Equation 2.19, it is easy to see that the interpolated values at the virtual points a and b will be given by

$$\phi_a = \{\Psi\}_a^{\mathrm{T}}\{\phi\}_a \qquad \phi_b = \{\Psi\}_b^{\mathrm{T}}\{\phi\}_b \tag{2.20}$$

where $\{\Psi\}_a$ is the interpolating weight vector associated with virtual point a, $\{\phi\}_a$ is the nodal value vector associated with the topology for virtual point a, $\{\Psi\}_b$ is the interpolating weight vector associated with virtual point b, $\{\phi\}_b$ is the nodal value vector associated with the topology for virtual point b, and ϕ_a and ϕ_b are the unknown field values at virtual nodes a and b, respectively. Therefore, substituting the values at the virtual points into Equation 2.18b results in the following expression

$$\left.\frac{\partial^2\phi}{\partial x^2}\right|_c = \frac{\phi_a - 2\phi_c + \phi_b}{\Delta x^2} \tag{2.21}$$

It has already been shown in Equation 2.2 that this expression can be expressed in terms of a weight vector times a set of nodal values. However, instead of the surrounding nodal values being directly known, they are now a set of vector-vector multiplications. 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), the following expression can be constructed

$$\left.\frac{\partial^2\phi}{\partial x^2}\right|_i = \left\{ \frac{1}{\Delta x^2}\{\Psi\}_a^{\mathrm{T}} \quad \frac{-2}{\Delta x^2} \quad \frac{1}{\Delta x^2}\{\Psi\}_b^{\mathrm{T}} \right\}^{\mathrm{T}} \begin{Bmatrix} \{\phi\}_a \\ \phi_c \\ \{\phi\}_b \end{Bmatrix} = \{\Psi_{xx}\}^{\mathrm{T}}\{\phi\} \qquad (2.22)$$

Note that through this process, the virtual points essentially vanish, and the final result is still simply a vector set of weights multiplied by a vector set of nodal values. Although this process was demonstrated for only one particular derivative, the same techniques can be used to develop the relationships for any derivative that can be expressed with a finite difference formulation.

Additionally, it is easy to see that for the special case of a region whose nodes are distributed in a Cartesian manner, this entire process can essentially be collapsed down into a conventional finite difference approximation. As such, in practical applications of this technique, it is important to first determine whether the region localized to the data center is structured in a Cartesian fashion. If the necessary difference points already exist, then Equations 2.18a-2.18i can be directly applied; if not, then virtual points can be projected and interpolated to, in order to generate the necessary evaluation locations.

CHAPTER 3
FRAMEWORK FORMULATION TECHNIQUES

Having developed the meshless collocation techniques, the next step in developing a working implementation is to demonstrate the techniques for formulating governing equations into a generalized, extensible form. Although the meshless collocation techniques allow for solutions without a structured mesh, it unfortunately introduces additional computation overhead as well as some minor numerical instability. Therefore, in an attempt to minimize both of these side-effects, the governing equations can be formulated into a form which minimizes the number of iterations necessary to convergence, as well as improving stability qualities of the system. Another major advantage of this formulation technique is that it can be generalized for any linear governing equation, and can be extended to non-linear systems as well.

The general concept of the meshless framework formulation is that at each data center, all unknown values can be solved for simultaneously, rather than decoupling the equations through lagging or other similar techniques. While the field values at the data center are solved for implicitly, the remaining nodes in the topology are still lagged to the previous iteration, which creates small local systems of equations, rather than a single global sparse matrix which needs to be solved over the entire field.

To demonstrate this formulation technique, two governing equations will be developed, the first being a scalar equation (Laplace equation) while the second being a set of coupled field equations which will demonstrate the advantages this technique from a generalized standpoint. The first governing equation that will be formulated in this fashion is a simple Laplace equation of the form

$$\nabla^2 \phi = 0 \tag{3.1}$$

It has already been shown in Chapter 2 that any derivative operator can be made into a vector-vector multiplication by implementing the meshless collocation techniques. Therefore, Equation 3.1 can be expressed as

$$\nabla^2\phi = \{\Psi_{\nabla^2}\}^{\mathrm{T}}\{\phi\} = 0 \tag{3.2}$$

where $\{\Psi_{\nabla^2}\}$ is the vector set of weights generated using the meshless collocation techniques and $\{\phi\}$ is the vector set of nodal values associated with the individual topologies. At this point, a new convention will be introduced to designate differences in these vectors between the steps in the formulation. For a given set of weights and nodal values, to designate that the data center is contained (or potentially contained) within the vectors, a hat symbol will be used over the respective variables. Thus, Equation 3.2 can be expressed using this convention as

$$\left\{\widehat{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\left\{\widehat{\phi}\right\} = 0 \tag{3.3}$$

At this point, the value at the data center is clearly going to be contained within the nodal vector, $\widehat{\phi}$ due to the radial nature of the meshless collocation techniques implemented. Since this is the value which is desired, by realizing that any term can be extracted out of this vector-vector multiplication, the nodal value at the data center (as well as the corresponding weight) can be removed, resulting in the following equation

$$\psi_{\nabla^2}\phi_c + \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\left\{\widetilde{\phi}\right\} = 0 \tag{3.4}$$

where ψ_{∇^2} is the scalar weight associated with the data center, ϕ_c is the value at the data center, $\widetilde{\Psi}_{\nabla^2}$ represents the weight values associated with the remaining topology nodes, and $\widetilde{\phi}$ represents the values associated with the remaining topology nodes. Note that instead of the hat symbol, in Equation 3.4 the tilde is now used to indicate that the data center has been removed from the respective weight and nodal value vectors.

The next step in this formulation is to realize that by lagging only the unknown field values at the remaining nodes in the topology, $\widetilde{\phi}$, the value at the data center can be solved for directly. Hence, performing this lagging and moving the now known values to the right hand side, results in the following expression

$$\psi_{\nabla^2}\phi_c^{\langle k\rangle} = -\left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\left\{\widetilde{\phi}\right\}^{\langle k-1\rangle} \tag{3.5}$$

where k is the current iteration value and $k-1$ is the previous iteration value. Thus, for an initial guess field $\phi^{\langle 0\rangle}$, Equation 3.5 can be used to iteratively update the field until some stopping criteria has been reached and the results reported as final. As already stated, the meshless framework formulation is optimal because it may be generalized to any linear governing equation set (regardless of the number of scalar field variables). Thus, to place Equation 3.5 into a standardized form, realize that this is of the classic matrix formulation,

$$[A]\{u\} = \{b\} \tag{3.6}$$

where, in this particular case,

$$[A] = [\,\psi_{\nabla^2}] \qquad \{u\} = \{\phi_c\}^{\langle k\rangle} \tag{3.7}$$
$$\{b\} = \left\{-\left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\left\{\widetilde{\phi}\right\}^{\langle k-1\rangle}\right\}$$

This is a optimal formulation for this type of equation because the coefficient matrix, $[A]$, is only dependant upon geometry and will not change during the solution process (since the weight values are dependant only on geometry).

Thus, the solution steps to solving this equation are as follows: First, generate the coefficient matrix, $[A]$, given the problem geometry. Then, using the initial guess values, update the right hand side vector, $\{b\}$, and solve the system of equations to update the unknown field values $\{u\}$. Once this has been done, the right hand side vector can be

updated for each node, then the field values can be found for the new iteration and the process repeated until some stopping criteria is met.

Having detailed the formulation process, there are a few important points that must be addressed regarding this technique. It can be shown that this type of formulation is essentially Jacobi iteration and while for simple governing equations, such as the Laplace equation, convergence may be unconditional, for more complex governing equations this may not be the case. As such, it is important to introduce a relaxation parameter into the iterative process to help stabilize the solution. Additionally, in its current form, the framework formulation technique is only applicable to linear sets of equations.

Having developed the framework formulation for the very simple case of the Laplace equation, the next goal is to demonstrate that the same can be done for a coupled set of linear equations. By adding a simple coupling term to the Laplace formulation, the following set of coupled equations may be developed

$$\begin{aligned} \nabla^2 u + \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) = 0 \\ \nabla^2 v + \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) = 0 \end{aligned} \tag{3.8}$$

As already stated, the first step in generating the meshless framework formulation for any set of equations is to replace the differential operators with the equivalent vector-vector representations, resulting in

$$\begin{aligned} \left\{ \widehat{\Psi}_{\nabla^2} \right\}^{\mathrm{T}} \{\widehat{u}\} + \left\{ \widehat{\Psi}_x \right\}^{\mathrm{T}} \{\widehat{u}\} + \left\{ \widehat{\Psi}_y \right\}^{\mathrm{T}} \{\widehat{v}\} = 0 \\ \left\{ \widehat{\Psi}_{\nabla^2} \right\}^{\mathrm{T}} \{\widehat{v}\} + \left\{ \widehat{\Psi}_x \right\}^{\mathrm{T}} \{\widehat{u}\} + \left\{ \widehat{\Psi}_y \right\}^{\mathrm{T}} \{\widehat{v}\} = 0 \end{aligned} \tag{3.9}$$

Once again, the hats indicate that these vectors contain the weights and value at the data center. Thus, since the goal is to generate an expression for the value at the data center, these terms can be factored from each of the vector-vector expressions to form

$$\psi_{\nabla^2} u_c + \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{u}\} + \psi_x u_c + \left\{\widetilde{\Psi}_x\right\}^{\mathrm{T}}\{\widetilde{u}\} + \psi_y v_c + \left\{\widetilde{\Psi}_y\right\}^{\mathrm{T}}\{\widetilde{v}\} = 0 \qquad (3.10)$$
$$\psi_{\nabla^2} v_c + \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{v}\} + \psi_x u_c + \left\{\widetilde{\Psi}_x\right\}^{\mathrm{T}}\{\widetilde{u}\} + \psi_y v_c + \left\{\widetilde{\Psi}_y\right\}^{\mathrm{T}}\{\widetilde{v}\} = 0$$

where the tildes indicate that the data center values have been removed from the respective derivative vectors. By moving all of the nodal value terms except the data centers to the right hand side, the following expression can be constructed

$$\psi_{\nabla^2} u_c + \psi_x u_c + \psi_y v_c = -\left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{u}\} - \left\{\widetilde{\Psi}_x\right\}^{\mathrm{T}}\{\widetilde{u}\} - \left\{\widetilde{\Psi}_y\right\}^{\mathrm{T}}\{\widetilde{v}\} \qquad (3.11)$$
$$\psi_x u_c + \psi_{\nabla^2} v_c + \psi_y v_c = -\left\{\widetilde{\Psi}_x\right\}^{\mathrm{T}}\{\widetilde{u}\} - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{v}\} - \left\{\widetilde{\Psi}_y\right\}^{\mathrm{T}}\{\widetilde{v}\}$$

which is clearly a linear set of equations which can be represented in the same form shown in Equation 3.6,

$$[A] = \begin{bmatrix} \psi_{\nabla^2} + \psi_x & \psi_y \\ \psi_x & \psi_{\nabla^2} + \psi_y \end{bmatrix} \qquad \{u\} = \begin{Bmatrix} u_c \\ v_c \end{Bmatrix}^{\langle k \rangle} \qquad (3.12)$$

$$\{b\} = \begin{Bmatrix} -\left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{u}\} - \left\{\widetilde{\Psi}_x\right\}^{\mathrm{T}}\{\widetilde{u}\} - \left\{\widetilde{\Psi}_y\right\}^{\mathrm{T}}\{\widetilde{v}\} \\ -\left\{\widetilde{\Psi}_x\right\}^{\mathrm{T}}\{\widetilde{u}\} - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{v}\} - \left\{\widetilde{\Psi}_y\right\}^{\mathrm{T}}\{\widetilde{v}\} \end{Bmatrix}^{\langle k-1 \rangle}$$

Thus, the steps for solving the unknown field variables at the data center are identical to that of the Laplace equation, except now the process is solving for two unknown values (u_c and v_c) simultaneously. It is this generality which makes this technique ideal for a framework that can be extended to any linear governing equation.

It has been shown that by utilizing the concept of a meshless framework formulation, the developed techniques are capable of providing a general method of solving any linear set of governing equations. Additionally, through this technique it can be shown that both stability qualities (due to the diagonal nature of the constructed $[A]$ matrix) as well as convergence speed are improved. Although some may argue that a globally implicit formulation (with local interpolation) may be faster when coupled with a sparse matrix solver, the meshless framework formulation benefits from the fact that

parallelization is extremely straightforward and scales linearly with the number of processors used. It is the generality and scaleability of these techniques which make them such an appealing method of solving these types of governing equations.

CHAPTER 4
APPLICATION TO STEADY-STATE HEAT CONDUCTION

The first type of problem to which the meshless techniques will be applied is the case of steady-state heat conduction. Steady-state heat conduction is one of the most basic governing equations and is an excellent first departure point for demonstrating the meshless framework.

4.1. FORMULATION OF GOVERNING EQUATION

In order to apply a governing equation to the meshless framework, it must first be formulated into the meshless framework form whereby the unknown values at a given node are coupled appropriately. For simple three-dimensional heat conduction, there is only one scalar field which needs to be solved for, which is the temperature, $T(x, y, z)$. Thus, the formulation begins from the differential equation which governs steady-state heat conduction with uniform generation, represented as

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) + \dot{q} = 0 \tag{4.1}$$

where k is the thermal conductivity of the material and $\dot{q}$ is the rate at which energy is generated per unit volume in the medium. If the assumption is made that the value of thermal conductivity is constant throughout the medium, then Equation 4.1 can be simplified into the following, recognizable form

$$\nabla^2 T = -\frac{\dot{q}}{k} \tag{4.2}$$

Equation 4.2 represents the governing equation implemented in the current version of the meshless framework detailed in this paper. The next step is to decompose the governing equation into the meshless framework form outlined in Chapter 3. To accomplish this, the differential operators must first be replaced by the equivalent

numerical representations, in the form of a vector of weights multiplied by a vector of nodal field values. In the case of steady-state heat conduction, there is only one differential operator present, the Laplacian, ∇^2. Thus, replacing this operator by the equivalent vector-vector representation gives

$$\left\{\widehat{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\left\{\widehat{T}\right\} = -\frac{\dot{q}}{k} \tag{4.3}$$

here $\widehat{\Psi}_{\nabla^2}$ represents the weight values associated with the Laplacian operator, and $\widehat{T}$ represents the nodal temperature values. At this stage, it is clear that the vector values $\widehat{\Psi}_{\nabla^2}$ and $\widehat{T}$ both contain the current topology data center (as indicated by the hat symbol). Thus, factoring out the weight and temperature value associated at the data center from these vectors results in

$$\psi_{\nabla^2} T_c + \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\} = -\frac{\dot{q}}{k} \tag{4.4}$$

where ψ_{∇^2} is the scalar weight associated with the temperature at the data center, T_c is the temperature at the data center, $\widetilde{\Psi}_{\nabla^2}$ represents the weight values associated with the remaining topology nodes, and $\widetilde{T}$ represents the temperature values associated with the remaining topology nodes. Note that instead of the hat symbol, the tilde is now used to indicate that the data center has been removed from this vector.

Equation 4.4 now represents the factored, meshless framework form of the governing equation shown in Equation 4.2 and can be solved globally as such. However, to improve performance and allow for parallelization, it is important to construct this equation in the final form outlined in Equation 3.6. Thus, by lagging the terms in the $\widetilde{T}$ vector and moving them to the right hand side of the equation, the following relationship can be formed,

$$\psi_{\nabla^2} T_c^{\langle k \rangle} = -\frac{\dot{q}}{k} - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\}^{\langle k-1 \rangle} \tag{4.5}$$

Here we have the weight and the temperature for the data center node at the next iteration, $\langle k \rangle$, and the remaining topology nodes at the previous iteration, $\langle k-1 \rangle$. Thus, Equation 4.5 can be expressed in the meshless form as

$$[A] = [\,\psi_{\nabla^2}] \qquad \{u\} = \{T_c\}^{\langle k \rangle} \tag{4.6}$$

$$\{b\} = \left\{ -\frac{\dot{q}}{k} - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}} \left\{\widetilde{T}\right\}^{\langle k-1 \rangle} \right\}$$

Thus, in this formulation, the coefficient matrix, $[A]$, is constant for all iterations, and only the right hand side vector, $\{b\}$, must be updated using temperature values from previous iterations.

4.2. FORMULATION OF BOUNDARY CONDITIONS

A similar formulation must also be made for all boundary condition types that can be applied to the model. In the case of steady-state heat conduction, there are three available boundary conditions:

1. Prescribed Temperature (Forced): $T = T_a$
2. Prescribed Flux (Natural): $k\frac{\partial T}{\partial n} = -q_a$
3. Prescribed Convection (Convective): $k\frac{\partial T}{\partial n} = -h(T - T_{ref})$

where the values T_a, q_a, and h signify prescribed values of temperature, flux, and convection coefficient, respectively, and $\partial/\partial n$ denotes derivative with respect to the outward-drawn normal to the boundary, $\boldsymbol{n}$. Thus, each of these boundary conditions must be formulated in a similar, meshless framework form as the governing equations.

For the case of a prescribed temperature boundary condition, it becomes immediately clear that the standard form of $[A]\{u\} = \{b\}$ can be developed directly without any manipulation. Because the temperature value at the data center is directly known, these terms are simply given by

$$[A] = [1] \qquad \{u\} = \{T_c\}^{\langle k \rangle} \qquad \{b\} = \{T_a\} \tag{4.7}$$

Obviously, for this case the temperature at the data center is only dependant on the prescribed value, and does not require any additional information.

For the case of prescribed flux on a boundary, steps similar to those used to develop the expressions for the governing equation must be used. Thus, rewriting the boundary condition with the meshless representation of the normal differential operator results in

$$\left\{\widehat{\Psi}_n\right\}^{\mathrm{T}}\left\{\widehat{T}\right\} = -\frac{q_a}{k} \tag{4.8}$$

where $\widehat{\Psi}_n$ represents the weight values associated with the normal derivative operator, and $\widehat{T}$ represents the nodal temperature values. Thus, by extracting the data center information from this vectors, the following factored expression can be found

$$\psi_n T_c + \left\{\widetilde{\Psi}_n\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\} = -\frac{q_a}{k} \tag{4.9}$$

Once again, the tilde indicates that the data center values have been removed from these topology vectors. Thus, by lagging the temperature values other than those associated with the data center, the final meshless framework form of the prescribed flux boundary conditions can be represented as

$$[A] = [\psi_n] \qquad \{u\} = \{T_c\}^{\langle k \rangle} \tag{4.10}$$

$$\{b\} = \left\{ -\frac{q_a}{k} - \left\{\widetilde{\Psi}_n\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\}^{\langle k-1 \rangle} \right\}$$

Lastly, for the case of prescribed convection on a boundary, the process begins the same; by separating the already known and prescribed values from the unknown temperature field

$$k\frac{\partial T}{\partial n} + hT = hT_{ref} \tag{4.11}$$

Dividing Equation 4.11 by k, to isolate the boundary normal derivative, and substituting the meshless operator for the normal derivative results in the following expression

$$\left\{\widehat{\Psi}_n\right\}^{\mathrm{T}}\left\{\widehat{T}\right\} + \frac{h}{k}T_c = \frac{h}{k}T_{ref} \tag{4.12}$$

Thus, performing similar factorization as in Equations 4.4 and 4.9 results in the following expression

$$\psi_n T_c + \frac{h}{k}T_c + \left\{\widetilde{\Psi}_n\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\} = \frac{h}{k}T_{ref} \tag{4.13}$$

Combining coefficients associated with the temperature at the data center, and lagging the $\widetilde{T}$ vector (as previously described) results in the following meshless framework form of the convective boundary condition

$$[A] = \left[\psi_n + \frac{h}{k}\right] \qquad \{u\} = \{T_c\}^{\langle k\rangle} \tag{4.14}$$

$$\{b\} = \left\{\frac{h}{k}T_{ref} - \left\{\widetilde{\Psi}_n\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\}^{\langle k-1\rangle}\right\}$$

Therefore, it becomes clear that regardless of the type of node (internal or boundary) the meshless framework formulation still results in solving a system of equations in the standard form $[A]\{u\} = \{b\}$. Note that in the case of steady-state heat conduction, the size of this system is 1×1, which is trivial to solve directly. However, it is important that the generality of this method not be overlooked as this is what allows the aforementioned framework to be applicable to any governing equation that can be formulated in this manner.

CHAPTER 5
APPLICATION TO IDEAL FLOW WITH HEAT ADVECTION-DIFFUSION

The next formulation that will be presented in this paper is that of steady-state ideal flow with coupled heat advection-diffusion. This set of equations is unique from the previous example of standard heat conduction as it involves upwinded first derivatives, something that is very well suited for the Virtual RBF Finite Difference techniques.

5.1. FORMULATION OF GOVERNING EQUATION

Ideal flow with heat advection-diffusion is an interesting set of governing equations as there is a strictly one-way coupling between the temperature field and the velocity field. As such, the velocity field is completely independent of the temperature field and can be solved for before any temperature solutions are determined. Additionally, the velocity field can be solved for as a scalar potential function where the components of the velocity are dependant on the derivatives of the field. Therefore, the governing equation for this case can be represented as

$$\nabla^2\phi = 0 \text{ where } \boldsymbol{V} = -\nabla\phi = \begin{cases} u = -\dfrac{\partial\phi}{\partial x} \\ v = -\dfrac{\partial\phi}{\partial y} \\ w = -\dfrac{\partial\phi}{\partial z} \end{cases} \tag{5.1}$$

$$\boldsymbol{V}\cdot\nabla T = \frac{k}{\rho C_p}\nabla^2 T + \frac{q'''}{\rho C_p}$$

where ϕ is the potential function of the velocity field, u, v, and w are the components of the velocity field, $\boldsymbol{V}$, in the x, y, and z directions, respectively, T is the temperature field, k is the thermal conductivity of the fluid, ρ is the density of the fluid, C_p is the

convection coefficient of the fluid, and q''' is the heat generation per unit volume in the flow.

Thus, it is clear that the first step in solving these equations is to determine the velocity potential, ϕ. To do this, Equation 3.7 can be directly implemented to solve the Laplace equation,

$$[A] = [\,\psi_{\nabla^2}] \qquad \{u\} = \{\phi_c\}^{\langle k \rangle} \tag{5.2}$$

$$\{b\} = \Big\{ - \{\widetilde{\Psi}_{\nabla^2}\}^{\mathrm{T}} \{\widetilde{\phi}\} \Big\}^{\langle k-1 \rangle}$$

Once the potential field has been iterated to a sufficient stopping criteria, the next step is to find the velocity field components from the expression $\boldsymbol{V} = -\nabla\phi$. To accomplish this, central differenced, Virtual RBF formulations can be developed to simply evaluate the derivatives directly at each node. The form would be similar to Equation 2.22 except that it would be for the first derivatives, instead of the second. Thus, these values can be represented as a set of vector weights multiplied by a set of vector nodal values

$$\begin{aligned} u &= -\frac{\partial \phi}{\partial x} = \{\Psi_x\}^{\mathrm{T}} \{\phi\} \\ v &= -\frac{\partial \phi}{\partial y} = \{\Psi_y\}^{\mathrm{T}} \{\phi\} \\ w &= -\frac{\partial \phi}{\partial z} = \{\Psi_z\}^{\mathrm{T}} \{\phi\} \end{aligned} \tag{5.3}$$

Having solved for the velocity components, the final step is to solve for the temperature field within the domain. To do this, the following expression for the temperature must first be placed into the meshless framework form

$$\boldsymbol{V} \cdot \nabla T = \frac{k}{\rho C_p} \nabla^2 T + \frac{q'''}{\rho C_p} \tag{5.4}$$

First, expanding the vector form of this equation and performing some rearranging results in

$$\frac{k}{\rho C_p}\nabla^2 T - u\frac{\partial T}{\partial x} - v\frac{\partial T}{\partial y} - w\frac{\partial T}{\partial z} = -\frac{q'''}{\rho C_p} \tag{5.5}$$

Note that in this equation, the first derivative terms multiplying the velocity components are the convective derivative terms and must be upwinded in order to accurately compute the values. To do this, the virtual nodes that are used in the collocations are projected into the flow direction, which, in this case, will always be static since the velocity field will be solved for prior to determining the temperature. Therefore, understanding that the first derivative terms are upwinded, the meshless representations of the derivative operators can be substituted, resulting in the following equation

$$\frac{k}{\rho C_p}\left\{\widehat{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widehat{T}\} - u\left\{\widehat{\Psi}_x\right\}^{\mathrm{T}}\{\widehat{T}\} - v\left\{\widehat{\Psi}_y\right\}^{\mathrm{T}}\{\widehat{T}\} - w\left\{\widehat{\Psi}_z\right\}^{\mathrm{T}}\{\widehat{T}\} = -\frac{q'''}{\rho C_p} \tag{5.6}$$

The next step is to factor out the value at the data center from each of these vector-vector expressions,

$$\begin{aligned}&\frac{k}{\rho C_p}\psi_{\nabla^2}T_c - u\psi_x T_c - v\psi_y T_c - w\psi_z T_c = \\ &u\left\{\widetilde{\Psi}_x\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\} + v\left\{\widetilde{\Psi}_y\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\} + w\left\{\widetilde{\Psi}_z\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\} - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\left\{\widetilde{T}\right\} - \frac{q'''}{\rho C_p}\end{aligned} \tag{5.7}$$

At this point it is clear that this equation can be placed into the standard matrix form represented in Equation 3.6 where the contents are given by

$$[A] = \left[\frac{k}{\rho C_p}\psi_{\nabla^2} - u\psi_x - v\psi_y - w\psi_z\right] \qquad \{u\} = \{T_c\}^{\langle k\rangle} \tag{5.8}$$

$$\{b\} = \left\{\left(u\left\{\widetilde{\Psi}_x\right\}^{\mathrm{T}} + v\left\{\widetilde{\Psi}_y\right\}^{\mathrm{T}} + w\left\{\widetilde{\Psi}_z\right\}^{\mathrm{T}} - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\right)\left\{\widetilde{T}\right\} - \frac{q'''}{\rho C_p}\right\}^{\langle k-1\rangle}$$

Therefore the solution process for the case of steady-state, ideal flow with heat advection-diffusion is to first solve Equation 5.2 for the velocity potential ϕ. Then,

having found the value of ϕ everywhere within the field, the velocity field can be calculated using the expressions shown in Equation 5.3 by implementing post processing operators (similar to those used to solve the governing equations). Finally, having determined the velocity field everywhere within the domain, the final step is to use Equation 5.8 to determine the corresponding temperature field. This process can be completed in a single structure such that the combined functions work as a single, cohesive model.

5.2. FORMULATION OF BOUNDARY CONDITIONS

Having formulated the governing equations, the next step is to do the same for the various boundary condition types. Because these two solvers are coupled only one way, the boundary conditions can be developed separately for the ideal flow and heat advection-diffusion. Therefore, for ideal flow there are three types of boundary conditions

1. Inlet (Prescribed Velocity): $V_n = V_{n_a}$
2. Slip Wall: $\frac{\partial \phi}{\partial n} = 0$
3. Outlet $\phi = 0$

where V_{n_a} represents a prescribed normal velocity magnitude at the boundary. Note, however, that because the problem is solved using the velocity potential, ϕ, this can be expressed in terms of the normal derivative as well:

1. Inlet (Prescribed Velocity): $\frac{\partial \phi}{\partial n} = V_{n_a}$
2. Slip Wall: $\frac{\partial \phi}{\partial n} = 0$
3. Outlet $\phi = 0$

Thus, for the case of idea flow, only natural and null conditions need to be placed on the velocity potential, which can be expressed in the following forms:

$$[A] = [\,\psi_n] \qquad \{u\} = \{\phi_c\}^{\langle k \rangle} \tag{5.9}$$
$$\{b\} = \left\{\mathcal{C} - \left\{\widetilde{\Psi}_n\right\}^{\mathrm{T}} \left\{\widetilde{\phi}\right\}^{\langle k-1 \rangle}\right\}$$

where the constant $\mathcal{C}$ is either the magnitude of the applied normal velocity (in the case of prescribed inlet velocity), or 0 (in the case of a wall), and for the null condition:

$$[A] = [\,1] \qquad \{u\} = \{\phi_c\}^{\langle k \rangle} \tag{5.10}$$
$$\{b\} = \{0\}$$

As far as the heat advection-diffusion conditions are concerned, they are very similar to those that were developed in Chapter 4 for heat conduction, however, it is important to realize that the outlet condition is a special case of a prescribed flux. Therefore, the boundary conditions for this problem are

1. Prescribed Temperature (Forced): $T = T_a$
2. Prescribed Flux (Natural): $k\frac{\partial T}{\partial n} = -q_a$
3. Prescribed Convection (Convective): $k\frac{\partial T}{\partial n} = -h(T - T_{ref})$
4. Outlet (Natural): $\frac{\partial T}{\partial n} = 0$

here it can be seen that the first three conditions are the same as those developed in Chapter 4, and the outlet condition is a special case of prescribed flux where $q_a = 0$ (insulated). It is important to note that although this is not an ideal condition for the outlet because the temperature can reflect back into the domain, it will provide correct solutions assuming the velocity and temperature fields are fully developed. Optimally, the governing equation should be solved at the outlet boundary, however, this causes stability concerns at the affected nodes.

Therefore, repeating the solutions obtained in Chapter 4, for the case of prescribed temperature the local implicit formulation will be

$$[A] = [\,1] \qquad \{u\} = \{T_c\}^{\langle k \rangle} \qquad \{b\} = \{T_a\} \tag{5.11}$$

for the case of prescribed flux (and outlet)

$$[A] = [\,\psi_n] \qquad \{u\} = \{T_c\}^{\langle k \rangle} \tag{5.12}$$

$$\{b\} = \left\{ \mathcal{C} - \{\widetilde{\Psi}_n\}^{\mathrm{T}} \{\widetilde{T}\}^{\langle k-1 \rangle} \right\}$$

where $\mathcal{C}$ is either $-q_a/k$ (for prescribed flux) or 0 (for outlet) and for the case of prescribed convection,

$$[A] = \left[\psi_n + \frac{h}{k} \right] \qquad \{u\} = \{T_c\}^{\langle k \rangle} \tag{5.13}$$

$$\{b\} = \left\{ \frac{h}{k} T_{ref} - \{\widetilde{\Psi}_n\}^{\mathrm{T}} \{\widetilde{T}\}^{\langle k-1 \rangle} \right\}$$

Therefore, the boundary conditions and governing equations have now been formulated into the meshless framework form for the solution of ideal flow with heat advection-diffusion. It is important to realize that the velocity solution is independent of the temperature field, and as such, can be solved for in terms of a velocity potential, which can then be post-processed to determine the actual velocity components. Additionally, because of this one-way coupling, the heat convection equations remain a linear set of governing equations and can be developed into the local implicit form used throughout this paper.

CHAPTER 6
APPLICATION TO STEADY-STATE ELASTICITY

The final formulation that will be presented is that of steady-state elasticity. Elasticity is a departure from the previous governing equations because the deflection components will now be fully coupled and must be solved for simultaneously. Therefore, the flexibility and generality of the meshless framework formulation techniques can be demonstrated as it is extended to a case with more than one simultaneous unknown.

6.1. FORMULATION OF GOVERNING EQUATION

The formulation begins, as always, from the governing equation, which, for steady-state elasticity, is the vector formulation of the Navier equations [35], with constant body force, given as

$$\nabla^2 \boldsymbol{u} + \frac{1}{(1-2\nu)} \nabla(\nabla \cdot \boldsymbol{u}) = -\frac{\boldsymbol{F}}{\mu} \tag{6.1}$$

where $\boldsymbol{u}$ is the deflection vector, ν is the value of Poisson's ratio for the given material, μ is the shear modulus given as $\mu = E/[2(1+\nu)]$, and $\boldsymbol{F}$ is the prescribed body force vector. Thus, from Equation 6.1, by making the substitution that $\beta = 1/(1-2\nu)$ and replacing the negative of the body force divided by the shear modulus by $\boldsymbol{\mathcal{F}}$, the following equation can be developed

$$\nabla^2 \boldsymbol{u} + \beta \nabla(\nabla \cdot \boldsymbol{u}) = \boldsymbol{\mathcal{F}} \tag{6.2}$$

At this point it becomes necessary to expand the vector formulation into three-dimensions such that the meshless operators can be substituted for the derivative terms. Thus, expanding Equation 6.2 about the x, y, and z directions results in

$$\nabla^2 u + \beta\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial x \partial y} + \frac{\partial^2 w}{\partial x \partial z}\right) = \mathcal{F}_x \tag{6.3}$$
$$\nabla^2 v + \beta\left(\frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 w}{\partial y \partial z}\right) = \mathcal{F}_y$$
$$\nabla^2 w + \beta\left(\frac{\partial^2 u}{\partial x \partial z} + \frac{\partial^2 v}{\partial y \partial z} + \frac{\partial^2 w}{\partial z^2}\right) = \mathcal{F}_z$$

where u, v, and w are the deflection components and $\mathcal{F}_x$, $\mathcal{F}_y$, and $\mathcal{F}_z$ are the body force components in the x, y, and z directions, respectively.

Equation 6.3 represents the governing equation implemented in the current version of the meshless framework detailed in this paper. The next step is to decompose the governing equation into the meshless framework form outlined in Chapter 3.

As previously described, the differential operators must now be replaced by the equivalent numerical representations, in the form of a vector of weights multiplied by a vector of nodal field values. In the case of steady-state elasticity, there are several differential operators present, the Laplacian, ∇^2, the second derivatives, $\partial^2/\partial x^2$, $\partial^2/\partial y^2$, and $\partial^2/\partial z^2$, as well as the cross derivatives, $\partial^2/\partial x \partial$y, $\partial^2/\partial y \partial z$, and $\partial^2/\partial y \partial z$. Thus, replacing these operators by the equivalent vector-vector representation gives

$$\left\{\widehat{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widehat{u}\} + \beta\left(\left\{\widehat{\Psi}_{xx}\right\}^{\mathrm{T}}\{\widehat{u}\} + \left\{\widehat{\Psi}_{xy}\right\}^{\mathrm{T}}\{\widehat{v}\} + \left\{\widehat{\Psi}_{xz}\right\}^{\mathrm{T}}\{\widehat{w}\}\right) = \mathcal{F}_x \tag{6.4a}$$

$$\left\{\widehat{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widehat{v}\} + \beta\left(\left\{\widehat{\Psi}_{xy}\right\}^{\mathrm{T}}\{\widehat{u}\} + \left\{\widehat{\Psi}_{yy}\right\}^{\mathrm{T}}\{\widehat{v}\} + \left\{\widehat{\Psi}_{yz}\right\}^{\mathrm{T}}\{\widehat{w}\}\right) = \mathcal{F}_y \tag{6.4b}$$

$$\left\{\widehat{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widehat{w}\} + \beta\left(\left\{\widehat{\Psi}_{xz}\right\}^{\mathrm{T}}\{\widehat{u}\} + \left\{\widehat{\Psi}_{yz}\right\}^{\mathrm{T}}\{\widehat{v}\} + \left\{\widehat{\Psi}_{zz}\right\}^{\mathrm{T}}\{\widehat{w}\}\right) = \mathcal{F}_z \tag{6.4c}$$

At this stage, it is clear that the vector for the meshless differential operators, as well as the unknown deflection values, contain the current topology data center (as indicated by the hat symbol). Thus, factoring out the weights and deflection values associated at the data center from these equations results in

$$\psi_{\nabla^2} u_c + \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{u}\} + \beta\psi_{xx} u_c + \beta\left\{\widetilde{\Psi}_{xx}\right\}^{\mathrm{T}}\{\widetilde{u}\} + \beta\psi_{xy} v_c + \beta\left\{\widetilde{\Psi}_{xy}\right\}^{\mathrm{T}}\{\widetilde{v}\} + \beta\psi_{xz} w_c + \beta\left\{\widetilde{\Psi}_{xz}\right\}^{\mathrm{T}}\{\widetilde{w}\} = \mathcal{F}_x \tag{6.5a}$$

$$\psi_{\nabla^2} v_c + \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{v}\} + \beta\psi_{xy} u_c + \beta\left\{\widetilde{\Psi}_{xy}\right\}^{\mathrm{T}}\{\widetilde{u}\} + \beta\psi_{yy} v_c + \beta\left\{\widetilde{\Psi}_{yy}\right\}\{\widetilde{v}\} + \beta\psi_{yz} w_c + \beta\left\{\widetilde{\Psi}_{yz}\right\}^{\mathrm{T}}\{\widetilde{w}\} = \mathcal{F}_y \tag{6.5b}$$

$$\psi_{\nabla^2} w_c + \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{w}\} + \beta\psi_{xz} u_c + \beta\left\{\widetilde{\Psi}_{xz}\right\}^{\mathrm{T}}\{\widetilde{u}\} + \beta\psi_{yz} v_c + \beta\left\{\widetilde{\Psi}_{yz}\right\}^{\mathrm{T}}\{\widetilde{v}\} + \beta\psi_{zz} w_c + \beta\left\{\widetilde{\Psi}_{zz}\right\}^{\mathrm{T}}\{\widetilde{w}\} = \mathcal{F}_z \tag{6.5c}$$

Thus, Equations 6.5a, 6.5b, and 6.5c are fully coupled in all directions with respect to u_c, v_c, and w_c. As discussed earlier, rather than lag specific terms behind in each equation, the unknown deflection values at the data center can be solved for in an implicit fashion. Thus, by collecting unknown values to the left and known values (for a given iteration) to the right, Equations 6.5a, 6.5b, and 6.5c can be rearranged to give

$$\psi_{\nabla^2} u_c + \beta\psi_{xx} u_c + \beta\psi_{xy} v_c + \beta\psi_{xz} w_c = \mathcal{F}_x - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{u}\} - \beta\left\{\widetilde{\Psi}_{xx}\right\}^{\mathrm{T}}\{\widetilde{u}\} - \beta\left\{\widetilde{\Psi}_{xy}\right\}^{\mathrm{T}}\{\widetilde{v}\} - \beta\left\{\widetilde{\Psi}_{xz}\right\}^{\mathrm{T}}\{\widetilde{w}\} \tag{6.6a}$$

$$\beta\psi_{xy} u_c + \psi_{\nabla^2} v_c + \beta\psi_{yy} v_c + \beta\psi_{yz} w_c = \mathcal{F}_y - \beta\left\{\widetilde{\Psi}_{xy}\right\}^{\mathrm{T}}\{\widetilde{u}\} - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{v}\} - \beta\left\{\widetilde{\Psi}_{yy}\right\}^{\mathrm{T}}\{\widetilde{v}\} - \beta\left\{\widetilde{\Psi}_{yz}\right\}^{\mathrm{T}}\{\widetilde{w}\} \tag{6.6b}$$

$$\beta\psi_{xz} u_c + \beta\psi_{yz} v_c + \psi_{\nabla^2} w_c + \beta\psi_{zz} w_c = \mathcal{F}_z - \beta\left\{\widetilde{\Psi}_{xz}\right\}^{\mathrm{T}}\{\widetilde{u}\} - \beta\left\{\widetilde{\Psi}_{yz}\right\}^{\mathrm{T}}\{\widetilde{v}\} - \left\{\widetilde{\Psi}_{\nabla^2}\right\}^{\mathrm{T}}\{\widetilde{w}\} - \beta\left\{\widetilde{\Psi}_{zz}\right\}^{\mathrm{T}}\{\widetilde{w}\} \tag{6.6c}$$

These equations can now be directly formulated in the standard equation form if the tilde terms are all lagged to the previous iteration. Thus, the final formulation of the elasticity governing equation can be expressed as follows

$$[A] = \begin{bmatrix} \psi_{\nabla^2} + \beta\psi_{xx} & \beta\psi_{xy} & \beta\psi_{xz} \\ \beta\psi_{xy} & \psi_{\nabla^2} + \beta\psi_{yy} & \beta\psi_{yz} \\ \beta\psi_{xz} & \beta\psi_{yz} & \psi_{\nabla^2} + \beta\psi_{zz} \end{bmatrix} \qquad \{u\} = \begin{Bmatrix} u_c \\ v_c \\ w_c \end{Bmatrix}^{\langle k \rangle} \tag{6.7}$$

$$\{b\} = \begin{Bmatrix} \mathcal{F}_x - \left(\{\widetilde{\Psi}_{\nabla^2}\}^{\mathrm{T}} + \beta\{\widetilde{\Psi}_{xx}\}^{\mathrm{T}}\right)\{\widetilde{u}\} - \beta\{\widetilde{\Psi}_{xy}\}^{\mathrm{T}}\{\widetilde{v}\} - \beta\{\widetilde{\Psi}_{xz}\}^{\mathrm{T}}\{\widetilde{w}\} \\ \mathcal{F}_y - \beta\{\widetilde{\Psi}_{xy}\}^{\mathrm{T}}\{\widetilde{u}\} - \left(\{\widetilde{\Psi}_{\nabla^2}\}^{\mathrm{T}} + \beta\{\widetilde{\Psi}_{yy}\}^{\mathrm{T}}\right)\{\widetilde{v}\} - \beta\{\widetilde{\Psi}_{yz}\}^{\mathrm{T}}\{\widetilde{w}\} \\ \mathcal{F}_z - \beta\{\widetilde{\Psi}_{xz}\}^{\mathrm{T}}\{\widetilde{u}\} - \beta\{\widetilde{\Psi}_{yz}\}^{\mathrm{T}}\{\widetilde{v}\} - \left(\{\widetilde{\Psi}_{\nabla^2}\}^{\mathrm{T}} + \beta\{\widetilde{\Psi}_{zz}\}^{\mathrm{T}}\right)\{\widetilde{w}\} \end{Bmatrix}^{\langle k-1 \rangle}$$

Therefore, similar to the equations formulated for steady-state heat conduction and ideal flow with heat advection-diffusion, this formulation is capable of implicitly solving for the unknown deflections at a given data center based on deflection values for the surrounding topology from the previous iteration. Also, by formulating this governing equation in this form, the system that we are solving becomes more diagonally dominant, and therefore, more stable. Additionally, by allowing coupling of the field variables, the solution takes fewer iterations to converge, and thus, can reduce computational time as well.

6.2. FORMULATION OF BOUNDARY CONDITIONS

Having developed the governing equation formulation for steady-state elasticity, the next step is to develop a similar formulation for the boundary conditions. In the case of steady-state elasticity, there are three generalized boundary conditions:

1. Prescribed Deflection: $\boldsymbol{u} = \boldsymbol{u}_a$
2. Prescribed Traction: $\boldsymbol{t} = \boldsymbol{t}_a$
3. Mixed Condition (combination of 1 and 2)

where the notation $\boldsymbol{u}_a$, and $\boldsymbol{t}_a$ signify prescribed vector values of deflection and surface traction, respectively. Thus, each of these boundary conditions must be formulated in a similar, form to the governing equations.

For the case of a prescribed deflection boundary condition, it becomes immediately clear that the standard form of $[A]\{u\} = \{b\}$ can be developed directly without any manipulation. Because the deflection value at the data center is applied, these terms are simply given by

$$[A] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \{u\} = \begin{Bmatrix} u_c \\ v_c \\ w_c \end{Bmatrix}^{\langle k \rangle} \qquad \{b\} = \begin{Bmatrix} u_a \\ v_a \\ w_a \end{Bmatrix} \tag{6.8}$$

This condition is most commonly used for the case of a fixed, or clamped boundary where $u_a = v_a = w_a = 0$. Also note that since the deflections are prescribed in terms of the x, y, and z components, no transformations need to take place in order to properly orient the results.

To develop the boundary equations for the case of prescribed traction, it is first necessary to realize that when tractions are applied, they will almost always be applied relative to the surface directions. In other words, rather than applying traction components in the x, y, and z directions, they will instead be applied in the normal and tangential directions. This has to do with the fact that normal tractions are physically applied as a pressure and tangential tractions are usually generated because of some component of frictional forces. Therefore, rather than formulate these equations in terms of u, v, and w, they must instead be formulated into deflection components in the surface aligned directions. Therefore, to help illustrate this concept, a representative surface is shown below in Figure 6.1.

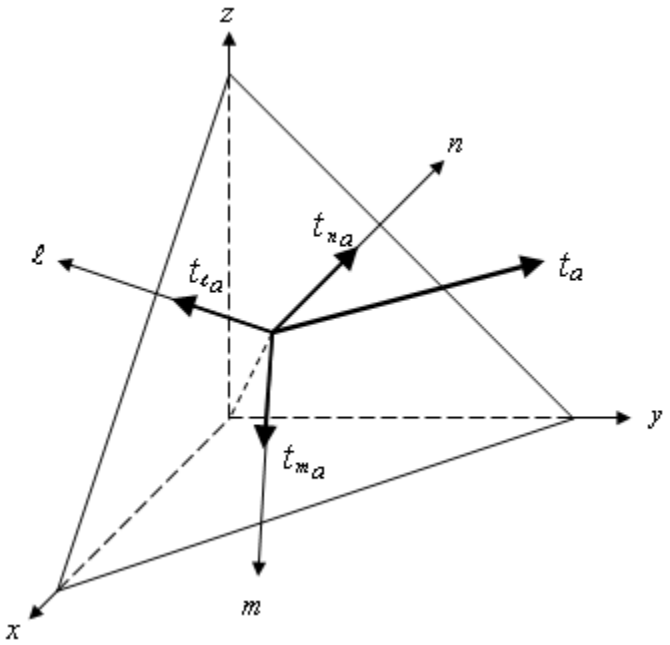

Figure 6.1. Resolution of Applied Traction into Surface Components

Therefore, from this figure, the resolution of the applied traction, $\boldsymbol{t}_a$, into the ℓ, m, and n directions can be expressed as:

$$\begin{Bmatrix} t_\ell \\ t_m \\ t_n \end{Bmatrix} = \begin{bmatrix} \sigma_{\ell\ell} & \sigma_{\ell m} & \sigma_{\ell n} \\ \sigma_{\ell m} & \sigma_{mm} & \sigma_{mn} \\ \sigma_{\ell n} & \sigma_{mn} & \sigma_{nn} \end{bmatrix} \begin{Bmatrix} n_\ell \\ n_m \\ n_n \end{Bmatrix} \tag{6.9}$$

where n_ℓ, n_m, and n_n are the components of the surface normal associated with the orthogonal ℓ-m-n system. However, it becomes clear that because the ℓ-m-n system was chosen specifically such that the n direction aligned with the surface normal, $n_\ell = n_m = 0$ and $n_n = 1$. Therefore, Equation 6.9 can be simplified to

$$\begin{Bmatrix} t_\ell \\ t_m \\ t_n \end{Bmatrix} = \begin{bmatrix} \sigma_{\ell\ell} & \sigma_{\ell m} & \sigma_{\ell n} \\ \sigma_{\ell m} & \sigma_{mm} & \sigma_{mn} \\ \sigma_{\ell n} & \sigma_{mn} & \sigma_{nn} \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} = \begin{Bmatrix} \sigma_{\ell n} \\ \sigma_{mn} \\ \sigma_{nn} \end{Bmatrix} \tag{6.10}$$

Therefore, the traction components can be written directly as

$$t_\ell = \sigma_{\ell n} \qquad t_m = \sigma_{mn} \qquad t_n = \sigma_{nn} \tag{6.11}$$

Now, in order to get by Equation 6.11 in terms of deflections (instead of stresses), the first step is to substitute the stress-strain relationships. Thus, in three dimensions, the stress strain equations [36-37] can be expressed in the ℓ-m-n coordinate system as

$$\sigma_{\ell\ell} = \frac{E}{(1+\nu)(1-2\nu)}[(1-\nu)\epsilon_{\ell\ell} + \nu(\epsilon_{mm} + \epsilon_{nn})] \tag{6.12}$$

$$\sigma_{mm} = \frac{E}{(1+\nu)(1-2\nu)}[(1-\nu)\epsilon_{mm} + \nu(\epsilon_{\ell\ell} + \epsilon_{nn})]$$

$$\sigma_{nn} = \frac{E}{(1+\nu)(1-2\nu)}[(1-\nu)\epsilon_{nn} + \nu(\epsilon_{\ell\ell} + \epsilon_{mm})]$$

$$\sigma_{\ell m} = \frac{E}{2(1+\nu)}\gamma_{\ell m} \quad \sigma_{\ell n} = \frac{E}{2(1+\nu)}\gamma_{\ell n} \quad \sigma_{mn} = \frac{E}{2(1+\nu)}\gamma_{mn}$$

where E is the Modulus of Elasticity for the material and ν is Poisson's Ratio. Substituting these relationships into Equation 6.11 results in the following expressions

$$t_\ell = \frac{E}{2(1+\nu)}\gamma_{\ell n} = G\gamma_{\ell n} \tag{6.13}$$

$$t_m = \frac{E}{2(1+\nu)}\gamma_{mn} = G\gamma_{mn}$$

$$t_n = \frac{E}{(1+\nu)(1-2\nu)}[(1-\nu)\epsilon_{nn} + \nu(\epsilon_{\ell\ell} + \epsilon_{mm})]$$

where G is the Shear Modulus given by $G = E/[2(1+\nu)]$. At this point, Equation 6.13 is now in terms of strains, but not deflections. Therefore, the strain displacement relationships for three-dimensions are given as follows,

$$\epsilon_{\ell\ell} = \frac{\partial a}{\partial \ell} \qquad \epsilon_{mm} = \frac{\partial b}{\partial m} \qquad \epsilon_{nn} = \frac{\partial c}{\partial n} \tag{6.14}$$

$$\gamma_{\ell m} = \frac{\partial b}{\partial \ell} + \frac{\partial a}{\partial m} \quad \gamma_{\ell n} = \frac{\partial c}{\partial \ell} + \frac{\partial a}{\partial n} \quad \gamma_{mn} = \frac{\partial b}{\partial n} + \frac{\partial c}{\partial m}$$

where a, b, and c are the deflection components in the ℓ, m, and n directions, respectively. Thus, plugging the relationships in Equation 6.14 into Equation 6.13 results in the following

$$
\begin{aligned}
t_\ell &= G\left(\frac{\partial c}{\partial \ell} + \frac{\partial a}{\partial n}\right) \\
t_m &= G\left(\frac{\partial b}{\partial n} + \frac{\partial c}{\partial m}\right) \\
t_n &= \frac{E}{(1+\nu)(1-2\nu)}\left[(1-\nu)\frac{\partial c}{\partial n} + \nu\left(\frac{\partial a}{\partial \ell} + \frac{\partial b}{\partial m}\right)\right]
\end{aligned}
\tag{6.15}
$$

Thus, Equation 6.15 represents the boundary equations for an applied traction vector. As usual, these equations can be placed into the meshless framework form, by replacing the derivative operators by their meshless representation, as shown

$$
\begin{aligned}
t_\ell &= G\left(\left\{\widehat{\Psi}_\ell\right\}^{\mathrm{T}}\{\widehat{c}\} + \left\{\widehat{\Psi}_n\right\}^{\mathrm{T}}\{\widehat{a}\}\right) \\
t_m &= G\left(\left\{\widehat{\Psi}_n\right\}^{\mathrm{T}}\left\{\widehat{b}\right\} + \left\{\widehat{\Psi}_m\right\}^{\mathrm{T}}\{\widehat{c}\}\right) \\
t_n &= H\left[(1-\nu)\left\{\widehat{\Psi}_n\right\}^{\mathrm{T}}\{\widehat{c}\} + \nu\left(\left\{\widehat{\Psi}_\ell\right\}^{\mathrm{T}}\{\widehat{a}\} + \left\{\widehat{\Psi}_m\right\}^{\mathrm{T}}\left\{\widehat{b}\right\}\right)\right]
\end{aligned}
\tag{6.16}
$$

where $H = E/[(1+\nu)(1-2\nu)]$ and the respective $\widehat{\Psi}$ vectors represent the weight vectors for the boundary topologies. Thus, extracting the values for a, b, and c at the data center from these expressions and rearranging to form a matrix set of equations (as described in Chapter 3), results in the following local implicit formulation of an applied traction condition,

$$
[A] = \begin{bmatrix} G\psi_n & 0 & G\psi_\ell \\ 0 & G\psi_n & G\psi_m \\ H\nu\psi_\ell & H\nu\psi_{_m} & H(1-\nu)\psi_n \end{bmatrix} \qquad \{u'\} = \begin{Bmatrix} a_c \\ b_c \\ c_c \end{Bmatrix}^{\langle k \rangle} \tag{6.17}
$$

$$
\{b\} = \begin{Bmatrix} t_\ell - G\left\{\widetilde{\Psi}_n\right\}^{\mathrm{T}}\{\widetilde{a}\} - G\left\{\widetilde{\Psi}_\ell\right\}^{\mathrm{T}}\{\widetilde{c}\} \\ t_m - G\left\{\widetilde{\Psi}_n\right\}^{\mathrm{T}}\left\{\widetilde{b}\right\} - G\left\{\widetilde{\Psi}_m\right\}^{\mathrm{T}}\{\widetilde{c}\} \\ t_n - H\nu\left\{\widetilde{\Psi}_\ell\right\}^{\mathrm{T}}\{\widetilde{a}\} - H\nu\left\{\widetilde{\Psi}_m\right\}^{\mathrm{T}}\left\{\widetilde{b}\right\} - H(1-\nu)\left\{\widetilde{\Psi}_n\right\}^{\mathrm{T}}\{\widetilde{c}\} \end{Bmatrix}^{\langle k-1 \rangle}
$$

It is important to note that although Equation 6.17 allows for solutions on the boundary relative to the surface normal and tangential directions, it does not allow for solutions in the x, y, and z directions. Therefore, in order to both construct the differential operator vectors (since values of a, b, and c will be required at any node in the topology), as well as transform the final values at the boundary node back to x-y-z coordinates, the following transformation must be applied

$$\begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = \begin{bmatrix} \boldsymbol{e_i} \cdot \boldsymbol{e_\ell} & \boldsymbol{e_j} \cdot \boldsymbol{e_\ell} & \boldsymbol{e_k} \cdot \boldsymbol{e_\ell} \\ \boldsymbol{e_i} \cdot \boldsymbol{e_m} & \boldsymbol{e_j} \cdot \boldsymbol{e_m} & \boldsymbol{e_k} \cdot \boldsymbol{e_m} \\ \boldsymbol{e_i} \cdot \boldsymbol{e_n} & \boldsymbol{e_j} \cdot \boldsymbol{e_n} & \boldsymbol{e_k} \cdot \boldsymbol{e_n} \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = [Q] \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{6.18}$$

and similarly for the backward transformation

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{bmatrix} \boldsymbol{e_i} \cdot \boldsymbol{e_\ell} & \boldsymbol{e_j} \cdot \boldsymbol{e_\ell} & \boldsymbol{e_k} \cdot \boldsymbol{e_\ell} \\ \boldsymbol{e_i} \cdot \boldsymbol{e_m} & \boldsymbol{e_j} \cdot \boldsymbol{e_m} & \boldsymbol{e_k} \cdot \boldsymbol{e_m} \\ \boldsymbol{e_i} \cdot \boldsymbol{e_n} & \boldsymbol{e_j} \cdot \boldsymbol{e_n} & \boldsymbol{e_k} \cdot \boldsymbol{e_n} \end{bmatrix}^{\mathrm{T}} \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = [Q]^{\mathrm{T}} \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} \tag{6.19}$$

where $\boldsymbol{e_i}$, $\boldsymbol{e_j}$, and $\boldsymbol{e_k}$ are the unit vectors of the x-y-z coordinate system, and $\boldsymbol{e_\ell}$, $\boldsymbol{e_m}$, and $\boldsymbol{e_n}$ are the unit vectors of the ℓ-m-n system expressed in the x-y-z system. Therefore, by using Equation 6.17, in conjunction with the forward and backward transformations listed in Equations 6.18 and 6.19, the case of applied tractions can be solved on the boundary. However, rather than perform these transformations on an ad hoc basis, they too can be incorporated into the differential operators needed to solve the meshless framework system of equations. Therefore, by substituting the transposed rotation matrix, $[Q]^{\mathrm{T}}$, into Equation 3.6, the following expression can be formed

$$[A][Q]^{\mathrm{T}}\{u'\} = [A']\{u\} = \{b\} \tag{6.20}$$

where the A' matrix will now return deflections in x-y-z space, instead of ℓ-m-n. Therefore, Equation 6.17 now becomes:

$$[A'] = \begin{bmatrix} G\psi_n & 0 & G\psi_\ell \\ 0 & G\psi_n & G\psi_m \\ H\nu\psi_\ell & H\nu\psi_m & H(1-\nu)\psi_n \end{bmatrix} [Q]^{\mathrm{T}} \qquad \{u\} = \begin{Bmatrix} u_c \\ v_c \\ w_c \end{Bmatrix}^{\langle k \rangle} \tag{6.21}$$

$$\{b\} = \begin{Bmatrix} t_\ell - G\{\widetilde{\Psi}_n\}^{\mathrm{T}}\{\widetilde{a}\} - G\{\widetilde{\Psi}_\ell\}^{\mathrm{T}}\{\widetilde{c}\} \\ t_m - G\{\widetilde{\Psi}_n\}^{\mathrm{T}}\{\widetilde{b}\} - G\{\widetilde{\Psi}_m\}^{\mathrm{T}}\{\widetilde{c}\} \\ t_n - H\nu\{\widetilde{\Psi}_\ell\}^{\mathrm{T}}\{\widetilde{a}\} - H\nu\{\widetilde{\Psi}_m\}^{\mathrm{T}}\{\widetilde{b}\} - H(1-\nu)\{\widetilde{\Psi}_n\}^{\mathrm{T}}\{\widetilde{c}\} \end{Bmatrix}^{\langle k-1 \rangle}$$

Notice how the unknown vector, $\{u\}$, now contains the deflections in x-y-z space directly.

Finally, the remaining condition that can be applied to an elasticity boundary is the so-called mixed condition. In this case, deflections are prescribed in some directions, while tractions are implied in others. Fortunately, because deflections are almost always implied normal and tangential to the surface, the values prescribed are not u, v, and w, but instead the components a, b, and c. This is fortunate because the only step necessary to solve this case is to replace the rows of the prescribed direction in Equation 6.21 with the given value. For example, say that there is a traction of zero applied to both tangential directions ($t_\ell = t_m = 0$), and that there is a deflection of zero applied to the normal direction ($c = 0$). This is a very common situation as this combination represents a roller type condition, where the surface is free to slide tangentially, while restricted to deflecting normally. Thus, in this case Equation 6.21 would be expressed as

$$[A'] = \begin{bmatrix} G\psi_n & 0 & G\psi_\ell \\ 0 & G\psi_n & G\psi_m \\ 0 & 0 & 1 \end{bmatrix} [Q]^{\mathrm{T}} \qquad \{u\} = \begin{Bmatrix} u_c \\ v_c \\ w_c \end{Bmatrix}^{\langle k \rangle} \tag{6.22}$$

$$\{b\} = \begin{Bmatrix} -G\{\widetilde{\Psi}_n\}^{\mathrm{T}}\{\widetilde{a}\} - G\{\widetilde{\Psi}_\ell\}^{\mathrm{T}}\{\widetilde{c}\} \\ -G\{\widetilde{\Psi}_n\}^{\mathrm{T}}\{\widetilde{b}\} - G\{\widetilde{\Psi}_m\}^{\mathrm{T}}\{\widetilde{c}\} \\ 0 \end{Bmatrix}^{\langle k-1 \rangle}$$

Note that nothing has changed as far as the solution technique, only in the building of the A' matrix and right hand side vector.

CHAPTER 7
GENERALIZED AUTOMATIC POINT DISTRIBUTION

Thus far this thesis has been concerned primarily with the theoretical aspects of developing the meshless formulations for several governing equations. However, another key aspect of developing a working framework is the required preprocessing necessary to take an analytical representation of a solid (generally in the form of an IGES file) and discretize it in an fashion which is appropriate for meshless solutions. Although reading an IGES model file and interpreting the data is not necessarily a straightforward process, it does not involve any new work, and as such the techniques needed are thoroughly documented in current literature [38]. Therefore, this chapter will deal primarily in discretizing the primitive geometry into a form appropriate for meshless solutions.

7.1. SURFACE DISCRETIZATION

The first set of primitive entities that must be discretized are the boundary surfaces. In general, surface geometry that is generated from an IGES file is expressed as an underlying analytical surface and a bounding curve in three-dimensional space. Thus, a very simple case of a flat plane with circular bounds is shown below in Figure 7.1.

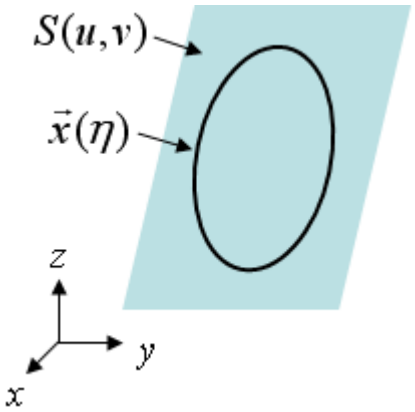

Figure 7.1. Representation of Planar Surface with Circular Bounds

In this figure, the underlying plane geometry is represented as an analytical parameterized function, $S(u,v)$, and the bounding curve in three-dimensional space is represented as a generalized function, $\boldsymbol{x}(\eta)$.

Therefore, before any discretization can take place, the generalized bounding curve must be discretized on the surface and transformed from η space into u-v space. To accomplish this task, a closest approach algorithm [39] is used to generate a set of simultaneous equations in u-v space which is solved using a simple multi-variable Newton-Raphson algorithm [40]. Having discretized the bounding curve and transforming the segments into u-v space, the surface representation can be visualized as

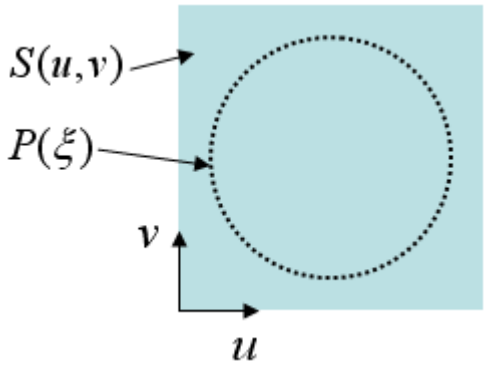

Figure 7.2. Representation of Surface Geometry and Discretized Bounding Curve

where the bounding curve has been replaced by a piecewise polynomial, $P(\xi)$, which is discretized in u-v space.

To apply a surface discretization to this geometry, the concept of two-dimensional quadtrees [41] are used to generate a preliminary "all inclusive" map of the surface. Then, the remaining unmapped area is closed using a two-dimensional triangulation utilizing Delauney triangulation. This mapping procedure will therefore generate a generalized, two-dimensional representation of the surface geometry which can be transformed into three-dimensional space (using the underlying surface equation) and discretized into the needed meshless nodes.

To implement these techniques, the first step in mapping this surface is to generate the quadtree representation. A quadtree is a tree data structure whose construction is based on the recursive decomposition of the Cartesian plane. Specifically, a quadtree representation of a closed region is defined by the recursive partitioning of the Cartesian plane into four equally sized quadrants which lie parallel to the coordinate axis. This recursion can be repeated until a model resolution criterion or partitioning limit is achieved. In the case of the mapped surface representation, only quadtrees that lie completely within the bounding curve are retained, as the remaining uncovered area will be mapped using a triangulation. Therefore, to illustrate the concept of quadtrees, the following figure demonstrate*s* how the level of refinement may be adapted to best represent the actual geometry.

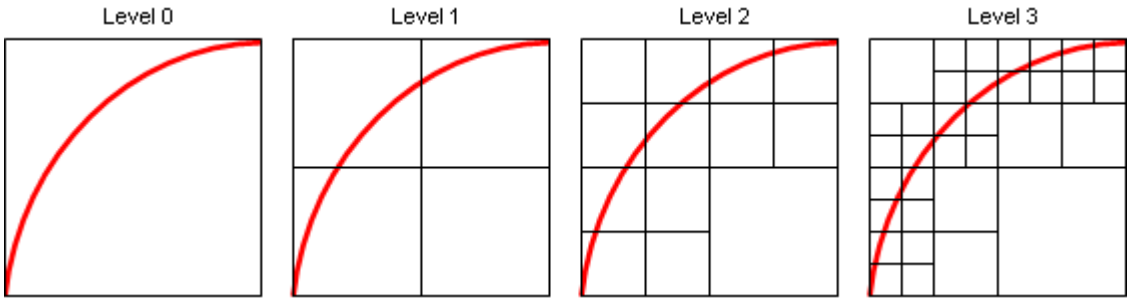

Figure 7.3. Example of Quadtree Discretization

In Figure 7.3 it is clear that a minimum number of discretizations need to be made in order to represent the given curved geometry. Thus, this technique can be applied to the planar surface with circular bounds represented in Figure 7.2 to generate an "all inclusive" quadtree map

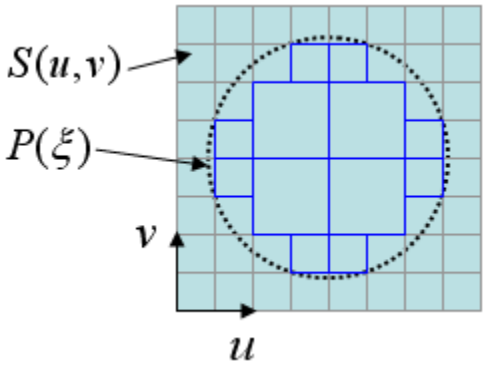

Figure 7.4. Quadtree Representation of Surface

In Figure 7.4 the quadtrees outlined in blue are "all inclusive," meaning that their bounds lie completely within the bounds of the surface. The remaining quadtrees are outlined in gray, meaning that they are either completely, or partially outside of the surface bounds. It is clear by examining this figure that although the quadtree discretization uses a minimal number of levels to represent the surface, it leaves rather large gaps between the bounds of the quadtree and the bounds of the surface. Therefore, to fill these gaps, a simple triangulation is performed between these two regions, shown in the figure below.

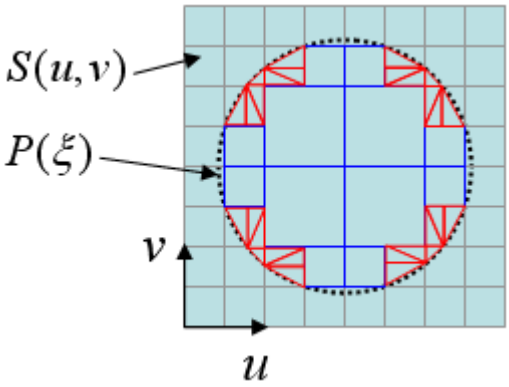

Figure 7.5. Quadtree and Triangulation Representation of Surface

As seen in Figure 7.5, the triangulation is shown in red, while the quadtrees are still shown in blue.

Using the combination of the quadtree and triangulated patches as a discretized map of the surface, nodes can now be placed on the surface at the center locations of each patch. For quadtrees, this location is simply the center of the individual quadtree area, for the triangles, barycentric coordinates can be used to determine the central location. Thus, shown below is an illustration of where a given node will be placed within each type of area

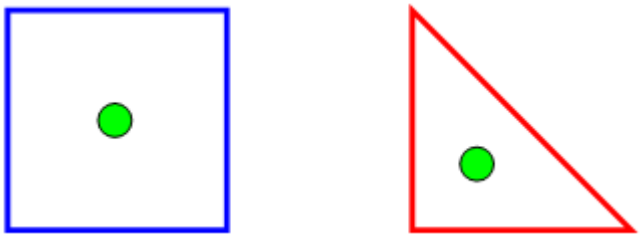

Figure 7.6. Placement of Nodes for Quadtree and Triangular Areas

Therefore, applying these placement rules to the surface map shown in Figure 7.5, the final surface discretization for this particular geometry would be

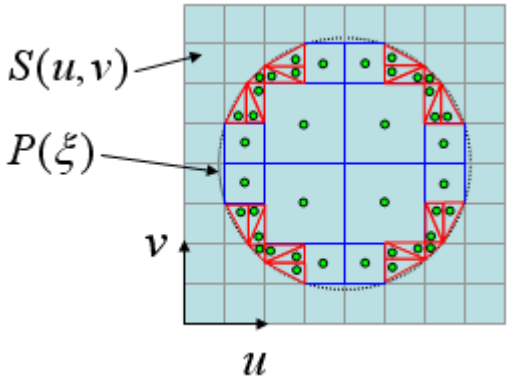

Figure 7.7. Nodal Distribution of Planar Surface with Circular Bounds

Of course, once the nodal distribution in Figure 7.7 has been generated in u-v space, using the underlying equation for the surface, these nodes can be transformed into x-y-z space and sent to the meshless framework.

It is important to note that although the meshless techniques do not require a structured mesh, they are still somewhat sensitive to the quality of the nodal distribution

(although not nearly as sensitive as more traditional meshed methods). Therefore, it is important when discretizing in this fashion that the size of the neighboring quadtrees are on the same order as the triangulated patches. Thus, although Figure 7.7 shows a fully discretized surface, more accurate solutions would be produced if the quadtrees were discretized a few more levels, to more appropriately correspond with the neighboring triangulation.

7.2. VOLUME DISCRETIZATION

Having discretized the boundary surfaces, the remaining discretization must take place within the model interior. A similar, recursive representation can be used, this time in three-dimensions instead of two. The three-dimensional counterpart to the quadtree is the octree, which represents a closed volume by defining a recursive partitioning of a Cartesian volume into eight equally sized quadrants which lie parallel to the coordinate axes. This recursion can once again be repeated until a model resolution criterion or partitioning limit is achieved. As with the surface mapping, only octrees that lie completely within the bounding volume are retained, however, unlike the surface mapping, no additional triangulation needs to be done in order to fill the volume. Since the interior nodes need not be associated with any particular control volume, the interior volume does not need to be completely accounted for using the octree discretization.

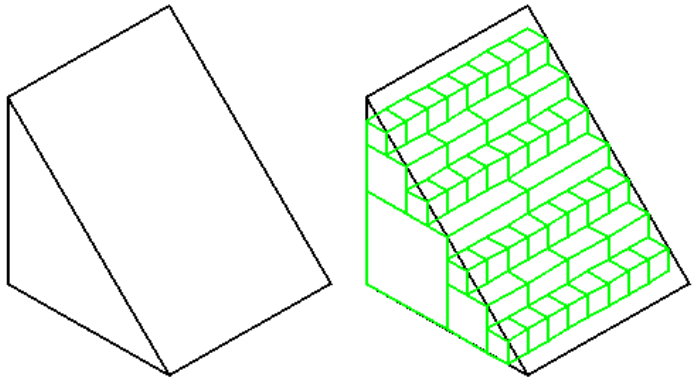

Figure 7.8. Octree Representation of Simple Three-Dimensional Volume

One of the major benefits of using an octree discretization for the interior is that by placing the interior nodes at the intersections of the octree volumes, a nearly Cartesian distribution can be generated for most of the interior. As already stated in Chapter 2, the meshless algorithms can be greatly simplified when a Cartesian point distribution is present, as the bulk of the interior will collapse to native finite differencing.

Also note that although the discretization shown in Figure 7.8 seems to suffer from the so-called staircase issue, recall that the green octree regions are only used for discretizing the interior volume. Since quadtree regions will be used to independently discretize the boundary, there are no problems associated with this sort of coarse correlation with the boundary surfaces.

Through the use of both quadtree and octree discretization techniques, any arbitrary model can be automatically discretized, and meshless points distributed, such that stable and efficient solutions may be found. These techniques illustrate the power and versatility of meshless methods to be applied to any generic model in a very generalized fashion.

CHAPTER 8
NUMERICAL VERIFICATION AND EXAMPLES

Having developed the concepts underlying the meshless framework, it is now appropriate to demonstrate the aforementioned techniques' ability to properly solve classic verification and example problems. A total of six problems are shown within this context, the first three dealing with steady-state heat conduction, the fourth dealing with ideal flow, the fifth dealing with heat advection-diffusion, and the last dealing with elasticity.

8.1. HEAT CONDUCTION VERIFICATION

The first verification that will be performed using the meshless framework outlined in this paper will be the case of a simple 3D domain with known temperature field applied to all boundaries. The problem description can be seen below in Figure 8.1.

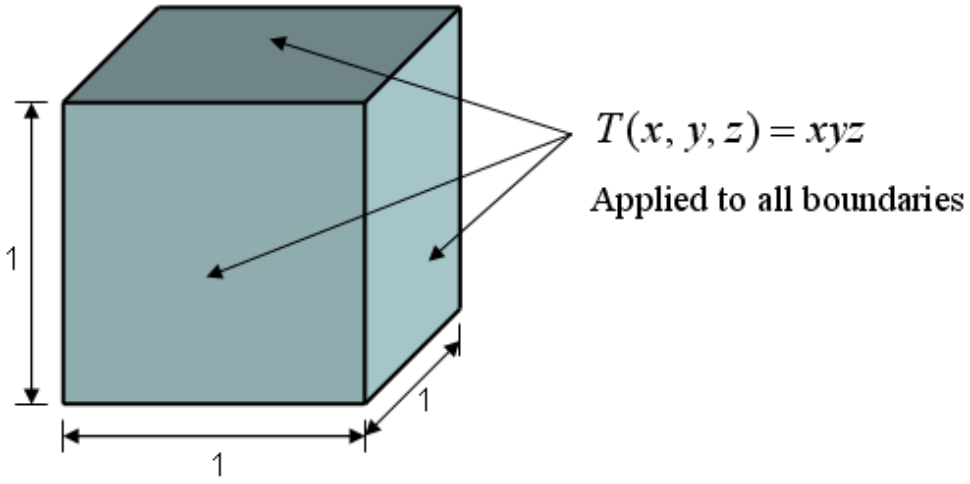

Figure 8.1. Heat Conduction Verification Problem Description

Note that for this case, the applied temperature conditions to the boundary are given as a function of geometry and can be expressed as:

$$T(x, y, z) = xyz \tag{8.1}$$

Therefore, the surface temperatures obtained after solving this problem with the indicated set of conditions using a grid spacing of $\Delta x = \Delta y = \Delta z = 0.05$ (9025 total solution nodes) can be seen in Figure 8.2.

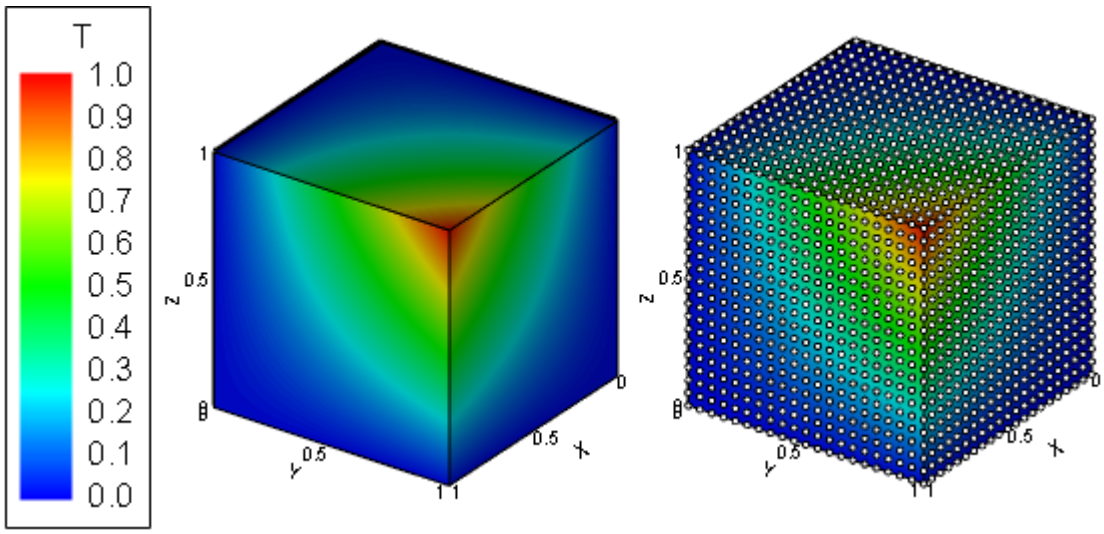

Figure 8.2. Heat Conduction Verification Surface Temperatures

However, since the temperature is prescribed everywhere on the boundary, this gives little indication as to the solution's accuracy. Therefore, several arbitrary lines of constant x and z were chosen and plotted against the known analytical results. These comparisons can be shown below in Figure 8.3.

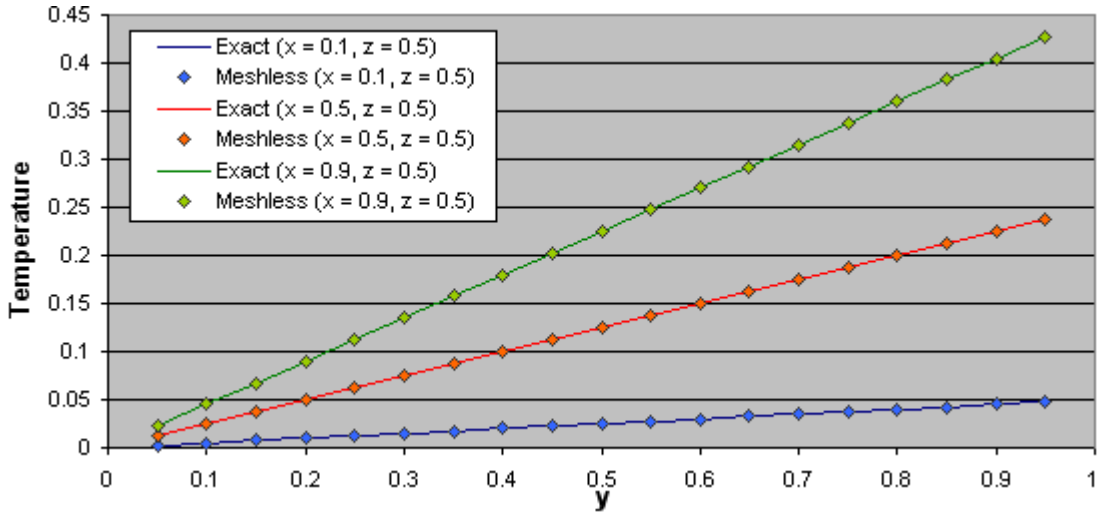

Figure 8.3. Interior Temperature Comparison

As shown, the results obtained using the meshless framework match very well with the exact solution imposed over the boundary.

Additionally, to further verify the solutions accuracy against the exact solution, the temperature along the diagonal, from point $(0,0,0)$ to point $(1,1,1)$, can be paramaterized with respect to a non-dimensional value, c and plotted for both solution methods.

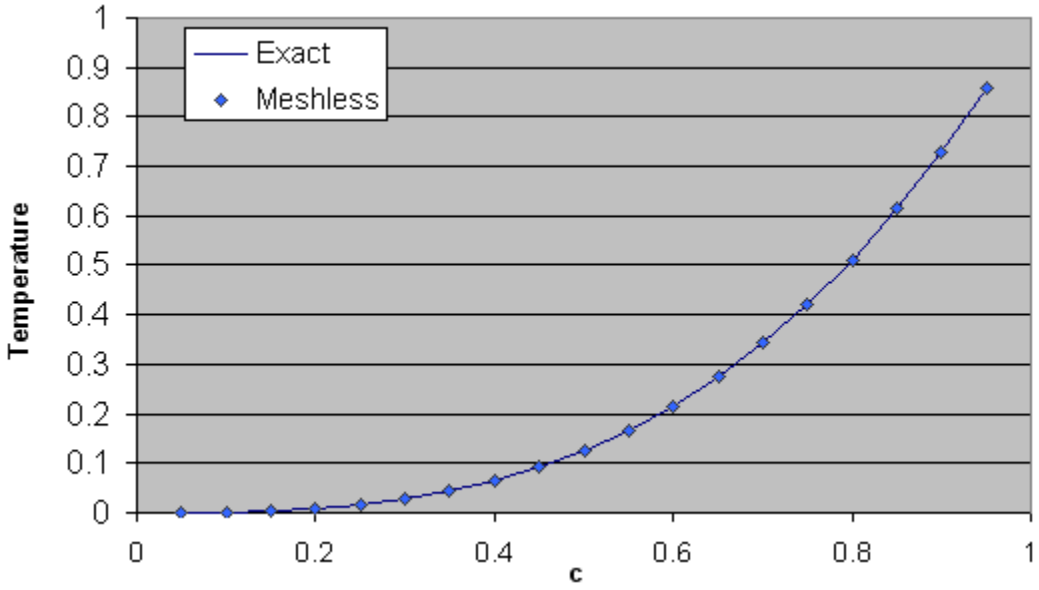

Figure 8.4. Diagonal Temperature Comparison

In Figure 8.4, it can once again be seen that the meshless solution is capable of representing the exact solution very closely.

To verify the overall quality of the solution throughout the domain, an L2 norm of the relative error was calculated by relating the difference between the meshless and analytical results as follows:

$$E = \frac{\sqrt{\frac{\sum_{i=1}^{N}(T_{\text{meshless}} - T_{\text{analytical}})^2}{N}}}{T_{\text{analytical}}(max) - T_{\text{analytical}}(min)} \tag{8.2}$$

Having completed this calculation for this particular problem and nodal distribution, the resulting error of the interior nodes was found to be 2.227×10^{-5} or 0.0022%, a nearly insignificant amount.

Obviously, for this simple case of imposed temperature, it would be expected that the solution is extremely accurate, especially considering the level of complexity of the imposed solution field. However, it is still important to verify that the techniques are able to accurately represented three-dimensional fields within a regular domain.

8.2. HEAT CONDUCTION WITH 1D SOLUTION

The second verification problem is the simple case of a one-dimensional temperature distribution within a solid. This is a classic example because it verifies that the solution is stable within the domain (by making sure that there aren't any fluctuations in solution based on position), as well as that applied flux conditions (insulated) are working properly on the surface. Thus, the problem description is shown below in Figure 8.5.

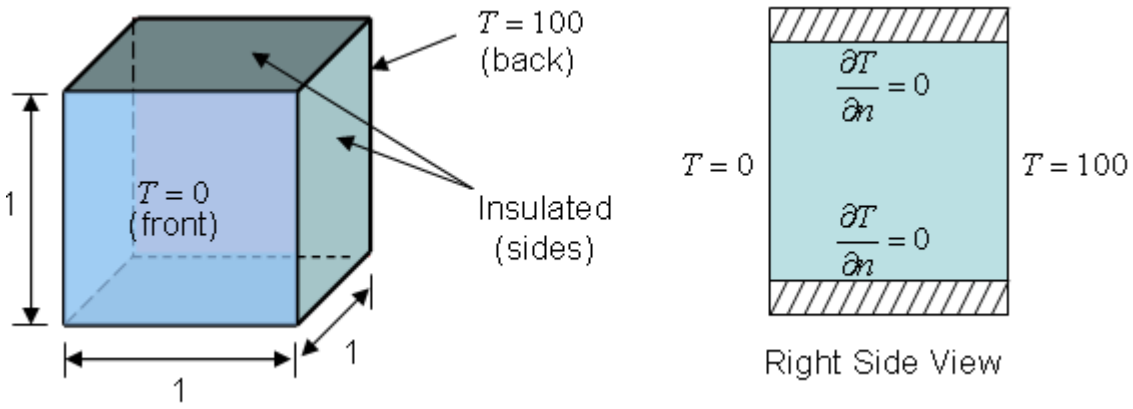

Figure 8.5. One-Dimensional Heat Conduction Example Problem Description

Note that for this case, the problem solution will be linear and one-dimensional between the front and back surfaces. Additionally, due to the insulated sides, the solution should be independent of position on any plane along the dependant dimension.

Thus, after solving this problem using the developed techniques, the surface solution produced is shown in Figure 8.6 (again, alongside the nodal distribution used).

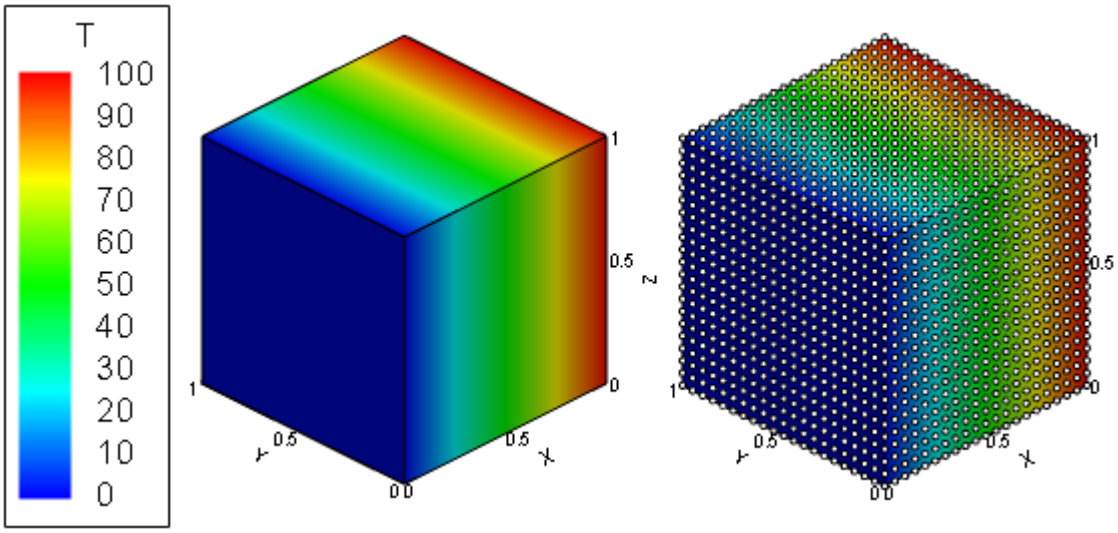

Figure 8.6. One-Dimensional Heat Conduction Example Surface Temperatures

For this problem a nodal spacing of $\Delta x = \Delta y = \Delta z = 0.05$ (9025 total solution nodes), was used

Additionally, to verify that this problem is correct regardless of position, several arbitrary vectors were chosen along the x direction and their solutions compared to the analytical results.

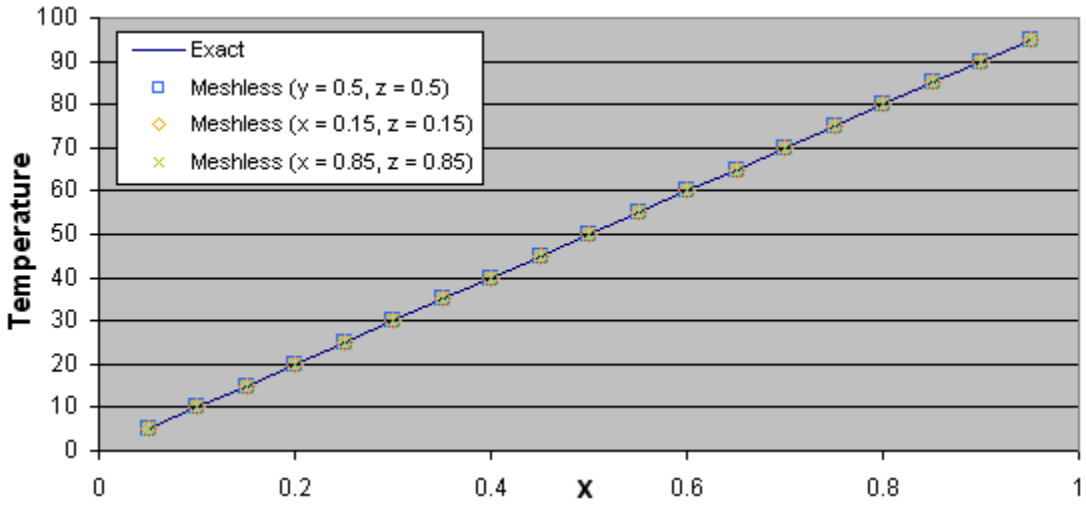

Figure 8.7. Interior Temperature Comparison

Once again, excellent correlation is obtained between the meshless and exact solutions for the interior nodes.

A similar error analysis was performed for this problem, and the final relative L2 norm of the error was computed as 2.833×10^{-4} or 0.0283%. Thus, the solution obtained using the meshless techniques once again did an excellent job of representing the exact field, even with applied flux conditions.

Additionally, in an attempt to identify the grid convergence qualities of the meshless framework developed, the initial verification problem (with applied temperature boundary conditions) and the current 1D solution problem was solved using a variety of nodal distributions and their relative error was plotted in Figure 8.8.

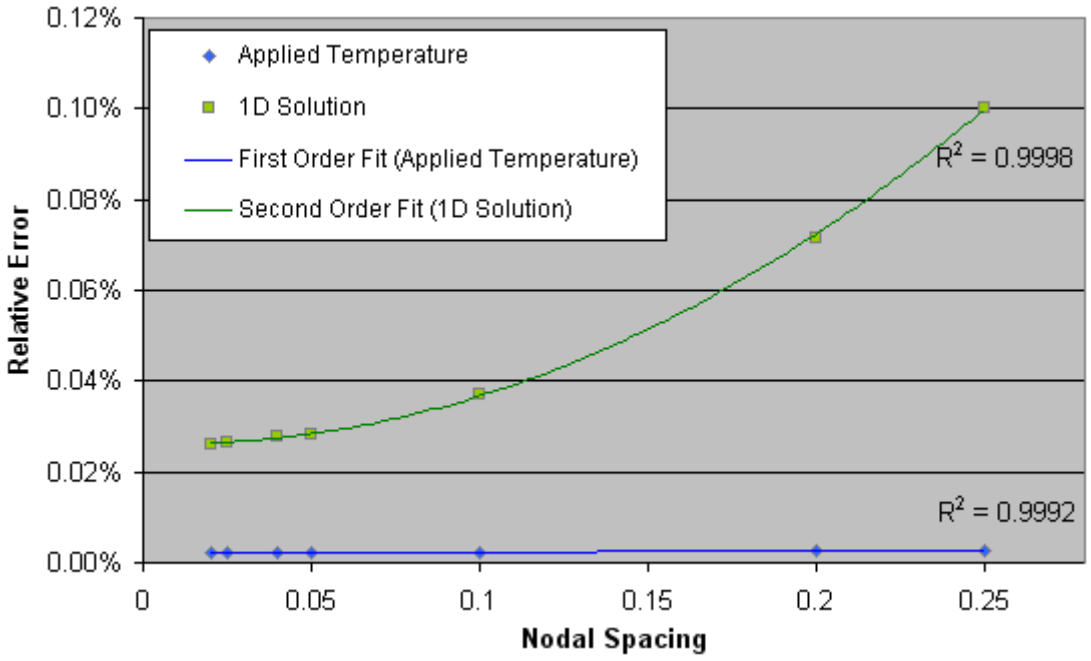

Figure 8.8. Grid Convergence Behavior for Heat Conduction

From this figure it can be seen that while the 1D solution converges on the order of $O(h^2)$, the verification problem actually converges on the order of $O(h)$. However, looking at the relative values of error, this anomaly can most likely be explained by the fact that the solution was found to such a high degree of accuracy that complete

convergence had been obtained. Also note that this range of nodal spacing represent the limits of the testing machine; in other words, at a nodal spacing greater than 0.25 there are not enough interior nodes to accurately generate interpolating functions, and at nodal spacing less than approximately 0.01, the memory limit of the workstations are exceeded.

8.3. HEAT CONDUCTION IN PIPE

The third example problem attempted in the area of heat conduction is the case of a radial heat pipe. In this case, a uniform temperature difference is applied to the inside and outside of a hollow, cylindrical pipe and the temperature distribution within the material is desired. Thus, the problem description is shown below in Figure 8.9.

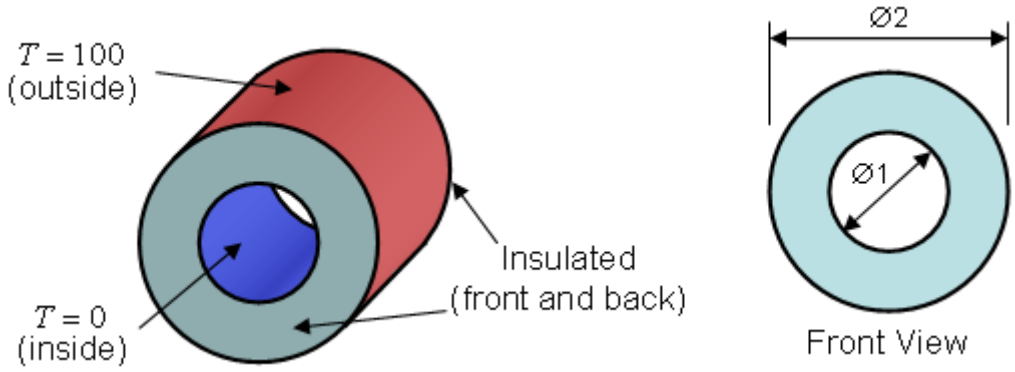

Figure 8.9. Heat Conduction in Pipe Example Problem Description

Note that for this problem the analytical solution can be found from classic texts [34] and is dependant only on the radial direction. Thus, the solution for this particular configuration is given as

$$T(r) = (T_2 - T_1)\frac{\ln\left(\frac{r}{r_1}\right)}{\ln\left(\frac{r_2}{r_1}\right)} + T_1 = \frac{100}{\ln(2)}\ln(2r) \tag{8.3}$$

Thus, by utilizing the quadtree and octree surface and volume discretization techniques, nodes were distributed on a 1 unit long model, at an average nodal distance

of $d = 0.05$, which resulted in a total of 20340 nodes for this problem. Thus, after the solution was obtained, a representative slice was taken at $z = 0.5$ and the temperature distribution (along side node locations for this slice) are plotted below.

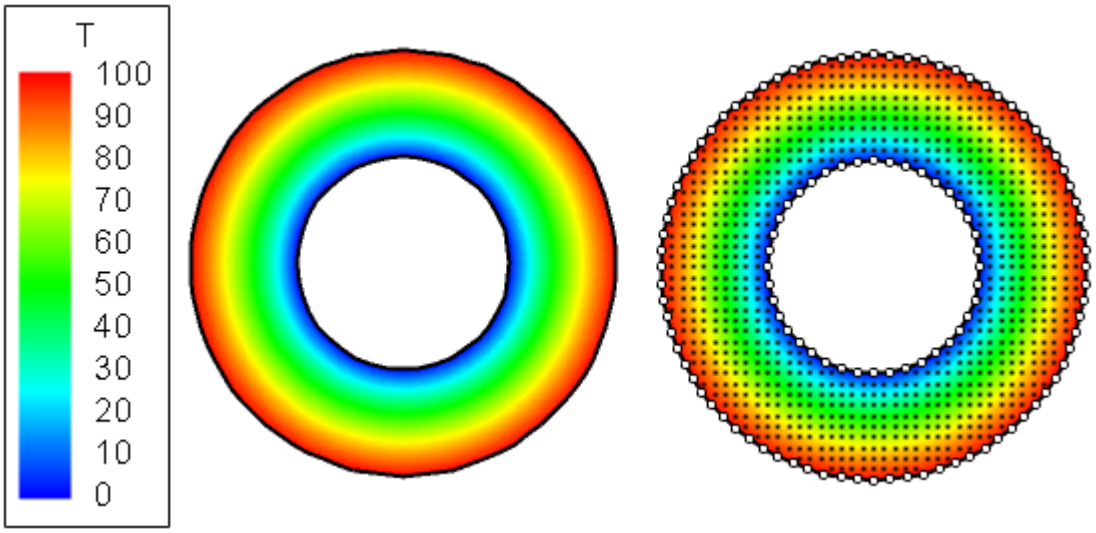

Figure 8.10. Heat Conduction in Pipe Example Problem Temperature Distribution

Note that although Figure 8.10 illustrates only a two dimensional cut of the model, the entire three dimensional model was solved using a similar nodal spacing projected into the z direction. Therefore, comparing the results obtained using the meshless techniques to that given by the exact solution resulted in an overall relative error of approximately 9.216×10^{-4} or 0.092%. It is a testament to the robustness of these techniques that even for a relatively coarse discretization such as this that the solution is obtained to such a high degree of accuracy.

Additionally, a plot of average radial temperature as a function of distance from the axis of rotation was performed, to further illustrate that the solution is consistent throughout the entire range of z.

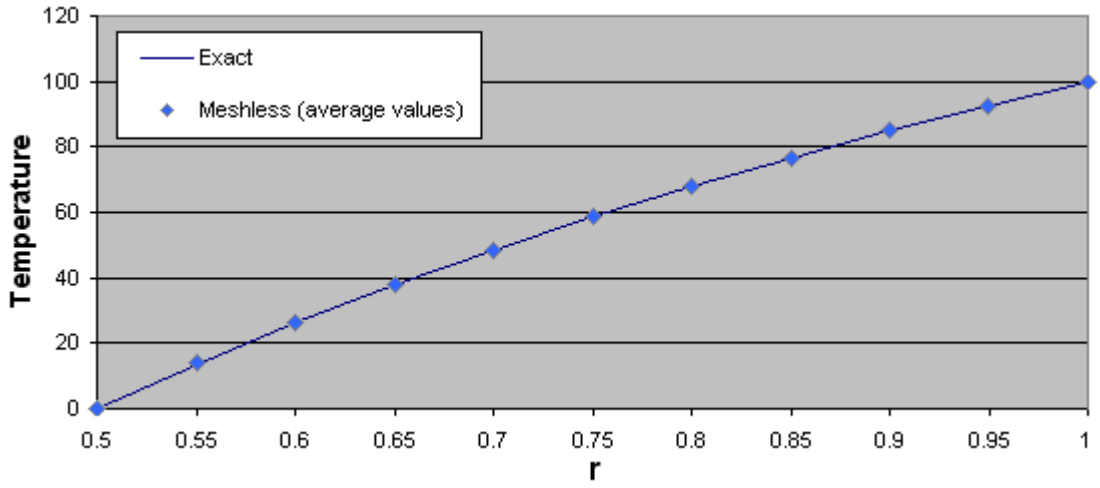

Figure 8.11. Average Radial Temperature Comparison

Figure 8.11 therefore represents the average of several meshless solution locations along the radial direction (4 per z value, total of 21 z locations, representing overall average of 84 radial lines) compared to the solution obtained from Equation 8.3.

8.4. IDEAL FLOW OVER A CYLINDER

Having demonstrated the ability of the meshless routines in accurately solving heat conduction problems, the next test case is that of ideal flow. For this case, an inlet flow is applied into a square channel with a cylindrical section removed. The problem description can be seen below in Figure 8.12.

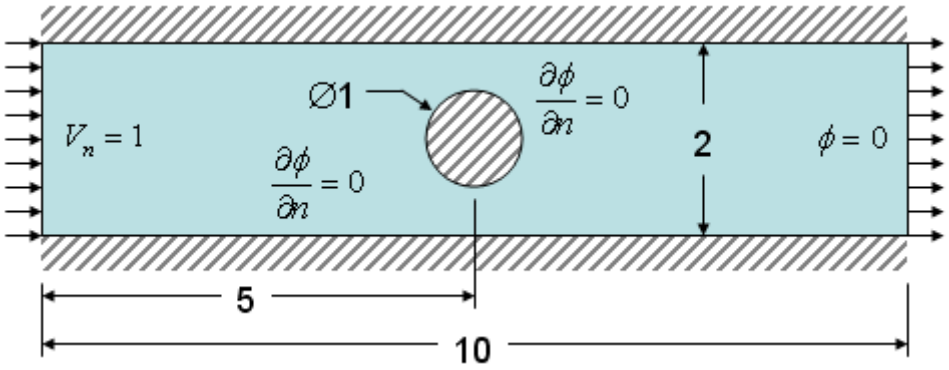

Figure 8.12. Ideal Flow Over Cylinder Problem Description

Note that although this figure represents the domain in two dimensions, this is only for clarity and the actual problem will be solved in three dimensional space (however, the problem is independent of the depth).

Therefore, having developed the model and problem description, the quadtree and octree techniques described in Chapter 7 were used to generate the nodal distribution, seen below in Figure 8.13.

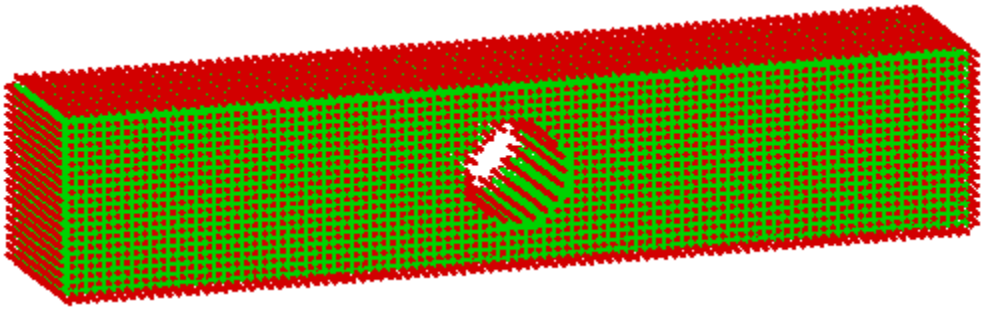

Figure 8.13. Nodal Distribution for Ideal Flow Over Cylinder

Note that for this particular distribution 42328 nodes were used (8432 boundary nodes and 33896 interior nodes) with an average nodal spacing of approximately 0.1.

Having automatically distributed nodes within the domain, and properly applying boundary conditions, the solution was obtained and a representative slice was plotted with respect to the length and height, seen below in Figure 8.14.

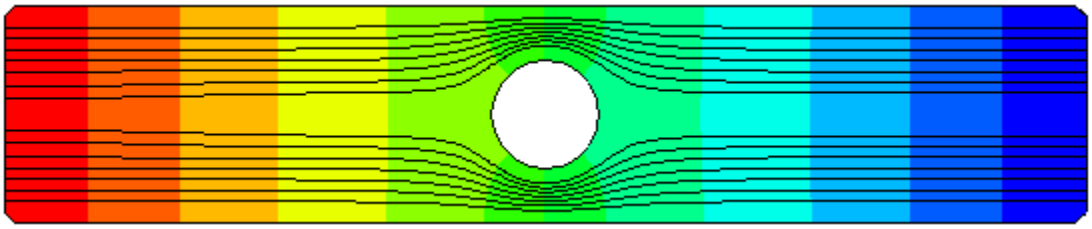

Figure 8.14. Streamlines and Velocity Potential Contours

Figure 8.14 shows two aspects of the solution process; the contours shown are those of the velocity potential, ϕ, while the streamlines were obtained using a post processing algorithm (working on the velocity potential). As shown, the streamlines

match the expected results and the velocity potential appears to be behaving correctly (max at inlet, min at outlet, perpendicular to walls).

Additionally, removing the velocity streamlines and adding the velocity vectors once again demonstrates excellent correlation to the expected result, shown below in Figure 8.15.

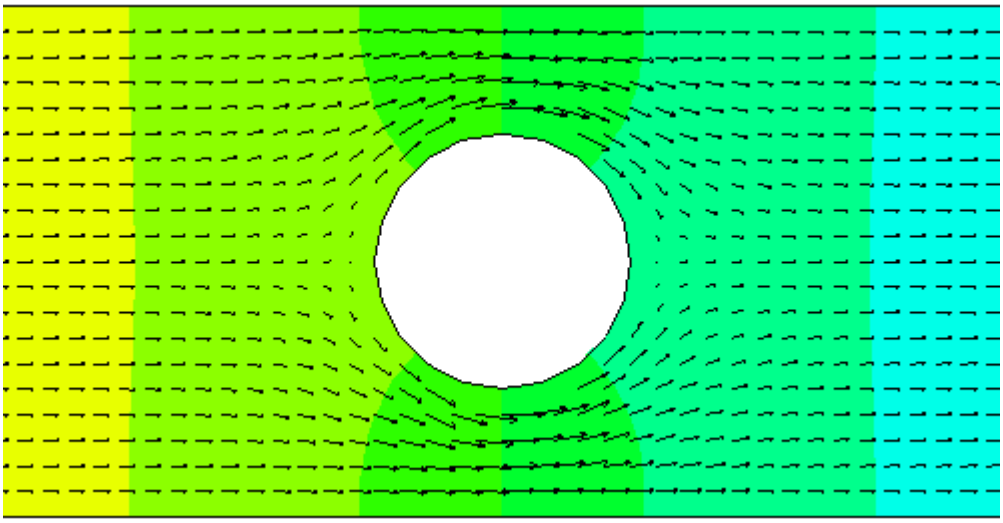

Figure 8.15. Velocity Potential Contours and Velocity Vectors

From this figure, the stagnation point at the front of the cylinder can be identified, as well as the "reattachment" point at the back side of the cylinder. Additionally, the velocity field is parallel far away from the cylinder, indicating that the velocity potential is being determined appropriately.

Having demonstrated the basic ability to represent the solution around a cylinder, both the channel height and width were increased to eliminate effects from the walls. At this point an analytical solution for the velocity potential may be found through superposition of elementary plane flows [42] as

$$\phi(r,\theta) = -Ur\left(1+\frac{a^2}{r^2}\right)\cos(\theta) \tag{8.4}$$

where U is the bulk stream velocity, r is the radial distance from the revolution axis of the cylinder, θ is the polar angle from the revolution axis of the cylinder (in plane), and a is the radius of the cylinder. Therefore, comparing the analytical results to those obtained along the midplane of the model at a sufficient distance from the walls results in the following plot of velocity potential versus height.

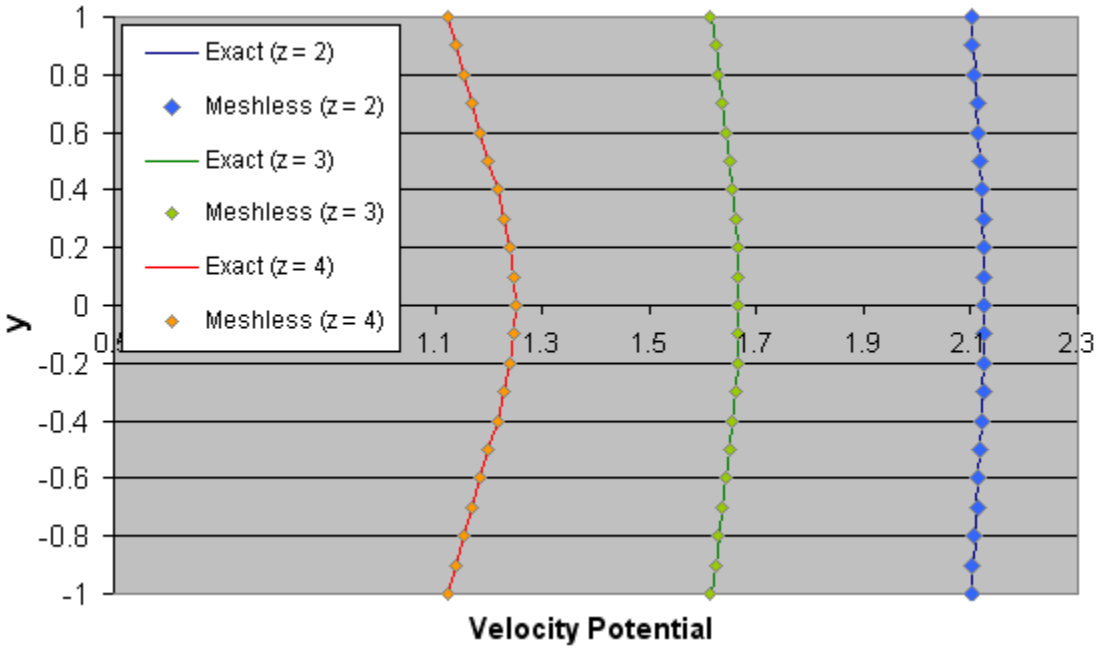

Figure 8.16. Comparison of Velocity Potential with respect to Vertical Position

Note that the meshless solutions were appropriately scaled by the outlet condition for appropriate comparison. Thus, after scaling, it is clear that the velocity potential is being accurately reproduced by the meshless techniques outlined in this framework.

This example is an important demonstration of the developed framework's ability to post process computation variables using the described meshless techniques. Oftentimes efficiency is lost when computing post-processing values at each iteration, and by being able to pre construct the derivative operators necessary for these operations, this loss can be minimized. Additionally, the geometry of this problem was much more complex than previous attempts and by using the automated techniques described in

Chapter 7 to build the nodal distributions, a very accurate solution was produced with minimal human involvement.

8.5. CONVECTIVE HEAT TRANSFER THROUGH A PIPE

Logically, the next step for verifying the ideal flow with heat advection-diffusion formulation is to solve a representative heat convection problem. Rather than solve for the temperature field along with the flow, to isolate the heat convection routines, a fully developed velocity profile will be imposed over the domain. Therefore, the problem to be solved is the case of convective heat transfer in a flow traveling through a cylindrical pipe, as described in Figure 8.17.

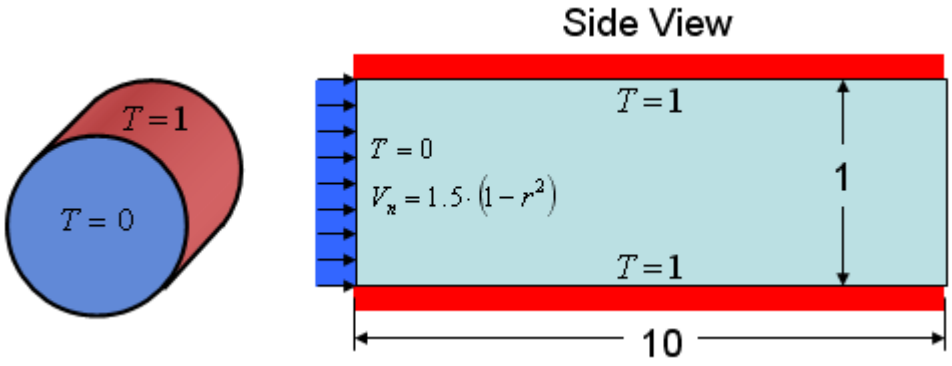

Figure 8.17. Convective Heat Transfer Problem Description

Notice that the velocity profile is given as a function of radial distance, where $r = 0$ corresponds to the centerline of the cylinder. Therefore, using the quadtree and octree techniques to distribute the nodes for this problem resulted in the following distribution.

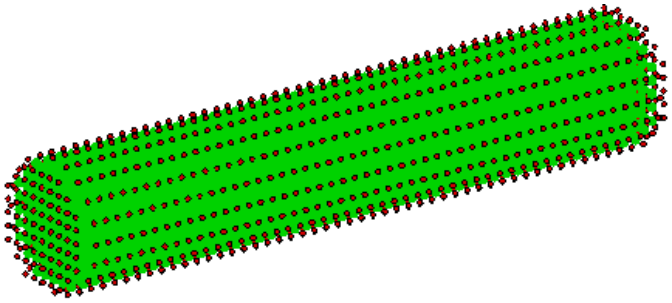

Figure 8.18. Nodal Distribution for Convective Heat Transfer

For this particular geometry, approximately 4500 nodes were used with an average nodal spacing of roughly 0.2.

Thus, for this particular fully developed velocity profile, a semi-analytic solution was obtained for a Prandtl number of 0.7 [43-44]. Thus, desiring to compare the meshless solution with this analytical solution, the problem was solved as described and the temperature profile is shown below.

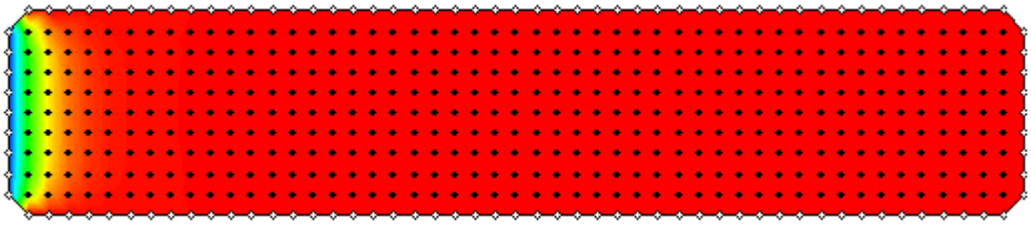

Figure 8.19. Temperature Profile Through Midplane

Here, it is clear that because the flow is moving at such a slow speed relative to the temperature difference, the entire flow reaches bulk temperature very quickly. However, since semi-analytic results exist for this problem, the temperature profile along the center line can be compared to those obtained using meshless.

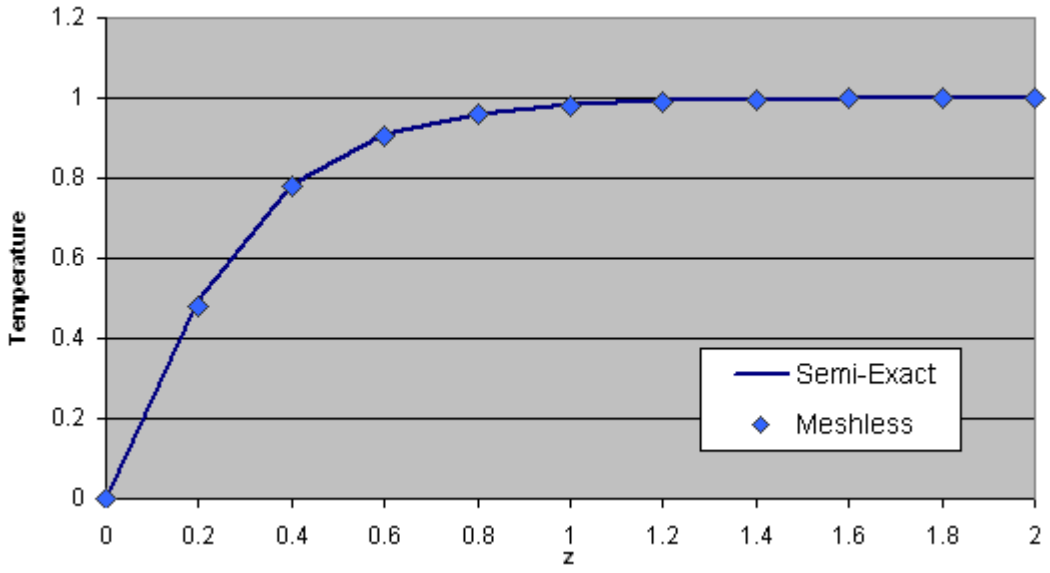

Figure 8.20. Temperature Comparison Through Centerline

Thus, there is an excellent correlation between the semi-analytic and meshless solutions with a root mean square error of approximately 0.0054.

To further demonstrate the solver's ability, the velocity profile was then increased to $V_n = 5 \cdot (1 - r^2)$, and the midplane temperature profile again plotted,

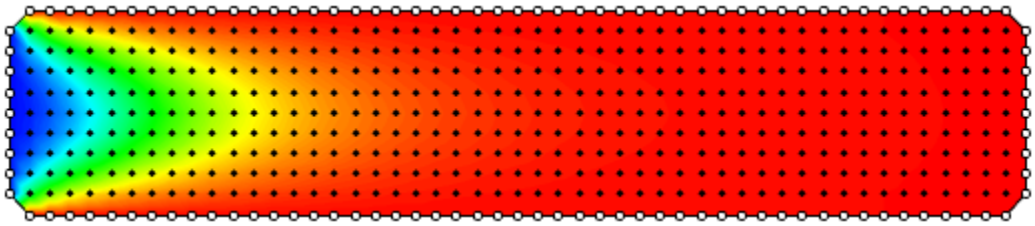

Figure 8.21. Temperature Profile Through Midplane (Increased Velocity)

Here it is clear that the temperature field is behaving appropriately and as the velocity magnitude is increased, it takes more distance for the flow temperature to reach the imposed value. Additionally, several slices were taken along the length of the cylinder (varying from $z = 0$ to approximately $z = 2$) and the temperature profiles were plotted.

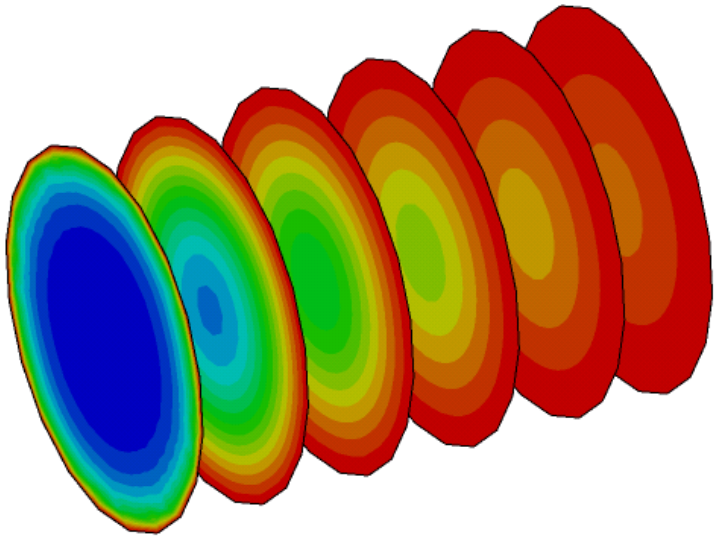

Figure 8.22. Representative Temperature Profile Slices Along Cylinder

Thus, the temperature is behaving well in the radial direction as the contours are symmetric about the axis of revolution for the cylinder.

8.6. STRESSES IN A CYLINDRICAL PRESSURE VESSEL

The final verification case will look at steady state elasticity and the problem of finding the radial and hoop stresses for a cylindrical pressure vessel, as shown below in Figure 8.23.

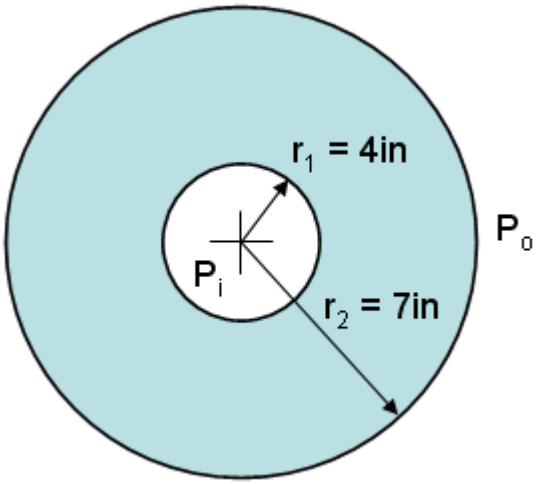

Figure 8.23. Cylindrical Pressure Vessel Description

Thus, for this problem the inner radius of the cylinder was set at 4in, the outer radius was set at 7in, and the inner and outer pressures, P_i and P_o, were set at 100psi and 0psi, respectively. As this is a very classic problem, analytical solutions for both radial and hoop stress are readily available [45] and are given by

$$\sigma_r(\rho) = \frac{r_1^2 P_i - r_2^2 P_o}{r_2^2 - r_1^2} - \frac{r_1^2 r_2^2 (P_i - P_o)}{(r_2^2 - r_1)\rho} \tag{8.5}$$

$$\sigma_h(\rho) = \frac{r_1^2 P_i - r_2^2 P_o}{r_2^2 - r_1^2} + \frac{r_1^2 r_2^2 (P_i - P_o)}{(r_2^2 - r_1)\rho}$$

where ρ is the point of interest in the radial direction.

Note that although at first glance this problem would appear to be ill-posed because of the pressure only conditions, because of the combination of roller and

symmetry conditions applied to the end caps this problem was able to be solved without arbitrarily fixing any nodes.

Therefore, having developed the problem model and setup, the meshless nodal discretization was creating using the described techniques and is shown in Figure 8.24 below.

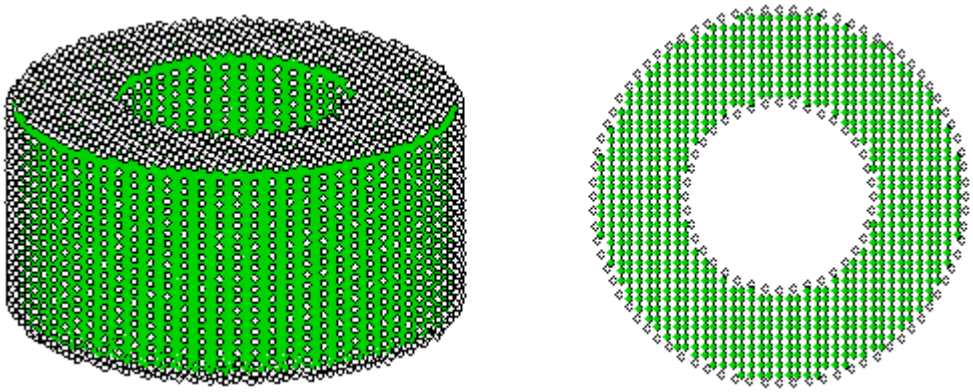

Figure 8.24. Meshless Nodal Distribution Pressure Vessel

Additionally, rather than simply solve this problem using meshless alone, a commercially availably Finite Element method solver was also used to develop a similarly dense mesh for this particular geometry. Notice that this was done not only to verify that both techniques obtained the correct results, but also to demonstrate how the automated meshless process can compare to a commercially available analysis package. Therefore, a sample layer of the finite element mesh used in solving this problem is shown below in Figure 8.25.

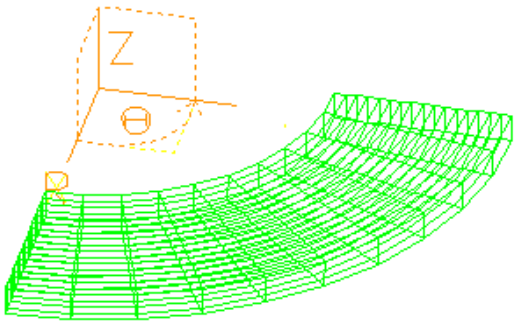

Figure 8.25. Finite Element Mesh for Pressure Vessel

To compare the results obtained using the meshless framework as well as the finite element package to the analytical solution, plots of the radial and hoop stress, as a function of radial distance, r, were created and can be seen below.

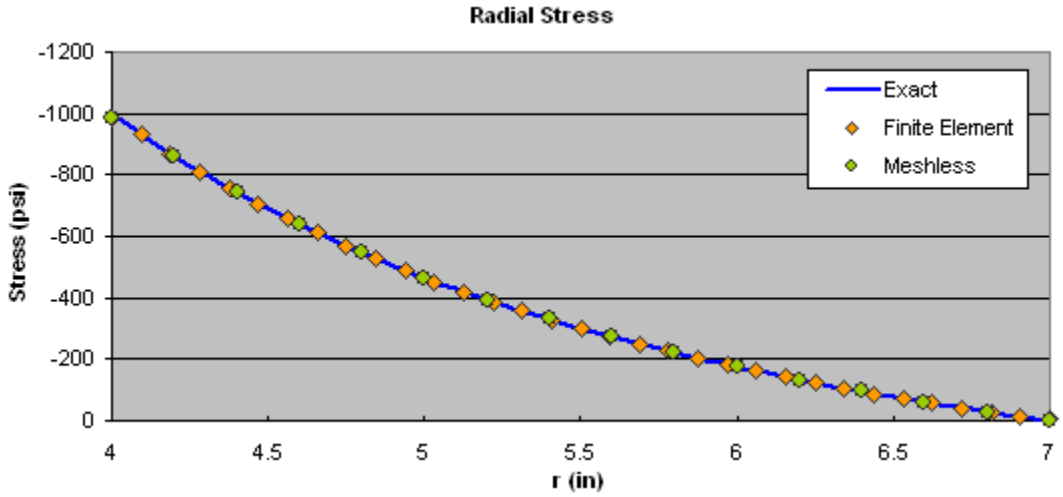

Figure 8.26. Radial Stress Comparison

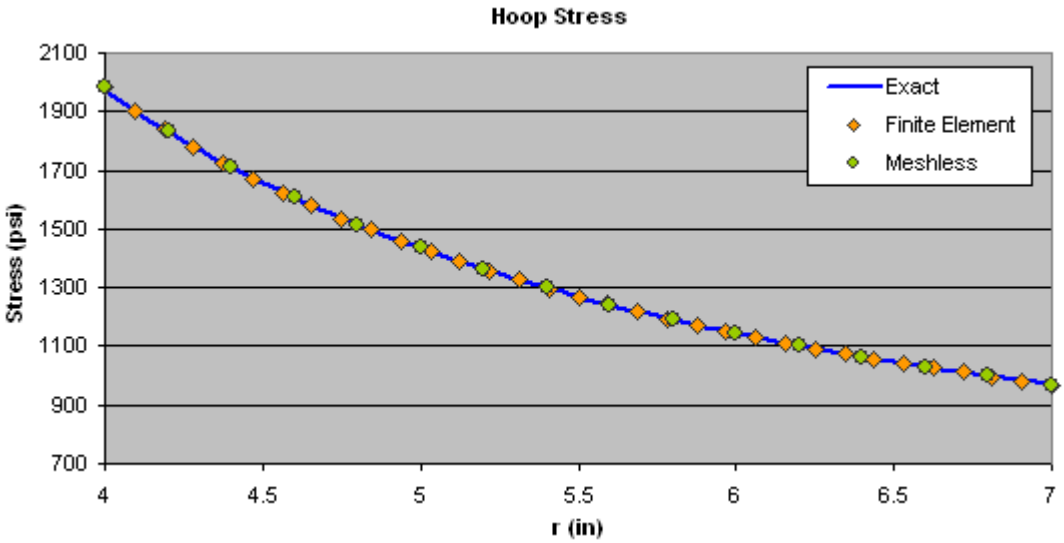

Figure 8.27. Hoop Stress Comparison

As shown, both techniques are very capable of accurately reproducing the analytical stress field within the cylinder. It is important to realize that for this problem, not only was the meshless routine comparable to the finite element package in terms of

speed and accuracy, but it also required no human interaction to develop the underlying nodal distribution. It is the elimination of this requirement which makes the aforementioned techniques so appealing.

CHAPTER 9
CONCLUSIONS AND FUTURE WORK

In conclusion, this thesis has presented a novel meshless method framework utilizing two different meshless collocation techniques. The first technique, Radial Basis Function collocation, has already been proven in many applications, but suffers from stability issues with non-radially-symmetric derivatives as well difficulties when dealing with upwinded and dynamically varying derivatives. In an attempt to alleviate these problems, this technique has been used to augment tradition finite element formulations, leading to Virtual RBF Finite Difference collocation. Together, these two techniques allow for solutions on unstructured nodal distributions requiring no connectivity mesh whatsoever. Additionally, in an attempt to develop a generalized, extensible meshless framework, a tailored formulation method was used to develop the governing equations into a standard form which can be solved in a interchangeable module environment. After developing the meshless framework formulation for several sample governing equations, the concepts of surface quadtrees and volume octrees were presented, which allow for generalized procedures for discretizing three-dimensional problem geometry. Finally, through several verification tests and example problems it was shown that the developed framework is capable of producing accurate results for realistic problems while minimizing the amount of required input by the user.

As far as future extension of the current work, the first step is understanding that, thus far, the primary focus of this work has been on three-dimensional formulations whose governing equations were of a linear form. Though this encompasses a wide range of physical phenomenon, a notable exception is the Navier-Stokes equations, which has non-linear, convective derivatives. As this is the logical progression from the case of ideal flow demonstrated in this thesis, a method of utilizing these techniques for nonlinear problems is desirable. As such, future work could be used to extend this

framework (specifically, the meshless framework formulation) to both linear, and non-linear governing equations.

Another current limitation in the framework is that there are no mechanisms for tackling transient problems. As solving problems accurately in time is often a requirement for a particular engineering analysis, this is definitely another important area which requires further study.

These techniques could also be further improved by introducing parallelization. As briefly mentioned in Chapter 3, there has been a large push for parallelizing routines in response to the increasing number of processor cores that are being packaged together in modern CPUs. As such, future effort could also be placed into parallelizing the developed meshless framework for optimal performance on these multi-core machines.

Lastly, with the increased emphasis on automatic model setup and improved computational time present with these techniques, new applications could be found in the areas of automatic part generation or shape optimization by applying existing optimization methods such as genetic algorithms. These applications would have been previously impossible due to the limitations present in a traditional meshed based method, however, with meshless techniques, they are now feasible.

LIST OF REFERENCES

[1] Belytscho, T., Lu, Y.Y., and Gu, L., "Element-free Galerkin methods, *Int. J. Num. Methods*," Vol. 37, 1994, pp. 229-256.

[2] Atluri, S.N. and Shen, S., *The Meshless Method*, Tech. Science Press, Forsyth, 2002.

[3] Atluri, S.N. and Zhu, T., "A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics," *Computational Mechanics*, Vol. 22, 1998, pp. 117-127.

[4] Atluri, S.N., The Meshless Method for Domain & BIE Discretizations, Forsyth, GA, Tech Science Press, 2004.

[5] Ofiate, E., Idelsohnl, S., Zienkiewicz, O.C., Taylor, R.L., and Sacco, C., ,"A stabilized finite point method for analysis of fluid mechanics problems," Computer Methods in Applied Mechanics and Engineering, Vol. 139, 1996, pp. 315-346

[6] Liu, G.R., *Mesh Free Methods: Moving Beyond the Finite Element Method.*, CRC Press, Boca Raton, 2002.

[7] Melenk, J.M. and Babuska, I., "The partition of unity finite element method: basic theory and application," *Comp. Meth. Appl. Mechanics and Eng.*, Vol. 139, 1996, pp. 289- 316.

[8] Gu, Y.T., and Liu, G.R., "Meshless techniques for convection dominated problems," Computational Mechanics, Vol. 38, 2006, pp. 171–182.

[9] Kansa, E.J., "Multiquadrics: a scattered data approximation scheme with applications to computational fluid dynamics I-surface approximations and partial derivative estimates," *Comp. Math. Appl.,* Vol. 19, 1990, pp. 127-145.

[10] 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," Comp. Math. Appl., Vol. 19, 1990, pp. 147-161.

[11] Kansa, E.J. and Hon, Y.C., "Circumventing the Ill-Conditioning Problem with Multiquadric Radial Basis Functions: Applications to Elliptic Partial Differential Equations," *Comp. Math. Appl.*, 2000, Vol. 39, pp. 123-137.

[12] Franke, R., "Scattered data interpolation: Test of some methods," *Math. Comput.*, Vol. 38, 1982, pp. 181-200.

[13] 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 Equation," *Engineering Analysis with Boundary Elements*, 2002, Vol. 26, pp. 133-156.

[14] Cheng, A.H.-D., Golberg, M.A., Kansa, E.J., Zammito, G., "Exponential Convergence and H-c Multiquadric Collocation Method for Partial Differential Equations," *Numerical Methods in P.D.E.*, Vol. 19, No. 5, 2003, pp. 571-594.

[15] Gottlieb, D. and Orzag, S.A., *Numerical Analysis of Spectral Methods: theory and applications*, Society for Industrial and Applied Mathematics, Bristol, England, 1977.

[16] Maday, Y. and Quateroni, A., "Spectral and Pseudo-Spectral Approximations of the Navier-Stokes Equations," *SIAM J. Numerical Analysis*, 1982, Vol. 19, No. 4, pp. 761-780.

[17] Patera, A., "A Spectral Element Method of Fluid Dynamics: laminar flow in a channel expansion," *J. of Computational Physics*, 1984, Vol. 54, pp. 468-488.

[18] Macaraeg, M. and Street, C.L., "Improvement in Spectral Collocation Discretization Through a Multiple Domain Technique," *Applied Numerical Mathematics*, 1986, Vol. 2, pp. 95-108.

[19] 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," *J. of Computational Physics*, 1989, Vol. 83, pp. 260-291.

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

[21] Powell, M.J.D., "The Theory of Radial Basis Function Approximation," in *Advances in Numerical Analysis, Vol. II,* Light, W., ed., Oxford Science Publications, Oxford, 1992. pp. 143-167.

[22] Buhmann, M.D., *Radial Basis Functions: Theory and Implementation*, Cambridge University Press, Cambridge, 2003.

[23] Dyn, N., Levin, D., and Rippa, S., "Numerical Procedures for Surface Fitting of Scattered Data by Radial Basis Functions," *SIAM J. of Sci. Stat. Computing*, 1986, Vol. 7, No. 2, pp. 639-659.

[24] Divo, E., Kassab, A.J., Mitteff, E., and Quintana, L., "A Parallel Domain Decomposition Technique for Meshless Methods Applications to Large-Scale Heat Transfer Problems." ASME Paper HT-FED2004-56004.

[25] Divo, E. and Kassab, A.J., "A Meshless Method for Conjugate Heat Transfer," Engineering Analysis, Vol. 29, No. 2, 2005, pp. 136-149.

[26] Divo, E. and Kassab, A.J., "An Efficient Localized RBF Meshless Method Applied to Fluid Flow and Conjugate Heat Transfer," ASME Paper IMECE2005-82150.

[27] Divo, E., Kassab, A.J., "Boundary Elements and RBF Meshless Methods Modeling in Thermo-Fluids." 2005 NASA Thermal & Fluids Workshop, 11 Aug. 2005

[28] Divo, E., Kassab, A.J., "Modeling of Convective and Conjugate Heat Transfer by a Third Order Localized RBF Meshless Collocation Method." Proc. of NHT-05 Eurotherm82, Bialecki, R.A. and Nowak, A.J. (eds.), 2005, pp. 357-366.

[29] Gerace, S.A., A Meshless Method Approach for Solving Coupled Thermoelasticity Problems. Diss. Univ. of Central Florida, 2006.

[30] Gerace, S.A., Divo, E., Kassab, A.J., "A Localized Radial-Basis Function Meshless Method Approach to Axisymmetric Thermoelasticity", AIAA/ASME Heat Transfer Summer Conference, June 2006.

[31] Erhart, K.J., Kassab, A.J., Divo, E., "An Inverse Meshless Technique For The Determination Of Non-Linear Heat Generation Rates In Living Tissue", EUROTHERM 82, Numerical Heat Transfer 2005, September 13-16, 2005, Gliwice-Cracow, Poland, Eds.: A. Nowak, R.A. Bialecki.

[32] Divo, E, Kassab, A.J., "An Efficient Localized Radial Basis Function Meshless Method for Fluid Flow and Conjugate Heat Transfer", ASME Journal of Heat Transfer, Volume 129, February 2007, pp. 179-183. DOI: 10.1115/1.2402181

[33] Hardy, R.L., Multiquadric Equations of Topography and Other Irregular Surfaces, Journal of Geophysical Research, Vol. 176, pp. 1905-1915.

[34] Tannehill, J.C., Anderson, D.S., Pletcher, R.H., Computational Fluid Mechanics and Heat Transfer, 2th ed., Washington D.C., Taylor & Francis Publishing., 1997, pp. 46-58.

[35] Fung, Y.C., A First Course in Continuum Mechanics for Physical and Biological Engineers and Scientists, 3rd ed. Englewood Cliffs, New Jersey: Prentice Hall, 1994, pp. 209-230.

[36] Dally, J.W., Riley, W.F., Experimental Stress Analysis, 4th ed., Knoxville, TN, College House Enterprises, LLC., 2005, pp. 3-53.

[37] Timoshenko, S P., and Goodier, J N., Theory of Elasticity, 3rd ed. New York, New York, McGraw-Hill, Inc., 1951, pp. 380-484.

[38] Smith, B., Rinaudot, G.R., Reed, K.A., Wright, T., Initial Graphics Exchange Specification (IGES), Version 4.0, Warrendale, PA, Society of Automotive Engineers, Inc., 1988.

[39] Schneider, P.J., Eberly, D.H., Geometric Tools for Computer Graphics, New York, Morgan Kaufmann Publishers, Inc., 2003, pp. 365-663.

[40] Agoston, M.K., Computer Graphics and Geometry Modeling: Implementation and Algorithms, London, Springer-Verlag, 2005, pp. 371-695.

[41] Mitteff, E.A., Automated Adaptive Data Center Generation for Meshless Methods. Diss. Univ. of Central Florida, 2006.

[42] Fox, R.W., McDonald, A.T., Introduction to Fluid Mechanics, 5th ed., New York, John Wiley & Sons, Inc., 1999, pp. 266-284.

[43] Fletcher, C.A.J., Computational Techniques for Fluid Dynamics, Volume 1, Berlin, Springer-Verlag, 1991, pp. 293-326.

[44] Brown, G.M., "Heat or mass transfer in a fluid in laminar flow in a circular or flat conduit", AIChE Journal, Volume 6, Issue 2, June 1960, pp. 179-183.

[45] Shigley, J.E., Mischke, C.R., Budynas, R.G., Mechanical Engineering Design, 7th ed., New York, McGraw-Hill Publishing, 2004, pp. 149-151.