THE DEVELOPMENT OF AN ALL QUADRILATERAL BOUNDARY CONFORMING GRID GENERATOR FOR HIGH ORDER FINITE ELEMENT METHODS

By

Mary Barker

Lafayette K. Taylor
Professor of Computational Engineering
(Chair)

Steve L. Karman jr.
Professor of Computational Engineering
(Committee Member)

James C. Newman III
Professor of Computational Engineering
(Committee Member)

Kidambi Sreenivas
Professor of Computational Engineering
(Committee Member)

THE DEVELOPMENT OF AN ALL QUADRILATERAL BOUNDARY CONFORMING GRID GENERATOR FOR HIGH ORDER FINITE ELEMENT METHODS

By

Mary Barker

A Thesis Submitted to the Faculty of the University of Tennessee
at Chattanooga in Partial Fulfillment of the Requirements
of the Degree of Master of Science: Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

December 2015

ABSTRACT

A grid generator is developed that produces all quadrilateral meshes. The scheme is automated to work for arbitrary choice of geometry. In addition, a Non-Uniform Rational B-Spline curve fitter is implemented to replicate the curvature of the geometries. The grid elements on the boundaries conform to the curved structure to support high order accuracy for a finite element scheme.

Various geometries are used to test the robustness and generality of the meshing algorithm. The initial problems that were encountered are discussed and the solutions explained. The speed of the algorithm is discussed together with the effect of grid and geometry size on runtime.

A finite element solver is used to validate the grids. The order of accuracy of the scheme is demonstrated for quadrilateral grids and increased order is compared with a refined grid study.

DEDICATION

This work is dedicated to my Parents, William and Prudence Barker, and to my sisters Heidi and Heather, who made it fun.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Professor Lafayette Taylor for his patience, encouragement and unending help during this process. Also I am deeply grateful to Arash Ghasemi for many stimulating discussions and helpful ideas. Without their help this work would not have succeeded.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The purpose of this research is to create an algorithm that will generate two dimensional unstructured grids composed entirely of quadrilateral elements. The grid conforms to curved boundaries to increase accuracy for high-order spacial discretizations. A Non-Uniform Rational B-Spline (NURBS) formulation for approximating boundaries is implemented and curved elements are stored in order to evaluate the edges of quadrilaterals that fall on a boundary.

Higher order Finite Element solution algorithms promise increased accuracy and profoundly higher convergence rates when compared with more traditional finite volume formulations. However, the convergence rate of these algorithms is highly dependent upon element type, particularly the preponderance of quadrilaterals in two dimensions and hexahedrons in three dimensions rather than triangles and tetrahedra, pyramids and prisms respectively. These meshes are considerably more difficult to create for complex geometries than mixed element meshes, and more prone to skewed or otherwise badly shaped elements. The manual construction of quadrilateral grids is the best method for ensuring quality control of meshes. However, it is also time consuming and often requires painstaking effort to create. Thus a quadrilateral mesh generator that can account for highly curved and irregular geometries is required.

The objectives for the research are speed, robustness, computational efficiency and generality. Speed is an important objective as the reason for automating the grid generation process is to

ease the solution process. Manual mesh generation is the best way of creating quality meshes and therefore a more time efficient method is desired. As the mesh will be used in conjunction with a finite element solver, speed and memory consumption are two important considerations. The more global importance of generality is also considered. The grid generator must be capable of creating valid meshes for any geometry. In addition, the quality of those meshes must be such as to validate the automated process. The best way to ensure a quality mesh is to create it manually, but a mesh generator capable of creating quality meshes without input is highly desirable.

CHAPTER 2

PREVIOUS EFFORTS AT QUADRILATERAL MESH GENERATION

The attempt to automate all-quadrilateral mesh generation is not a new undertaking. There is a wide array of methods that have been implemented for structured and unstructured meshes.

Several methods for generating quadrilateral grids, as described in [1], [2] and [3] begin by first initializing the domain with a triangular mesh and then combining triangles by some method into quadrilaterals. This technique is very robust and creates meshes where every element has nonzero area, however the resulting meshes are predominately made up of very poor quality elements that may require a great deal of smoothing in order to achieve optimum results. Also, this scheme is not highly efficient apart from the need for a smoothing algorithm since it requires the construction of two meshes in order to achieve one.

Another common grid generating procedure is first to subdivide the given domain into simple subsets or patches that are connected sets, and then to generate quadrilateral grids for each patch using a simple interpolation, or in the case of [4], a more advanced shape-decomposition technique. Although this method generally ensures good quality meshes, the complexity of an algorithm capable of implementing such a scheme is beyond the scope of this project. The stipulation that the borders between patches have the same number of nodes together with the variety of ways of dividing the overall domain into patches require specific geometric user input.

Many methods that have been developed for quadrilateral mesh generation involve the creation of an overlay Cartesian mesh that spans the entire domain. The main difference between these methods is the algorithm used to introduce and join the geometry into the mesh. Of these, there are hierarchical cases where a quadtree method is used to decompose the domain into quadrilaterals and sub-quadrilaterals based upon boundary curvature [5]. Druyor implements a hierarchichal method where the overlay mesh is successively refined as the elements are closer to the boundaries [6]. The mesh is then 'grown' from the boundaries to create viscous layering and joined to the overlay mesh using a triangulating scheme at the interface. The scheme is capable of being generalized, but misses the goal of producing an all-quadilateral mesh. A procedure described by Cui is to create an overlay mesh and then remove the elements that overlap the boundaries and connect the boundaries to the overlay mesh by directly matching the remaining grid points on the edges of the domain to points distributed along the boundaries [7]. This scheme requires user input based on the specific geometry in order to first determine the number of points to be added to each boundary and then to connect the points together. Therefore this scheme lacks the criterion of generality.

The grid generating algorithm developed for this project is one which involves the construction of an overlay grid from which to generate a mesh with curved boundary conforming elements. The details of the structure will be discussed further in the work.

CHAPTER 3

GRID GENERATION

In this chapter the structure of the mesh generator is outlined and some test geometries are shown to illustrate the utility of the scheme. Section 3.1 describes the means used to replicate boundary curvature using a Non-Uniform Rational B-Spline generating scheme. The curved edges are then used to create the remaining mesh. This is seen in Section 3.2 where the body of the algorithm is discussed. The problems that arise with some different geometries are described in Section 3.3 together with examples of successful applications of the mesh generator. Finally, the speed of the algorithm is considered in Section 3.4.

3.1 Boundary Curves

The replication of curved geometries by means of constructing interpolating curves to fit a collection of points is an important facet of the grid generator as the curved edges allow for high order point placement with boundary conforming accuracy when using a finite element solver. The point placement and effect on the overall solution is discussed further in later sections.

There are many schemes for interpolating a set of points using some mathematical function. Some of the most commonly used are splines such as B-Splines and T-Splines. Non-Uniform Rational B-Splines are a generalization of B-Splines. They are parametric functions that describe complicated shapes smoothly. One of the benefits of NURBS interpolation as opposed to other

splines is that NURBS are locally evaluated while retaining global smoothness. This means that NURBS curves can interpolate a high number of points without introducing oscillatory error to which other splines are prone.

3.1.1 Mathematical Formulation for NURBS Curve

The complete description of a NURBS curve has five main components: a knot vector (k), B-Spline basis functions($N_i(t)$), order (p), control points(P_i) and a weight vector(w). The general form of a NURBS curve of order p defined by n control points is

$$C(t) = \frac{\sum_{i=0}^{n} w_i N_i^p(t) P_i}{\sum_{i=0}^{n} w_i N_i^p(t)} \tag{3.1}$$

For this research, a constant value of $p = 3$ was used.

The knot vector is a vector of some m values $k_{i=1,\ldots,m}$ where $m = n + p + 1$ and $k_i \leq k_{i+1}$. A NURBS curve is called Nonuniform when the values in the knot vector are unequally spaced. The number of times a value k is repeated within the knot vector is called the multiplicity of k. The multiplicity of a value affects the continuity of the curve at that point by decreasing the differentiability of the function by the same amount. At a value with multiplicity m, the curve is only $p - m$ continuous. Thus the values k_i cannot be repeated more than p times.

B-Spline basis functions are parametric functions defined in terms of the knot vector k. For a value $t \in [0, 1]$ these functions are defined recursively.

$$N_i^p(t) = \frac{t - k_i}{k_{i+p} - k_i} N_i^{p-1}(t) + \frac{k_{i+p+1} - t}{k_{i+p+1} - k_{i+1}} N_{i+1}^{p-1}(t) \tag{3.2}$$

Where

$$N_i^0(t) = \begin{cases} 1, & \text{if} \quad k_i \le u \le k_{i+1} \\ 0, & \text{else} \end{cases} \tag{3.3}$$

Control points lie outside of the curve and control the shape of the curve by 'pulling' the curve as shown in Figure 3.1. Shifting the control point can cause the curve to stretch. Although represented as a single value P_i in Equation 3.1, each control point has x, y and z components. Thus Equation 3.1 has in fact three components where in two dimensions the third equation has all $P_{i,z} = 0$.

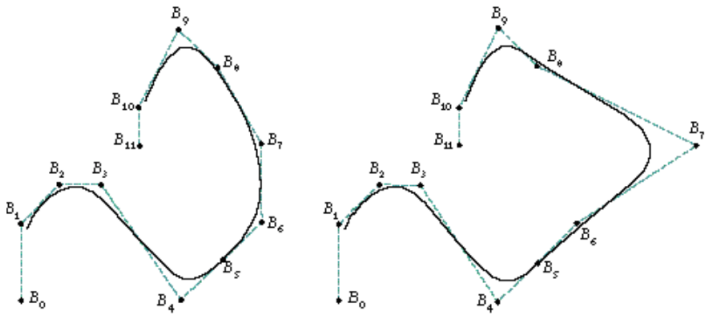


Figure 3.1 Effect of control points in NURBS formulation

The weight vector w controls the influence of each control point with respect to how far the curve can be pulled in its direction. For each control point P_i there is a corresponding weight w_i. The general rules for the strength of each control point P_i are as follows.

1. If $w_i = 0$, then P_i has no effect whatsoever

2. if w_i increases, the effect of P_i increases

3. If w_i decreases, the effect of P_i decreases

The major difference between interpolating functions and B-Splines in general is that interpolating functions are defined in terms of a set of points through which the function passes. B-Splines however, are defined in terms of a series of control points which govern the shape of the function without necessarily satisfying the function itself at that point. This idea is illustrated in Figure 3.2 where interpolated points and control points are shown.

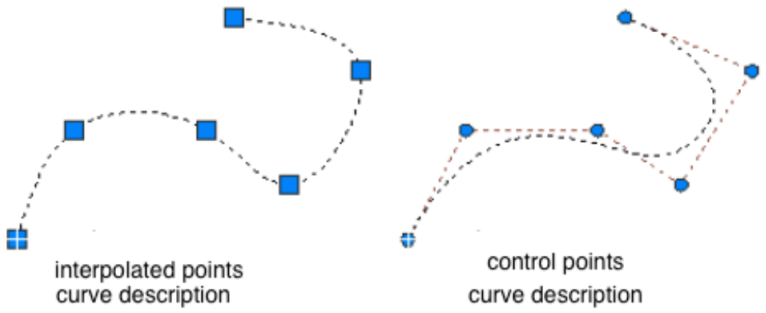


Figure 3.2 Interpolating points and control points

Before attempting to find weights and control points to fit a set of points, it is necessary to introduce a parametrization $t \in [0, 1]$ of the points to be fitted. For any given curve this was accomplished by defining t_i for np points as

$$t_i = \frac{1}{L} \sum_{j=2}^{i} \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2 + (z_j - z_{j-1})^2} \tag{3.4}$$

where L is the sum of the lengths of the segments in the curve.

The knot vector is a vector with $np + p$ values in the range $[0, 1]$. The requirements for the knot vector are that the first p elements are equal to 0 and the last p elements are equal to 1. Thus in order to create a knot vector, any set of $np + p$ elements ranging in ascending order from 0 to 1 with the specified end conditions will suffice. Piegl and Tiller recommend averaging the parametrized values, as the resulting knots more closely reflect the spacing of the original points [8]. This scheme generates values for the knot vector for arbitrary order p using the formula shown in Equation 3.5.

$$k_i = \begin{cases} 0, & 0 \leq i \leq p \\ \frac{1}{p} \sum_{j=i-p}^{i-1} t_j, & p+1 \leq i \leq np \\ 1 & np+1 \leq i \leq np+p \end{cases} \tag{3.5}$$

With the knot vector defined, the basis functions N are defined as well, and setting $w_i = 1$ for all i, $i = 1, ..., np$, a curve of arbitrary order p can be generated for a set of points $X_i = (x_i, y_i), i = 1, ..., np$ by solving the system of linear equations

$$X_i = \sum_{j=0}^{np} N_j^p(t_i) P_i \tag{3.6}$$

for the control points P_i. Although there is not a unique NURBS curve for each collection of points, this scheme produces results that reflect the spacing of the initial points. In addition, the method used to compute NURBS is robust and accurate.

3.1.2 Quality Metric for Curve Fitting

The quality of the NURBS curve fitting scheme can be evaluated by defining a metric for the closeness of the fit. For this study the error is computed as the maximum distance between the original set of points to be interpolated and the corresponding points evaluated using the NURBS formulation. All of the geometries used in this section are shown in Appendix B

The first question to be considered is whether the number of grid points has effect on the curve fit. For this case a simple circle of unit radius is described with a varying number of points, and then interpolated with a NURBS curve. The errors are shown in Table 3.1.

Table 3.1 Errors for NURBS Curve Fitting Various Numbers of Points

Number of Boundary Points	Maximum Error
10	2.48253E-016
50	3.51083E-016
100	4.47545E-016
500	4.00297E-016
1000	4.96507E-016

In every case the maximum error is on the order of machine zero. This is a good indication. Although the error does increase slightly from the first case to the last, the change is so negligible that it can be ignored. The maximum value of the error fluctuates slightly, however it does not change one order of magnitude with respect to the number of points fitted.

Another consideration is the effect of curvature upon the error for curve fitting. To test this a series of geometries were fitted that have varying levels of curvature and combinations of curvature, straight lines and acute angles. The first geometry is a straight line, the second, a square. The third and fourth geometries are an upper semicircle and a full circle respectively. The fifth geometry is a NACA 0012 airfoil, and the sixth a 30P30N airfoil with multiple sections. The errors for all of these cases are shown in Table 3.2.

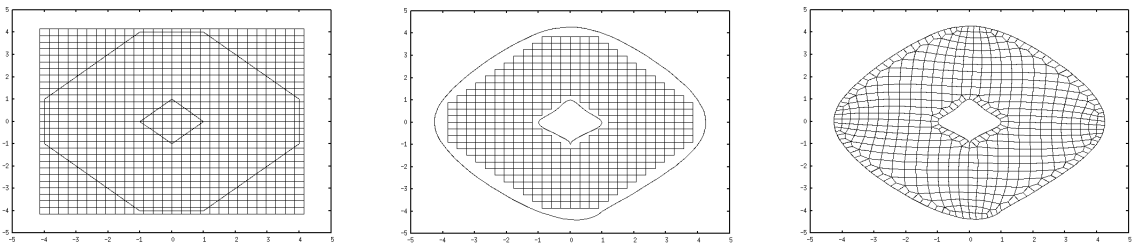
Table 3.2 Error for NURBS Curve Fitting Various Geometries

Geometry	Maximum Error
Straight line	5.55112E-017
Square	4.57757E-016
Semicircle	2.00148E-016
Circle	2.48253E-016
NACA 0012 Airfoil	3.33139E-016
30P30N Airfoil front tip	1.98603E-015
30P30N Airfoil main body	3.33356E-016
30P30N Airfoil rear flap	4.44956E-016

Although there is some slight variation with respect to geometry curvature, the errors are all on the order of machine zero and therefore acceptable. The NURBS curve fitter is demonstrated to be robust and accurate for a range of geometries.

3.2 Grid Generator

The general structure of the grid generator introduced in this research is one that uses an overlay Cartesian mesh as an initial stencil. An overarching grid that spans the entire domain is initialized and the quadrilaterals that intersect the boundaries together with any nodes exterior to a boundary are deleted. Once the boundary quadrilaterals are deleted, the remaining grid is attached to the boundary curves to create an unstructured quadrilateral mesh.



(a) Overlay Mesh Initialization (b) External Quads Deletion (c) Boundary Connection

Figure 3.3 Grid generation process

An outline of the algorithm that follows the procedure in Figure 3.3 is shown in Figure 3.4. The scheme is similar to that described by Cui [7], but has been automated and generalized to work for any geometry. Although of simple design, there are several difficulties that must be met in order to achieve a robust scheme. These are explored later and illustrated with several test cases that describe how the issues are resolved.

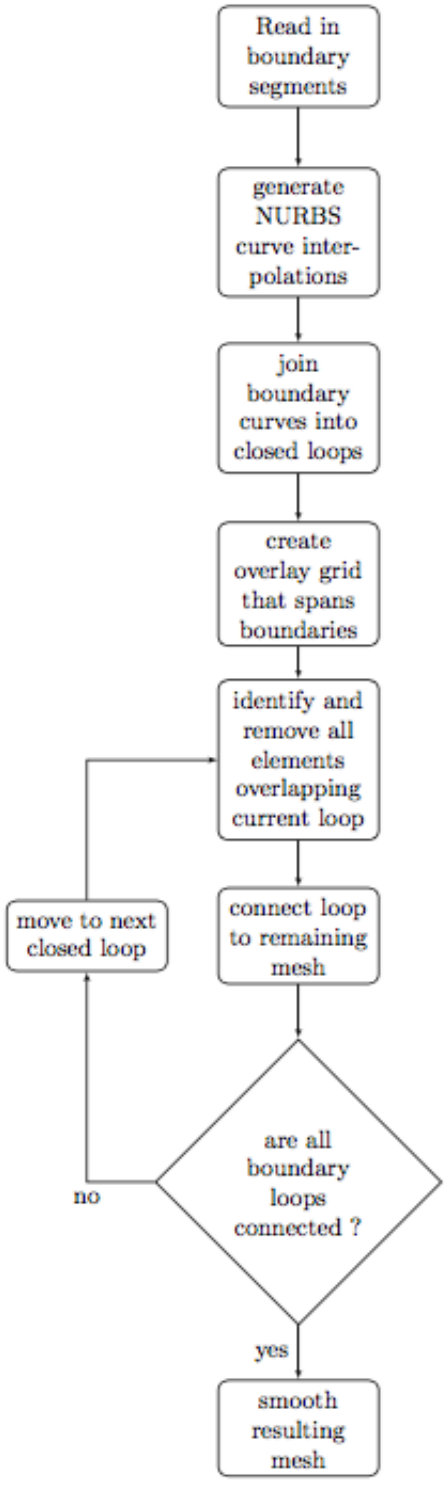


Figure 3.4 Flowchart of grid generating algorithm

There are two constraints upon the geometries that can be used for this project. The first is that the domain must be connected, that is, that any two points in the domain can be connected by a finite number of consecutive line segments that are fully within the domain. The second is that the segments must be oriented such that the outer curve is counterclockwise and interior boundaries are clockwise. These two conditions are necessary in order to determine whether a given point is inside the domain or not.

One of the fundamental issues to be dealt with when identifying the quadrilaterals in the overlay grid which are outside of or spanning a boundary segment is the problem of how to accurately identify the intersection of two line segments. The analytic solution to this problem is a simple procedure outlined in most elementary algebra textbooks. However, when computing the solution using a finite precision computer, many issues arise.

First is the problem of machine accuracy. The slope of a given segment is computed using finite precision mathematics and therefore subject to machine error. Therefore in comparing two quantities a tolerance must be added. This is most problematic when dealing with lines that are close to parallel or parallel and non-overlapping and therefore the values to be compared are close to the value of the tolerance itself.

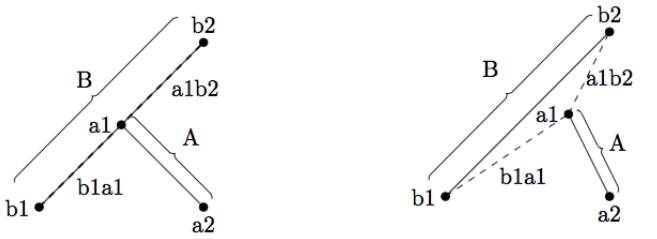
The second problem is that of extent. Due to the fact that the segments are finite in length, it is possible to have two non-intersecting segments such that the lines running through them do intersect at a point beyond the extent of one or more of the segments. In addition, machine accuracy often induces errors when two segments intersect at an endpoint. For a more detailed examination of the issue of segment intersection in finite precision arithmetic together with a rigorous study of the errors arising in such tests see [9].

For this discussion, let two segments A and B have endpoints $a1$, $a2$ and $b1$, $b2$ respectively where $a1 = (a1_x, a1_y)$, $a2 = (a2_x, a2_y)$, $b1 = (b1_x, b1_y)$, $b2 = (b2_x, b2_y)$. The segments are parametrized $A = a1(1-t) + a2t$ and $B = b1(1-t) + b2t$ for $t \in (0, 1)$. Three intersection types were tested in this project, each with multiple possible variations. The first is when two segments cross at some point within the length of both segments. This can be tested by solving the equation $a1(1-t_1) + a2t_1 = b1(1-t_2) + b2t_2$, which is equivalent to

$$\begin{bmatrix} a2_x - a1_x & b1_x - b2_x \\ a2_y - a1_y & b1_y - b2_y \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} b1_x - a1_x \\ b1_y - a1_y \end{bmatrix} \tag{3.7}$$

If both of the variables $t_1, t_2 \in [0, 1]$, then the lines through A and B intersect at a point coincident with both segments.

As there can arise accuracy issues regarding the intersection of two points, it is necessary to add a second condition to the first test for intersection. The second type of intersection is when two segments cross at a point that is the endpoint of one or both of the segments, or when either of the computed values t_1, t_2 are close to 1 or 0. This can be tested by connecting one endpoint from a segment with the two endpoints of the other segment and computing the cross product of the resulting lines. If they are parallel, then the endpoint lies on the second line. For example, if $a1$ is incident with the line B, then the cross product of the segments $b1a1$ and $a1b2$ will be equal to 0. Examples of endpoint intersection and possible case where the endpoint is close but non overlapping are shown in Figure 3.5.



(a) Intersecting endpoint (b) Non-intersecting endpoint

Figure 3.5 Line intersection test

The third type of intersection is when the segments are parallel and overlapping. This case fails the intersection test since the left hand side matrix in Equation 3.7 is noninvertible. However, a positive result for two or more endpoint intersection tests will accurately reflect that there are parallel lines.

With these tests, it is clear that some false positive cases can occur. For example, if a given edge in a quadrilateral is intersected twice by a boundary curve then both nodes will be counted as exterior to the domain when they are actually interior. This is demonstrated in Figure 3.6. Note that the arrows denote the normal vector to the boundary

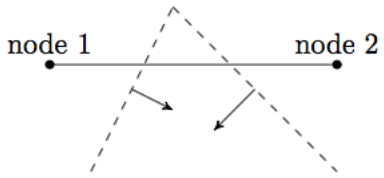


Figure 3.6 Line segment with multiple intersections

In order to delete only nodes that are exterior to the domain, the overlay mesh must be constructed in such a way that no edge has more than one intersection with a boundary. In addition to the problems with edge intersections, the complexities of the range of possibilities for different boundary intersections increase when considering quadrilaterals as the basic pieces for deleting. This is a necessary consideration, since after deleting everything exterior to the domain, the remaining structure must be rebuilt into a mesh made up of quadrilaterals rather than a collection of edges or points. One issue is that unconnected nodes can be left when the edges connecting a given node are all deleted. An example is shown in Figure 3.7 where only nodes 3, 4, 5 and 7 are left after deleting external nodes leaving 3 completely disconnected.

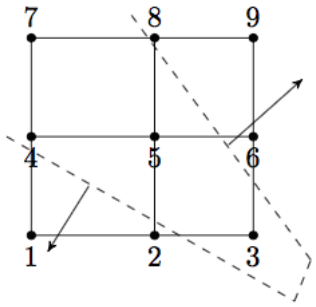


Figure 3.7 Deletion process resulting in unconnected interior node

The solution to this difficulty is to ensure that the overlay mesh fulfills the criterion that no quadrilateral has more than 3 sides intersecting with boundaries. This issue arises often with certain airfoil geometries where the boundaries are narrow relative to the mesh spacing and oriented diagonally to the principle axes of the overlay quadrilaterals. Figure 3.8 shows how this can lead to an unacceptable number of intersections for a given quadrilateral.

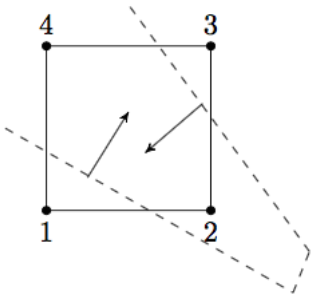


Figure 3.8 Example of a problematic boundary

In order to generate acceptable overlay meshes, a preprocessor was implemented. Since the overlay mesh is structured, the preprocessor subdivides the rows and columns of the mesh based on the number of intersections detected. A rudimentary step toward guessing appropriate placement for the initial quadrilaterals is also in place. This algorithm aligns the grid lines passing through or near a given geometry to conform to the general curvature of that structure. The way in which the overlay grid reflects some geometrical features is illustrated in Figures 3.9 and 3.10. The first example, shown in Figure 3.9 is that of extent. The grid lines that are close to the boundaries mimic some of the curvature in order to minimize intersections. The second example is that of orientation. The longest direction spanned by each boundary curve is considered and if the general orientation is at an angle to the coordinate axes, the overlay lines within range of the geometry are angled to reflect the angle of the curve.

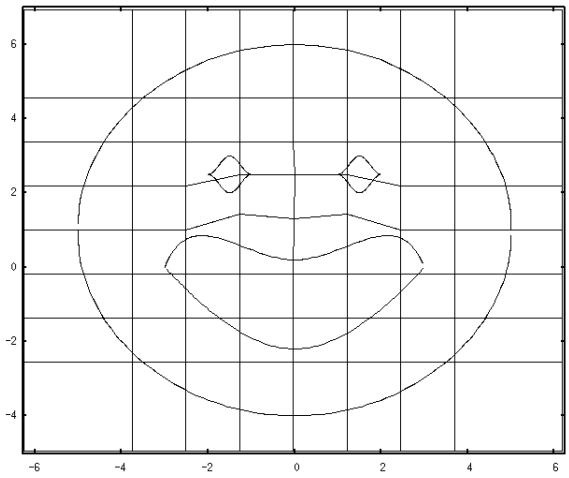


Figure 3.9 Conforming grid lines: geometry extent

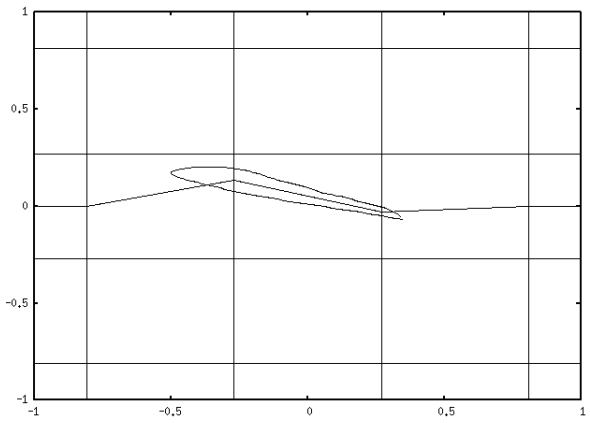
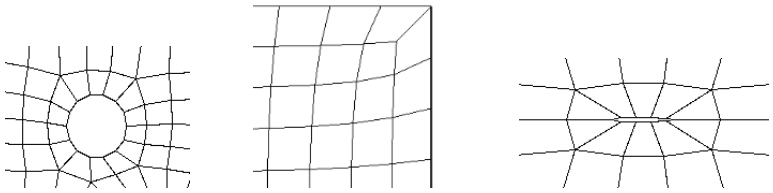


Figure 3.10 Conforming grid lines: geometry orientation

Once the appropriate quadrilaterals in the overlay mesh have been identified and removed, the next step is to connect the remaining mesh to the boundary curve. In the deleting process, the nodes on the edges of the mesh have been associated with a particular boundary determined by which boundary curve intersected with the edge connecting it to a deleted node.

The connecting quadrilaterals formed between the overlay mesh and the boundary curves are determined by the shape that they must fill. Therefore the 'best' connection is often not the one which is the closest connection, but rather what creates the most orthogonal angles, or minimizes the difference between all angles and a right angle. Some connections that demonstrate the orthogonal connection are shown in Figure 3.11. This is discussed further in the implementation of several boundary cases in Section 3.3.



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

Figure 3.11 Best choice for point connection

The process of filling in the buffer layer between the overlay mesh and the boundaries can result in quadrilaterals with high aspect ratio. Therefore, when all boundaries have been connected to form a complete quadrilateral grid, the resulting mesh is smoothed using an optimization smoother.

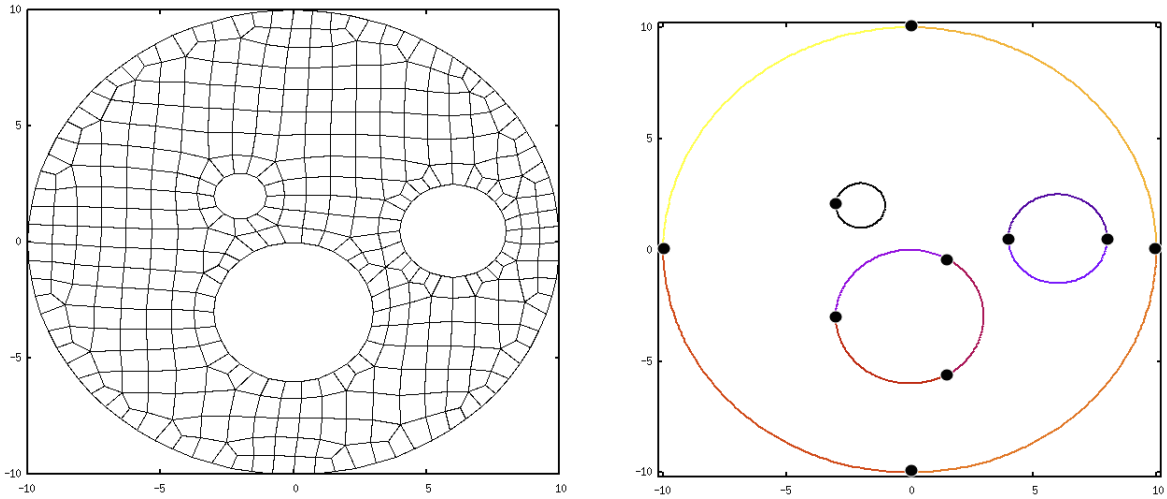
3.3 Grid Test Cases

In order to test the generality of the grid generator, cases were run for a variety of geometries. Below are shown some of the geometries which address the immediate weaknesses of the algorithm structure, together with descriptions of the methods implemented to solve each problem.

3.3.1 Multiple Curves and Geometries

The first consideration in evaluating the scheme is whether it will work for complex structures such as multiple disconnected pieces for a geometry and variable numbers of NURBS curves to describe each boundary. First and most basic is a simple set of circles with multiple curves for each circle. In Figure 3.12 is shown a case of three interior circles increasing in size that are comprised of 1, 2, and 3 curves respectively. The exterior circle has 4 curved components. The NURBS

curve components of each boundary are shown also with the endpoints of each curve marked with a black dot.



(a) Mesh

(b) NURBS Curve segments

Figure 3.12 Test case involving multiple geometries and multiple boundary curves for each geometry

The maximum error for the curve fitting for each curve is shown in Table 3.3.

Table 3.3 Error for NURBS Curve Fitting Circles Geometry

Geometry	Maximum Error
Smallest Circle	9.9301366129890925E-016
Medium Circle Curve 1	1.83103E-015
Medium Circle Curve 2	1.83103E-015
Large Circle Curve 1	6.28037E-016
Large Circle Curve 2	9.93014E-016
Large Circle Curve 3	1.98603E-015
Outer Circle Curve 1	2.512148E-015
Outer Circle Curve 2	2.512148E-015
Outer Circle Curve 3	2.512148E-015
Outer Circle Curve 4	2.512148E-015

The grid was generated by stipulating an overlay mesh with fixed spacing of 1 in both the x and y directions. Due to the fact that all of the boundaries are circular, there was no additional work done by the preprocessor to align grid lines to conform to geometry orientation. There are no dominant directions with respect to coordinate axes for a circle. The automated procedure produced the grid in Figure 3.12 with mesh quality statistics shown in Table 3.4.

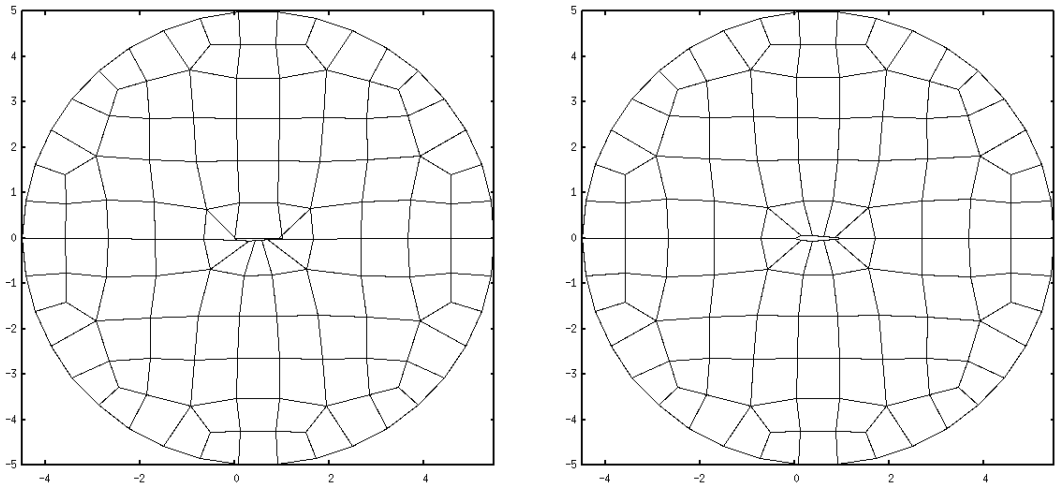
Table 3.4 Mesh Statistics for Circles Geometry

Metric	Minimum value	Maximum Value	Average
Quadrilateral Area	9.638E-002	0.311	0.221
Corner Angle	51.184	146.518	90.000
Aspect Ratio	1.000	1.291	1.0433
Condition Number	1.000	1.828	1.026

3.3.2 Buffer Layer Creation

One of the immediate concerns in developing the algorithm was how to choose the best node connectivity when filling in the buffer layer. The resulting quadrilaterals can become twisted and even entirely inverted unless a robust method for connecting the interior mesh to the boundaries is implemented.

A simple and obvious choice for connecting nodes is to find the closest pair of mesh and boundary nodes and to follow the connectivity around the geometry until all of the nodes have been connected. However, this does not always result in a valid mesh. As an example, the NACA 0012 airfoil is shown in Figure 3.13 with two different choices for connecting the airfoil and farfield boundary to the mesh. Mesh (a) was created using a closest-point criterion for connecting the boundaries whereas Mesh (b) was created by finding the point most orthogonal to the boundary at each endpoint.



(a) Closest point connection

(b) Orthogonal point connection

Figure 3.13 Different connectivity options for buffer layer quadrilaterals

The quality statistics for the mesh with orthogonal connectivity is shown in Table 3.5. The orthogonal boundary connection means that the minimum angles of the quadrilaterals in the buffer layer are maximized.

Table 3.5 Mesh Statistics for NACA 0012 Airfoil

Metric	Minimum value	Maximum Value	Average
Quadrilateral Area	0.296	1.034	0.697
Corner Angle	49.134	172.687	90.000
Aspect Ratio	1.001	2.090	1.169
Condition Number	1.000	18.163	1.211

One of the most challenging cases is a multiple segment 30P30N airfoil as shown with meshes in Figure 3.14 and Figure 3.15. This geometry has several points of interest. However, due to the extremely angular nature of some of the segments, the resulting meshes contained some very high aspect ratio elements as is illustrated in Figure 3.14.

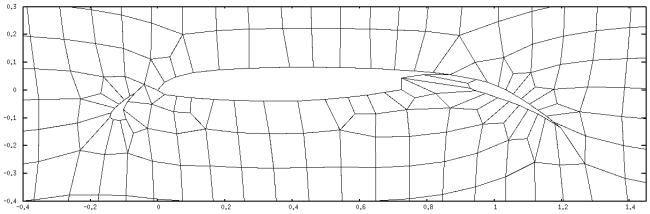


Figure 3.14 30P30N airfoil with coarse mesh

In order to achieve a high quality mesh, a finer overlay grid can be used. However, the fine meshes are computationally expensive and unnecessarily fine given the higher order solver. The separate components of this geometry pose distinct challenges such as a diagonally dominant geometry and acute angles that are not aligned with the coordinate axes determined by the majority of the boundary curves which are discussed earlier. An example of a good quality mesh about the 30P30N airfoil is given in Figure 3.15.

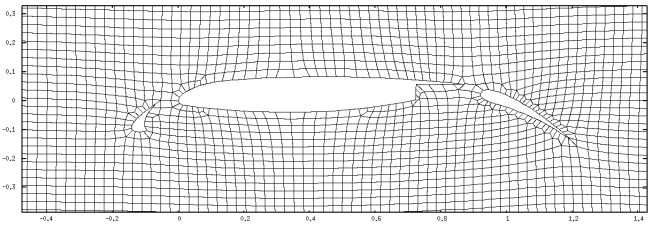


Figure 3.15 30P30N airfoil with fine mesh

The mesh statistics for the mesh in Figure 3.15 are shown in Table 3.6. It is a sign of the quality of the mesh that the corner angle, aspect ratio and condition number values are comparable to the much simpler NACA 0012 mesh that is shown as the orthogonal point connection example in Figure 3.13.

Table 3.6 Mesh Statistics for 30P30N Airfoil

Metric	Minimum value	Maximum Value	Average
Quadrilateral Area	1.644E-004	1.380E-003	8.682E-004
Corner Angle	26.853	176.755	90.000
Aspect Ratio	1.000	2.124	1.051
Condition Number	1.000	18.118	1.005

The best method for meshing this geometry is then manual grid generation. Although time consuming and difficult, the computational cost of using the fine mesh generated with the automated procedure outweighs the benefits of implementing it. For a finite volume or other low order solver, however, the automated process is still helpful for this airfoil as grid resolution is important for these.

3.4 Time Study

One of the important qualities for a useful mesh generator is speed. The algorithm must be quick enough to justify the automating process. Overall, the meshes created in the studies that

follow were generated within the space of 10 minutes, and most of them considerably under 2 minutes.

The average time taken to create a series of grids meshed about three different geometries is illustrated in Table 3.7. The geometries are simple circles with different numbers of points distributed about the perimeter. The only difference in the cases is the number of points distributed.

Table 3.7 Average Run Times for given Mesh Size

Time in Seconds	Number of Grid Nodes
7.100E-03	40
0.283	12,000
6.667	30,000

These times are sufficiently small to justify the use of the generator. However, it is also important to understand the rate at which the time taken to generate the mesh changes with respect to the total size of the resulting mesh. Figure 3.16 illustrates the effect of grid size upon runtime visually. The three lines denote three test geometries run with a range of values for the initial spacing which in turn determines the eventual grid size. The first and smallest geometry file has 10 points, the second has 250, and the third 500 points.

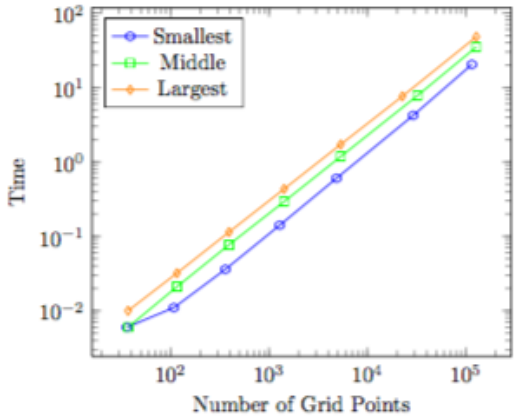


Figure 3.16 Time taken to generate meshes for three trial geometries of varying size

The size of the geometry file has some effect upon the overall time taken. However, the overall rate of change is governed by the grid size. The change in the number of points in the geometry adds a near constant value to the overall runtime behavior. A more detailed discussion of this study is in Appendix B.

CHAPTER 4

SIMULATIONS UTILIZING HIGH ORDER CURVED BOUNDARY DISCRETIZATION AND THE FINITE ELEMENT METHOD

In this chapter the finite element method is briefly outlined. The grid generating algorithm is designed to work for such a method and therefore the requirements for a high order solution algorithm are important to understand. The high order node spacing in particular is important, since the curved boundary elements are designed to distribute such points with more accurate results.

4.1 Overview of Integration Technique

The finite element solution algorithm implements a numerical integration scheme that is based on Gauss quadrature, in which the solution is approximated as a series of unknown values at each node multiplied by an interpolating polynomial. A variable u is approximated on an element Ω with k points as $u = \sum_{n=1}^{k} \phi_n u_n$ where u_n denote the values of u at each point n and ϕ_n are the basis functions of order k such that $\phi_i(x_j, y_j) = \delta_{i,j}$.

With a polynomial interpolation of the solution variable at each node, numerical integration can be implemented by setting

$$\int_a^b u(x)dx \approx \sum_{i=1}^{k} w_i \phi_i u_i \tag{4.1}$$

where w_i denote the quadrature weights. Note that for this research, the order of numerical integration scheme is always the same as the order of the interpolation polynomials. The quadrature rule used in this research implements a Chebyshev point distribution. Chebyshev points are computed easily for a segment $[a, b]$. These points are the projection onto the diameter of the circle centered at $(\frac{a+b}{2}, 0)$ of radius $\frac{b-a}{2}$ of equally spaced points distributed along the upper half of the circle [10] that has the line segment as diameter. The distribution for six points over the segment $[-1, 1]$ is shown in Figure 4.1.

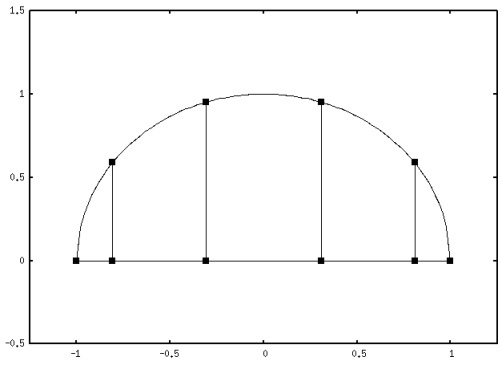


Figure 4.1 Chebyshev points

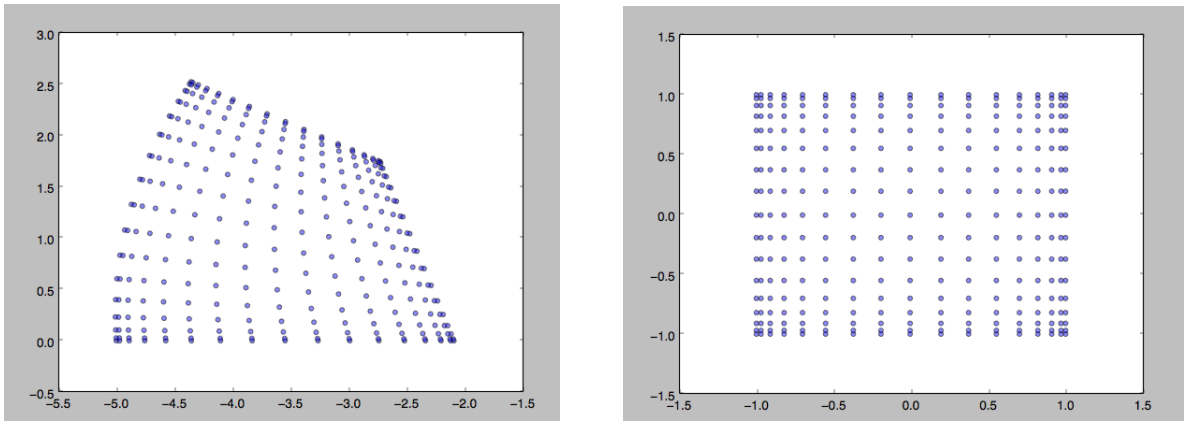
The formula for computing n Chebyshev points $x_j, j = 0, ..., n$ is given by

$$x_j = \frac{1}{2}(a + b) + \frac{1}{2}(b - a)cos\left(\frac{\pi j}{n}\right), j = 0, ..., n \tag{4.2}$$

One of the benefits of using quadrilateral elements is that the extension to two dimensional elements is simple for quadrilaterals as it is merely the combination of all of the coordinates points

taken from the distribution of points along the x and y directions. This is not true for triangular or other non-quadrilateral element meshes.

The points are distributed on an ideal square and then a transformation from each quadrilateral to the ideal square is defined for the problem domain. The ideal square is the domain $[-1, 1] \times [-1, 1]$, for which the weights for integration are defined [10]. Figure 4.2 shows a quadrilateral of order 14 with the ideal square. The transformation maps the points on the ideal square distributed using Chebyshev point placement to points on the mesh quadrilateral using a bijective relation.



(a) Physical coordinate rectangle

(b) Ideal square

Figure 4.2 Chebyshev point distribution for physical rectangle and ideal distribution

For each segment, the distribution of Chebyshev points of order p results in a total of $p + 1$ points that divide the domain into p segments. Thus the total number of points for a p order quadrilateral is $(p + 1) \times (p + 1)$.

Because of this multiplication in two dimensions, the resulting solution domain can contain considerably more points for high order schemes than are in the original mesh. This suggests a comparison between grid spacing and element order. The differences and respective value of these two approaches are studied in Section 5.2. Meanwhile, the ability to implement higher order elements is demonstrated. Some examples of meshes with Chebyshev point spacing are shown in Figure 4.3 and Figure 4.4 .

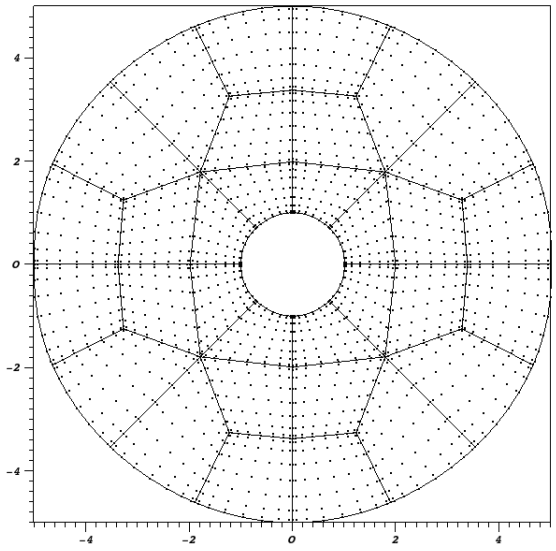


Figure 4.3 Concentric circles mesh with higher order point distribution

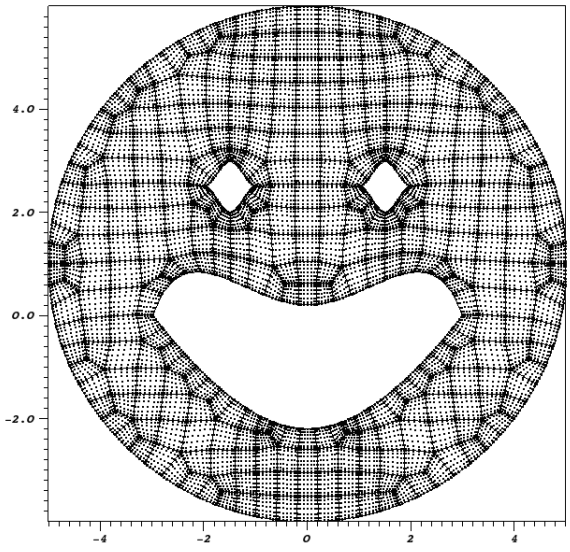


Figure 4.4 Smile mesh with higher order point distribution

The points are distributed throughout each of the elements by means of the transformation defined for each element and are fitted to the curved edges of the boundary to capture high resolution data for solution accuracy. This transformation is computationally slow and an analysis of the runtime costs of using high order schemes and fine meshes is performed in Section 5.2.

4.2 Weak Formulation

The weak formulation of a differential equation is a useful restatement of the original problem in which the differentiability requirement of the original variable is weakened. This form is derived using the identity from calculus known as the product rule.

$$\frac{d}{dx}(f(x)g(x)) = \frac{df(x)}{dx}g(x) + f(x)\frac{dg(x)}{dx} \tag{4.3}$$

Rearranging the terms and integrating over a domain Ω yields

$$\int_{\Omega} \frac{df(x)}{dx} g(x) d\Omega = \int_{\delta\Omega} f(x) g(x) n_x d\delta\Omega - \int_{\Omega} f(x) \frac{dg(x)}{dx} d\Omega \tag{4.4}$$

Where n_x, n_y are the unit normal vectors around the boundary $\delta\Omega$.

This is useful particularly when dealing with higher derivatives. Thus for example, in order to solve the Equation 4.5

$$\frac{\partial^2 f}{\partial x^2} = 0 \tag{4.5}$$

First multiply by another variable g, and then integrate over a domain Ω

$$\int_{\Omega} g \frac{\partial^2 f}{\partial x^2} d\Omega = 0 \tag{4.6}$$

However, using the calculus identity, this equation can be rewritten

$$\int_{\Omega} g \frac{\partial^2 f}{\partial x^2} d\Omega = \int_{\delta\Omega} g \frac{\partial f}{\partial x} d\delta\Omega - \int_{\Omega} \frac{\partial g}{\partial x} \frac{\partial f}{\partial x} d\Omega \tag{4.7}$$

One of the important points that makes this process useful is that it decreases the highest derivative. In Equation 4.7 there is no higher derivative term than first order.

The finite element method is commonly used to solve the weak form of an equation. The problem is first cast in terms of an approximation to the solution and then multiplied by a function ϕ in the same family of polynomials as the interpolating polynomials used to approximate the solution. The weak form of the problem is taken and results in a system of matrix equations.

4.3 Circular Waveguide Applications

An example that shows how the finite element scheme is implemented using a weak formulation is an initial value problem describing the electric and magnetic components in a circular waveguide as discussed in [11].

The equations for electric and magnetic fields in a circular waveguide are most conveniently dealt with using the cylindrical coordinate system. For the transverse magnetic modes, the field components are computed using the vector potential $A = A_z(\rho, \phi, z)\hat{k}$ where $\hat{k}$ is the unit vector in the z-direction. This potential satisfies the equation

$$\nabla^2 A(\rho, \phi, z) + \beta^2 A(\rho, \phi, z) = 0 \tag{4.8}$$

which in cylindrical coordinates becomes

$$\frac{\partial^2 A}{\partial \rho^2} + \frac{1}{\rho}\frac{\partial A}{\partial \rho} + \frac{1}{\rho^2}\frac{\partial^2 A}{\partial \phi^2} + \frac{\partial^2 A}{\partial z^2} + \beta^2 A = 0 \tag{4.9}$$

The solutions to Equation 4.9 are of the form

$$A(\rho, \phi, z) = [A_1 J_m(\beta_\rho \rho) + B_1 Y_m(\beta_\rho \rho)] \times [A_2 cos(m\phi) + B_2 sin(m\phi)] \times [A_3 e^{-i\beta_z z} + B_3 e^{i\beta_z z}] \tag{4.10}$$

Where $\beta^2 = \beta_z^2 + \beta_\rho^2$ and J_m, Y_m are m-order bessel functions of the first and second kind respectively and the constants A_1, A_2, A_3, B_1, B_2, B_3 can be determined by applying boundary conditions

which are shown below [11].

$$
\begin{aligned}
&E_\phi(\rho = a, \phi, z) = E_\phi(\rho, \phi, z)\big|_{\rho \in \delta\Omega} = 0 \\
&E_z(\rho = a, \phi, z) = E_z(\rho, \phi, z)|_{\rho \in \delta\Omega} = 0 \\
&\text{The fields must be infinite everywhere} \\
&\text{The fields must repeat every } 2\pi \text{ radians in } \phi
\end{aligned}
\tag{4.11}
$$

Using the resulting complete form A, the field components are given by

$$
\begin{aligned}
E_\rho = \quad & -i\tfrac{1}{\omega\mu\epsilon}\tfrac{\partial^2 A}{\partial\rho\partial z} && = -B_{mn}\tfrac{\beta_z\beta_\rho}{\omega\mu\epsilon}J'_m(\beta_\rho\rho)[A_2 cos(m\phi) + B_2 sin(m\phi)]e^{-i\beta_z z} \\
E_\phi = \quad & -i\tfrac{1}{\omega\mu\epsilon}\tfrac{1}{\rho}\tfrac{\partial^2 A}{\partial\phi\partial z} && = -B_{mn}\tfrac{m\beta_z}{\omega\mu\epsilon}\tfrac{1}{\rho}J_m(\beta_\rho\rho)[-A_2 sin(m\phi) + B_2 cos(m\phi)]e^{-i\beta_z z} \\
E_z = \quad & -i\tfrac{1}{\omega\mu\epsilon}(\tfrac{\partial^2}{\partial z^2} + \beta^2)A && = -iB_{mn}\tfrac{\beta_\rho^2}{\omega\mu\epsilon}J_m(\beta_\rho\rho)[A_2 cos(m\phi) + B_2 sin(m\phi)]e^{-i\beta_z z} \\
H_\rho = \quad & \tfrac{1}{\mu}\tfrac{1}{\rho}\tfrac{\partial A}{\partial\phi} && = B_{mn}\tfrac{m}{\mu}\tfrac{1}{\rho}J_m(\beta_\rho\rho)[-A_2 sin(m\phi) + B_2 cos(m\phi)]e^{-i\beta_z z} \\
H_\phi = \quad & -\tfrac{1}{\mu}\tfrac{\partial A}{\partial\rho} && = -B_{mn}\tfrac{\beta_\rho}{\mu}J'_m(\beta_\rho\rho)[-A_2 sin(m\phi) + B_2 sin(m\phi)]e^{-i\beta_z z} \\
H_z = \quad & 0
\end{aligned}
\tag{4.12}
$$

The formulation implemented in the finite element solver was derived by using the fact that the solution is time harmonic. That is, $\mathbf{E}(\rho, \phi, z, t) = E(\rho, \phi, z)e^{i\omega t}$ and therefore $\frac{\partial^2\mathbf{E}}{\partial t^2} = -\omega^2\mathbf{E}$. Thus the Helmholtz equation can be solved by computing

$$
\frac{\beta^2}{\omega^2}\frac{\partial^2\mathbf{E}}{\partial t^2} - \nabla^2\mathbf{E} = 0 \tag{4.13}
$$

Using the weak form this becomes

$$\begin{aligned}\left(\frac{\beta^2}{\omega^2}\int_\Omega \phi_i\phi_j d\Omega\right)\frac{\partial^2 E_j}{\partial t^2} - \left(\int_\Omega\left(\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x}+\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y}\right)d\Omega\right)E_j + \\ \left(\int_{\delta\Omega}\phi_i\left(\frac{\partial\phi_j}{\partial x}n_x+\frac{\partial\phi_j}{\partial y}n_y\right)d\delta\Omega\right)\frac{\partial E_j}{\partial t} = \quad 0\end{aligned} \tag{4.14}$$

Note that this is a series of scalar equations that multiply by the vector quantities E_j, $\frac{\partial E_j}{\partial t}$, $\frac{\partial^2 E_j}{\partial t^2}$. The boundary integral terms cancel at interior edges of the mesh due to the normal component, and at the boundary the values are specified with Dirichelet boundary conditions. Using numerical integration, the integral quantities are approximated as follows.

$$M_{i,j} = \frac{\beta^2}{\omega^2}\sum_{n=1}^{k}\phi_i\phi_j w_k \tag{4.15}$$

$$K_{i,j} = \sum_{n=1}^{k}\left(\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x}+\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y}\right)w_k \tag{4.16}$$

And the final form of the solution algorithm is

$$M_{i,j}\frac{\partial^2 E_j}{\partial t^2} + K_{i,j}E_j = 0 \tag{4.17}$$

The time integration was evaluated using a leapfrog algorithm that follows a Velocity Verlet stepping scheme described by Young [12]. The solution was computed for the TM_{31} and TM_{61} modes on a mesh with initial spacing of 0.25 in each direction and $p = 4$ elements. This mesh is shown in Figure 4.5.

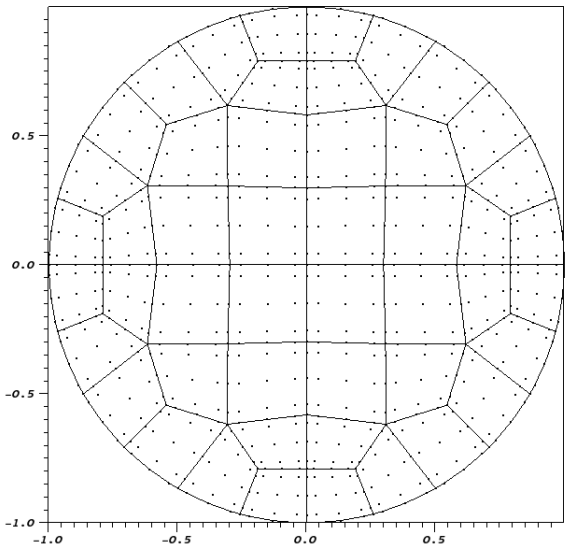
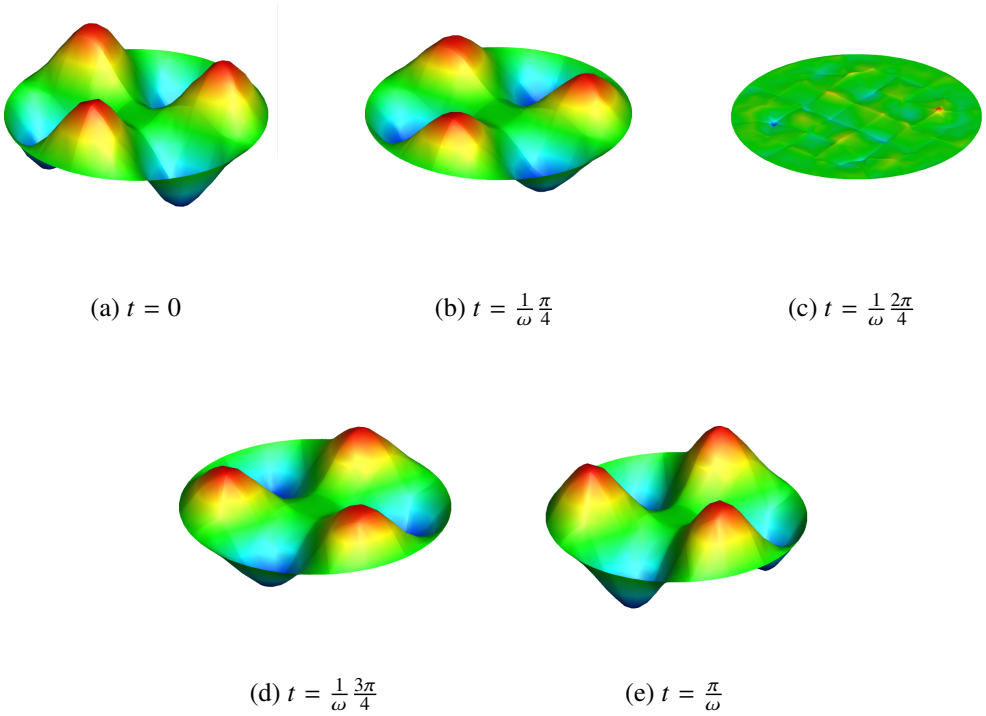


Figure 4.5 Mesh For Circular Waveguide Solutions

Some snapshots detailing the first half oscillation for each mode are shown in Figure 4.6. Note that the time t represents the values $\frac{1}{\omega}\frac{\pi}{4}$, $\frac{1}{\omega}\frac{2\pi}{4}$, $\frac{1}{\omega}\frac{3\pi}{4}$ and $\frac{\pi}{\omega}$ only approximately, and therefore these are not the exact maxima and minima. This is also the reason why the solutions for each mode at $t = \frac{2\pi}{4\omega}$ are not uniformly 0 but show slight variations in value.

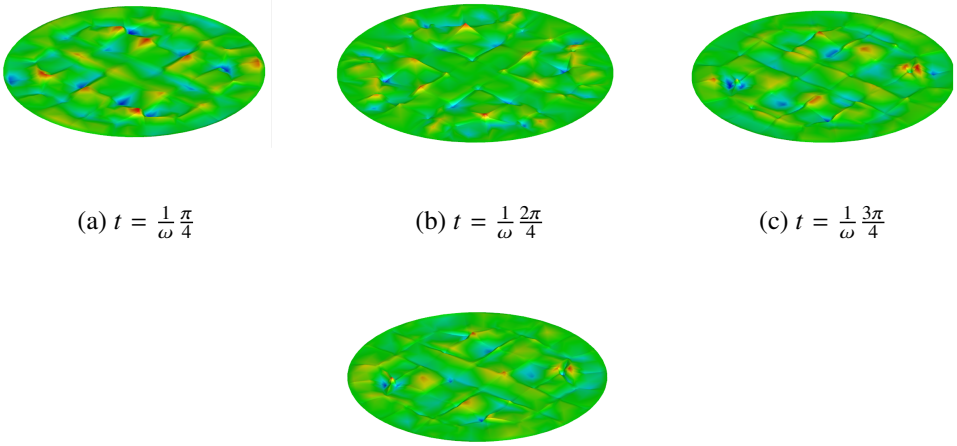


(a) $t = 0$ (b) $t = \frac{1}{\omega}\frac{\pi}{4}$ (c) $t = \frac{1}{\omega}\frac{2\pi}{4}$

(d) $t = \frac{1}{\omega}\frac{3\pi}{4}$ (e) $t = \frac{\pi}{\omega}$

Figure 4.6 Solutions to Helmholtz equation for transverse magnetic mode TM_{31}

The error for each of these snapshots was also computed. For the first case it is simply the error from the interpolation. Therefore only the errors for all the snapshots at time $t > 0$ are shown in Figure 4.7. Because the scheme is of order $p = 4$, the second order time integration scheme contributes the dominant error to the entire procedure.

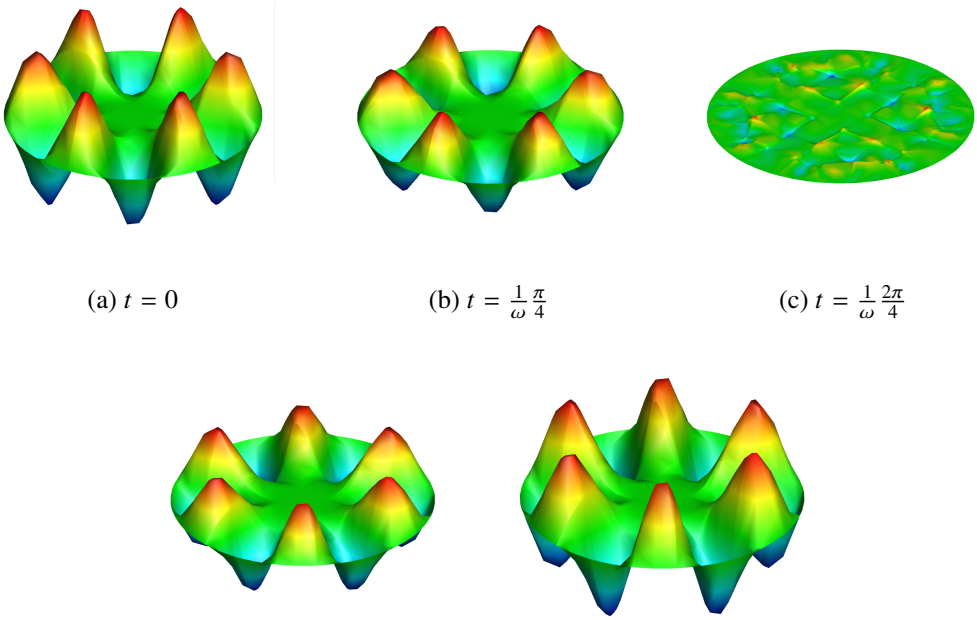


(a) $t = \frac{1}{\omega}\frac{\pi}{4}$ (b) $t = \frac{1}{\omega}\frac{2\pi}{4}$ (c) $t = \frac{1}{\omega}\frac{3\pi}{4}$

(d) $t = \frac{\pi}{\omega}$

Figure 4.7 Error for computed Helmholtz equation for transverse magnetic mode TM_{31}

In addition to the TM_{31} mode, the electric field component E_z was computed for the 61 mode as well. The solution at points during the first half oscillation are shown in Figure 4.8. Since this mode has the same time period for oscillation as the 31 mode, the snapshots are at the same time step as in Figure 4.6. They are all on the order of

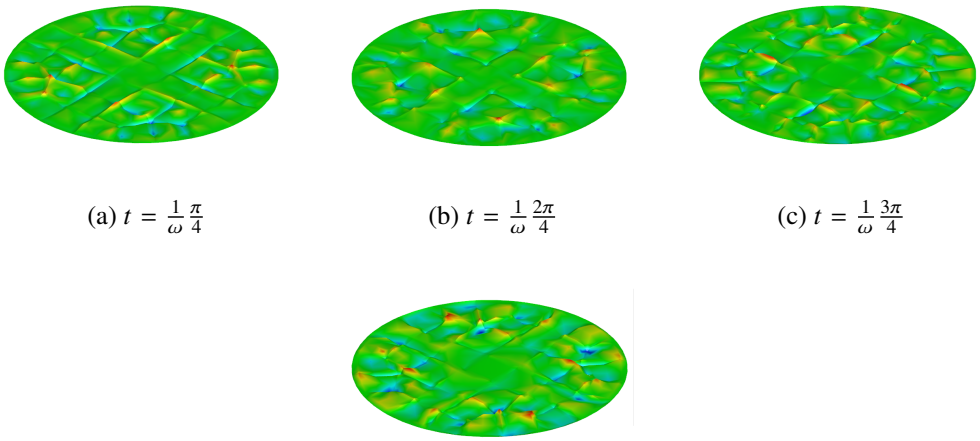


(a) $t = 0$ (b) $t = \frac{1}{\omega}\frac{\pi}{4}$ (c) $t = \frac{1}{\omega}\frac{2\pi}{4}$

(d) $t = \frac{1}{\omega}\frac{3\pi}{4}$ (e) $t = \frac{\pi}{\omega}$

Figure 4.8 Solutions to Helmholtz equation for transverse magnetic mode TM_{61}

The errors for the last four snapshots for the 61 mode are shown in Figure 4.9.



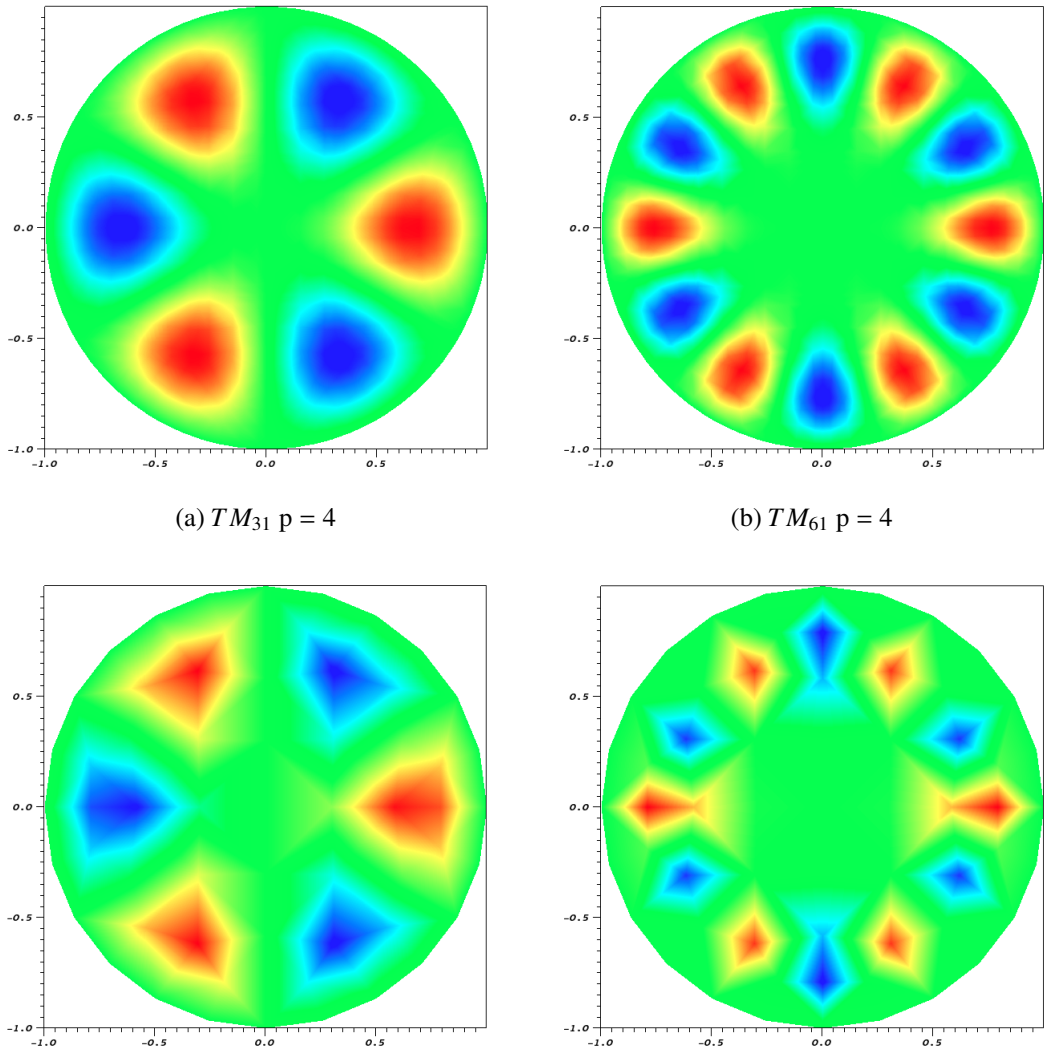
(a) $t = \frac{1}{\omega}\frac{\pi}{4}$ (b) $t = \frac{1}{\omega}\frac{2\pi}{4}$ (c) $t = \frac{1}{\omega}\frac{3\pi}{4}$

(d) $t = \frac{\pi}{\omega}$

Figure 4.9 Error for computed Helmholtz equation for transverse magnetic mode TM_{61}

4.4 Higher Order Elements

The example of the Transverse magnetic mode solutions in a circular waveguide can also be used to illustrate the immense effect of node spacing on the solution accuracy. The results shown in Figures 4.6 and 4.8 were computed using higher order elements. The same solution is shown in Figure 4.10 with $p = 4$ elements and $p = 1$ elements. The resolution of the solution alone is dramatic. In addition, the information is more exact due to the improved resolution at the boundaries and in the areas of high solution gradients.



(a) TM_{31} p = 4

(b) TM_{61} p = 4

(c) TM_{31} p = 1

(d) TM_{61} p = 1

Figure 4.10 Difference between solution quality for higher and lower order elements

The effect upon solution quality of using higher order elements is further illustrated in the following example. The governing equations for Maxwell's equations in two dimensions for a rectangular waveguide with Perfect Electric Conductor (PEC) boundary conditions are given by Equation 4.18 [11].

$$\begin{aligned} \frac{\partial E_x}{\partial t} &= \frac{\partial H_z}{\partial y} \\ \frac{\partial E_y}{\partial t} &= -\frac{\partial H_z}{\partial x} \\ \frac{\partial H_z}{\partial t} &= \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} \end{aligned} \tag{4.18}$$

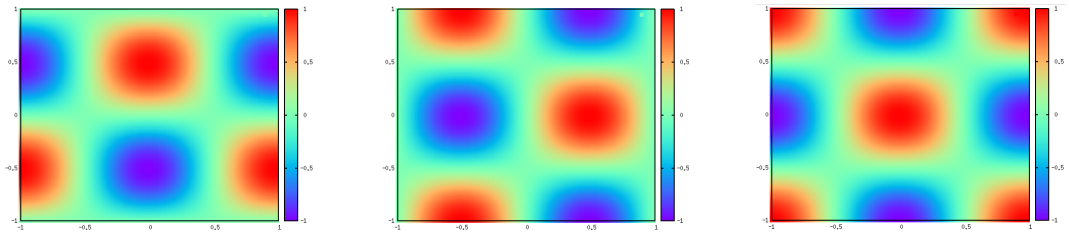
which can be written in matrix form as

$$\frac{\partial}{\partial t}\begin{bmatrix} E_x \\ E_y \\ H_z \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} E_x \\ E_y \\ H_z \end{bmatrix} + \frac{\partial}{\partial y}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} E_x \\ E_y \\ H_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.19}$$

The solution is

$$\begin{aligned} E_x &= -\frac{n\pi}{\omega} cos(m\pi x) sin(n\pi y) sin(\omega t) \\ E_y &= \frac{n\pi}{\omega} sin(m\pi x) cos(n\pi y) sin(\omega t) \\ H_z &= cos(m\pi x) cos(n\pi y) cos(\omega t) \end{aligned} \tag{4.20}$$

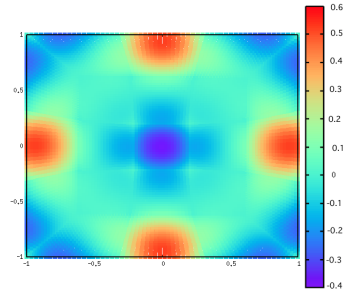
The field components in a rectangular waveguide are shown in Figure 4.11 for the TE_{11} mode at time $t = 0$.



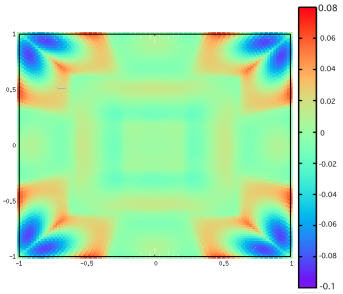
(a) E_x (b) E_y (c) H_z

Figure 4.11 Field components for rectangular waveguide using transverse electric Maxwell's equations

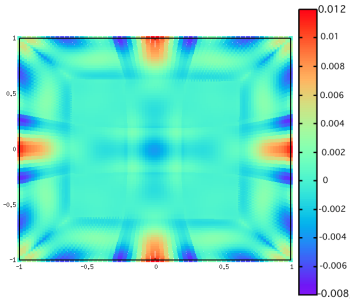
With a solution that has large gradients, grid spacing is crucial. A large number of points are required to capture the curvature of solutions with high curvature. A visual example of the importance of higher order elements is shown below. The error for the initialized magnetic field component H_z at $t = 0$ with respect to the analytical solution is shown plotted for different values of p ranging from 1 to 8 with constant grid spacing.



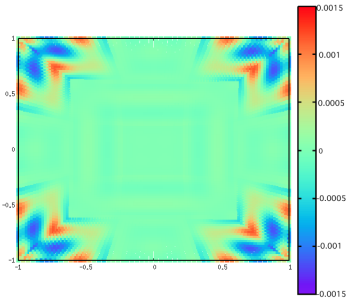
(a) p = 1



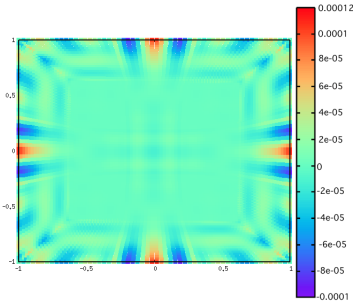
(b) p = 2



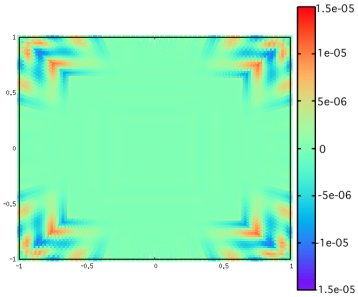
(c) p = 3



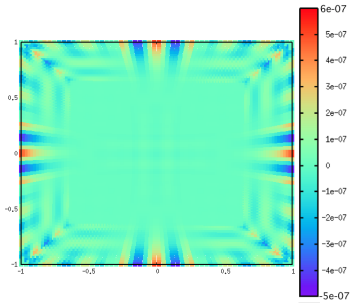
(d) p = 4



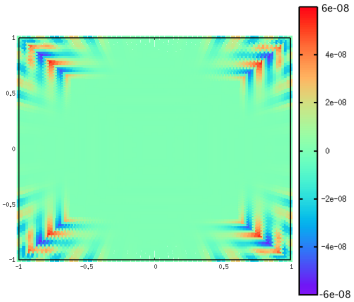
(e) p = 5



(f) p = 6



(g) p = 7



(h) p = 8

Figure 4.12 Error for magnetic field component computed with increasing orders of accuracy

Not only are the errors more smoothly varying between elements, but the values of the errors themselves decrease with the order of the approximation as well. Although the same color scale is used in each plot to illustrate the variation within the error between maximum and minimum values, the maximum and minimum in each case decreases from plot to plot. The scaling is necessary as the errors for $p = 8$ elements do not show up if plotted using the same color bar as with lower order elements. This phenomenon was explored further by instantiating the solution and comparing the initial error on a series of grids of varying size. The difference in values for solutions on two different grids gives a slope when compared to the size of the grids themselves. When graphed, the slopes reflect the order of accuracy. The results are shown in Figure 4.13.

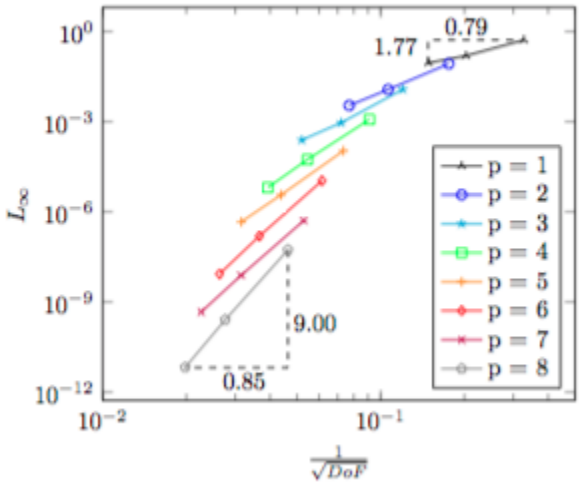


Figure 4.13 Error of magnetic field distribution with respect to grid size

Thus the demonstrated accuracy of the scheme for a p order element is $p + 1$. When compared with more traditional Finite Volume solvers, the ease of increasing accuracy of such a scheme to improve the solution is a powerful incentive for using the Finite Element method.

CHAPTER 5

EXAMPLES

This chapter deals with various cases that arise when using a finite element solver that demonstrate the utility of the grid generator. Not only is automated mesh creation helpful for meshing for complex geometries, but functionality for changing the initial spacing of the overlay mesh makes it possible to perform studies with large numbers of meshes of the same geometry but with different spacing with ease. Section 5.1 contains a study of the observed order of accuracy of the finite element scheme for different values of p. Section 5.2 is devoted to a comparison of mesh refinement and increased element order with respect to solution accuracy.

5.1 Order of Accuracy

One of the outstanding features of the finite element method is the facility for higher order solution accuracy. Therefore an important consideration in implementing the solver with an all-quadrilateral mesh is whether the order of accuracy of the solver is preserved for specified values of p. The quadrature rules for quadrilaterals and triangles are different and therefore it is important to check this step.

A simple test case that involves second derivatives in both x and y directions is Laplace's equation. To compute the observed order of accuracy of the scheme, the Laplace equation is solved

on a series of quadrilateral meshes of increasing grid spacing. The error for each mesh between the computed and the analytical solution varies with element order.

The Laplace equation in two dimensions is given by

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0 \tag{5.1}$$

The solution to this equation can be computed by using separation of variables. First identify Φ as the product of two functions of the coordinate directions x and y respectively.

$$\Phi(x, y) = X(x)Y(y) \tag{5.2}$$

And denoting $\frac{\partial^2 f}{\partial x^2}$ and $\frac{\partial^2 f}{\partial y^2}$ by $\ddot{f}_x, \ddot{f}_y$ respectively we obtain the equality

$$\frac{\ddot{X}_x}{X} = -\frac{\ddot{Y}_y}{Y} = C \tag{5.3}$$

for some constant C. This can be simplified by dividing through with the following result.

$$\ddot{X}_x = CX(x), \ddot{Y}_y = -CY(y) \tag{5.4}$$

These equations are solved for any combination of C_0e^{Cx}, C_0e^{-Cy} or the trigonometric components $cos(Cx, Cy)$, $cosh(Cx, Cy)$, $sin(Cx, Cy)$ and $sinh(Cx, Cy)$ that preserve the conditions

$$\begin{aligned} \ddot{\Phi}_x &= -\ddot{\Phi}_y \\ \ddot{X}_x &= CX \\ \ddot{Y}_y &= -CY \end{aligned} \tag{5.5}$$

Thus there are a range of values for X and Y and consequently Φ which will satisfy Equation 5.1 for given boundary conditions.

For this example, Dirichelet boundary conditions are chosen that correspond to an analytical solution of $\Phi(x, y) = cos(\frac{4\pi x}{5})cosh(\frac{4\pi y}{5})$ as pictured in Figure 5.1.

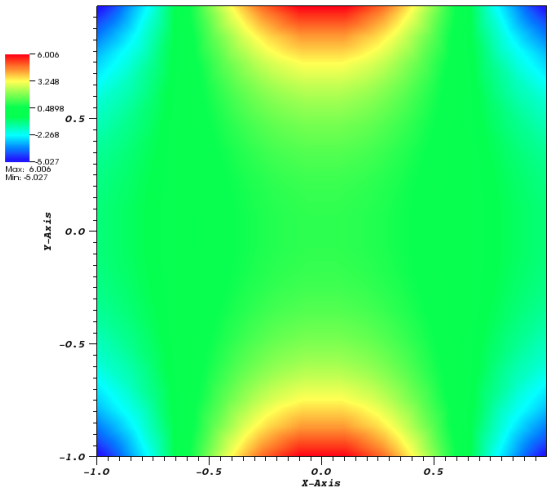
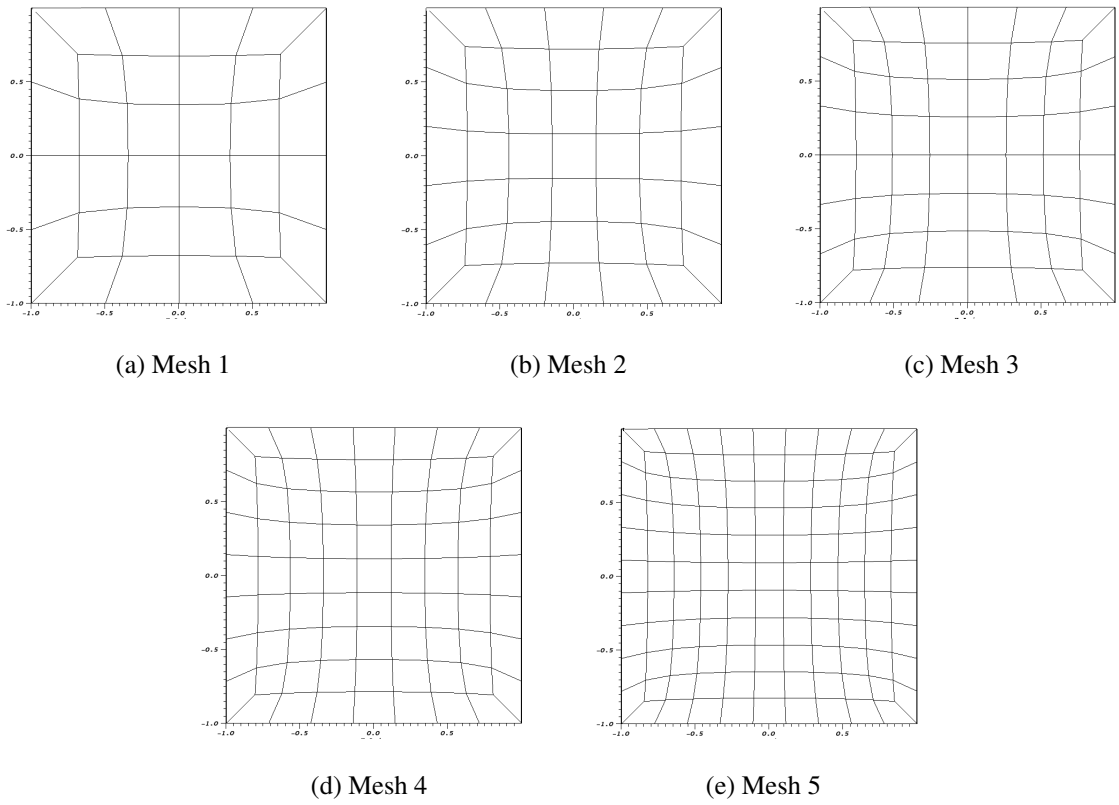


Figure 5.1 Solution to Laplace equation

The solution was computed for five grids of successively larger spacing as shown in Figure 5.2. These grids were generated by altering the input spacing of the overlay mesh for each successive case.



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

(d) Mesh 4 (e) Mesh 5

Figure 5.2 Meshes with varying coarseness for error analysis

For each of the grids created, the solution was computed using p-order elements where $p = 1, ..., 8$. The L_∞ norms of the error for each grid size was compared with fixed p. This is shown in Figure 5.3. As can be seen in the plot, the slopes are very close to the expected values.

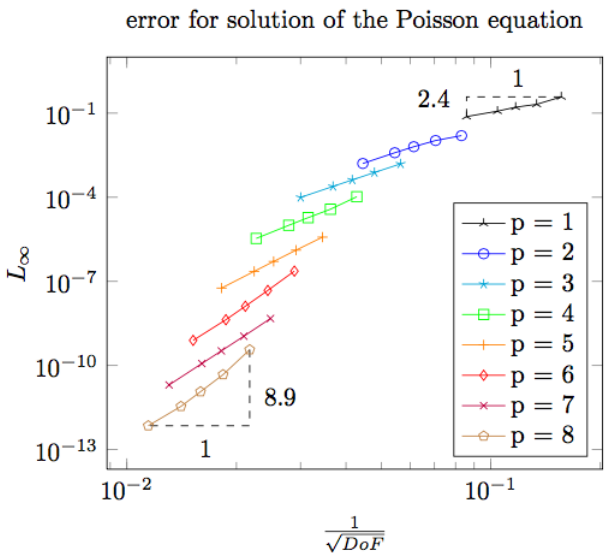


Figure 5.3 Error study for Laplace equation

Thus the accuracy of the scheme is demonstrated for higher order elements.

5.2 Refinement Studies for a Finite Element Solver

There are two very common ways to increase resolution and accuracy for the solution of a problem. The first of these is to implement a higher order scheme as described in the previous chapter, and the second is to solve the problem on a grid with more dense point placement. The grid generator is capable of creating grids of increasingly fine spacing, and therefore a higher order scheme might not be necessary. The order of accuracy of the higher order scheme is clearly maintained for quadrilateral grids, but the overall effectiveness of such a scheme as a means of improving solution accuracy is not established.

Two case studies are presented here in order to guage the extent of the different effects of grid refinement and element order on the accuracy of the finite element solution algorithm. The pressure coefficient C_p for potential flow solution about a NACA 0012 airfoil is used as the solution computed for comparison. The geometry is shown in Figure 5.4.

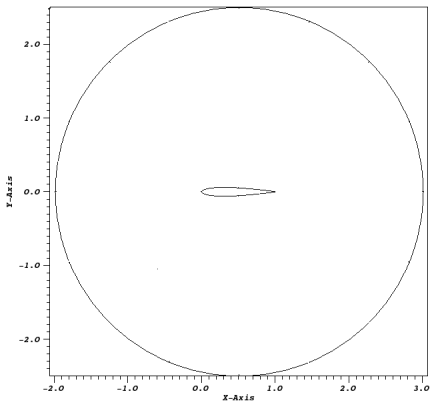
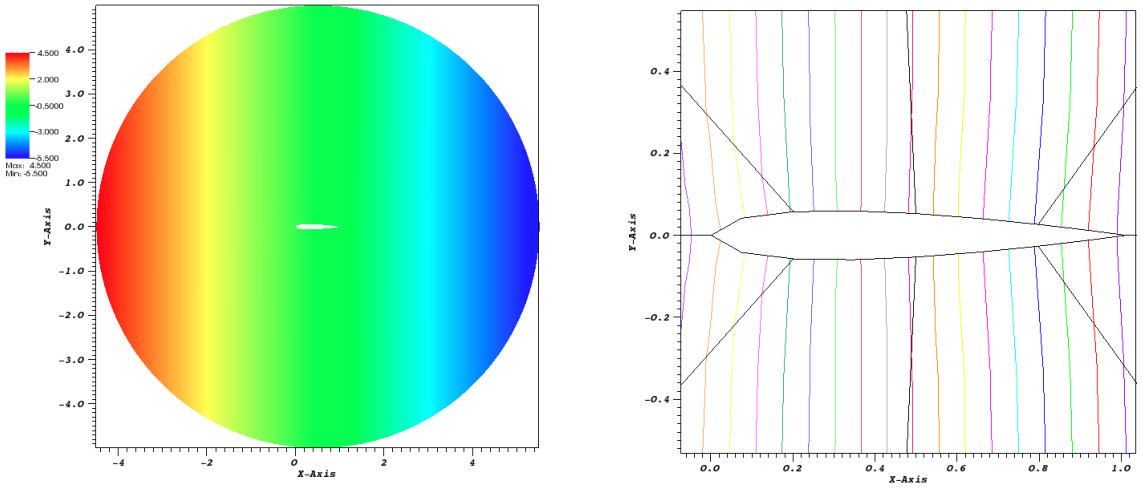


Figure 5.4 NACA 0012 geometry

The equation used to describe the potential flow solution is also the Laplace equation which gives the velocity potential formulation

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0 \tag{5.6}$$

With Dirichelet boundary conditions $\Phi(x, y) = -x$ specified on the farfield and von Neumann boundary conditions $\frac{\partial \Phi}{\partial x} n_x + \frac{\partial \Phi}{\partial y} n_y = 0$ on the airfoil surface. The velocity potential solution is shown in Figure 5.5 together with a closeup view of the geometry with solution contour lines.



(a) Velocity potential solution Φ

(b) Contours of solution at boundary

Figure 5.5 Potential flow solution

In order to visualize the impact of element order upon the accuracy of the solution, a mesh with fixed spacing $dx = dy = 0.5$ was used with varying values of the order p. This mesh is shown in Figure 5.6.

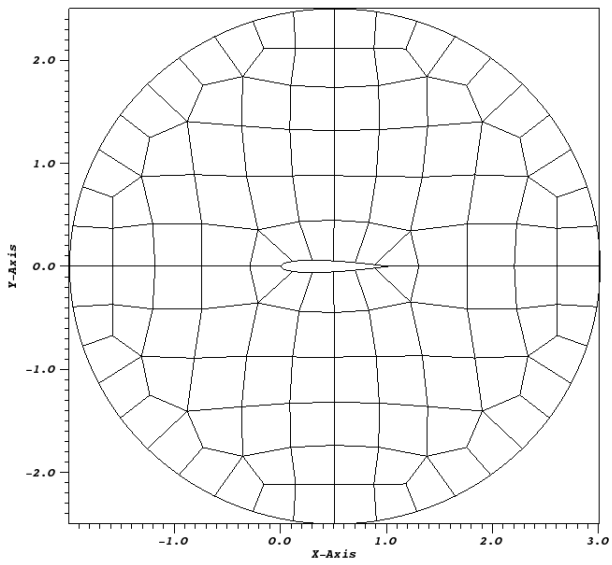


Figure 5.6 Mesh used for p-refinement

The computed values of C_p for elements of order 4, 5, 6 and 8 are shown in Figure 5.7. It can be seen that the increased order has particular effect on the quality of the solution at the endpoints of the airfoil, where the points are more dense with a higher order implementation. Most particularly at the front tip of the airfoil where the curvature is greatest the resolution of points improves solution quality dramatically.

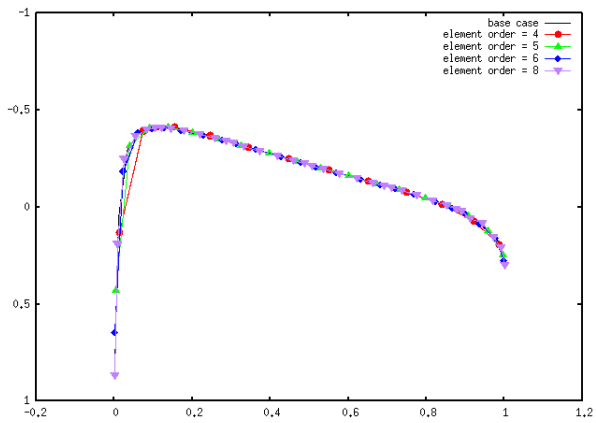


Figure 5.7 Effect of p refinement on C_p

The same set of cases were run on the coarsest mesh with similar results. The solutions for this mesh were roughly the same as for the finer mesh but with the effect of diminishing the element order by 2. Thus the solution with $p = 3$ elements on the fine mesh is similar to the solution with $p = 5$ elements on the coarse mesh. The solutions with $p = 8$ elements on the fine mesh and $p = 10$ elements on the coarse mesh are shown in Figure 5.8

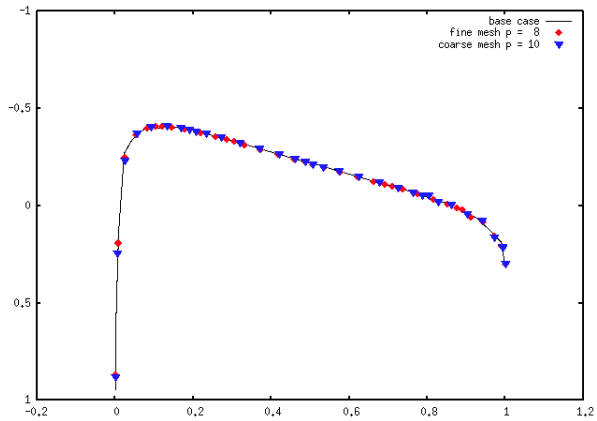


Figure 5.8 Element order compared for different grid sizes

For h-refinement, the element order was held constant at 6 while the mesh spacing was varied. The initial mesh had an average spacing of $dx = dy = 2.0$, and for each successive case the spacing was halved in each direction.

The computed values of C_p with case 1 being the coarsest mesh and case 4 the finest are shown in Figure 5.9.

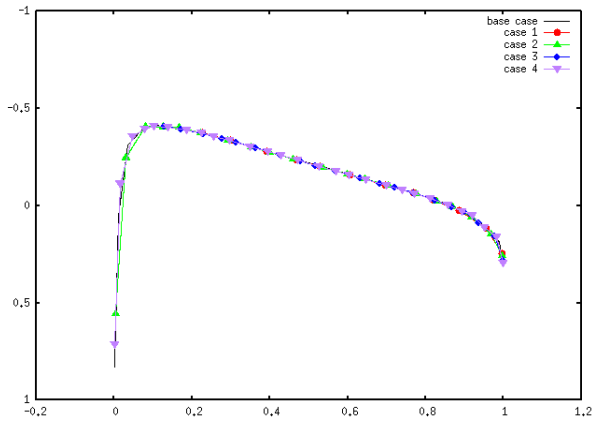


Figure 5.9 Effect of h refinement on C_p

As can be seen, the quality of the solution is improved by using a finer mesh but with a much smaller change than is observed in the p refinement study. The overall solutions for the h refinement study are much better than the ones computed in the p refinement case. However, this is because a high value for p was used in the first place. The results for the same set of meshes with a constant element order of 4 is shown in Figure 5.10

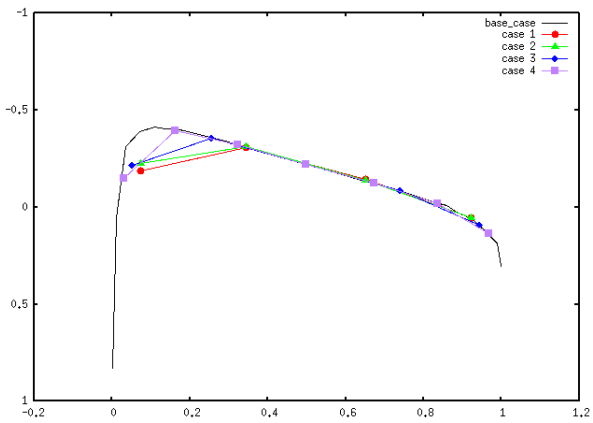


Figure 5.10 Effect of h refinement on C_p with lower order elements

The solution is considerably degraded, and the change in grid resolution has very little positive effect when compared with the change in solution quality demonstrated in Figure 5.7. Therefore, the comparative accuracy of the solutions in the first h-refinement study do not signify a better rate of change in solution accuracy.

The result of this visual comparison is a clear indication of the effectiveness of higher order elements in increasing the accuracy of a computed solution. Although increasing both the element order and the grid size does improve the overall solution quality, the rate of change in accuracy with respect to grid spacing is much lower than that of accuracy with respect to element order.

Although the high order scheme produces better results more quickly with respect to grid size, it takes a much longer time computationally. The runtimes for the p refinement study and the second h refinement study are shown in Tables 5.1 and 5.2. The total time is divided into the amount of time taken to generate the mesh and the total time taken to distribute higher order points for each element.

Table 5.1 Runtimes for p refinement study

Mesh Size	Mesh Generation	Point Distribution
588	4.400E-02	3.000E-02
1024	4.200E-02	5.200E-02
1580	4.100E-02	8.400E-02
3052	4.100E-02	0.204

Table 5.2 Runtimes for h refinement study

Mesh Size	Mesh Generation	Point Distribution
588	1.700E-02	9.00E-03
1560	4.400E-02	2.900E-02
5022	1.060	1.93
19440	0.492	2.620

The amount of time taken in mesh generation is comparable to that of the point distribution on the small mesh for $p = 4$ or 5, but it is overshadowed completely by the time taken to generate Chebyshev points for higher orders. Since the times scale with the size of the meshes, a very

simple way to speed up runtime for large cases is to generate a finer mesh and use a slightly lower value of p for each element.

Using a high order solver has been shown to improve solution quality more efficiently than increasing the number of grid points. However, this does not necessarily mean that high order schemes are more time efficient. The mesh generating algorithm has been shown to create large grids very quickly whereas the high order scheme is extremely expensive in comparison. Therefore in order to obtain the best results in the quickest time, a mesh of moderately small spacing with an implementation of $p = 3$ to $p = 5$ elements is a practical option that maximizes ratio of accuracy and runtime cost.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The mesh generator that was developed satisfies the requirements of efficiency and generality. For each case, a single mesh is created and then refined to fit the geometry without the need to generate a triangular mesh or subdivide the domain into patches. Generality is achieved by implementing a preprocessor that insures a valid overlay mesh that has the correct quadrilateral intersection types to be tested at a later stage in the algorithm. The method used is automated and can be used for arbitrary geometry as has been shown by meshing various geometries. The scheme is also time efficient. Although the runtime does scale with the size of the mesh to be generated, the overall runtimes for a variety of mesh sizes are small. Therefore the automating process is justified for speed.

Different geometries illustrating some of the challenges that arise in quadrilateral mesh generation are studied and meshes successfully created to fit them. One of the most challenging types of geometries, a multi section 30P30N airfoil is successfully meshed. Test cases including examples from aerodynamics and electomagnetics are studied. The utility of having an automated procedure with capability for refinement has been shown with both error analysis and refinement studies.

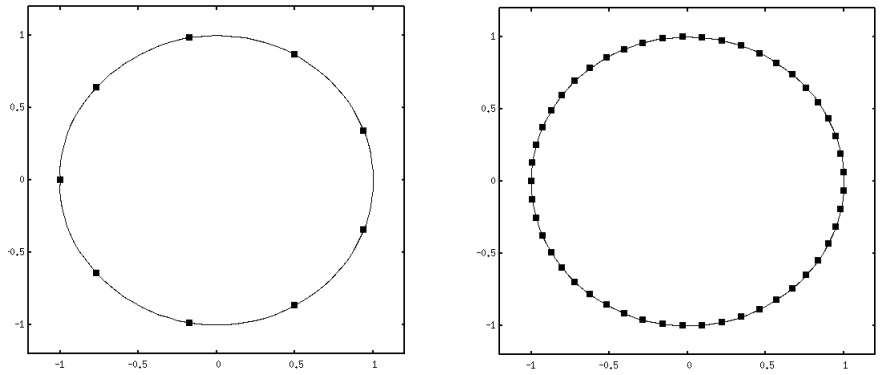
Future work includes a more sensitive and dynamic preprocessor for the grid generator. At its present stage, the preprocessor has the rudimentary architecture in place to sense the general

shape of geometries and adjusts the initial mesh to fit. However, a better and more detailed set of conditions to orient the entire overlay mesh according to the curvature of geometries would be helpful. Another helpful addition is to extend the current scheme to 3 dimensions. The structure of the mesh generator is such that adding a third dimension is a feasible extension. A similar approach to the process implemented in this research is used in [13]. However, the complexities of surface and volume intersections in the buffer layer stage makes it a considerable undertaking and adds complexity to the overlay requirements as well.

APPENDIX A

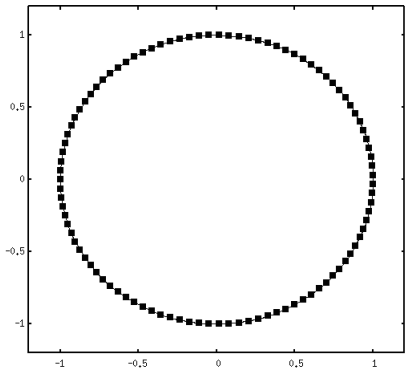
GEOMETRIES FOR NURBS CURVE FITTING

The geometries used in Section 3.1.2 are listed in Figures A.1 - A.7.

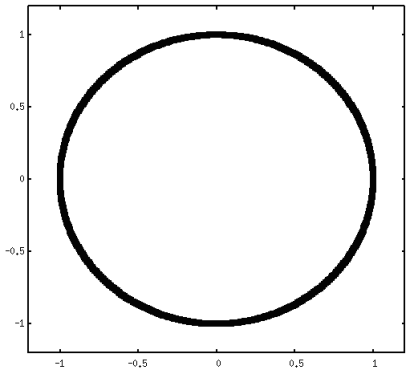


(a) 10 Points

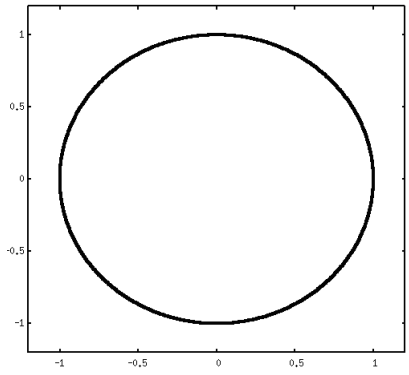
(b) 50 Points



(c) 100 Points



(d) 500 Points



(e) 1000 Points

Figure A.1 Circles with varying numbers of points

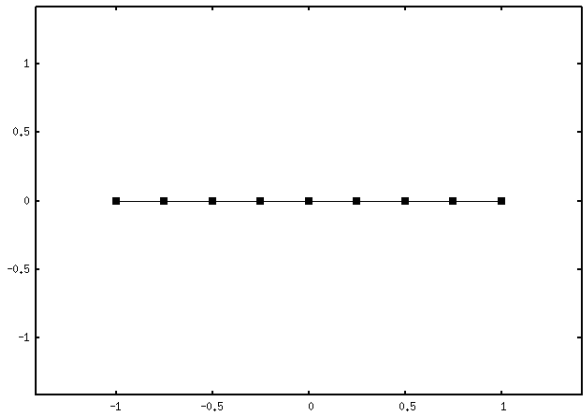


Figure A.2 Straight line

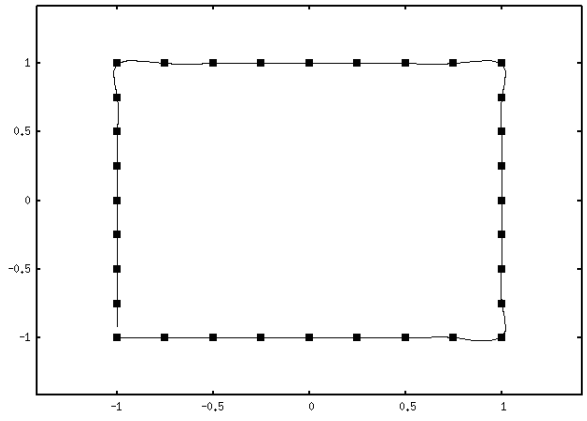


Figure A.3 Square

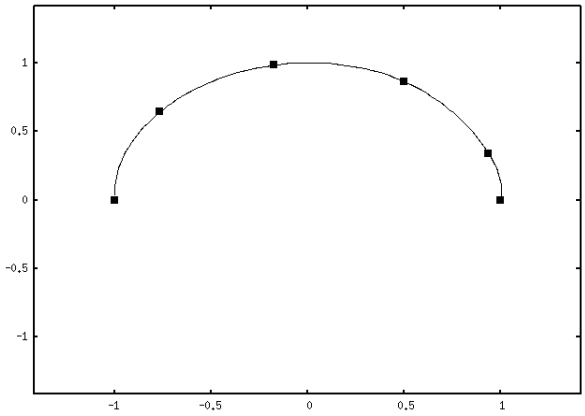


Figure A.4 Semicircle

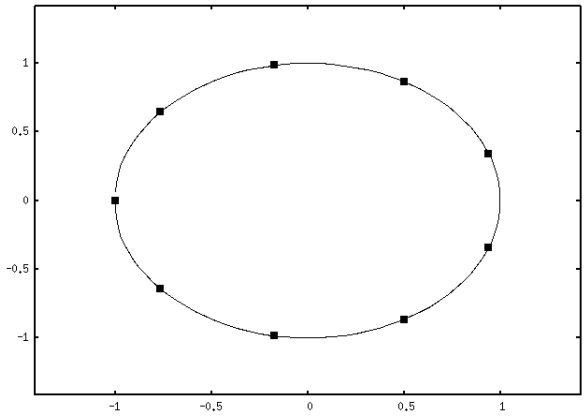


Figure A.5 Circle

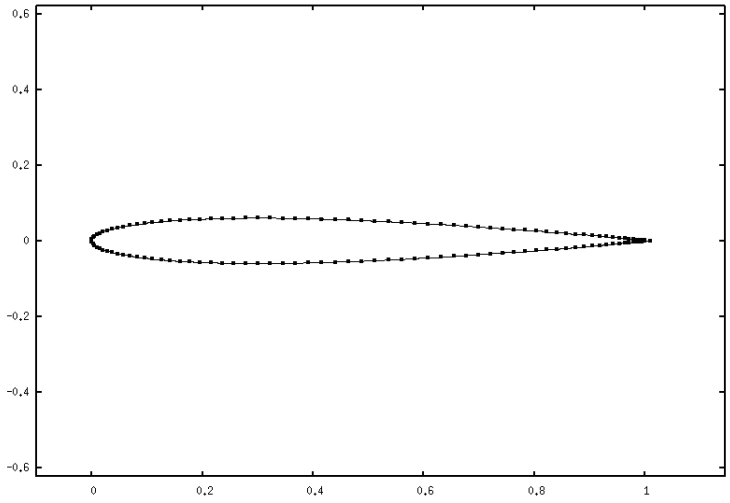


Figure A.6 NACA 0012 Airfoil

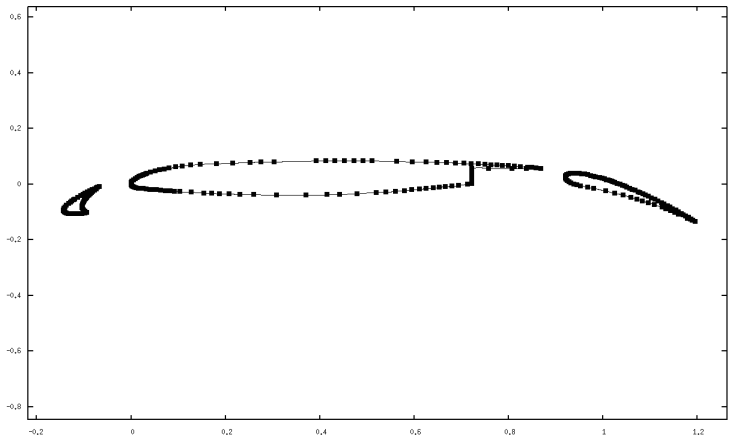


Figure A.7 30P30N Airfoil

APPENDIX B

RUNTIME ANALYSIS

The runtime study in Section 3.4 showed the general effect upon the runtime of the mesh generator of the grid size. However, the effect of the geometry size was not discussed.

For clarity, the size that is being considered as the defining aspect of a geometry is the total number of points used to define boundaries, and not the extent of the domain. The three geometries that were used were three unit circles with varying numbers of points used to describe each one.

The exact values for each run time are shown in Tables B.1 - B.3.

Table B.1 Run Times for Small Circle Geometry

Number of Grid Nodes	Time in Seconds
36	5.99999E-003
108	1.09999E-002
360	3.59999E-002
1284	0.14099
4844	0.60299
29080	4.20800
114736	20.35200

Table B.2 Run Times for Medium Circle Geometry

Number of Grid Nodes	Time in Seconds
37	6.00000E-003
116	2.09999E-002
392	7.70000E-002
1420	0.29200
5340	1.17700
32222	7.79100
127270	34.96900

Table B.3 Run Times for Large Circle Geometry

Number of Grid Nodes	Time in Seconds
37	9.99999E-003
116	3.19999E-002
392	0.11400
1420	0.43199
5340	1.71400
22577	7.60199
127270	47.93000

The number of points used for each geometry file are listed in Table B.4.

Table B.4 Number of Nodes for Geometry File

Geometry	Number of Nodes
Smallest Circle	10
Middle Circle	250
Largest Circle	500

The amount of time that the algorithm uses for three main steps is also considered. The three sections considered are the initialization stage, which includes NURBS curve fitting and initialization of the overlay mesh, adding in the geometry to the overlay mesh and creating the final connectivity, and finally, smoothing the final mesh. A plot showing the distribution of time for 6 different mesh cases for each geometry is shown in Figure B.1. The cases are chosen to be as closely corresponding in resulting size of mesh as possible.

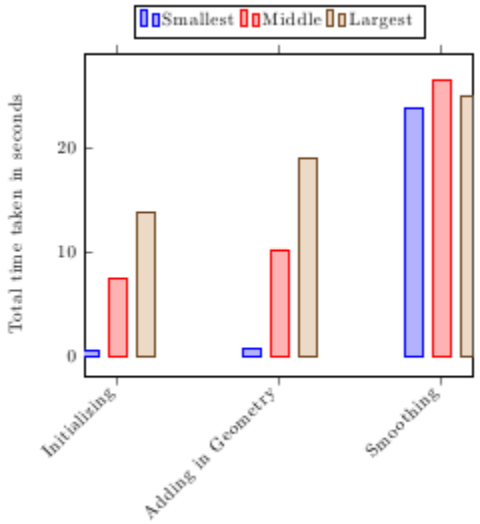


Figure B.1 Time distribution of mesh generating algorithm

Despite the fact that the first two steps in the grid generating procedure scale with the size of the geometry, the overall contribution to the runtime is still less than the last step even for very large geometry file. The majority of the runtime is allotted to the smoothing algorithm, and therefore in order to speed up the scheme most efficiently, a more time efficient smoothing algorithm should be implemented.

REFERENCES

[1] Lee, Y. and Lee, C. K., "A new Indirect Anisotropic Quadrilateral Mesh Generation Scheme with Enhanced Local Mesh Smoothing Procedures," *International journal for Numerical Methods in Engineering*, Vol. 58, No. 2, 2003, pp. 277–300.

[2] Remacle, J.-F., Lambrechts, J., Seny, B., Marchandise, E., Johnen, A., and Geuzainet, C., "Blossom-Quad: a Non-Uniform Quadrilateral Mesh Generator using a Minimum-Cost Perfect-Matching Algorithm," *International Journal for Numerical Methods in Engineering*, Vol. 89, No. 9, 2012, pp. 1102–1119.

[3] Lee, K.-Y., Kim, I.-I., Cho, D.-Y., and wan Kim, T., "An Algorithm for Automatic 2D Quadrilateral Mesh Generation with Line Constraints," *Computer-Aided Design*, Vol. 35, No. 12, 2003, pp. 1055 – 1068.

[4] de Oliveira Miranda, A. and Martha, L., "Quadrilateral Mesh Generation Using Hierarchical Templates," *Proceedings of the 21st International Meshing Roundtable*, edited by X. Jiao and J.-C. Weill, Springer Berlin Heidelberg, 2013, pp. 279–296.

[5] Yerry, M., Shephard, M. S., et al., "Modified Quadtree Approach to Finite Element Mesh Generation." *IEEE Comp. Graphics & Applic.*, Vol. 3, No. 1, 1983, pp. 39–46.

[6] Druyor, C. T., *An Adaptive Hybrid Mesh Generation Method for Complex Geometries*, Master's thesis, University of Tennessee at Chattanooga, 2011.

[7] Cui, J., *Body-fitting Meshes for the Discontinuous Galerkin Method*, Ph.D. thesis, TU Darmstadt, 2013.

[8] Piegl, L. and Tiller, W., *The NURBS Book*, Springer, Berlin, 1995.

[9] Gavrilova, M. and Rokne, J. G., "Reliable Line Segment Intersection Testing," *Computer-Aided Design*, Vol. 32, No. 12, 2000, pp. 737–745.

[10] Trefethen, L. N., *Spectral Methods in MATLAB*, Siam, 2000.

[11] Balanis, C. A., *Advanced Engineering Electromagnetics*, John Wiley & Sons, Ltd., 1979.

[12] Young, P., "Leapfrog Method and Other 'Symplectic' Algorithms for Integrating Newton's Laws of Motion," University Lecture, 2011.

[13] Karman Jr, S. L., "Hierarchical Unstructured Mesh Generation," *AIAA Aerospace Sciences Meeting and Exhibit*, Vol. 613, University of Tennessee at Chattanooga, American Institute of Aeronautics and Astronautics, 2004.

VITA

Mary Barker was born and grew up in St. Louis, Missouri. She earned a B.A. in Mathematics from Covenant College on Lookout Mountain, Georgia and moved to Chattanooga to pursue a degree in Computational Engineering at the SimCenter, University of Tennessee at Chattanooga which she completed in August, 2015. She loves mathematics and hopes to continue as a student until she cannot learn any more.