University of Reading

Department of Mathematics

Final Year Project

Mesh Generation and its application to Finite Element Methods

Author:
Mary Pham

Supervisor:
Dr. Stephen Langdon

A dissertation submitted in partial fulfilment of the requirement for the degree of Masters of Scientific and Industrial Computations (MOSAIC)

August 24, 2009

Abstract

The objective of this dissertation is to investigate the effects on the solution of the modified Helmholtz equation, via the finite element method, of uniform and non-uniform meshes.

The modified Helmholtz equation is solved analytically on a simple two-dimensional geometry and compared to numerical solutions with both mesh schemes. Errors between the analytic and numerical solutions are then analysed before further investigations are made into the behaviour of solutions with other mesh schemes.

Acknowledgements

I would like to acknowledge Dr. Stephen Langdon for his supervision, unending knowledge of this subject, time and patience during the course of this dissertation. Without his encouragement and constant guidance, I could not have finished this dissertation. He was always there to meet and talk about my ideas, to proof read and mark up my paper and chapters, and to ask me good questions to help me think through my problems. I am also greatly indebted to many academic staff in the Department of Mathematics at the University of Reading, and in particular Professor Mike Baines, who, without his generosity, time, support and our many chats, this MSC would of been more difficult. I would also like to thank Jason Pinto and Peter Spence for their help and support throughout this year, in addition to my colleagues on the MSC course. Finally I would like to thank my family and friends for their love, for their unconditional support and encouragement to pursue my interests.

Declaration

Contents

Chapter 1

Introduction

1.1 Aim

In this thesis we use the finite element method for solving the Modified Helmholtz equation in a two dimensional domain. Mesh generators used in conjunction with the finite element method do not always produce good quality meshes[1]. Within the mesh generation process, ill shaped elements can be created, and this can lead to significant solution errors, particularly in regions where there is rapid solution change. Generally, if the elements in such regions are too big, and the order of the finite element method too low, the deviation in calculated solution from the true solution will be large.

If there is a limitation on computational resources, then depending upon the solution sought, it may not be judicious to utilise those resources using a uniform mesh across the entire solution domain. However, if a method of producing non-uniform meshes can be found, the computing resources can be better used by generating a dense mesh in regions of rapid solutions change, and a less dense mesh otherwise. Furthermore, in this case, a more accurate solution (than in the uniform mesh case) can be found with less resources.

In the problem considered in this dissertation, we are only concerned with a simple disc-like region. On this region (with the appropriate boundary conditions), the modified Helmholtz equation generates solutions that change rapidly towards the boundary, and as such, we are looking to find an adaptive mesh generation method that utilises this fact.

[1]A good quality triangular mesh, for example, would have all (or as near as) equilateral triangular elements (see [2]).

1.2 Background

The Helmholtz equation ([8]) is an elliptic partial differential equation of the form

$$\nabla^2 u + \alpha u = 0, \tag{1.1}$$

where $u = u(\mathbf{x})$ and α is a constant. [**?**] This equation arises in many areas of mathematics, and for example, is readily seen to arise in the construction of a separation solution to the wave equation

$$\nabla^2 u = \frac{\partial^2 u}{\partial t^2}, \tag{1.2}$$

where it is assumed that the function $u(x, y)$ can be written

$$u(\mathbf{x}, t) = \hat{U}(\mathbf{x})\hat{T}(t). \tag{1.3}$$

Applying the form (1.3) to (1.2) we have

$$\nabla^2 \left(\hat{U}\hat{T}\right) - \frac{\partial^2}{\partial t^2}\left(\hat{U}\hat{T}\right) = 0, \tag{1.4}$$

which on expansion gives

$$\nabla \cdot \left(\hat{T}\nabla\hat{U} + \hat{U}\nabla\hat{T}\right) - \frac{\partial}{\partial t}\left(\hat{T}\frac{\partial \hat{U}}{\partial t} + \hat{U}\frac{\partial \hat{T}}{\partial t}\right) = 0, \tag{1.5}$$

or

$$\frac{1}{\hat{U}}\nabla^2\hat{U} = \frac{1}{\hat{T}}\frac{d^2\hat{T}}{dt^2}. \tag{1.6}$$

Using the fact that both sides of (1.6) are independent of each other and so must be constant in order to be equal for all $\mathbf{x}$ and t, we obtain the one and two-dimensional Helmholtz equations

$$\begin{aligned} \frac{d^2\hat{T}}{dt^2} - \kappa^2\hat{T} = 0 \qquad \text{and,} \\ \nabla^2\hat{U} - \kappa^2\hat{U} = 0, \end{aligned} \tag{1.7}$$

where κ is the separation constant.

The Helmholtz equation considered in this thesis is of the form

$$-\nabla^2 u + K^2 u = g, \qquad \text{in} \quad \Omega \tag{1.8}$$

where Ω is some domain and the function $u = f$ on the domain boundary $\partial\Omega$.

1.3 Solution Approach

In Chapter 2 (which summarises [2]), we begin by using the Mesh Generator (described in [2]) to discretise a circular solution domain in a number of ways, creating a number of meshes upon which (1.8) can be solved. In Chapter 3 we solve (1.8) on the same circular domain (in radial coordinates) using the separation of variables method to reduce it to a pair of ODEs from which the analytic solution is fairly easily generated. In Chapter 4 we solve (1.8) numerically by applying the finite element method to meshes already generated in Chapter 2. Once confidence is established that the numerical method works for a uniformly graded mesh, we then modify our mesh generation algorithm and introduce a method for adaptively controlling the mesh size. This method is then used to generate non-uniform meshes on the same domain (Chapter 7), and the solutions generated further explored by comparing them to the analytic solution already determined.

Chapter 2

A Simple Mesh Generator in Matlab

The process of discretising regions or creating meshes is used in many areas of mathematics such as in scientific computing and the production of computer graphics. Mesh generators are often considered to be black boxes, where we input some variables and the generator produces an output, but we do not have any knowledge of its internal working. However, for the purposes of this dissertation, we use a commercially available mesh generator, where the source code is freely accessible, and details of this are given in [2]. This enables us to understand and modify, if required, the mesh generation process and to incorporate such a mesh generator in other computer codes.

As part of the mesh generation process, a region must be broken up into a set of tesselated subregions, the vertices of which have distinct nodal points. The critical features of the mesh generation process are therefore the identification of the nodal point locations, and of the individual subregions that comprise the full region of interest.
The process by which the mesh generator used in this dissertation creates the triangles from the nodes is called Delaunay triangulation [2]. The process by which the nodes are located are described in more detail in the following sections.

'The mesh generator code'

The particular code detailed in [2], that we will be using in this dissertation, identifies whether a point lie either inside or outside the region of interest. This is done by using a distance function $d(x, y)$ that is either negative or positive, respectively. The distance function in this project returns the signed distance function from each node position. This helps to determine if we need to reject the node if it is outside of our region (when $d(x, y) > 0$) or move the points back to the closest boundary point after each iteration.

In this generator there are three important steps:

1. Move nodes according to the force relationship.
2. Retriangulate the bars after applying force to the points (using Delaunay Algorithm).
3. Keep iterating and moving the points until the movement of the nodes is smaller than the tolerance, (where tolerance is the error term).

'Mesh Generator' Triangulation Method

The triangulation process for a given set of nodal points P, joins the nodes in such a way that most of the triangles produced are almost equilateral. By retriangulating skinny triangles this leads to more accurate results. It does this by re-triangulating bad triangles (skinny triangles as shown in Figure(2.1)). If necessary, the nodes within the region are moved until such a condition is met. Triangultion is carried out using the Delaunay algorithm.

Any set of points can be triangulated by using the Delaunay algorithm which considers the edges of the triangular subregions as rigid bars and the triangle vertices as movable joints joining the bars.

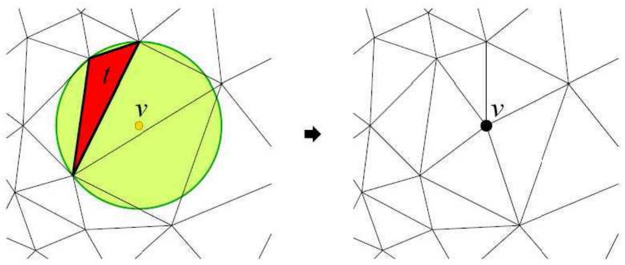


Figure 2.1: Delaunay triangulation-Retriangulating Skinny triangle t [4]

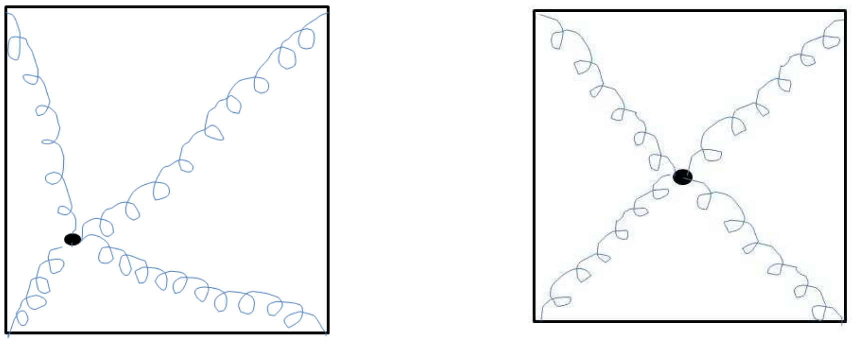
2.1 Algorithm

To summarise, the mesh generator begins by generating a uniform mesh on a bounding box around our shape of interest, rejecting nodes that lie outside the shape. It then attempts to move the nodes around in an iterative fashion by the force, retriangulating the nodes until the nodes are moved by a distance less than a specific tolerance. Each of the bars in the mesh has a force displacement relationship F(P), where F(P) depends on the internal and external forces (internal forces come from the bar and external from the boundary).

F(P) is the vector relating the forces between all of the nodes, this is made up of $\mathrm{f}(l, l_0)$ where $\mathrm{f}(l, l_0)$ is the force between any two nodes. There are many choices for the force displacement function $\mathrm{f}(l, l_0)$ in each bar, the one used in [2] models the ordinary linear spring so shown below.

$$\mathrm{f}(l, l_0) = \begin{cases} k(l_0 - l) & \text{if } l < l_0 \\ 0 & \text{if } l \geq l_0. \end{cases} \tag{2.1}$$

(k =constant, l=current length and l_0=initial length). We want the force $\mathrm{f}(l,l_0)$ function to give repulsive forces hence $\mathrm{f}(l, l_0)$ >0, the reason being that this will help the nodes spread out over the region giving more accurate results and produce better quality meshes. Also for nodes which do not move around and stay fixed in one position, $\mathrm{f}(l,l_0)$=0. This makes sense since fixed nodes are not allowed to move at each iteration implying that initial length and current length are of the same length.



(a) Starting point (b) Steady state

Figure 2.2: Suppose we have a point placed inside a square box with four springs of equal lengths attached to it, some stretched more than others. There will be forces acting on the point moving it around until it reaches it steady state, i.e. its equilibruim position.

F(P) is a function which has to be solved for equilibrium mesh points. This is a difficult problem to solve because of the discontinuity in the force function F(P) due to the fact that the structure is changed by Delaunay as the points move due to the forces and also the external reaction forces at the boundaries keeping the nodes inside the region.
In order to try and get an equation for this force function, we introduce an artificial time-dependence for P , the locations of the points, with P(0)=P_0 and t$\geq t_0$. This gives the system of Ordinary Differential Equations

$$\frac{dP}{dt} = F(P) \quad \text{t} \geq 0. \tag{2.2}$$

To find stationary solutions (when F(P)=0) the system can then be approximated using Forward Euler, (other methods can also be used however this is the one mentioned in the paper[2]). The main advantage of using this method is that it is simple:

$$P_{n+1} = P_n + \Delta t F(P_n). \tag{2.3}$$

where $P_n \approx P(t_n)$ and t_n=nΔt. If any of the points move outside the region after the update, they are moved back to the closest boundary point using the distance function implying that the points can only move along the boundary but not go outside.
For uniform meshes l_0=constant (defined in (2.1)) but sometimes it is better to have regions of different size since, in some cases the solution method may require small elements close to a

singularity to give good global accuracy in which case a high resolution is required. In some cases

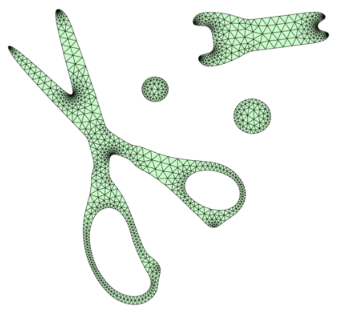


Figure 2.3: An example of a nonuniform mesh- A high resolution (non-uniformity) is required to capture this discontinuity, this is done by placing extra nodes near singularity-([2]).

the desired edge length provided by the user h(x,y) does not equal the actual size. For example if h(x,y)=1+x in a unit square, the edge length close to x=0 will be about half the length of the edge close to x=1. So to find the scaling factor between the actual and desired mesh size we compute the ratio between the mesh area from the actual edge lengths l and 'desired size'.

$$\text{Scaling Factor} = (\sum \frac{l_i^2}{h(x_i, y_i)^2})^{\frac{1}{2}}. \tag{2.4}$$

The scaling factor is used in the implementation to give the desired lengths l from (2.1) from which we can calculate the force and use this to update the node position.

2.2 Implementation

The complete source code for the mesh generator can be found in [2], however this section gives a brief overview of how the meshes are produced, illustrated in Figure (2.2)

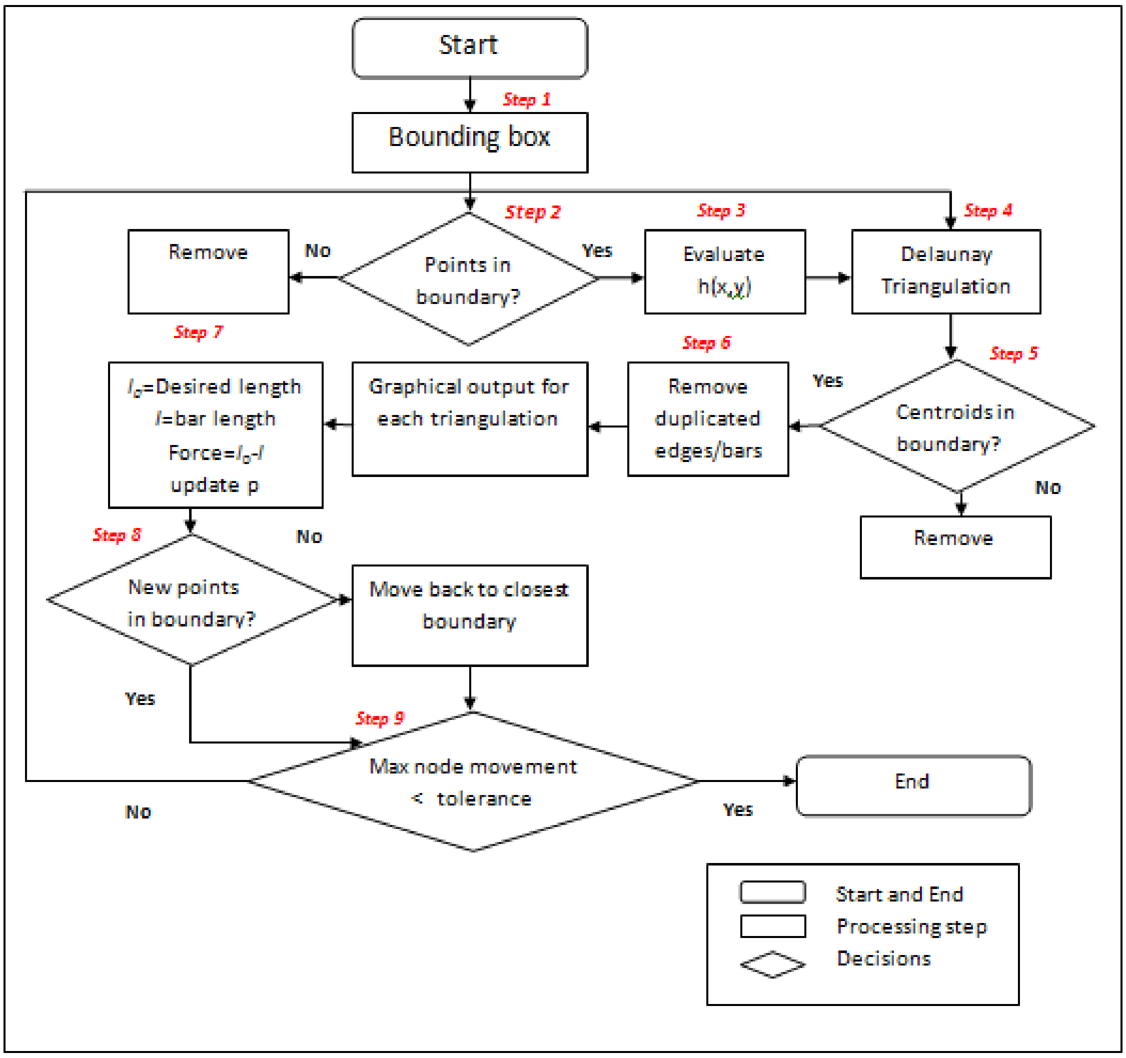


Figure 2.4: A simple flow chart representing how the mesh works

The mesh function produces the following outputs:

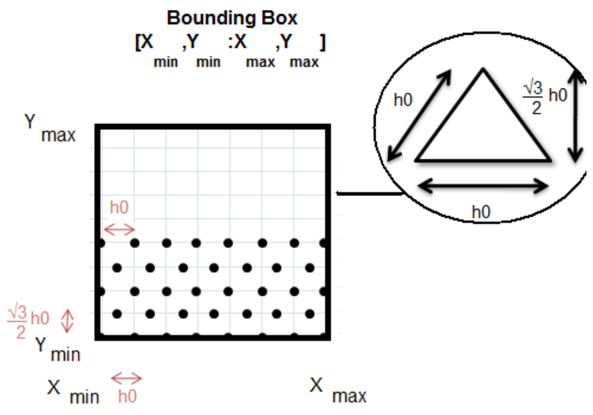
- node position p.
- triangle indices t.

Input arguments are as follow:

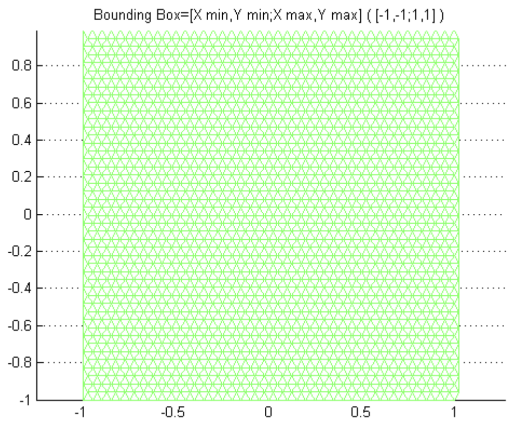
- fd- Distance function which returns all the signed distances from each node to the closest boundary, this helps decide if the nodes are inside or outside the boundary.
- fh-desired edge length function which returns all h=h(x,y) for all input points (h=h(x,y)=maximum size length, constants for uniform meshes, otherwise functions for non uniform meshes).
- h0-distance between points in the initial distribution $p0$ (In this thesis h_0=h).
- bbox-bounding box=[x_{min},y_{min}:x_{max},y_{max}] places a box over our shape so we can calculate the signed distances from the nodal points to the boundary (See Figure(2.5(a)) and (2.5(b)) Step 1)
- pfix-fixed node position, for example if we want to look at a unit square with a hole we want to fix the four corners of the square so it does not move.

Step 1

The first step in the program creates a bounding box with a uniform distribution of nodes, corresponding to equilateral triangles. The domain that we seek is a mesh which must lie inside the bounding box.



(a) $\frac{\sqrt{3}h0}{2}$ in the y direction, shifting all even rows $\frac{h0}{2}$ to the right, all the points distance h0 from their closest neighbors



(b) Bounding box

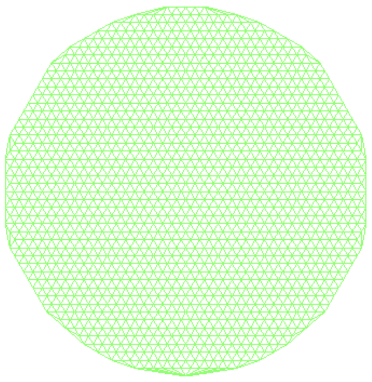
Figure 2.5: Bounding Box=[x_{min},y_{min}:x_{max},y_{max}] the one used is [-1,-1;1,1]

Step 2

Remove all the points outside the desired geometry (d(x,y)>0 where d(x,y) is the distance to the boundary).

Step 3

Evaluate h(x,y) (from the fh function) at each node and keeping the points with a probability proportional to $\frac{1}{h(x,y)^2}$, giving us a much finer grid. (Explained in Chapter 6). If we reject too many points there will be very few nodes in our domain to triangulate using Delaunay hence the maximum node movement is never smaller than the tolerance causing non termination of the algorithm.


(a) Uniform triangulation

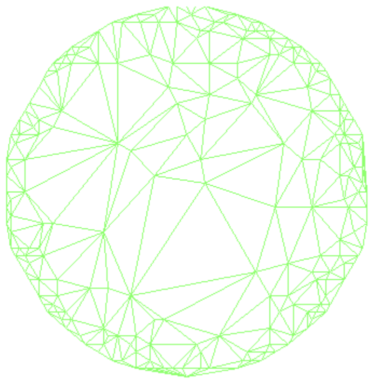

(b) Non uniform triangulation

Figure 2.6: Figure(2.6(a)) shows a uniform mesh and Figure(2.6(b)) a non uniform mesh rejecting the exterior points and some interior ones

Step 4

Now the code enters the main loop, Delaunay triangulation triangulates the nodes.

Step 5

If the centre of the triangle is outside the region it has to be removed.

Step 6

Since most edges will appear twice, duplicates have to be removed.

Step 7

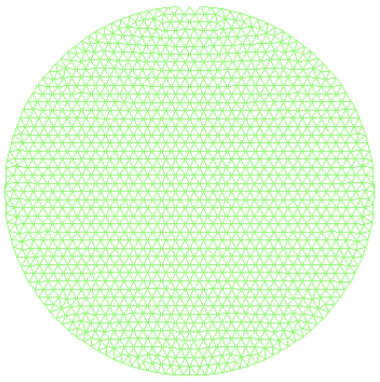
Calculate the f(l, l_0)=$l_0 - l$ (where f(l, l_0)>0 to give repulsive forces) and update using (2.3) (f(l, l_0)=0 for fixed nodes).

Step 8

Is the new node outside the boundary? If so, then it is moved back to the closest boundary (using the distance function).

Step 9

If the maximum node movement in the final iteration is smaller than the tolerance then we terminate the iteration, if not we go back to the Delaunay triangulation and repeat the whole algorithm in an iterative fashion until the maximum node movement is smaller than the tolerance.


(a) Uniform Mesh where the maximum node movement is smaller than the tolerance

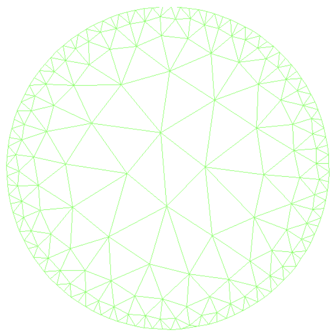

(b) Non-uniform mesh where the maximum node movement is smaller than the tolerance

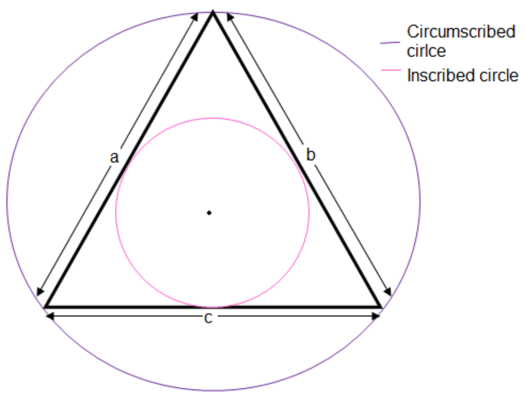
Figure 2.7: After applying the Delaunay Algorithm to Figure (2.6(a)) we get (2.7(a)) and to Figure (2.6(b)) we get (2.7(b)) where Figures (2.7(a)) and (2.7(b)) are better because maximum node movement in the final iteration is smaller than the tolerance hence giving the optimal mesh.

2.3 Quality of the meshes produced by the ‘Mesh Generator’

One way of measuring the quality of the mesh is calculating the ratio between the radii of the largest inscribed circle (times two) and the smallest circumscribed circle:

$$q = 2\frac{r_{in}}{r_{out}} = \frac{(b+c-a)(c+a-b)(a+b-c)}{\text{abc}} \tag{2.5}$$

Where a, b, c are side lengths.The mesh generator used in this thesis produces triangles that are almost equilateral which implies that q=1, giving good quality meshes where q>0.5 and also good uniformity. This is advantageous because all the triangles will of a similar shape, and most



of the angles will be close to 60^o giving good numerical results.

2.4 Future Improvement

Advantages

- Code is short and simple with full documention [2] provided explaining how it works.
- Produce good distances to the boundary which in turn produces high quality meshes.

Disadvantages

- Slow execution and sometimes the possibility of non execution (because the maxmimum node movement is never smaller than the tolerance).
- Sometimes it does not produce very good meshes especially when we reject too many points and our region becomes unstable.

In Chapter 8 we will look at how changing the tolerance will affect the number of points in the mesh or how big or small the errors are and whether it will give us a good solution.

Chapter 3

Analytical Solution to The Modified Helmholtz Equation

This chapter will look at solving the Modified Helmholtz equation with both a Neumann and Dirichlet boundary conditions establishing the exact solution on a circle with radius 1.

Taking the modified Helmholtz equation to be of the form

$$-\nabla^2 u + K^2 u = 0 \tag{3.1}$$

where K=constant, we seek $u(r,\theta)$ on a circular domain of radius 1. In order to establish the solution we assume that $u(r,\theta)$ can be written in the form

$$u = R(r)\Theta(\theta), \tag{3.2}$$

and begin by substituting (3.2) into (3.1) by writing the operator ∇^2 in polar coordinates as

$$\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}. \tag{3.3}$$

Calculating the derivatives

$$\frac{\partial^2 u}{\partial r^2} = \frac{\partial^2 R}{\partial r^2}\Theta = R''\Theta \tag{3.4}$$

$$\frac{1}{r}\frac{\partial u}{\partial r} = \frac{1}{r}\frac{\partial R}{\partial r}\Theta$$

$$= \frac{R'\Theta}{r} \tag{3.5}$$

$$\frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = \frac{1}{r^2}\frac{\partial^2 \Theta}{\partial \theta^2}$$

$$= \frac{\Theta'' R}{r^2}, \tag{3.6}$$

we obtain by substitution into (3.3)

$$-R''\Theta - \frac{R'\Theta}{r} - \frac{\Theta'' R}{r^2} + K^2 R\Theta = 0. \tag{3.7}$$

Dividing both sides of (3.7) by Θ and R, and rearranging, we obtain

$$\frac{-r^2 R''}{R} - \frac{rR'}{R} + K^2 r^2 = \frac{\Theta''}{\Theta} = C, \tag{3.8}$$

where C is the separation constant. As in §1.2, since both sides are independent of one another, they must both be constant giving rise to the two ODEs

$$r^2\frac{R''}{R} + \frac{rR'}{R} - (K^2 r^2 + C^2) = 0, \quad \text{and,} \tag{3.9}$$

$$\Theta'' - C\Theta = 0. \tag{3.10}$$

where Θ is 2π periodic.

Considering (3.10) we have three possible cases for the separation constant C, and these are:

$\underline{C = 0}$:

In this case $\Theta{=}A\Theta + B$, and we must choose $A = 0$ and hence $\Theta = B$ to ensure a 2π periodic solution.

$\underline{C > 0}$:

In this case, since $\Theta = A\sinh(\theta) + B\cosh(\theta)$, the solution must be disregarded (i.e. $C \not> 0$) as the hyperbolic functions *sinh* and *cosh* are not 2π periodic.

$\underline{\text{C=-}\alpha^2 < 0}$:

In this case, the solution is the periodic function $\Theta{=}A\cos(\alpha\theta){+}B\sin(\alpha\Theta)$. In order to establish 2π periodicity, we must have

$$\sin(\alpha(\theta + 2\pi)) = \sin(\alpha\theta)$$

or

$$\sin(\alpha\theta)\cos(\alpha 2\pi) + \cos(\alpha\theta)\sin(\alpha 2\pi) = \sin(\alpha\theta).$$

For this to be so, we conclude that $\alpha = n$, where $n \in \mathbb{Z}$.

Writing $C = -n^2$ and defining a new variable $\tilde{r} = Kr$ allows (3.9) to be written as the modified Bessel equation

$$\tilde{r^2}\tilde{R}'' + \tilde{r}\tilde{R}' - (\tilde{r}^2 + n^2)\tilde{R} = 0. \tag{3.11}$$

This equation has the two modified Bessel function solutions (see [7])

$$R = I_{\pm n}(\tilde{r}), \quad \text{and,} \tag{3.12}$$

$$R = K_n(\tilde{r}). \tag{3.13}$$

The limiting form of $K_n(\tilde{r})$ for small arguments is of the form (referring to [7])

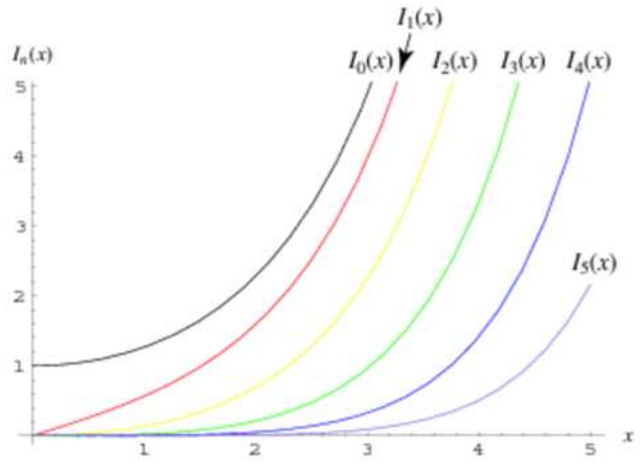


Figure 3.1: The variation in I_{+nx} for n=1,2...,5 (see [6]).

$$K_n(\tilde{r}) \sim \ln\tilde{r}, \qquad n = 0,$$
$$K_n(\tilde{r}) \sim \tfrac{1}{2}\Gamma(n)(\tfrac{1}{2}\tilde{r})^{-n}, \quad n \neq 0.$$

When $r = 0$, $K_n(r)$ becomes infinite in both cases and so cannot represent the solution to (3.1). This implies that $R(r)$ must take form $R(r) = I_{\pm n}(Kr)$ so that the solution to (3.1) for a single mode n must be

$$u_n(r,\theta) = I_{\pm n}(Kr)[A\cos(n\theta) + B\sin(n\theta)]. \tag{3.14}$$

Since (3.1) is linear, all such modes (3.14) are solutions, and so the general solution is

$$u(r,\theta) = \sum_{n=0}^{\infty} I_n(Kr)[A_n \cos(n\theta) + B_n \sin(n\theta)], \tag{3.15}$$

where the constants A_n and B_n are determined by the applied boundary conditions.

3.1 Dirichlet Boundary Condition

We now consider the Dirichlet boundary condition

$$u(1,\theta) = 1. \tag{3.16}$$

Substituting (3.16) into (3.15) we obtain

$$u(1,\theta) = \sum_{n=0}^{\infty} I_n(K)(A_n \cos(n\theta) + B_n \sin(n\theta)) = 1 \tag{3.17}$$

To determine the constants A_n and B_n we proceed in a standard manner by pre multiplying (3.17) by $\cos(m\theta)$ and integrating over a 2π period giving

$$\int_0^{2\pi} \cos(m\theta)d\theta = \sum_{n=0}^{\infty} I_n(K)\left[A_n \int_0^{2\pi} \cos(n\theta)\cos(m\theta)d\theta + B_n \int_0^{2\pi} \sin(n\theta)\cos(m\theta)d\theta\right]. \tag{3.18}$$

Calculating each of the terms on the RHS of (3.18) we have

- $$\int_0^{2\pi} \cos(m\theta)d\theta = \begin{cases} 0 & \text{for all m}\neq 0 \\ 2\pi & \text{m=0} \end{cases}$$

- $$\int_0^{2\pi} \cos(m\theta)\cos(n\theta)d\theta = \begin{cases} 0 & \text{if n+m and n-m both } \neq 0 \\ \pi & \text{if n=-m or n=m but not both} \\ 2\pi & \text{n=-m and n=m so n=m=0} \end{cases}$$

- $$\int_0^{2\pi} \cos(m\theta)\sin(n\theta)d\theta = \begin{cases} 0 & \text{if n-m}\neq 0 \text{ and m+n}\neq 0 \\ 0 & \text{if n-m=0 or n+m=0} \end{cases}$$

When these coefficients are substituted into (3.18) they must satisfy the following:

For <u>m=0</u>:

$$2\pi = \sum_{n=0}^{\infty} I_n(K)\left[A_n \begin{cases} 2\pi & \text{if n=0} \\ 0 & \text{if n}\neq 0 \end{cases} + B_n(0)\right]$$

$$A_0 = \frac{1}{I_0(K)}. \tag{3.19}$$

For <u>m≠0</u>:

$$0 = \sum_{n=1}^{\infty} I_n(K) \left[A_n \begin{cases} \pi & \text{if n=m or n=-m} \\ 0 & \text{otherwise} \end{cases} + B_n(0) \right]$$

implying

$$0 = \pi A_m I_m(K)$$

So, either A_m=0 or I_m(K)=0, I_m(K)=0 is only true if K=0 however this is a contradiction because K≠0 because K cannot equal to zero, and so $I_n(K)$ ≠0 implying

$$A_m = 0. \tag{3.20}$$

for all m≠0.

We proceed in the same manner to find B_m, the typical sine coefficient: Return to (3.17), multiply both sides by $\sin(m\theta)$ and integrate giving us the following:

$$\int_0^{2\pi} \sin(m\theta) d\theta = \sum_{n=0}^{\infty} I_n(K) \left[A_n \int_0^{2\pi} \cos(n\theta)\sin(m\theta) d\theta + B_n \int_0^{2\pi} \sin(n\theta)\sin(m\theta) d\theta \right]. \tag{3.21}$$

Each of the term is calculated as follows

- $$\int_0^{2\pi} \sin(m\theta) d\theta = \begin{cases} 0 & \text{for all m}\neq 0 \\ 0 & \text{m=0} \end{cases}$$
- $$\int_0^{2\pi} \cos(n\theta)\sin(m\theta) d\theta = 0$$
- $$\int_0^{2\pi} \sin(m\theta)\sin(n\theta) d\theta = \begin{cases} \pi & \text{if n=m or n=-m} \\ 0 & \text{otherwise} \end{cases}$$

When these coefficients are substituted into (3.21) they must satisfy the following:

For <u>m=0</u>:

$$0 = I_0(K)[\pi B_0] \tag{3.22}$$

This implies that either I_0=0 or B_0=0, however I_0 is a constant and cannot equal zero which means that B_0 must equal zero.

$$B_0 = 0. \tag{3.23}$$

For <u>m≠0</u>:

$$B_m = 0. \tag{3.24}$$

Finally we go back to the equation (3.15) and substitute in the coefficients to get the following solution

$$u(r,\theta) = A_0 I_0(Kr) = \frac{I_0(Kr)}{I_0(K)}. \tag{3.25}$$

3.2 Neumann boundary condition

We now consider the Neumann boundary condition

$$\frac{\partial u}{\partial r}(1,\theta) = 1 \quad \text{on } \Gamma \quad \text{(r=1)} \tag{3.26}$$

Differentiating (3.15), we get the following

$$\sum_{n=1}^{\infty} K I_n'(K)[A_n \cos(n\theta) + B_n \sin(n\theta)] = 1 \tag{3.27}$$

Using the recurrence relation from [7] (shown below)

$$I_n'(K) = \frac{1}{2}(I_{n-1}(K) + I_{n+1}(K)) \tag{3.28}$$

Substitue into (3.27) we get the following

$$\frac{\partial u}{\partial r}(1,\theta) = 1 = \sum_{n=1}^{\infty} \frac{K}{2}\left(I_{n-1}(K) + I_{n+1}(K)\right)[A_n \cos(n\theta) + B_n \sin(n\theta)]. \tag{3.29}$$

To determine the constants A_n and B_n we proceed in a standard manner by pre multiplying (3.29) by $\cos(m\theta)$ and integrating over a 2π period giving

$$\int_0^{2\pi} \cos(m\theta)d\theta = \sum_{n=0}^{\infty} \frac{K}{2}(I_{n-1}(K)+I_{n+1}(K)) \left[A_n \int_0^{2\pi} \cos(n\theta)\cos(m\theta)d\theta + B_n \int_0^{2\pi} \sin(n\theta)\cos(m\theta)d\theta\right]. \tag{3.30}$$

Calculating each of the terms on the RHS of (3.30) we have

For <u>m=0</u>:

$$2\pi = \sum_{n=0}^{\infty} \frac{K}{2}(I_{n-1}(K) + I_{n+1}(K)) \left[A_n \begin{cases} 2\pi & \text{if m=0} \\ 0 & \text{otherwise} \end{cases} + B_n(0) \right]$$

so

$$A_0 = \frac{2}{K(I_{-1}(K) + I_1(K))}$$

We can apply one of the properties of the Modified Bessel equation (from [7])

$$I_{-\alpha}(K) = I_\alpha(K) \tag{3.31}$$

implying

$$A_0 = \frac{1}{KI_1(K)}. \tag{3.32}$$

For <u>m≠0</u>:

$$A_m = 0. \tag{3.33}$$

The same method produces B_n, the typical sine coefficient: Return to (3.27), multiply both sides by $\sin(m\theta)$ and integrate giving us the following:

$$\int_0^{2\pi} \sin(m\theta)d\theta = \sum_{n=0}^{\infty} \frac{K}{2}(I_{n-1}(K)+I_{n+1}(K)) \left[A_n \int_0^{2\pi} \cos(n\theta)\sin(m\theta)d\theta + B_n \int_0^{2\pi} \sin(n\theta)\sin(m\theta)d\theta \right] \tag{3.34}$$

For <u>m=0</u>:

$$B_0 = 0. \tag{3.35}$$

For <u>m≠0</u>:

$$0 = \sum_{n=0}^{\infty} \frac{K}{2}(I_{n-1}(K) + I_{n+1}(K)) \left[A_n \times 0 + B_n \begin{cases} -\pi & \text{if n=m or n=-m} \\ 0 & \text{otherwise} \end{cases} \right]$$

$$0 = \sum_{n=0}^{\infty} \frac{K}{2}[I_{n-1}(K) + I_{n+1}(K)][B_n(-\pi)]$$

So either $[I_{n-1}(K) + I_{n+1}(K)]$=0 or B_n=0 we know that $[I_{n-1}(K) + I_{n+1}(K)]$cannot equal,(because if it does then K would have to be equal to zero, however this is a contradiction because K cannot equal to zero). Hence

$$B_m = 0. \tag{3.36}$$

Finally we go back to the equation (3.15) u(r,θ) and substitute in the coefficients to get the following solution

$$u(r,\theta) = \frac{I_0(Kr)}{KI_1(K)}. \tag{3.37}$$

Chapter 4

Numerical Solution to the Modified Helmholtz Equation

4.1 Finite Element Method

In general, there are many ways of solving Partial Differential Equations (PDEs) numerically with advantages and disadvantages. This dissertation will use a finite element method to compute the numerical solution to the Modified Helmholtz equation (3.1).

Advantages

- Can handle complex geometry.
- Neumann boundary conditions can be naturally incorporated into the Finite Element formulation.
- Can solve a lot of complicated problems, for example the Helmholtz equation.
- Can handle complex boundaries for example a polygon.

Disadvantages

- Only approximate solutions.
- Has errors.
- Sometimes non-termination for the mesh generator.

This chapter briefly explains the formulation of the finite element approximation. The finite element method is a numerical method for solving the partial differential equation. Which place piecewise polynomials in the region, where the pieces can be chosen to fit the geometry of the problem and the computer can generate the polynomials.

4.2 Finite Element Discretization

We will be considering Modified Helmholtz Equation with a Neumann boundary condition, and if we have time look at the Dirichlet boundary condition. The difference between these two boundary conditions is that with a Neumann boundary condition all the basis functions have half hats at the boundary. This is easier to calculate and can be more easier to implement than that of the Dirichlet boundary condition condition's, which have full hats at the boundary. This means that we would have to choose the basis functions such that ω=0 on the boundary. We consider the problem

$$-\nabla^2 u + K^2 u = g \quad \text{in } \Omega \text{ (domain of a circle)} \tag{4.1}$$

and the boundary condition given by

$$\frac{\partial u}{\partial n} = 1 \quad \text{in } \Gamma \text{ (on the boundary)} \tag{4.2}$$

To construct a weak form we multiply (4.1) by $\omega(x, y)$ and integrate over Ω (where $\omega(x, y)$ is a two dimensional function belonging to a set of test functions which are at least once differentiable, square integrable, piecewise once differentiable and continuous. Also u$\in$ C^2(a,b), is at least twice differentiable in the domain (a,b))

$$-\int_\Omega \omega \nabla^2 u d\Omega + K^2 \int_\Omega u\omega d\Omega = g. \tag{4.3}$$

Now we use the two-dimensional equivalent of intergration by parts, which is Green's theorem in the form

$$\int_\Omega \omega \nabla^2 u + \nabla\omega \cdot \nabla u d\Omega = \oint_\Omega \omega \nabla u \cdot \hat{n} d\Omega. \tag{4.4}$$

which we substitute into the first term of (4.3) giving

$$\oint_\Omega \omega \nabla u \cdot \hat{n} d\tau + \int_\Omega \nabla\omega \cdot \nabla u d\Omega + \int_\Omega K^2 u\omega d\Omega = (g) \int_\Omega \omega d\Omega. \tag{4.5}$$

We want the first term (boundary term) to vanish, so we need to find a function that satisfies $\frac{\partial f}{\partial r}$ =1 at r=1 and $\frac{\partial f}{\partial r}$ =0 at r=0.

The function f where f=$\frac{r^2}{2}$ satisfies the two conditions hence

Using change of variables

$$\tilde{u} = u - f = u - \frac{r^2}{2} \tag{4.6}$$

Gives us a new boundary condition

$$\frac{\partial \tilde{u}}{\partial n} = \frac{\partial u}{\partial n} - r = 0 \text{ at r=1} \tag{4.7}$$

so $\frac{\partial \tilde{u}}{\partial n} = 0$. Substituting $\tilde{u}$ into (4.1) we get the following

$$-\nabla^2 u + K^2 u = -\Delta\left(\tilde{u} + \frac{r^2}{2}\right) + K^2\left(\tilde{u} + \frac{r^2}{2}\right) = 0$$

Rewritting this in a simpler form:

$$-\nabla^2 \tilde{u} + K^2 \tilde{u} + \left(-\Delta\frac{r^2}{2} + \frac{K^2 r^2}{2}\right) = 0. \tag{4.8}$$

rewriting $\Delta\frac{r^2}{2} = \nabla^2\frac{r^2}{2}$ as a Laplacian equation in polar coordinates (3.3), we obtain

$$\begin{aligned} -\nabla^2\frac{r^2}{2} &= -\frac{\partial^2}{\partial r^2}\left[\frac{r^2}{2}\right] + \frac{1}{r}\frac{\partial}{\partial r}\left[\frac{r^2}{2}\right] \\ &= \frac{\partial}{\partial r}[r] + \frac{1}{r}[r] = 2. \end{aligned} \tag{4.9}$$

Substituting (4.9) back into (4.8) gives the following:

$$-\nabla^2 u + K^2 u = -\nabla^2 \tilde{u} + K^2 \tilde{u} + \left(-2 + \frac{K^2 r^2}{2}\right) = 0 \tag{4.10}$$

giving

$$-\Delta \tilde{u} + K^2 \tilde{u} = 2 - \frac{K^2 r^2}{2}. \tag{4.11}$$

So by replacing $\tilde{u}$=u-f we now obtain a new equation where the first term in (4.5) (because of our new boundary condition $\frac{\partial \tilde{u}}{\partial n}$=0)vanishes and hence the new equation is given by

$$\int_\Omega \nabla\omega \cdot \nabla\tilde{u} d\Omega + \int_\Omega K^2 \tilde{u}\omega d\Omega = \left(2 - \frac{K^2 r^2}{2}\right)\int_\Omega \omega d\Omega. \tag{4.12}$$

Since the derivatives in (4.12) are of first order compared to that of (4.1) we can approximate $\tilde{u}(x, y)$ by a linear combination of piecewise linear trial functions given by:

$$\tilde{u}(x, y) = \sum_{j=1}^{N_e} U_j \phi_j(x, y), \quad \nabla\tilde{u}(x, y) = \sum_{j=1}^{N_e} U_j \nabla\phi_j(x, y), \quad \omega(x, y) = \phi_i(x, y) \tag{4.13}$$

where $\phi_j(x,y)$ are piecewise linear basis functions on a triangulation of the region Ω and the corners given by i=1,.....N_e (where N_e are the number of nodes elements). We then triangulate in Ω where the corners are $(\tilde{x_i}, \tilde{y_i})$ where i=1......N_t (where N_t are the number of triangles) and choose basis functions $\phi_i(\tilde{x_i}, \tilde{y_i}) = \delta_{i,j}$. Note that $\tilde{u}$ is differentiable only once and then only between $x_i's$ and U is piecewise once differentiable.
Substituting into (4.12) we get the following algebraic system:

$$\sum_{j=1}^{N_e} U_j \int_\Omega \nabla\phi_j(x,y)\nabla\phi_i(x,y)d\Omega + K^2 \sum_{j=1}^{N_e} U_j \int_\Omega \nabla\phi_j(x,y)d\Omega = (2-\frac{K^2r^2}{2})\int_\Omega \phi_i(x,y)d\Omega \quad (4.14)$$

rearranging gives

$$\sum_{j=1}^{N_e} U_j [\int_\Omega \nabla\phi_i(x,y)\nabla\phi_j(x,y)d\Omega + K^2 \int_\Omega \nabla\phi_i(x,y)\nabla\phi_j(x,y)d\Omega] = (2-\frac{K^2r^2}{2})\int_\Omega \phi_i(x,y)d\Omega. \quad (4.15)$$

The system of equations (4.15) is of the form

$$(K+M)U = F \quad (4.16)$$

where U is a vector of unknowns U_j. The stiffness matrix K, mass matrix and load vector are given by

$$K_{i,j} = \int_\Omega \nabla\phi_i(x,y)\nabla\phi_j(x,y)d\Omega, \quad j = 1,....N_e. \quad (4.17)$$

$$M_{i,j} = K^2 \int_\Omega \phi_i(x,y)\phi_j(x,y)d\Omega. \quad (4.18)$$

$$F_i = (2-\frac{K^2r^2}{2})\int_\Omega \phi_i(x,y)d\Omega. \quad (4.19)$$

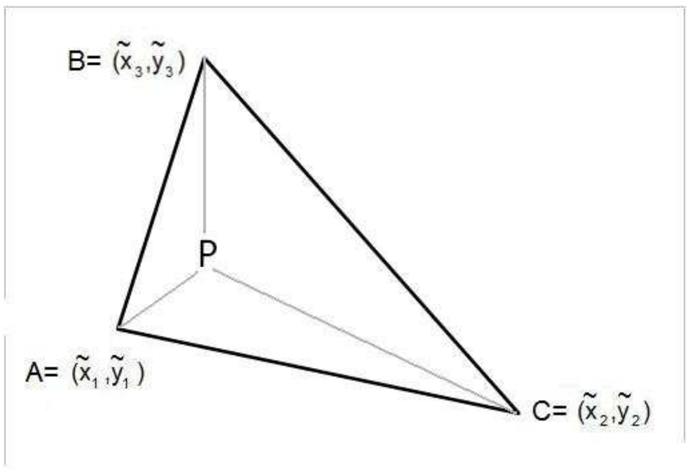
4.3 Stiffness matrix

In order to construct the stiffness matrix, we need to consider a general point P in a single triangle. This can be described by the coordinates with vertices ϕ_A=$\phi_{(\tilde{x}_1,\tilde{y}_1)}$, ϕ_B=$\phi_{(\tilde{x}_2,\tilde{y}_2)}$, ϕ_C=$\phi_{(\tilde{x}_3,\tilde{y}_3)}$ (refer to figure on other page) given by

$$\phi_A = \frac{Area(PBC)}{Area(ABC)} \quad \phi_B = \frac{Area(PCA)}{Area(ABC)} \quad \phi_C = \frac{Area(PAB)}{Area(ABC)} \quad (4.20)$$

where clearly ϕ_A+ϕ_B+ϕ_C=1. In this triangle there are three local linear basis functions,

$$\phi_A, \phi_B, \phi_C \quad (4.21)$$


B= $(\tilde{x}_3,\tilde{y}_3)$
P
A= $(\tilde{x}_1,\tilde{y}_1)$
C= $(\tilde{x}_2,\tilde{y}_2)$

The area of a triangle with vertices $\phi_{(\tilde{x}_1,\tilde{y}_1)}$, $\phi_{(\tilde{x}_2,\tilde{y}_2)}$, $\phi_{(\tilde{x}_3,\tilde{y}_3)}$

$$Area_{\Omega_i} = \frac{1}{2}D = \frac{1}{2}\begin{vmatrix} 1 & \tilde{x_1} & \tilde{y_1} \\ 1 & \tilde{x_2} & \tilde{y_2} \\ 1 & \tilde{x_3} & \tilde{y_3} \end{vmatrix}.$$

The basis function centred at $(\tilde{x_1}, \tilde{y_1})$ is given by

$$\phi_A = \phi_{(\tilde{x}_1,\tilde{y}_1)} = \frac{\begin{vmatrix} 1 & \tilde{x} & \tilde{y} \\ 1 & \tilde{x_2} & \tilde{y_2} \\ 1 & \tilde{x_3} & \tilde{y_3} \end{vmatrix}}{\begin{vmatrix} 1 & \tilde{x_1} & \tilde{y_1} \\ 1 & \tilde{x_2} & \tilde{y_2} \\ 1 & \tilde{x_3} & \tilde{y_3} \end{vmatrix}} = \frac{\tilde{x}_2\tilde{y}_3 - \tilde{x}_3\tilde{y}_2 + (\tilde{y}_2 - \tilde{y}_3)x + (\tilde{x}_3 - \tilde{x}_2)y}{(\tilde{x}_2 - \tilde{x}_1)(\tilde{y}_3 - \tilde{y}_1) - (\tilde{x}_3 - \tilde{x}_1)(\tilde{y}_2 - \tilde{y}_1)} = \begin{cases} 1 & \text{at } (\tilde{x}_1,\tilde{y}_1) \\ 0 & \text{at } (\tilde{x}_2,\tilde{y}_2) \\ 0 & \text{at } (\tilde{x}_3,\tilde{y}_3) \end{cases}$$

and hence

$$\frac{\partial\phi_1}{\partial x} = \frac{\partial\phi_{\tilde{x}_1,\tilde{y}_1}}{\partial x} = \frac{\tilde{y}_2 - \tilde{y}_3}{(\tilde{x}_2 - \tilde{x}_1)(\tilde{y}_3 - \tilde{y}_1) - (\tilde{x}_3 - \tilde{x}_1)(\tilde{y}_2 - \tilde{y}_1)} \tag{4.22}$$

$$\frac{\partial\phi_1}{\partial y} = \frac{\partial\phi_{\tilde{x}_1,\tilde{y}_1}}{\partial x} = \frac{\tilde{x}_3 - \tilde{x}_2}{(\tilde{x}_2 - \tilde{x}_1)(\tilde{y}_3 - \tilde{y}_1) - (\tilde{x}_3 - \tilde{x}_1)(\tilde{y}_2 - \tilde{y}_1)}. \tag{4.23}$$

Similar formulae hold for the basis functions centred at $\phi_B = \phi_(\tilde{x}_2, \tilde{y}_2)$, $\phi_C = \phi_(\tilde{x}_3, \tilde{y}_3)$. The stiffness matrix is given by

$$K_{i,j} = \int_\Omega \nabla\phi_i(x,y)\nabla\phi_j(x,y)d\Omega = \sum_{j=1}^{N_e}\int_\Omega \nabla\phi_i(x,y)\nabla\phi_j(x,y)d\Omega. \tag{4.24}$$

The stiffness matrix is made of the entries

$$K = \frac{1}{4Area_{\Omega_i}} \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix}. \qquad (4.25)$$

we compute this, where the nine combinations of nodes is displayed below, and then assembled into their appropriate locations in the stiffness matrix.

K_{11}=$\tilde{y}_2 - \tilde{y}_3)^2 + (\tilde{x}_3 - \tilde{x}_2)^2$.

K_{12}=$(\tilde{y}_2 - \tilde{y}_3)(\tilde{y}_3 - \tilde{y}_1) + (\tilde{x}_3 - \tilde{x}_2)(\tilde{x}_1 - \tilde{x}_3)$.

K_{13}=$(\tilde{y}_2 - \tilde{y}_3)(\tilde{y}_1 - \tilde{y}_2) + (\tilde{x}_3 - \tilde{x}_2)(\tilde{x}_2 - \tilde{x}_1)$.

K_{21}=$(\tilde{y}_2 - \tilde{y}_3)(\tilde{y}_3 - \tilde{y}_1) + (\tilde{x}_3 - \tilde{x}_2)(\tilde{x}_1 - \tilde{x}_3)$.

K_{22}=$(\tilde{y}_3 - \tilde{y}_1)^2 + (\tilde{x}_1 - \tilde{x}_3)^2$.

K_{23}=$(\tilde{y}_3 - \tilde{y}_1)(\tilde{y}_2 - \tilde{y}_1) + (\tilde{x}_1 - \tilde{x}_3)(\tilde{x}_2 - \tilde{x}_1)$.

K_{31}=$(\tilde{y}_2 - \tilde{y}_3)(\tilde{y}_1 - \tilde{y}_2) + (\tilde{x}_3 - \tilde{x}_2)(\tilde{x}_2 - \tilde{x}_1)$.

K_{32}=$(\tilde{y}_3 - \tilde{y}_1)(\tilde{y}_1 - \tilde{y}_2) + (\tilde{x}_1 - \tilde{x}_3)(\tilde{x}_2 - \tilde{x}_1)$.

K_{33}=$(\tilde{y}_1 - \tilde{y}_2)^2 + (\tilde{x}_2 - \tilde{x}_1)^2$.

Where K is a symmetric and positive definite matrix.

4.4 Mass Matrix

The mass matrix is complicated to compute because we now have linear functions. The components of the mass matrix are always of the form

$$M_{i,j} = K^2 \int_\Omega \phi_i(x,y)\phi_j(x,y)d\Omega \quad i,j = 1, ...N_e. \qquad (4.26)$$

Each of these integrals are evaluated using the centroid rule on each triangle. One way of avoiding this complication is to reduce this matrix to one with a more managable size and structure (preferably diagonal). This will be a sparse and real symmetric matrix banded in structure. Reducing the amount of computing time and saving more space

$$\int_\Omega \phi_i(x,y) \times \phi_j(x,y)d\Omega \approx \int_\Omega \prod(\phi_i\phi_j)d\Omega = \begin{cases} \text{volume of pyramid} & \text{if i=j} \\ 0 & \text{if i}\neq\text{j} \end{cases} \qquad (4.27)$$

Where $\prod$ is the sum of all the piecewise functions, and hence to evaluate our approximation to M we add the contribution (single area)

$$\frac{1}{3} area_{\Omega_i} \phi(\tilde{x}_i, \tilde{y}_j) \tag{4.28}$$

to the correct location for each of the three basis functions corresponding to the nodes of the triangles i. In turn for i=1,...N_t giving the entry for one triangle:

$$M_{\Delta} = \frac{K^2}{3} \begin{pmatrix} area_{\phi_1} & 0 & 0 \\ 0 & area_{\phi_2} & 0 \\ 0 & 0 & area_{\phi_3} \end{pmatrix}. \tag{4.29}$$

4.5 Load Vector

This can be calculated in the same way as the mass matrix. However instead of having a 3 by 3 matrix we have a vector.

$$F_i = (2 - \frac{K^2 r^2}{2}) \int_{\Omega} \phi_i(x, y) d\Omega \tag{4.30}$$

We can take $\phi_i(x, y)$ outside the integral as is evaulated at the centroid and equals to $\frac{1}{3}$.

$$F_i = (2 - \frac{K^2 r^2}{2}) \times \frac{1}{3} \begin{pmatrix} area_{\phi_1} \\ area_{\phi_2} \\ area_{\phi_3} \end{pmatrix}. \tag{4.31}$$

Where we use the same area as the one used in the mass matrix

4.6 Linear Solver

The linear system can be solved using any linear solver. The conjugate gradient scheme is a good choice because K is symmetric, positive definite and M is a diagonal matrix.

- runtime depends on size and sparsity pattern

Chapter 5

Exact Solution to the Modified Helmholtz Equation

From Chapter 3

$$u(r) = \frac{I_0(Kr)}{KI_1(K)}.$$

Refering to (3.37), the investigation conducted in Figure(5.1) and Figure (5.2) shows that the constant K determines how fast the exact solution approaches infinity. For example a small K results in the exact solution converges to inifinity a lot slower than that of a large K which converges very quickly at r=1.

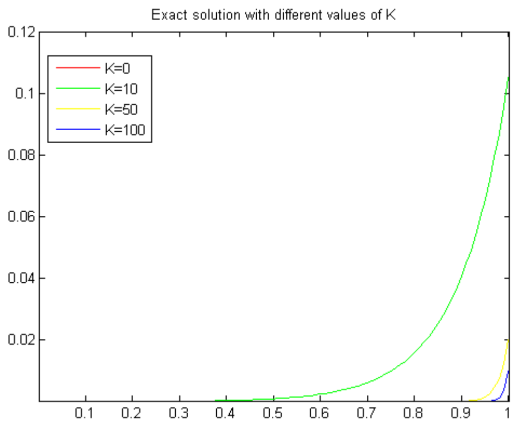


Figure 5.1: Exact solution plotted with different values of K

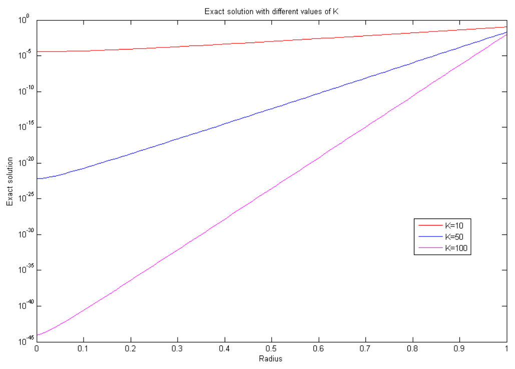


Figure 5.2: This graph shows the rate of change as you increase K, where a logarithm of base zero scale is used for the y axis

Further investigation shows that when approximating the Exact solution numerically we will have to consider two different cases:

- Uniform: interval is spilt into equal partitions.
- Non-uniform: interval is spilt into unequal partitions.

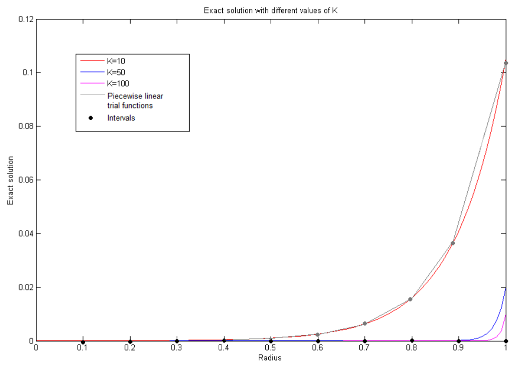
The numerical method being used to obtain our approximate solution is the Finite Element Method:

Finite Element

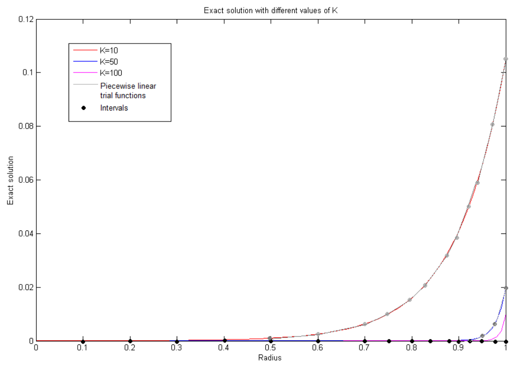
We wish to compare exact solution with our approximate solution method. We let our domain be a closed interval [0,1] and let

$$0 = x_0, x_1, ...x_{N+1} = 1$$

be a partition of the interval. Then place piecewise linear functions through each of these nodes, as shown in Figure (5.3(a)) and Figure (5.3(b)).



(a) Uniform mesh with equal partitions accross the x axis



(b) Non uniform mesh with more partitions at the boundary r=1

<u>Case 1: Uniform meshes</u>

As K tends to zero a uniform mesh is perfect due to the fact that the solution does not change very much since the exact solution is almost constant. So splitting the interval up into equal partitions will be very beneficial because piecewise linear functions can be easily passed through the nodes. Each of the partitions still give a good approximations (as seen in Figure (5.3(a))) whereas for large K this is not a very good approximation to the exact solution as can be seen for K=10, 50 and 100 (Figure (5.3(a))).

<u>Case two: Non-uniform meshes</u>

Close inspection shows that choosing K to be large (Figure (5.2)) places more nodes in the interval [0,1] (near the boundary r=1 and less nodes at r=0) (Figure (5.3(b))) will be more advantegous than that of an equal partition (Uniform mesh) which will not capture the steepness of the solution (values close to r=1) (Figure (5.2)). Non-uniform meshes will give a better approximation than that of uniform meshes. The reason being is that there are now more partitions in the region where the solution is rapidly approaching infinity enabling us to capture the steepness of the slope and give a better approximation at the boundary. (We could of spilt the interval up into very small uniform partitions, the disadvantages of this is that it will take a long time to compute, when a non-uniform mesh could compute this alot faster.)

Chapter 6

Investigation into the Different Types of Meshes

In this chapter, we will be looking at two different types of meshes, uniform and non-uniform. The quality of different sized meshes is investigated by seeing if changing h ($h_0 = h$ - the distance between the points in the initial distribution and h - the maximum side length of the triangle) will give a more accurate mesh hence more accurate results. Throughout this section the values h used are h=0.4, 0.2, 0.1 and 0.05 in a bounding box $-1 \leq x \leq 1$, $-1 \leq y \leq 1$ with no fixed points. The geometry is given as a distance function fd which returns the signed distance from each node position. This is given by

$$fd = \sqrt{x^2 + y^2} - 1 \quad \text{(on a unit circle)}. \tag{6.1}$$

A program has been written in Matlab [2] to produce the meshes to calculate the uniformity and quality for both uniform and non uniform meshes. There are two factors to be considered.

- Minimum Quality: Referring to (2.5) good quality meshes when q>0.5, poor quality meshes either have vertices very close to each other or are very coarse.

- Uniformity of triangle shapes: everything is regular and unvarying all of the triangles have similar characteristics. This helps tell how close the element sizes in the mesh p, t are to the desired function fh.[5]

where fd is the gemometry given as a distance function which returns all the signed distance from each node location to the boundary (from Chapter 1).

Three different types of meshes will be considered (where fh returns all the h(x,y) (desired edge length) for input):

- Uniform meshes-fh=h uniform.
- Non-uniform-fh=a-r.
- Non-uniform-fh captured by
$$\text{Error} \approx h^2 \left\| \frac{\partial^2 u}{\partial r^2} \right\|.$$

6.1 Rejection Method

Ideally we do not want to reject too many points because the mesh will become unstable and produce inaccurate results, or leave too many and it takes a long time to iterate. The probabilty to keep the points mentioned in the implementation is given by

$$r_0 = \frac{1}{(fh)^2} \tag{6.2}$$

Rejecting the points if,

$$\frac{r_0}{max(r_0)} < \text{rand} \quad \text{rand}\in(0,1) \tag{6.3}$$

ans also if,

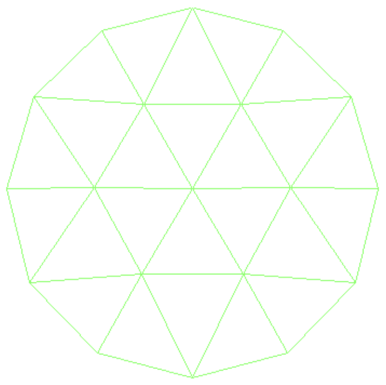
$$\frac{\max(r_0)}{\min(r_0)} \tag{6.4}$$

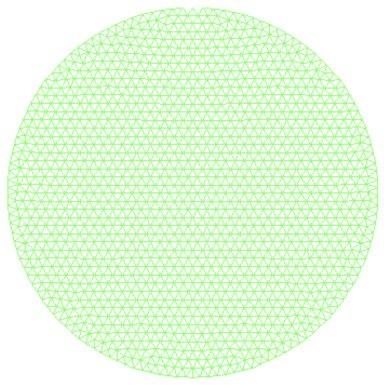
is too large, we reject most of the points, or if it is very small (very close to 1) most of the points are accepted.

6.1.1 Uniform meshes fh=h uniform

For a uniform mesh (6.3) $\frac{r_0}{max(r_0)}$=1 hence no points are rejected because (0,1)<1.

Uniform meshes		
h	Minimum quality	Uniformity (percent)
0.4	0.94	3.6
0.2	0.73	4
0.1	0.79	2.2
0.05	0.84	1.5


(c) h=0.4, Uniform mesh


(d) h=0.05, Non-uniform mesh

Inspection of h shows that all of the above meshes have a good minimum quality since q>0.5, this is what to be expected. By making h small more points are being placed in the circle (as can be seen by comparing Figure (6.1(c)) with Figure (6.1(d))) which has high resolution everywhere capturing the discontinuity better than large h producing more accurate results hence the approximate solution will be very close to the actual solution, one of the drawbacks of using smaller h is it will take longer to produce hence slow execution and sometimes non termination. There is no obvious pattern when inspecting the uniformity and the quality of each of these meshes, but what can clearly be seen is that both the quality and the uniformtiy of Figure (6.1(c)) is very high compared to that of h=0.05 (Figure (6.1(d))). This is because the mesh generator produces triangles that are almost equilateral however this may not give good results due to the error at the boundary.

6.1.2 Non uniform meshes fh=a-r

Suppose instead that r_0 is small at most of the points and large at a few points, $\frac{r_0}{max(r_0)}$ <<1. So we reject lots of the points and mesh becomes graded (the density of the triangles becomes greater at the boundary of the triangle see Figure (6.2(a)) and (6.2(b))). This can be done by using

$$\text{fh=a-r=}(\min(a-\sqrt{x^2+y^2}), b) \quad \text{min=minimum} \tag{6.5}$$

Where r (the radius) is given by $\sqrt{x^2+y^2}$, constant a >1 is the stabilty coefficient and b is the length of the triangles in r=$a-b$ (inside the radius r=$a-b$ the mesh is uniform and the length of the triangles is given b).

The fh function (6.5) can be defined as:

The minimum of $(a-\sqrt{x^2+y^2})$ can be found when $x_i^2+y_i^2=1$, giving $h=fh=(a-1)$ which is non uniform in the region $a-b$ <r<1.

Or if (a-r)=b, r=$a-b$ implying that all the triangles length inside the circle of radius 0<r≤ $a-b$ are equal to b and are uniform.

Rejection method for fh=a-r

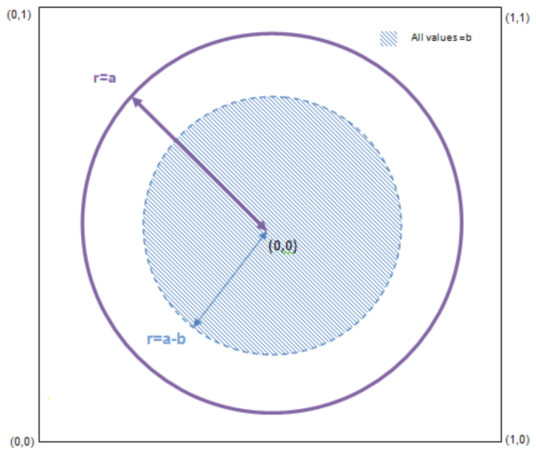


Figure 6.1: Rejection method-where the shaded region is uniform

For each point r, we compute

$$r_0=\frac{1}{fh^2}=\frac{1}{(\min(a-\sqrt{x^2+y^2}),b)^2} \tag{6.6}$$

a=1.1 and b=0.8

Inside 0<r≤0.3 all the h are equal to 0.8 and all the triangles produced by the fh function are uniform, with maximum triangles length (h). The function fh=(min(1.1-r)) is given when $x_i^2+y_i^2$=1. This implies fh=0.1, so the maximum length of the triangles h=0.1 are non uniform in the region 0.3<r≤1.

$r_0=\frac{1}{((\min(1.1-r))^2,(0.8)^2)}$ implying $\frac{\max(r_0)}{\min(r_0)}$ ≈64 so we will be rejecting quite a lot of points, hence the mesh become graded.

a=1.1 and b=0.95

Inside 0<r≤0.15 the maximum length of the triangles are equal to 0.95 and in here the region is uniform, otherwise non uniform. Here $\frac{\max(r_0)}{\min(r_0)} \approx 90$, hence we will be rejecting more points than that of a=1.1 and b=0.8. (See Figure (6.2(a)) and Figure (6.2(c)) for different sizes h).

a=1.3 and b=0.8

Inside 0<r≤0.5 the maximum length of the triangles are equal to 0.8, inside this region the mesh is uniform. Here $\frac{\max(r_0)}{\min(r_0)} \approx 4$, hence giving a much more uniform mesh since we are keeping most of the points. (See Figure 6.2(f))

<u>a=1.3 and b=0.95</u>

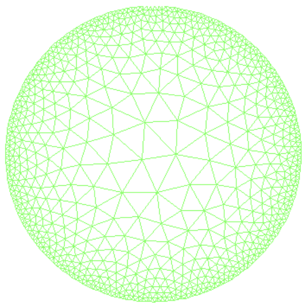
Inside 0<r≤0.35 the maximum legth of the triangles are equal to 0.95, inside this region the mesh is uniform. Here $\frac{\max(r_0)}{\min(r_0)} \approx 10$, hence keeping less points than that where b=0.8 and a=1.3. (See Figure (6.2(b)) and (6.2(e))).

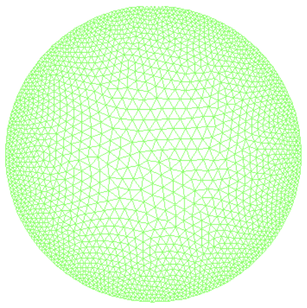
Non-uniform meshes where a=1.1 and b=0.95		
h	Quality	Uniformity (percent)
0.1	no output	no output
0.05	no output	no output
0.025	0.67	2.97
0.0125	0.69	2.4

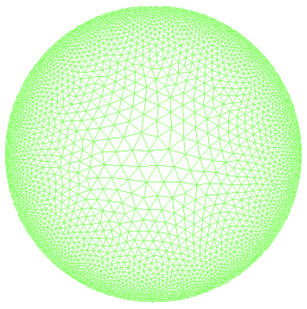
Non-uniform meshes where a=1.3 and b=0.95		
h	Quality	Uniformity (percent)
0.1	0.66	3.7
0.05	0.76	2.8
0.025	0.7	2.4
0.0125	0.8	2.2

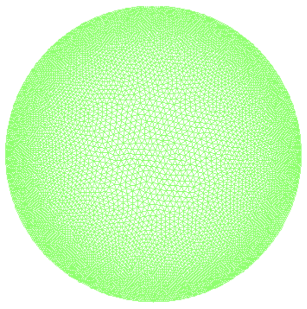
Non-uniform meshes where a=1.1 and b=0.8		
h	Quality	Uniformity (percent)
0.1	no output	no output
0.05	0.62	3.8
0.025	0.71	2.8
0.0125	0.71	2.5

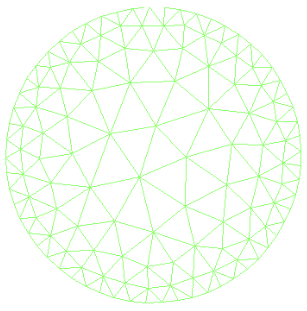
Non-uniform meshes where a=1.3 and b=0.8		
h	Quality	Uniformity (percent)
0.1	0.62	3.6
0.05	0.69	3
0.025	0.73	2.9
0.0125	0.76	2.2

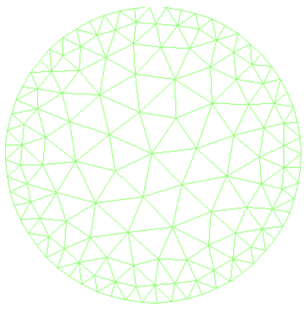

(a) a=1.1 b=0.95 and h=0.025


(b) a=1.3 b=0.95 and h=0.025


(c) a=1.1 b=0.95 and h=0.0125


(d) a=1.3 b=0.95 and h=0.0125


(e) a=1.3 b=0.95 and h=0.1


(f) a=1.3 b=0.8 and h=0.1

From looking at the tables above as h gets smaller the mesh become more defined at the boundary giving a better quality mesh but worse uniform mesh. This is due to the fact that the shapes of the triangles inside the mesh are no longer regular.
Increasing a (the coefficient of stabilty) tends to give a better quality mesh and more points in the centre but a decrease in uniformity. This is because when increasing the stabilty coefficient more nodes are placed in the centre of the region of interest making more triangles like that of an equilateral triangle (q=1) giving a good accuracy everywhere. However it takes longer to execute because not only do we have to move the nodes around the boundary we also have to move the many nodes in the centre (Figure (6.2(b)) and Figure (6.2(a))). So a decrease in the coefficient of stabilty may be more appropriate. This will be investigated later on.
Another important factor which must also be taken into consideration is the mesh size as a decrease in the size of h produces more points at the boundary giving a better defined mesh and

small errors. For example comparing Figure (6.2(b)) and Figure (6.2(a)) with Figure (6.2(c)) and Figure (6.2(d)). However sometimes this may not be nessesary since it takes longer and the mesh is already well defined for a bigger h.

A change in b (the length of the triangles in r=$a-b$) for example in increasing b for large h does not make a lot of difference e.g. Figure (6.2(e)) and (6.2(f)). In comparing the two tables above where a=1.1 and b=0.8 with a=1.1 and b=0.95 it can be seen that the quality and uniformity for them are both very similar showing that by changing b will not make a lot of difference. However further investigation has shown that when b=0.5 the mesh become uniform.

Overall choosing a to be very small gives a more accurate answer than that of a very big a. This is because when looking at the Exact solution (Chapter 5) at r=0 or r very close to zero the exact solution is almost constant. So by placing more nodes near r=0 will not affect the error much but only take up more time. The region of interest is at the boundary (at r=1) for large K when the exact solution tends to infinity very quickly with a very steep gradient. We want to try and caputure this and to get a good approximation close to the exact solution. A way of doing this is by placing more nodes near the boundary then form triagles using Delaunay to fit around the boundary. It is no good placing h very big at the boundary because we will not get a very well defined boundary. Hence we will get very big errors and the centre of our region will be very big with some very skinny triangles causing non-termination of the mesh generator because Deluanay cannot triangulate.

So a good choice of a is in choosing a to be very small where a>1 ideally 1.1. (Notice that when choosing a to be very small, no mesh is produced for big h, this is because when a is very small we are rejecting a lot of points (as seen in the tables above)). Ideally we want h to be very small at the boundary and 0.5< b <1. If we make b too small sometimes it will not terminate or the mesh will become uniform.

6.1.3 Non uniform meshes-part 2

We will choose our fh to be of the form:

$$\text{fh} \approx h^2 \left\| \frac{\partial^2 u}{\partial r^2} \right\| .$$

Error Analysis

From investigating the error

$$\text{Error} \approx h^2 \left\| \frac{\partial^2 u}{\partial r^2} \right\|. \tag{6.7}$$

we can decide what our h is so it gives the best possible solution with very small errors.

- h uniform-Uniform meshes, where the error is biggest when $\left\| \frac{\partial^2 u}{\partial r^2} \right\|$ is big.
- h non-uniform-Non uniform meshes, by choosinse h small $\left\| \frac{\partial^2 u}{\partial r^2} \right\|$ will be large or when h is large $\left\| \frac{\partial^2 u}{\partial r^2} \right\|$ will be small.

We will investigate both the errors on a Dirichlet and Neumann boundary.

<u>Dirichlet Boundary condition</u>

Consider the following boundary condition

$$u(r,\theta) = \frac{I_0(Kr)}{I_0(K)}, \quad \text{from (3.25)} \tag{6.8}$$

$$\frac{\partial u}{\partial r} = \frac{1}{I_0(K)} K I_0'(Kr) \tag{6.9}$$

$$= \frac{K}{2I_0(K)}[I_{-1}(Kr) + I_1(Kr)], \quad \text{(from [7])} \tag{6.10}$$

$$\frac{\partial u}{\partial r} = \frac{K}{I_0(K)}[I_1(Kr)] \tag{6.11}$$

Differentiating $\frac{\partial u}{\partial r}$ gives the following

$$\frac{\partial^2 u}{\partial r^2} = K^2 \frac{I_1'(Kr)}{I_0(K)} = \frac{K^2}{2I_0(K)}[I_0(Kr) + I_2(Kr)] \quad \text{(from [7])} \tag{6.12}$$

$$= \text{a} \times [I_0(Kr) + I_2(Kr)] \quad \text{where a=constant}$$

We are looking at when Kr is large, hence we can rewrite $I_n Kr$ as an asymptotic expansion (from [7]) of the form

$$I_n(Kr) = \frac{e^{Kr}}{\sqrt{2\pi Kr}}(1 + O(\frac{1}{Kr})). \tag{6.13}$$

Giving

$$\frac{\partial^2 u}{\partial r^2} \approx \text{C} \times \frac{e^{Kr}}{\sqrt{Kr}} \quad \text{(C=constant)} \tag{6.14}$$

So if we want to fix $h^2 \left\| \frac{\partial^2 u}{\partial r^2} \right\|$=constant

we set

$$h^2 \text{C} \frac{e^{Kr}}{\sqrt{Kr}} = \text{constant}$$

which on rearrangement gives

$$h^2 = \frac{\text{constant}\sqrt{Kr}}{\text{C}e^{Kr}} = \text{b}e^{-Kr}\sqrt{Kr} \quad \text{(b=constant)}$$

hence

$$h = \text{b}(Kr)^{\frac{1}{4}}e^{-\frac{Kr}{2}}. \tag{6.15}$$

when Kr is large.

<u>Neumann Boundary condition</u>

This time consider

$$u(r,\theta) = \frac{I_0(Kr)}{KI_1(K)}$$

$$\frac{\partial u}{\partial r} = \frac{KI_0'(Kr)}{KI_1(K)} \quad = \frac{1}{2I_1(K)}[I_{-1}(Kr) + I_1(Kr)]$$

Differentiating twice gives the following

$$\frac{\partial^2 u}{\partial r^2} = K\frac{I_1'(Kr)}{I_1(K)} = K\frac{I_0(Kr) + I_2(Kr)}{2I_1(K)} \tag{6.16}$$

$$= \text{a} \times [I_0(Kr) + I_2(Kr)] \quad \text{a=constant}$$

When Kr is large we can rewrite $I_n Kr$ as an asymptotic expansion similar to when using Dirichlet boundary condition.

Giving

$$\frac{\partial^2 u}{\partial r^2} \approx \text{C} \times \frac{e^{Kr}}{\sqrt{Kr}} \quad \text{(C=constant)} \tag{6.17}$$

So if we want to fix $h^2 \left\|\frac{\partial^2 u}{\partial r^2}\right\|$=constant

we set

$$h^2\text{C}\frac{e^{Kr}}{\sqrt{Kr}} = \text{constant}$$

which on rearrangement gives

$$h^2 = \frac{\text{constant}\sqrt{Kr}}{\text{C}e^{Kr}} = \text{b}e^{-Kr}\sqrt{Kr} \quad (b\text{=constant})$$

hence

$$h = b(Kr)^{\frac{1}{4}}e^{-\frac{Kr}{2}}. \tag{6.18}$$

when Kr is large.

An ideal choice is when:

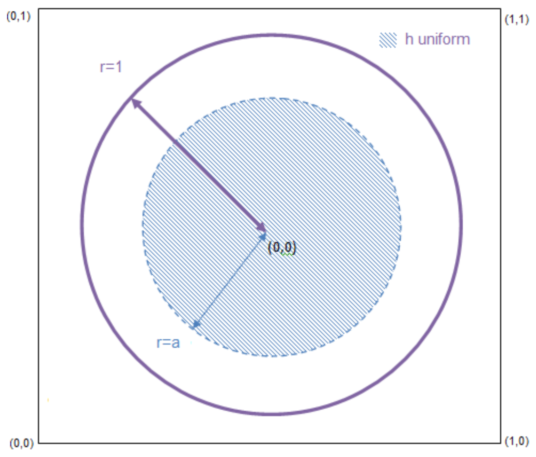


Figure 6.2: Rejection method for case 2

- h=uniform away from the boundary.
- h$\approx$ $b(Kr)^{\frac{1}{4}}e^{-\frac{Kr}{2}}$ when r$\approx$1.

We want h to satisfy the following two conditions:

- if r$\leq$a we choose h to be uniform
- if a<r<1 we choose h$\approx$$b(Kr)^{\frac{1}{4}}e^{-\frac{Kr}{2}}$
- and finally if r=a we choose h=$b(Ka)^{\frac{1}{4}}e^{-\frac{Ka}{2}}$

We can rewrite this as the following (where $e^{-\frac{Ka}{2}}$ is the dominant term):

- if 0<r$\leq$a, h$\approx$ $be^{-\frac{Ka}{2}}$ (region in which all the triangles are uniform)
- if a<r<1, h$\approx$ $be^{-\frac{Kr}{2}}$

Changing the constant b will help determine what choices of b to choose in order to produce a good mesh and small errors.
So for example a choice of b could be

b=$e^{\frac{K}{2}}$

function fh produces all the h for input, where

$$\text{fh} = \min(\left(be^{\frac{K}{2}(1-r)}\right), B). \tag{6.19}$$

This is defined as the following:

B=h (h=maximum length of the triangles) is uniform in the region (0<r≤a). Having established the rejection method in Section 6.1, we can apply it to fh where fh is now is given by (6.19). We want to see how the constants *b* and K affect the amount of points we reject and hence the accuracy of the solution.

The different values of h's in the different regions is calculated as follows:

<u>K=10, a=0.95</u>:

h=1 (Substituting r=1 into fh), 0.95<r<1 implying that h=1 in the region 0.95<r<1.

B=h=12 (Substituting r=0.95 into fh, where a=0.95 radius in which it is uniform) implying that B=h=12 in the region r≤0.95.

<u>K=100, a=0.95</u>: B=h=22026.5 is uniform in the region r≤0.95.

and h=1 where 0.95<r<1.

B is to big and hence no mesh can be generated.

b=$e^{\frac{K}{4}}$

function fh produces all the h for input, where

$$\text{fh} = \min(\left(e^{\frac{K}{4}(1-2r)}\right), B) \tag{6.20}$$

so h=min$(\left(e^{\frac{K}{4}(1-2r)}\right), B)$.

Part 1:

<u>K=10, a=0.95</u>:

B=h=0.105399 is uniform in the region 0<r≤0.95.

and h=0.082 in the region 0.95<r<1.

From the rejection method $\frac{\max(r_0)}{\min(r_0)}$ ≈60 showing that some points are rejected. This gives a very stable uniform mesh as shown in Figure (6.3(a)).

Part 2:

<u>K=100, a=0.95</u>:

B=h=0.0000000000169 is uniform in the region 0<r≤0.95.

and h=0.0000000000139 in the region 0.95<r<1.

The rejection method shows that $\frac{\max(r_0)}{\min(r_0)} \approx 148.4$ this shows that some points are being rejected more than that of K=10 (in Part 1 Figure(6.3(a))). The mesh produced here Figure (6.3(b)) is very similar to the one produced in in Part 1 Figure (6.3(a)).

b=$e^{\frac{K}{4}}$

function fh produces all the h for input, where

$$\text{fh} = (\min\left(e^{\frac{K}{4}(1-2r)}\right), B) \tag{6.21}$$

so h=(min $\left(e^{\frac{K}{4}(1-2r)}\right), B$).

Part 3:

<u>K=10, a=0.8</u>:

B=h=0.22313016 is uniform in the region 0<r≤0.8.

and h=0.082 in the region 0.8<r<1.

The rejection method shows that $\frac{\max(r_0)}{\min(r_0)} \approx 7.39$ (See Figure(6.3(c))) showing that less points are being rejected compared to that of (Part 1) (Figure6.3(a)) with a smaller a. This produces a non uniform mesh.

Part 4:

<u>K=100, a=0.8</u>:

B=h=0.0000003059 is uniform in the region 0<r≤0.8

and h=0.000000000013888 in the region 0.8<r<1.

The rejection method here shows that $\frac{\max(r_0)}{\min(r_0)} \approx 48515195$ is very large hence most of the points are rejected leading to an unstable mesh (Figure (6.3(d))).

b=$e^{\frac{K}{8}}$

Function fh produces all the h for input, where

$$\text{fh} = \min(\left(e^{\frac{K}{8}(1-4r)}\right), B) \tag{6.22}$$

so h=min($\left(e^{\frac{K}{8}(1-4r)}\right), B$).

Part 5:

<u>K=10, a=0.95</u>:

B=h=0.0302 is uniform in the region 0<r≤0.95.

and h=0.02352 in the region 0.95<r<1.

From the rejection method $\frac{\max(r_0)}{\min(r_0)} \approx 1.648$ showing that very few points are being rejected (Figure (6.3(e))) compared to that of part 1 Figure(6.3(a)) but for a different b.

Part 6:

K=100, a=0.95:

B=h=0.0000000000000006305 is uniform in the region 0<r≤0.95.

and h=0.000000000000000051755 in the region 0.95<r<1.

The rejection method shows $\frac{\max(r_0)}{\min(r_0)} \approx 148.413$ (Figure (6.3(f))) showing that more points are being rejected than for a smaller K (Part 5, Figure(6.3(e)))and also h is very close to zero, hence no output as shown in Figure(6.3(f)).

Let

b=$e^{\frac{K}{4}}$

function fh produces all the h for input, where

$$\text{fh} = \min(\left(e^{\frac{K}{4}(1-2r)}\right), B) \tag{6.23}$$

so h=min$(\left(e^{\frac{K}{4}(1-2r)}\right), B)$.

Part 7:

K=10, a=0.8:

B=h=0.0639 is uniform in the region 0<r≤0.8.

h=0.02352 in the region 0.8<r<1.

The rejection method here shows that $\frac{\max(r_0)}{\min(r_0)} \approx 7.4$ showing that very few points are being rejected giving a uniform mesh (See Figure (6.3(g))).

Part 8:

K=100, a=0.8:

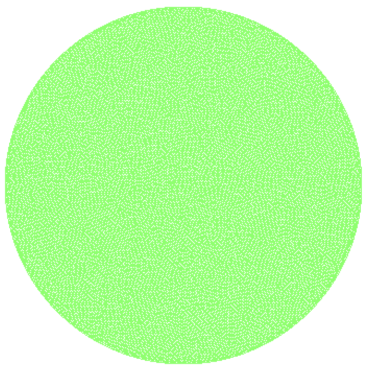
B=h=0.00000000000114 is uniform in the region 0<r≤0.8.

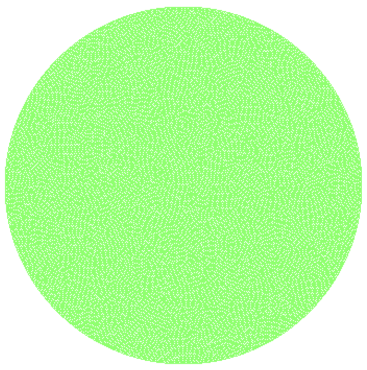
h=0.000000000051755 in the region 0.8<r<1.

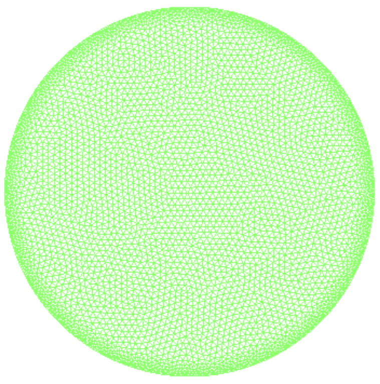
The rejection method $\frac{\max(r_0)}{\min(r_0)} \approx 485164$, which is very large hence most of the points are rejected giving us a very unstable mesh (See Figure (6.3(h))).

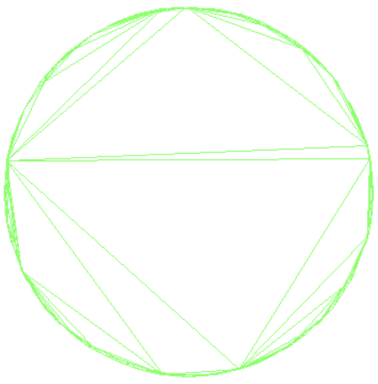
Investigation shows that ***Part 3*** (Figure 6.3(c))and ***7*** (Figure 6.3(g)) gives non uniform meshes when K=10, unstable meshes when K=100 for ***Part 4*** (Figure 6.3(d))and ***Part 8*** (Figure

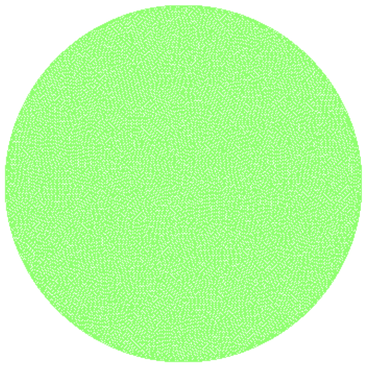
6.3(h)), where the unstable meshes comes from rejecting too many points. Finally no mesh ***Part 6*** (Figure 6.3(f))when h is very small in the two region 0<r≤a and a<r<1.


(a) Part 1: K=10 h=0.0125


(b) Part 2: K=100 h=0.0125

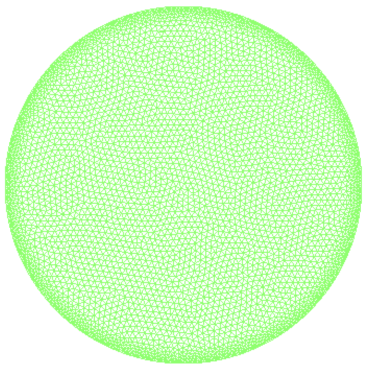

(c) Part 3: K=10 h=0.0125


(d) Part 4: K=100 h=0.0125

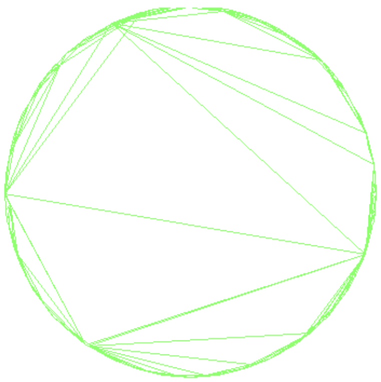


(e) Part 5: K=10 h=0.0125

(f) Part 6: K=10 h=0.0125



(g) Part 7: K=10 h=0.0125



(h) Part 8: K=100 h=0.0125

Chapter 7

Numerical Simulations and Results for the Modified Helmholtz Equation

In this section, the results from the finite element method described in Chapter 4 are compared with the exact solution described in Chapter 3.
Our aim is to investigate what mesh properties will give good results with the smallest possible maximum errors.
Throughout this section we will choose our h (maximum length of triangles) to be of the values 0.4, 0.2, 0.1 and 0.05, because when choosing our maximum length to be very big the program produces meshes that are not very good, giving bad results with very big errors. We will also choose K (constant) to be very big, since investigation has shown that for small K the exact solution and the finite element solution behave in a very similar fashion leading to very small errors. However when choosing K to be very large our exact solution u(r,θ)(3.37)will be highly peaked near the boundary. We want to be able to choose suitably sized triangles so that the mesh can capture this steepness hence giving a very good approximation and very small errors.

7.1 Uniform meshes

We start off by looking at a uniform mesh for different values of K. Comparing the number of points, the smallest maximum error and the location of the errors for different mesh size (h).

7.1.1 K=10

Uniform meshes			
h	number of points	maximum error	location of error
0.4	19	0.1333	edge
0.2	88	0.0395	edge
0.1	362	0.009	edge
0.05	1452	0.0026	edge
0.025	5809	0.00069074	edge

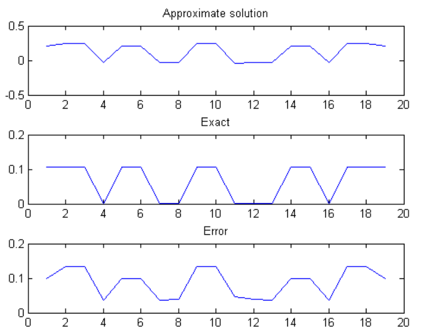
7.1.2 K=50

Uniform meshes			
h	number of points	maximum error	location of error
0.4	19	0.1963	edge
0.2	88	0.0911	edge
0.1	362	0.0295	edge
0.05	1452	0.0099	edge
0.025	5809	0.0026	edge

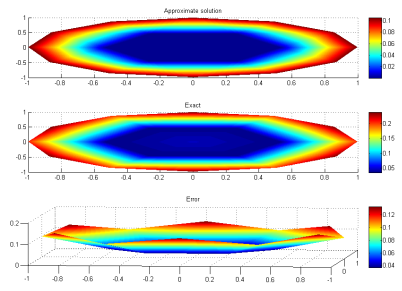
7.1.3 K=100

Uniform meshes			
h	number of points	maximum error	location of error
0.4	19	0.2057	edge
0.2	88	0.1004	edge
0.1	362	0.0367	edge
0.05	1452	0.0218	edge
0.025	5809	0.0049	edge

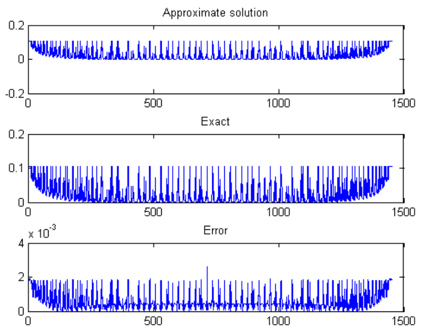
Initially the procedure was applied to K=10. Close inspection of the tables above shows that as h decreases the number of nodes inside the mesh increases as shown in the table above and also in Chapter 6. Hence the graph for the approximate and actual solution becomes more dense (Comparing Figure (7.1(i)) and Figure(7.1(k))) giving larger errors at the boundary. This is because for large triangles, the triangles cannot capture the solution near the boundary of the circle very well (comparing Figure (7.1(j)) with Figure (7.1(l))).



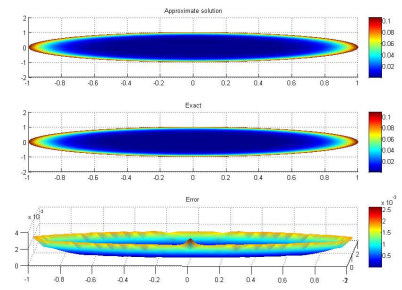
(i) K=10, h=0.4, this graph shows the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u



(j) K=10, h=0.4, this graph shows an overhead view of the solution at each node

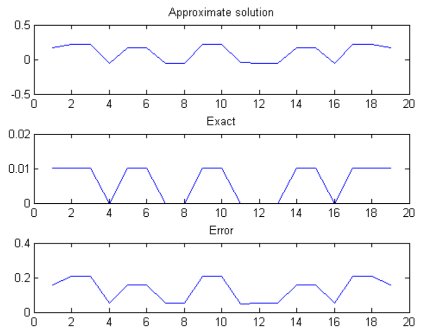


(k) K=10, h=0.05, this graph shows the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u

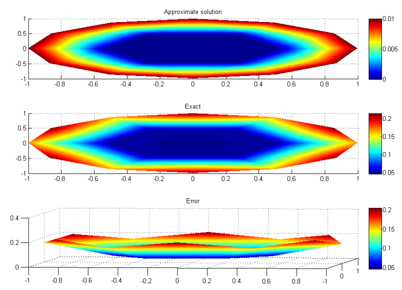


(l) K=10, h=0.05, this graph shows an overhead view of Figure (7.1(k))

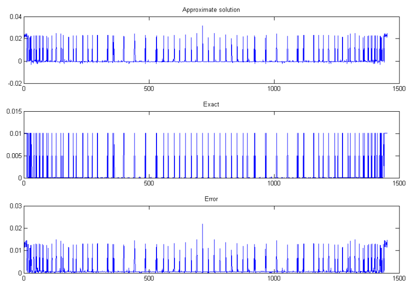
Further inspection has shown that as K increases (Figure (7.1(m))) the exact solution mentioned in Chapter 5 u(r,θ)(3.37) converges to infinity a lot faster at the boundary than small K (Figure (7.1(i))). Thus making it harder to capture the steepness of the slope (7.1(j))(at the boundary of the circle) as mentioned before, hence giving bigger errors as shown in the table above. (Note that we do not reject any points inside the mesh because it satisfies the condition



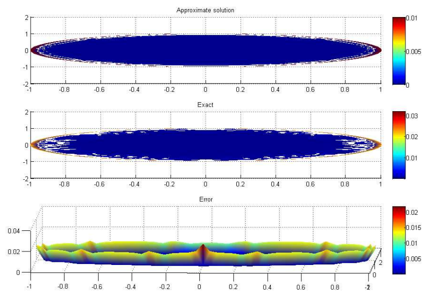
(m) K=100, h=0.4, this graph shows the solutions at each node, the x axis representing the nodes and the y axis representing the solutions



(n) K=100, h=0.4, this graph shows an overhead view of Figure (7.1(m))



(o) K=100, h=0.05, this graph shows the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u



(p) K=100, h=0.05, this graph shows an overhead view of Figure (7.1(o))

mentioned in Chapter 6 (6.3).)
As a result of this when choosing a uniform mesh and K to be large, we want h to be very small to capture the peak near the boundary (comparing Figure (7.1(n)) with Figure (7.1(p))) in order to give smaller errors (shown in the table above). However if K is very small, because

the errors are already small, it may be more appropriate to use a larger h, because the solution is not changing very much, making it easier to capture the peak. The graph (Figure (7.1.3))

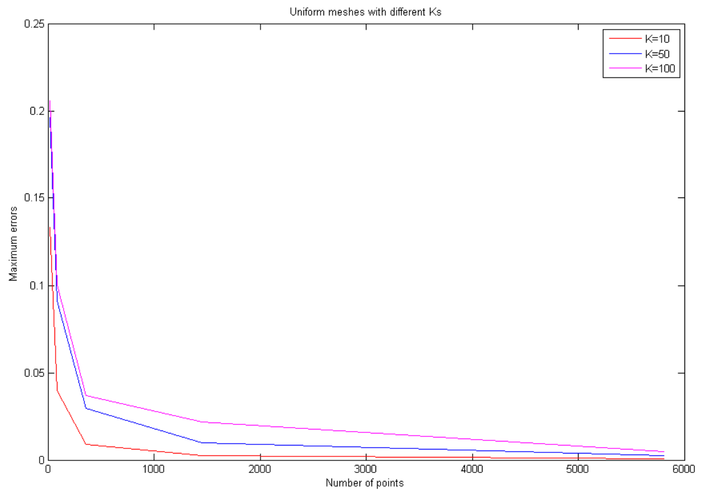


Figure 7.1: Comparing the number of points with the maximum errors for a uniform mesh

shows that when K is large the solution becomes highly peaked at the boundary. This is because as K increases the solution (3.37) will get very large and become very steep at the boundary. Hence the maximum errors and the number of points will increase. So having a uniform mesh is not ideal when K is very big.

7.2 Non uniform meshes fh=a-r

We start off by looking at a Non uniform mesh (fh=a-r) with different values of K, a and b. Comparing the number of points, the smallest maximum error and the location of the errors for different mesh size (h). The coefficient a is the stabilty coefficient and b is the length of the triangles in r=$a-b$ (inside the radius r=$a-b$ the mesh is uniform and the length of the triangles is given b) (In §6.1.2).

7.2.1 K=10

a=1.1 and b=0.95

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	219	0.0291	centre
0.025	887	0.0074	centre
0.0125	3560	0.0017	centre

a=1.3 and b=0.95

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	473	0.0055	centre and edge
0.025	1989	0.0012	centre and edge
0.0125	7870	0.00033848	centre and edge

a=1.1 and b=0.8

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	224	0.0281	centre
0.025	886	0.0066	centre
0.0125	3558	0.0015	centre

a=1.3 and b=0.8

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	547	0.0047	edge
0.025	2056	0.0013	edge
0.0125	8126	0.0003319	centre

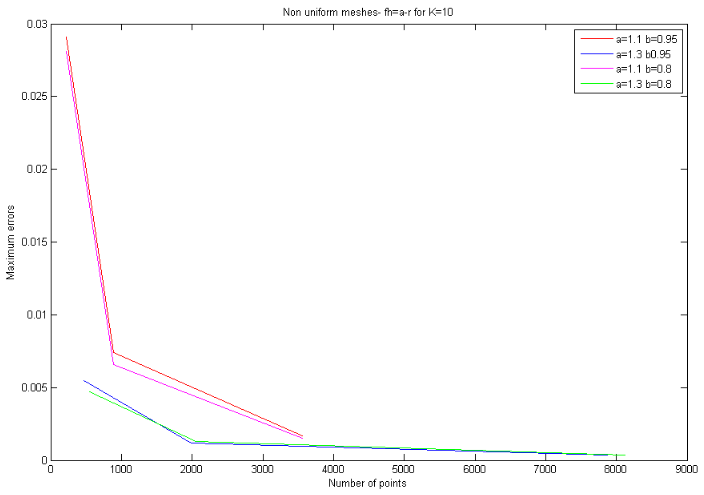


Figure 7.2: Comparing the number of points with the maximum errors for a non uniform mesh, where fh=a-r. It can be seen from Figure(2.1) that choosing a=1.3 gives the best result capturing the boundary a lot better than when a=1.1. This is because $\frac{\max(r_0)}{\min(r_0)}$ is very small.

7.2.2 K=100

a=1.1 and b=0.95

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	216	0.0402	centre
0.025	877	0.0109	centre
0.0125	3546	0.0028	centre

a=1.3 and b=0.95

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	518	0.0169	edge
0.025	911	0.001	edge
0.0125	3568	0.0031	centre

a=1.1 and b=0.8

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	219	0.0364	centre
0.025	908	0.0094	centre
0.0125	3540	0.0033	centre

a=1.3 and b=0.8

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	499	0.0182	edge
0.025	2010	0.0055	edge
0.0125	7984	0.0022	centre

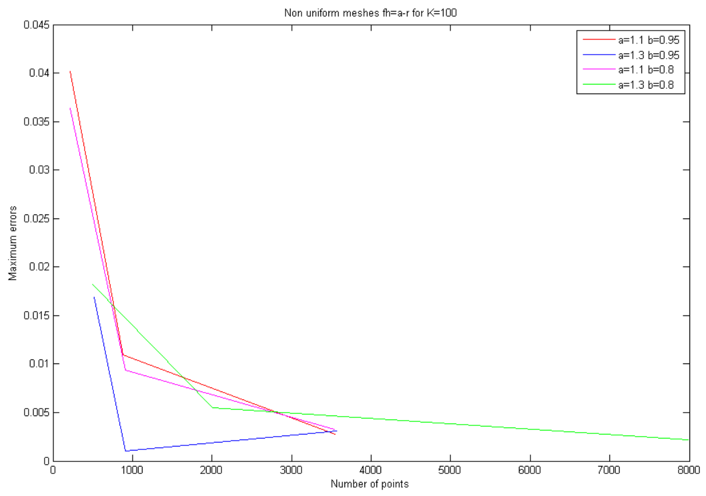
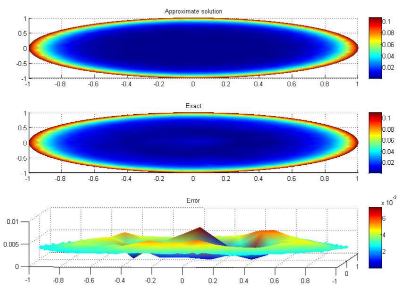
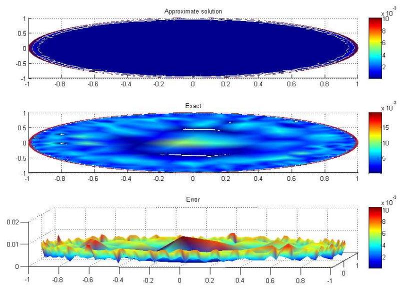


Figure 7.3: Comparing the number of points with the maximum errors

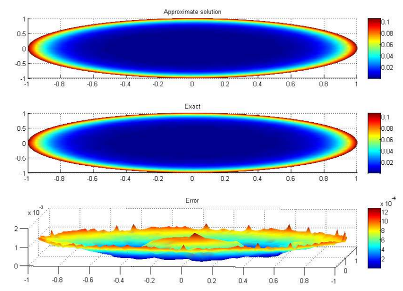
From the graph above Figure (7.3), it can be seen that for K=100 choosing a=1.1 is better than a=1.3 regardless of what b is. This is because the points and the errors converges a lot faster than when a=1.3. From chapter 6 it can be seen that the mesh (Figure (6.2(d))) for a=1.3 is much denser and contains more nodes than Figure (6.2(c)), this is the same for Figure (6.2(a)) and Figure (6.2(b)).



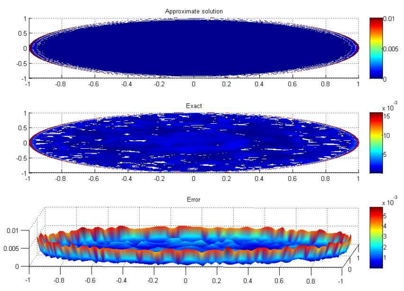
(a) Non-uniform mesh where K=10, a=1.1 and b=0.95 where h=0.025, this graph shows an overhead view of the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u



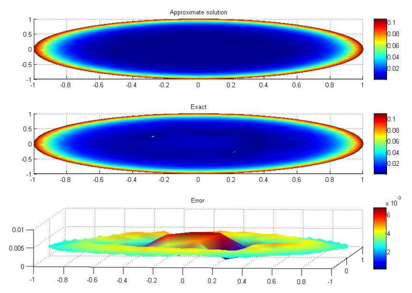
(b) Non-uniform mesh where K=100, a=1.1 and b=0.95 where h=0.025, this graph shows an overhead view of the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u



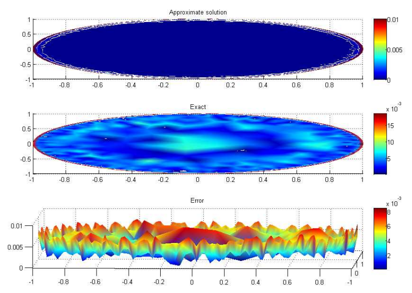
(c) Non-uniform mesh where K=10, a=1.3 and b=0.95 where h=0.025, this graph shows an overhead view of the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u



(d) Non-uniform mesh where K=100, a=1.3 and b=0.95 where h=0.025, this graph shows an overhead view of the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u



(e) Non uniform mesh where K=10, a=1.1 and b=0.8 where h=0.025, this graph shows an overhead view of the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u

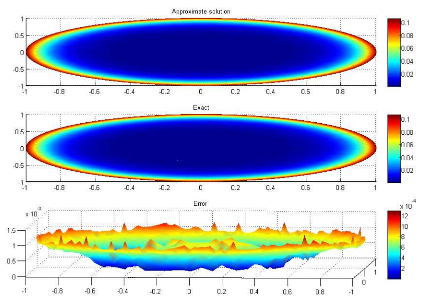


(f) Non uniform mesh where K=100, a=1.1 and b=0.8 where h=0.025, this graph shows an overhead view of the solutions at each node, the x axis representing the nodes and the y axis representing the solutions u

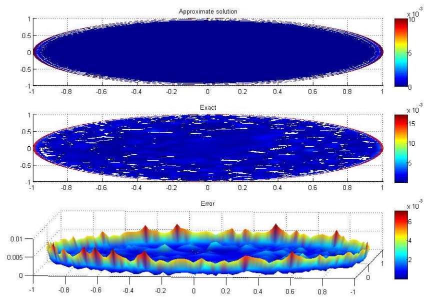
Close inspection shows that when choosing h=0.4, the non uniform meshes behave almost the same as the uniform meshes. As h gets smaller the meshes become better quality especially at the boundary as shown in all the tables above and also in the previous chapter. This is because smaller nodes are being placed in the b region (in which the values are uniform) giving smaller errors than that of a larger h. The two important constants in this non uniform mesh are a and b defined earlier in chapter 6. Increasing the coefficient of stabilty a, keeping b the same, gives a better quality mesh with decreasing errors (hence taking longer to iterate). This is because when a is very small, the radius in which all the values are equal to b is small resulting in more points being placed in the centre as shown in Chapter 6. However an increase in b will result in rejecting more points (at the centre of the mesh) leading to larger errors, as shown in Chapter 6. (Figure (7.4(a)) and Figure (7.4(b)) compared with that of Figure(7.4(e)) and Figure (7.4(f))).

We want to compare this with different K, where K=10 and 100. A noticable change when making K larger is that a is small for b=0.95 and 0.8. The number of points inside the region decreases leading to an increase in errors, because as K gets larger the exact solution becomes more peaked at the boundary. This in turns lead to bigger errors when placing too little nodes in the outer boundary because we will not be able to capture the peak at the boundary of the circle.

The coefficients a and b behave similarly to that of small K and large K.



(g) Non uniform mesh where K=10, a=1.3 and b=0.8 where h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u



(h) Non uniform mesh where K=100, a=1.3 and b=0.8 where h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u

By looking at Figure (7.2.2) it can be seen that choosing fh=a-r, a=1.1 gives the best results.

7.3 Non uniform meshes fh=$be^{-\frac{Kr}{2}}$

Finally looking at fh where fh is corporated by coding at exact values of u"

Part 1

K=10, b=$e^{\frac{K}{4}}$ and a=0.95			
h	number of points	maximum error	location of error
0.4	19	0.1333	boundary
0.2	47	0.0680	boundary
0.1	238	0.0131	boundary
0.05	914	0.0041	boundary
0.025	3584	0.0011	boundary
0.0125	14287	0.0002633	boundary

From Chapter 6

h=0.082 is uniform in the region 0.95<r<1.

and h=0.105399 in r≤0.95.

A close inspection shows that h is very similar in the two regions hence the mesh produced looks almost uniform (see Figure (7.5(a))). Similar to that of a uniform mesh where K=10 which has sligthly more points hence smaller errors than the one produced comparing the table above. This is what is to be expected since Part 1 $\frac{\max(r_0)}{\min(r_0)} \approx 60$.

Part 2

K=100, b=$e^{\frac{K}{4}}$ and a=0.95			
h	number of points	maximum error	location of error
0.4	19	0.2057	boundary
0.2	64	0.1005	boundary
0.1	241	0.0467	boundary
0.05	914	0.0201	boundary
0.025	3866	0.006	boundary
0.0125	15542	0.0065	boundary

From Chapter 6

h=1.39×10^{-11} ≈0 is uniform in the region 0.95<r<1.

and h=1.69×10^{-11} $\approx$0 in r$\leq$0.95.

By inspection the table shows that the results produced from part 1 and part 2 are very similar, however this time the exact solution tends to infinity a lot faster.

So by making h very small we can capture this discontinuity a lot better than with a big h. From comparing the uniform mesh where K=100 with Part 2 (see Figure (7.5(b)))shows that Part 2 gives a better solution because it contains less points and smaller error because Part 2 $\frac{\max(r_0)}{\min(r_0)} \approx 148$ and Part 1 ≈ 60.

Part 3

K=10, b=$e^{\frac{K}{4}}$ and a=0.8			
h	number of points	maximum error	location of error
0.4	19	0.2040	top
0.2	0	0	0
0.1	0	0	0
0.05	266	0.0150	boundary
0.025	1123	0.0032	boundary
0.0125	4488	0.00069624	boundary

From Chapter 6

h=0.082 is uniform in the region 0.8<r<1.

and h=0.2231301 in r$\leq$0.8.

Notice however this time when making *a* smaller there is a more noticeable difference in h for 0.8<r<1 and r$\leq$ 0.8 resulting in a non uniform mesh and $\frac{\max(r_0)}{\min(r_0)}$ approximately 7 . Comparing this with the previous non uniform mesh where K=10 and fh=a-r. Part 3 (see Figure (7.5(c)))gives a better solution than that of all of the above non uniform results and of a better quality. However there are no solutions for h=0.2 and 0.1.

Part 4

K=100, b=$e^{\frac{K}{4}}$ and a=0.8			
h	number of points	maximum error	location of error
0.4	0	0	0
0.2	0	0	0
0.1	0	0	0
0.05	17	0.4132	middle
0.025	67	00.3908	middle
0.0125	247	0.4269	middle

From Chapter 6

h=$1.389\times10^{-11}\approx0$ is uniform in the region 0.8<r<1.

and h=$5.039\times10^{-7}\approx0$ in r$\leq$0.8.

As can be seen from the previous chapter $\frac{\max(r_0)}{\min(r_0)}$ quite big so lot of points are rejected (see Figure (7.5(d))) especially in the centre giving very inaccuarate results which in turn leads to bigger errors in the cenre of the mesh as can be seen in Chapter 6 hence this is not a good choice and is to be avoided.

Part 5

K=10, b=$e^{\frac{K}{8}}$ and a=0.95			
h	number of points	maximum error	location of error
0.4	19	0.1333	boundary
0.2	53	0.0587	boundary
0.1	2208	0.0144	boundary
0.05	891	0.0040	boundary
0.025	3501	0.0011	boundary
0.0125	14173	0.0002735	boundary

From Chapter 6

h=0.02352 is uniform in the region 0.95<r<1.

and h=$5.039\times10^{-7}\approx0$ in r$\leq$0.95.

From inspection we can see that there is not a lot of different between the two different values of h in the two regions (see Figure (7.5(e))), hence producing a uniform mesh as can be seen in Chapter 6 $\frac{\max(r_0)}{\min(r_0)}\approx1.64$ whereas other uniform meshes produced in this section (Part 1)

$\frac{\max(r_0)}{\min(r_0)} \approx 60$ when K=10 giving a better solution than the one produced here.

Part 6

K=100, b=$e^{\frac{K}{8}}$ and a=0.95			
h	number of points	maximum error	location of error
0.4	19	0.2057	boundary
0.2	no ouput	no ouput	no ouput
0.1	no ouput	no ouput	no ouput
0.05	no ouput	no ouput	no ouput
0.025	no ouput	no ouput	no ouput
0.0125	no ouput	no ouput	no ouput

From Chapter 6

h=5.1755×10^{-11} ≈0 is uniform in the region 0.95<r<1.

and h=6.30511×10^{-7} ≈0 in r≤0.95.

This choice of a, b and K, causes too many points to be rejected so no results is produced. (see Figure (7.5(f)))

Part 7

K=10 b=$e^{\frac{K}{8}}$ and a=0.8			
h	number of points	maximum error	location of error
0.4	19	0.2037	top
0.2	no ouput	no ouput	no ouput
0.1	no ouput	no ouput	no ouput
0.05	295	0.0137	boundary
0.025	1170	0.003	boundary
0.0125	4521	0.0071477	boundary

From Chapter 6

h=1.0.02352 is uniform in the region 0.8<r<1.

and h=0.0639 in r≤0.8.

Behave similarly to that of Part 3 (see Figure (7.5(c))), however this one produces better result because the region contains less points and hence smaller errors. (see Figure (7.5(g)))

Part 8

K=100, b=$e^{\frac{K}{8}}$ and a=0.8			
h	number of points	maximum error	location of error
0.4	no ouput	no ouput	no ouput
0.2	no ouput	no ouput	no ouput
0.1	0	no ouput	no ouput
0.05	14	0.4538	everywhere
0.025	67	0.4135	centre
0.0125	243	0.3958	centre

From Chapter 6

We can clearly see that we are rejecting too many points hence for Part 8 it produces a very unstable mesh and hence produces very inaccurate solutions (see Figure (7.5(h))).

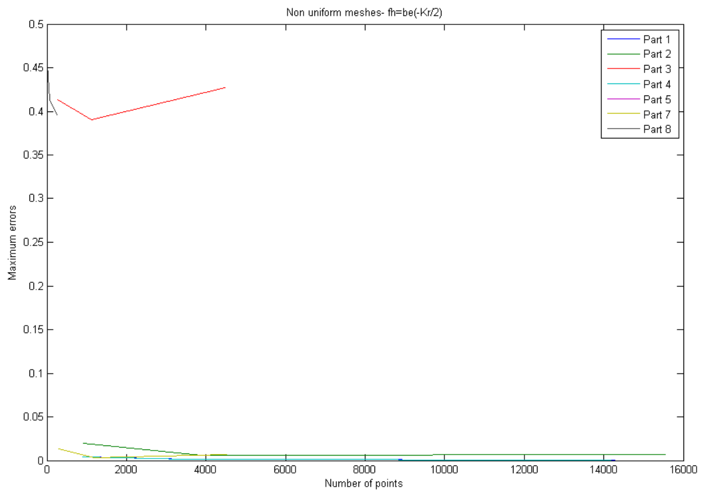
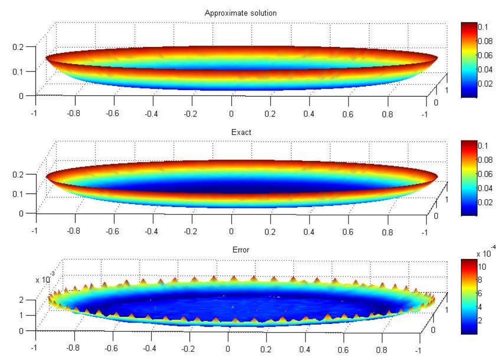
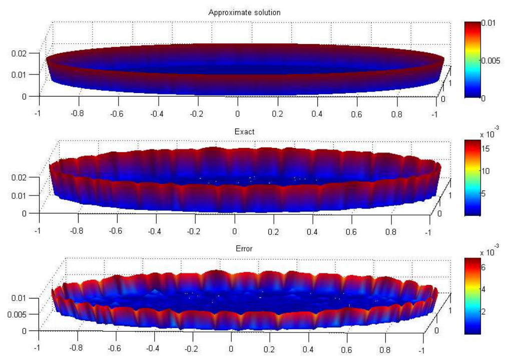


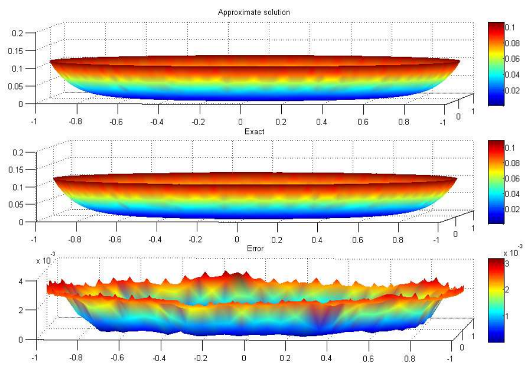
Figure 7.4: Looking at the figure above we see that Part 3 and Part 8 give bad results. For Part 8, K=10, b=$e^{12.5}$ and a=0.8


(a) Non uniform mesh: Part 1 K=10 h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u

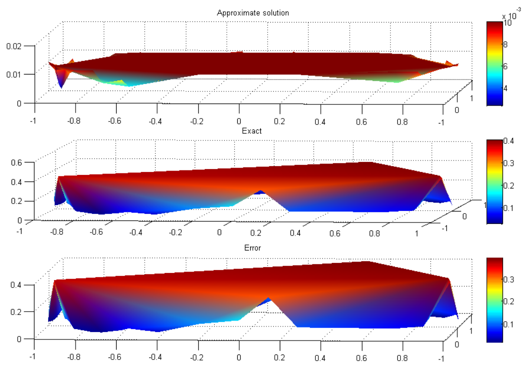

(b) Non uniform mesh: Part 2 K=100 h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u

It can be seen from Figure (7.5(a)) that the errors are at the boundaries of the circle and that in Figure (7.5(h)), the errors are larger due to a coarser mesh. All the other Parts (1,2,4,5 and 7) produce better meshes. Part 1 and Part 5 have jagged rims as can be seen from Figure (7.5(a)) and Figure (refpart5trisurf). This is when K is small.
For b=$e^{12.5}$ when K=100 and a=0.95 (Part 6), we have a very unstable mesh by virtue of rejecting too many points (from Chapter 6). Hence there is no output Figure (7.5(f)). Overall we notice that for large K (for example K=100) that the output shown in Figure (7.5(f)) and Figure (7.5(h)) produces very unstable meshes. This is because for large K, the error is peaked at the boundary.

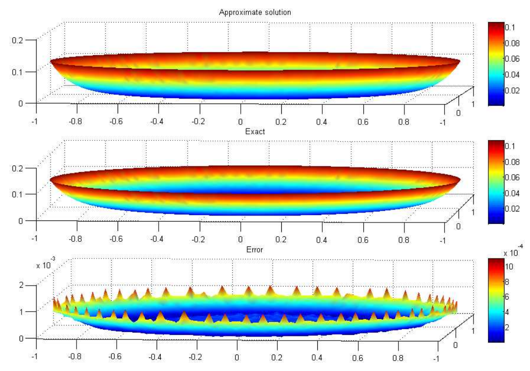
A close examination of a non uniform mesh where fh=a-r is better than a uniform mesh by comparing graphs, this is what is to be expected as fh is more graded at the boundarywhere a=1.1 gives the best result. Changing fh to fh=h^2u" is better than a uniform mesh.



(c) Non uniform mesh: Part 3 K=10 h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u

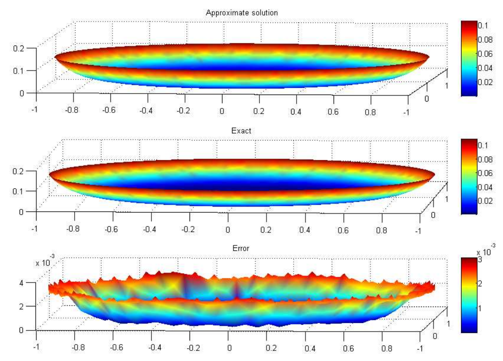


(d) Non uniform mesh: Part 4 K=100 h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u

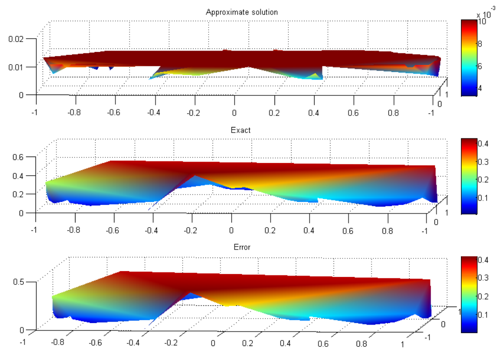


(e) Non uniform mesh: Part 5 K=10 h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u

(f) Non uniform mesh: Part 6 (No output) K=100 h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u



(g) Non uniform mesh: Part 7 K=10 h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u



(h) Non uniform mesh: Part 8 K=100 h=0.025, this graph shows an overhead view of the solutions at each node where the x axis representing the nodes and the y axis representing the solutions u

Chapter 8

Investigation in a Change in the Tolerance in the Mesh Generator

It is expected that when we change tolerance (in the generator) by making it bigger, the program iterate and produce the solution and mesh a lot faster than of a smaller tolerance. This is because the maximum node movement in the mesh generator has to be less than 0.02 whereas before it had to be less than 0.01. Causing the program to terminate quicker.

8.1 Uniform meshes

8.1.1 K=10

Uniform meshes			
h	number of points	maximum error	location of error
0.4	19	0.1332	edge
0.2	88	0.0395	edge
0.1	362	0.0091	edge
0.05	1452	0.0026	edge
0.025	5809	0.0007059	edge

8.1.2 K=50

Uniform meshes			
h	number of points	maximum error	location of error
0.4	19	0.1962	edge
0.2	88	0.092	edge
0.1	362	0.0295	edge
0.05	1452	0.0100	edge
0.025	5809	0.0367	edge

8.1.3 K=100

Uniform meshes			
h	number of points	maximum error	location of error
0.4	19	0.2056	edge
0.2	88	0.1006	edge
0.1	362	0.0367	edge
0.05	1452	0.0217	edge
0.025	5809	0.005	edge

8.2 Non uniform meshes fh=a-r

8.2.1 K=10

a=1.1 and b=0.95

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	220	0.0241	edge
0.025	878	0.0072	centre
0.0125	3526	0.0018	centre

a=1.3 and b=0.95

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	486	0.005	centre and edge
0.025	1988	0.0013	centre and edge
0.0125	3496	0.0016	centre and edge

a=1.1 and b=0.8

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	216	0.0316	centre
0.025	904	0.0066	centre
0.0125	7896	0.00034524	centre and boundary

a=1.3 and b=0.8

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	531	0.00053	boundary
0.025	2056	0.0013	bnoundary
0.0125	8028	0.00030822	boundary

8.2.2 K=100

a=1.1 and b=0.95

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	211	0.0539	centre
0.025	903	0.0095	centre
0.0125	904	0.0120	centre

a=1.3 and b=0.95

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	473	0.0226	edge
0.025	1918	0.0062	edge
0.0125	7816	0.0023	centre

a=1.1 and b=0.8

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	240	0.0346	edge
0.025	864	0.0105	centre
0.0125	3550	0.0034	centre

a=1.3 and b=0.8

Non-uniform meshes			
h	number of points	maximum error	location of error
0.05	492	0.0195	edge
0.025	1962	0.0060	edge
0.0125	7995	0.0026	centre

From comparing the first set of tables in Chapter 6 for uniform meshes it can be seen that:

by making the tolerance larger the results are roughly similar.

Whereas for fh=a-r:

By changing the tolerance we notice that for a=1.3 and b=0.95 the maximum error is now a lot

bigger than when the tolerance was equal to 0.01.
This is what is to be expected, because when increasing the tolerance the mesh generator produces meshes which still can be iterated to give a better mesh. However the majority of the time it produces worse meshes because the nodes in the generator and still be moved around until it is less than a smaller tolerance where it will give a better quality mesh. In order to be able to analyse this properly more investigation is needed.

Chapter 9

Conclusion

The Modified Helmholtz equation on a mesh was constructed and initially solved (both analytically and numerically). Exploring the solution to this problem using a uniform and non uniform mesh was as expected. We confirmed that the instabilities are from rejecting too many points (when using non-uniform meshes) making it hard for Delaunay to triangulate, or not placing enough points in the region where the solution is blowing up rapidly giving very inaccurate results.

This also depends on a the coefficient of stability and b the radius in which the mesh is convergence mentioned in Chapter 6. In addition we confirmed that as h (maximum size of the triangles) is reduced, the solution becomes more accurate, this is because we are now placing more points in the region hence capturing the slope at r=1.

The Finite Element Method solution reveals that the behaviour at the boundary behaves exactly as expected.

In addition we confirmed that by choosing fh=a-r gives the best results. However more investigation of fh=h^2u" will give better results where the error is almost constant. We have also shown that the smaller the mesh size the smaller the error. Also the smaller the tolerance the better the meshes becomes.

Chapter 10

Future work

Given more time, it would be worth investigating the following points:

Further investigation of the boundary condition

Instead of looking at a Neumann boundary condition, we would look at a Dirichlet boundary condtion to see how the finite element method solution varies with the exact solution. Generation of a mesh with greater concentration of elements at the boundaryh would be desirable.

Stabilty

During many runs of the iterative numerical methods, certain coefficients such as a and b (mentioned in Chapter 6) affects the error. It would be very beneficial to investigate this and see which one gives a better solution.

Change in the tolerance

Given more time, we would investigate further into the change of tolerance and see how this will affect the quality of the mesh and look at the different meshes.

Bibliography

[1] Dr.Stephen Langdon, *Domain embedding boundary integral equation methods and parabolic PDES*; Univerisity of Bath; 1999

[2] Per-olof Persson and Gilbert Strang, *A Simple Mesh Generator in Matlab*; SIAM Review, Volume 46 (2), pp. 329-345; June 2004

[3] *2D Mesh Generation by Delaunay refinement in CGAL*, Wikipedia Article, `http://www-sop.inria.fr/geometrica/courses/slides/CGAL-delaunay-refinement-2D-pa.pdf`, 9

[4] *Delaunay Triangulation*, Wikipedia Article, `http://www-sop.inria.fr/geometrica/courses/slides/CGAL-delaunay-refinement-2D-pa.pdf`, July 2009

mass matrix

http://www.ansys.com/events/proceedings/2004/PAPERS/50.PDF Effect of Mass Matrix Formulation Schemes on Dynamics of Structures

[5] *matlab .m DISTMESH A Simple Mesh Generator in MATLAB*, Mesh Generator Article, `’http://stef2cnrs.wordpress.com/2008/07/27/matlab-m-distmesh-a-simple-mesh-generator-in-` 2009

[6] *Modified Bessel Equation*, Wolfram MathWorld, `http://mathworld.wolfram.com/ModifiedBesselFunctionoftheFirstKind.html`, **Date**

[7] Milton Abramowitz and A. Stegun , *Handbook of mathematical functions*;**Document Date**

[8] *Helmholtz Equation*, Wikipedia Article,
`http://en.wikipedia.org/wiki/Helmholtz_equation`, June 2009.