University of Reading

School of Mathematics, Meteorology & Physics

DATA DEPENDENT MESH GENERATION FOR PIECEWISE LINEAR INTERPOLATION

by

Jonathan Aitken

August 2005

This dissertation is submitted to the Department of Mathematics in partial fulfilment of the requirements for the degree of Master of Science.

Abstract

In this dissertation we draw together several of the commonly used data dependent mesh adaption strategies for meshes consisting of triangular elements. Such triangulations are often of particular interest when considering the numerical solution of mathematical models. The strategies are based on three primary building blocks for mesh adaption. These being, nodal reconnection, nodal movement and mesh refinement.

The supporting theory under which this work is conducted is formally defined and the motivation for developing mesh adaption techniques is presented. Under this framework, we provide implementations of the ideas discussed, and conduct numerical experiments in order to asses their success.

Finally we suggest modifications to the existing methods that may provide more flexible and efficient mesh adaption strategies.

Declaration

I confirm that this is my own work, and the use of all material from other sources has been properly and fully acknowledged.

Acknowledgements

I thank Dr. P. K. Sweby for his supervision, guidance and investment of time over the past year, and in particular during the lifetime of this dissertation. Additionally, thanks are owing to Professor M. J. Baines for his interest and assistance both with this dissertation and the other components of the degree. It has been a delight to have experienced their influence on my education.

To other colleagues and friends, both within the Department of Mathematics and within the wider academic community, I wish to express my gratitude for helping make this year enjoyable and rewarding.

I acknowledge with grateful thanks the financial support by the Engineering and Physical Sciences Research Council which has enabled me to take this course.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The simulation of physical situations by mathematical models is a common tool in both the academic and commercial arenas. The solution to these models is rarely obtainable by analytic methods so numerical techniques are required. Many of these numerical methods employ a discretisation of the spatial domain and it is the nature of this spatial discretisation, or mesh, that we are concerned with in this dissertation.

For many numerical solution techniques the accuracy and efficiency of the method is often a direct consequence of the inherent features of the mesh they depend on. Additionally for models involving a temporal progression, the accuracy of the simulation is often limited by how well the initial data is represented. Clearly, one method of attaining the required accuracy in data representation is to imposed a discretisation with a large number of discrete values. This is at best inefficient, and at worst unusable, given the limitations of computational resources. Here we discuss how we may adapt the spatial discretisation in order for the representation of the initial data to be the best possible for given limitations in computational power.

In Chapter 2 we describe the rules, nature and framework for spatial discretisation by triangles. Chapter 3 gives possible criteria for triangular mesh generation, and how they may be implemented. The practicalities of implementing these techniques are presented in Chapter 4. Finally, we present the results of our numerical experiments and suggest further developments that may be made to these strategies.

Chapter 2

What is a mesh?

The discretisation of a 2D domain can be achieved in many ways. The fundamental idea is to divide the area into a finite number of non-overlapping subdomains. Meshes can be constructed using triangular, quadrilateral or other shaped subdomains [7]. Indeed, the mesh could incorporate a variety of different shaped subdomains. This choice is influenced by the intended use of the mesh.

In order to formalise our discussion let us consider the function $u : \Omega \to \mathbb{R}$ where $\Omega \subset \mathbb{R}^2$ is a bounded connected region and Γ is the boundary of Ω. Here we restrict to the case where Ω is the region enclosed inside a polygon, i.e. Γ is a polygon[1].

In this project we use continuous piecewise linear interpolation over Ω as a numerical method for producing an approximation. Whilst other methods[2] would be equally useful, interpolation provides us with a powerful and efficient tool for testing our methods. A piecewise linear interpolant of u is a function which is piecewise linear on the domain, and equal to u at a finite number of points in the domain.

Formally, we wish to construct a continuous piecewise linear function

$$u_h(x, y) : \Omega \to \mathbb{R}$$

[1] In theory Ω need not have a polygonal boundary as any bounded region $\Omega \subset \mathbb{R}^2$ can be approximated arbitrarily well by a domain with a polygonal boundary.

[2] The classical Finite Element Method with piecewise linear basis functions used for solving a differential equation on Ω would also produce a piecewise linear function over Ω, although the computational cost could be significantly greater. See Section 2.1 for a comparison of the computational costs.

such that

$$u_h(x, y) = U_i \qquad (i = 1, \ldots, N),$$

where $\mathcal{N} = \{\mathbf{n_i} = (x_i, y_i) \in \mathbb{R}^2, \quad i = 1, \ldots, N\}$ is a set of distinct and non-collinear data points and $U = \{U_1, \ldots, U_N\}$ is a vector with $U_i = u|_{\mathbf{n_i} \in \mathcal{N}}$.

In order to produce the interpolant u_h we require a spatial discretisation of Ω by non-degenerate open triangles. In general, we could consider interpolation by piecewise polynomial functions. Our justification for choosing the space of piecewise linear functions is that the interpolant is uniquely defined by its value at the nodes. Should higher order polynomials be considered, we would require more information, such as the gradient at the nodes or values at positions other than the nodes, in order to uniquely determine the interpolant.

We define a suitable non-overlapping covering of Ω by triangles formally by the following definition.

Definition 2.0.1. A **triangulation**, or **mesh**, $\mathcal{T}$ is a sub-division of Ω into a non-overlapping set of open triangles (or **elements**) $\{\tau_1, ..., \tau_N\}$, i.e.

$$\Omega \cup \Gamma = \bigcup_{i=1}^{N} \overline{\tau}_i$$

and

$$\tau_i \cap \tau_j = \emptyset \quad \forall i, j \ s.t. \ i \neq j.$$

For ease of notation, we will refer to $\mathcal{N}$ as the set of *nodes* and $\mathcal{E}$ as the set of *edges* of $\mathcal{T}$.

Although the above definition describes a general triangulation, in order to apply this mesh in the manner we intend, we require a further restriction on the definition.

Definition 2.0.2. A triangulation $\mathcal{T}$ is said to be **conforming** if for any pair of elements in $\mathcal{T}$, the intersection of their closure is either the empty set, their common edge or their common node, i.e.

$$\forall i, j \in \{1, ..., N\} \ \text{ with } i \neq j,$$

$$\overline{\tau}_i \bigcap \overline{\tau}_j = \begin{cases} \emptyset; \\ S \in \mathcal{E}, & \text{where } S \text{ is the common edge of } \tau_i \text{ and } \tau_j; \\ \mathbf{n} \in \mathcal{N}, & \text{where } \mathbf{n} \text{ is the common node of } \tau_i \text{ and } \tau_j. \end{cases}$$

The requirement that a mesh is conforming precludes the possibility of a **hanging node** as illustrated in Figure 2.1. This is only a stipulation since we are considering *continuous* piecewise linear interpolants. Clearly, unless the data values at nodes 1, 2 and 3 are colinear the piecewise linear interpolant would not be continuous.

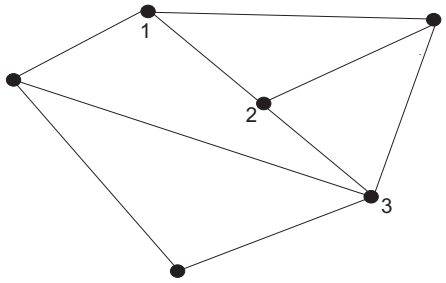

Figure 2.1: Example of a hanging node.

2.1 Non-uniqueness of mesh.

For a given set of nodes, $\mathcal{N}$, in the domain Ω there are many possible triangulations. As an illustration, a convex domain with four nodes on the boundary can be triangulated in two ways, depicted in Figure 2.2.

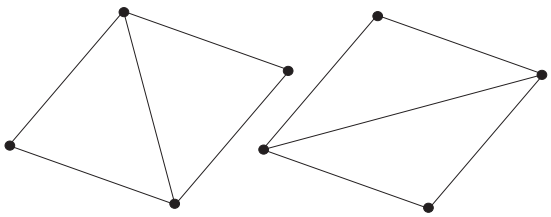

Figure 2.2: Non-unique meshing of four nodes.

In addition to this, the position and quantity of nodes is variable giving an even greater number of possible triangulations. In general the interpolant u_h of u is different for each possible triangulation. This gives us our primary goal:

How should we triangulate Ω so that u_h is the 'best'[3] possible representation of u?

The computational cost of finding u_h by a particular method is usually related to the number of nodes, elements or edges of a triangulation $\mathcal{T}$. If N is the total number of nodes in $\mathcal{T}$ and N_b is the number of nodes on Γ then $T = 2(N-1)-N_b$ is the number of elements and $E = 3(N-1) - N_b$ is the number of edges.

Continuous piecewise linear interpolation The function u_h is planar on each element τ. Therefore for each element τ_i we have an expression for the plane on that element, $P_i = a_i x + b_i y + c_i \quad (i = 1, \ldots, T)$. Hence in order to find the coefficients of the P_i we are required to solve a 3×3 linear system for each element. Whilst this still gives a computational cost of only $O(T)$, the number of FLOPS required may still be significant.

Classical Finite Element Method If u_h is the solution of a problem found using the classical Finite Element Method with the mesh given by $\mathcal{T}$, then the computational cost of the method is dominated by the cost of solving the linear system[1]. If a direct method is used then the computational cost is $O(N^3)$[11].

For the above approximation methods, it is clear that we desire a mesh to have the minimum number of nodes possible to achieve a given accuracy. A large number of nodes has the potential to make the mesh unusable in practice.

In the remaining part of this introduction we discuss possible criteria for constructing meshes.

2.2 Mesh criteria.

A **mesh criterion** is a device that when given two meshes as input indicates which of the two it 'prefers'. The notation $\mathcal{T}_1 < \mathcal{T}_2$ is used to denoted that the triangulation $\mathcal{T}_1$ is preferred to $\mathcal{T}_2$ with respect to a particular criterion.

A criterion is comprised of two parts.

Cost vector For each component of the mesh, $\mathcal{T}$, we assign a real cost function $h_j(\mathcal{T})$. The components of the mesh we consider may be edges, elements or

[3]For methods of comparing the qualities of meshes, see Section 4.5.

nodes, and this choice is dependent upon the nature of the cost function. The value of the cost function for a particular component of the mesh is a gauge of how 'bad' that component is for the approximation. This in turn can be thought of as an indicator of how significant a contribution that part of the mesh makes towards the error in the approximation[4]. The **cost vector** is then $\mathcal{H}(\mathcal{T}) = \{h_1(\mathcal{T}), \ldots, h_M(\mathcal{T})\}$ where M is the number of components in $\mathcal{T}$.

An order defined on $\mathbb{R}^M$ For the criterion to choose between possible triangulations we define an order on the space of cost vectors. Then $\mathcal{T}_1 \leq \mathcal{T}_2$ if and only if $\mathcal{H}(\mathcal{T}_1) \leq \mathcal{H}(\mathcal{T}_2)$ in that particular order.

2.3 Meshes dependent on geometry.

Meshes consisting of triangular elements are of interest to Finite Volume Methods and Finite Element Methods using piecewise linear basis elements. In particular, when using such Finite Element Methods, the geometry of the elements within the mesh is of significant importance since long thin elements lead to a badly conditioned linear system [10]. This gives rise to a class of geometry dependent criteria.

2.3.1 The Min-Max and Max-Min angle criteria.

With the Max-Min criterion we seek the mesh which maximises

$$\alpha(\mathcal{T}) = \min_{\tau_i \in \mathcal{T}} (\text{smallest angle in } \tau_i).$$

Analogously, the Min-Max criterion selects the mesh which minimises

$$\beta(\mathcal{T}) = \max_{\tau_i \in \mathcal{T}} (\text{largest angle in } \tau_i).$$

Using the Max-Min criterion as an example, we define the cost functions

$$h_j(\mathcal{T}_k) = \text{ smallest angle in } \tau_j \qquad (j = 1, \ldots, T) \ (k = 1, 2).$$

[4]How reliable this indicator is is subject to the type of cost function we are using, the measure of error we are considering and what component of the mesh we are examining.

Then $\mathcal{T}_1 \leq \mathcal{T}_2$ if $\alpha_1 \geq \alpha_2$ in the maximum norm where $\alpha_1 = \mathcal{H}(\mathcal{T}_1)$ and $\alpha_2 = \mathcal{H}(\mathcal{T}_2)$.

2.3.2 Delaunay triangulations.

Delaunay triangulations are a well established as geometrically appealing triangulations and there are many fast and efficient methods for producing such a triangulation[6].

Given a set of nodes $\mathcal{N}$, the **Delaunay triangulation** is the triangulation that connects each point to its natural neighbors. A method and algorithm for constructing a Delaunay triangulation is given by Hardwick [6].

A Delaunay triangulation is the dual of the Voronoi tessellation with the vertices of patches at the nodes. Additionally, it has been proved by Sibson [9] that the Delaunay triangulation is in fact the triangulation which is optimal with respect to the Max-Min criterion above.

2.3.3 Spring analogy.

It is possible to think of the edges in a triangulation $\mathcal{T}$ as springs connected to each other at the nodes [8]. If we fix the possition of the nodes on the boundary, then the system of connected springs in the interior of the domain will have a state of equilibrium. The location of the interior nodes in this state of equilibrium is dependent on the value of the spring constants.

The method for finding the state of equilibrium is to examine each interior node, and the 'patch' of elements that have this node in common, at a time. For each patch we move the central node in order to satisfy the local system of springs consisting of the interior edges of the patch. This process is repeated iteratively for all patches until a global state of equilibrium is attained.

If we set all the spring constants to be equal, then this method is synonymous with Laplacian smoothing of the mesh. At each step of smoothing each node is moved to the centre of the patch of elements around it.

These geometry dependent criteria are only dependent on the set of nodes $\mathcal{N}$ and are not influenced by any properties of the underlying function u. As such, the resolution of the mesh may not be fine enough in portions of the domain to

produce a representation of the data that is sufficiently accurate for our needs. In other cases the mesh may be finer in portions of the domain than is required, thus unnecessarily using computational time. Methods developed to avoid these scenarios are discussed below.

2.4 Meshes dependent on data.

To improve the geometry-dependent criteria above we wish to examine criteria which are dependent on the underlying data in some way. In many cases we may not know the form of u explicitly, but only the value of u at the nodes. However, even in these cases, we may know some property of u which can be used to formulate a criterion. Data dependent criteria are comprised of the same components as in Section 2.2, however the cost functions and therefore the cost vectors are functions of the underlying data as well as the mesh. That is $\mathcal{H} : (\mathcal{T}, u) \longmapsto \mathbb{R}^M$ with

$$\mathcal{H}(\mathcal{T}, u) = \{h_1(\mathcal{T}, u), \ldots, h_M(\mathcal{T}, u)\}.$$

A mesh that is optimal with respect to this criterion can then be sought.

2.4.1 Min error criterion.

If we know u explicitly we can calculate the error between u and u_h. A possible criterion is then one which seeks a triangulation that minimises this error.

Definition 2.4.1. Let $e = u - u_h$ be the error in the interpolant.

We wish to minimise e in some norm by changing the mesh. An obvious choice would be to seek to minimise the quantity

$$\|e\|_{L2} = \int_\Omega |u - u_h|^2 \, \mathrm{d}\Omega.$$

2.4.2 NC1 criteria.

Dyn et al[4] present a class of data dependent criteria that can be used in cases where u is not explicitly known. Here we present the background motivation for the development of these criteria.

In many cases we may know that $u \in C^1$. i.e. u has a continuous first derivative. Furthermore we assume that u is 'smooth' relative to u_h. This can be regarded as assuming that the total variations of u_h and u are not significantly different on each element. A common class of criteria is those that attempt to get close to this 'smooth' property by examining the behaviour of the interpolant at the edges within the mesh. This class of criteria is termed 'nearly C^1' (NC1).

2.4.3 Spring analogy criteria.

We can adapt the stretegy of a system of springs set out in Section 2.3.3 to take account of the underlying data. The most obvious method for doing this is to change the spring constants to reflect some known property of the function we wish to represent.

Malcolm [8] suggests several methods where the spring constants are functions of u_{ss} and/or u_s where u_{ss} is the directional second derivative along the edge S and u_s is the first derivative.

Chapter 3

Methods of mesh adaption.

A popular method for the production of a data dependent mesh is to start with a geometrically optimal mesh, and adapt this mesh so that it becomes optimal with respect to some data dependent criterion. There are many ways in which a mesh can be adapted. These are shown in Figure 3.1 where Figure 3.1 (a) is the initial mesh. The methods shown are:

Changing the connectivity of a mesh We maintain the nodal positions, but change the connection of the nodes[1]. Shown in Figure 3.1 (b).

Mesh refinement We add nodes to the mesh and add connectivity so that the mesh retains conformity[2]. Shown in Figure 3.1 (c)

Nodal movement We retain the connectivity of the mesh, but move the nodes. Shown in Figure 3.1 (d)

Other methods exist where the current mesh is discarded and a new mesh is generated using information obtained from the current mesh. Examples of such methods are advancing front techniques. For further information see Eriksson et al [5] and Malcolm [8].

3.1 Changing connectivity.

In order to discuss the class of criteria associated with this adaption method we require some terminology. Again the notation $\mathcal{T} < \mathcal{T}'$ is used to denote that the

[1]See Section 3.1.
[2]See Section 3.2.

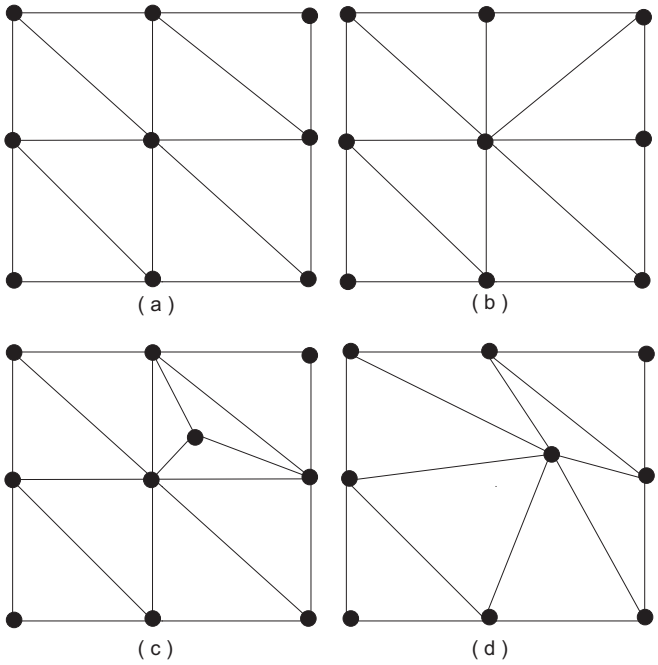

Figure 3.1: Methods of mesh adaption. (a) Initial mesh. (b) Nodal reconnection. (c)Nodal insertion. (d)Nodal movement.

triangulation $\mathcal{T}$ is preferred to $\mathcal{T}'$ with respect to a particular criterion.

Since there are a finite number of triangulations of Ω it is natural to ask which of them is the best with respect to a given criterion. We define this **optimal** triangulation formally below.

Definition 3.1.1. Given a specific criterion, an **optimal** triangulation, $\mathcal{T}_{opt}$ of Ω is such that

$$\mathcal{T}_{opt} \leq \mathcal{T} \qquad \text{for all possible triangulations } \mathcal{T} \text{ of } \Omega.$$

Although an optimal triangulation is guaranteed to exist[3] due to the finite number of possible triangulations, it may be difficult to find computationally. Dyn et al [4] allege that a localised optimality is sufficient. The method of 'edge swapping' they use for finding this locally optimal mesh is detailed below.

Definition 3.1.2. Let S be an internal edge of $\mathcal{T}$ and let Q be the quadrilateral consisting of the union of the two elements having S as their common edge. S

[3]Although an optimal triangulation always exists, it is not guaranteed to be unique. It is feasible that several triangulations may be optimal. If this occurs one strategy is to use a geometry dependent criterion to select which one is optimal.

is said to be a **locally optimal edge** if:

- Q is not strictly convex;
- $\mathcal{T} \leq \mathcal{T}'$ where $\mathcal{T}'$ is obtained by replacing S by the other diagonal of Q.

If Q is not strictly convex then swapping S for the other diagonal of Q would lead to overlapping elements which is not permitted by Definition 2.0.1. This is demonstrated in Figure 3.2. If we consider the union of the elements described by nodes 1-2-4 and 1-4-3 in Figure 3.2 (a), then swapping their common edge, 1-4, for the other diagonal, 2-3, leads to two overlapping elements at 1-2-3 and 2-3-4 as shown in Figure 3.2 (b).

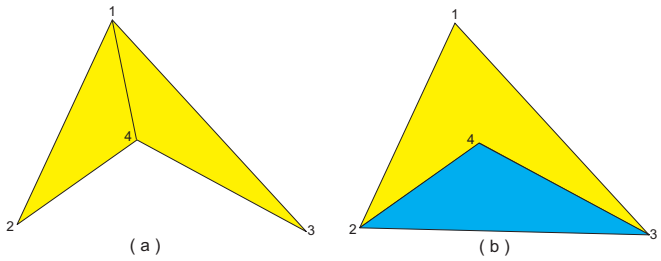

Figure 3.2: Prohibited edge swapping of a non-convex quadrilateral.

Definition 3.1.3. A *mesh* is said to be **locally optimal** if all the internal *edges* are locally optimal.

This method of edge swapping in order to find a locally optimal mesh provides an algorithm known as the *local optimisation procedure* (LOP) as used by Dyn et al [4].

Algorithm 3.1.4. Local Optimisation Procedure.

1. Construct an initial mesh $\mathcal{T}$ of Ω.
2. For each internal edge S in $\mathcal{T}$ check if the edge is locally optimal. If not, swap it for the other diagonal of the quadrilateral formed by the union of the elements having S as their common edge.
3. If no edge swaps have been made the mesh is locally optimal so end the procedure. Else return to Step 2.

Clearly the resultant mesh of Algorithm 3.1.4 is dependent upon the order in which the edges are inspected in Step 2. Furthermore, Dyn et al [4] claim that the LOP converges after a finite number of steps since there are a finite number of triangulations of Ω. However it is possible for the algorithm to enter a cyclic process of edge swapping and not terminate. In order to avoid this scenario we need to impose a further constraint that we do not allow an edge swap if it produces a mesh that has previously been constructed.

3.1.1 Edge based data dependent criteria suitable for use with the LOP.

By their nature, criteria in the NC1 class are suited to connectivity adapting techniques since they examine properties of the interpolant at interior edges.

Here we present two of the cost functions and two measures given by Dyn et al [4] that conform to the form stipulated in Section 2.4.

Let τ_1 and τ_2 be two elements which have $S \in \mathcal{E}$ as a common edge. Then define $P_1(x,y) = a_1x + b_1y + c_1$ and $P_2(x,y) = a_2x + b_2y + c_2$ as the restrictions of the interpolant to the elements τ_1 and τ_2 respectively. That is $P_i := u_h|_{\tau_i} \quad i = 1, 2$.

3.1.2 Jump in normal derivatives (JND).

This cost function measures the jump in the normal derivatives of P_1 and P_2 across the edge S. We calculate this by

$$h_S(\mathcal{T}, u_h) = |n_x(a_1 - a_2) + n_y(b_1 - b_2)|$$

where $n = (n_x, n_y)^T$ is a unit vector that is orthogonal to the direction of S.

3.1.3 Angle between normals (ABN).

This cost function measures the angle between the normals of P_1 and P_2 calculated by

$$h_S(\mathcal{T}, u_h) = \theta = \cos^{-1}(A)$$

where

$$A = \frac{a_1 a_2 + b_1 b_2 + 1}{((a_1^2 + b_1^2 + 1)(a_2^2 + b_2^2 + 1))^{\frac{1}{2}}}.$$

As $\mathcal{H}(\mathcal{T}, u_h)$ tends to the zero vector, the interpolant u_h tends to being smooth.

In order to compare the cost vectors produced by the above cost functions we require a measure. Two successful measures presented by Dyn et al [4] are the L_1-Norm and the L_2-Norm. That is we use the measures

$$\|\mathcal{H}\|_{L_1} = \sum_{i=1}^{M} |h_i|$$

and

$$\|\mathcal{H}\|_{L_2} = \left(\sum_{i=1}^{M} |h_i|^2 \right)^{\frac{1}{2}}.$$

For example if we were considering the L_1-Norm then $\mathcal{T}_1 \leq \mathcal{T}_2$ if and only if

$$\|\mathcal{H}(\mathcal{T}_1, u_h)\|_{L_1} \leq \|\mathcal{H}(\mathcal{T}_2, u_h)\|_{L_1}$$

3.1.4 Efficiency.

By swapping an edge, the mesh and therefore the interpolant is only changed locally so we can reduce the size of the cost vector. It will therefore only contain the cost functions relating to the edges across which the cost function may be altered. In practice for our purpose it means $\mathcal{H}$ only needs to have five elements and the interpolant only needs to be considered on six adjoining elements. This is shown in Figure 3.3.

3.2 Mesh refinement.

In order for a linear interpolation to be useful, the resolution of the nodes needs be sufficiently high enough to reflect the features of the underlying data. In some cases it may be necessary to add nodes to the mesh. This is know as **mesh refinement**.

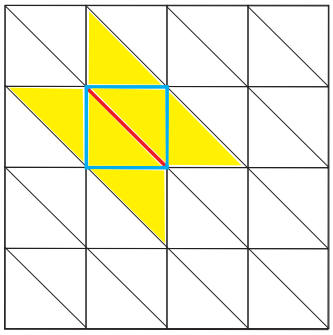

Figure 3.3: The localised effect of edge swapping. If we are considering the edge highlighted in red, then the only other edges that could be affected by changing this edge are those highlighted in blue. This means we need only consider the interpolant restricted to the elements highlighted in yellow.

Mesh refinement should be the last resort of a mesh adaption strategy since the efficiency of an approximation technique is often directly linked with the number of nodes in a mesh. However, if a given accuracy can not be attained through other mesh adaption methods, then a mesh with a higher resolution should be used. A higher resolution mesh could be constructed from scratch[4] and then possibly adapted with respect to the required criterion. However, for many possible features of underlying data a higher resolution is required in some areas than in others. In this case nodes could be added to areas of the mesh where it is known that a higher resolution is required. This is an attempt to keep the number of new nodes to a minimum. Most mesh refinement strategies are operations on the elements of an existing mesh, in that they form smaller elements by adding new nodes and subdividing one or more elements. Whereas our criteria are functions of either elements, nodes or edges.

3.2.1 Element based criterion.

In the case where a particular criterion is a function of elements such as the minimum error criterion (Section 2.4.1) we can use the cost vector to determine which elements should be refined by selecting to refine a proportion of the total number of elements which have the worst cost as per the criterion. The proportion of elements we wish to refine can be chosen to reflect the detrimental

[4]See Eriksson et al [5] for more information about using a mesh function to construct a finer mesh. The data dependent component of this mesh generation strategy resides in the construction of the mesh function.

effect on the efficiency of the approximation methods by having more nodes in the mesh. If we are using a computationally expensive method of approximation such as the Finite Element Method, we may wish to refine less elements than if we were using a relatively cheap method.

3.2.2 Edge based criterion.

When the criterion is a function of edges the cost vector will have an element corresponding to each edge of a mesh. There are two approaches to determine which elements to refine when using this type of cost vector.

By edge We select a proportion of the edges that have the worst contribution to the cost vector and for each of these edges we set the elements which have this edge as their common edge for refinement. This has the disadvantage that we are refining two elements when only one of these may be contributing significantly to the error.

By element We define a temporary additional cost vector that has an entry for each element of the mesh. The entries are the average in some sense of the costs of each of its edges as defined by the criterion. We then proceed as in Section 3.2.1 using the temporary cost vector.

An example of a possible average could be the straight arithmetic average, $A(\tau) = \frac{1}{3}\sum_{i=1}^{3} h_i$ where $h_i \quad i = 1, \ldots, 3$ are the cost functions associated with the three *edges* of τ.

3.2.3 Node based criterion.

If a criterion is nodal based then similar approaches can be taken.

By node We select a proportion of the nodes which have the worst contribution to the cost vector and for each one we mark the elements that have this node in common for refinement. In the example given in Figure 3.4, six elements are refined when only one node is selected. This could lead to significant over refinement.

By element Again we define a temporary cost vector that has an entry for each element of the mesh. As before, the entries are the average in some sense

of the costs of each of its edges as defined by the criterion. Analogously to before, an example of a possible average could be $A(\tau) = \frac{1}{3}\sum_{i=1}^{3} h_i$ where $h_i \quad i = 1, \ldots, 3$ are the cost functions associated with the three *nodes* of τ. We then proceed as in Section 3.2.1 using this temporary cost vector.

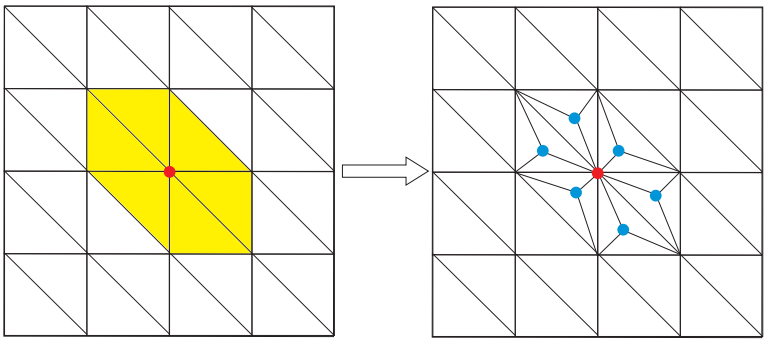

Figure 3.4: Refinement of a patch of elements.

There are many ways in which to refine, and therefore add resolution to an existing mesh. Some of the most common methods are detailed below.

3.2.4 Edge bisection.

For each element we wish to refine we create three new nodes, one at the midpoint of each edge. These nodes are then connected so that the original element is subdivided into four congruent elements of equal area. This is shown in Figure 3.5.

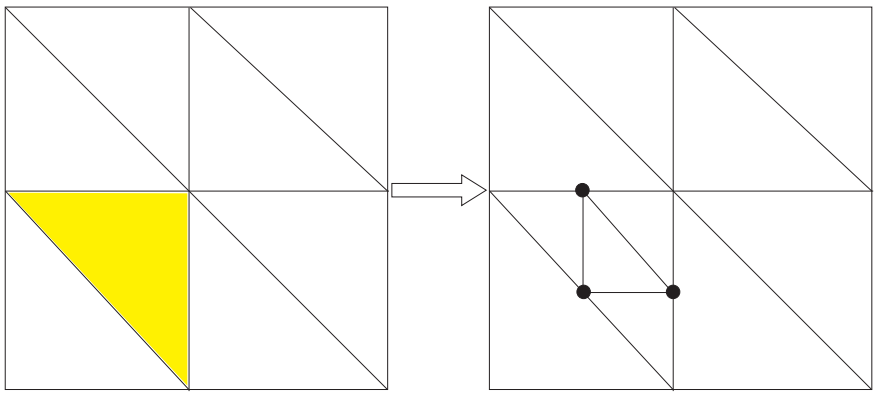

Figure 3.5: Mesh refinement: Edge bisection.

If only a subset of the set of elements are refined then the mesh will have hanging nodes as can be seen in Figure 3.5. In order to produce a conforming

mesh we bisect the elements on which the hanging node lies, shown in Figure 3.6. The bisection of *edges* refinement is known as "red" refinement and the bisection of *elements* refinement used to produce a conforming mesh is known as "green" refinement[1]. It is possible that an element has hanging nodes on two or three of its edges. In these cases it is not possible to "green" refine these elements, and the element is "red" refined as shown in Figure 3.7.

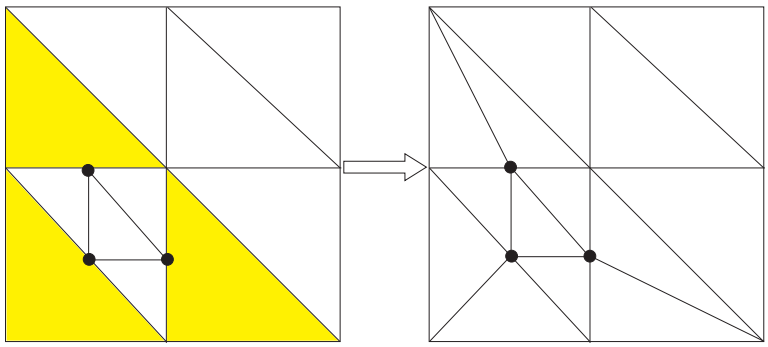

Figure 3.6: "Cleaning up" hanging nodes produced by edge bisection refinement.

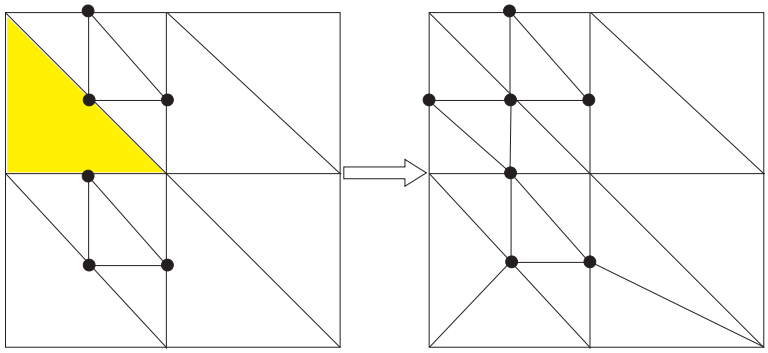

Figure 3.7: "Red" refinement to remove hanging nodes produced by edge bisection refinement.

The method of edge bisection is appealing since the "red" refinement retains the geometrical properties of the elements since the new elements are congruent to the original element. This property is however lost in the "green" refinement. If our method of mesh adaption required the avoidance of long, thin elements this method of refinement could be problematic. Figure 3.8 shows that successive refinement will result in long thing elements. A possible strategy to avoid this [1] is to set the condition that an element originally produced by "green" refinement can not be refined further. Instead the original green refinement is removed and the element is "red" refined.

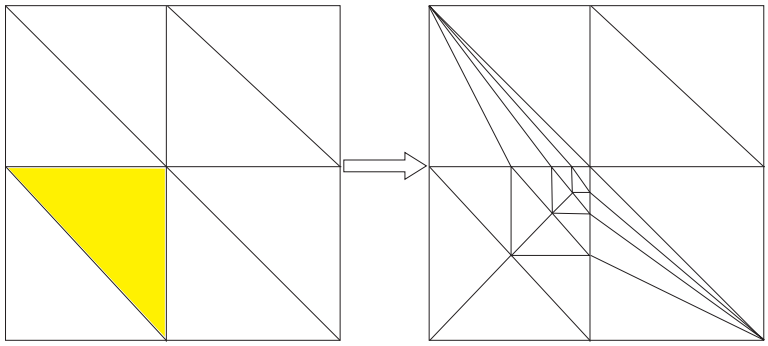

Figure 3.8: Long thin elements produced by edge bisection refinement.

3.2.5 Midpoint node insertion.

Using this method of refinement, an element is refined by adding a node at the centre of the element and then connecting it to the nodes of the original element, as shown in Figure 3.9. This results in three new elements replacing the original element without the possibility of producing any hanging nodes.

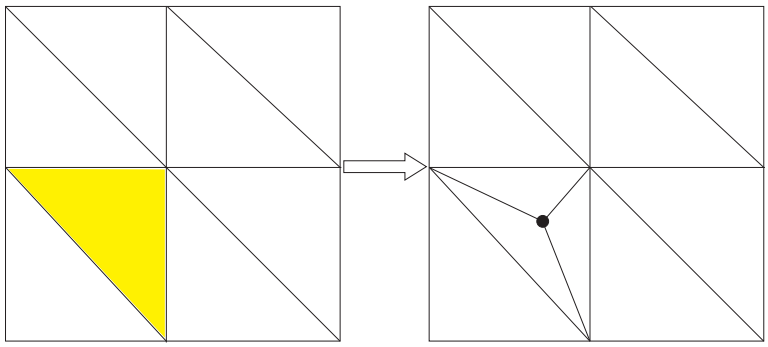

Figure 3.9: Midpoint node insertion.

This method of mesh refinement will not retain the geometrical properties of the mesh. If successive refinement stages are used it could produce long thin elements.

3.2.6 Element bisection.

Here we bisect an element by adding a single node at the midpoint of the longest edge of the element as shown in Figure 3.10. This method will produce hanging nodes and this is dealt with in a similar way to the edge bisection refinement strategy in Section 3.2.4.

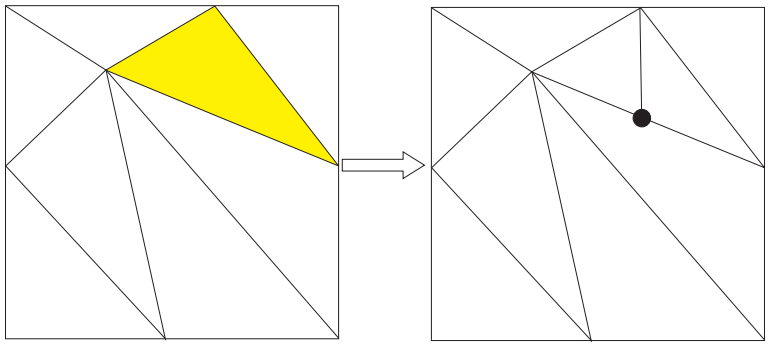

Figure 3.10: Mesh refinement by bisecting the longest edge.

An advantage of this method is that, baring the 'green' refinement process, the angles of refined elements are always bounded away from 0 and π thus providing be a useful tool if we were to consider geometrically dependent criteria such as the Min-Max criterion in Section 2.3.

3.3 Moving nodes.

When retaining the connectivity of a mesh and moving the nodes it is possible for the mesh to become 'tangled'. This occurs when a node is moved outside the boundary of the patch of elements having that node in common. A simple test for this is that the sum of the area of each element must be equal to the area of the domain. If this is not the case then the mesh has become tangled.

It may also be necessary to impose the condition that the mesh does not become 'nearly tangled'. A test for this would be to check whether the area of an element is reduced to below a threshold value by a node movement step.

Chapter 4

Practicalities

Now that we have defined the components of our mesh adaption methods, we wish to construct numerical experiments with which to test and compare various strategies. In this chapter we detail several of the practicalities of conducting these numerical experiments. These are:

- The nature of the domain Ω we use;
- The data sets $\mathcal{N}$;
- Which test functions, u;
- The initial meshes we use on the data sets;
- The data structures we use to represent the mesh;
- How we assess the quality of a mesh.

4.1 Test domain and data sets.

In order to test the methods detailed above we set Ω to be the unit square. This will allow for easy mapping to other more complicated domains should it be required. On this domain we define two sets of nodes.

4.1.1 81 point data set.

The first data set we use is an 81 point set where all the nodes are evenly spaced. This data set is shown in Figure 4.1. The 81 nodes are produced by starting

with the mesh shown in Figure 4.2 and adding nodes using three successive steps of mesh refinement. In each step of the process we refine all the elements in the current mesh by the *edge bisection* method described in Section 3.2.4.

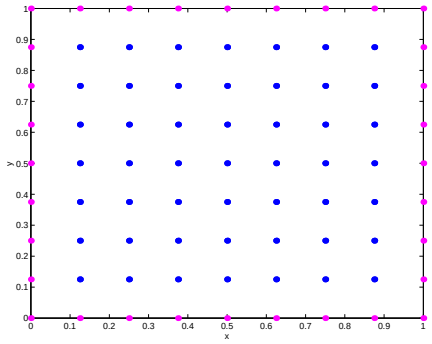

Figure 4.1: 81 point evenly distributed data set.

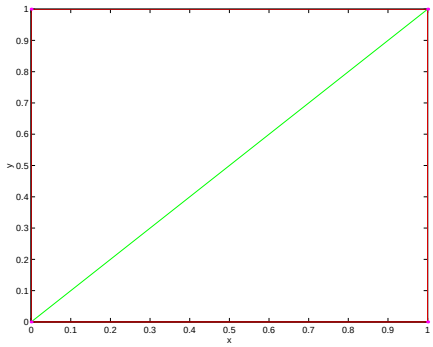

Figure 4.2: 4 node initial mesh.

By this method of refinement, a further step of refinement would produce an evenly spaced data set with 289 data points.

4.1.2 33 point data set.

In order to test some of the features of our mesh adaption methods, it is useful to use a data set which does not have an even distribution of the nodes. For this purpose we employ the 33 point data set used by Malcolm [8]. The positions of the nodes are given in Appendix A and are shown in Figure 4.3.

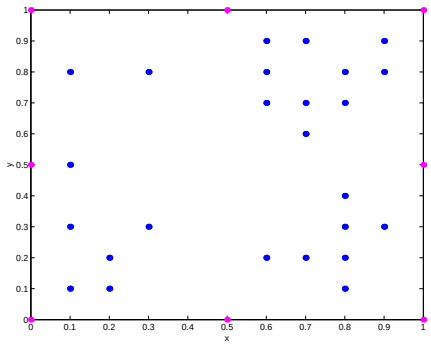

Figure 4.3: 33 point data set.

4.2 Test functions.

In order to implement the data dependent strategies we require the data vector U. This vector is sampled from a test function u at the nodes. As test functions we use four of the most illustrative of the functions Malcolm [8] uses. We preserve the labelling used by Malcolm.

SR1 The 'SR1' function consists of two plateaux with a smooth ramp between them running downhill along the $y = 1 - x$ line. The ramp is produced by the tanh function

$$SR1(x, y) = \frac{\tanh(9y - 9x) + 1}{9}.$$

It should be noted that 'SR1' has a continuous first derivative. The function is shown in Figure 4.4.

SR2 The 'SR2' function also consists of a smooth ramp, however the direction of the slope is perpendicular to the direction of the 'SR1' function. 'SR2' is produced by

$$SR2(x, y) = \frac{\tanh(9y + 9x - 9) + 1}{2}$$

and is shown in Figure 4.5.

SR3 'SR3' is produced by

$$SR3(x, y) = 1 + \tanh(-3g(x, y))$$

where

$$g(x, y) = 0.595576(y + 3.79762)^2 - x - 10.$$

'SR3' has contours along the lines $g = \text{const}$. It is shown in Figure 4.6.

DD1 The 'DD1' function is of a different flavour. Firstly, while the function itself is continuous, its first derivative is not. Secondly, the function has two features of interest. It contains a 'ramp' and a 'spike'.

$$DD1(x, y) = \begin{cases} 1 & y - \phi \geq \frac{1}{2} \\ 2(y - \phi) & 0 \leq (y - \phi) < \frac{1}{2} \\ \frac{1+\cos(4\pi r)}{2} & r \leq \frac{1}{4} \\ 0 & \text{otherwise,} \end{cases}$$

where

$$\phi = 2.1x - 0.1$$

and

$$r = \sqrt{\left(\left(\phi - \frac{3}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2\right)}.$$

'DD1' is shown in Figure 4.7.

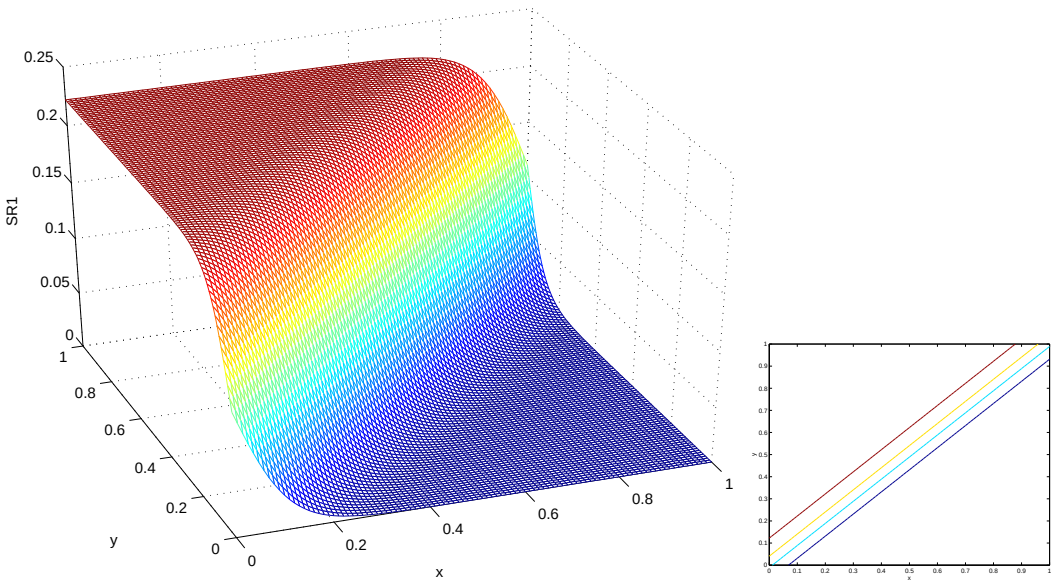

Figure 4.4: Test function 'SR1'.

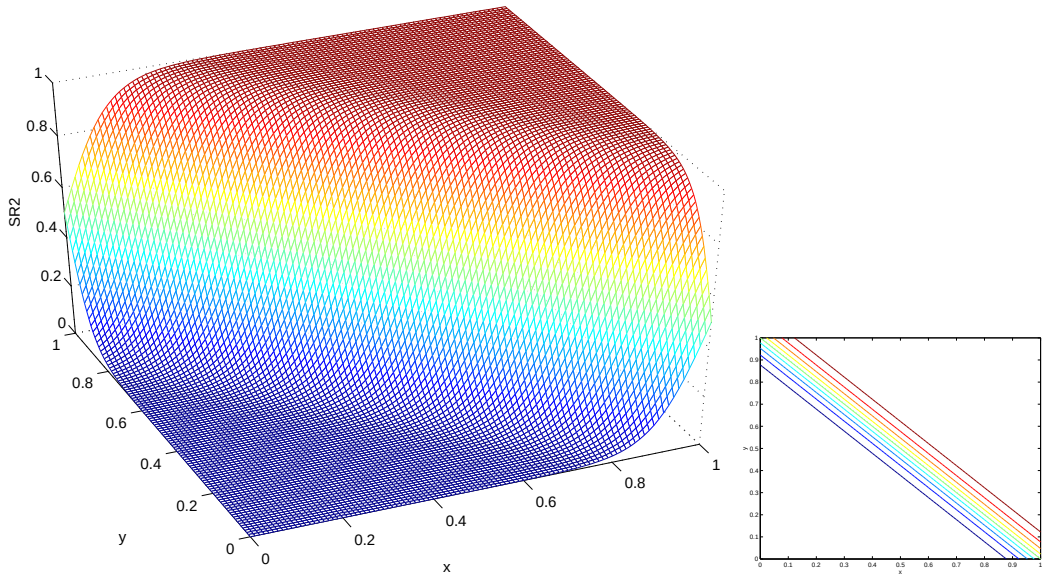

Figure 4.5: Test function ‘SR2’.

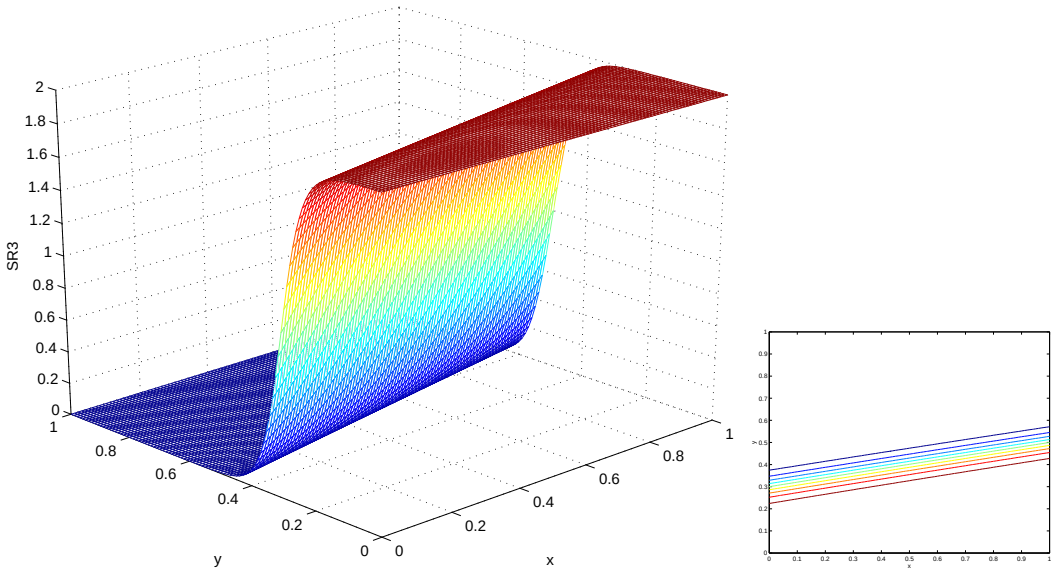

Figure 4.6: Test function ‘SR3’.

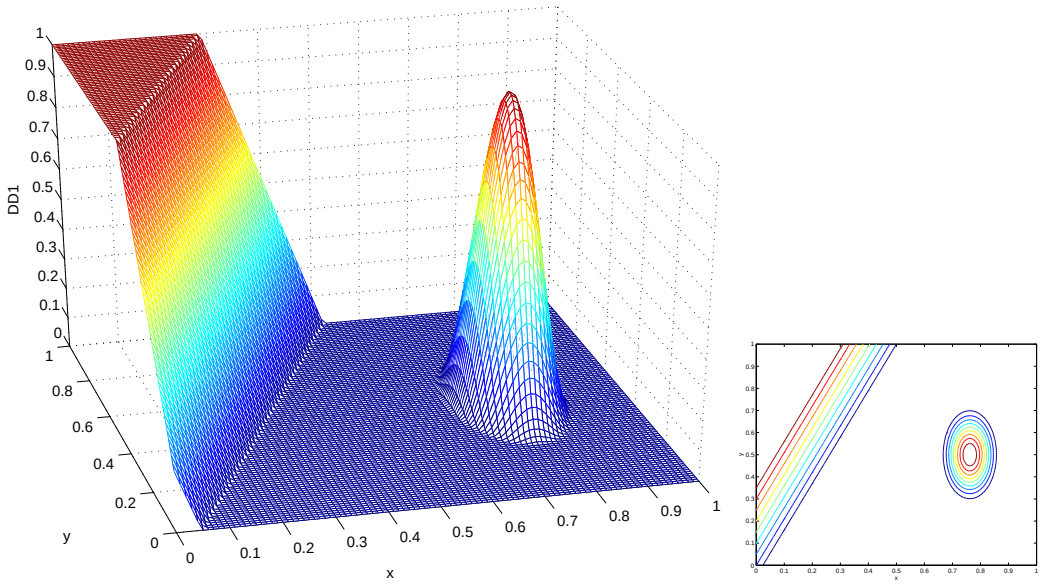

Figure 4.7: Test function ‘DD1’.

4.3 Initial meshes.

One of the first steps when using a mesh adaption procedure is to construct a starting mesh. Throughout this project we use the Delaunay triangulation on a given set of nodes as our initial mesh. The Delaunay mesh is optimal in a non data dependent sense[1].

The Delaunay mesh for the 33 point data set from Section 4.1 is shown in Figure 4.8.

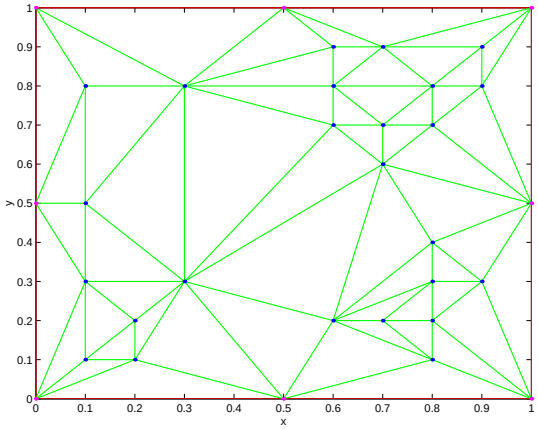

Figure 4.8: Delaunay mesh on the 33 point data set.

For the 81 point data set a Delaunay mesh is given in Figure 4.9. It should be noted that the Delaunay mesh on the 81 point data set is not unique due to the equidistribution of the nodes. This particular mesh is chosen as it helps to illustrate interesting behaviour of the mesh adaption routines.

4.4 Data structures.

In order for the implementation of these methods to be portable and easily usable by other approximation methods, we have chosen to utilise the same data structures as those used by the Numerical Analysis Group at the University of Bath in their suite of Finite Element codes. As such we have three arrays, one each for the nodes, edges and elements.

Nodal array The nodal array has a row for each node. The rows contain the x and y positions of the nodes and a third value that is used as a marker

[1]See Section 2.3.2.

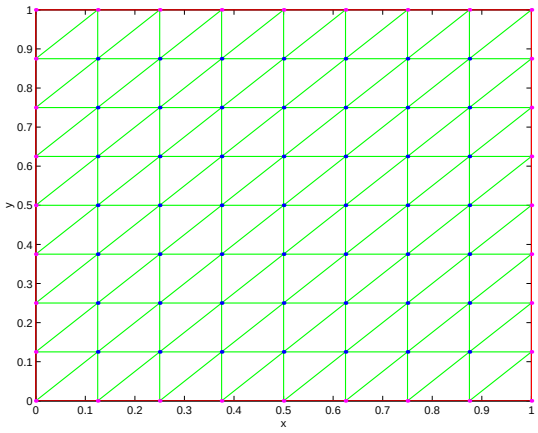

Figure 4.9: Delaunay mesh on the 81 point data set.

for whether the node is on the boundary of the domain. The row index is used as the node number.

Edge array The edge array has a row for each edge. Each row has three entries. The first two are the numbers of the nodes which are connected to form the edge and again the third entry of each row is used to mark whether the edge forms part of the boundary.

Element array The element array has a row for each element. Each row contains the three node numbers for the nodes that are connected to make up the boundary of the element. We must note here that the nodes of each element are numbered anti-clockwise. This is to ensure compatibility with other methods.

4.5 Assessing the quality of a mesh.

We wish to assess the quality of a generated mesh with respect to:

- How well the interpolant u_h approximates u.
- How useful the mesh is for other purposes.

4.5.1 Accuracy of the interpolant.

There are many way of gauging the accuracy of the interpolant.

By the error If we know u explicitly then we can investigate $e = |u - u_h|$. This can either be plotted as a surface or we can evaluate $\|e\|$ where $\|.\|$ represents some norm. In this report I will use the L_2 error as an indicator of how good the interpolant is.

As such we wish to calculate $\|e\|_{L_2} = \left(\int_\Omega |u - u_h|^2\right)^{\frac{1}{2}}$. We do this by imposing a 200×200 grid on our domain and using a 2D trapezium rule to evaluate the integral. Although there are more efficient quadrature methods for calculating this integral, we are not concerned with the computational cost of this part of the procedure and the accuracy of this method when using a 200×200 grid is sufficient for our purposes.

By the criterion For the mesh produced by a particular criterion we can construct the cost vector and examine it's properties. Again we could compute the norm of this vector, denoted by $\|.\|_{CRITERION}$, and use this indicator to compare meshes. If we know u explicitly and we are using the minimum error criterion, then it is clear that comparing meshes in this way, is synonymous with the above method. This is because in this case

$$\begin{aligned}
\|u - u_h\|_{L_2} &= \left(\int_\Omega |u - u_h|^2\right)^{\frac{1}{2}} \\
&= \left(\sum_{\tau_i \in \mathcal{T}} \left| \left(\int_{\tau_i} |u - u_h|^2\right)^{\frac{1}{2}} \right|^2\right)^{\frac{1}{2}} \\
&= \|u - u_h\|_{MIN-ERROR}\,.
\end{aligned}$$

4.5.2 Application of the mesh with other methods.

If the mesh is to be applied to another method such as the Finite Element Method it may be desirable that the mesh conforms to a different set of criteria than those that purely attempt to optimise the representation of the data. In the example of the Finite Element Method we want the internal angles of the elements of the mesh to be bounded away from 0 and π. If we have elements which are long and thin, then the linear system produced may be ill-conditioned and also known error bounds may become unusable.

We employ three methods for assessing the geometrical properties of the mesh. These are:

Skewness The measure of *skewness* used by Malcolm [8] is defined on each element by $sk(\tau) = \frac{1}{3}\sum_{i=1}^{3}\left|\frac{\pi}{3} - \theta_i\right|$. The skewness of an element is a measure of departure from equiangularity.

Aspect ratio The *shape regularity* or *aspect ratio* of an element τ is given by Bank et al [3] as

$$ar(\tau) = \frac{4\sqrt{3}\mu(\tau)}{|l_1|^2 + |l_2|^2 + |l_3|^2}$$

where $\mu(\tau)$ is the area of τ and the vectors $l_i \quad i = 1, \ldots, 3$ are the vectors of the edges of τ with anticlockwise orientation.

Maximum angle We measure the maximum internal angle of each element as an additional indicator of the geometry of the elements.

For each of the sets of numerical results in Section 5 we give indications as to the mesh's "quality" by the above methods.

Chapter 5

Results

In this section we give the most illustrative examples of the methods described above. For each set we state which starting mesh we have used from Section 4.3 and show the final mesh and data representation after the adaption procedure. In addition to this we give relevant numerical measures of the mesh's quality as described in Section 4.5.

5.1 Initial data representation.

Before we present the results of our mesh adaption routines we show the data representation on the starting meshes. These are shown in Figures 5.1 to 5.7. In Tables 5.1 and 5.2 we also give the measures of skewness, aspect ratio and maximum angle for each mesh, and the L_2 error for each initial interpolant. These are included as benchmarks against which we assess the success of the adaption routines.

In order to provide a meaningful representation of 'DD1' we require a higher resolution data set than the 33 point data set. In these cases we produce our starting mesh by uniformly refining the Delaunay mesh on the 33 point data set using the bisection of edges refinement method from Section 3.2.4. This mesh is shown in Figure 5.6. In Figure 5.7 we show the continuous piecewise linear interpolant of the 'DD1' function on this refined mesh.

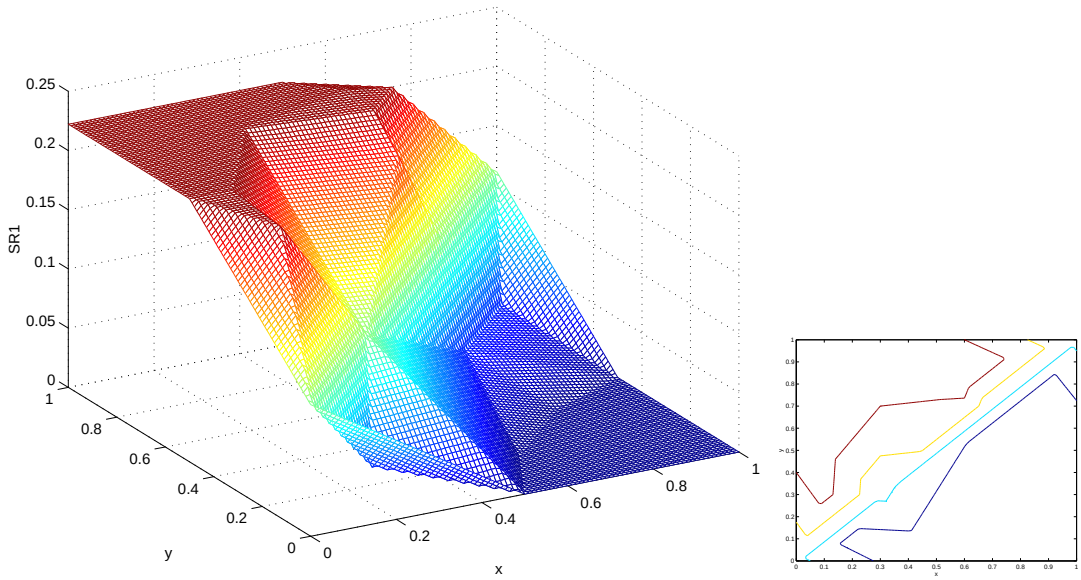

Figure 5.1: The continuous piecewise linear interpolant of the 'SR1' function on the Delaunay mesh of the 33 point data set.

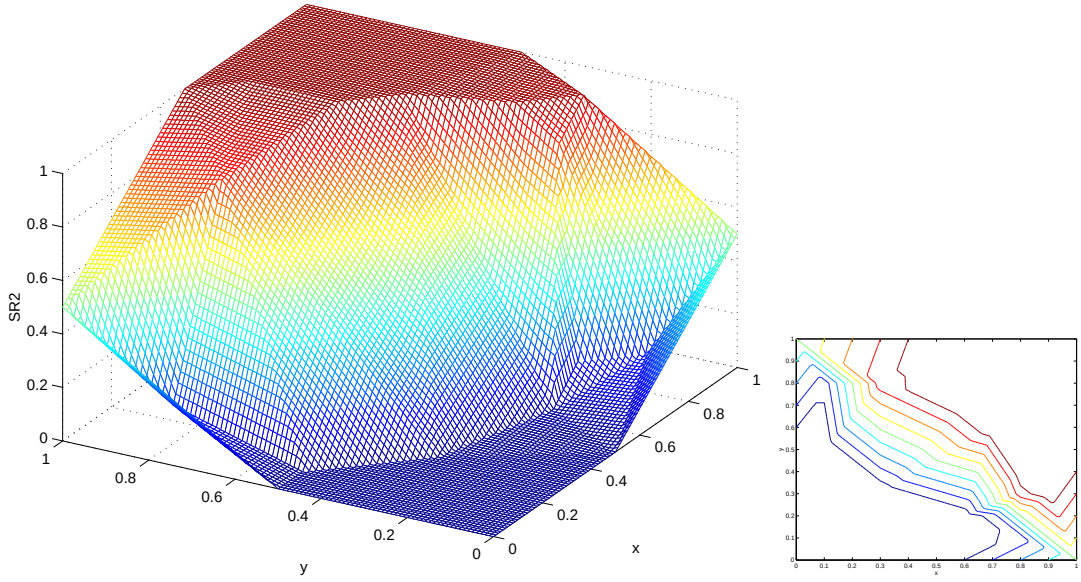

Figure 5.2: The continuous piecewise linear interpolant of the 'SR2' function on the Delaunay mesh of the 33 point data set.

Measure	**81 pt data set** Mean, Max	**33 pt data set** Mean, Max	**Refined 33pt data set** Mean, Max
Maximum angle	90, 90	103.0, 135.0	103.0, 135.0
Skewness	0.3491, 0.3491	0.5210, 0.8727	0.5210, 0.8727
Aspect ratio	0.8660, 0.8660	0.6831, 0.8661	0.6831, 0.8661

Table 5.1: Geometrical mesh quality measures for the initial meshes.

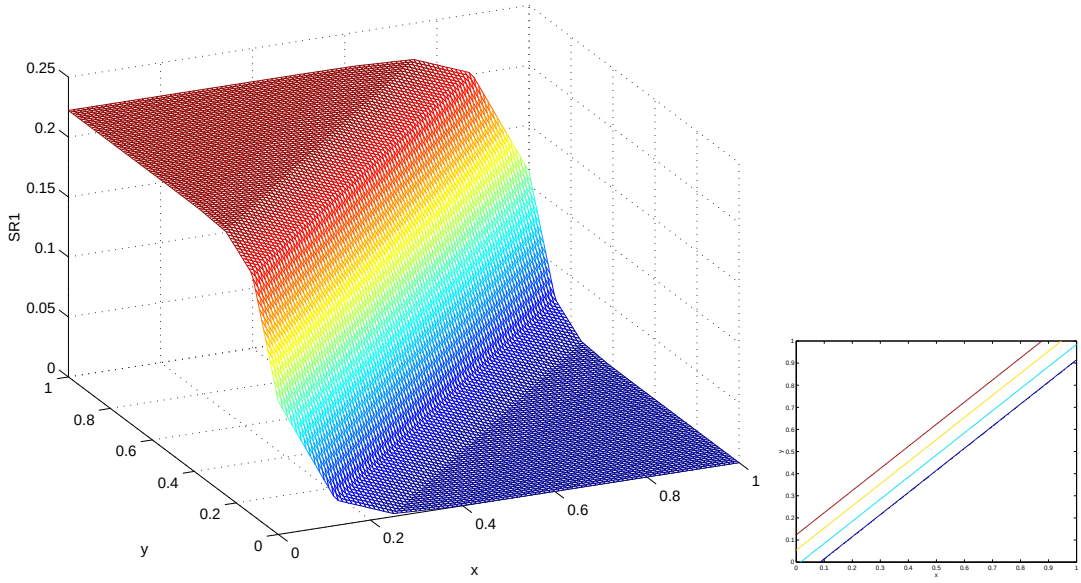

Figure 5.3: The continuous piecewise linear interpolant of the 'SR1' function on the Delaunay mesh of the 81 point data set.

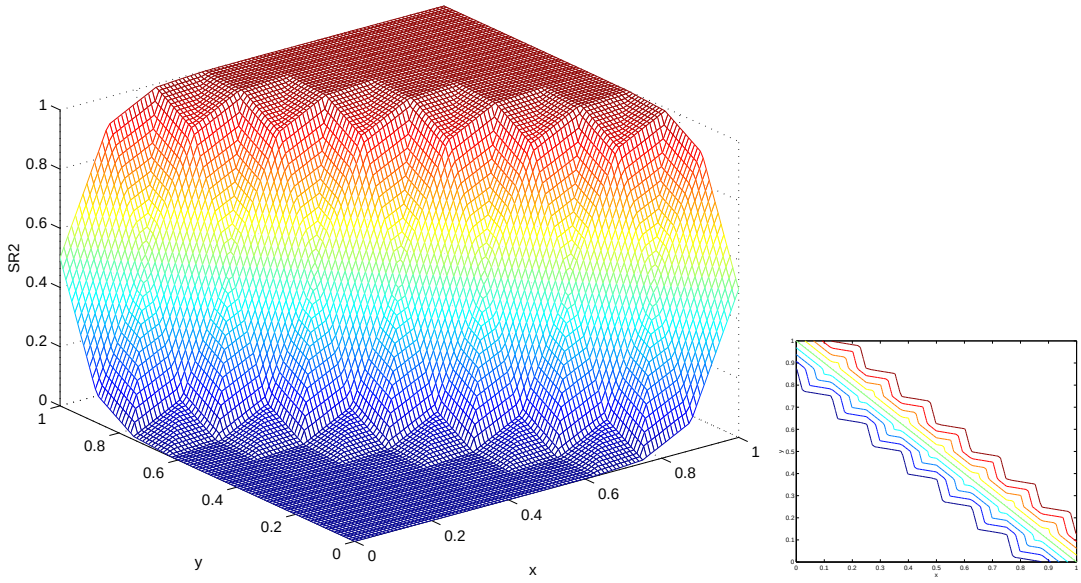

Figure 5.4: The continuous piecewise linear interpolant of the 'SR2' function on the Delaunay mesh of the 81 point data set.

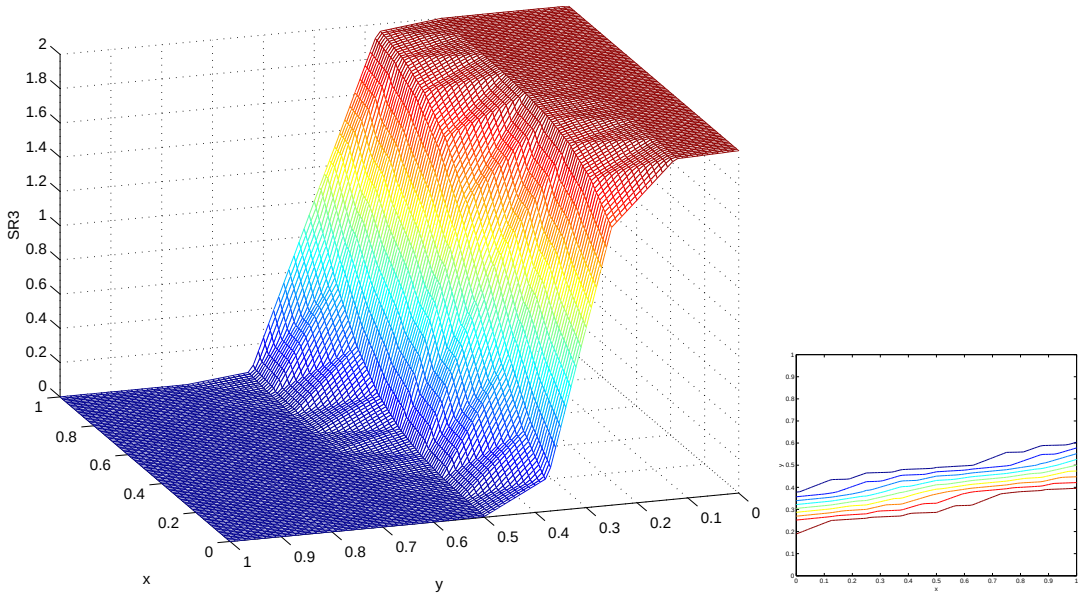

Figure 5.5: The continuous piecewise linear interpolant of the 'SR3' function on the Delaunay mesh of the 81 point data set.

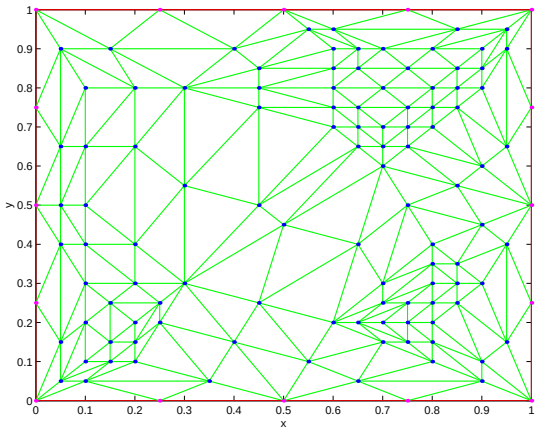

Figure 5.6: Refined 33 point Delaunay mesh.

L_2 errors of the interpolant			
Function	**81 pt data set**	**33 pt data set**	**Refined 33pt data set**
SR1	0.004325	0.02164	-
SR2	0.04384	0.1131	-
SR3	0.05970	-	-
DD1	-	-	0.0508

Table 5.2: L_2 error of initial data representations.

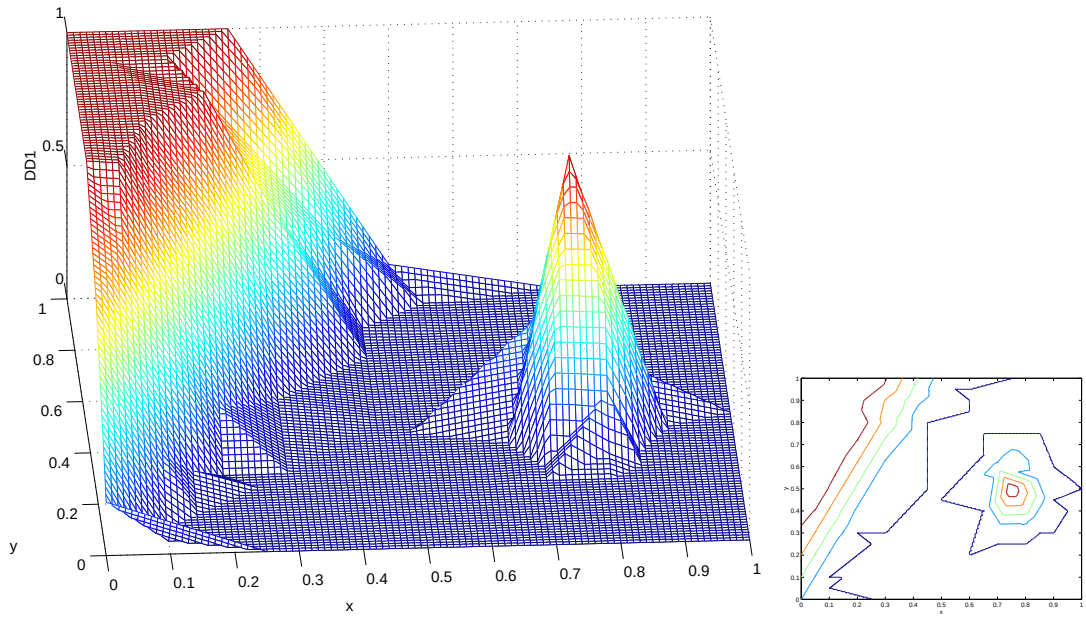

Figure 5.7: The continuous piecewise linear interpolant of the 'DD1' function on the refined mesh of the 33 point data set.

5.2 Edge swapping approach.

Here we give the results of the edge swapping routine developed to implement the theory given in Section 3.1.

5.2.1 81 point data set with the JND-L_2 criterion.

Using the 81 point equidistributed data set, the Jump in Normal Derivative cost function and the L_2 measure, we obtain the following adapted data dependant meshes for the representation of 'SR2' and 'SR3'.

Figure 5.8 is to be compared with Figures 5.4 and 4.5. Most illustrative are the plots of lines of constant value, where we can clearly see the improvement in the data representation. This is achieved by reconnecting the edges so that the elements are aligned across the direction of the slope. It should be noted though that due to the alignment of the elements in the initial mesh, significant reconnection is necessary in order to align elements across the direction of the slope. See Table 5.3 for the numerical mesh quality measures for these results.

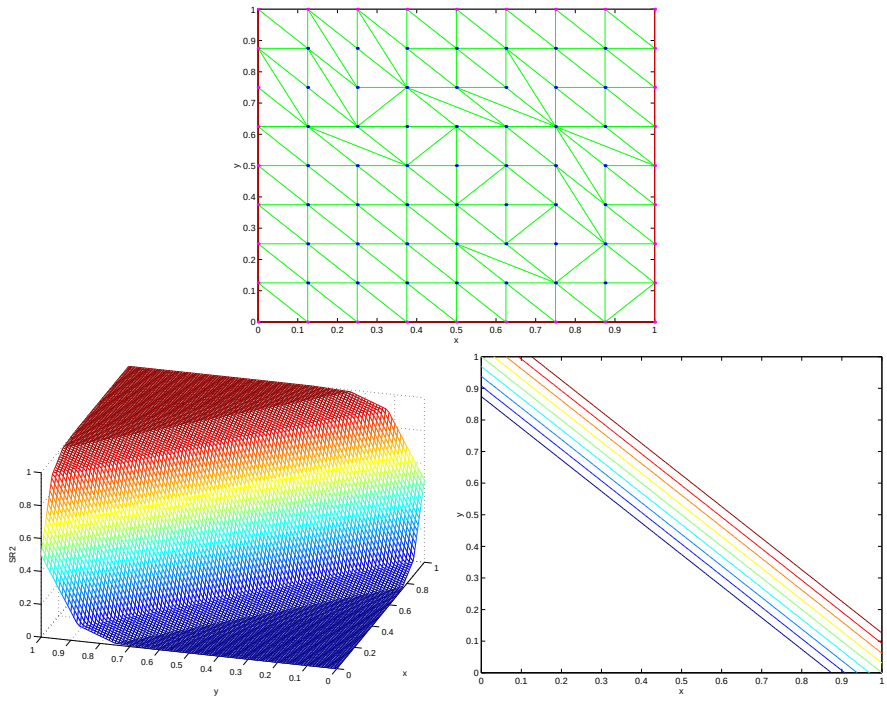

Figure 5.8: Edge reconnection of 'SR2' – JND – L_2 – 81 point data set.

In Figure 5.9 we show result of the same approach applied to the 'SR3' test

Measure	**'SR2'** Mean, Max	**'SR3'** Mean, Max
Maximum angle	98.4, 135.0	141.2, 168.7
Skewness	0.4472, 0.8726	0.9448, 1.264
Aspect ratio	0.7848, 0.8660	0.3450, 0.8660
L_2 error	0.01946	0.03502

Table 5.3: Numerical mesh quality measures for Section 5.2.1.

function. The significance of this choice of function is that the initial mesh is not aligned with the slope as it was for the 'SR2' function. Therefore we expect the adaption routine to significantly modify the nodal connectivity. Although the L_2 error of the final data representation for 'SR3' is noticeably worse than for 'SR2', it is still a great improvement on the initial representation. The degree to which the mesh is 'skewed' in the 'SR3' case should also be noted. These two examples highlight the importance of choosing a suitable initial mesh. This skewing of the elements can attributed to the fact that many of the data points in the 81 point data set are colinear in a direction not dissimilar to the 'downhill' direction of the ramp in the underlying data. Therefore in order for the elements to be aligned across the slope, they have to become highly skewed.

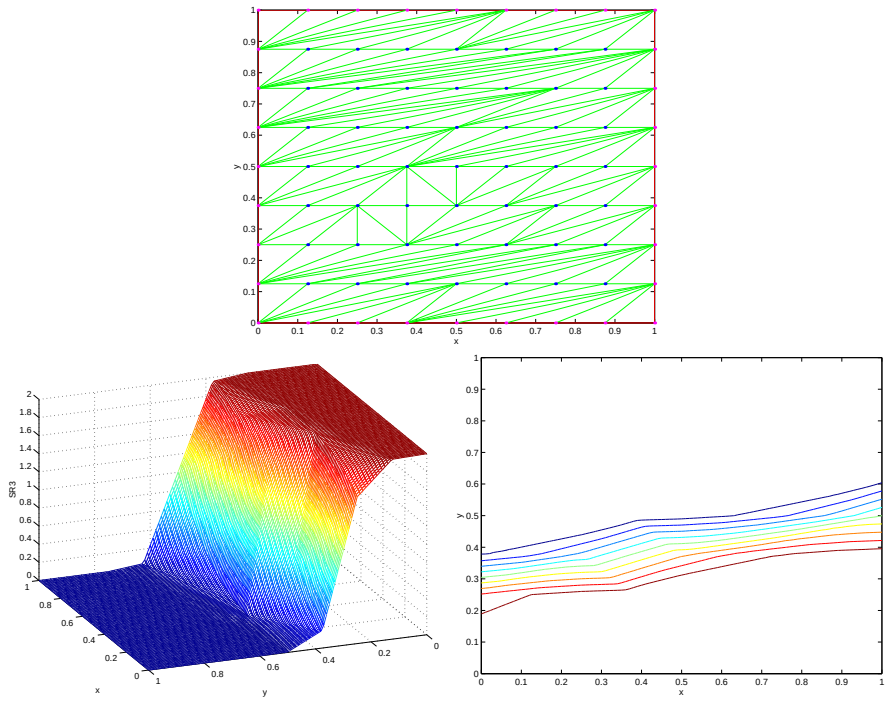

Figure 5.9: Edge reconnection of 'SR3' – JND – L_2 – 81 point data set.

5.2.2 33 point data set with the ABN-L_1 and the ABN-L_2 criteria.

The previous results demonstrated the importance of a suitable data set. In this section we demonstrate the effect of having a non-uniformly distributed data set. We implement the edge swapping routine with the Angle Between Normals cost function and the L_1 and L_2 measures. The routine is applied to the 'SR1' and 'SR2' test functions with the results shown in Figures 5.10 to 5.12 and Table 5.4.

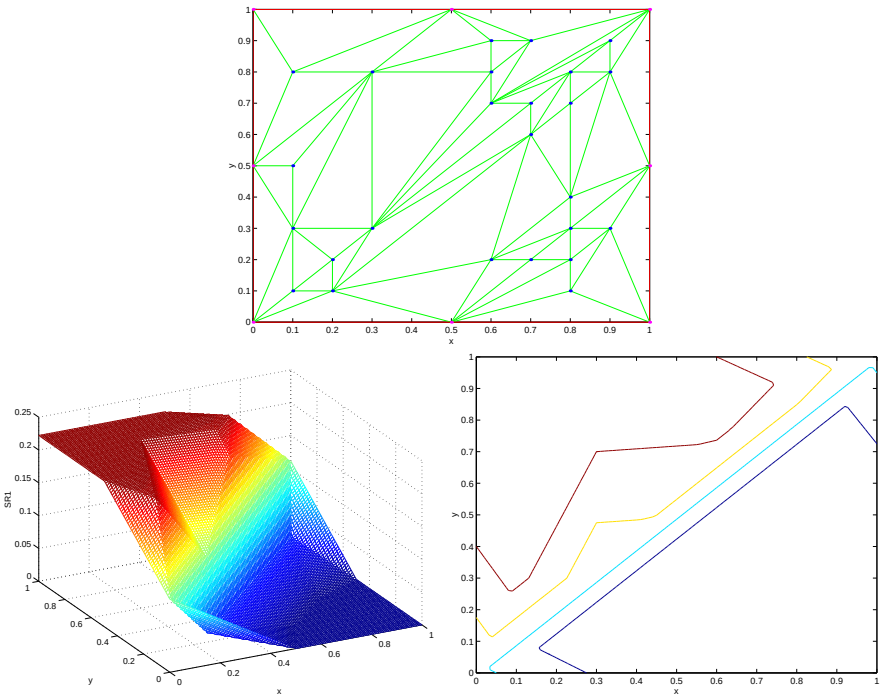

Figure 5.10: Edge reconnection of 'SR1' – ABN – L_2 – 33 point data set.

Measure	**'SR1' – ABN–L_2** Mean, Max	**'SR1' – ABN–L_1** Mean, Max	**'SR2' – ABN–L_1** Mean, Max
Maximum angle	119.4, 168.7	124.1, 174.8	126.0, 168.7
Skewness	0.6970, 1.264	0.7569, 1.335	0.7771, 1.264
Aspect ratio	0.5248, 0.8660	0.4745, 0.8660	0.4539, 0.9897
L_2 error	0.0170	0.0124	0.09458

Table 5.4: Numerical mesh quality measures for Section 5.2.2.

Although these resultant adapted meshes all provide a significant improvement in representing the data over the initial meshes, they all perform differently.

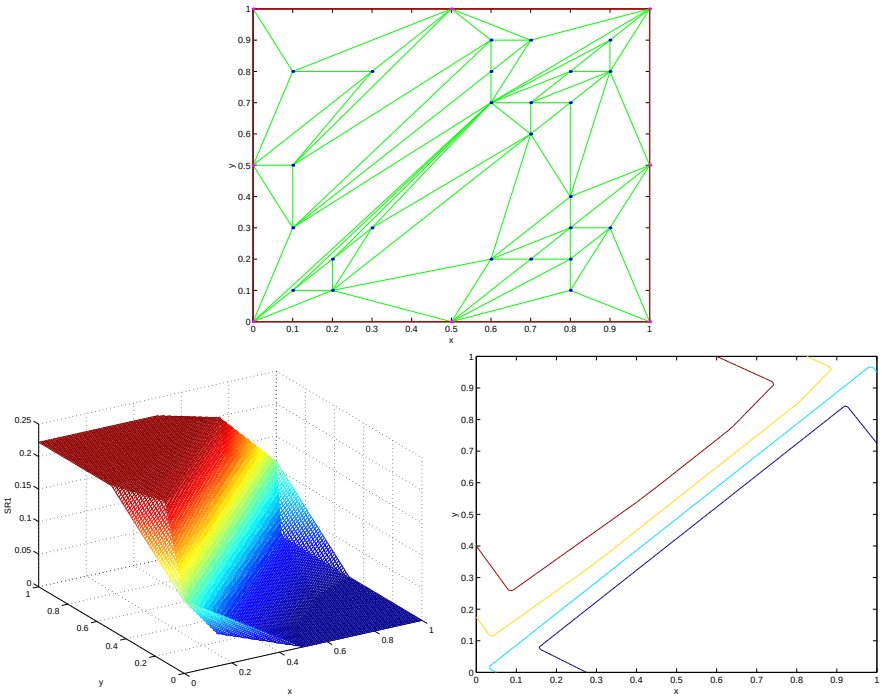

Figure 5.11: Edge reconnection of 'SR1' – ABN – L_1 – 33 point data set.

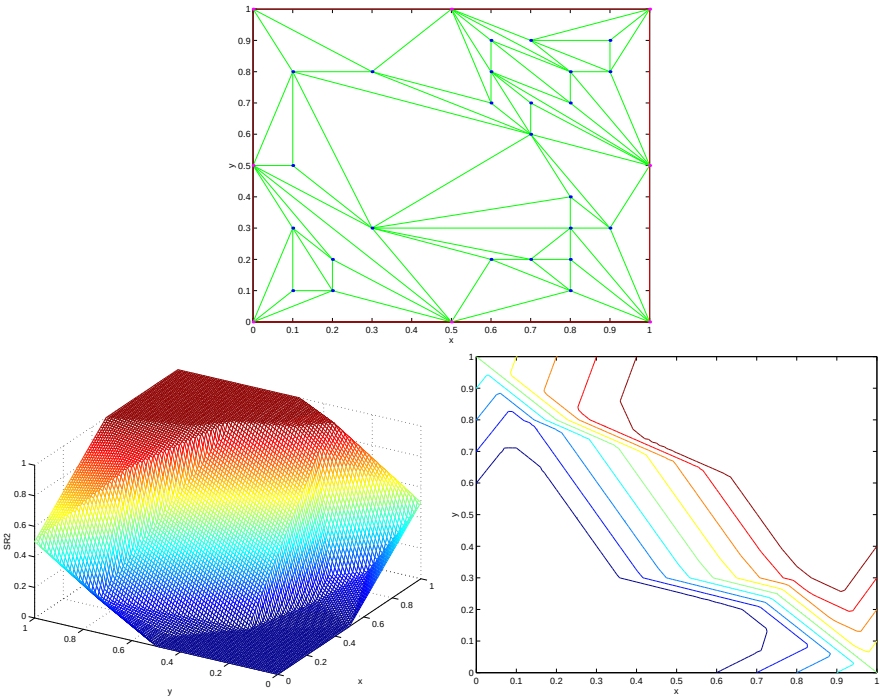

Figure 5.12: Edge reconnection of 'SR2' – ABN – L_1 – 33 point data set.

In particular we draw notice to a comparison between Figures 5.10 and 5.11 and between the corresponding L_2 errors in Table 5.4. Clearly the nature of the measure we use in our data dependent criterion can greatly affect its success.

These three examples also illustrate the importance of having a higher mesh resolution near complexities in the boundary of the domain. None of these final adapted meshes give a good representation of the data near the boundary if the data there is varying.

It should be noted that while there is no clear orientation of the initial mesh, the orientation of the features in the test data still have an affect on the final data representation. This can be seen by comparing Figures 5.11 and 5.12 and the corresponding L_2 errors.

5.2.3 Refined 33 point data set with the ABN-L_2 criterion.

All of the test functions for which we have shown results so far have only had one significant feature, namely the smooth ramp. The 'DD1' test function is made up of two features. The following example shows that the mesh adaption routine can cope with having more than one feature in the initial data. In order to achieve this, the two features must be distinct in the interpolant. That is, the resolution of the mesh needs to be sufficiently fine so that a change in the connectivity near one feature does not affect the representation of the other feature unduly. If the resolution of the mesh is not fine enough for the criterion to be able to consider the features of the data separately, improving the representation of one feature by edge swapping can have a negative effect on the representation of the other. To construct the initial mesh we refine the 33 point Delaunay mesh as described in Section 5.1.

Figure 5.13 shows the result of the adaption routine with the criterion comprised of the Angle Between Normals cost function and the L_2 measure. The numerical mesh quality measures are given in Table 5.5.

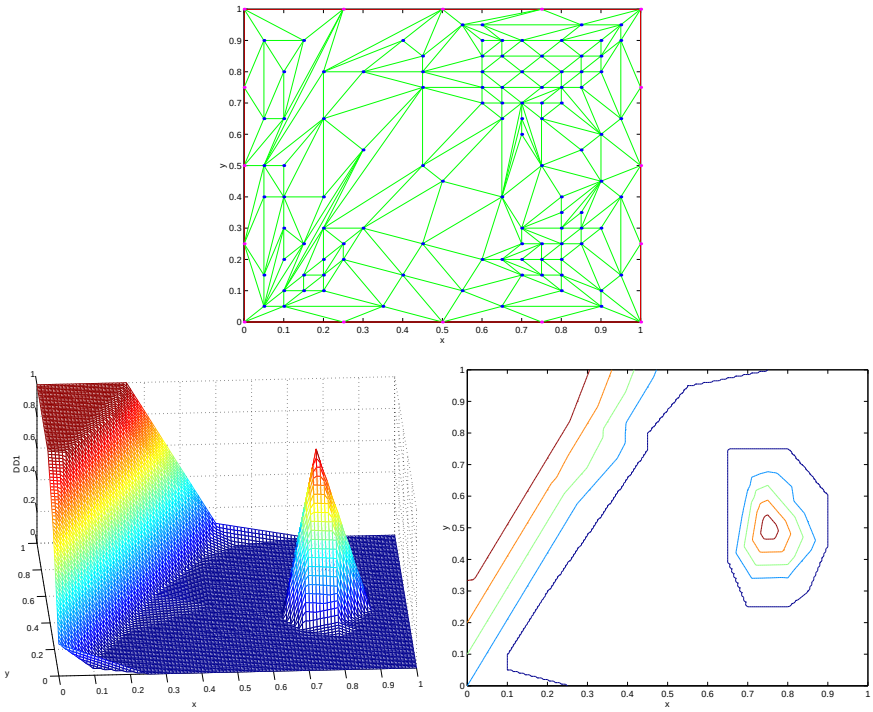

Figure 5.13: Edge reconnection of 'DD1' – ABN – L_2 – Refined 33 point data set.

Measure	**'DD1'** Mean, Max
Maximum angle	112.0, 175.2
Skewness	0.6209, 1.340
Aspect ratio	0.5929, 0.9897
L_2 error	0.03336

Table 5.5: Numerical mesh quality measures for Section 5.2.3.

5.3 Mesh refinement approach.

If the desired accuracy in the interpolant can not be attained by nodal reconnection or nodal movement strategies then it may be necessary to refine the mesh to add resolution. In Section 3.2 we detailed many strategies for mesh refinement. Here we give examples of implementing the *node insertion* method of Section 3.2.5 with both the edge and element based strategies of Section 3.2.2.

5.3.1 By edge.

In Figures 5.14 and 5.15 we show the results of refining the meshes on both the 81 point and 33 point data sets by the edge based method of Section 3.2.2. We choose to refine 20% of the elements. This proportion is chosen as a compromise between adding desired resolution to improve the data representation and keeping the number of nodes to a minimum. The L_2 errors of these representations are given in Table 5.6. Although, by examining the lines of constant value, the initial data representation appears to be better than the final data representation, we note that the L_2 error is reduced. This is owing to the initial mesh not having sufficient resolution to accurately represent the top and bottom of the ramp in 'SR1'.

Clearly mesh refinement will always reduce the error in the data representation, and we could continue to refine the mesh by this method until the desired error threshold is attained. In the case of the 33 point data set, refining 20% of the elements is not sufficient to improve the accuracy at the centre of the domain since the edges with the worst cost are those at the lower left and upper right corners. The data representation is improved in these areas only. The uniformity and symmetry of the refinement of the 81 point data set is a reflection on the correlation between the alignment of the mesh and the prominent feature of the 'SR1' function.

	81 pt	**33 pt**
L_2 error	0.003669	0.01935

Table 5.6: L_2 error in data produced by mesh refinement by edge.

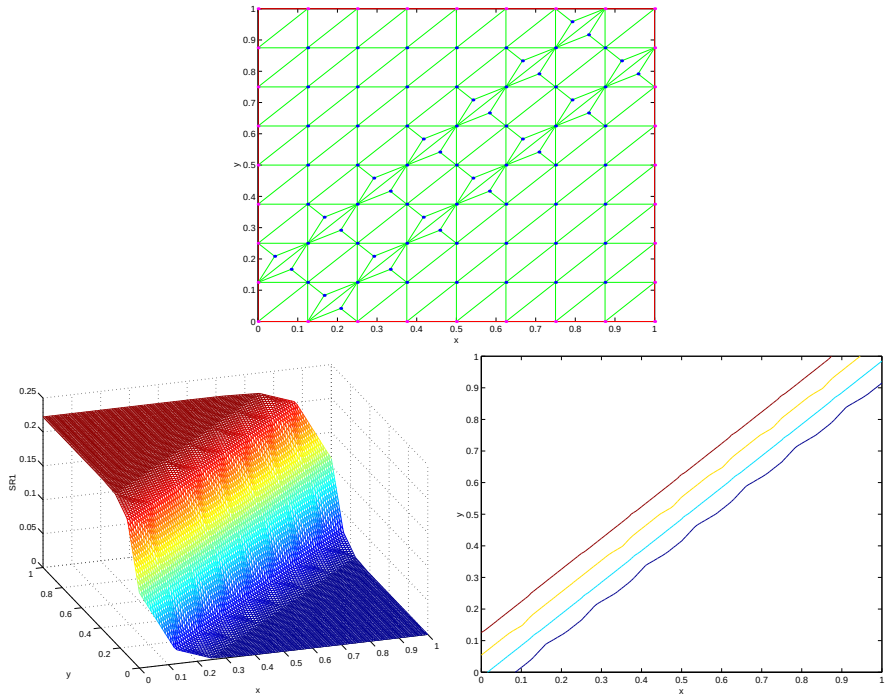

Figure 5.14: Mesh refinement by edge of 'SR1' – ABN – 20% – 81 point data set.

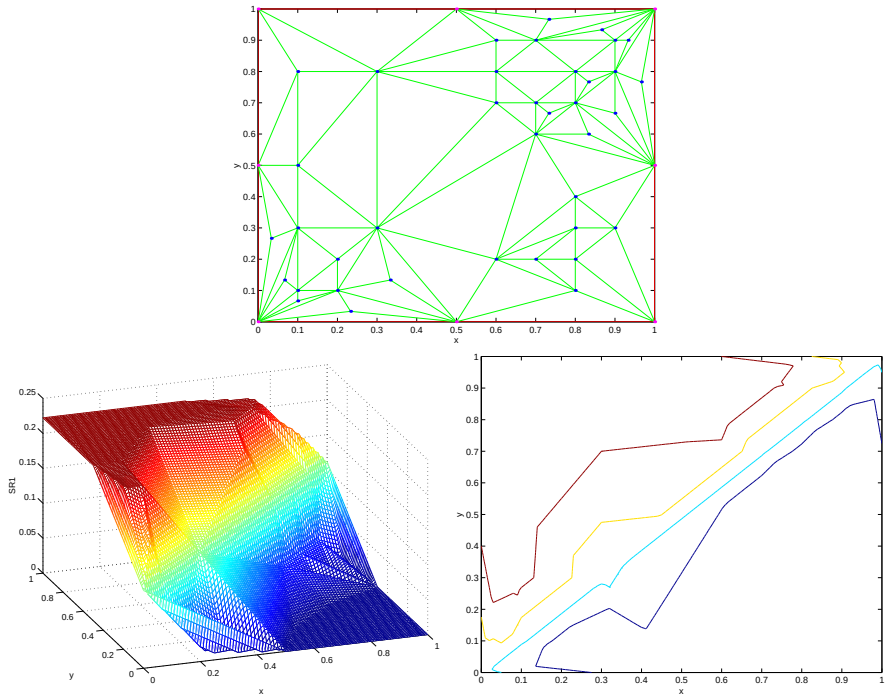

Figure 5.15: Mesh refinement by edge of 'SR1' – ABN – 20% – 33 point data set.

5.3.2 By element.

As a comparison to the results in Section 5.3.1 we implement the element based refinement method of Section 3.2.2. We are considering criteria where the cost functions are defined on the edges of the mesh. In order to obtain a cost function that is defined on the elements, for each element we construct the cost as the arithmetic average of the costs of its edges. Again we choose to refine 20% of the elements.

For the 81 point data set the mesh obtained by this approach is identical to the mesh for the edge based refinement method above. This is due to the uniformity of the data set and the correlation between the alignment of the mesh and the alignment of the ramp in 'SR1'.

For the 31 point data set the resultant mesh, shown in Figure 5.16, is similar to the mesh produced by the edge based refinement routine with only a few differences. With the element based refinement procedure the refinement is focused along slope. In the edge based refinement procedure the refinement is focused closer to the boundaries. The L_2 error for the interpolant by this mesh is $\|e\|_{L_2} = 0.01916$.

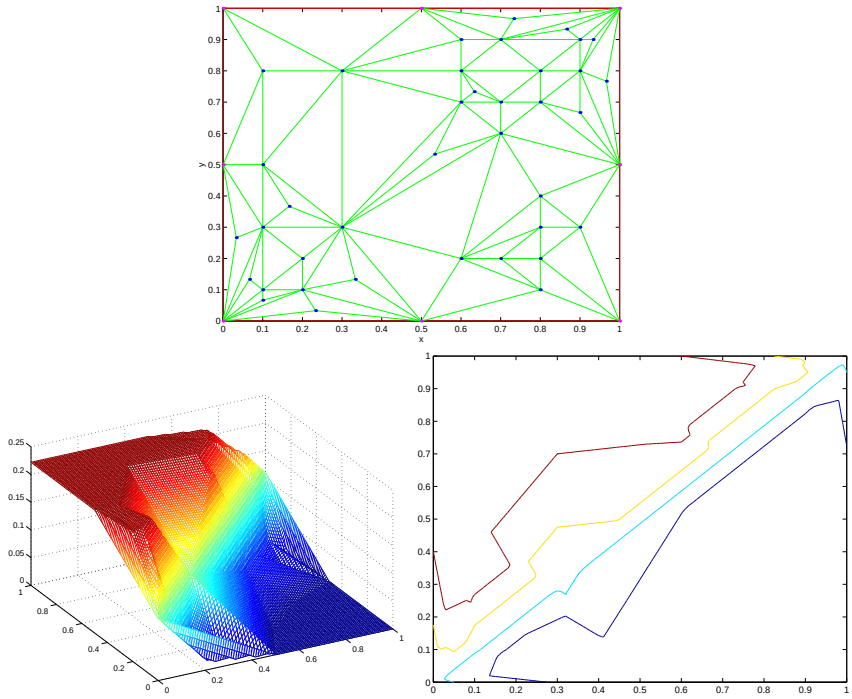

Figure 5.16: Mesh refinement by element of 'SR1' – ABN – 20% – 33 point data set.

5.4 Moving nodes.

Whilst not a data-dependent mesh adaption procedure, we have implemented an iterative Laplacian smoothing routine. The result of smoothing the Delaunay mesh on the 33 point data set is shown in Figure 5.17. Laplacian smoothing of the Delaunay mesh on the 81 point data set does not move the nodes. This is owing to the uniform distribution of the nodes and connectivity.

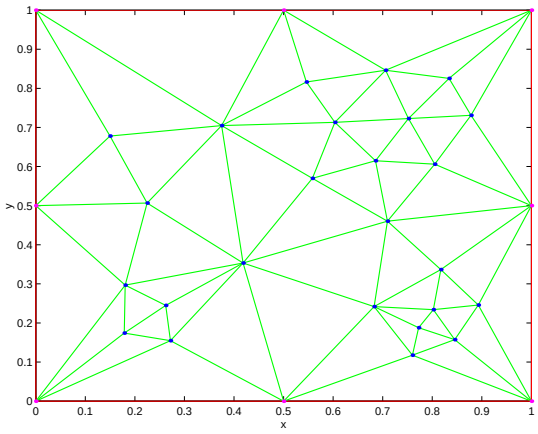

Figure 5.17: Laplacian smoothed mesh.

Chapter 6

Modifications to the mesh adaption strategy.

In this section we describe and implement some modifications that can be made to the mesh adaption strategy.

6.1 Geometrical constraints.

If the mesh we generate is to be used with a numerical differential equation solver, it may be necessary to impose constraints on the geometry of the elements of the mesh. Malcolm [8] suggests imposing the constraint of a lower bound on the possible area of each element or a lower bound on the minimum angle of an element. These constraints are built into Algorithm 3.1.4 by not allowing an edge swap if it would violate one of the constraints.

Out of the results in Section 5 the adapted mesh with the most skewed elements was the representation of 'SR3' on the 81 point mesh adapted by the JND-L_2 criterion. For this example we show the effect of limiting the minimum angle of each element to $8\,^\circ$. The effects of imposing this constraint can be seen in Figure 6.1 and Table 6.1. These should be compared with Figure 5.9 and the 'SR3' column of Table 5.3 respectively.

It is clear that the geometry of the elements is more conducive to use with a method such as the Finite Element Method, but this is at the expense of the accuracy of the representation of the data.

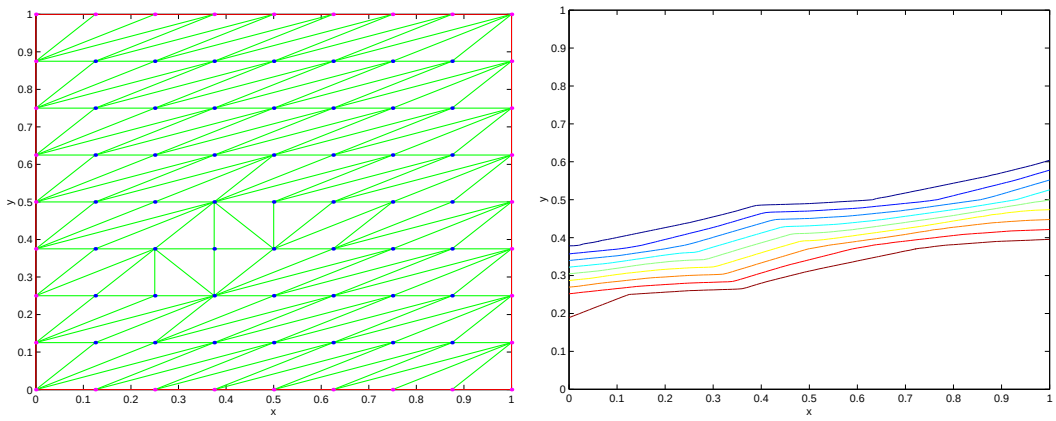

Figure 6.1: Placing geometric limiters on the edge swapping routine.

Measure	**'SR3'** Mean, Max
Maximum angle	137.7, 153.4
Skewness	0.9052, 1.087
Aspect ratio	0.3822, 0.8660
L_2 error	0.05502

Table 6.1: Mesh quality indicators for geometry limiting procedure.

6.2 Alternative averaging in mesh refinement strategy.

In Section 3.2 we use the arithmetic average of three edge or node based cost functions as a representation of the cost on an element. The refinement strategy then refines the elements with the worst cost. This has the effect of attempting to produce a mesh on which all elements have a similar cost. While this is a perfectly valid approach, we could consider an average cost per unit area. Such an average would be $A(\tau) = \frac{1}{\mu(\tau)} \sum_{i=1}^{3} h_i$ where $\mu(\tau)$ is the area of the element τ and $h_i \quad i = 1, \ldots, 3$ are the cost functions associated with the three edges, or nodes of τ depending on the type of cost function we are using. We call this measure the **cost density**. This measure of cost has an intuitively nice property when combined with the mesh refinement strategy since it encourages an even spread of the cost density over the domain. Hence a larger cost is allowed on an element with a larger area.

In order to illustrate this we produce a refined mesh using the same routine

as the 33 point data set example in Section 5.3.2. The results are shown in Figure 6.2 which should be compared with Figure 5.16. The L_2 error of the interpolant of 'SR1' on this mesh is $\|e\|_{L_2} = 0.02150$.

This method could be particularly useful where meshes with a large variation in the element size is desirable. Such a situation could be ocean modelling where very fine mesh sizes are required in coastal regions, but coarser meshes are acceptable away from the coasts.

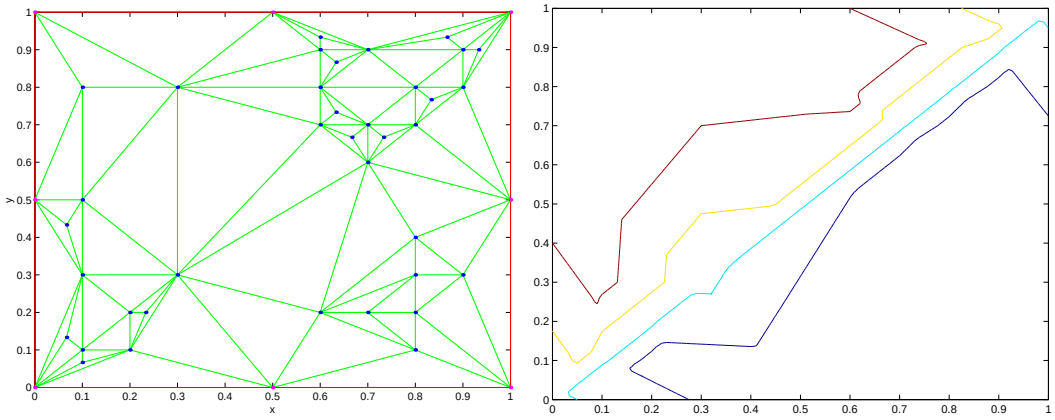

Figure 6.2: Mesh refinement by the cost density.

This measure of cost density could also be used when considering the absolute error of a mesh. On each element, τ, we define the L_2 error density as

$$e_\mu = \frac{1}{\mu(\tau)} \left(\int_\tau |u - u_h|^2 \right)^{\frac{1}{2}}.$$

This approach will not affect the edge swapping mesh adaption routine since swapping the diagonal of a convex quadrilateral will not change the quadrilateral's area. It will however have a significant effect on the mesh refinement procedure since elements with a large error will not necessarily be refined if their area is large.

6.3 Ordering of the edges.

In Section 3.1 we mention that the order in which the edges are searched when using Algorithm 3.1.4 can affect the final locally optimal mesh. To demonstrate this, we reorder the edges or our initial mesh at random and repeat the numerical

experiment in Section 5.2.3. In Figure 6.3 we show the two locally optimal meshes produced, noting that they are largely similar except for a small number of cases. In Table 6.2 we give the numerical mesh quality indicators for the two meshes.

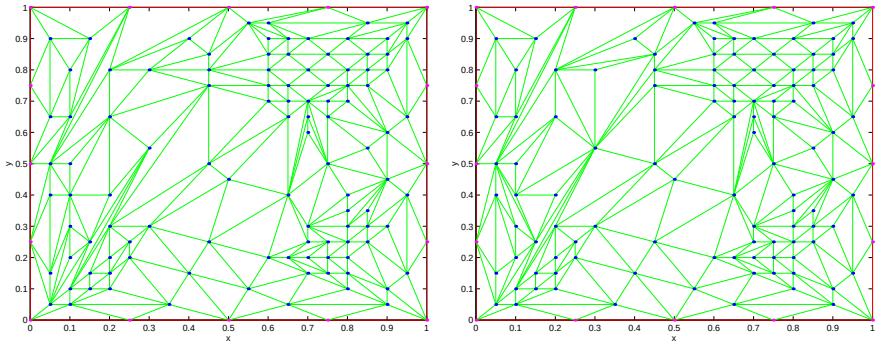

Figure 6.3: Changing the ordering of the edge searching in the LOP.

Measure	**Original** Mean, Max	**Reordered** Mean, Max
Maximum angle	112.0, 175.2	112.6, 175.2
Skewness	0.6209, 1.340	0.6270, 1.340
Aspect ratio	0.5929, 0.9897	0.5893, 0.9897
L_2 error	0.03336	0.03100

Table 6.2: Numerical mesh quality measures for Section 6.3.

It should be noted that the L_2 errors in the interpolants are noticeably different, while the geometrical properties of the meshes are very similar. This demonstrates that a *locally optimal* mesh is not necessarily the *globally optimal* mesh.

Chapter 7

Further work.

In this section we give examples of possible areas for development that would follow on directly from what we have discussed so far.

7.1 Implementation of other mesh refinement methods.

In this report we have implemented the mesh refinement method by *midpoint node insertion* as described in Section 3.2.5. The method of refinement by *edge bisection* is implemented and analysed in [1] and in this report we make use of it to produce the initial mesh on the 81 point data set.

For completeness it would be useful to develop an implementation for the *element bisection* method of mesh refinement. This would enable us to make a comparative study of the use of these methods in data dependent mesh adaption strategies.

7.2 Other methods of assessing the quality of a mesh.

Section 4.5 covers some methods of assessing and quantifying the quality of the mesh however there are many more which may be suitable in other circumstances. Some of these are detailed below.

Uniformity For a given element τ_1, its uniformity is defined as

$$un(\tau_1) = \max\left(\frac{\mu(\tau_1)}{\mu(\tau_2)}, \frac{\mu(\tau_2)}{\mu(\tau_1)}, \frac{\mu(\tau_1)}{\mu(\tau_3)}, \frac{\mu(\tau_3)}{\mu(\tau_1)}, \frac{\mu(\tau_1)}{\mu(\tau_4)}, \frac{\mu(\tau_4)}{\mu(\tau_1)}\right)$$

where τ_2, τ_3 and τ_4 are the neighbouring elements of τ_1 and $\mu(\tau)$ is the area of element τ.

This provides a measure over the mesh of whether the area of the elements is smoothly varying over the mesh. This measure would be of use if we were considering a numerical method in which this property is desirable.

The number of elements having a node in common For each node we record the number of elements that have this node in common. This is of particular interest in classical Finite Element Methods since an increase in this quantity will result in a higher cost in assembling the linear system.

Boundary of the domain We have only been considering domains which have a convex polygonal boundary and we have made the assumption that the convex hull of the data set we impose is identical to the boundary. This means that there is a node at each vertex of the boundary of the domain. In general the physical domain we are considering may not have a polygonal boundary in which case our computational domain would consist of the convex hull of the data set.

Therefore a possible measure for the quality of part of the data set and hence the mesh could be the deviation of this computational domain from the physical domain. The method for improving this measure would be to add nodes on the boundary of the physical domain and data dependent methods could be developed to determine where these should be placed. A candidate for such a method would be to make a cut in the boundary of the domain and approach the problem as a 1 dimensional mesh adaption problem with cyclic boundary conditions[1].

[1]For a possible approach to a 1 dimensional mesh adaption method see Baines [2].

7.3 Data set alignment.

One of the striking features of the results shown in Section 5 is that the alignment of the nodes in the data set with the features of the underlying data can significantly affect the outcome of a mesh adaption routine. Further analysis of this issue would be of benefit. At this stage is it clear that the path formed buy a series of adjoining edges should not be similar to lines of constant value in the underlying data if we wish to avoid a skewed data dependent mesh when adapting by an edge swapping routine.

7.4 Implementation of moving nodes.

In Section 5.4 we give the results of a Laplacian smoothing routine. This routine could be adapted to include data dependency in the spring constants as described by Malcolm [8]. As well as being a data dependent adaption strategy in its own right, it could provide a method of alleviating some of the negative affects of poorly aligned data sets as described above before using an edge swapping approach.

7.5 Combined strategy.

One particular area of interest is to combine the strategies described in this report. At present it is possible to implement each method in turn, but it would be desirable to have one routine that at step decides which strategy to implement.

The primary complexity of such a method is the fact that our existing routines are operations on different components of the mesh. i.e. Edge swapping being an operation on edges and node movement routines operating on patches of elements.

In order to overcome this difficulty, we propose the following algorithm:

Algorithm 7.5.1. Combined strategy.

1. Produce an initial mesh.

2. For each internal node and the patch of elements surrounding it:

(a) Treat the patch of elements as a subdomain and apply a nodal movement strategy to determine a position of the central node that provides the best improvement in the error indicator.

(b) Return to the original patch and apply an edge swapping routine and record the improvement in the error indicator for this strategy.

(c) Select the approach that provides the best improvement in the error indicator.

3. Treat this process iteratively until no edge swaps are made and the change in the position of nodes becomes negligible.

It is not immediately clear that this algorithm will terminate and this is something that would need to be investigated.

Chapter 8

Conclusions.

In this dissertation we have presented some of the commonly used methods of adapting two dimensional meshes consisting of triangular elements. The adaption is aimed at producing meshes that are dependent on the underlying data they are being used to represent. The motivation for this is that limitations in computational resources require such ingenuity in order to produce representations of data that meet accuracy criteria.

For the data dependent adaption strategies discussed, we developed an implementation and provided illustrative examples of their success and limitations. The results conclude that the accuracy of data representations by continuous piecewise linear functions can be significantly improved by applying data dependent adaption procedures to the underlying mesh. It is also apparent that the adaption strategies have a tendency to 'skew' the mesh, a property which is not desirable if it is intended for use with a numerical differential equation solver such as the Finite Element Method. However, we demonstrate that the degree of this 'skewing' can be limited, albeit at the expense of the accuracy of the data representation.

As a development of existing mesh refinement strategies, we suggest an alternative measure of the cost of each element. This measure results in a different focus for the location of mesh refinement, and may be of use in cases where uniformity of the size of elements in the mesh is not a requirement.

Finally a strategy for combining adaption routines is presented. It is expected that this approach would provide a powerful tool for developing data dependent meshes.

Bibliography

[1] J. Aitken, *Adaptive Local Mesh Refinement for Poisson's Equation in 2D*, MMath Final Year Project, University of Bath, 2003.

[2] M. J. Baines, Algorithms for Optimal Discontinuous Piecewise Linear and Constant L_2 Fits to Continuous Functions With Adjustable Nodes in One and Two Dimensions. *Mathematics of Computation*, **Vol. 62** Num. 206, (1994), pp. 645–669.

[3] R. E. Bank, R. K. Smith, Mesh Smoothing Using A Posteriori Error Estimates. *SIAM J. Numerical Analysis*, **Vol. 34** Num. 3, (1997), pp. 979–997.

[4] N. Dyn, D. Levin, S. Rippa, Data Dependent Triangulations for Piecewise Linear Interpolation. *IMA Journal of Numerical Analysis*, 10, (1990), pp. 137–154.

[5] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Computational Differential Equations*, Cambridge University Press, 1996.

[6] J. C. Hardwick, Implementation and evaluation of an efficient parallel Delaunay triangulation algorithm. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures,* June 1997.

[7] W. Huang, D. M. Sloan, A Simple Adaptive Grid Method in Two Dimensions. *SIAM J. Sci. Comput.*, **Vol. 15** Num. 4, (1994), pp. 776–797.

[8] A. J. Malcolm, *Data Dependent Triangular Grid Generation*, PhD Thesis, University of Reading, 1991.

[9] R. Sibson, Locally Equiangular Triangulations. *Comp, J.*, 21, (1978), pp. 243–245.

[10] G. Strang, G. J. Fix, *An analysis of the finite element method*, Prentice-Hall, 1973.

[11] E. Süli and D. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003.

[12] Y. Tourigny, M. J. Baines, Analysis of an Algorithm for Generating Locally Optimal Meshes for L_2 Approximation by Discontinuous Piecewise Polynomials. *Mathematics of Computation*, **Vol. 66** Num. 218, (1997), pp. 623–650.

Appendix A

33 point data set.

Here we include the 33 point data set as used by Malcolm [8]. The table below gives the x and y coordinates for each node.

Node number	**x**	**y**	**Node number**	**x**	**y**
1	1.0	1.0	18	0.9	0.3
2	1.0	0.0	19	0.8	0.3
3	0.1	0.3	20	1.0	0.5
4	0.6	0.2	21	0.5	0.0
5	0.1	0.8	22	0.8	0.7
6	0.5	1.0	23	0.2	0.1
7	0.6	0.8	24	0.1	0.5
8	0.0	1.0	25	0.9	0.8
9	0.6	0.7	26	0.6	0.9
10	0.7	0.7	27	0.3	0.3
11	0.8	0.1	28	0.7	0.9
12	0.9	0.9	29	0.8	0.4
13	0.2	0.2	30	0.0	0.5
14	0.8	0.2	31	0.1	0.1
15	0.0	0.0	32	0.8	0.8
16	0.7	0.2	33	0.7	0.6
17	0.3	0.8			