# Data-Dependent Criteria for Nodal Placement with Fixed Connectivity in Triangular Grids

Andrew Malcolm

Numerical Analysis Report 16/90

**Abstract**

In classical triangulations usually either a regular set of vertices is used or a set of vertices is constructed using geometrical considerations of the region to be triangulated. With the advent of Moving Finite Element Methods, whose accuracy are highly dependent on the representation of the initial data on the initial grid, we need better nodal positioning based on the underlying data. In this report two possible procedures for positioning nodes for better representations of data on triangulations with fixed connectivity are discussed and their effects compared.

# 1 Introduction

In classical triangulations usually either a regular set of vertices is used or a set of vertices is constructed using geometrical considerations of the region to be triangulated. These take no account of the underlying data or its behaviour.

Our aim is to produce a set of vertices which, with a prescribed connection, accurately represent underlying data, (e.g. initial data), when interpolated on it. This can then be used in connection with the strategy in Malcolm [9], whereby the connections in an unstructured grid are changed to produce better representations of data.

These strategies could be applied, for example, to producing initial grids for Moving Finite Elements [10], which move the grid according to the solution of the PDE and, hence, its accuracy is highly influenced by the representation of the initial data on the initial grid.

In this report we outline two procedures that were used to produce nodal redistribution. The first procedure is a movement criteria based on treating edges in a triangulation as springs, with spring constants dependent on properties of the underlying function to be interpolated on the grid, e.g. Catherall [2], Lohner et al. [8]. The second procedure used is a movement criteria based on approximately equalising the error of interpolation on each triangle in the triangulation, based on error estimates derived from work by Nadler [11]. This is akin to work done by Sewell [13].

In Section 2 we outline the nodal movement criteria in greater detail. In Section 3 the implementation of the redistribution procedures are outlined, while in Section 4 the testing procedure, the test functions and the data sets used

are stated. In Section 5 the numerical and graphical results are presented, and, finally, in Section 6 a summary of the results and conclusions are given.

# 2 Nodal Movement Criteria

There are many different ways that we could choose to redistribute nodes towards an "ideal" grid with fixed connectivity given an arbritrary initial grid. Some methods involve smoothing, Cavendish [3], or when solving a differential equation using the equation variables to smooth the grid, Palmeiro et al. [12].

Here however we concentrate on two general forms of criteria. Firstly we consider a spring analogy in which we imagine the triangle edges to be comprised of springs whose stiffness is given by some monitor of the underlying function, Eisemann et al. [6], Catherall [2]. Secondly we strive to approximately equalise some measure of interpolation error on all triangles.

Both methods involve a local iteration technique around the elements, in which local patches of triangles are put in equilibrium according to the corresponding criteria. This process is then repeated over all such patches until global equilibrium is achieved. This can be thought of as being similar to a Gauss-Siedel iterative process.

We now proceed to describe the techniques in more detail :-

## 2.1 Spring Analogy

This is a global method, achieved by local iteration, of finding the new position of nodes. Each node is surrounded by a patch of triangles.

Thus the patch P around node $v_i$ has $p_i$ surrounding nodes, $v_{i,j}$, $j = 1 \ldots p_i$,

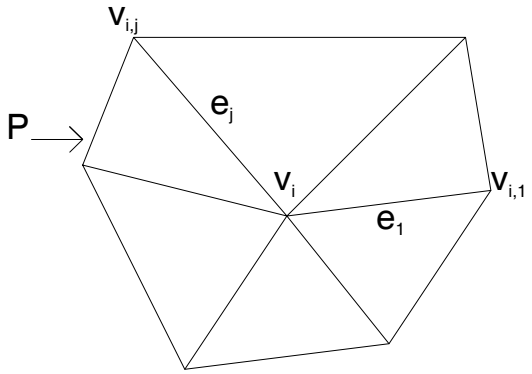


Figure 1: Patch of Triangles

with node $v_i$ connected to node $v_{i,j}$ by edge $e_j$.

For this criterion edge $e_j$ has associated with it a spring constant $k_j$ which is dependent on certain properties of the underlying function. We then attempt to find an equilibrium position for node $v_i$.

Originally the "springs" were given a specified unextended length and the equation for the tension, $T$, was

$$T = \frac{k_j \times \text{extension}}{\text{original length}} \tag{2.1}$$

where

$$\text{extension} = \text{separation of the two nodes - original length}$$

This was an attempt to control the nodal separation. However, this produced poor results so it was decided, following Thacker et al. [14] and Palmeiro et al. [12], to assume that all springs had negligible length when unstretched. This meant that as we are trying to reach an equilibrium for the tensions around node $v_i$, the original length in Equation (2.1) can be ignored as it can just be factored out. Hence the tension, $T_j$, for edge $e_j$ is

$$T_j = k_j \sqrt{(x_i - x_{i,j})^2 + (y_i - y_{i,j})^2}$$

If we separate $T_j$ into $x$ and $y$ components we get

$$T_{j_x} = k_j(x_i - x_{i,j})$$

$$T_{j_y} = k_j(y_i - y_{i,j})$$

resolving the tensions over the whole patch. To find the new position $(\tilde{x_i}, \tilde{y_i})$ of node $v_i$, we get

$$\sum_{j=1}^{p_i} T_{j_x} = 0.$$

Hence,

$$\tilde{x_i} = \frac{\sum_{j=1}^{p_i} k_j x_{i,j}}{\sum_{j=1}^{p_i} k_j}$$

and similarly for $\tilde{y_i}$.

It is now necessary to decide on which property or properties of the underlying function the spring constant,$k_j$, is to be based. The original choice was

$$k_j = (u_{ss})^{2/5}$$

where $u_{ss}$ is the directional second derivative along the element edge. This is the two-dimensional analogy of the equidistribution theory of Carey and Dinh, [1]. $u_{ss}$ is calculated at the mid-point of edge $e_j$. However this produced excessive clumping of nodes in regions of high curvature and poor representations of smooth regions. The defects were rectified by choosing

$$k_j = 1 + \alpha \frac{(u_{ss})^{2/5}}{[(u_{ss})^{2/5}]\max}$$

The addition to $k_j$ of unity gives some control of nodal separation and helps place the nodes in smooth regions. $[(u_{ss})^{2/5}]_{max}$ is the maximum value of $(u_{ss})^{2/5}$ over all the edges and $\alpha$ was a parameter chosen to control the weighting given to the equidistribution measure, usually with a value of 10 or 20.

An alternative choice for the spring constant is

$$k_j = 1 + \beta \frac{|u_s|}{|u_s|_{\max}}$$

This is based on the one-dimensional expression for arc-length and would tend to produce movement of nodes to positions of high gradient, the unit constant keeping a control on nodal separation. $k_j$ was calculated for the edge, $e_j$, joining $(x_a,\ y_a)$ to $(x_b,\ y_b)$ by

$$u_s = \frac{F(x_a, y_a) - F(x_b, y_b)}{\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}}$$

and $\beta$ was chosen to give a weighting to the gradient, (usually 1, 2 or 5). The main motivation for this was to produce better representation in regions where $u_{ss} = 0$ but the function is not smooth, (see Function F7, Fig 6(b), for an example).

Eventually a combination of the above was used so

$$k_j = 1 + \alpha \frac{(u_{ss})^{2/5}}{[(u_{ss})^{2/5}]_{\max}} + \beta \frac{|u_s|}{|u_s|_{\max}} \tag{2.2}$$

## 2.2 Error Equalisation

The second criteria is a global interpolation error equalisation criteria treated iteratively in a local manner. Based on modifications to linear interpolants of expressions produced by Nadler [11], for the error of least squares fits on triangles, we can, by substituting numerical values into these expressions, attempt to equalise the error on all triangles. These values are then used on patches of triangles to produce a formula for relocating the central node in the patch. The formulation is as follows :-

$$(\text{error})^2 \text{on triangle} = W_i$$

$$W_i \approx C_i(detH) \times [\text{area}(T_i)]^3 \times |detH|$$

where $H = \begin{pmatrix} u_{xx} & u_{xy} \\ u_{yx} & u_{yy} \end{pmatrix}$ is the Hessian at the centroid of $T_i$, and $C_i$ is a constant depending on the sign of $detH$.

Since for most uses we are interested in the $L_2$-error, we deal with (error)$^2$ rather than error. To get the same (error)$^2$, say $W_{opt}$, on each triangle in the patch we get

$$\sum_{i=1}^{P} W_i = P \times W_{opt}$$

Then for each triangle in the patch

$$\frac{W_i}{W_{opt}} = \frac{C_i \times [\text{area}(T_i)]^3 \times |detH|}{C_i \times [\text{area}_{opt}(T_i)]^3 \times |detH|}$$

where $\text{area}_{opt}(T_i)$ is the area which triangle, $T_i$, should be to give an (error)$^2$ of $W_{opt}$, i.e.

$$\text{area}_{opt}(T_i) = \left[\frac{w_{opt}}{w_i}\right]^{1/3} \times \text{area}(T_i)$$

A further condition which can be imposed is that the area of a patch should be conserved, i.e. the central vertex remains within the patch.

$$\sum_{i=1}^{P} \text{area}(T_i) = \sum_{i=1}^{P} \overline{\text{area}_{opt}}(T_i)$$

where

$$\overline{\text{area}_{opt}}(T_i) = \lambda \ \ \text{area}_{opt}(T_i),$$

and

$$\lambda = \frac{\sum_{i=1}^{P} \text{area}(T_i)}{\sum_{i=1}^{P} \text{area}_{opt}(T_i)}.$$

We then define

$$\epsilon(i) = \frac{\overline{\text{area}_{opt}}(T_i)}{\text{area}(T_i)} = \lambda \times \left[\frac{w_{opt}}{w_i}\right]^{1/3} \tag{2.3}$$

Thus $\epsilon(i)$ can be thought of as an expansion factor :-

$\epsilon(i) > 1$, triangle is enlarged

$\epsilon(i) < 1$, triangle is shrunk.

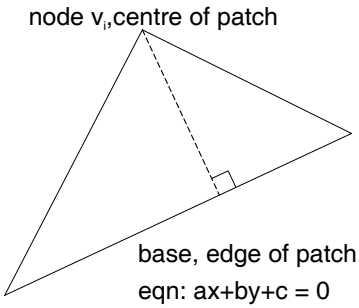


Figure 2: Triangle

Having found $\epsilon(i)$ as defined in Equation (2.3), we can find the equation of the line representing the "height" on the new triangle to which the point should move to, such that triangle $T_i$ has area $\overline{\text{area}}_{opt}(T_i)$, using the following method :-

$$\text{area of triangle} = 0.5 \times \text{base length} \times \text{perpendicular height} \tag{2.4}$$

The Perpendicular distance, D, of point (u, v) from line $ax + by + c = 0$, as shown in Figure 2, is

$$D = \frac{au + bv + c}{\sqrt{a^2 + b^2}} \tag{2.5}$$

Using Equations (2.3), (2.4) and (2.5) we get

$$a\tilde{x_i} + b\tilde{y_i} + c = \epsilon(i)(ax_i + by_i + c)$$

Since the base length remains constant, with $(\tilde{x_i}, \tilde{y_i})$ the new position of $(x_i, y_i)$. Thus the new point moves on a line parallel to the base of the triangle.

Having found the equations from all the triangles in the patch, we have P equations for the position of the new node. We can now use some or all of these equations to find the new nodal position.

Amongst the procedures used to find the nodal position were :-

1) least squares fit :- since P lines will almost certainly not meet at a point.

2) use equations 1 and 2, i.e. the equations produced by the first triangle and its neighbour to give the new nodal position: this has the problem of not guaranteeing that the node will stay inside the patch, especially if $\epsilon(1)$ or $\epsilon(2)$, as defined by (2.3), are greater than 1.

3) if possible, use equations j and j+1 where $\epsilon(j)$ and $\epsilon(j+1)$ are both less than or equal to 1. If this is not possible choose j so that

$$\epsilon(j) + \epsilon(j+1) < \epsilon(k) + \epsilon(k+1). \qquad k = 1 \ldots p. \qquad j \neq k.$$

This improves the chances of a node remaining inside a patch.

In all cases, however, it is still necessary to check for degenerate triangles, (see Figure 3).

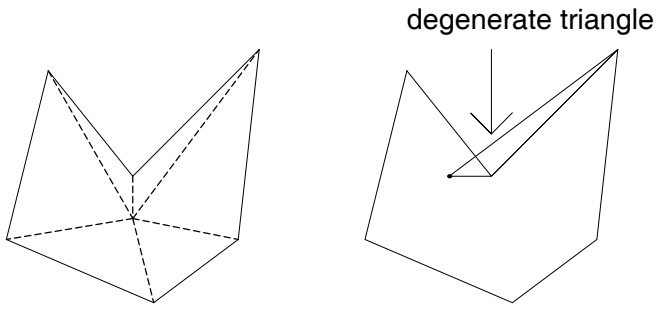


Figure 3: Production of Degenerate Triangle

If a degenerate triangle occurs then only a proportion of the move is made,

and another check is carried out. If it still degenerates then we do not move the node.

# 3 Implementation

The procedures that were used in the testing of the criteria are outlined here. Firstly an original triangulation was set up, this was usually Delaunay [4], but various grids produced by data dependent reconnection criteria, [9], were also used: this was to see if nodal movement depended heavily on the original triangulation. Originally only the interior nodes were moved, with each node being moved inside its patch to a new position, and then the next node in the list being moved. This is analogous to a Gauss-Siedel iteration procedure as the new position is used in all subsequent moves and all the spring constants are recalculated for each node. The method is analogous to a S.O.R method if we only take a proportion, w, of the calculated move. It was decided not to use a complete global method, akin to the Jacobi iteration, where all spring constants were found at the start of a cycle and an $n_i$ by $n_i$ matrix system is solved, where $n_i$ is the number of interior nodes.

Following each sweep through the list of nodes, a check is made to see if the nodes have converged to a steady solution. The check used depended on the criteria in use. For the spring analogy the check was to calculate the maximum distance moved in each sweep and if this was less than some tolerance then the method had converged. For the error equalisation criteria a note was made of the minimum value of $\epsilon(i)$, the expansion factor for a triangle, over all patches in a sweep. When this minimum value is greater than a given tolerance less than 1,

usually about 0.8, then the procedure was stopped as it could be assumed that all triangles were close to the "ideal" size.

It was then decided, in the spring analogy case, to move the boundary nodes, but to constrain them to move along the boundary. Originally this was done totally before the interior nodes were moved, by using the same spring analogy as for the interior nodes, but using only the connections on the boundary. This produced some problems as it was possible that in moving the boundary nodes to their optimal points and keeping the connections the same, degenerate triangles could be produced. This could be tackled in two ways :-

1. include all nodes in a sweep but constrain the vertices of the region to stay fixed and the boundary nodes to remain on the boundary, and then continue until all nodes have moved to their optimal positions.
2. when the boundary nodes only have been moved, re-Delaunay the resulting nodal positions to produce a new delaunay triangulation and work on the new triangulation for the interior points only.

The advantage of (1) is the lack of extra work to find a new triangulation. The disadvantages are that the nodes do not necessarily move to the best position but are constrained by the connectivity, and that if the boundary nodes are constrained to the boundary but all connections must be used as springs, then ghost points must be used (see Figure 4).

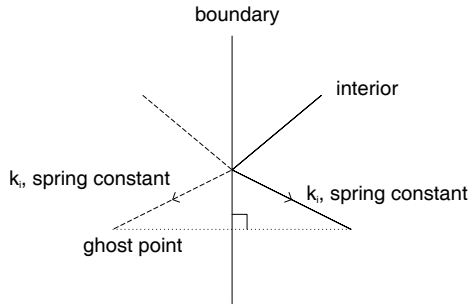


Figure 4: Ghost Points

# 4 Testing Procedures

In this section the underlying functions and the sets of data points which were used in the calculations are detailed and the procedure for numerical error comparison is outlined.

## 4.1 Data Sets

Two different data sets were used in the calculations. The first data set was of 33 data points and was that used by Franke [7], Dyn et al. [5], Malcolm [9]. The second data set was of 81 points, set on a regular cartesian grid.

## 4.2 Test Functions

The test functions are again those used by Dyn et al. [5], and the numbering is consistent with theirs. The test functions are mainly smooth curved surfaces, although F7 has discontinuous first derivatives. Detailed expressions are given for the most used functions. All the functions are defined on the unit square.

$$F1 \quad = \quad \frac{3}{4}e^{-\frac{1}{4}((9x-2)^2+(9y-2)^2)} + \frac{3}{4}e^{-\frac{1}{49}(9x+1)^2-\frac{1}{10}(9y+1)}$$

$$+\frac{1}{2}e^{-\frac{1}{4}((9x-7)^2+(9y-3)^2)} - \frac{1}{5}e^{-((9x-4)^2+(9y-7)^2)}$$

$$F2 = \frac{(\tanh(9y-9x)+1)}{9}$$

$$F7 == \begin{cases} 1 & y-\psi \geq \frac{1}{2} \\ 2(y-\psi) & 0 \leq (y-\psi) < \frac{1}{2} \\ \frac{1+\cos(4\pi r)}{2} & r \leq \frac{1}{4} \\ 0 & \text{otherwise} \end{cases}$$

where $\psi = 2.1x - 0.1$

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

This function represents a "mountain" on a plane and a ramp leading to another plane. It is a function with discontinuous first derivatives.

$$F8 = 1 + \tanh(-3g(x,y)) \quad \text{where} \quad g(x,y) = (.595576(y+3)^2 - x - 1)$$

$$F9 = x^6(1-\frac{y}{2})^6 + y^6(1-\frac{x}{2})^6 + (1-\frac{x}{2})^6(1-\frac{y}{2})^6 + (10xy)^3(1-x)^3(1-y)^3$$

## 4.3 Error Calculations

The error calculation was changed from that used in Malcolm [9]. Rather than pointwise error on a 33 by 33 regular grid on the unit square, the method used was to calculate the $L_2$-error of the interpolant to the actual function. This gives a clearer idea of the overall quality of the interpolant and fits in well with Finite Element calculations.

The $L_2$-error was calculated as follows:-

$$\text{with interpolant} f_T \text{ to function} f$$

$$L_2\text{-error} = \sqrt{\sum_{i=1}^{T_n} \left( \int_{T_i} [f - f_{T_i}]^2 dxdy \right)}$$

where $T_n$ is the number of triangles in the triangulation T and

$$T = \bigcup_{i=1}^{T_n} T_i \qquad\qquad T_i \bigcap T_j = \emptyset \qquad\qquad i \neq j,$$

and $F_{T_i}$ is the interpolant $f_T$ on triangle $T_i$.

The integral was calculated using 13-point Gaussian quadrature on the triangle.

# 5 Results

In this section numerical and graphical results are presented for most of the test functions, all the data sets and both the criteria described previously in the report. The tables of numerical results are set out for data sets of 33 and 81 points and for most of the functions. A key to the letters down the left hand side of the table follows here :-

A) Delaunay triangulation of the original data set, (regular pattern for 81 data points),i.e. the nodes were ordered from bottom left to top right, along the rows.

B) Delaunay triangulation of the reordered set of 81 data points. The ordering of the nodes in (A) was changed completely so that the ordering was randomised.

C) This is the result of using the reconnection procedure ABN-2 as described in [9].

D) This is the result of using the reconnection procedure PF-1 as described in [9].

The preceeding categories (C) and (D) are to give a comparison to results produced in [9].

E) This is the result produced by using the spring analogy with, in Equation (2.2), $\alpha = 2, \beta = 1$, (i.e mainly curvature). The boundary nodes move.

F) This is the result produced by using the spring analogy with, in Equation (2.2), $\alpha = 1, \beta = 5$, (i.e mainly gradient). The boundary nodes move.

G) This is the result produced by using the spring analogy with, in Equation (2.2), $\alpha = 10, \beta = 2\text{r}$. The boundary nodes move.

H) This is the result produced by using the spring analogy with, in Equation (2.2), $\alpha = 10, \beta = 2$. The boundary nodes do not move.

I) This is the result produced by using the spring analogy with, in Equation (2.2), $\alpha = 10, \beta = 2$. The boundary nodes move. The reordered data set is used.

J) Reconnection procedure ABN-2 is employed on the grid produced by methods (E)-(I) which had the smallest $L_2$-error.

K) Method (G) is employed on the grid produced by method (C)

L) The error equalisation technique is used on the Delaunay grid.

| $L_2$errors for 81 points | | | | | |
|---|---|---|---|---|---|
| **Method** | $F1$ | $F2$ | $F7$ | $F8$ | $F9$ |
| | $10^{-2}$ | $10^{-3}$ | $10^{-2}$ | $10^{-2}$ | $10^{-2}$ |
| A | 2.08434 | 4.32505 | 4.44908 | 5.96940 | 1.09879 |
| B | 2.03668 | 7.47902 | 4.78889 | 7.03185 | 1.22257 |
| C | 1.91816 | 4.32775 | 4.49762 | 3.87988 | 0.68699 |
| D | 2.04762 | 4.32775 | 3.91496 | 5.00737 | 0.71525 |
| E | 1.75760 | 0.97138 | 12.1595 | 1.58498 | 0.81894 |
| F | 3.55918 | 1.52432 | 4.57950 | 2.54537 | 1.07889 |
| G | 1.87650 | 1.11700 | 8.46255 | 1.66004 | 0.84348 |
| H | 2.04940 | 3.31976 | 8.15408 | 2.83537 | 0.91046 |
| I | 2.01808 | 1.66920 | 9.39773 | 1.65603 | 0.93596 |
| J | 1.59973 | 0.64647 | 3.52716 | 1.00133 | 0.63720 |
| K | 2.38256 | 1.33465 | 10.8113 | 2.64847 | 0.95160 |
| L | 7.65151 | 4.32505 | 9.54344 | 32.9481 | 1.67338 |

Table 1: $L_2$-errors for 81 points

The graphical results show 3-D isoparametric surface representations (isoplots), contours produced by the interpolants on the grids and the grids themselves. The two different Delaunay triangulations produced by the 81 data point set are shown in Figure 5. The actual pictures of functions F2 and F7 are shown in Figures 6(a) and 6(b) respectively.

| $L_2$ errors for 33 points | | | | |
|---|---|---|---|---|
| **Method** | $F1$ | $F2$ | $F7$ | $F9$ |
| | $10^{-2}$ | $10^{-2}$ | $10^{-1}$ | $10^{-2}$ |
| A | 6.95143 | 2.37108 | 1.20728 | 7.37058 |
| C | 5.81319 | 1.41924 | 1.01215 | 6.92590 |
| D | 5.19476 | 2.00543 | 1.03717 | 6.93231 |
| E | 5.96320 | 1.41626 | 2.17230 | 6.36400 |
| F | 6.08437 | 0.84774 | 1.19977 | 5.60495 |
| G | 5.04153 | 1.07470 | 1.83076 | 6.23826 |
| H | 4.32820 | 1.69124 | 1.36581 | 6.87378 |
| J | 4.63891 | 0.63647 | 1.02559 | 5.20079 |
| K | 6.73206 | 0.69915 | 2.43962 | 6.14087 |
| L | 8.94941 | 2.37108 | 1.27543 | 5.56620 |

Table 2: $L_2$-errors for 33 points

As can be seen from the numerical results in Table 1, which further illuminate the results from Table 2, there can be considerable improvements in data representation by the interpolant. We can see that a change in the ordering of the nodes can greatly change the error of a representation. In most cases nodal movement compares favourably with nodal reconnection. It can also be seen that the movement of the boundary nodes has a positive effect on the error, i.e. compare results G and H on function F2. In all the cases documented here either large $\alpha$ or large $\beta$ in equation (2.2) provides a better representation, while a moderate

weighting of both $\alpha$ and $\beta$ provides a better but intermediate result.

In all cases the best result is produced by moving the nodes and then reconnecting them using a data dependent criterion. This, it should be noted, produces vastly different results from reconnecting and then moving nodes. Also to be seen is the failure of the error equalisation criteria which in all cases provides poorer representations than Delaunay.

In the graphical output certain things can be seen. In Figure 9 the movement of nodes towards the line $y = x$ can be seen clearly, while in Figure 10 long, thin triangles parallel to the contours are created. The anomaly of the two "well shaped" triangles in the centre of the grid is probably due to the order in which the triangle edges were reconnected. Figures 10 and 11 clearly show the difference between "moving and reconnecting" and "reconnecting and moving".

Figures 12-17 all deal with function F7 which should have the widest range of results, due to the smooth portions of the function. Figure 13 shows that nodal reconnection improves the representation, by the interpolant, of the ramp while Figure 14 shows that using a spring analogy weighted towards $(u_{ss})^{2/5}$ clusters the nodes about the "mountain" due to its property of being the only feature with a 2nd derivative, while poorly representing the ramp. Figure 15 shows that if "arc length" is weighted in the spring analogy, then the ramp is better represented and the nodes do not cluster as severely near the "mountain". The use of data dependent reconnection on the grid in Figure 15, see Figure 16, produces excellent representation of the ramp and a much better representation of the "mountain". Moving the nodes, as connected in Figure 13, produces, in Figure 17, an awful representation of the whole function, by the interpolant.

Figures 18-22, which show function F8, and Figures 24-28, which show function F9, reinforce the comments made above and the numerical results in Table 1. However, Figure 23 gives an example of how bad the error equalisation criteria can be.

# 6 Conclusions

There is an increasing use of numerical techniques which require an initial triangulation able to well represent the initial data when this is interpolated on it. In a previous report it was shown that reconnecting nodes improved representation of the data. In this report it is shown that techniques do exist which can redistribute nodes with a fixed connectivity so that better representations, by the interpolant, can be obtained. However it is also shown that the error equalisation technique set out here is very poor at nodal redistribution.

The spring analogy criterion is seen to be quite good at redistributing nodes, although some idea of the form of the underlying data does seem to help with the choice of parameters and hence the final redistribution. Additionally, major improvements in representation can be seen if the redistributed nodes are then reconnected using data dependent criteria.

Further work will involve the production of an initial data dependent grid using a moving front technique, rather than relying on an initial triangulation, i.e. Delaunay, which is non-data dependent. This will, hopefully, result in the production of a fully data-dependent grid generator using moving front generation, nodal reconnection and nodal movement.

# References


[1] **G.F. Carey and H.T. Dinh**, (1985), *Grading Functions and Mesh Redistribution*, SIAM J. Num. Anal. **22**, pp 1028-1040

[2] **D. Catherall**, (1988), *A Solution-Adaptive-Grid Procedure for Transonic Flows Around Aerofoils*, RAE Technical Report 88020

[3] **J.C. Cavendish**, (1974), *Automatic Triangulation of Arbritrary Planar Domains for the Finite Element Method*, Int. J. Num. Meth. Engng. **8**, pp 679-696

[4] **B. Delaunay**, (1934), *Sur la sphere vide*, Bull.Acad.Sci. USSR (VII): Classe Sci. Mat. Nat., pp 793-800

[5] **N. Dyn, S. Rippa and D. Levin**, (1990), *Data Dependent Triangulations for Piecewise Linear Interpolation*, IMA J. Num. Anal. **10**, pp 137-154

[6] **P.R. Eiseman and G. Erlebacher**, (1987), *Grid Generation for the Solution of Partial Differential Equations*, ICASE report 87-57. NASA Langley,

[7] **R. Franke**, (1979), *A critical comparison of some methods for interpolation of scattered data*, Report NPS-53-79-003, Naval Postgraduate School

[8] **R. Lohner, K. Morgan and O.C. Zienkiewicz**, (1986), *Adaptive Grid Refinement for the Compressible Euler Equations*, in: I. Babuska *et al.* eds., Accuracy Estimates and Adaptive Refinements in Finite Element Computations (John Wiley and Sons Ltd) pp 281-297

[9] **A.J. Malcolm**, (1990), *A Survey of some Data-Dependent Criteria for Triangular Tessalations using Fixed Nodes*, Report 1/90, University of Reading

[10] **K. Miller and R.N. Miller**, (1990), *Moving Finite Elements, Part 1*, SIAM J. Num. Anal. **18**, pp 1019-1032

[11] **E.J. Nadler**, (1985), *Piecewise Linear Approximation on Triangulations of a Planar Region*, PhD thesis, Brown University

[12] **B. Palmerio, L. Fezoui, C. Olivier and A. Dervieux**, (1990), *On TVD Criteria for Mesh Adaption for Euler and Navier-Stokes Calculations*, INRIA report 1175,

[13] **E. G. Sewell**, (1972), *Automatic Generation of Triangulations for Piecewise Polynomial Approximation*, PhD Thesis, Purdue University

[14] **W.C. Thacker, A. Gonzalez and G.E. Putland**, (1980), *A Method for Automating the Construction of Irregular Grids for Storm Surge Forecast Models*, J. Comp. Phys. **37**, pp 371-387