Andrew Malcolm

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 lement 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.

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,

SNceOuffr]of1CSN523]4N1CSNucted4N4Ne

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 riteria

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, isemann 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. ach node is surrounded by a patch of triangles.

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

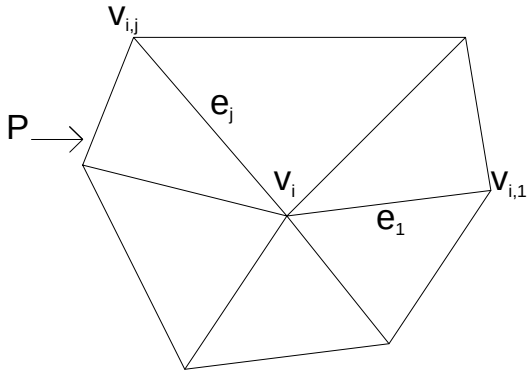


Figure 1: Patch of Triangles

with node $_i$ connected to node $_{i,j}$ by edge $_j$.

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

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

$$= \frac{_j \quad \text{extension}}{}$$

$_i$

$_j$ $_j$

$$\frac{}{_j \quad _j \quad _i \quad _{i,j}{}^2 \quad _i \quad _{i,j}{}^2}$$

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 α 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 β 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).

ventually 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 :-

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

$$_i \quad _i(\quad) \quad [\text{area}(\ _i)]^3$$

where $= \begin{matrix} ^{xx} & ^{xy} \\ ^{yx} & ^{yy} \end{matrix}$ is the Hessian at the centroid of $_i$, and $_i$ is a constant depending on the sign of .

Since for most uses we are interested in the $_2$-error, we deal with (error)^2 rather than error. To get the same (error)^2, say $_{opt}$, on each triangle in the patch we get

$$\sum_{i=1}^{P} \ _i = \ _{opt}$$

Then for each triangle in the patch

$$\frac{_i}{_{opt}} = \frac{_i \quad [\text{area}(\ _i)]^3}{_i \quad [\text{area}_{opt}(\ _i)]^3}$$

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

$$\text{area}_{opt}(\ _i) = \left(\frac{_{opt}}{_i} \right)^{1/3} \text{area}(\ _i)$$

A further condition which can be imposed is that the area of a patch should

$$\sum_{i=1}^{P} \qquad _i \qquad \sum_{i=1}^{P} \qquad \overline{\qquad _{opt}} \quad _i$$

$$\overline{\qquad _{opt}} \quad _i \qquad _{opt} \quad _i$$

$$\frac{\sum_{i=1}^{P} \qquad _i}{\sum_{i=1}^{P} \qquad _{opt} \quad _i}$$

$$\frac{\overline{\qquad _{opt} \quad _i}}{_i} \qquad \left(\frac{_{opt}}{_i} \right)^{1/3}$$

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

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

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

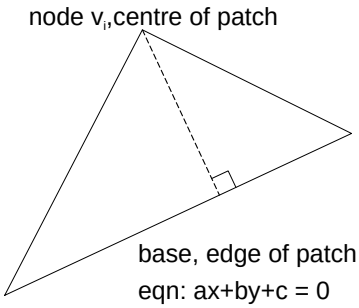


Figure 2: Triangle

Having found $\epsilon(i)$ as defined in quation (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

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

Using quations (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 :-

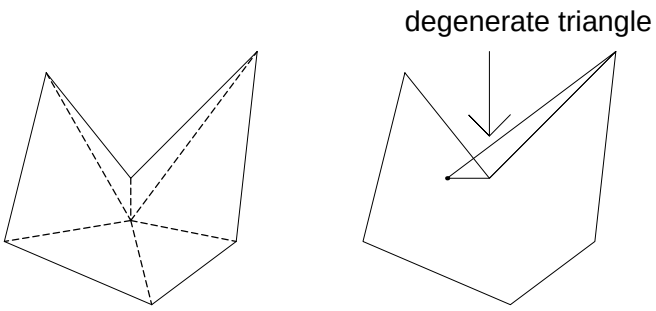
1) least squares fit :- since P lines will almost certainly not meet at a point.

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 (1) or (2), as defined by (2.3), are greater than 1.

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

( )+ (+1) ( )+ (+1) =1... =

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



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

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

i i i

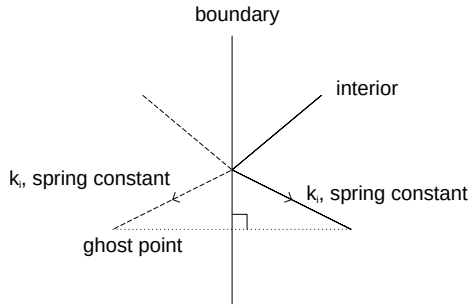


Figure 4: Ghost Points

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.

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

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].

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

H) This is the result produced by using the spring analogy with, in quation (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 quation (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 ( )-(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_2errors 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: $_2$-errors for 81 points

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 α or large β in equation (2.2) provides a better representation, while a moderate

weighting of both α and β 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] , (1985),

, SIAM J. Num. Anal. , pp 1028-1040

[2] , (1988),

, RA Technical Report 88020

[3] , (1974),

, Int. J. Num. Meth. ngng. , pp 679-696

[4] , (1934), , Bull.Acad.Sci. USSR (VII): Classe Sci. Mat. Nat., pp 793-800

[5] , (1990),

, IMA J. Num. Anal. , pp 137-154

[6] , (1987),

, ICAS report 87-57. NASA Langley,

[7] , (1979),

, Report NPS-53-79

[9] , (1990),

, Report 1/90, University of Reading

[10] , (1990), ,

SIAM J. Num. Anal. , pp 1019-1032

[11] , (1985),

, PhD thesis, Brown University

[12] , (1990),

, INRIA