Numerical Optimisation Centre

TECHNICAL REPORT NO. 136 JUNE 1983

EXPERIENCE SOLVING NONLINEAR PDE's

BY FINITE ELEMENT OPTIMISATION ON THE DAP,

1: THE LEAST SQUARES CONJUGATE GRADIENT SOLUTION

OF MORGAN'S PROBLEM

by

P G Ducksbury

The Hatfield Polytechnic

PAUL DUCKSBURY

THE HATFIELD POLYTECHNIC

NUMERICAL OPTIMISATION CENTRE

EXPERIENCE SOLVING NONLINEAR PDE's BY FINITE ELEMENT OPTIMISATION ON THE DAP, 1: THE LEAST SQUARES CONJUGATE GRADIENT SOLUTION OF MORGAN'S PROBLEM

by

P G Ducksbury

TECHNICAL REPORT NO. 136 JUNE 1983

Abstract

In a previous report by Dixon, Ducksbury & Singh [1] the solution of the two-dimensional heat conduction equation on the ICL-DAP was examined, the solution method followed the Galerkin finite element approach. This led to a set of linear simultaneous equations which was then solved by conjugate gradients. In this report we extend this approach to the solution of nonlinear problems by employing the finite element least squares approach, then completing the solution by an implementation of the nonlinear conjugate gradient algorithm.

It is shown that large nonlinear finite element problems can be solved efficiently on the ICL-DAP machine by this approach.

1. Introduction

This report describes the implementation of a parallel nonlinear conjugate gradient algorithm for the solution of Morgan's problem on the ICL-DAP computer. Comparisons are made with a sequential code which is available on the Hatfield DEC system 10 machine.

2. The Problem

The problem that was chosen to illustrate this approach was the two-dimensional Morgan equation given by

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} - Ru\frac{\partial u}{\partial x_1} = 0 \qquad (1)$$

A solution to the equation is known

$$u = {}^{6x_1}/_{Rx_2^2}$$

2.1 Solution by Least Squares

By the least squares finite element method the solution of the problem can be written as

$$\min I = \iint_V \underline{e}^T \underline{e}\, dV \qquad (2)$$

where $\underline{e} = (e_1, \ldots e_n)$ and the minimisation is subject to some suitable boundary conditions.

By letting

$$A = u,\ B = \frac{\partial u}{\partial x_1} \text{ and } C = \frac{\partial u}{\partial x_2}$$

equation (1) can be written as 3 first order equations

$$e_1 = \frac{\partial B}{\partial x_1} + \frac{\partial C}{\partial x_2} - RAB = 0 \qquad (1a)$$

$$e_2 = B - \frac{\partial A}{\partial x_1} = 0 \qquad (1b)$$

$$e_3 = C - \frac{\partial A}{\partial x_2} = 0 \qquad (1c)$$

Since we have three simultaneous equations the function (2) can be written as

$$\min I = \iint_V (e_1^2 + e_2^2 + e_3^2)dV \qquad (2a)$$

and the gradient I can be expressed as

$$\nabla I = \iiint_V \frac{\partial}{\partial \underline{z}} (e_1^2 + e_2^2 + e_3^2)\, dV \tag{3}$$

$$= 2 \iiint_V (e_1 \cdot \frac{\partial e_1}{\partial \underline{z}} + e_2 \cdot \frac{\partial e_2}{\partial \underline{z}} + e_3 \cdot \frac{\partial e_3}{\partial \underline{z}})\, dV \tag{3a}$$

$$= 2 \iiint_V \underline{e} \cdot \frac{\partial \underline{e}}{\partial \underline{z}}\, dV \tag{3b}$$

where $\underline{z}_j = (a_j, b_j, c_j)$: the three degrees of freedom at each node j.

A diagonal weighting matrix W can be introduced which modifies equations (2) and (3b) into

$$I = \iiint_V \underline{e}^T W \underline{e}\, dV \tag{4}$$

and

$$\nabla I = \iiint_V \underline{e}^T W \frac{\partial \underline{e}}{\partial \underline{z}}\, dV \tag{4b}$$

The matrix $W_{ii} = \frac{1}{3}$ $\quad i = 1, \ldots, 3$ was used.

When using finite elements the domain v is covered by a set of grid points P_k and divided into a set of elements V_i having grid points at the intersections.

On each element V_i we define 4 bilinear shape functions $\phi_{ki}(x)$ such that

$$\phi_{ki}(x) = 1 \quad \text{at } P_k \in V_i$$

$$\phi_{ki}(x) = 0 \quad \text{at } P_j \text{ for } j \neq k \; P_j \in V_i \text{ and at all points } x \notin v_i.$$

It is then possible to approximate the unknown variables A, B and C (of equations (1a) - (1c)) by linear combinations of these shape functions, as follows

$$A = \sum_k a_k \phi_{ki}(x), \quad B = \sum_k b_k \phi_{ki}(x) \quad \text{and} \quad C = \sum_k c_k \phi_{ki}(x)$$

where $x \in v_i$ and the equations (1a), (1b), (1c) respectively become

$$e_1 = \sum_{k=1}^{4} b_k \frac{\partial \phi_{ki}}{\partial x_1} + \sum_{k=1}^{4} c_k \frac{\partial \phi_{ki}}{\partial x_2} - R \sum_{k=1}^{4} a_k \phi_{ki} \sum_{k=1}^{4} b_k \phi_{ki} = 0 \tag{5}$$

$$e_2 = \sum_{k=1}^{4} b_k \phi_{ki} - \sum_{k=1}^{4} a_k \frac{\partial \phi_{ki}}{\partial x_1} = 0 \tag{5a}$$

$$e_3 = \sum_{k=1}^{4} c_k \phi_{ki} - \sum_{k=1}^{4} a_k \frac{\partial \phi_{ki}}{\partial x_2} \qquad \text{if } x \in v_i. \tag{5b}$$

The function and gradient can now be completely defined and calculated from the available information

$$F = \sum_i \iint_{el_i} (w_1 e_1^2 + w_2 e_2^2 + w_3 e_3^3)\, dV \tag{6}$$

and utilising the fact that $e_{l=1,2,3}$ only depends on z_j if $P_j \in v_i$

$$\frac{\partial F}{\partial z_j} = 2 \sum \iint_{el_i} (w_1 e_1 \frac{\partial e_1}{\partial \underline{z}_j} + w_2 e_2 \frac{\partial e_2}{\partial \underline{z}_j} + w_3 e_3 \frac{\partial e_3}{\partial \underline{z}_j})\, dV \tag{7}$$

where the Jacobian $\frac{\partial e_1}{\partial \underline{z}}$ is obtained by differentiating equations (5) - (5b). If $x \in V_i$ then a typical term $\frac{\partial e_1}{\partial a_k}(x)$ is identically zero if $P_k \notin V_i$. If $P_k \in V_i$ the expressions for the derivatives are given in Table 1 where the summation is over the four shape functions relating to V_i.

e_1	e_2	e_3
$\frac{\partial e_1}{\partial a_k} = - R\phi_{ki} \sum_{j=1}^{4} b_j \phi_{ji}$	$\frac{\partial e_2}{\partial a_k} = - \frac{\partial \phi_{ki}}{\partial x_1}$	$\frac{\partial e_3}{\partial a_k} = - \frac{\partial \phi_{ki}}{\partial x_2}$
$\frac{\partial e_1}{\partial b_k} = \frac{\partial \phi_{ki}}{\partial x_1} - R\phi_{ki} \sum_{j=1}^{4} a_j \phi_{ji}$	$\frac{\partial e_2}{\partial b_k} = \phi_{ki}$	$\frac{\partial e_3}{\partial b_k} = 0$
$\frac{\partial e_1}{\partial c_k} = \frac{\partial \phi_{ki}}{\partial x_2}$	$\frac{\partial e_2}{\partial c_k} = 0$	$\frac{\partial e_3}{\partial c_k} = \phi_{ki}$

Table 1

In Section 3 we briefly introduce the conjugate gradient approach and then describe the nonlinear algorithm used to complete the solution on the DAP parallel processor.

2.2 Boundary Conditions

The problem was solved for a square domain (4.0, 4.0) to (4.5, 4.5) with the values at the boundaries being fixed at their values from the given solution, i.e.

$$A = u = \frac{6x_1}{Rx_2^2}, \quad B = u_{x_1} = \frac{6}{Rx_2^2}, \quad C = u_{x_2} = - \frac{12x_1}{Rx_2^3}$$

3. The Conjugate Gradient Method

The conjugate gradient method was first introduced for solving linear simultaneous equations by Hestenes and Stiefel (1952)[2] and then modified for use in solving nonlinear optimisation problems by Fletcher and Reeves (1964)[3].

A modified form of the Fletcher Reeves Algorithm has been programmed on the DAP specially to solve the nonlinear optimisation problems that arise when solving nonlinear partial differential equations by the least squares finite element formulation.

It is our intention to show that this combination allows such nonlinear P.D.E.s to be solved efficiently on the DAP.

3.1 The Algorithm

The conjugate gradient algorithm we used is based on the Fletcher and Reeves [3] proposals and is described below.

1. Select an initial estimate for $\underline{x}^{(k)}$ where $k = 0$.
2. Perform initialisation for the line search.
3. Calculate $f(\underline{x}^k)$.
4. Calculate $g(\underline{x}^k)$.
5. Evaluate $g(\underline{x}^k)^T.g(\underline{x}^k)$.
6. If either $g(\underline{x}^k)^T.g(\underline{x}^k) < \varepsilon_o$ or $k >$ maximum number of iterations THEN STOP.
7. If first iteration

   THEN set direction $\underline{p}^{(k)} = -g(\underline{x}^k)$

   OTHERWISE set $\beta^{(k)} = \dfrac{g(\underline{x}^k)^T.g(\underline{x}^k)}{g(\underline{x}^{k-1})^T.g(\underline{x}^{k-1})}$

   and then $\underline{p}^{(k)} = - g(\underline{x}^k) + \beta^{(k)}.\underline{p}^{(k-1)}$.
8. Perform scaling of the step length α^k prior to the line search.

9. Call a suitable line search routine.

10. Set $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \alpha^{(k)} . \underline{p}^{(k)}$.

11. Set k = k + 1.

12. Go to Step 4.

The line search used at Step 8 combines the quadratic prediction technique with the Armijo [4] method.

4. Implementation

In the algorithm there are just three items that need to be evaluated, namely,

(a) $F = \sum_{i=1}^{E} F_i$

where F_i is the contribution from element i,

(b) $\nabla F_j = g_j = \sum_{i \epsilon el_j}^{4} \frac{\partial F_i}{\partial z_k}$

where el_j are the four elements containing P_j and where k is the local node in element el_i which correspondsto the global node j.

Note there are 3 components to $^{\partial F}/_{\partial z_j}$.

(c) $g^T g = \sum_{j=1}^{N} (\sum_{i \epsilon el_j}^{4} \frac{\partial F_i}{\partial z_k})(\sum_{i \epsilon el_j}^{4} \frac{\partial F_i}{\partial z_k})$

for convenience let us define

$$g_i^{el} = \frac{\partial F_i}{\partial z_k}$$

where N = no. of nodes

E = no. of elements.

In implementing this onto the DAP the calculations for each gridpoint are carried out on 1 processor. The details of the calculations are shown in Appendix 3.

5. Results

The problem was solved over the square domain (4.0,4.5) with an initial solution which was 90% of the true solution. The termination of the conjugate gradient iteration occurred when $g^T g < \varepsilon_o$ (=1E-10)(with a fail-safe on the maximum number of iterations). Because of the number of unknowns involved (3*Number of points) the errors were examined by looking only at the centre grid point of the mesh i.e. (4.25,4.25). The results obtained are given in Appendix 1 with some associated graphs in Appendix 2. The first test consisted of taking a mesh size of h = 0.125 (a 5 x 5 grid) and then varying R from 5000 down to 5. For this test the DAP results are given in Table 1 and the corresponding sequential ones in Table 2. It can be seen that there is a good correspondence between the parallel and sequential results with only a slight difference in the timings.

The second test carried out was the same as the previous one except that the mesh size was reduced to h = 0.0625, (a 9 x 9 grid : parallel on Table 3, sequential on Table 4). Again we see a good correspondence between the parallel and sequential results but the timings are now more favourable for the DAP.

It became evident from examining these two tests that the problem was more difficult to solve for small values of R.

In Test 3, (Table 5 - DAP only), R was fixed at 500 and the grid size varied up to 63 x 63. The results show unexpectedly that refining the mesh size made no improvement on the solution (at least to within the accuracy specified by E_o). No sequential runs were made for this test because it was considered the CPU time would be excessive.

This test was then repeated for R =.5. The results of Test 4 are shown in Tables 6 (DAP) and 7 (sequential). We can see from the results that the function value is gradually being reduced as are the errors in the

second and third unknown (B and C) as the mesh size is refined. However, the smaller error in U is fluctuating. Again these results demonstrate the speed up obtained by using the parallel processor although even with the finest mesh tested less than ⅛ of the DAP's processors were in use. Appendix 2 contains graphs illustrating some of the results mentioned previously. Graphs 1 and 2 show the relationship between LOG (R) and time for h = 0.125 and h = 0.0625 respectively. Graphs 3 and 4 illustrate the results obtained from Table 5 (DAP), namely, CPU time versus grid size (a linear relationship) and effective function evaluations (eFe's) versus grid size (approximately of order n^2) respectively. Graph 5 (DAP and 1091) shows the relative CPU times versus grid size for R = 0.5.

6. Conclusions

In this report we have shown that the ICL DAP machine can be used to solve nonlinear P.D.E.s. In particular it has been demonstrated that when the problem is formulated as an optimisation problem by the least squares approach and when this problem is solved by the conjugate gradient approach, all the calculations can be mapped efficiently onto the DAP's parallel processors. This extends the work on linear P.D.E.s reported in Dixon, Ducksbury and Singh [1].

A number of improvements can be made to the existing algorithm. The present implementation was written and tested before the recent extension of the DAP store (from 2Mb to 8Mb). This extension would enable the run times to be substantially reduced as, for example, in the calculation of $\frac{\partial \phi_i}{\partial x_1}$ and $\frac{\partial \phi_i}{\partial x_2}$ per element. We will have 4 values w.r.t. x and 4 values w.r.t. y (1 for each node) and, since we are using a 3 x 3 set of quadrature points, this then gives a total of (4 + 4) x 9 = 72 values involved

in the calculation for each element. This set of values once calculated for each element remain constant. However, with the 2Mb DAP the program size and workspace needed meant that we had to recalculate $\frac{\partial \phi_i}{\partial x_1}$ and $\frac{\partial \phi_i}{\partial x_2}$ at each iteration. With the 8Mb DAP this can be avoided by storing at the first iteration.

The run times should also be substantially reduced when a suitable preconditioned form of the conjugate gradient algorithm is available on the DAP. It is, of course, essential that the arithmetic implied by such preconditioning should map efficiently onto the DAP's parallel processors. It is hoped to report on such a preconditioner shortly.

References

1. Dixon, L C W, Ducksbury, P G and Singh, P
A Parallel Version of the Conjugate Gradient Algorithm for Finite Element Problems, NOC TR. 132, The Hatfield Polytechnic, December 1982.

2. Hestenes, M and Stiefel, E
Methods of Conjugate Gradients for Solving Linear Systems, J.Res.Nat. Bu.Standards, No. 49, 1952.

3. Fletcher, R and Reeves, C M
Function Minimisation by Conjugate Gradients, Computer Journal, Vol.7, 1964.

4. Armijo, L
Pacific J.Math., No.16, pp 1-16.

APPENDIX 1

TEST		ICL-DAP	DEC 1091
1	H = 0.125 Varying R	TABLE 1 GRAPH 1	TABLE 2 GRAPH 1
2	H = 0.0625 Varying R	TABLE 3 GRAPH 2	TABLE 4 GRAPH 2
3	R = 500 Varying H	TABLE 5 GRAPHS 3 & 4	-
4	R = 0.5 Varying H	TABLE 6 GRAPH 5	TABLE 7 GRAPH 5

Notes

1. The region of solution was (4.0,4.0).(4.5,4.5).
2. The three errors (U, Ux1, Ux2) quoted, were in each case taken from the centre node located at (4.25,4.25).
3. The terminating tolerance used was 1E-10, except for Tables 6 & 7 where it became necessary to increase it to 1E-8.
4. The initial solution used was 90% of the true solution.

R	ERROR AT CENTRE NODE			GTG	FVAL	EFE	CPU (m:sec)
	U	Ux1	Ux2				
5000	2.018875 E-6	6.14395 E-6	1.286367 E-5	6.496446 E-11	4.828166 E-11	317	5.89
1000	1.596054 E-6	3.12735 E-5	6.48396 E-5	9.99654 E-11	5.47037 E-10	309	3.92
500	3.933907 E-6	5.945168 E-5	1.26476 E-4	4.87958 E-11	1.836411 E- 9	465	6.37
100	2.268731 E-6	1.53437 E-5	1.028329 E-4	9.987313 E-11	5.017881 E- 9	1233	15.65
50	1.341119 E-6	6.456015 E-6	5.453898 E-6	9.187348 E-11	1.333588 E- 8	2694	33.74
10	2.44379 E-6	4.56459 E-5	2.72512 E-4	5.72106 E-11	2.96056 E- 7	4853	1:01.8
5	1.788139 E-7	8.87524 E-5	5.378127 E-4	9.366781 E-11	1.183997 E- 6	4776	1:08.6

Table 1.

R	ERROR AT CENTRE NODE			GTG	FVAL	EFE	CPU (m:sec)
	U	Ux1	Ux2				
5000	2.01928 E-6	6.14399 E-6	1.28637 E-5	6.49457 E-11	4.52776 E-11	317	7.83
1000	1.59610 E-6	3.12738 E-5	6.48397 E-5	9.965126 E-11	5.470226 E-10	309	5.97
500	3.93356 E-6	5.94519 E-5	1.26476 E-4	4.879238 E-11	1.836379 E- 9	465	9.15
100	2.25531 E-6	1.53380 E-5	1.02789 E-4	9.931477 E-11	5.017751 E- 9	1233	23.09
50	1.31899 E-6	6.99254 E-6	5.21145 E-4	8.388215 E-11	1.33247 E- 8	2694	49.21
10	1.27219 E-6	3.23551 E-6	2.95801 E-4	6.001879 E-11	2.960657 E- 7	4619	1:28.77
5	4.06057 E-7	9.19895 E-5	5.3223 E-4	9.764816 E-11	1.184153 E-6	4465	1:33.02

Table 2.

R	ERROR AT CENTRE NODE			GTG	FVAL	EFE	CPU (m:sec)
	U	Ux1	Ux2				
5000	8.407642 E-7	6.267248 E-6	1.299415 E-5	1.799766 E-11	3.472132 E-11	1471	5.88
1000	2.07545 E-6	3.11399 E-5	6.48736 E-5	9.363889 E-11	5.65534 E-10	1714	6.37
500	6.62878 E-7	5.94607 E-5	1.2758 E-4	7.89408 E-11	1.772412 E- 9	2454	10.57
100	1.49011 E-8	5.40586 E-6	1.811236 E-5	6.50855 E-11	2.82259 E- 9	4899	19.09
50	1.326218 E-6	2.38567 E-5	1.04989 E-4	8.95349 E-11	6.69263 E- 9	9065	36.07
10	8.34465 E-7	8.355943 E-6	1.102106 E-4	7.311977 E-11	9.13534 E- 8	27685	1:50.5
5	1.013279 E-6	2.765679 E-5	2.111823 E-4	9.627174 E-11	3.635412 E- 7	29891	1:59.5

Table 3.

R	ERROR AT CENTRE NODE			GTG	FVAL	EFE	CPU (m:sec)
	U	Ux1	Ux2				
5000	8.40475 E-7	6.26729 E-6	1.29942 E-5	1.7997 E-11	3.472002 E-11	1471	39.78
1000	2.07479 E-6	3.11399 E-5	6.4874 E-5	9.363055 E-11	5.655051 E-10	1714	44.79
500	6.61792 E-7	5.94612 E-5	1.275811 E-4	7.89354 E-11	1.772325 E- 9	2454	1:06.3
100	7.79982 E-9	5.39614 E-6	1.814886 E-5	6.49394 E-11	2.82243 E- 9	4899	2:02.5
50	1.37533 E-6	2.37963 E-5	1.04823 E-4	8.80158 E-11	6.6921 E- 9	9065	3:40.7
10	2.03028 E-7	9.88552 E-6	1.182342 E-4	9.80005 E-11	9.23273 E- 8	24009	9:26.4
5	3.27825 E-7	2.89389 E-5	2.13088 E-4	9.480157 E-11	3.635957 E- 7	28174	10:32.6

Table 4.

GRID SIZE	ERROR AT CENTRE NODE			GTG	FVAL	EFE	CPU (m:sec)
	U	Ux1	Ux2				
5	3.933907 E-6	5.945168 E-5	1.26476 E-4	4.87958 E-11	1.836411 E-9	465	6.37
9	6.62878 E-7	5.94607 E-5	1.2758 E-4	7.89408 E-11	1.772412 E-9	2454	10.57
19	1.21444 E-6	6.07164 E-5	1.2899 E-4	5.246011 E-11	1.880792 E-9	20,614	18.18
39	7.26209 E-7	6.1573 E-5	1.29528 E-4	8.27902 E-11	2.163079 E-9	159,774	33.99
51	6.34239 E-7	6.20207 E-5	1.29736 E-4	9.75599 E-11	2.38465 E-9	327,809	40.94
63	7.60901 E-7	6.197067 E-5	1.29727 E-4	9.7236 E-11	2.39782 E-9	619,267	50.76

Table 5.

Sequential

5×5 9·5
9×9 1:06
12×12 2:04·86
19×19 8:43·31 $g^T g = 5.2445\ E-11$ $F = 1.88067 E-9$

GRID SIZE	ERROR AT CENTRE NODE U	Ux1	Ux2	GTG	FVAL	EFE	CPU (m:sec)
3	8.7738 E-5	1.9925 E-3	8.80814 E-3	2.61105 E-9	3.79527 E-4	695	23.75
5	2.86102 E-6	9.2995 E-4	5.2938 E-3	8.82436 E-9	1.184173 E-4	4544	57.75
7	5.72205 E-6	7.8845 E-4	5.3913 E-3	8.4086 E-9	5.96144 E-5	14,459	1:40.0
9	8.58306 E-6	2.86162 E-4	2.13339 E-3	9.10557 E-9	3.63568 E-5	28,912	1:56.1
19	5.72204 E-6	1.5974 E-5	1.94549 E-4	9.52492 E-9	8.33969 E-6	345,050	5:17.84

Table 6.

GRID SIZE	ERROR AT CENTRE NODE U	Ux1	Ux2	GTG	FVAL	EFE	CPU (m:sec)
3	6.94692 E-5	1.126386 E-3	8.85025 E-3	2.51266 E-9	3.79552 E-4	753	10.93
5	2.89082 E-6	1.34527 E-4	7.2335 E-3	9.9228 E-9	1.18825 E-4	4773	1:27.3
7	1.78814 E-6	8.67128 E-4	5.47384 E-3	5.03387 E-9	5.96167 E-5	14,159	5:38.71
9	7.8082 E-6	2.79396 E-4	2.076134 E-3	7.61803 E-9	3.63683 E-5	28,174	10:33.7
19	NOT RUN DUE TO CPU TIME INVOLVED						

Table 7.

* 12 1.78813E-6 8.7485E-5 9.4947E-4 5.8438 E-11 2.050894E-5 99,830 42:13.25

15 8.9407E-7 5.95301E-5 3.5615E-4 8.46119E-11 1.31812E-5 203,814 1:27:55.8

19 1.66893E-6 2.2083E-5 5.7592E-5 9.2818E-11 8.2425E-6 401,464 2:49:44.63

APPENDIX 2

GRAPH 1	H = 0.125 Varying R	CPU TIME IN SECONDS v LOG (REYNOLDS NUMBER). TWO LINES: ONE FOR DAP ONE FOR DEC 10
GRAPH 2	H = 0.0625 Varying R	
GRAPH 3	R = 500 Varying H	CPU TIME IN SECONDS v THE NUMBER OF UNKNOWNS
GRAPH 4		EFE's v THE NUMBER OF UNKNOWNS
GRAPH 5	R = 0.5 Varying H	CPU TIME v NO. OF UNKNOWNS

Notes

1. EFE's are Effective Function Evaluations and they act as a measure of the amount of work involved in the computation

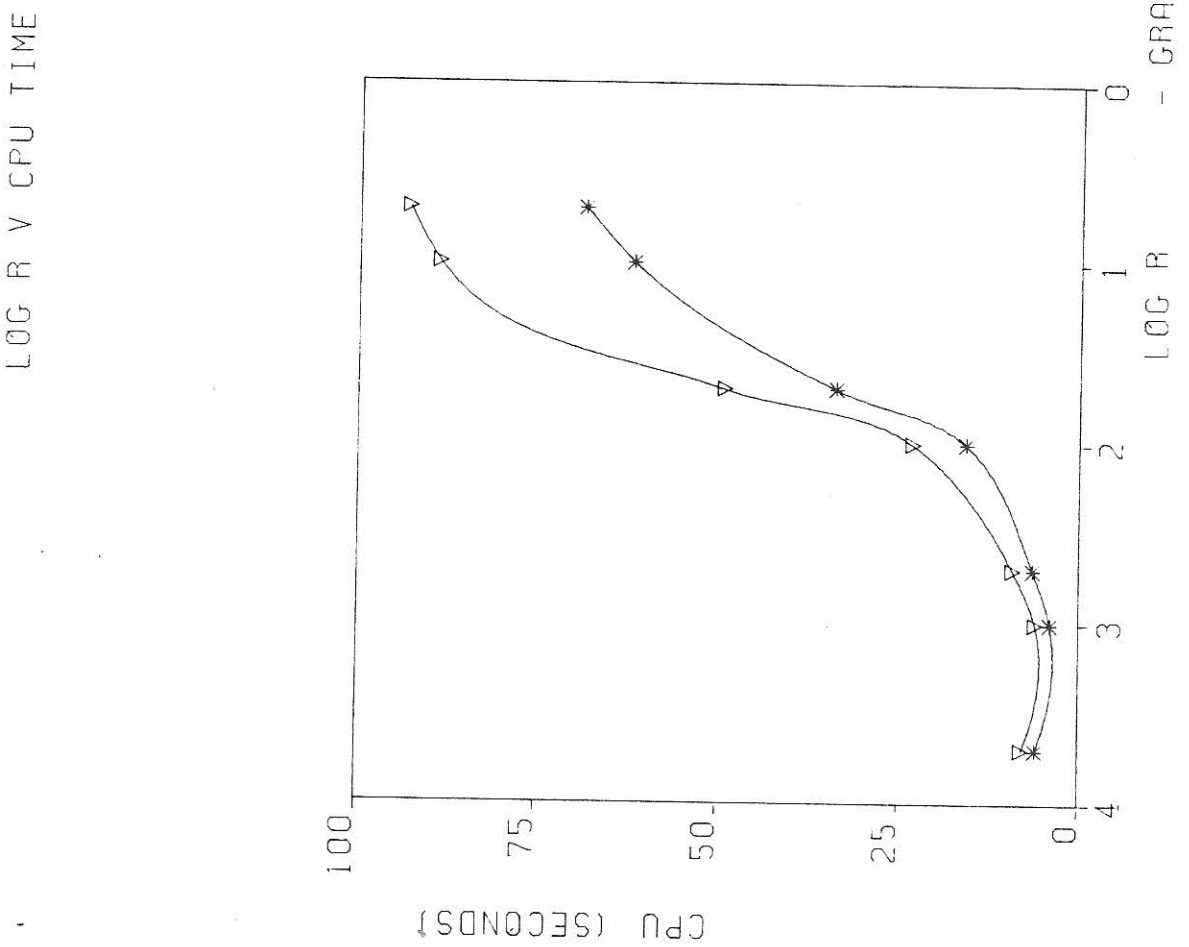
$$EFE's = F + n.G$$

where

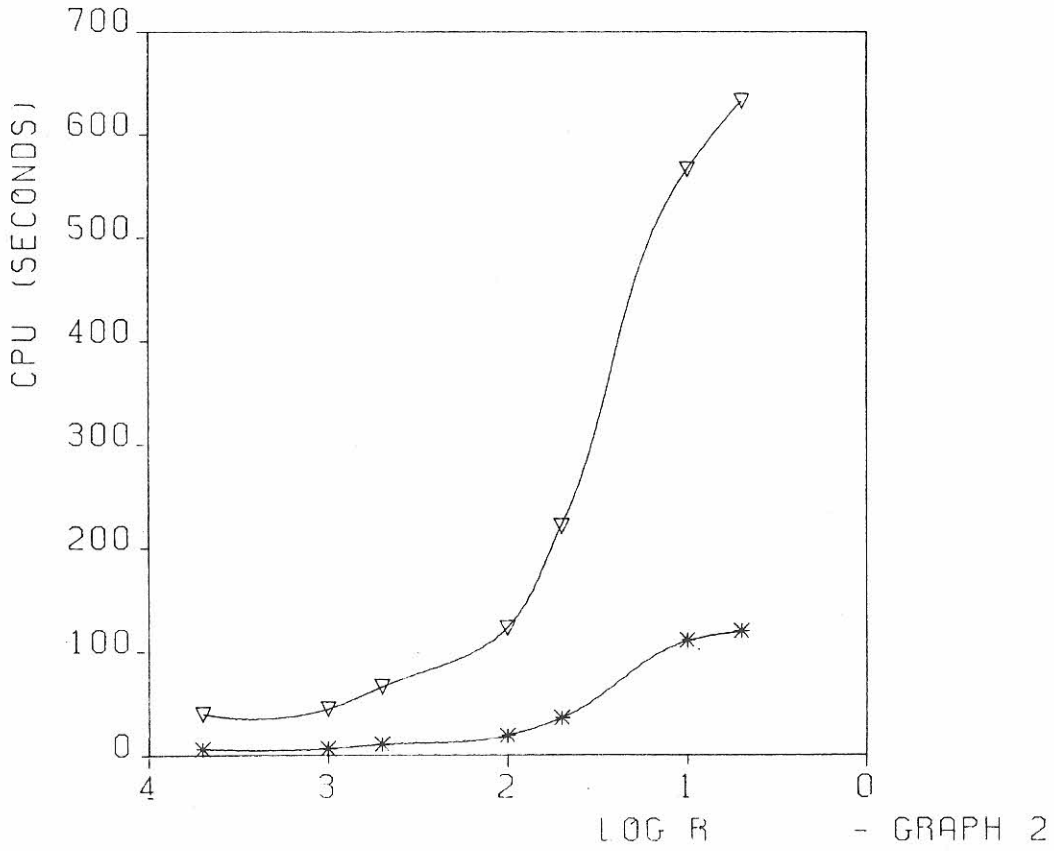
F = the number of function calls,

n = the number of unknowns,

G = the number of gradient calls.

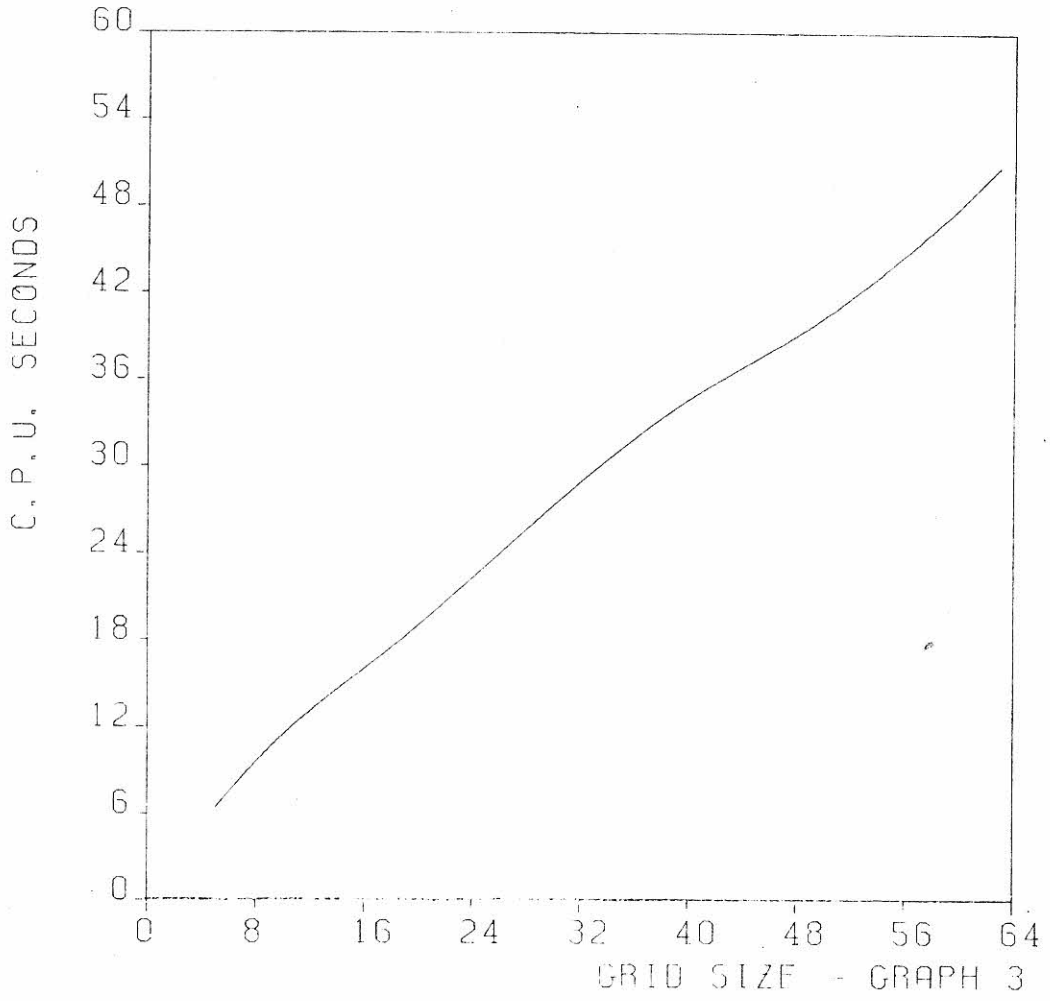

LOG R V CPU TIME
* DAP, 5X5
▽ 1091, 5X5
CPU (SECONDS)
100
75
50
25
0
0
1
2
3
4
LOG R
- GRAPH 1

LOG R V CPU TIME

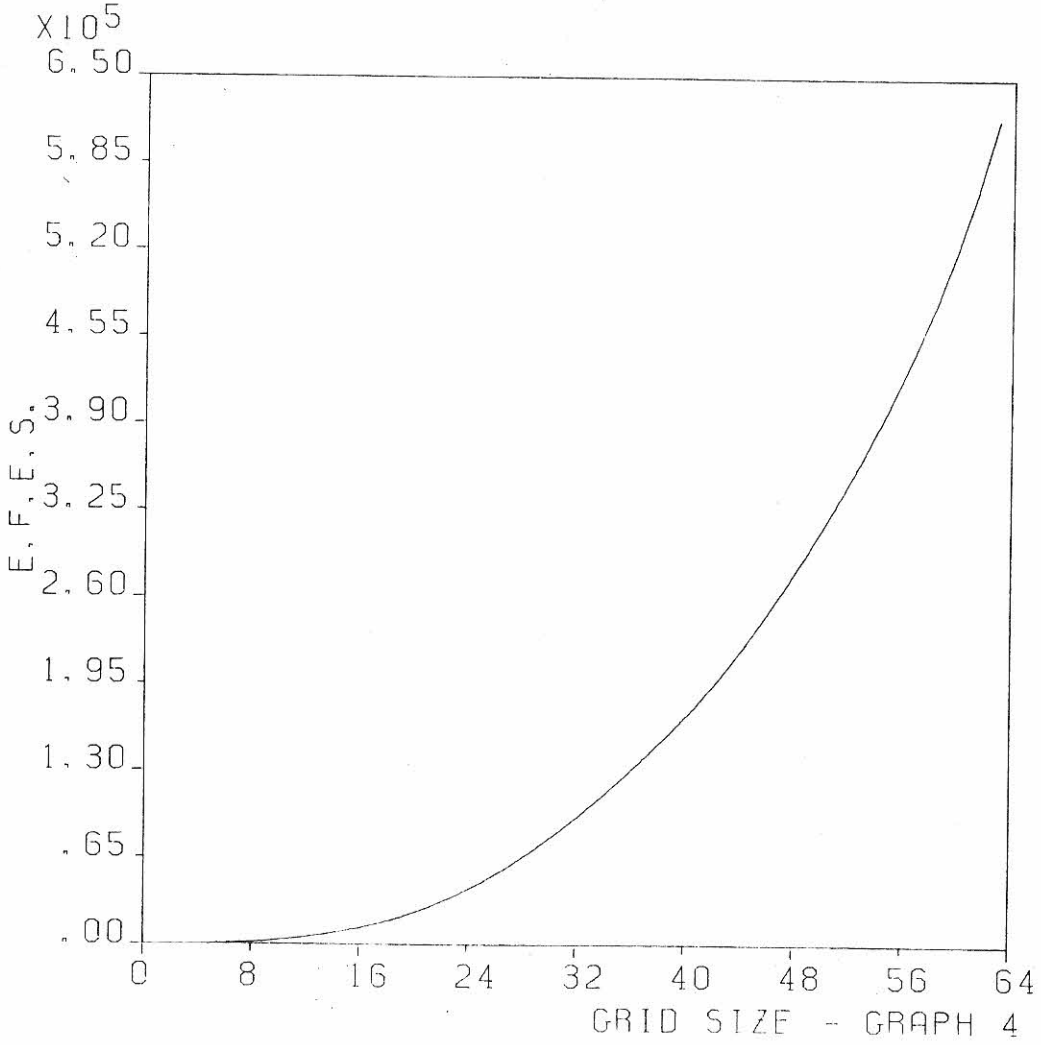


- GRAPH 2

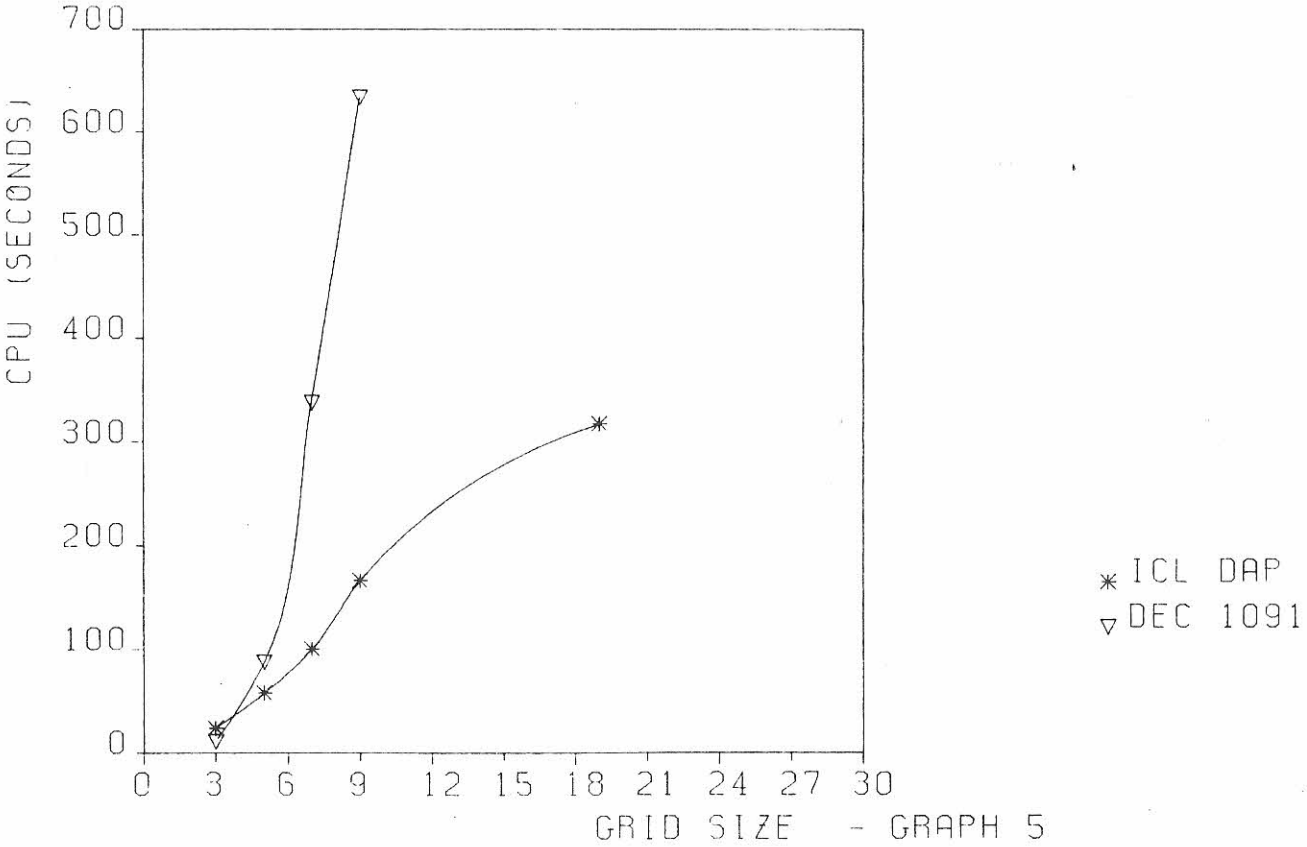
UNKNOWNS V CPU TIME



UNKNOWNS V E.F.E.S



UNKNOWNS V CPU TIME



APPENDIX 3

Mapping onto the DAP

The mapping used is the same as that described in Dixon, Ducksbury and Singh [1] with an extension to incorporate the three degrees of freedom per node.

Given an n x n set of grid points there is an (n-1) x (n-1) set of elements, both grid points and the element information are held one per processor as in Figure 1.

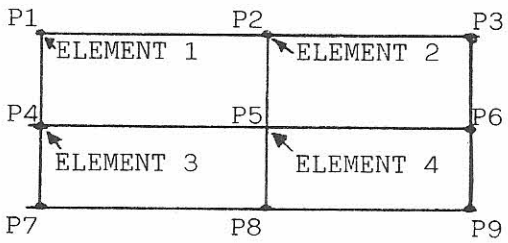


Figure 1. (3 x 3 global storage)

Two logical masks are created to mask out any unwanted grid points and elements.

The calculation of g at the grid points is illustrated, given that the least squares code has been performed and in each element we have 12 values for g^{el} (3 unknowns x 4 nodes) stored consecutively in an array VGEL (,, 12). Local node numbering and the element information storage is given below in Figure 2.

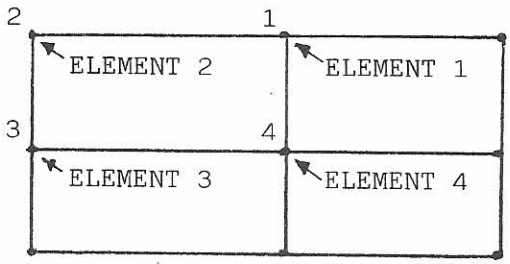


Figure 2.

Figure 3 shows how each group of three values (from the 12) are mapped onto the nodes. Assume for the example that we are centred on node 5 and wish to calculate the 3 values of g at this point.

The information from element 1 will come by a shift index (-,)

" " " 2 " " " " " (-,-)

" " " 3 " " " " " (,-)

and " " " 4 is at our current position, i.e. node 5.

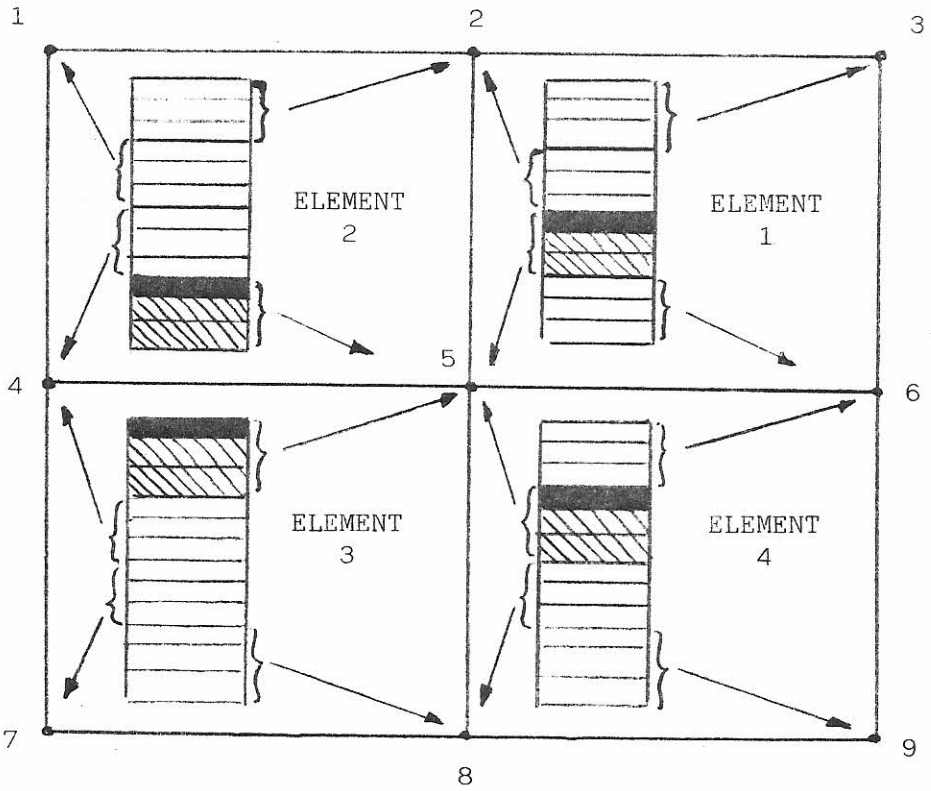


Figure 3.

Using this information and the four groups indicated on Figure 3 we can evaluate g.

1st Unknown

VG(,,1) = 1st value of 3rd group from element 1

+ 1st value of 4th group from element 2

+ 1st value of 1st group from element 3

+ 1st value of 2nd group from element 4

= VGEL(-,,7) + VGEL(-,-,10) + VGEL(,-,1) + VGEL(,,4).

2nd Unknown

VG(,,2) = 2nd value of 3rd group from element 1

+ 2nd value of 4th group from element 2

+ 2nd value of 1st group from element 3

+ 2nd value of 2nd group from element 4

= VGEL(-,,8) + VGEL(-,-,11) + VGEL(,-,2) + VGEL(,,5).

3rd Unknown

```
VG(,,3) = VGEL(-,,9) + VGEL(-,-,12) + VGEL(,-,3) + VGEL(,,6)
```

which can, of course, be generalised to

```
DO 100 I = 1,3
 VG(,,I) = VGEL(-,,I + 6) + VGEL(-,-,I + 9) + VGEL(,-,I) + VGEL(,,I + 3)
100 CONTINUE
```

The vector product $g^T g$ can easily be performed as

```
    GTG = 0.0
    DO 150 I = 1,3
    GTG = GTG + SUM(VG(,,I)**2)
150 CONTINUE
```

Appendix 4 contains the details of the implementation of the least squares calculation used to produce VGEL.

APPENDIX 4

The calculation of the gradient as mentioned in Section 2.1 (Equation 7) is given by

$$2 \iiint (e_1 w_1 \frac{\partial e_1}{\partial \underline{z}} + e_2 w_2 \frac{\partial e_2}{\partial \underline{z}} + e_3 w_3 \frac{\partial e_3}{\partial \underline{z}}) \, dV$$

Now considering one element with the local node numbering as in Figure 1,

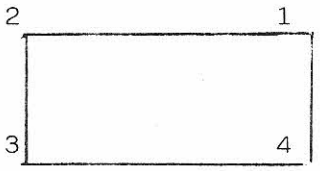


Figure 1.

and using the information given in Table 1, Section 2.1, we get 12 equations as follows.

From Node 1

$$\text{Unknown 1} = - e_1 w_1 R\phi_1 \Sigma\phi_i b_i - e_2 w_2 \frac{\partial \phi_1}{\partial x_1} - e_3 w_3 \frac{\partial \phi_1}{\partial x_2}$$

$$\text{Unknown 2} = e_1 w_1 \frac{\partial \phi_1}{\partial x_1} - e_1 w_1 R\phi_1 \Sigma a_i \phi_i + e_2 w_2 \phi_1$$

$$\text{Unknown 3} = e_1 w_1 \frac{\partial \phi_1}{\partial x_2} + e_3 w_3 \phi_1$$

From Node 2

$$\text{Unknown 1} = - e_1 w_1 R\phi_2 \Sigma\phi_i b_i - e_2 w_2 \frac{\partial \phi_2}{\partial x_1} - e_3 w_3 \frac{\partial \phi_2}{\partial x_2}$$

$$\text{Unknown 2} = e_1 w_1 \frac{\partial \phi_2}{\partial x_1} - e_1 w_1 R\phi_2 \Sigma a_i \phi_i + e_2 w_2 \phi_2$$

$$\text{Unknown 3} = e_1 w_1 \frac{\partial \phi_2}{\partial x_2} + e_3 w_3 \phi_2$$

From Node 3

$$\text{Unknown 1} = e_1 w_1 R\phi_3 \Sigma\phi_i b_i - e_2 w_2 \frac{\partial \phi_3}{\partial x_1} - e_3 w_3 \frac{\partial \phi_3}{\partial x_2}$$

$$\text{Unknown 2} = e_1 w_1 \frac{\partial \phi_3}{\partial x_1} - e_1 w_1 R\phi_3 \Sigma a_i \phi_i + e_2 w_2 \phi_3$$

$$\text{Unknown 3} = e_1 w_1 \frac{\partial \phi_3}{\partial x_2} + e_3 w_3 \phi_3$$

From Node 4

$$\text{Unknown 1} = e_1 w_1 R\phi_4 \Sigma\phi_i b_i - e_2 w_2 \frac{\partial\phi_4}{\partial x_1} - e_3 w_3 \frac{\partial\phi_4}{\partial x_2}$$

$$\text{Unknown 2} = e_1 w_1 \frac{\partial\phi_4}{\partial x_1} - e_1 w_1 R\phi_4 \Sigma a_i \phi_i + e_2 w_2 \phi_4$$

$$\text{Unknown 3} = e_1 w_1 \frac{\partial\phi_4}{\partial x_2} + e_3 w_3 \phi_4$$

The important point to note for the implementation is the similarity in the above equations.

Let us first make the following assumptions:

i) We are at node 2 in Figure 1

therefore to get to node 1 we use a shift index of (,+)

" " " " 2 no shift required

" " " " 3 we use a shift index of (+,)

" " " " 4 " " " " " " (+,+)

ii) $\frac{\partial\phi_i}{\partial x_1}$, $\frac{\partial\phi_i}{\partial x_2}$, ϕ_i and w_j are readily available as PDX(,,4), PDY(,,4), SN(4), WT(3)

iii) U(,,3) contains the complete set of unknowns

i.e. U(,,1) contains all the a_i's

U(,,2) " " " b_i's

and U(,,3) " " " c_i's.

The calculations can now be performed as follows:

a) $\Sigma a_i \phi_i$, $\Sigma b_i \phi_i$ and $\Sigma c_i \phi_i$

ASN(,) = U(,+,1)*SN(1) + U(,,1)*SN(2) + U(+,,1)*SN(3) + U(+,+,1)*SN(4)

BSN(,) = U(,+,2)*SN(1) + U(,,2)*SN(2) + U(+,,2)*SN(3) + U(+,+,2)*SN(4)

CSN(,) = U(,+,3)*SN(1) + U(,,3)*SN(2) + U(+,,3)*SN(3) + U(+,+,3)*SN(4).

b) e_1, e_2 and e_3 as defined by equations (5), (5a) and (5b) in Section 2.1

E1(,) = PDX(,,1)*U(,+,2) + PDY(,,1)*U(,+,3)

```
          + PDX(,,2)*U(,,2) + PDY(,,2)*U(,,3)
          + PDX(,,3)*U(+,,2) + PDY(,,3)*U(+,,3)
          + PDX(,,4)*U(+,+,2) + PDY(,,4)*U(+,+,3)
          - REN*ASN*BSN
    E2(,) = BSN - PDX(,,1)*U(,+,1) - PDX(,,2)*U(,,1)
                - PDX(,,3)*U(+,,1) - PDX(,,4)*U(+,+,1)
    E3(,) = CSN - PDY(,,1)*U(,+,1) - PDY(,,2)*U(,,1)
                - PDY(,,3)*U(+,,1) - PDY(,,4)*U(+,+,1).
```

We now have the information necessary to calculate the twelve equations previously mentioned. We will first make the declaration EGD(,,12) for storing them and the code becomes:

```
    DO 100 I = 1,4
      PLANE = 2*(I - 1) + I
C
C   Plane will always point to the start of group i.
C
C   Calculate unknown 1 for node i.
C
    EGD(,,PLANE) = - E1*REN*WT(1)*SN(1)*BSN - E2*WT(2)
                   *PDX(,,I) - E3*WT(3)*PDY(,,I)
C
C   Calculate unknown 2 for node i.
C
    EGD(,,PLANE + 1) = E1*WT(1)*PDX(,,I) - E1*WT(1)*REN
                       *SN(I)*ASN + E2*WT(2)*SN(I)
C
C   Calculate unknown 3 for node i.
```

```
C
      EGD(,,PLANE + 2) = E1*WT(1)*PDY(,,I) + E3*WT(3)*SN(I)
  100 CONTINUE
      DO 150 I = 1,12
      VGEL(,,I) = VGEL(,,I) + EGD(,,I)*GMULT
  150 CONTINUE
```

where GMULT is an integration factor described below.

This has now given us the twelve required values for the complete set of elements. The only additional point to be mentioned is the fact that we are using a 3 x 3 quadrature set, hence the above code must be contained within a loop which will obtain the appropriate shape functions from the quadrature, calculate the appropriate integration factor GMULT and then calculate a contribution for each of the 9 quadrature points and accumulate the integrated sum in VGEL(,,12) as is indicated.

There may be some redundancy in the above mentioned code but this has been left in to avoid complicating the example with additional loops and variables.