INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 17, 449–461 (1993)

A FULLY IMPLICIT DIRECT NEWTON'S METHOD FOR THE STEADY-STATE NAVIER–STOKES EQUATIONS*

DANA A. KNOLL AND PAUL R. McHUGH

Idaho National Engineering Laboratory, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, Idaho 83415-2414, U.S.A.

SUMMARY

Newton's method and banded Gaussian elimination can be a CPU efficient method for steady-state solutions to two-dimensional Navier–Stokes equations. In this paper we look at techniques that increase the radius of convergence of Newton's method, reduce the number of times the Jacobian must be factored, and simplify evaluation of the Jacobian. The driven cavity and natural convection problems are used as test problems, and finite volume discretization is employed.

KEY WORDS Fully implicit Modified Newton's method Numerical Jacobian Mesh sequencing Navier–Stokes

INTRODUCTION

Robust, fully implicit, direct solution techniques for solving Navier–Stokes equations have become increasingly popular with advances in computer technology. While direct methods require more computer memory than iterative methods, they provide a more dependable numerical alogorithm. These methods have been used for some time by the finite element community,[1] but have only been considered recently in the finite volume community. In cases of non-linear, strongly coupled fluid equations, the direct methods have been shown to be more CPU efficient than iterative methods. In this study we use Newton's method to solve the two-dimensional (2-D) driven cavity and natural convection problems using finite volume discretization. While previous research[2–4] compared various direct methods to popular iterative methods, this work concentrates on improving CPU efficiency and simplifying the use of Newton's method. Improved CPU efficiency is achieved by minimizing the number of times large Jacobian matrices must be formed and factored. Simplification is achieved by minimizing the programming complexity associated with forming the Jacobian matrix. The required number of Jacobian factors is reduced by using mesh sequencing and two different modified Newton iteration algorithms. Mesh sequencing can be thought of as multigrid in one direction. It serves the purpose of providing a good initial guess to the solution on the final grid. By modified Newton iteration, we refer to the use of a single Jacobian for two or more successive iterations. Since forming and factoring the Jacobian often account for more than 90% of the CPU time for a single iteration, successful use of a modified Newton iteration can lead to substantial CPU time savings. However, the price paid is the loss of quadratic convergence. The modified Newton iteration algorithms used here allow the code to determine adaptively when a new Jacobian

* Work performed under the auspices of the U.S. Dept. of Energy (DOE), Idaho Operations Office, under DOE Contract No. DE-AC07-76ID01570, the INEL Long Term Research Initiative in Computational Mechanics.

0271–2091/93/180449–13$11.50

Received January 1992
Revised January 1993

matrix must be formed and factored. As the physics of the problem becomes complex, the task of deriving and coding an analytical Jacobian becomes immense and error-prone. A novel implementation and use of the numerical Jacobian is used to minimize the complexity of this task. This numerical evaluation is well suited for parallel vector supercomputer architecture.

The number of researchers studying fully implicit direct solution techniques for 2-D computational fluid dynamics is growing. Vanka,[2,5] Braaten[6] and MacArthur[3,4] have studied direct solvers for the solution of incompressible Navier–Stokes equations. These studies looked at 2-D sudden expansion flows, driven cavity flows and natural convection flows. The Yale sparse matrix package (YSMP)[7] has been used in conjunction with an analytically derived Jacobian on a single computational grid. Venkatakrishnan[8,9] and Bailey and Beam[10] studied direct solvers for the compressible Navier–Stokes equations. These studies used both YSMP and banded LINPACK routines. Venkatakrishnan used a series of grids of increasing refinement to obtain a good initial guess on the final grid, i.e. mesh sequencing. Bailey and Beam evaluated parts of their Jacobian numerically. Knoll[11,12] and Vu[13] used Newton's method to solve the fluid equations of the tokamak edge plasma. All these researchers have discussed the possibilities and benefits of a modified Newton iteration. However, none has implemented this idea in an adaptive fashion.

NEWTON'S METHOD

We are generally interested in the solution of a system of non-linear equations, which can be expressed as

$$\mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})]^{\mathrm{T}} = 0. \tag{1}$$

Implementation of Newton's method requires use of the Jacobian matrix, **J**. The element $J(i,j)$ of the Jacobian matrix is the derivative of $f_i(\mathbf{x})$, with respect to the jth element of the state vector, **x** (i.e. x_j). As a specific example, consider the 2-D natural convection problem. The state vector for this problem is defined here as

$$\mathbf{x} = \{\ldots, u(i,j), v(i,j), P(i,j), T(i,j), u(i+1,j), v(i+1,j), \ldots\}^{\mathrm{T}}. \tag{2}$$

The components of **F**(**x**) are then some discretized form of the u-momentum equation, the v-momentum equation, the continuity equation and the energy equation for each computational cell. Thus, the rows $i = 1, 5, 9, 13, \ldots$ of **F**(**x**) are u-momentum difference equations. The structure of **F**(**x**) is block pentadiagonal. The five block diagonals result from the five point difference stencil used in discretizing these equations. Applying Newton linearization, we must solve the following linearized system at each Newton iteration.

$$\mathbf{J}^n \delta\mathbf{x}^n = -\mathbf{F}(x^n) = -\mathbf{res}^n, \tag{3}$$

$$\mathbf{x}^{n+1} = \delta\mathbf{x}^n + \mathbf{x}^n. \tag{4}$$

This iteration is continued until the norm of $\delta\mathbf{x}$ and/or the norm of **res** are below some tolerance level. Each iteration involves the LU decomposition of **J** and a forward/backward solve.

The advantages of this algorithm are the robustness of the direct solve and the quadratic convergence of Newton's method. The disadvantages of the algorithm are the large memory and CPU time requirements for the LU decomposition, the small radius of convergence of Newton's method and the complicated evaluation of the Jacobian. In many instances there may also be a problem associated with the condition number of the Jacobian matrix. For all cases the system of equations being solved must be non-dimensionalized to reduce the effects of a poorly conditioned matrix. This is also important in the use of our numerical Jacobian and for monitoring convergence.

PERFORMANCE IMPROVEMENT AND SIMPLIFICATION TECHNIQUES

Numerical Jacobian

As the complexity of the physical model increases, so does the level of effort required to form the Jacobian matrix. Most researchers to date have relied on an analytical evaluation of the Jacobian. While this task can be straightforward for simple problems, it becomes increasingly more difficult for more complicated problems such as chemically reacting flow.[14] The most frequent statements made against a numerically evaluated Jacobian are high CPU cost and reduced accuracy. We discuss here an algorithm for numerically evaluating the Jacobian which requires a small fraction of the total CPU time per Newton iteration and is sufficiently accurate to yield superlinear convergence. This algorithm allows code modifications to be performed relatively quickly and easily. The effort required to add a new equation, change an existing equation, change the functional form of transport coefficients or change the differencing scheme is minimized within this algorithm.

The numerical Jacobian involves first-order forward perturbations of statement functions. As an example, we demonstrate the application of this algorithm in solving the one-dimensional, non-linear heat conduction problem. The equation governing one-dimensional heat conduction is

$$\frac{\partial}{\partial x}\left(-K\frac{\partial T}{\partial x}\right)=0. \tag{5}$$

We define the thermal conductivity K as follows:

$$K=c\sqrt{T}. \tag{6}$$

Finite volume differencing applied to equation (5), using the grid shown in Figure 1, results in the following difference equation or statement function

$$f(Tw, Tp, Te)=-K_E\frac{Te-Tp}{\mathrm{d}x}+K_W\frac{Tp-Tw}{\mathrm{d}x}=0, \tag{7}$$

where

$$K_E=c\frac{\sqrt{(Te)}+\sqrt{(Tp)}}{2}, \tag{8}$$

$$K_W=c\frac{\sqrt{(Tw)}+\sqrt{(Tp)}}{2}. \tag{9}$$

For this simple problem, the Jacobian is a tridiagonal matrix whose elements are given by

$$J(i,j)=\begin{cases}\dfrac{\partial f}{\partial Tw}, & j=i-1\\[2ex] \dfrac{\partial f}{\partial T_p}, & j=i,\\[2ex] \dfrac{\partial f}{\partial T_e}, & j=i+1,\\[2ex] 0, & |i-j|>1.\end{cases} \tag{10}$$

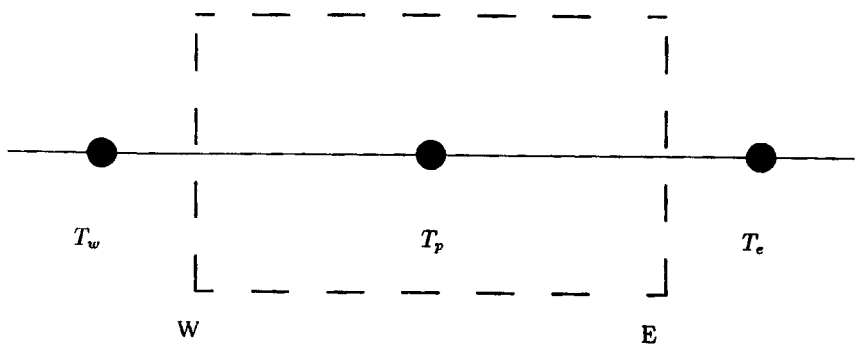

Figure 1. Grid used with the sample one-dimensional heat conduction problem

The partial derivatives in equation (10) are evaluated numerically using finite difference approximations, i.e.

$$\frac{\partial f}{\partial Te} = \frac{f(Tw, Tp, Te+\varepsilon) - f(Tw, Tp, Te)}{\varepsilon}, \tag{11}$$

where ε is a small perturbation. In terms of the above statement function, the elements of the Jacobian matrix can be computed as follows:

Do $i=1, nx$

$$jw(i) = \{f[aa*T(i-1)+bb, T(i), T(i+1)] - f[T(i-1), T(i), T(i+1)]\} / [bb*T(i-1)+bb],$$

$$jc(i) = \{f[T(i-1), aa*T(i)+bb, T(i+1)] - f[T(i-1), T(i), T(i+1)]\} / [bb*T(i)+bb],$$

$$je(i) = \{f[T(i-1), T(i), aa*T(i+1)+bb] - f[T(i-1), T(i), T(i+1)]\} / [bb*T(i+1)+bb],$$

End Do

where $jw(i)$ represents the finite difference approximation to $\partial f/\partial Tw$, $jc(i)$ represents the finite difference approximation to $\partial f/\partial Tc$, and $je(i)$ represents the finite difference approximation to $\partial f/\partial Te$. To reduce function evaluations, the unperturbed function is evaluated and stored outside the above loop. The perturbation parameters aa and bb are defined as

$$bb = \text{sqrt}(\text{roundoff})$$

$$aa = 1 + bb, \quad \text{and} \quad \varepsilon = bb + bb*T(\cdot).$$

Note that these parameters allow the numerical perturbation to vary proportionately with the state variable, yet they require a non-zero perturbation. For instance, if the state variable is zero, the addition of the constant bb term is necessary to ensure that a non-zero perturbation is used to compute this Jacobian element. Some modifications are required for boundary conditions, multidimensions and systems of equations. For the above example in two dimensions, a simple five-point difference stencil yields a Jacobian matrix with a pentadiagonal structure. The single

Do loop becomes a double loop over i and j, and two additional derivative evaluations would be required to account for the dependance on Tn and Ts. There would be a total of six function evaluations. For an extension treating a system of equations consider the 2-D driven cavity problem, which is governed by one continuity equation and two momentum equations. Using a staggered grid and power law differencing, each of the momentum equations is a function of 11 surrounding variables while the continuity equation is a function of only four variables. Thus, calculating the Jacobian elements for each momentum equation in each finite volume requires a total of 12 function evaluations. Calculating the Jacobian elements associated with the continuity equation requires a total of five function evaluations per finite volume. Note that the Jacobian elements for each conservation equation are computed in their own subroutine to enhance code modularity.

Mesh sequencing

Mesh sequencing is used to obtain an improved initial guess on the final grid. This is important since Newton's method only converges if the initial guess lies within the radius of convergence.[15] Mesh sequencing is analogous to the first upward cycle of a Full MultiGrid. (FMG) algorithm.[16] The difference is that we use banded Gaussian elimination to solve the linearized problem on each grid instead of an iterative method. In this work we use bilinear interpolation to move through a series of uniform grids that are each generated from the previous grid by doubling the grid dimension in both directions. We move through this sequence of grids using an interpolated previous grid solution as our initial guess.

There is a well-known combination of direct and multigrid methods where direct methods are used only on the coarsest grids. In this paper we are only concerned with improving the CPU efficiency of the direct method through the use of mesh sequencing.

Adaptive modified Newton iteration

Modified Newton iteration requires that the Jacobian matrix be formed and factored only at the beginning of the iteration, i.e.

$$\mathbf{J}^0 \delta \mathbf{x}^n = -\mathbf{res}^n, \tag{12}$$

Use of this method is warranted when the cost of forming and factoring the Jacobian matrix is a significant part of the total cost of the calculation. In the test problems discussed herein, approximately 90% of the CPU time per iteration is spent forming and factoring the Jacobian matrix. Although use of the modified Newton iteration results in considerable savings in CPU time per iteration, more iterations may be required due to a slower convergence rate. Therefore, the overall computational efficiency should be measured by the total computational time required to obtain a converged solution. The results in the following section clearly demonstrate that for the test problems considered, adaptive modified Newton iteration algorithms are significantly more efficient than full Newton iteration.

The difficulty in using the modified Newton iteration is in determining when a new Jacobian matrix should be formed and factored. Two possible criteria for making this determination are considered:

(1) The first method requires the norm of the residual vector to decrease by a certain fraction on each modified Newton step,[14,17] i.e.

$$\frac{\|\mathbf{F}(\mathbf{x}^{n+1})\|_\infty}{\|\mathbf{F}(\mathbf{x}^n)\|_\infty} \leq \frac{1}{4}, \tag{13}$$

where the vector norm is defined as

$$\|\mathbf{x}\|_\infty = \max_{1\le i\le N} |x_i|. \tag{14}$$

If the inequality in equation (13) is violated, the update is rejected and a new Jacobian matrix is formed and factored. The value of 1/4 is not the only choice for this inequality; its use here was based on the results of Reference 14.

(2) The second method uses an error estimate that sets an upper bound for the size of the modified Newton updates. This error estimate, derived by Smooke,[14] is given by

$$\|\delta\mathbf{x}^n\|_\infty \le hbA_n, \tag{15}$$

where

$$A_1 = 1/2, \tag{16}$$

$$A_n = hA_{n-1}\left(C_{n-1} + \frac{h}{2}A_{n-1}\right), \tag{17}$$

$$C_1 = 1, \tag{18}$$

$$C_n = C_{n-1} + hA_{n-1}. \tag{19}$$

Smooke[14] derived this error estimate by assuming that the hypotheses of the Kantorovich theorem[14,18] were satisfied, namely, the inverse of the Jacobian matrix exists and its norm is bounded by

$$\|\mathbf{J}^{-1}(\mathbf{x}^0)\|_\infty \le a, \tag{20}$$

the norm of the initial Newton update satisfies

$$\|\mathbf{x}^1 - \mathbf{x}^0\|_\infty = \|\mathbf{J}^{-1}(\mathbf{x}^0)\mathbf{F}(\mathbf{x}^0)\|_\infty \le b, \tag{21}$$

and the components of $\mathbf{F}(\mathbf{x})$ have continuous second partial derivatives that satisfy,

$$\sum_{k=1}^{N} \frac{\partial^2 f_i(\mathbf{x})}{\partial x_j \partial x_k} \le \frac{c}{N}, \qquad i,j = 1, 2, \ldots, N, \tag{22}$$

where

$$h = abc \le \tfrac{1}{2}. \tag{23}$$

The matrix norm used above is defined as

$$\|\mathbf{A}\|_\infty = \max_{1\le i\le N} \sum_{j=1}^{N} |a_{ij}|. \tag{24}$$

These hypotheses are sufficient to guarantee convergence of the modified Newton iteration. Thus, equation (15) represents an upper bound for the modified Newton updates in the form of a polynomial in h scaled by the maximum update from the initial Newton iteration (b). This error estimate can be applied recursively for $h = 1/2$ since this value of h maximizes the right-hand side of equation (15). If the norm of the updates exceeds this upper bound, then the Kantorovich[18] hypotheses are not satisfied. Since these conditions are only sufficient for convergence, continued use of the modified Newton iteration may or may not yield a converged solution. If the inequality in equation (15) is violated, we choose to form and factor a new Jacobian matrix and to restart the modified Newton iteration.

The Jacobian matrix is often computed numerically ($\tilde{\mathbf{J}}$) rather than analytically. This numerical approximation results in a perturbation in the analytic Jacobian given by

$$\| \tilde{\mathbf{J}}^{-1}(\mathbf{x}_0)\mathbf{J}(\mathbf{x}_0) - \mathbf{I} \|_\infty \leq \varepsilon, \tag{25}$$

where $\mathbf{I}$ is the identity matrix and ε represents a measure of the numerical perturbation. Smooke derived an error estimate similar to equation (15) in terms of ε. However, ε is often difficult to obtain, and so we follow Smooke's practice of using equation (15) even when the Jacobian matrix is computed numerically.

In summary, application of the adpative modified Newton iteration is performed on each grid, testing against the criteria of the selected method [i.e. using either equation (13) or (15)]. If these criteria are violated then a new Jacobian matrix is formed and factored, and the modified Newton iteration is restarted using $\mathbf{x}^n$ as the initial solution. We have chosen to implement the two criteria separately in order to study their individual advantages and disadvantages. Note that more flexibility could be added by combining the two criteria into a single algorithm. Although not studied here, the algorithm could be made more conservative by requiring satisfaction of both criteria; while requiring the satisfaction of either criteria would presumably lead to fewer Jacobian formulations and factorizations in some cases.

RESULTS

In this section the solution procedure outlined above is applied to the well-known driven cavity and natural convection model problems. These model problems are used to demonstrate the convergence characteristics of the solution algorithm. Steady-state solutions on a 60×60 uniform grid have been obtained using power-law differencing[19] and a standard staggered placement of variables. We have used the LINPACK routines DGBFA and DGBSL for the direct matrix solution. The equations for the model problems are in dimensionless form which helps keep the condition number of the Jacobian matrix within a reasonable range. These results are presented below using $bb = 1{\cdot}0 \times 10^{-8}$ in the evaluation of the numerical Jacobian. The infinity norm of the residual vector and the update vector were required to be below $1{\cdot}0 \times 10^{-6}$ for convergence. The effects of a numerically evaluated Jacobian, mesh sequencing and adaptive modified Newton iteration are discussed.

Driven cavity problem

The equations governing incompressible fluid flow in a two-dimensional driven cavity are the incompressible Navier–Stokes equations. These equations can be written in dimensionless form as

$$\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y} = 0, \tag{26}$$

$$U\frac{\partial U}{\partial X} + V\frac{\partial U}{\partial Y} = -\frac{\partial P}{\partial X} + \frac{1}{Re}\left(\frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2}\right), \tag{27}$$

$$U\frac{\partial V}{\partial X} + V\frac{\partial V}{\partial Y} = -\frac{\partial P}{\partial Y} + \frac{1}{Re}\left(\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2}\right). \tag{28}$$

The boundary conditions for the driven cavity problem can be expressed as

$$\begin{aligned} &U = 1, && Y = 1, \\ &U = V = 0, && \text{all other boundaries.} \end{aligned} \tag{29}$$

Tables I–III show the convergence data for the solution of the driven cavity problem at Reynolds numbers of 500, 1000 and 5000, respectively. Each table shows the results using: (a) full Newton iteration on a single 60×60 grid; (b) full Newton iteration and mesh sequencing (grid dimensions of 15×15, 30×30 and 60×60); (c) the first modified Newton iteration method and mesh sequencing; (d) the second modified Newton iteration method and mesh sequencing. A zero initial guess was used for all variables. The number of iterations required for convergence and the number of times the Jacobian matrix was formed and factored on each grid are listed for each solution technique. The total CPU time required to obtain a converged solution on the 60×60 grid is given in the last column of each table.

Mesh sequencing uses less costly coarse grid solutions to provide an improved initial guess for the fine grid calculation. Comparing full Newton iteration in Table I, with and without mesh

Table I. Solution of driven cavity problem for $Re = 500$

Driven cavity problem using Newton's method Single precision on CRAY-XMP 2/16 $Re = 500$, tol $= 1{\cdot}0 \times 10^{-6}$, $bb = 1{\cdot}0 \times 10^{-8}$				
Method	15×15 Iter–Fac	30×30 Iter–Fac	60×60 Iter–Fac	Total CPU
(a) Full	—	—	8–8	73·8
(b) Full MS	6–6	5–5	4–4	45·7
(c) Modified-1	9–5	7–3	7–3	32
(d) Modified-2	34–3	9–2	13–1	14·7

Table II. Solution of driven cavity problem for $Re = 1000$

Driven cavity problem using Newton's method Single precision on CRAY-XMP 2/16 $Re = 1000$, tol $= 1{\cdot}0 \times 10^{-6}$, $bb = 1{\cdot}0 \times 10^{-8}$				
Method	15×15 Iter–Fac	30×30 Iter–Fac	60×60 Iter–Fac	Total CPU
(a) Full	—	—	—	diverged
(b) Full MS	7–7	5–5	5–5	46·7
(c) Modified-1	10–5	7–3	7–3	33
(d) Modified-2	16–4	20–2	8–2	21·4

Table III. Solution of driven cavity problem for $Re = 5000$

Driven cavity problem using Newton's method Single precision on CRAY-XMP 2/16 $Re = 5000$, tol $= 1{\cdot}0 \times 10^{-6}$, $bb = 1{\cdot}0 \times 10^{-8}$				
Method	15×15 Iter–Fac	30×30 Iter–Fac	60×60 Iter–Fac	Total CPU
(a) Full	—	—	—	diverged
(b) Full MS	8–8	13–13	7–7	69·2
(c) Modified-1	11–7	19–10	11–3	37·1
(d) Modified-2	25–5	31–9	12–3	35·1

sequencing, shows that half as many full Newton factors are required on the finest grid when mesh sequencing is used. The time necessary to converge to a solution on the 60×60 grid was thereby reduced by approximately 50%. Another advantage of mesh sequencing is that it can extend the range of convergence of Newton's method to finer grids by providing an initial guess that lies within the radius of convergence of Newton's method. Tables II and III demonstrate that mesh sequencing does in fact lead to convergence in regions where full Newton iteration without mesh sequencing fails to converge.

The benefits of the modified Newton iteration algorithms are also demonstrated in Tables I–III. In all cases the use of these algorithms resulted in fewer full factorizations of the Jacobian matrix on each grid. Even though more iterations were required to obtain convergence, the overall CPU time was significantly reduced. Compared with full Newton iteration with mesh sequencing, the second modified Newton iteration algorithm reduced the CPU time by at least a factor of two. The use of the modified Newton iteration thereby further increased the overall computational efficiency of the algorithm.

Table I shows that Method (d) significantly out-performed Method (c) for $Re = 500$. However, for higher Reynolds number problems ($Re = 1000$ and 5000) a similar performance was not observed. This inconsistent behaviour can be explained as follows. As the Reynolds number increases, a very thin boundary layer forms along the walls of the cavity. The initial guess supplied from a coarse grid cannot accurately resolve this boundary layer, and so more factorizations are required for convergence. Use of non-uniform grids with more resolution near the cavity walls would presumably eliminate this problem.

Natural convection problem

Using the Boussinesq approximation, the governing equations for the natural convection problem are expressed in dimensionless form as

$$\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y} = 0, \tag{30}$$

$$U\frac{\partial U}{\partial X} + V\frac{\partial U}{\partial Y} = -\frac{\partial P}{\partial X} + \frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2} + Gr\,T\sin\phi, \tag{31}$$

$$U\frac{\partial V}{\partial X} + V\frac{\partial V}{\partial Y} = -\frac{\partial P}{\partial Y} + \frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2} + Gr\,T\cos\phi, \tag{32}$$

$$U\frac{\partial T}{\partial X} + V\frac{\partial T}{\partial Y} = \frac{1}{Pr}\left(\frac{\partial^2 T}{\partial X^2} + \frac{\partial^2 T}{\partial Y^2}\right), \tag{33}$$

where Ra is the Rayleigh number ($Ra = Gr\,Pr$).

The boundary conditions for this problem are given by

$$\begin{aligned} T &= 0, \quad X = 0, \\ T &= 1, \quad X = 1, \\ \frac{\partial T}{\partial Y} &= 0, \quad Y = 0, \\ \frac{\partial T}{\partial Y} &= 0, \quad Y = 1, \\ U &= V = 0, \quad \text{all boundaries.} \end{aligned} \tag{34}$$

A simple damping scheme has been implemented into the basic solution algorithm for the natural convection model problem. This scheme is based on the scalar variable temperature. When damping is employed, the magnitude of the temperature is not allowed to change by more than a specified percentage over the iteration. The update vector is then scaled to satisfy this temperature restriction. This scheme is adaptive in that as the true solution is approached, small temperature variations $\leq 20\%$ require no scaling of the update vector. Damping was implemented, when necessary, for solutions at Rayleigh numbers of 10^6 and 10^7. For all cases the initial temperature field was set to a value of 0·5.

Tables IV–VII show the convergence data for the solution of the natural convection problem at Rayleigh numbers of 10^4, 10^5, 10^6 and 10^7, respectively. The data are presented in the same form as the tables for the driven cavity problem.

Comparing Methods (a) and (b) in each of these tables shows that mesh sequencing reduced the number of factors required on the finest grid by at least a factor of two. Note that the required total CPU time was reduced almost proportionately. Table VII also demonstrates that mesh sequencing extends the range of convergence of the full Newton's method [Method (a)]. Even with damping, the full Newton's method was not able to converge to a solution for $Ra = 10^7$.

Significant reductions in total CPU time were observed with the use of the modified Newton iteration algorithms. Only one full factorization was required on the finest grid when the modified Newton iteration was employed. Compared with Method (a), Table IV shows that the total CPU

Table IV. Solutions of the natural convection problem for $Ra = 10^4$

Natural convection problem using Newton's method
Single precision on CRAY-XMP 2/16
$Ra = 1{\cdot}0 \times 10^4$, tol $= 1{\cdot}0 \times 10^{-6}$, $bb = 1{\cdot}0 \times 10^{-8}$
No damping

Method	15 × 15 Iter–Fac	30 × 30 Iter–Fac	60 × 60 Iter–Fac	Total CPU
(a) Full	—	—	8–8	316·3
(b) Full MS	8–8	4–4	4–4	173·26
(c) Modified-1	9–5	6–1	5–1	47.1
(d) Modified-2	16–4	6–1	5–1	47·0

Table V. Solutions of the natural convection problem for $Ra = 10^5$

Natural convection problem using Newton's method
Single precision on CRAY-XMP 2/16
$Ra = 1{\cdot}0 \times 10^5$, tol $= 1{\cdot}0 \times 10^{-6}$, $bb = 1{\cdot}0 \times 10^{-8}$
No damping

Method	15 × 15 Iter–Fac	30 × 30 Iter–Fac	60 × 60 Iter–Fac	Total CPU
(a) Full	—	—	13–13	468·1
(b) Full MS	12–12	5–5	5–5	199·6
(c) Modified-1	15–10	10–1	6–1	45.9
(d) Modified-2	26–8	10–1	6–1	44·7

time was reduced by a factor of approximately seven for $Ra = 10^4$, while Tables V and VI demonstrate order of magnitude reductions in the total CPU time. Compared with Method (b), the modified Newton iteration reduced the required total CPU time by factors ranging approximately from three to four.

Note that Methods (c) and (d) yield similar results for the natural convection problem. Therefore, for this problem use of Method (c) may be preferred over Method (d) because of the simplicity of the algorithm.

Recall that increased CPU time and reduced accuracy are the most common statements against a numerically computed Jacobian. For these model problems, the fraction of the full iteration CPU time required to evaluate the Jacobian numerically ranged from 0·15 on the coarse grid to 0·05 on the fine grid. Thus, the time required to evaluate the Jacobian matrix numerically is a small fraction of the total time required for a full Newton iteration. In fact, the percentage becomes even less significant as the grid size is refined. We have also witnessed this scaling of the numerical Jacobian in more realistic problems such as our fluid modelling of the boundary layer of a tokamak fusion reactor.[20] Figure 2 plots the L_2 and the L_{inf} norm of the Newton update vector versus the iteration count for a natural convection problem with $Ra = 10^5$ using mesh sequencing and a full Newton iteration. These results are from the 60×60 grid. This figure demonstrates that the numerical Jacobian is sufficiently accurate to yield excellent convergence behaviour. These observations tend to support the use of a numerical Jacobian.

Table VI. Solutions of the natural convection problem for $Ra = 10^6$

Natural convection problem using Newton's method
Single precision on CRAY-XMP 2/16
$Ra = 1{\cdot}0 \times 10^6$, tol $= 1{\cdot}0 \times 10^{-6}$, $bb = 1{\cdot}0 \times 10^{-8}$
Temperature not allowed to change by more than 20% per iteration on first grid

Method	15×15 Iter–Fac	30×30 Iter–Fac	60×60 Iter–Fac	Total CPU
(a) Full	—	—	26–26	930·9
(b) Full MS	18–18	6–6	6–6	243·2
(c) Modified-1	22–13	8–3	13–1	54.9
(d) Modified-2	61–12	66–1	13–1	52·0

Table VII. Solutions of the natural convection problem for $Ra = 10^7$

Natural convection problem using Newton's method
Single precision on CRAY-XMP 2/16
$Ra = 1{\cdot}0 \times 10^7$, tol $= 1{\cdot}0 \times 10^{-7}$, $bb = 1{\cdot}0 \times 10^{-8}$
Temperature not allowed to change by more than 20% per iteration on first grid

Method	15×15 Iter–Fac	30×30 Iter–Fac	60×60 Iter–Fac	Total CPU
(a) Full	—	—	—	Diverged
(b) Full MS	30–30	10–10	9–9	389·7
(c) Modified-1	34–25	13–5	10–3	142.7
(d) Modified-2	75–21	21–4	181–1	105·6

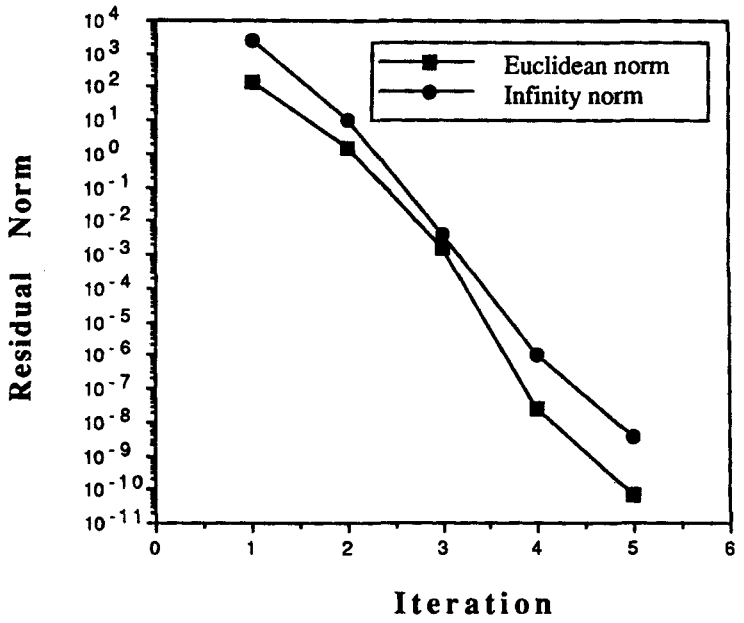

Figure 2. Convergence of Newton's method using a numerical Jacobian

SUMMARY AND CONCLUSIONS

Newton's method was used to solve the two-dimensional driven cavity and natural convection model problems. The efficiency of Newton's method was improved using mesh sequencing and modified Newton iteration. The programing complexity of evaluating the numerical Jacobian matrix was simplified using statement functions. It was shown that the numerical Jacobian was sufficiently accurate to yield a superlinear convergence rate.

Mesh sequencing extends the range of convergence of Newton's method by using coarse grid solutions to supply an initial guess that is within the radius of convergence of the fine grid. Additionally, with an improved initial guess Newton's method requires fewer iterations to converge. Most of the work is thereby shifted to the coarse grids where the cost per iteration is low relative to the fine grid.

Adaptive modified Newton iteration algorithms resulted in fewer full factorizations of the Jacobian matrix. Although more iterations were often required for convergence, the overall CPU time was reduced. Thus, the overall efficiency of the algorithm was increased. In conjunction with mesh sequencing, adaptive modified Newton iteration, in some cases, resulted in order of magnitude reductions in the total CPU time compared with the full Newton method on a single grid.

Elements of the Jacobian matrix were approximated by finite differences, which were computed using simple forward perturbations of statement functions. Statement functions considerably reduced the programming complexity of forming the Jacobian matrix and also simplified code modifications. It was shown that the cost of forming the numerical Jacobian was negligible in comparison to the cost of a full Newton iteration. Excellent convergence of Newton's method with a numerically computed Jacobian was also demonstrated.

The results show that mesh sequencing and adaptive modified Newton iteration significantly improve the robustness and efficiency of Newton's method as a fully implicit direct solver for the two-dimensional Navier–Stokes equations. Implementation of Newton's method may be simplified by using statement functions to minimize the programming complexity of evaluating the Jacobian matrix numerically.

REFERENCES

1. O. C. Zienkiewicz, *The Finite Element Method*, McGraw-Hill, London, 1977.
2. S. P. Vanka and G. K. Leaf, 'Fully-coupled solution of pressure-linked fluid-flow equations', *Technical Report ANL-83-73*, Argonne National Laboratory, 1983.
3. J. W. MacArthur, 'Development and implementation of robust direct finite-difference methods for the solution of strongly coupled elliptic transport equations', *Ph.D. Thesis*, University of Minnesota, 1986.
4. J. W. MacArthur and S. V. Patankar, 'Robust semidirect finite difference methods for solving the Navier–Stokes and energy equations', *Int. j. numer. methods fluids*, **9**, 325–340 (1989).
5. S. P. Vanka, 'Block-implicit calculation of steady turbulent recirculating flows', *Int. J. Heat Mass Transfer*, **28**, 2093–2103 (1985).
6. M. E. Braaten, 'Development and evaluation of iterative and direct methods for the solution of the equations governing recirculating flows', *Ph.D. Thesis*, University of Minnesota, 1985.
7. S. G. Eisenstat *et al.*, 'Yale sparse matrix package: the nonsymmetric codes', *Technical Report 114*, Department of Computer Science, Yale University, 1978.
8. V. Venkatakrishnan, 'Newton solution of invisid and viscous problems', *AIAA J.*, **27** (1989).
9. V. Venkatakrishnan, 'Viscous computations using a direct solver', *Comput. Fluids*, **18** (1990).
10. H. E. Bailey and R. M. Beam, 'Newton's method applied to finite-difference approximations for the steady-state compressible Navier–Stokes equations', *J. Comput. Phys.*, **93**, 108–127 (1991).
11. D. A. Knoll, 'Development and application of a direct Newton solver for the two-dimensional tokamak edge plasma fluid equations', *Ph.D. Thesis*, University of New Mexico, 1991.
12. D. A. Knoll, A. K. Prinja and R. B. Campbell, 'A direct Newton solver for the two-dimensional tokamak edge plasma fluid equations', *J. Comput. Phys.*, 1992, to appear.
13. H. X. Vu, 'Plasma collection by an obstacle', *Ph.D. Thesis*, California Institute of Technology, 1990.
14. M. D. Smooke, 'Solution of burner-stabilized premixed laminar flames by boundary value methods', *J. Comput. Phys.*, **48**, 72–105 (1982).
15. James S. Vandergraft, *Introduction to Numerical Computations*, Academic Press, New York, 1983.
16. A. Brandt, 'Multigrid techniques: 1984 guide with applications to fluid dynamics', *Technical report*, von Karman Institue, 1984.
17. U. Ascher, J. Christiansen and R. D. Russell, 'COLSYS: a collocation code for boundary-value problems', in B. Childs *et al.* (ed), *Codes for Boundary Value Problems in ODE's*, Springer, New York, 1979.
18. L. V. Kantorovich and G. P. Akilov, *Functional Analysis In Normal Spaces*, Pergamon, Oxford, 1964.
19. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere, New York, 1980.
20. D. A. Knoll and P. R. McHugh, 'NEWEDGE: a 2-D fully implicit edge plasma fluid code for advanced physics and complex geometries', *J. Nuc. Mat.*, (1992), to appear.