INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 14, 817–841 (1992)

A FINITE ELEMENT SOLUTION OF THREE-DIMENSIONAL MIXED CONVECTION GAS FLOWS IN HORIZONTAL CHANNELS USING PRECONDITIONED ITERATIVE MATRIX METHODS

ERIK O. EINSET*

Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455, U.S.A.

AND

KLAVS F. JENSEN

Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

SUMMARY

An algorithm is presented for the finite element solution of three-dimensional mixed convection gas flows in channels heated from below. The algorithm uses Newton's method and iterative matrix methods. Two iterative solution algorithms, conjugate gradient squared (CGS) and generalized minimal residual (GMRES), are used in conjunction with a preconditioning technique that is simple to implement. The preconditioner is a subset of the full Jacobian matrix centred around the main diagonal but retaining the most fundamental axial coupling of the residual equations. A domain-renumbering scheme that enhances the overall algorithm performance is proposed on the basis of an analysis of the preconditioner. Comparison with the frontal elimination method demonstrates that the iterative method will be faster when the front width exceeds approximately 500. Techniques for the direct assembly of the problem into a compressed sparse row storage format are demonstrated. Elimination of fixed boundary conditions is shown to decrease the size of the matrix problem by up to 30%. Finally, fluid flow solutions obtained with the numerical technique are presented. These solutions reveal complex three-dimensional mixed convection fluid flow phenomena at low Reynolds numbers, including the reversal of the direction of longitudinal rolls in the presence of a strong recirculation in the entrance region of the channel.

KEY WORDS Mixed convection Finite elements CVD Iterative methods Preconditioning

INTRODUCTION

Three-dimensional mixed convection flows arise in pipes, channels and ducts when a forced flow is superimposed on a free convection, buoyant flow driven by temperature gradients in the enclosure. Such flows arise in heat transfer systems when convection enhances cooling over pure conduction[1] and in chemical reactors when mixed convection causes non-uniform growth rates in thin film processes such as in chemical vapour depostion (CVD) systems.[2] Calculating the flow structures that arise is a difficult numerical problem in both complexity and size, since fine meshes are required to capture small-scale flow structures in the enclosure.

* Present address: General Electric Company, Corporate Research and Development, PO Box 8, Schenectady, NY 12301, U.S.A.

0271–2091/92/070817–25$12.50

Received 6 July 1990
Revised 25 June 1991

For conditions that give rise to three-dimensional flows, namely supercritical Rayleigh numbers, marching methods based on a parabolic approximation to the governing equations are a useful technique when the forced flow is sufficiently strong that diffusion of momentum and energy in the axial direction can be neglected.[2-4] The parabolic approximation, however, cannot predict recirculatory flows which may occur at low Reynolds numbers. Furthermore, it is unclear when the parabolic approximation breaks down, since solutions may be computed for arbitrarily low flow rates even though in reality, back flows may exist. In the light of these difficulties a fully three-dimensional solution over the whole domain is required. The resulting computational requirement can be large.

In the present work we present an approach for the solution of low Reynolds number mixed convection flows of gases in channels heated from below. The solution strategy is based on the finite element method (FEM)[5] implemented in Newton's method using sparse storage and iterative matrix methods. We emphasize the development of a simple preconditioner for fluid flow in channels. This study is motivated by the need to understand transport phenomena in horizontal CVD reactors where a gas containing dilute concentrations of film precursors flows over a heated substrate where chemical reactions and film deposition occur.[2]

MODEL EQUATIONS

The geometry is shown in Figure 1. Isothermal gas at room temperature flows into a horizontal enclosure which is heated from below. The flow in this geometry is governed by the Cauchy momentum equations, the energy equation and the continuity equation. Several approximations may be made for CVD reactor flows.[2] The gas is considered ideal, so that the density variation with temperature T is given by

$$\rho=\left(\frac{PM}{RT_0}\right)\frac{T_0}{T}=\rho_0\frac{T_0}{T}, \tag{1}$$

where M is the molecular weight of the gas and R is the universal gas constant. Since pressure variations in the reactor are small, the pressure P is taken to be constant at the reactor operating pressure. This implies that the density can be expressed in term of its value ρ_0 at the inlet temperature T_0. In this and the following equations the subscript '0' indicate values evaluated at the inlet.

Typical low Reynolds number flows have Mach numbers much less than unity, so compressibility effects may be neglected except for the thermal expansion of the gas. The fluid is Newtonian and the physical properties of viscosity μ, thermal conductivity k and heat capacity C_P are only functions of temperature and are given in Table I. Energy contributions from pressure changes, viscous dissipation and Dufour effects are small and may be neglected.[6]

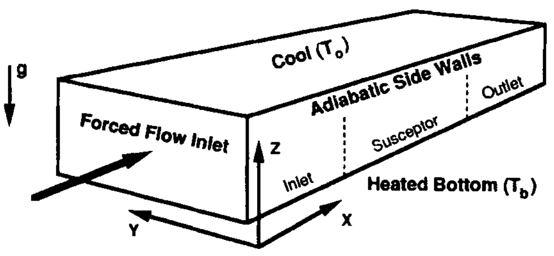


Figure 1. Geometry and co-ordinate system of channel

Table I. Expressions for physical parameters of H_2 and N_2 (from Reference 7)

Property	Hydrogen	Nitrogen
Viscosity μ (g cm s^{-1})	$2{\cdot}224\times10^{-6}\ T^{0{\cdot}648}$	$4{\cdot}1772\times10^{-6}\ T^{0{\cdot}6584}$
Thermal conductivity k (cal cm^{-1} s^{-1} K^{-1})	$8{\cdot}0149\times10^{-6}\ T^{0{\cdot}69086}$	$8{\cdot}804\times10^{-7}\ T^{0{\cdot}7515}$
Heat Capacity $C_P\times M$ (cal mol^{-1} K^{-1})	$6{\cdot}9417-1{\cdot}742\times10^{-4}\ T$ $+4{\cdot}572\times10^{-7}\ T^2$	$6{\cdot}9036-3{\cdot}748\times10^{-4}\ T$ $+1{\cdot}9306\times10^{-6}\ T^2$ $-6{\cdot}86\times10^{-10}\ T^3$

The assumptions outlined above lead to the following set of governing equations:

$$\rho_0\frac{T_0}{T}\mathbf{v}\cdot\nabla\mathbf{v}=-\nabla p+\rho_0\frac{T_0-T}{T}\mathbf{g}+\nabla\cdot\mu\,[\nabla\mathbf{v}+(\nabla\mathbf{v})^{\mathrm{T}}-\tfrac{2}{3}\mathbf{I}\nabla\cdot\mathbf{v}], \tag{2}$$

$$\rho_0\frac{T_0}{T}C_P\mathbf{v}\cdot\nabla T=\nabla\cdot(k\nabla T), \tag{3}$$

$$\nabla\cdot\left(\rho_0\frac{T_0}{T}\mathbf{v}\right)=0. \tag{4}$$

In these equations the hydrostatic contribution to the pressure has been subtracted; the pressure field P is defined above a hydrostatic pressure datum. $\mathbf{v}$ is the velocity and $\mathbf{g}$ is the gravitational acceleration.

There are two fundamental ways to non-dimensionalize the problem. The scaling factor for velocity, v_0, can be chosen as in forced convection, $v_0=v_{\max}$, or as in free convection, $v_0=\mu_0/\rho_0 h$, where h is the characteristic length, the height of the channel. The resulting dimensionless equations contain the Grashof number $Gr=gh^3\Delta T\rho_0^2/\mu_0^2 T_0$ and, depending on the scaling factor, may include the Reynolds number $Re=\rho_0 v_0 h/\mu_0$. In these definitions ΔT represents the temperature difference between the heated susceptor (T_b) and the top wall of the channel (T_0). In both non-dimensionalizations the dimensionless temperature $\Theta=(T-T_0)/\Delta T$ arises explicitly along with a dimensionless temperature scaling factor $\Delta T/T_0$. For the channel geometry with no imposed transverse flow it is appropriate to scale the axial velocity with the maximum inlet velocity and the transverse velocity with the viscous velocity. The non-dimensional equations then take the following form:

x-momentum,

$$\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\left(Re\,u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-Re\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left[\eta\left(\frac{4}{3}\frac{\partial u}{\partial x}-\frac{2}{3}\frac{1}{Re}\frac{\partial v}{\partial y}-\frac{2}{3}\frac{1}{Re}\frac{\partial w}{\partial z}\right)\right]$$
$$+\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial u}{\partial y}+\frac{1}{Re}\frac{\partial v}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[\eta\left(\frac{1}{Re}\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)\right]; \tag{5}$$

y-momentum,

$$\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\left(u\frac{\partial v}{\partial x}+\frac{1}{Re}v\frac{\partial v}{\partial y}+\frac{1}{Re}w\frac{\partial v}{\partial z}\right)=-Re\frac{\partial p}{\partial y}+\frac{\partial}{\partial y}\left[\eta\left(\frac{4}{3}\frac{1}{Re}\frac{\partial v}{\partial y}-\frac{2}{3}\frac{\partial u}{\partial x}-\frac{2}{3}\frac{1}{Re}\frac{\partial w}{\partial z}\right)\right]$$
$$+\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial u}{\partial y}+\frac{1}{Re}\frac{\partial v}{\partial x}\right)\right]$$
$$+\frac{\partial}{\partial z}\left[\eta\left(\frac{1}{Re}\frac{\partial w}{\partial y}+\frac{1}{Re}\frac{\partial v}{\partial z}\right)\right]; \qquad (6)$$

z-momentum,

$$\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\left(u\frac{\partial w}{\partial x}+\frac{1}{Re}v\frac{\partial w}{\partial y}+\frac{1}{Re}w\frac{\partial w}{\partial z}\right)=-Re\frac{\partial p}{\partial z}+\frac{\partial}{\partial z}\left[\eta\left(\frac{4}{3}\frac{1}{Re}\frac{\partial w}{\partial z}-\frac{2}{3}\frac{\partial u}{\partial x}-\frac{2}{3}\frac{1}{Re}\frac{\partial v}{\partial y}\right)\right]$$
$$+\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial u}{\partial z}+\frac{1}{Re}\frac{\partial w}{\partial x}\right)\right]$$
$$+\frac{\partial}{\partial y}\left[\eta\left(\frac{1}{Re}\frac{\partial w}{\partial y}+\frac{1}{Re}\frac{\partial v}{\partial z}\right)\right]$$
$$+\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\frac{Gr}{Re}\Theta; \qquad (7)$$

energy,

$$\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\xi\left(Re\,u\frac{\partial\Theta}{\partial x}+v\frac{\partial\Theta}{\partial y}+w\frac{\partial\Theta}{\partial z}\right)=\frac{1}{Pr}\left[\frac{\partial}{\partial x}\left(\gamma\frac{\partial\Theta}{\partial x}\right)+\frac{\partial}{\partial y}\left(\gamma\frac{\partial\Theta}{\partial y}\right)+\frac{\partial}{\partial z}\left(\gamma\frac{\partial\Theta}{\partial z}\right)\right]; \qquad (8)$$

continuity,

$$\left(Re\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)-\frac{\Delta T}{T_0}\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\left(Re\,u\frac{\partial\Theta}{\partial x}+v\frac{\partial\Theta}{\partial y}+w\frac{\partial\Theta}{\partial z}\right)=0. \qquad (9)$$

In the above equations η is the dimensionless viscosity, γ is the dimensionless thermal conductivity and ξ is the dimensionless heat capacity. The scaling leads to the appearance of the Prandtl number $Pr=\mu_0 C_{P0}/k_0$ and the Peclet number $Pe=Pr\,Re$ in the energy equations. For most gases Pr is approximately constant at 0·7. Taking advantage of this fact and using the Rayleigh number $Ra=Gr\,Pr$ instead of Gr as the dimensionless temperature group, the problem is well described by three dimensionless numbers: Re, Ra and $\Delta T/T_0$. For conditions of high Re the equations reduce to the parabolic flow description.[3,4]

For situations where ΔT is very small, temperature variations over the domain are small and the physical properties given in Table I are nearly constant, independent of T. Furthermore, as $\Delta T/T_0 \to 0$, the forced and free convection equations reduce to the equations that would arise under the Boussinesq approximation. Thus the dimensionless group $\Delta T/T_0$ may be considered a measure of the deviation from the behaviour of a Boussinesq fluid.

The inlet velocity profile is fixed as fully developed, isothermal Poiseuille flow.[8] Fluid velocities are set equal to zero on solid walls, fluid total normal stresses are set equal to zero at the outlet and tangential stresses are set equal to zero at symmetry planes. To avoid deleterious effects of the outlet boundary conditions, a long exhaust region is used to insure that outlet effects do not influence the flow in the entrance region of the duct. Fluid flow across a plane of symmetry is disallowed. The top wall is fixed at the inlet temperature T_0, while the heated substrate holder (the susceptor) in the central region of the bottom is fixed at a high temperature T_b. The side walls and the channel floor downstream of the susceptor are considered insulting. The temperature in the channel floor upstream of the susceptor is fixed with a temperature profile modelled as a thermal 'cooling fin',[6] with the dimensionless heat transfer coefficient $Q^2 = HL/k_w B$ being a function of the wall thickness B, the thermal conductivity of the wall material, k_w, the total length of the entrance region, L, and an overall heat transfer coefficient H for natural and convection heat transfer from the wall. In the expression below ζ is the dimensionless distance from the leading edge of the susceptor back towards the inlet. In the boundary conditions summarized in equations (10), $\mathbf{n}$ is the unit normal to the indicated boundary surface:.

$$\begin{gathered}
\mathbf{v}_{\text{solid walls}} = \mathbf{O}, \qquad \mathbf{v}_{\text{inlet}} = \text{fully developed Poiseuille flow,}^{8} \\
\mathbf{n}\cdot\mu[\nabla\mathbf{v}+(\nabla\mathbf{v})^{\mathrm{T}} - \tfrac{2}{3}\mathbf{I}\nabla\cdot\mathbf{v} - p\mathbf{I}]_{\text{outlet}} = 0, \\
T_{\text{inlet}} = T_{\text{top wall}} = T_0, \qquad T_{\text{susceptor}} = T_b, \\
T_{\text{bottom wall of inlet region}} = T_0 + \left(\frac{\cosh[Q(1-\zeta)]}{\cosh Q}\right)\Delta T, \\
\mathbf{n}\cdot\nabla T_{\text{side walls}} = \mathbf{n}\cdot\nabla T_{\text{outlet}} = \mathbf{n}\cdot\nabla T_{\text{bottom wall of exhaust}} = 0.
\end{gathered} \tag{10}$$

FINITE ELEMENT FORMULATION AND SOLUTION ALGORITHM

The primitive variable formulation is used with the now dimensionless variables $\mathbf{v}$, Θ and p expanded in basis functions Φ and Ψ as shown in equations (11). M_q and M_l are respectively the total number of quadratic and linear basis functions in the domain.

$$\mathbf{v} = \sum_{i=1}^{M_q} \mathbf{v}_i \Phi_i, \qquad \Theta = \sum_{i=1}^{M_q} \Theta_i \Phi_i, \qquad p = \sum_{i=1}^{M_l} p_i \Psi_i. \tag{11}$$

Galerkin's method is applied by forming the inner product of the governing equations with the basis functions over the domain (Ω and $\partial\Omega$). Gauss' theorem is applied to handle the boundary conditions resulting in residual equations for momentum (R_x^i, R_y^i, R_z^i), energy (R_E^i) and continuity (R_c^i) as shown in equations (12)–(16). The problem is solved when the set of residual equations is sufficiently close to zero ($\mathbf{R}(u, v, w, \Theta, p) \approx \mathbf{0}$). Further details of the formulation of FEM problems can be found in References 5 and 9.

$$\begin{aligned}
R_x^i = \int_\Omega & \left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1} \left(Re\, u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right)\Phi^i \\
& + \frac{\partial \Phi^i}{\partial x}\left[-Re\, p + \eta\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{1}{Re}\frac{\partial v}{\partial y} - \frac{2}{3}\frac{1}{Re}\frac{\partial w}{\partial z}\right)\right] + \frac{\partial \Phi^i}{\partial y}\left[\eta\left(\frac{\partial u}{\partial y} + \frac{1}{Re}\frac{\partial v}{\partial x}\right)\right] \\
& + \frac{\partial \Phi^i}{\partial z}\left[\eta\left(\frac{1}{Re}\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right]\partial\Omega,
\end{aligned} \tag{12}$$

$$R_y^i=\int_\Omega\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\left(u\frac{\partial v}{\partial x}+\frac{1}{Re}v\frac{\partial v}{\partial y}+\frac{1}{Re}w\frac{\partial v}{\partial z}\right)\Phi^i$$
$$+\frac{\partial\Phi^i}{\partial y}\left[-Rep+\eta\left(\frac{4}{3}\frac{1}{Re}\frac{\partial v}{\partial y}-\frac{2}{3}\frac{\partial u}{\partial x}-\frac{2}{3}\frac{1}{Re}\frac{\partial w}{\partial z}\right)\right]+\frac{\partial\Phi^i}{\partial x}\left[\eta\left(\frac{\partial u}{\partial y}+\frac{1}{Re}\frac{\partial v}{\partial x}\right)\right]$$
$$+\frac{\partial\Phi^i}{\partial z}\left[\eta\left(\frac{1}{Re}\frac{\partial w}{\partial y}+\frac{1}{Re}\frac{\partial v}{\partial z}\right)\right]\partial\Omega, \tag{13}$$

$$R_z^i=\int_\Omega\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\left(u\frac{\partial w}{\partial x}+\frac{1}{Re}v\frac{\partial w}{\partial y}+\frac{1}{Re}w\frac{\partial w}{\partial z}-\frac{Gr}{Re}\Theta\right)\Phi^i$$
$$+\frac{\partial\Phi^i}{\partial z}\left[-Rep+\eta\left(\frac{4}{3}\frac{1}{Re}\frac{\partial w}{\partial z}-\frac{2}{3}\frac{\partial u}{\partial x}-\frac{2}{3}\frac{1}{Re}\frac{\partial v}{\partial y}\right)\right]+\frac{\partial\Phi^i}{\partial x}\left[\eta\left(\frac{\partial u}{\partial z}+\frac{1}{Re}\frac{\partial w}{\partial x}\right)\right]$$
$$+\frac{\partial\Phi^i}{\partial y}\left[\eta\left(\frac{1}{Re}\frac{\partial w}{\partial y}+\frac{1}{Re}\frac{\partial v}{\partial z}\right)\right]\partial\Omega, \tag{14}$$

$$R_E^i=\int_\Omega\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\xi\left(Re\,u\frac{\partial\Theta}{\partial x}+v\frac{\partial\Theta}{\partial y}+w\frac{\partial\Theta}{\partial z}\right)\Phi^i$$
$$+\frac{\gamma}{Pr}\left(\frac{\partial\Phi^i}{\partial x}\frac{\partial\Theta}{\partial x}+\frac{\partial\Phi^i}{\partial y}\frac{\partial\Theta}{\partial y}+\frac{\partial\Phi^i}{\partial z}\frac{\partial\Theta}{\partial z}\right)d\Omega, \tag{15}$$

$$R_c^i=\int_\Omega\left(Re\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)-\frac{\Delta T}{T_0}\left(1+\Theta\frac{\Delta T}{T_0}\right)^{-1}\left(Re\,u\frac{\partial\Theta}{\partial x}+v\frac{\partial\Theta}{\partial y}+w\frac{\partial\Theta}{\partial z}\right)\Psi^i\,d\Omega. \tag{16}$$

The basis functions Θ_i and Ψ_i are chosen in the finite element method to be non-zero over a localized region of the domain. The domain is tessellated into finite elements which support the basis functions. Φ is chosen to be a quadratic function so that the three velocity components and the temperature are expanded in triquadratic basis functions on 27-node brick elements. Ψ is taken to be a linear function so that pressure is expanded in trilinear basis functions on eight-node brick elements. The 27- and eight-node elements are superimposed on each other in the domain discretization. This use of mixed-order basis functions, known as mixed interpolation, is standard in the primitive variable formulation of the Navier–Stokes equations.[10] The calculation of the volume integrals required in the residuals is done by 27-point ($3\times3\times3$) Gauss quadrature.

The zeros of equations (12)–(16) are found by Newton iteration. In Newton's method an initial guess to the solution is modified by a correction calculated by

$$\mathbf{J}(\mathbf{u}_{k+1}-\mathbf{u}_k)=\mathbf{J}\Delta\mathbf{u}=\mathbf{LU}\Delta\mathbf{u}=-\mathbf{R}, \tag{17}$$

where the new guess $\mathbf{u}_{k+1}$ varies from the old guess $\mathbf{u}_k$ by a contribution determined by the residual vector $\mathbf{R}$ and the Jacobian matrix $\mathbf{J}$, where $J_{ij}=\partial R_i/\partial u_j$. Equation (17) can be solved by LU decomposition and back substitution. The use of LU factorization of the Jacobian has advantages in the continuation scheme discussed below.

At each Newton step the Jacobian matrix must be calculated and assembled and a linear system of the form $\mathbf{Ax}=\mathbf{b}$ solved. A modified Newton[11] method is implemented by calculating $\mathbf{J}$ only at the first step and using the same Jacobian in subsequent iterations. The present implementation uses a variant of this modified scheme where the Jacobian is evaluated at every second step until the solution is sufficiently close to convergence, at which point the Jacobian is no longer updated. This strategy represents a significant saving in computational time.

Because of the non-linear nature of the fluid flow problem, continuation techniques are needed to obtain a good initial guess for the Newton iteration and to map out the solution behaviour in the Re–Ra–$\Delta T/T_0$ parameter space. First-order continuation[12] is used. The next initial guess is determined by

$$\mathbf{u}_{\text{new}} = \mathbf{u}_{\text{old}} + (\partial \mathbf{u}/\partial \alpha)_{\text{old}} \Delta \alpha, \tag{18}$$

where α is the continuation parameter in question (e.g. Re). The vector $\partial \mathbf{u}/\partial \alpha$ is calculated from

$$(\partial \mathbf{R}/\partial \mathbf{u})\ \partial \mathbf{u}/\partial \alpha = \mathbf{J} \partial \mathbf{u}/\partial \alpha = -\partial \mathbf{R}/\partial \alpha. \tag{19}$$

Since $\mathbf{J}$ is known from the solution of the matrix problem (equation (17)), it is only necessary to evaluate $-\partial \mathbf{R}/\partial \alpha$ and solve the linear system. This can be done by a back substitution in order N operations (N is the order of the system) if using the previously calculated LU factors, or iteratively as will be described later.

NUMERICAL PROCEDURES

Traditionally, the solution of non-linear boundary value problems by discretization methods is dominated by two processes: the calculation and assembly of the Jacobian, which might include numerical quadrature, and the solution of the matrix problem $\mathbf{J} \Delta \mathbf{u} = -\mathbf{R}$. In the frontal method of Irons[13] and Hood[14] these two processes are done simultaneously and the merit of frontal methods is a low memory overhead. Once an unknown is fully assembled, it is eliminated from the matrix problem. In this way, only a limited number of matrix entries must be stored in core memory. Only the elements contributing to the active 'frontal matrix' are manipulated and partial and full pivoting in the frontal matrix is possible. The pivoting characteristics of the frontal method makes it particularly well suited to the primitive variable formulation of the Navier–Stokes equations. Pivoting is required in that case to treat the zero on the diagonal of the Jacobian arising, since the pressure unknown does not appear explicity in the continuity equation.

In non-linear problems where several modified Newton iterations are required, it is advantageous to keep the full **LU** decomposition of **J** in core for quick back substitutions using updated residuals. In cases where solid state disk space is unavailable, the I/O overhead can be seriously detrimental to the overall solution efficiency. This offsets the memory advantage of frontal methods. There is, however, also an operation count advantage of the frontal method over conventional band solvers. This advantage arises because the front width in the present formulation is roughly one-half the total band width. Here we demonstrate how a preconditioned iterative method is more efficient for large front widths despite the inherent advantages of the frontal elimination method.

Matrix structure

The compressed sparse row format[15] was chosen to store the Jacobian matrix. It is a compact storage scheme for non-symmetric matrices that stores only the non-zero entries of J utilizing 2*MSIZE + N + 1 storage locations, where MSIZE is the number of non-zeros in J and N is the order of the system. The Jacobian entries are stored in an array A (MSIZE). The column number corresponding to $A(i)$ is contained in location $JA(i)$, where JA is an integer array of length MSIZE. The first entry in row k of the matrix is stored at $A(IA(k))$. In other words, row k contains $IA(k+1)-IA(k)$ entries in the locations $A(IA(k))$ to $A(IA(k+1)-1)$. The corresponding columns are stored in $JA(IA(k))$ to $JA(IA(k+1)-1)$.

Reduction of system order

The order of the matrix system was reduced by applying the Dirichlet boundary conditions directly at the element level. When this is done, equations corresponding to fixed degrees of freedom or Dirichlet boundary conditions are eliminated from the matrix system and the order of the system is reduced. By reducing the system order, large FEM problems are solved more efficiently. Examples of the order reduction are presented later.

Preconditioner

In any iterative method for the solution of $\mathbf{Ax}=\mathbf{b}$, a preconditioner $\mathbf{P}$ is required to make the solution method competitive with direct solution methods.[17,18] This is done by finding a preconditioning matrix $\mathbf{P}$ such that

(1) $\mathbf{P}$ is an approximation to $\mathbf{A}$
(2) $\mathbf{P}$ is easily factored so that $\mathbf{Py}=\mathbf{u}$ is quickly solved for many right-hand sides
(3) $\mathbf{P}^{-1}\mathbf{A}$ is better conditioned than the original matrix $\mathbf{A}$.

In this problem the preconditioner not only needs to be a good approximation to $\mathbf{J}$, but pivoting is also needed in the factorization to properly treat the zero on the diagonal of the continuity equation. Because factorization generally fils in the band of a banded matrix,[19] a banded storage format is chosen and a band around the main diagonal of the Jacobian is used as the preconditioner. To enhance the performance of the preconditioner, the domain is renumbered. Further details of the preconditioners used and discussion of the renumbering are presented below.

The preconditioner was assembled in the LINPACK[20] banded matrix storage format simultaneously with the assembly of $\mathbf{J}$. LINPACK is a standard mathematical library using the basic linear algebra subroutines (BLAS). This allowed for the utilization of existing routines for the LU factorization of $\mathbf{P}$ and re-solution of the problem $\mathbf{Py}=\mathbf{u}$, two operations required in preconditioned iterative matrix solution algorithms. The LINPACK factorization routine SGBFA also supports partial pivoting, which is required for the present problem.

Solution algorithm

The solution methods tested are applicable to non-symmetric systems. The first, the conjugate gradient squared (CGS) algorithm of Sonneveld,[21] is a descent-type algorithm closely related to standard conjugate gradient methods. The second, GMRES(k) by Saad and Schultz,[22] is a *restarted* minimal residual algorithm[23,24] which constructs a Krylov subspace in which to minimize some measure of the residual vector. GMRES is in fact a more general version of standard descent-type iterative methods which includes CGS and other conjugate gradient methods. A major difference between the two methods is that GMRES(k) requires storage of the basis vectors of the Krylov subspace, which makes it potentially more storage-intensive than CGS.

For the CGS algorithm a left preconditioning was used. In the notation of Reference 21, $P_L^{-1}=P$ and $P_R=I$. The algorithm is given below:[21]

```
$r_0=P^{-1}(Ax_0-b)$, $\hat{r}_0=r_0$
$q_0=p_{-1}=0$
$\rho_{-1}=1$, $n=0$.
Do while residual norm > tolerance
    $\rho_n=\hat{r}_0^T r_n$, $\beta_n=\rho_n/\rho_{n-1}$
    $u_n=r_n+\beta_n q_n$
    $p_n=u_n+\beta_n(q_n+\beta_n p_{n-1})$
    $v_n=P^{-1}(Ap_n)$
    $\sigma_n=\hat{r}_0^T v_n$, $\alpha_n=\rho_n/\sigma_n$
    $q_{n+1}=u_n-\alpha_n v_n$
    $v_n=\alpha_n(u_n+q_{n+1})$
    $x_{n+1}=x_n-v_n$
    $r_{n+1}=r_n-P^{-1}(Av_n)$
    $n=n+1$.
Continue
```

In the above algorithm, expressions of the type $x=P^{-1}y$ indicate solution of the system $Px=y$ by back substituion with the LU-factored preconditioning matrix P.

The GMRES(k) algorithm is essentially Algorithm 4 from Reference 22 with preconditioning P added directly into the iterative method. The algorithm reads as follows:[22]

```
0. For the original problem $Ax=f$, calculate $b=P^{-1}f$
     new problem is now $P^{-1}Ax=b$.
1. Start—choose $x_0$ and compute $r_0=b-P^{-1}Ax_0$, $v_1=r_0/\|r_0\|$.
2. Iterate—do $j=1, 2, \ldots, k$
     $h_{i,j}=(P^{-1}Av_j, v_i)$, $i=1, 2, \ldots, j$
     $\hat{v}_{j+1}=P^{-1}Av_j-\sum_{i=1,j}(h_{i,j}v_i)$
     $h_{j+1,j}=\|\hat{v}_{j+1}\|$
     $v_{j+1}=\hat{v}_{j+1}/h_{j+1,j}$.
3. From approximate solution $x_k=x_0+V_k y_k$ where $y_k$ minimizes $\|\beta e_1-H_k y\|$, $y\in\mathbb{R}^k$
4. Restart—compute $r_k=b-P^{-1}Ax_k$
     if converged then stop
     else compute $x_0=x_k, v_1=r_k/\|r_k\|$, go to 2.
```

General solution outline

The solution algorithm is shown in Figure 2, with a detailed algorithm of the assembly and solution shown in Figure 3. On the first call to the solver the structure of the sparse matrix is determined. In finding the reduced-order system, two new arrays are initialized. One points to the location in the global matrix where a given entry of the element stiffness matrix for a given element belongs, while the second records the amount of shift needed to get to the reduced

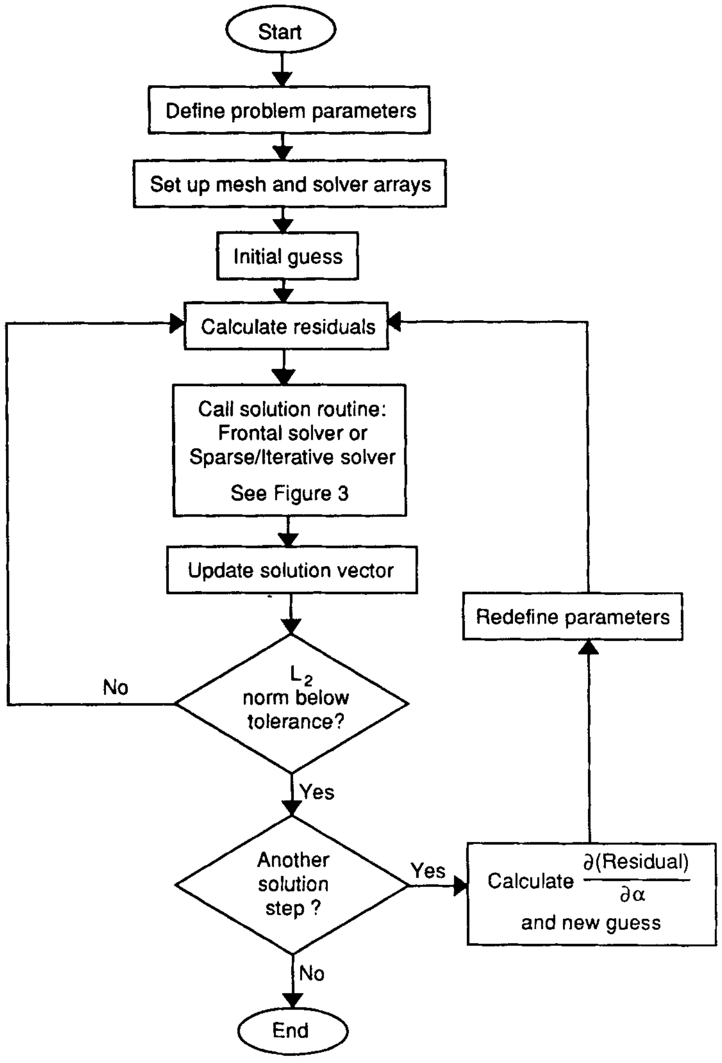

Start
Define problem parameters
Set up mesh and solver arrays
Initial guess
Calculate residuals
Call solution routine: Frontal solver or Sparse/Iterative solver
See Figure 3
Update solution vector
L_2 norm below tolerance?
No
Yes
Another solution step ?
Yes
No
Calculate $\frac{\partial(\text{Residual})}{\partial\alpha}$ and new guess
Redefine parameters
End

Figure 2. Flow diagram of overall solution algorithm

problem. In general, these arrays are calculated previously and read from an unformatted data file for calculations at other parameter values.

In the assembly loop, assembly of the preconditioner is optional. If a pointing array is initialized that points into the 'matrix' vector A, then the inner loop of assembly can be vectorized. However, this array is relatively large, and if memory requirements are stringent and the pointer array is not used, this assembly loop cannot be vectorized at all. Element-by-element assembly can, however, be parallelized or multitasked.

Once the assembly is done, the preconditioner P must be factored if this has not already been done. This process is time-consuming for large-band-width preconditioners, although most of the

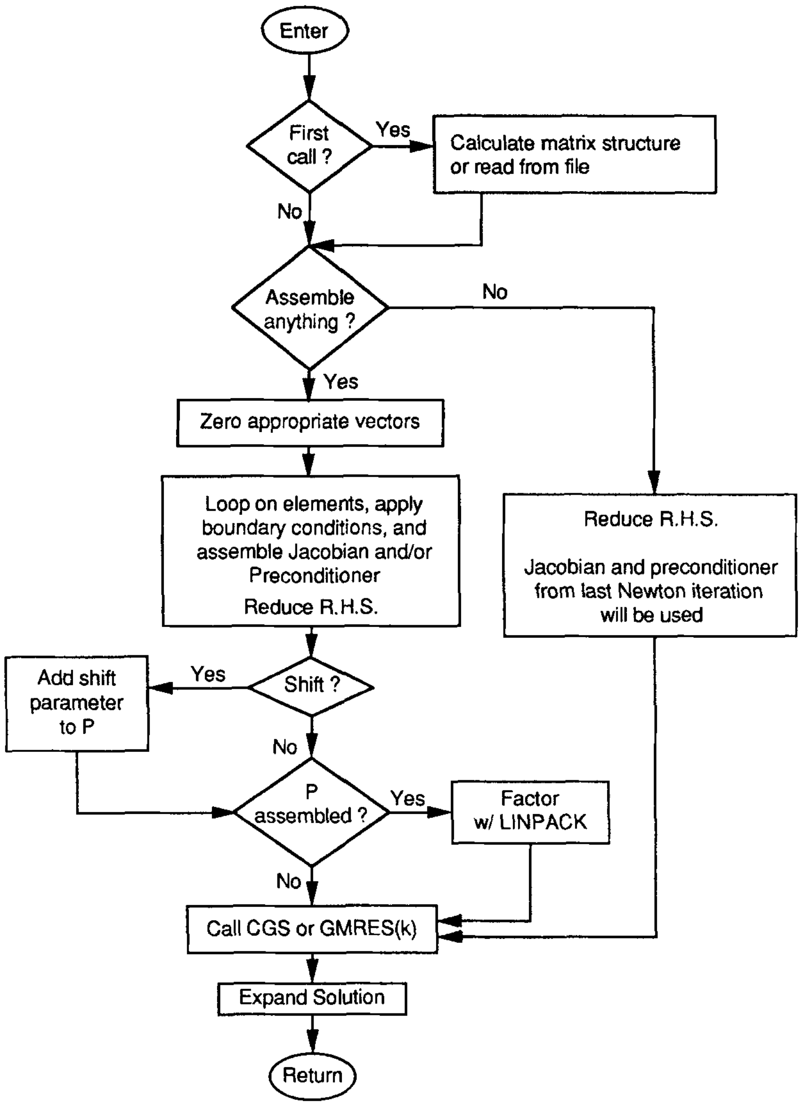


Figure 3. Flow diagram of sparse/iterative solver

time spent in factorization is spent in the BLAS routine SAXPY which executes the operation

```
      DO 10 I=1, N
            Y(I)=Y(I)+E*X(I)
   10 CONTINUE
```

where E is some constant real value and Y and X are vectors of length N. A way of avoiding the costly factorization step is to assemble and factor only on the first Newton iteration. This preconditioner was found to be adequate for the solution at all the subsequent Newton iterations in the solution algorithm and thereby reduced the computational time by avoiding the costly factorization.

Finally, the problem is sent to an appropriate iterative solution algorithm, either CGS or GMRES(k). The time spent in the iterative algorithm is dominated by two processes. The first is the back substitution using P which is done with the LINPACK routine SGBSL. The time in SGBSL is again spent mostly in SAXPY. The second time-intensive routine is the matrix/vector multiplication. This is particularly a problem because of the indirect addressing required in the compressed sparse row storage format. The inner loop of the matrix/vector multiplication is vectorized automatically by the use of GATHER/SCATTER hardware by the compiler.[25] The relative merits and time and iteration counts of these algorithms are presented below.

RESULTS AND DISCUSSION

The advantage of sparse storage is the efficient utilization of core memory so that the whole matrix is available for rapid access to its entries. But LU decomposition or Gauss elimination with pivoting results in matrix fill-in. Since the pivotal element is determined by a conditional statement during the elimination, the location of this fill-in together with the necessary storage is exceedingly difficult to determine *a priori.*[19] This leads to the use of iterative solution methods which use the sparse matrix only in matrix vector multiplications; but iterative methods depend critically on the quality of the preconditioner.[24] The performance of the present preconditioner is discussed after the memory usage in the finite element formulation.

Storage requirements

Table II presents the memory requirements for various meshes. The first column gives the mesh dimensions, with transverse discretizations given in parentheses. The second column is the total number of unknowns in the FEM problem. These unknowns include all fixed as well as free boundary conditions. When the Dirichlet conditions are applied and the appropriate fixed degrees of freedom eliminated from the matrix problem, the number of unknowns in the problem decreases as shown in column 3. Note that a significant decrease in the size of the resultant matrix problem for fine FEM discretizations suggests that this application of fixed boundary conditions is worthwhile. For problems with more than 40 000 unknowns in the original formulation, more than 10 000 degrees of freedom can be eliminated and the problem reduced by 25%. Of course, the savings are less dramatic for coarse meshes. In general, more nodes on the boundaries result in a

Table II. Memory requirements

Mesh	Total number of unknowns in system	Order of reduced system	Number of non-zeros in Jacobian	Total storage required for Jacobian (M words)
$(4\times4)\times40$	27269	19027	3766948	7·2
$(5\times5)\times6$	6544	4606	912715	1·7
$(5\times5)\times15$	15580	11476	2399083	4·6
$(5\times5)\times25$	25620	19036	4031563	7·8
$(5\times5)\times40$	40680	30376	6480283	12·4
$(5\times5)\times80$	80840	61166	13153003	25·1
$(6\times6)\times40$	56765	44367	9922564	19·0

greater reduction of the problem dimension. For very fine transverse discretizations a smaller percentage of nodes and degrees of freedom is eliminated on the boundaries.

Column 4 shows the number of non-zeros in the Jacobian matrix for the reduced problem. The total storage required for the Jacobian is given in column 5, taking into account pointer arrays. For a mesh with approximately 80 000 unknows the Jacobian matrix is stored in about 25 Mwords. For problems of the order of 27 000 unknowns the whole Jacobian is stored in under 8 Mwords.

Further storage would be gained by using short integers for the pointing arrays IA and JA. This would decrease the space needed to store the Jacobian matrix so that total storage would now take about 1·5 MSIZE. Also, although the Jacobian is not symmetric, its structure is symmetric. Taking advantage of this would further reduce the required storage for JA.

Assembly

In the assembly loop over the elements, four distinct processes may take place. There is always a call to the subroutine ABFIND which calculates the element stiffness matrix. There are a maximum of 116 degrees of freedom per element, so the element stiffness matrix has 116×116 entries. The time spent in ABFIND is mainly spent on multiplications, divisions, additions and subtractions. Because vector dependences do not exist, the inner loop of the element stiffness matrix calculation is force-vectorized using a complier directive. The second process that takes place is the application of the boundary conditions. This process can be skipped, however, in this problem since the boundary conditions array is filled with zeros (fixed boundary conditions correspond to zero entries in $\Delta\mathbf{u}$).

The third process is the assembly of the elemental contributions into the banded preconditioner. This process is relatively fast since the location in the preconditioner array can be calculated directly with no searching. The fourth process is the search in the global matrix vector A for where a given element contribution belongs. This is a slow process. The assembly speed can be increased by initializing a large pointer array of dimension $116 \times 116 \times$ NE0, where NE0 is the maximum number of elements. For every entry in the element stiffness matrix of a given element, it points to the location in A where that contribution belongs. This array is large and aggravates the storage requirements of the programme execution. However, it enables the vectorization of the assembly loop.

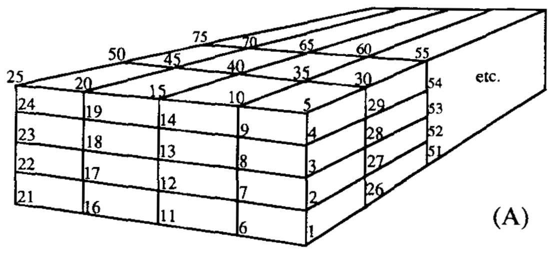
Preconditioner

The preconditioner is one of the most important parts of any practical iterative solution scheme. Most of the time spent in any solution algorithm using iterative solution techniques is related to operations on or with the preconditioner. There are two major operations that need to be undertaken. One is the LU factorization of the preconditioner and the other is the re-solution or back substitution using the LU factors. If a good preconditioner is found, the iterative method will converge rapidly. If calculations with the preconditioner are inexpensive, the time saving can be significant. Usually there is a trade-off between preconditioning quality and the time needed to evaluate and factor the preconditioner.

In the present algorithm the preconditioner was factored only once at the first Newton iteration. The calculated LU factors were used for all subsequent calculations involving the preconditioner, i.e. the preconditioner was not updated in the course of the execution. No observable increase in preconditioning quality resulted when the preconditioner was calculated and factored at each Newton step. Since factorization is costly, this resulted in a significant saving in total CPU usage, especially when relatively large preconditioner band widths were used.

The ideal preconditioner is the exact LU decomposition of the Jacobian matrix; but finding this decomposition is equivalent to a direct solution of the problem itself. As stated earlier, in this study a subset of the full Jacobian matrix around the main diagonal is stored in band format and factored with the LINPACK routine SGBFA. The choice of the size of the band used for preconditioning is the critical question in this problem.

For a sample mesh of 4×4 elements in a cross-section and 10 elements in the longitudinal direction, the nodes (and correspondingly the equations) are numbered starting at (000) and moving first up in the z-direction, then across in the y-direction and finally in the longitudinal or x-direction as shown in Figure 4(A). This numbering scheme leads to a matrix structure of the reduced problem as shown in Figure 4(B), with a close-up of the upper 20% in Figure 4(C). Very rapid convergence of the iterative method (around 25 iterations for a $(4 \times 4) \times 10$ mesh) is obtained with a preconditioner encompassing the three central bands as shown in the shaded region of Figure 4(C). This preconditioner will hereafter be called preconditioner I. These bands include the very basic axial coupling of the residual equations. This suggests that coupling in all three spatial dimensions is necessary for an effective preconditioner. Although this preconditioner leads to rapid convergence of the iterative methods, its band width is roughly the same as the front width of a frontal solution. The suggested band widths and memory requirements shown in



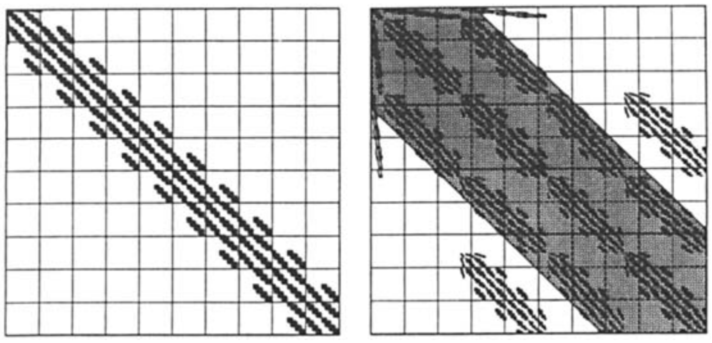


Figure 4. (A) Schematic numbering scheme for preconditioner I. (B) Matrix structure of a reduced $(4 \times 4) \times 10$ finite element discretization. The size of the reduced system is 5007. (C) Upper 20% (1000×1000) of matrix shown in (B) demonstrating the structure of the bands. Shaded region indicates preconditioner I

Table III demonstrate that the band width of the preconditioner rapidly becomes very large for fine transverse discretizations.

If the nodes are renumbered, the resultant matrix graph can contain more information about the overall solution near the diagonal of the Jacobian. Since the overall band width of the problem is not an issue in the sparse matrix storage, it is desirable to renumber the domain so that coupling of the equations in all spatial dimensions is packed near the diagonal. This is done by renumbering the domain as shown in Figure 5(A). The domain is divided into a number of subdomains with smaller band width. These subdomains are then numbered consecutively, resulting in a matrix graph as shown in Figures 5(B) and 5(C). Now the packed major band of this matrix is factored and the LU factorization is used as the preconditioner for the CGS and GMRES algorithms. This preconditioner, shown in the shaded region of Figure 5(C), will be called preconditioner II. Example runs of this preconditioner for various meshes are presented in Table IV.

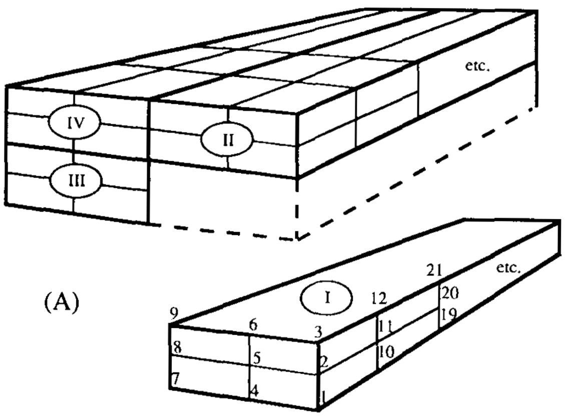
Table III. Required band width of preconditioner I for various transverse discretizations ($n \times n$) for a system of 30000 equations (reduced system)

$n \times n$ (FEM cross-section mesh)	Required band width	Preconditioner storage (M words)
4×4	310	26·6
5×5	460	39·5
6×6	655	56·2
8×8	1120	96·2
10×10	1170	151·9

Table IV. Computational results for preconditioner II (($2\times2\times6$) subdomain, convergence criterion $\|r_n\|/\|r_0\|<10^{-8}$)

Mesh	Total unknowns (reduced)	Solution method	Iterations to convergence at each modified Newton step: Step 1	Step 2	Step 3	Total time for solution (s)	Estimated time for frontal (s)‡
$(4\times4)\times6$	4387	CGS	41	41	49	35·9	12·7
	(3007)	GMRES(25)	127	127	112	45·2	
		GMRES(50)	85	85	87	35·9	
		GMRES	65	65	64	31·4	
$(6\times6)\times6$†	9131	CGS*	106	106	105	149·2	94
	(6887)	GMRES(50)	291	290	242	186·1	
		GMRES(100)	161	161	153	119·0	
		GMRES*	122	122	117	105·9	
$(8\times8)\times6$	15595	CGS	194	199	186	431	431
	(12359)	GMRES	192	192	186	278·1	
$(10\times10)\times6$	23779	CGS	No convergence		(>300)	NA	1465
	(19423)	GMRES	272	272	264	659·8	

* Runs used in Table V.
† Convergence shown in Figure 6.
‡ Assumes 4 ns per operation and disregards Jacobian calculation ($4NF^2+9NF$ total operations).



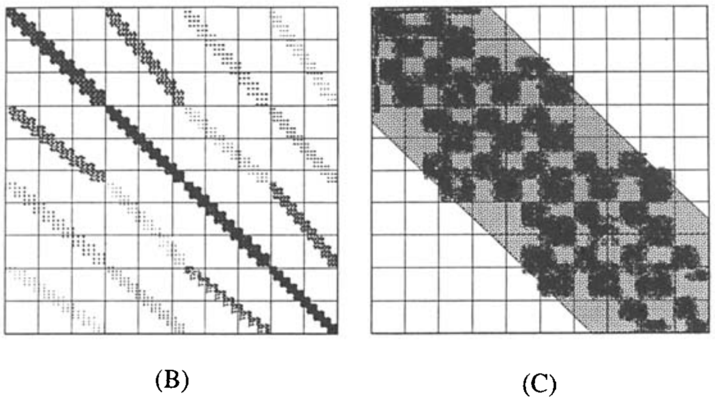


Figure 5. (A) Schematic numbering scheme for preconditioner II. Same physical problem and mesh as in Figure 4. (B) Matrix structure of renumbered FEM discretization shown in (A). (C) Upper 10% of matrix shown in (B). Shaded region indicates preconditioner II

Solution algorithms

As stated earlier, two general solution algorithms that are applicable to non-symmetric systems are investigated. The CGS algorithm is attractive because it stands 'as is' with no parameters that need to be chosen as in GMRES(k), where the subspace dimension k must be chosen. Furthermore, CGS requires less storage because the subspace basis vectors that are stored in GMRES(k) are not required.

The convergence characteristics of the two methods are very different as shown in Figure 6. The convergence of the CGS algorithm can be rapid but erratic, with large variations up and down in the residual norm while overall convergence is approached. In GMRES(k) convergence is monotonic, with a steady decrease in the residual norm. However, the convergence can be slow if the subspace dimension is chosen too small such that several restarts are required. In general, it is suggested to use as large a Krylov subspace as is practical, since non-restarted GMRES is apparently significantly better than restarted GMRES.

Table IV gives iteration counts and solution times for various meshes using the CGS and GMRES algorithms with various Krylov subspace sizes. For the present application we favoured

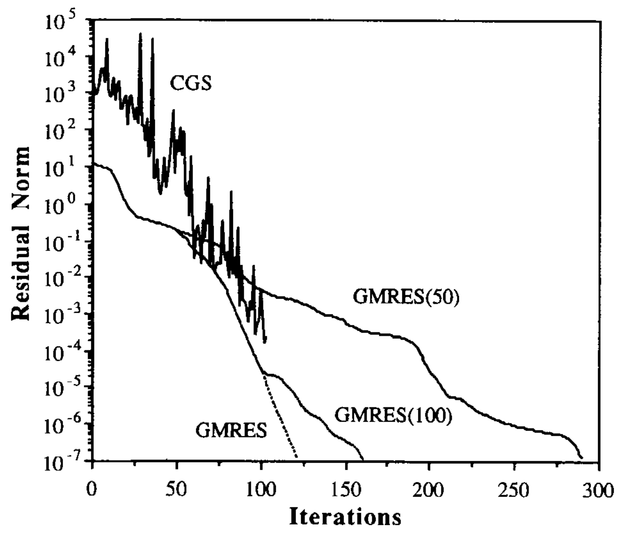


Figure 6. Convergence CGS and GMRES(k) algorithms using Preconditioner II

the GMRES algorithm over the CGS algorithm because it required fewer operations per iteration as discussed below. However, if memory restrictions exist, CGS is preferred. Further comparative analysis of the quality of the two algorithms is required to determine if one is superior for this preconditioner and problem.

Continuation

The use of the iterative solution methods for the calculation of a continuation step (solution of equation (19)) was found to work well using preconditioner I. Typically, only 20–30 CGS iterations were required to converge to a solution. It is unknown, however, how well this technique will work near bifurcation and turning points when the Jacobian matrix is approaching singularity. It is possible that as the matrix becomes less well conditioned, the convergence properties deteriorate. However, a good preconditioning matrix may allow for robust continuation. A disadvantage of the use of sparse matrix techniques is that it is difficult to calculate the determinant of the Jacobian. The sign of the determinant is a useful indicator of a change in the stability of the solution.[12]

Timings and operation counts

The time-intensive operations required for typical runs using CGS and GMRES are shown in Table V. In these runs a pointer array was used to avoid the search in the 'matrix' vector A. Furthermore, the structure of A was read from an unformatted file created earlier. As can be seen, the majority of time is spent in the preconditioner factorization, the assembly loop and the iterative method. Furthermore, the time in the solution algorithm is dominated by back substitutions and matrix/vector multiplication. For a given problem it was found that each GMRES iteration was approximately twice as fast as each CGS iteration. Because the GMRES iterations are faster than the CGS iterations, if memory permits, the GMRES algorithm is favoured.

Table V. Time-intensive operations ($(6 \times 6) \times 6$ FEM mesh, preconditioner II with $(2 \times 2) \times 6$ subdomain)

Operation	CGS	GMRES
Factorization (one time with LINPACK)	5·5%	7·7%
Jacobian calculation (call to ABFIND)	9·9%	13·8%
Re-solution with LU decomposition (LINPACK)	43·1%	35·2%
Matrix/vector multiplication	32·3%	26·1%

The test of a numerical method is its memory and operation count comparison with established methods. The present method is compared with the frontal method of Irons[13] and Hood[14] which is a direct method. A conservative analysis takes the operation count of frontal methods as essentially the same as that of band solvers with the front width substituted for the band width. For large values of N (total number of unknowns) and a front width F the total number of operation counts for one complete factorization (or solution if standard Gauss elimination is used) with pivoting is about $2NF^2$. For the formulation considered, F is approximately one-half the total band width. Since back substituions are accomplished in $O(NF)$ operations, they are insignificant compared with the factorization. The total operation count for the frontal method is then about $2NF^2$ for large values of N and F. The memory requirements of the frontal method depend on whether or not the LU factors are stored. If L and U are not stored, the memory overhead is very small (of the order of F^2 if only the 'frontal' matrix is stored). The disk requirements are quite large, though, for large problems. If the full LU decomposition is required, it is advantageous to keep the full LU decomposition in core and the memory requirement is even larger.

The present iterative method is dominated by the factorization of the preconditioner (stored in band storgage) and the CGS or GMRES iterations, which are dominated by back substitutions and matrix/vector multiplications. For a preconditioner band width P the factorization with pivoting takes $2NP^2$ operations[16, 19] and a storage of $3NP+1$.[20] Each iteration of the CGS iterative method requires two matrix/vector multiplication (approximately $2NP$ operations) and two back substitutions (approximately $6NP$ operations). The operation count for the GMRES algorithm is about half of this but is also a function of the restart parameter (subspace size). Notice that for this comparison the Jacobian calculation and assembly is not considered since it is the same for both methods. This leads to a total operation count of about $2NP^2+8MNP$ for the preconditioned CGS iterative method, where M is the total number of iterations required for convergence. The total storage is about $3NP+J$, where J is the storage overhead of the Jacobian discussed earlier.

The relative efficiency of the CGS iterative algorithm execution is defined to be

$$E=(\text{frontal operations})/(\text{iterative operations})\approx NF^2/(NP^2+4MNP). \qquad (20)$$

This efficiency is not completely independent of the total number of unknows, N since the condition number gets larger and M increases[17] as the size of a system increases. If the full front width is used as the preconditioner ($P=F$), then the frontal method will always be faster for the first Newton step because of the added cost of the iterations. However, after the preconditioner is factored, each successive Newton step takes only the cost of the iterative method, which is cheap (approximately $8MNP$). This can make the iterative algorithm with preconditioner I potentially faster than the frontal method if many full Newton steps are required.

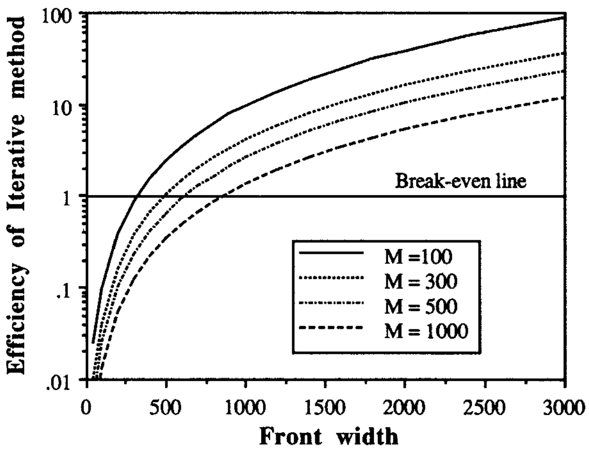


Figure 7. Efficiency of CGS iterative method versus front width. M is the total number of CGS iterations to convergence

Using preconditioner II ((2 × 2) × NX subdomain preconditioning), it was found that $P \approx 180$ is adequate such that

$$E = F^2/(180^2 + 720M). \tag{21}$$

If $M \approx 100$, then the critical F for equal operation counts is about 320, which is below the front widths considered in the present problem ($F \approx 2000$ for a 10 × 10 cross-discretization, $F \approx 425$ for a 4 × 4 cross-discretization). Figure 7 shows the efficiency of the CGS iterative method compared with the frontal method for various iteration counts. For all the cases there is a break-even point in front width above which the iterative solver is more efficient. As the number of iterations decreases, the break-even front width decreases. For reasonable counts the break-even point is for a front width between 300 and 800. This is smaller than the scale of the present problem. This analysis demonstrates that for large problems (large band widths or large front widths) the iterative solution is worthwhile and the problem is solved in a shorter amount of CPU time. In addition, the cost of the Newton steps after the first step is much lower. It should be noted that a similar analysis using the GMRES algorithm would show the iterative method as being even more favourable, since the GMRES iteration was found to be faster than the CGS iteration in this study.

Further speed-up of the solution algorithm may come from various sources. A candidate for speed-up is in the calculation of the Jacobian entries. This can be done with eight-point Gauss quadrature instead of 27-point quadrature with some loss of accuracy. This option should be investigated to determine how much solution accuracy is lost. In preliminary tests it was found that convergence of the iterative solution algorithms suffered greatly when using lower-order quadrature. This is attributed to a worsening of the condition number of the resultant Jacobian matrix when lower-order quadrature was used.

In the solution algorithm, operations with the preconditioner done with LINPACK routines are dominated by SAXPY calls, so a faster SAXPY routine would decrease the execution time drastically. Further speed-up requires optimization of the assembly and matrix/vector multiplication operations. These two operations are candidates for parallel processing. If non-contiguous elements are operated on simultaneously, the assembly process may be multitasked. As for the matrix/vector multiplication, since the inner loop is already vectorized, the next logical option is optimization of the outer loop. This can be multitasked on a smaller scale—'microtasked' in Cray language.[26]

Fluid flow solutions

Gas flow in a long rectangular duct of aspect ratio two and heated from below is simulated as an example of the use of this solution algorithm. The channel is 2·5 cm high and 5·0 cm wide. A symmetry plane is assumed at the half-width so only half the domain is solved for. The entrance region is 10 cm long and the heated susceptor region is 30 cm long. A 10 cm exhaust region insures that outlet effects do not influence the flow in the susceptor region. The discretization in the cross-section is 8×8, with eight, 20 and three elements in the three axial regions. For this discretization the total number of equations was 75 420, with 63 268 equations in the reduced problem. Using the restarted GMRES algorithm with a Krylov subspace size of 300, 530 iterations were required for convergence at each Newton step. The total time for the 530 iterations was about 1100 s on a Cray-2 512 M word supercomputer. The LU decomposition of the preconditioner took 82 s and the assembly of the Jacobian and preconditioner took 83 s. Because the essential physics was captutred with a coarser mesh, a $(4 \times 4) \times 40$ mesh was used for the second and third examples below.

The first case considered is nitrogen flow. The susceptor is heated to 294·5 K while the inlet fluid and top wall are fixed at 293 K, leading to a value of $\Delta T/T_0 = 0{\cdot}005$. This small value of $\Delta T/T_0$ indicates that this N_2 flow is nearly Boussinesq. Figure 8 shows two symmetric particle path lines, velocity profiles at various channel heights and axial distances and transverse arrow plots for a Rayleigh number $Ra = 2385$ and Reynolds number $Re = 24{\cdot}8$ ($v_{\mathrm{max}} = 1{\cdot}5$ cm s^{-1}). Note that the values of Re and Ra are defined with respect to the inlet temperature T_0 and the maximum inlet velocity (centre-line velocity). To find Re based on the average inlet velocity, the present Re should be divided by 1·9981.

For this supercritical Ra it is well known longitudinal rolls develop downstream[27] as observed in Figure 8. For this Re there is no back flow in the entrance region of the channel. If the inlet flow rate is reduced to $Re = 8{\cdot}3$ ($v_{\mathrm{max}} = 0{\cdot}5$ cm s^{-1}), a recirculation forms in the entrance region of the susceptor as seen in Figure 9. The longitudinal circulation downstream is in the direction also predicted by marching techniques for adiabatic side walls.[27] However, this solution could not be correctly obtained with numerical marching methods since back flow exists.

If the inlet flow rate is decreased further to $Re = 6{\cdot}7$ ($v_{\mathrm{max}} = 0{\cdot}404$ cm s^{-1}), the longitudinal roll direction downstream reverses because of the strengthening of the transverse roll as shown in Figure 10. This solution was reached with first-order continuation from the solution in Figure 9. Since the iterative scheme makes it difficult to evaluate the determinant of the Jacobian matrix, it is not clear whether or not the solution is stable. It is possible that this solution is unstable and that the stable solution in this parameter regime is time-dependent. The assumed plane of symmetry at the half-width for this calculation may further constrain the possible solutions.

Fluid flow calculations have been performed for larger values of $\Delta T/T_0$ which are relevant to chemical vapour deposition (CVD) reactor flows. These simulations demonstrate how non-Boussinesq conditions affect the fluid solutions. Similar fluid phenomena have been observed. For example, the reversal of the direction of the downstream longitudinal roll has been observed in H_2 flow simulations with $\Delta T = 600$ K ($\Delta T/T_0 = 2$) and $v_{\mathrm{max}} = 4{\cdot}0$ cm s^{-1}. It is again unknown whether or not these solutions are stable, but if these flows are realizable in CVD reactor operation, they will have a strong effect on the growth rate uniformity realized in the reactor.[2]

The present solution technique has also been applied to H_2 flow simulations with Rayleigh number as high as 22 000 and Reynolds numbers as high as 60 (average v_0 basis) and there was no

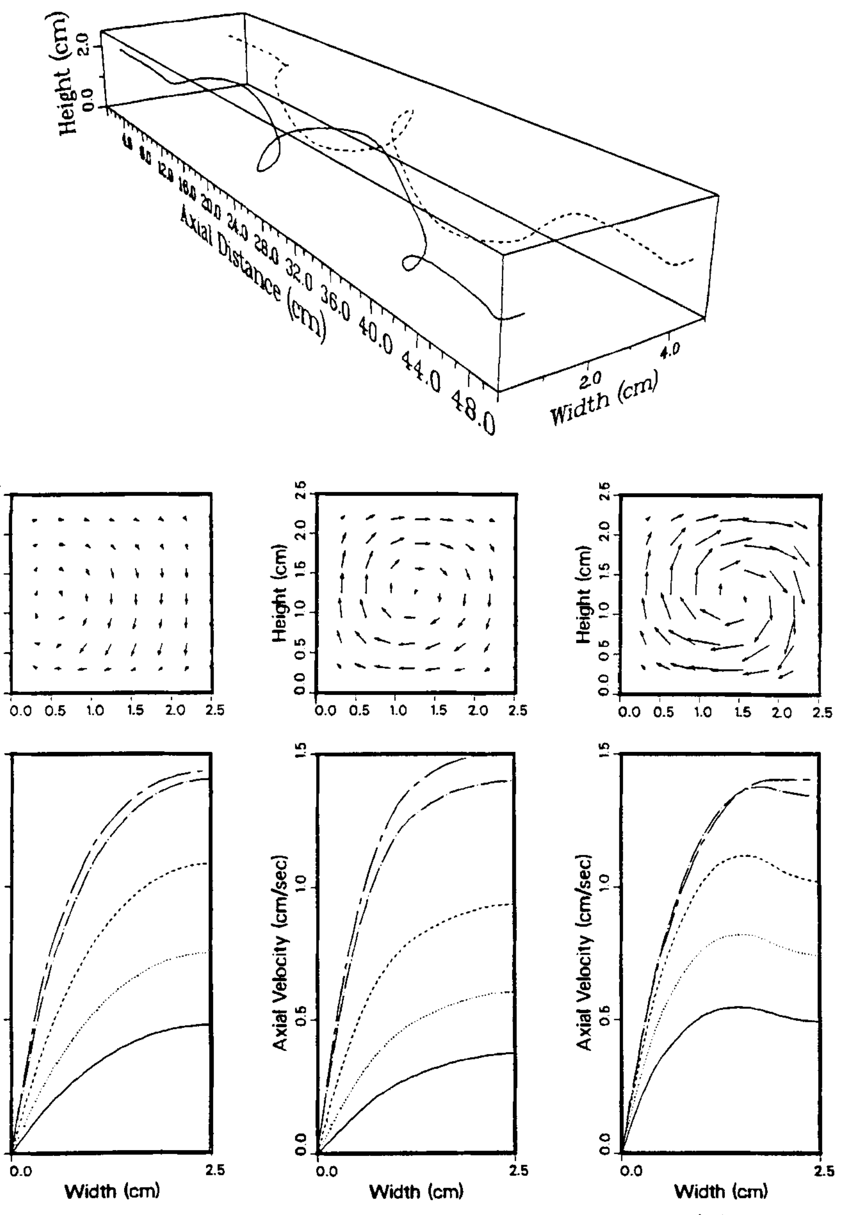


Figure 8. Particle path lines, cross-section arrow plots and axial velocity profiles for nitrogen flow at $Re=24{\cdot}8$, $Ra=2385$ and $\Delta T/T_0=0{\cdot}005$. Axial velocity profiles are shown at: solid line, 0·88 H; dotted, 0·80 H; dashed, 0·68 H; dash/dot, 0·48 H; dash/dash, 0·40 H. (a) 10 cm (start of heated region); (b) 12 cm; (c) 30 cm

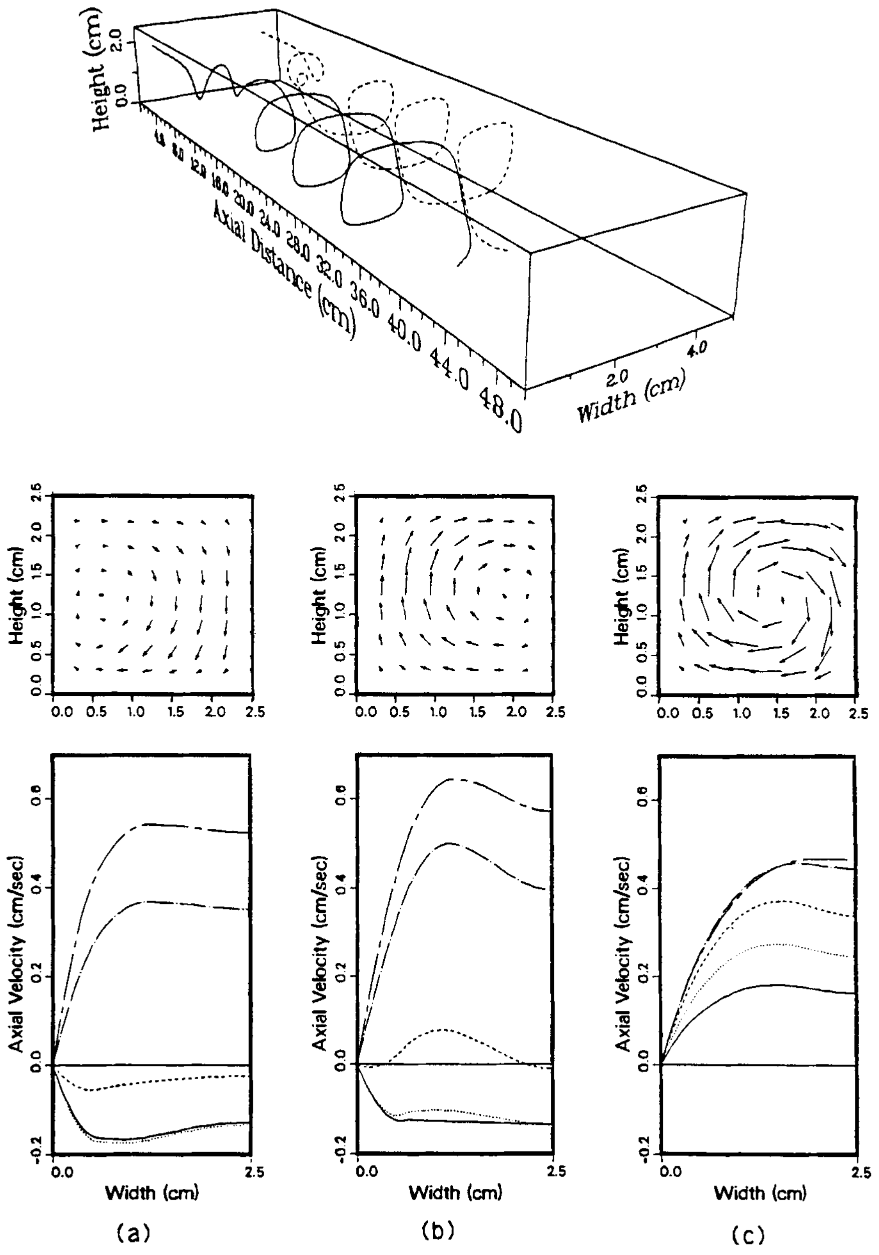
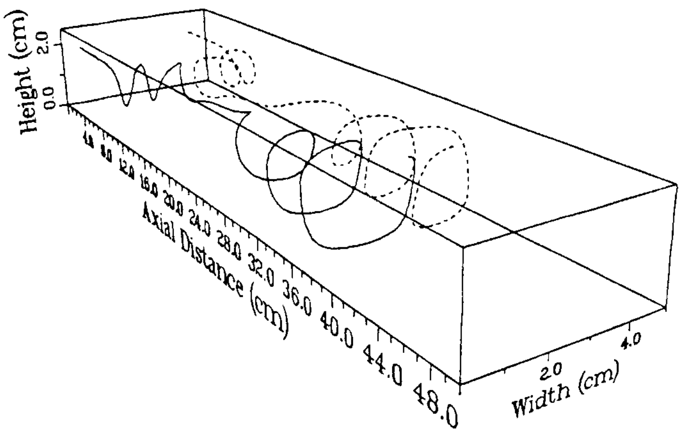


Figure 9. Particle path lines, cross-section arrow plots and axial velocity profiles for nitrogen flow at $Re = 8{\cdot}28$, $Ra = 2385$ and $\Delta T/T_0 = 0{\cdot}005$. Axial velocity profiles are shown at: solid line, 0·88 H; dotted, 0·80 H; dashed, 0·68 H; dash/dot, 0·48 H; dash/dash, 0·40 H. (a) 10 cm (start of heated region); (b) 12 cm; (c) 30 cm



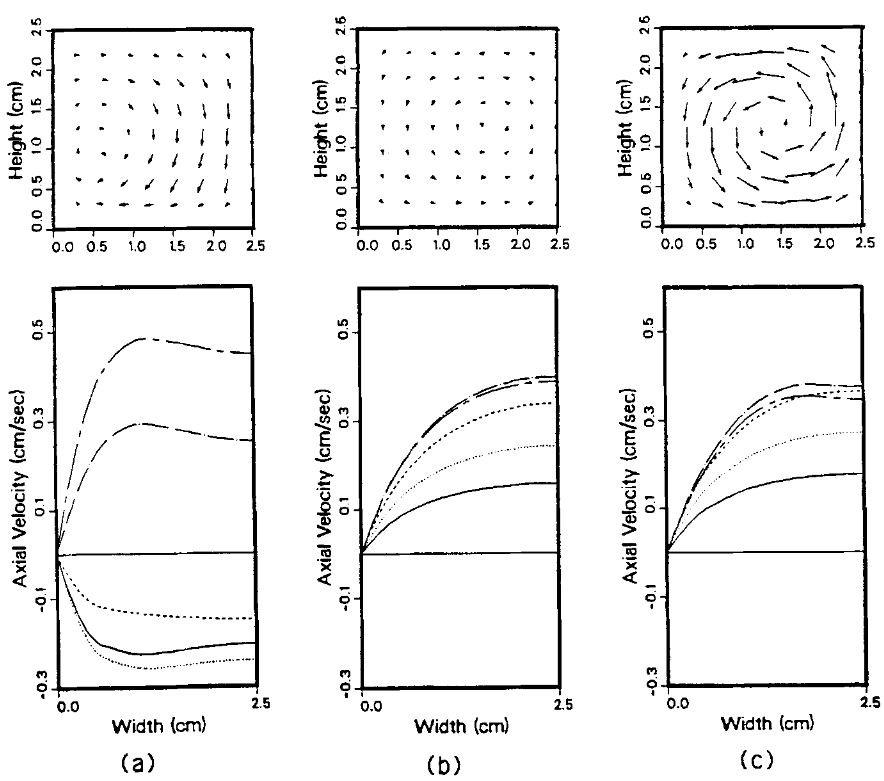


Figure 10. Particle path lines, cross-section arrow plots and axial velocity profiles for nitrogen flow at $Re = 6{\cdot}7$, $Ra = 2385$ and $\Delta T/T_0 = 0{\cdot}005$. Axial velocity profiles are shown at: solid line, 0·88 H; dotted, 0·80 H; dashed, 0·68 H; dash/dot, 0·48 H; dash/dash, 0·40 H. (a) 10 cm (start of heated region); (b) 12 cm; (c) 30 cm

apparent degradation of the performance of the solution method. At such high Reynolds numbers, however, it would be more efficient to use a parabolic assumption and integrate in the axial dimension of the channel.[28]

CONCLUSIONS

The present study was undertaken to develop an efficient finite element technique for the assembly and solution of a three-dimensional mixed convection fluid flow problem. The scale of the model and the sparsity of the resultant matrix system lead to the use of sparse matrix storage techniques and subsequently iterative solution algorithms for the solution of the problem. Several conclusions regarding the implementation of a solution for this type of problem may be drawn:

1. A preconditioner centred around the diagonal of the Jacobian and encompassing the fundamental spatial coupling of the residual equations in all three dimensions is adequate for convergence of the tested iterative matrix solution algorithms.
2. Assembly and factorization of the preconditioner only at the first Newton step is adequate for subsequent Netwon steps and greatly enhances the overall algorithm efficiency.
3. CGS and GMRES are comparable solution algorithms in total iteration counts, but although GMRES requires slightly more storage, it is faster and therefore favoured. Also, if memory permits, restarting the GMRES iterations is not suggested.
4. First-order continuation steps can be readily calculated using iterative solution methods; however, it is not known how robust the convergence will be near bifurcation and turning points when the Jacobian matrix approaches singularity.

The conclusions above relate directly to the general solution algorithm that was used in this problem. The bottle-necks that keep the solution time slow are primarily related to operations with the preconditioner and the assembly of the Jacobian. The bottle-necks in the solution algorithm can be addressed for faster execution time with

(1) a faster SAXPY routine
(2) parallel element-by-element Jacobian calculation and assembly
(3) multitasked matrix/vector multiplication.

To enhance the performance of the preconditioner, the use of a shift parameter[29] may be a viable option. The use of a shift parameter could also eliminate the problem of a non-diagonally dominant row such as the continuity equation contribution to the physical problem. Simple tests using a shift parameter have not been conclusive, but the preservation of some form of pivoting to preserve the nature of the continuity equation seems to be necessary. Another related option is to use some form of diagonal lumping.

It has been found that the preconditioning performance depends critically on including coupling of the equations of motion in all three spatial dimensions. The bands around the diagonal of the Jacobian give an approximate description of the physical problem. By solving exactly over subdomains of the global domain, we have constructed an effective preconditioner. The present renumbering is probably not the most efficient, since other renumbering schemes that lump an approximate description of the problem near the diagonal may lead to more effective preconditioners.

ACKNOWLEDGEMENTS

We would like to acknowledge fruitful discussions with H. Moffat, D. Fotiadis and Dr. J. Shadid. We would also like to thank R. C. Hamlen for information and help with the CGS algorithm and the reviewers for very helpful comments. This work was supported by the National Science Foundation and the Minnesota Supercomputer Institute.

REFERENCES

1. M. E. Braaten and S. V. Patankar, 'Analysis of laminar mixed convection in shrouded arrays of heated rectangular blocks', *Int. J. Heat Mass Transfer*, **28**, 1699–1709 (1985).
2. K. F. Jensen, E. O. Einset and D. I. Fotiadis, 'Flow phenomena in chemical vapor deposition of thin films', *Ann. Rev. Fluid Mech.*, **23**, 197–223 (1991).
3. W. R. Briley, 'Numerical method for predicting three-dimensional steady viscous flow in ducts', *J. Comput. Phys.*, **14**, 8–28 (1974).
4. R. Chilukuri and R. H. Pletcher, 'Numerical solutions to the partially parabolized Navier–Stokes equations for developing flow in a channel', *Numer. Heat Transfer*, **3**, 169–188 (1980).
5. G. Dhatt and G. Touzot, *The Finite Element Method Displayed*, Wiley, New York, 1984.
6. R. B. Bird, W. E. Stewart and E. N. Lightfoot, *Transport Phenomena*, Wiley, New York, 1960.
7. H. K. Moffat and K. F. Jensen, 'Three-dimensional flow effects in silicon CVD in horizontal reactors', *J. Electrochem. Soc.*, **135**, 459–471 (1988).
8. R. K. Shah and A. L. London, *Advances in Heat Transfer: Laminar Flow Forced Convection in Ducts*, Academic, New York, 1978.
9. G. Strang and G. Fix, *An analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.
10. P. S. Huyakorn, C. Taylor, R. L. Lee and P. M. Gresho, 'A comparsion of various mixed-interpolation finite elements in the velocity–pressure formulation of the Navier–Stokes equations', *Comput. Fluids*, **6**, 25–35 (1978).
11. C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, New York, 1987.
12. H. B. Keller, 'Numerical solution of bifurcation and nonlinear eigenvalue problems', in P. H. Rabinowitz (ed.), *Applications of Bifurcation Theory*, Academic, New York, 1977.
13. B. M. Irons, 'A frontal solution program for finite element analysis', *Int. j. numer. methods eng.*, **2**, 5–32 (1970).
14. P. Hood, 'Frontal solution program for unsymmetric matrices', *Int. j. numer. methods eng.*, **10**, 379–399 (1976).
15. *SMPACK Users Guide*, Scientific Computing Associates, New Haven, CT, 1985.
16. A. Jennings, *Matrix Computations for Engineers and Scientists*, Wiley, New York, 1985.
17. L. A. Hageman and D. M. Young, *Applied Iterative methods*, Academic, New York, 1981.
18. D. L. Boley, D. G. Truhlar, Y. Saad, R. E. Wyatt and L. A. Collins (eds), *Minnesota Supercomputer Institute Workshop on Practical Iterative Methods for Large Scale Computations*, North-Holland, Amsterdam, 1989.
19. G. Dahlquist and Å. Björk, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
20. J. J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, *Linpack Users Guide*, SIAM, Philadelphia, PA 1979.
21. P. Sonneveld, 'CGS (conjugate gradients squared), a fast Lanczos-type solver for nonsymmetric linear systems', *Delft University of Technology, Reports of the Department of Mathematics and Informatics No. 84-16*, 1984.
22. Y. Saad and M. H. Schultz, 'GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **7**, 856–869 (1986).
23. S. C. Eisenstat, H. C. Elman and M. H. Schultz, 'Variational iterative methods For nonsymmetric systems of linear equations', *SIAM J. Numer. Anal.*, **20**, 345–357 (1983).
24. C. P. Jackson and P. C. Robinson, 'A numerical study of various algorithms related to the preconditioned conjugate gradient method', *Int. j. numer. methods eng.*, **21**, 1315–1338 (1985).
25. *CFT77 Reference Manual, SR-0018(C)*, Cray Research, Mendota Heights, MN, 1987.
26. *Cray-2 Multitasking Programmer's Guide, SN-2026* (*C*), Cray Research, Mendota Heights, MN, 1988.
27. H. K. Moffat and K. F. Jensen, 'Complex flow phenomena in MOCVD reactors I. Horizontal reactors', *J. Cryst. Growth*, **77**, 108–119 (1986).
28. E. O. Einset, 'Transport and kinetic modeling of horizontal metalorganic chemical vapor deposition (MOCVD) reactors', *Ph.D. Thesis*, University of Minnesota, Minneapolis, MN, 1991.
29. T. A. Manteuffel, 'An incomplete factorization technique for positive definite linear systems', *Math. Comput.*, **34**, 473–497 (1980).