INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **30**: 609–634 (1999)

ITERATIVE AND MULTIGRID METHODS IN THE FINITE ELEMENT SOLUTION OF INCOMPRESSIBLE AND TURBULENT FLUID FLOW

N. LAVERY* AND C. TAYLOR

Department of Civil and Mechanical Engineering, University of Wales, Swansea, Swansea SA2 8PP, UK

SUMMARY

Multigrid and iterative methods are used to reduce the solution time of the matrix equations which arise from the finite element (FE) discretisation of the time-independent equations of motion of the incompressible fluid in turbulent motion. Incompressible flow is solved by using the method of reduce interpolation for the pressure to satisfy the Brezzi–Babuska condition. The k–l model is used to complete the turbulence closure problem. The non-symmetric iterative matrix methods examined are the methods of least squares conjugate gradient (LSCG), biconjugate gradient (BCG), conjugate gradient squared (CGS), and the biconjugate gradient squared stabilised (BCGSTAB). The multigrid algorithm applied is based on the FAS algorithm of Brandt, and uses two and three levels of grids with a 'V-cycling' schedule. These methods are all compared to the non-symmetric frontal solver. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: incompressible; turbulent; finite element method (FEM); reduced interpolation; k–l turbulence model; non-symmetric matrix solvers; frontal solver; least squares conjugate gradient (LSCG); biconjugate gradient (BCG); conjugate gradient squared (CGS); biconjugated gradient squared stabilized (BCGSTAB); multigrid method

1. INTRODUCTION

Computational fluid dynamics (CFD) plays an increasingly important role in the design of modern day industrial components. Of the various computational methods [1–6] of discretising the governing equations and spatial domains, the finite element method (FEM) has the capability of accepting complex geometries in an integrated fashion, making it particularly interesting to the designers of complex components such as gas turbine blades and the cooling system for these. However, it has always been the problem of the FEM that larger computational times have been associated with it. This is especially the case in the area of incompressible and turbulent fluid flow.

For the numerical solution of incompressible fluid flow, it is possible to class the most commonly used computational methods into those that solve by using an artificial compressibility method, those that use an integrated approach and those that use a projection (or

* Correspondence to: Department of Materials Engineering, University of Wales, Swansea, Singleton Park, Swansea SA2 8PP, UK.

Contract/grant sponsor: ABB Gas Turbine Development

CCC 0271–2091/99/140609–26$17.50
Copyright © 1999 John Wiley & Sons, Ltd.

Received May 1997
Revised October 1997

fractional step) method. A prototype of the artificial compressibility methods is the algorithm presented by Chorin [7]. In the area of projection methods, the early work of Tuan and Olson [8], and Schneider and Raithby [9] form a solid basis for these types of methods, with more recent elaboration by Patankar [10] and Gresho [11].

In this investigation, the method of reduced interpolation originally proposed by Taylor and Hood [12] is adopted, in which a lower-order of interpolation at element level is used for the pressure variable in comparison with the velocity variables. This specification ensures that the Brezzi–Babuska condition [13] is satisfied, and part of the attraction of the method is that no artificial up-winding methods are required. The method has been used successfully in applications of incompressible flow to laminar and turbulent flow regimes, using first- and second-order turbulence models [14–16] in two- and three-dimensions with varying levels of geometric complexity. The method can be said to be solid and reliable, in the sense that no convergence problems are associated with it. However, the growth of computing time with problem size renders it costly for large problems, especially in three-dimensions, and with the second-order closure models, in which more variables need to be solved for at each non-linear iteration. In a first instance, this could be attributed to the fact that until recently workers had stuck to using the non-symmetric frontal solver for the matrix equations, particularly for the velocity–pressure matrix equations. It is well-known that the frontal solver, being a direct matrix solver, shares similar asymptotic convergence rates as these, which can be as good as $O(\alpha^2 N)$ where α is the front width, and N is the number of unknowns [17].

Iterative solvers on the other hand, give theoretically better asymptotic convergence rates, such as $O(N^{1.6})$ [18], but it is only recently that they have been applied to the solution of incompressible fluid flow with the reduced interpolation and FEM [19]. However, the method of reduced interpolation was found to create very ill-conditioned stiffness matrices partly due to large off-diagonal non-symmetric terms, and zeros on the leading diagonal due to the pressure. In view of this ill-conditioning, the regions in which computational time improves could only be observed with exceptionally large problems.

Another type of solution algorithms are the multigrid methods. These methods were originally developed for elliptic partial differential equations by Fedorenko [19] and were subsequently formalised by Brandt [5] to cover other systems of equations. In CFD, Brandt [20] gave some early outlines of the application of the method. There have since been many applications to the Euler and Navier–Stokes equations, but the application to turbulent fluid flow is more recent, Yokota [21], Bai and Fuchs [22], and others, mostly using either finite difference or finite volume methods. In terms of incompressible flow and FEM, it is only very recently that work has been appearing, e.g. Lonsdale [23], Webster [24], Xia and Taylor [25] and Wille [26].

The main attraction of multigrid methods is that they offer the theoretical possibility of obtaining convergence rates that are directly proportional to the number of unknowns, i.e. asymptotic rates of $O(N)$.

In this investigation, a selection of Lanczos-type solvers (based on gradient reduction schemes) are applied to the FEM solution of turbulent and incompressible flow using the k–l model of turbulence. Subsequently, the fastest of these iterative solvers is coupled with a multigrid algorithm, and the results are compared with the frontal solver in terms of the CPU (central processing unit) time used.

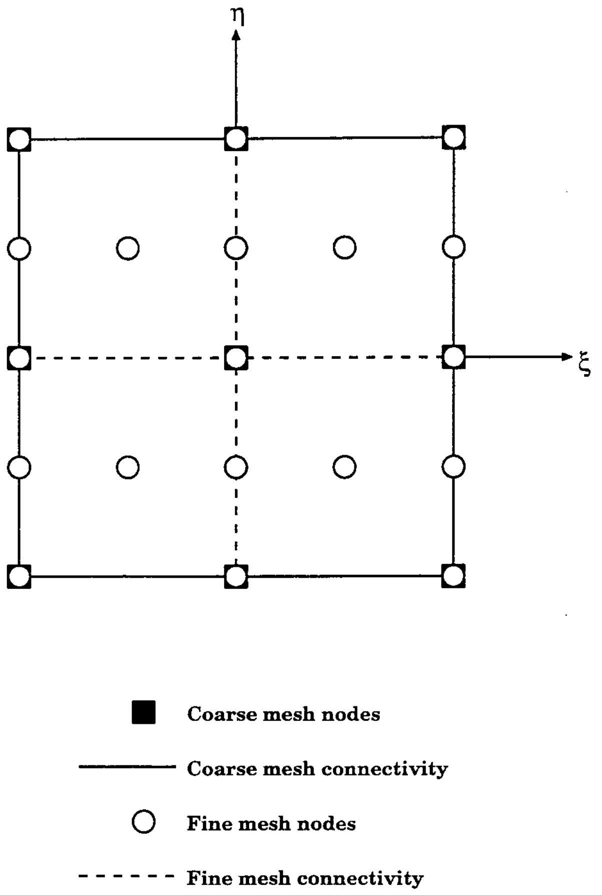

η
ξ
■ Coarse mesh nodes
——— Coarse mesh connectivity
○ Fine mesh nodes
- - - - - - Fine mesh connectivity

Figure 1. Showing the type of subdivision used in the multigrid solution procedure to get from of a coarse mesh to a finer mesh.

2. THE EQUATIONS OF MOTION

Standard manipulations of the Navier–Stokes equations using the concept of fluctuating variables and time-averaging are used to arrive at the following set of equations:

1. Time-averaged continuity

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0. \tag{1}$$

2. Time-averaged momentum

$$\rho\left(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right)=-\frac{\partial P}{\partial x}+\frac{\partial}{\partial x}\left(2\mu_e\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu_e\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right). \tag{2}$$

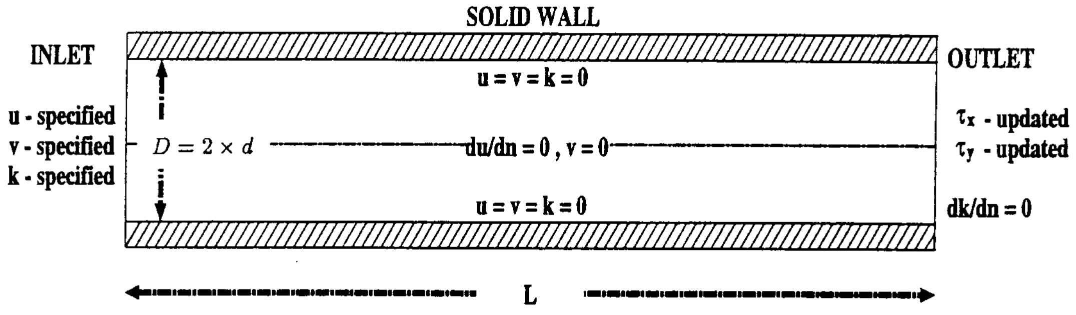
$$\rho\left(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right)=-\frac{\partial P}{\partial y}+\frac{\partial}{\partial y}\left(2\mu_e\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial x}\left(\mu_e\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right). \tag{3}$$

3. Turbulence kinetic energy equation

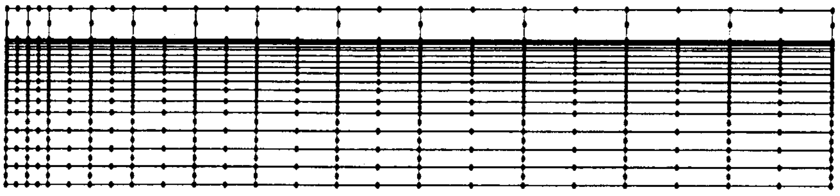
$$\rho\left(u\frac{\partial k}{\partial x}+v\frac{\partial k}{\partial y}\right)=\frac{\partial}{\partial x}\left(\mu_S\frac{\partial k}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu_S\frac{\partial k}{\partial y}\right)+\mu_T S(u,v)-C_D\rho\frac{\sqrt{k^3}}{l}, \tag{4}$$

where

$$S(u,v)=2\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial v}{\partial y}\right)^2+\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2, \tag{5}$$

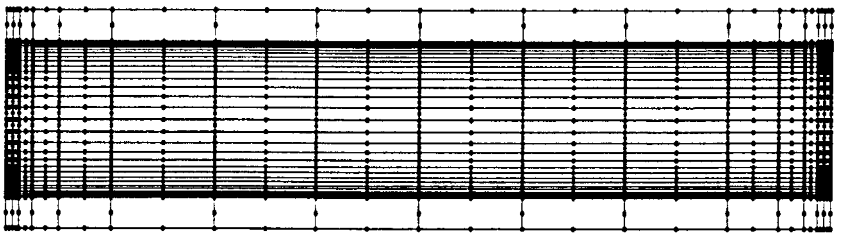


Geometry and boundary conditions for the channel



Title : Finite element mesh for half channel
Number of elements: 228
Number of points: 747
Number of nodes per element: 8

Half mesh used for the FE discretisation



Title : Finite element mesh for the full channel
Number of elements: 672
Number of points: 2133
Number of nodes per element: 8

Full mesh used for the FE discretisation

Figure 2. Geometry and typical meshes used for the example of turbulent flow between parallel plates

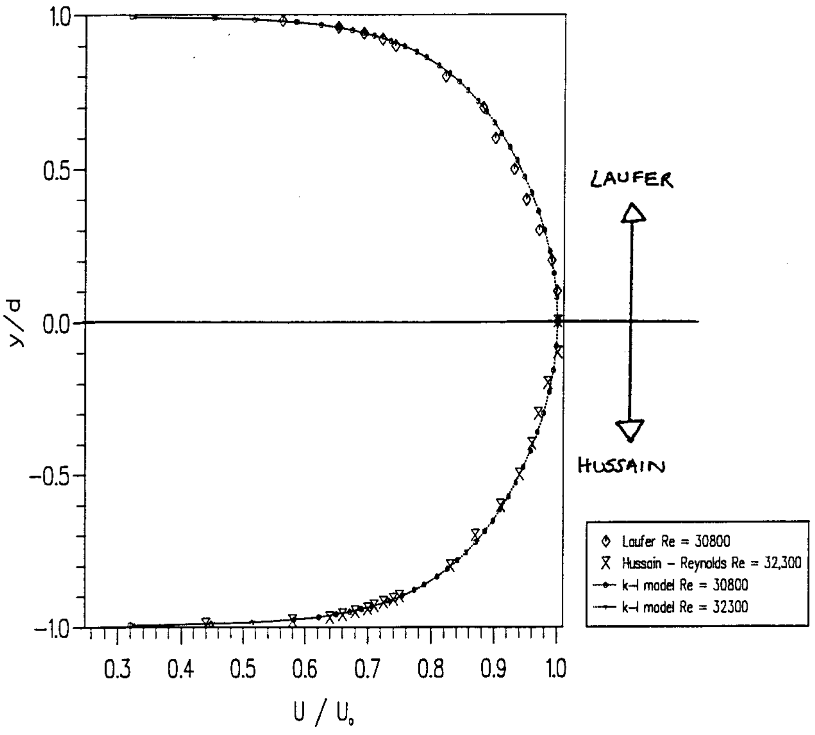


Figure 3. Horizontal velocity, u, predicted by the k–l model compared with the experimental results of Laufer [28] and Hussain and Reynolds [29].

and,

$$\mu_S = \frac{\mu_T}{\sigma_K} + \mu. \tag{6}$$

In the above equations, the following notation has been adopted: u and v refer to the time-averaged horizontal and vertical velocity, respectively; k refers to the turbulence kinetic energy and l refers to the Prandtl length scale; μ_T refers to the localised turbulence viscosity and is given by

$$\mu_T = C_\mu \, \rho \, k^{1/2} \, l, \tag{7}$$

where ρ is the density. In all the equations, $\mu_e = \mu_T + \mu$, is known as the effective viscosity and is made up of the sum of the laminar viscosity of the fluid and the localised turbulence viscosity. Furthermore, σ_K, C_μ and C_D are constants which have been taken as 1.0, 0.22 and 0.418, respectively, following the work of Rance [27].

The length scale, l, was determined by an equation dependent on the geometric position within the flow field.

3. THE FINITE ELEMENT SOLUTION TECHNIQUE

Nine-noded quadrilateral isoparametric elements, with reduced order of interpolation for the pressure, are used and the resulting non-linear, non-symmetric matrix equation can be written in the form

$$\mathbf{H}\underline{\beta} = \mathbf{f}, \tag{8}$$

where

$$\underline{\beta}^k = \begin{Bmatrix} u_k \\ p_k \\ v_k \end{Bmatrix} \text{ for corner nodes,} \qquad \underline{\beta}^k = \begin{Bmatrix} u_k \\ v_k \end{Bmatrix} \text{ for midside nodes,}$$

and

$$\mathbf{H} = \sum_{e=1}^{Ne} \begin{bmatrix} K^{uu} & K^{up} & K^{uv} & 0 \\ K^{pu} & 0 & K^{pv} & 0 \\ K^{vu} & K^{vp} & K^{vv} & 0 \\ 0 & 0 & 0 & K^{kk} \end{bmatrix}. \tag{9}$$

The terms in the finite element stiffness matrix **H** and the right-hand side vector **f** can be found in Reference [16]. A *Picard* scheme is used for the non-linear solution that, if $\underline{\beta}_0$ is an initial approximation to Equation (7), can be expressed as a recurrence relationship of the form

$$\begin{aligned} \underline{\beta}_n^* &= \underline{\beta}_{n-1} \\ \underline{\beta}_{n+1} &= \lambda \underline{\beta}_n^* + (1-\lambda)\underline{\beta}_n, \end{aligned} \tag{10}$$

where, n is the current iteration, $\underline{\beta}_n^*$ is the value used for the non-linear terms in the stiffness matrix and λ is the relaxation factor which usually takes values in the [0, 1] range. In the current investigation, λ was taken as 0.5 and the scheme was said to have converged when

$$\frac{|\underline{\beta}_{n+1} - \underline{\beta}_n|}{|\underline{\beta}_n|} \leq \varepsilon, \tag{11}$$

where ε is the tolerance, which was taken in this investigation as 10^{-2}.

4. MATRIX SOLVING TECHNIQUES

At each iteration of the non-linear scheme, it is necessary to solve a set of linearised matrix equations on the form

$$\underline{\underline{\mathbf{A}}}\underline{\mathbf{u}} = \underline{\mathbf{b}}, \tag{12}$$

where $\underline{\mathbf{u}}$ is the vector of unknowns.

In this investigation, matrix solving algorithms belonging to each of the direct, iterative and multigrid groups have been studied to see which are able to cope with the type of matrix presented by $\underline{\underline{\mathbf{A}}}$, namely, large and non-symmetric, with zeros possibly occurring on the leading diagonal causing severe ill-conditioning.

4.1. The frontal method

This method was first devised by Irons [15] and proved to be effective for symmetric matrices. The prolongation to non-symmetric matrices was then implemented by Hood in 1976 [6]. Since then, the method has been used exhaustively in many finite element applications.

The difference between direct Gaussian elimination and the frontal method is that in Gaussian elimination, the whole of the stiffness matrix has to be stored in the core or main

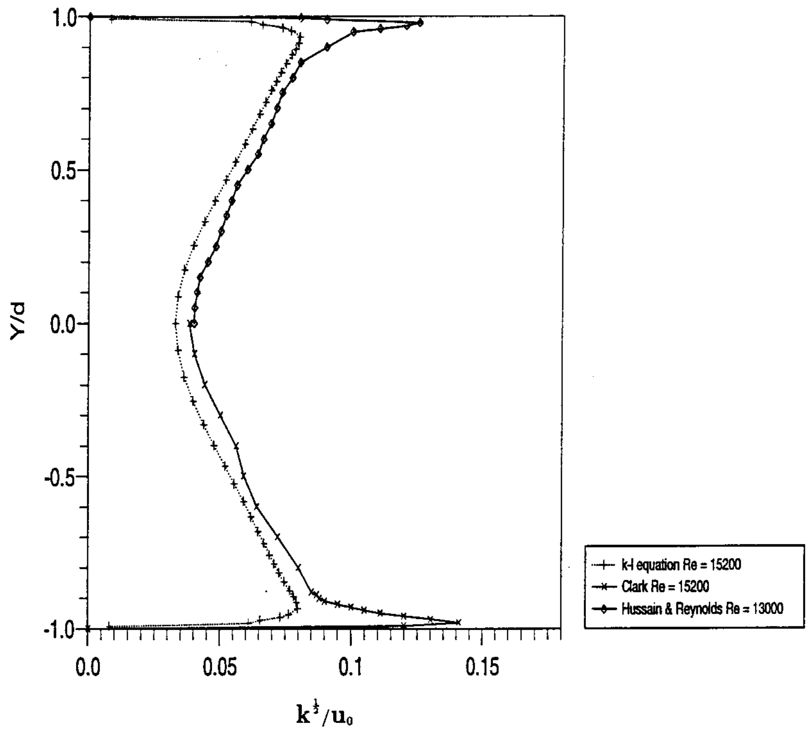


Figure 4. Turbulence kinetic energy, k, predicted by the k–l model compared with the experimental results of Laufer [28] and Hussain and Reynolds [29].

memory of the computer, whereas in the frontal method, only certain element stiffness matrices are stored at any one time in the core. The equations from a limited number of element stiffness matrices are assembled and then solved by Gaussian elimination and the contributions towards the variables that they represent are stored in the secondary memory. The elements of the mesh are traversed one by one in the order in which they are numbered. The line on the mesh dividing elements for which all equations have been assembled and those which have not is called the front and the number of unknown variables along the front is called the front width. The maximum front width encountered on the mesh is used to determine the maximum number of equations stored in the core at any one time and can be minimised by numbering the elements of the mesh in a certain manner.

The frontal method can be divided into four main steps:

1. A prefrontal operation. In this step, the last appearance of nodes is recorded and the maximum front width encountered on the mesh is computed.
2. Assembly. The element stiffness matrices are assembled into the frontal matrix, while the front width is less than the maximum front width, or until all element stiffness matrices have been assembled.
3. Elimination. The component which is to be used as a pivot is determined and the pivotal row is then normalised. The row and column of the pivotal component are then eliminated and deleted, and the pivotal equation written to disk.
4. Back-substitution. All equations previously written to disk are then reloaded in reverse sequence and subsequently a normal back-substitution process is undertaken.

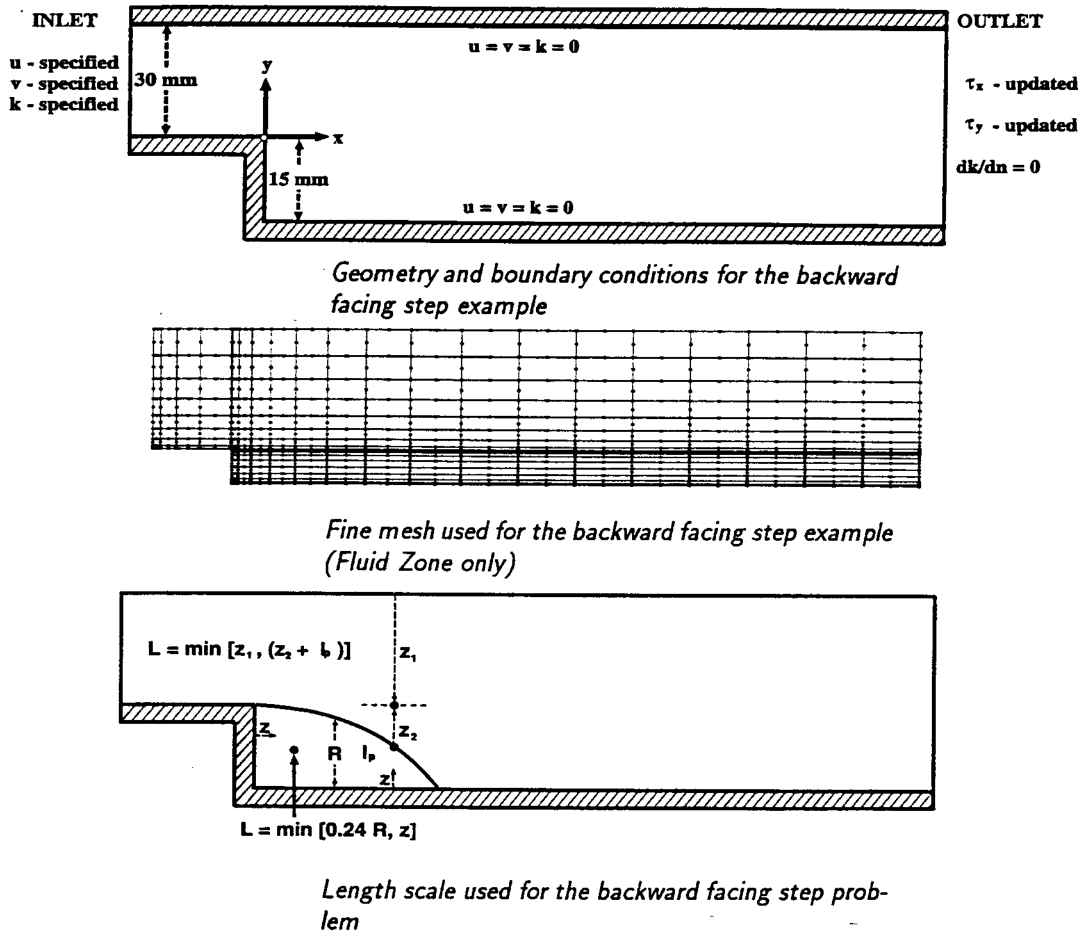

INLET
u - specified
v - specified
k - specified
30 mm
y
x
15 mm
u = v = k = 0
u = v = k = 0
OUTLET
τx - updated
τy - updated
dk/dn = 0
Geometry and boundary conditions for the backward facing step example
Fine mesh used for the backward facing step example (Fluid Zone only)
L = min [z1, (z2 + lp)]
z1
z2
z
R
lp
z
L = min [0.24 R, z]
Length scale used for the backward facing step problem

Figure 5. Geometry and typical mesh used for the example of turbulent flow over a backward facing step at $Re = 3025$.

4.2. *The iterative matrix solution techniques*

Most iterative methods work by splitting the matrix **A** as **A** = **K** − **R**. The basic method then comes down to a recurrence relationship of the form

$$\underline{\underline{\mathbf{K}}}\underline{\mathbf{u}}_i = \underline{\underline{\mathbf{R}}}\underline{\mathbf{u}}_{i-1} + \underline{\mathbf{b}}, \tag{13}$$

where $\underline{\mathbf{u}}_0$ is an initial approximation to the solution. In this paper, all vectors have be denoted by an underlined, bold character, scalars have been denoted by Greek symbols and matrices have been denoted by a double underline.

The residual is defined at iteration i to be

$$\underline{\mathbf{r}}_i = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\underline{\mathbf{u}}_i. \tag{14}$$

Using Equation (13), Equation (12) can be rewritten as

$$\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_{i-1} + \underline{\underline{\mathbf{K}}}^{-1}\,\underline{\mathbf{r}}_i = \underline{\mathbf{u}}_0 + \alpha_0\,\underline{\underline{\mathbf{K}}}^{-1}\underline{\mathbf{r}}_0 + \cdots + \alpha_{i-1}(\underline{\underline{\mathbf{K}}}^{-1}\underline{\underline{\mathbf{A}}})^{i-1}\underline{\underline{\mathbf{K}}}^{-1}\underline{\mathbf{r}}_0. \tag{15}$$

Thus, $\underline{\mathbf{u}}_i$ can be defined as a vector from the ith-dimensional *Krylov* subspace, denoted

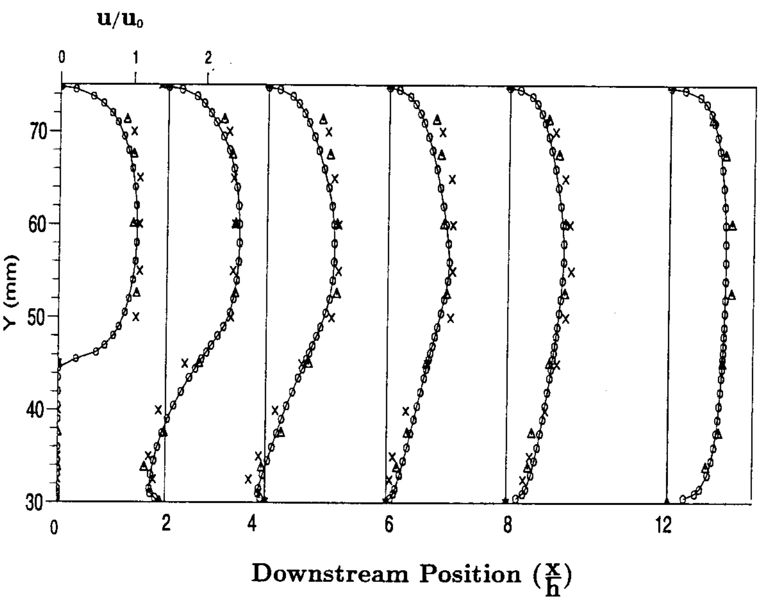


Figure 6. Horizontal velocity, u, predicted by the k–l model compared with the experimental results of Briard [30] and Atkins [31] backward facing step at $Re = 3025$.

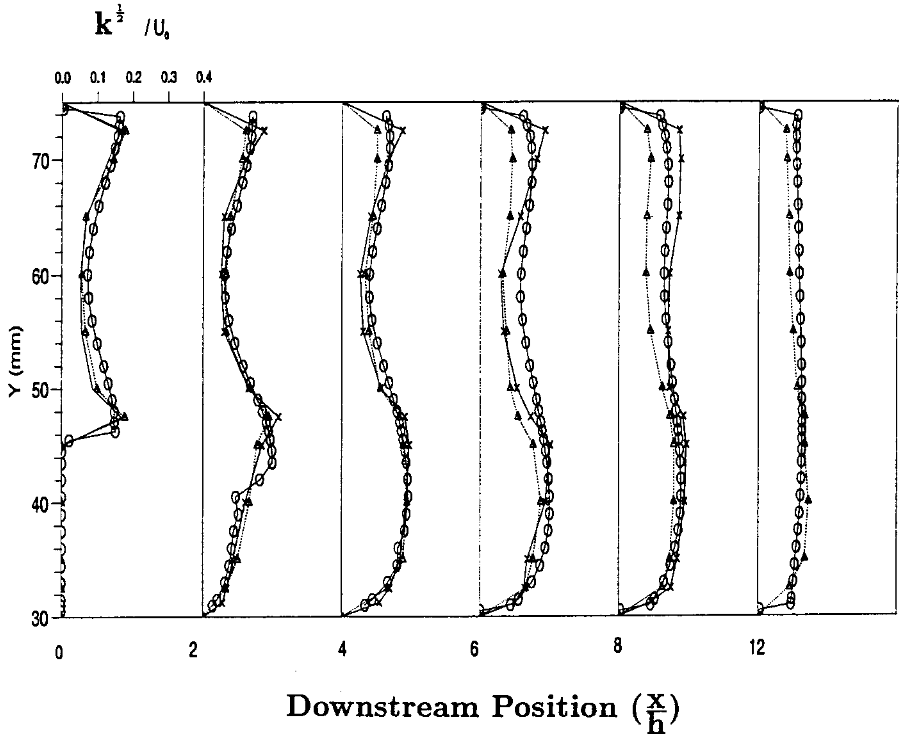


Figure 7. Turbulence kinetic energy, k, predicted by the k–l model compared with the experimental results of Briard [30] and Atkins [31] for the backward facing step at $Re = 3025$.

$K_i(\underline{\underline{K}}^{-1}\underline{\underline{A}}; \underline{\underline{K}}^{-1}\underline{r}_0)$, which is spanned by the vectors $\langle \underline{\underline{K}}^{-1}\underline{r}_0, \ldots, (\underline{\underline{K}}^{-1}\underline{\underline{A}})^{i-1}\underline{\underline{K}}^{-1}\underline{r}_0 \rangle$. For *Lancosz*-type solvers, a suitable basis is chosen for $K(\underline{\underline{A}}; \underline{r}_0)$ and Equation (11) is solved by its projection onto this subspace.

In the following subsections, the iterative solvers examined in the course of this investigation will be described in terms of the above notation.

4.2.1. The conjugate gradient (CG) algorithm. In the conjugate gradient (CG) algorithm of Hestenes and Stiefel [1], the recursion is chosen so that the A-norm of the error is minimised over the current subspace, i.e. the current $\underline{u}_i$ is chosen such that $(\underline{u}_i - \underline{u}, \underline{\underline{A}}(\underline{u}_i - \underline{u}))$ is minimal over all $\underline{u}_i$ in $K_i(\underline{\underline{A}}; \underline{r}_0)$.

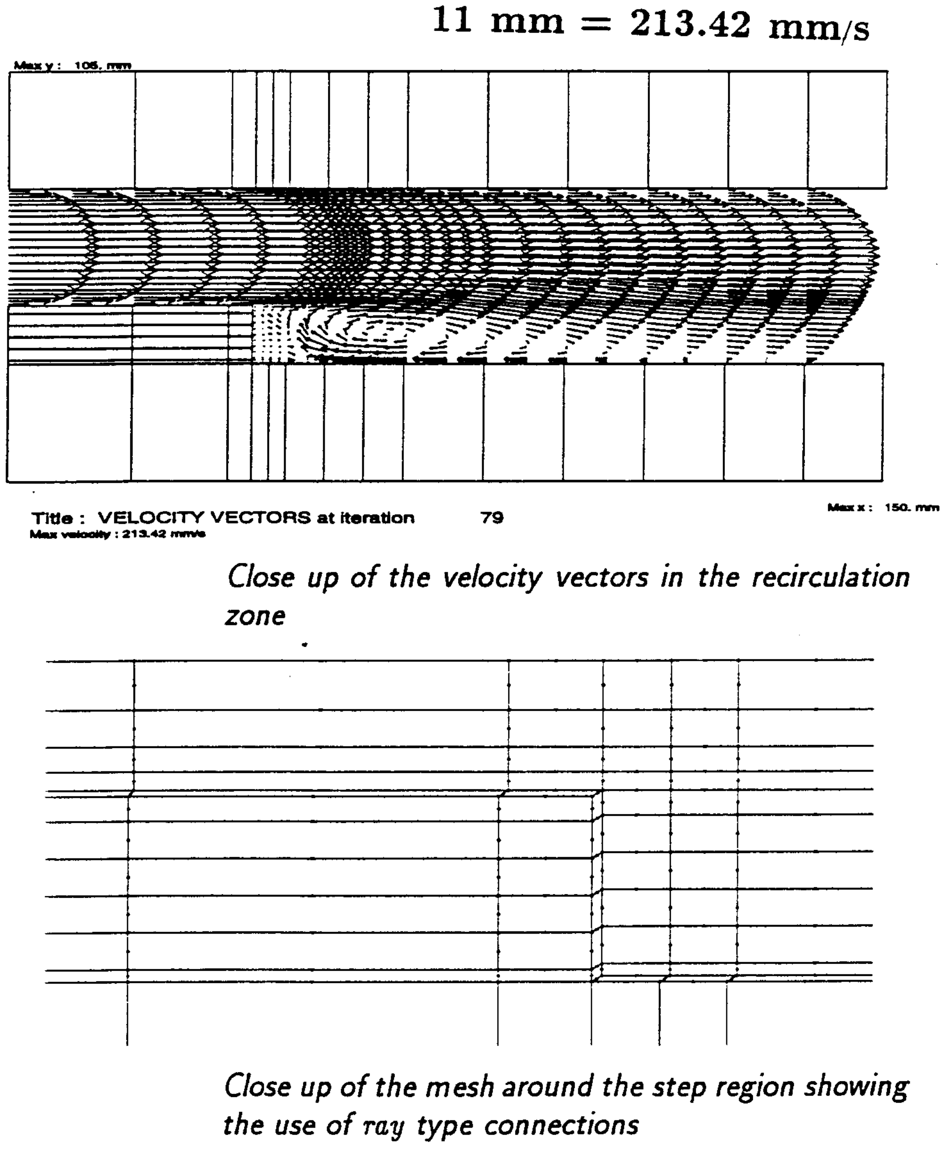

11 mm = 213.42 mm/s
Max y : 105. mm
Title : VELOCITY VECTORS at iteration 79
Max velocity : 213.42 mm/s
Max x : 150. mm
Close up of the velocity vectors in the recirculation zone
Close up of the mesh around the step region showing the use of ray type connections

Figure 8. Velocity vectors for turbulent flow over a backward facing step at $Re = 3025$.

For the CG algorithm, positive definiteness and symmetry of $\underline{\underline{\mathbf{A}}}$ is required in order to guarantee convergence. A robust way to create a converging algorithm for non-symmetric systems is obtained by applying the CG method to the normalised equations for the linear system

$$\underline{\underline{\mathbf{A}}}^T\underline{\underline{\mathbf{A}}}\underline{\mathbf{u}} = \underline{\underline{\mathbf{A}}}^T\underline{\mathbf{b}}. \tag{16}$$

This application of the CG to the normalised equations results in an algorithm known as the least squares conjugate gradient (LSCG) and can be given as

```
$\underline{\mathbf{r}}_0 = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\underline{\mathbf{u}}_0$
$\underline{\mathbf{p}}_0 = \underline{\mathbf{s}}_0 = \underline{\underline{\mathbf{A}}}^T\underline{\mathbf{r}}_0$
do for $i = 1 \ldots$
    $\underline{\mathbf{q}}_i = \underline{\underline{\mathbf{A}}}\underline{\mathbf{p}}_{i-1}$
    $\alpha_i = (\underline{\mathbf{s}}_{i-1}, \underline{\mathbf{s}}_{i-1})/(\underline{\mathbf{q}}_i, \underline{\mathbf{q}}_i)$
    $\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_{i-1} + \alpha_i\underline{\mathbf{p}}_{i-1}$
    $\underline{\mathbf{r}}_i = \underline{\mathbf{r}}_{i-1} - \alpha_i\underline{\mathbf{q}}_{i-1}$
    $\underline{\mathbf{s}}_i = \underline{\underline{\mathbf{A}}}^T\underline{\mathbf{r}}_i$
    $\beta_i = (\underline{\mathbf{s}}_i, \underline{\mathbf{s}}_i)/(\underline{\mathbf{s}}_{i-1}, \underline{\mathbf{s}}_{i-1})$
    $\underline{\mathbf{p}}_i = \underline{\mathbf{p}}_{i-1} + \beta_i\underline{\mathbf{p}}_{i-1}$
    If $\frac{\|\underline{\mathbf{r}}_i\|}{\|\underline{\mathbf{b}}_i\|} \leq \varepsilon$ then stop
End do loop
```

This algorithm can be shown to have the following features:

1. If the matrix **A** is symmetric positive definite and of size $n \times n$ then the algorithm will always converge in n steps [18].
2. The reduction of the norm of the residual is monotonic decreasing [18].

4.2.2. The biconjugate gradient (BCG) algorithm. Another *Lancosz*-type algorithm which can be applied directly to the original system of equations is the biconjugate gradient (BCG) algorithm of Fletcher [2] in which an augmented system

$$\begin{pmatrix} \underline{\underline{\mathbf{A}}} & 0 \\ 0 & \underline{\underline{\mathbf{A}}}^T \end{pmatrix}\begin{pmatrix} \underline{\mathbf{u}} \\ \tilde{\underline{\mathbf{u}}} \end{pmatrix} = \begin{pmatrix} \underline{\mathbf{b}} \\ \underline{\mathbf{b}} \end{pmatrix}, \tag{17}$$

is used to compute $\underline{\mathbf{u}}$. With this system, there comes a *biresidual* defined as

$$\tilde{\underline{\mathbf{r}}}_i = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\tilde{\underline{\mathbf{u}}}_i. \tag{18}$$

It can be shown that the residual and biresidual vector can be written as polynomials, i.e. $\underline{\mathbf{r}}_i = P_i(\underline{\underline{\mathbf{A}}})\underline{\mathbf{r}}_0$ and $\tilde{\underline{\mathbf{r}}}_i = P_i(\underline{\underline{\mathbf{A}}}^T)\underline{\mathbf{r}}_0$. Furthermore, it can be shown that these polynomials can be used as a basis for the *Krylov* subspace $K_i(\underline{\underline{\mathbf{A}}}^T; \tilde{\underline{\mathbf{r}}}_0)$.

The BCG algorithm can be written as:

```
$\underline{\mathbf{r}}_0 = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\underline{\mathbf{u}}_0$
$\underline{\tilde{\mathbf{r}}}_0 = \underline{\mathbf{r}}_0$;   $\underline{\mathbf{q}}_0 = \underline{\tilde{\mathbf{p}}}_0 = 0$;   $\rho_0 = 1$
do for $i = 1 \ldots$
    $\rho_i = (\underline{\tilde{\mathbf{r}}}_{i-1}, \underline{\mathbf{r}}_{i-1})$
    $\beta_i = \frac{\rho_i}{\rho_{i-1}}$
    $\underline{\mathbf{p}}_i = \underline{\mathbf{r}}_{i-1} + \beta_i \underline{\mathbf{p}}_{i-1}$
    $\underline{\tilde{\mathbf{p}}}_i = \underline{\tilde{\mathbf{r}}}_{i-1} + \beta_i \underline{\tilde{\mathbf{p}}}_{i-1}$
    $\sigma_i = (\underline{\tilde{\mathbf{p}}}_i, \underline{\underline{\mathbf{A}}}\underline{\mathbf{p}}_i)$
    $\alpha_i = \frac{\rho_i}{\sigma_i}$
    $\underline{\mathbf{r}}_i = \underline{\mathbf{r}}_{i-1} - \alpha_i \underline{\underline{\mathbf{A}}}\underline{\mathbf{p}}_i$
    $\underline{\tilde{\mathbf{r}}}_i = \underline{\tilde{\mathbf{r}}}_{i-1} - \alpha_i \underline{\underline{\mathbf{A}}}^T \underline{\tilde{\mathbf{p}}}_i$
    $\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_{i-1} + \alpha_i \underline{\mathbf{p}}_i$
    If $\frac{\|\underline{\mathbf{r}}_i\|}{\|\underline{\mathbf{b}}_i\|} \leq \varepsilon$   then stop
End do loop
```

The BCG algorithm does not have a monotonic decreasing residual, as in the case of LSCG, but the algorithm has proven to be stable and more efficient than the LSCG solver for a certain class of problems [17].

4.2.3. The conjugate gradient squared (CGS) algorithm. For the conjugate gradient squared (CGS) algorithm of Sonneveld [3] instead of calculating the biresidual, the residual is calculated directly from

$$\underline{\mathbf{r}}_i = P_i^2(\underline{\underline{\mathbf{A}}})\underline{\mathbf{r}}_0. \tag{19}$$

The CGS algorithm can be summarized as

```
$\underline{\mathbf{r}}_0 = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\underline{\mathbf{u}}_0$
$\underline{\tilde{\mathbf{r}}}_0 = \underline{\mathbf{r}}_0$;   $\underline{\mathbf{q}}_0 = \underline{\mathbf{p}}_0 = 0$;   $\rho_0 = 1$
do for $i = 1 \ldots$
    $\rho_i = (\underline{\tilde{\mathbf{r}}}, \underline{\mathbf{r}}_{i-1})$
    $\beta_i = \frac{\rho_i}{\rho_{i-1}}$
    $\underline{\mathbf{z}}_i = \underline{\mathbf{r}}_{i-1} + \beta_i \underline{\mathbf{q}}_{i-1}$
    $\underline{\mathbf{p}}_i = \underline{\mathbf{z}}_i + \beta_i(\underline{\mathbf{q}}_{i-1} + \beta_i \underline{\mathbf{p}}_{i-1})$
    $\underline{\mathbf{v}}_i = \underline{\underline{\mathbf{A}}}\underline{\mathbf{p}}_i$
    $\sigma_i = (\underline{\mathbf{r}}_i, \underline{\mathbf{v}}_i)$
    $\alpha_i = \frac{\rho_i}{\sigma_i}$
    $\underline{\mathbf{q}}_i = \underline{\mathbf{z}}_i - \alpha_i \underline{\mathbf{v}}_i$
    $\underline{\mathbf{r}}_i = \underline{\mathbf{r}}_{i-1} - \alpha_i \underline{\underline{\mathbf{A}}}(\underline{\mathbf{z}}_i + \underline{\mathbf{q}}_i)$
    $\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_{i-1} + \alpha_i(\underline{\mathbf{z}}_i + \underline{\mathbf{q}}_i)$
    If $\frac{\|\underline{\mathbf{r}}_i\|}{\|\underline{\mathbf{b}}_i\|} \leq \varepsilon$   then stop
End do loop
```

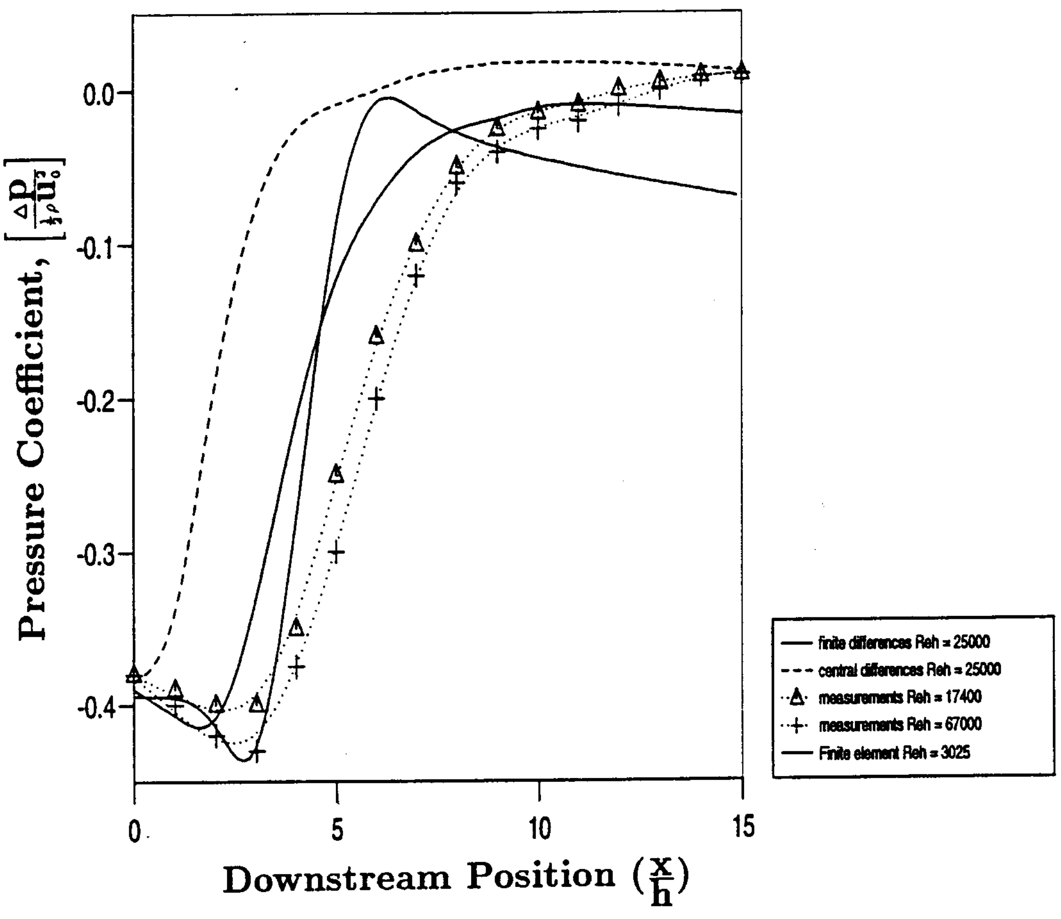


Figure 9. Pressure distribution along the lower wall compared with the experimental data of Briard and Atkins backward facing step at $Re = 3025$.

This algorithm was found by Sonneveld [3] to be more efficient than BCG and LSCG for a certain problems, but he found that it suffered from instability on convergence, particularly for problems with highly variable coefficients.

4.2.4. The biconjugate gradient squared stabilised (BCGSTAB) algorithm. Finally, for the biconjugate gradient squared stablised (BCGSTAB) algorithm of Van Der Vorst [4], a different basis is chosen, and in this case the residual can be written in the form

$$\underline{\mathbf{r}}_i = Q_i(\underline{\underline{\mathbf{A}}})P_i(\underline{\underline{\mathbf{A}}})\underline{\mathbf{r}}_0, \tag{20}$$

where $Q_i(\underline{\underline{\mathbf{A}}}) = (\underline{\underline{\mathbf{I}}} - \omega_1\underline{\underline{\mathbf{A}}}) \times (\underline{\underline{\mathbf{I}}} - \omega_2\underline{\underline{\mathbf{A}}}) \times \cdots \times (\underline{\underline{\mathbf{I}}} - \omega_i\underline{\underline{\mathbf{A}}})$.

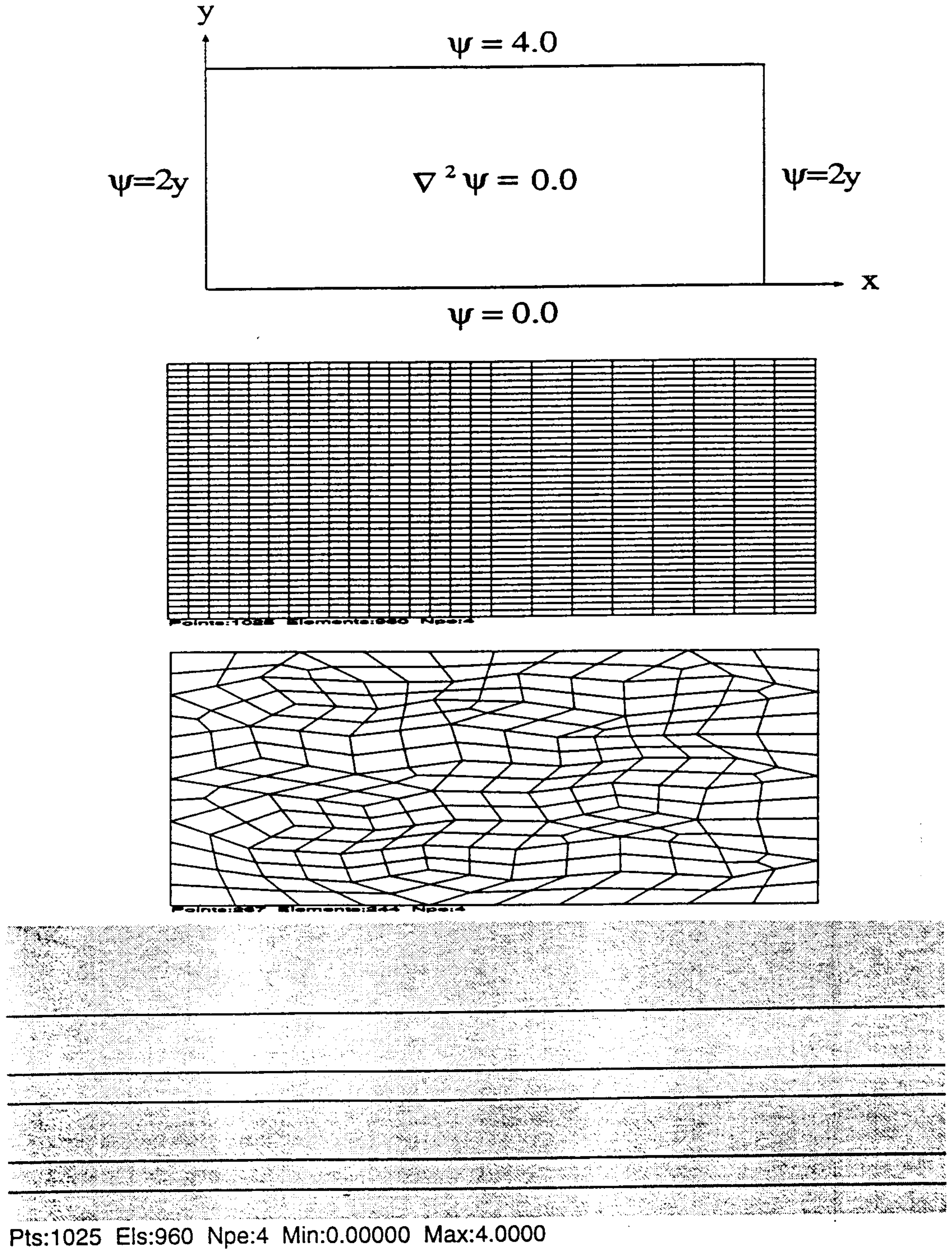


Figure 10. (a) Geometry and boundary conditions of the Stokes flow problem, (b) example of a structured quadrilateral mesh used for problem, (c) example of an unstructured quadrilateral FE mesh used for problem, (d) contours of ψ from the FE solution of Stoke's equation.

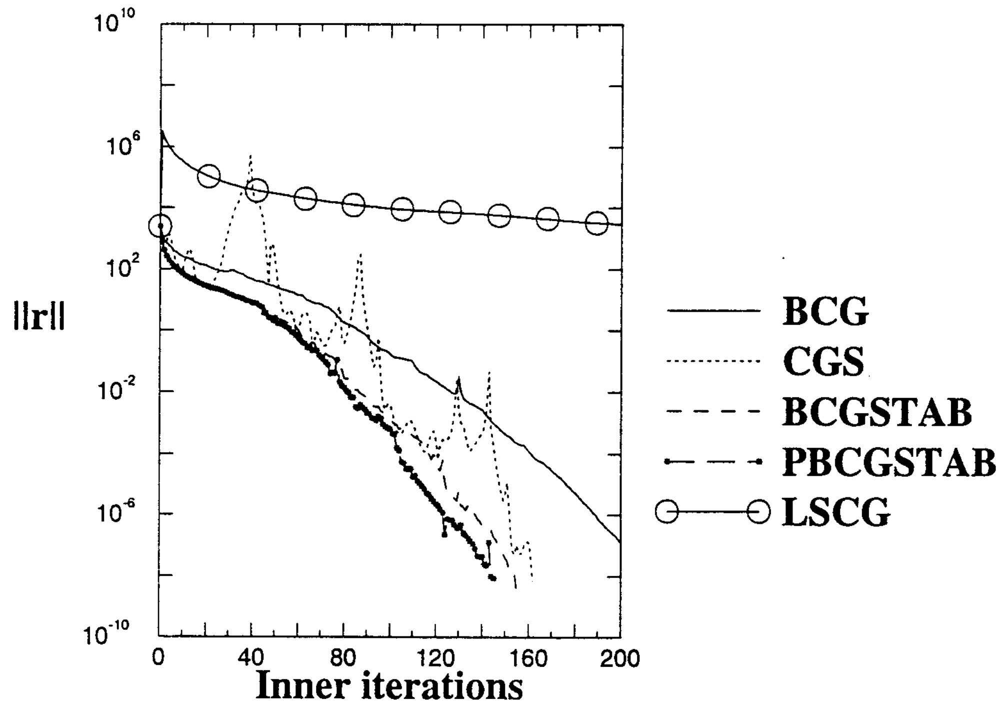


Figure 11. Reduction of the residual for all the solvers for the Stoke's problem on a given mesh.

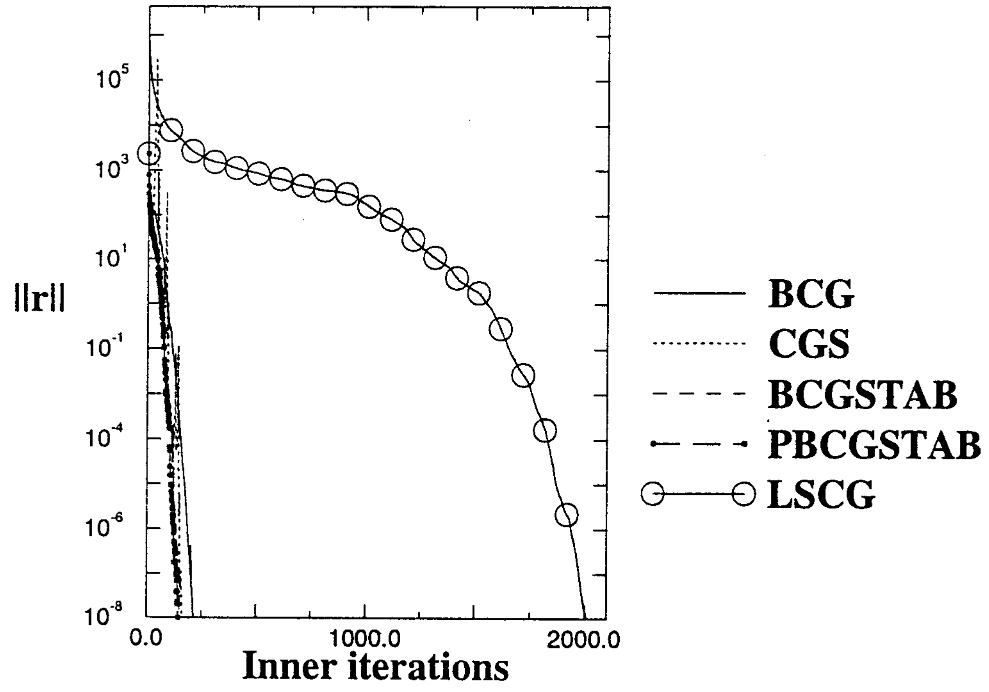


Figure 12. Same as previous figure, but showing how more inner iterations are required by the LSCG solver compared with the other solvers for the Stoke's flow problem.

The full BCGSTAB algorithm is:

```
$\underline{\mathbf{p}}_0 = \underline{\mathbf{r}}_0 = \underline{\mathbf{b}} - \underline{\underline{\mathbf{A}}}\underline{\mathbf{u}}_0$
$\underline{\mathbf{v}}_0 = \underline{\mathbf{q}}_0 = 0$
$\tilde{\omega}_0 = \alpha_0 = \beta = 1$
do for $i = 0 \ldots$
    $\tilde{\beta} = (\underline{\mathbf{p}}, \underline{\mathbf{r}}_{i-1})$
    $\omega_i = \frac{\tilde{\beta}}{\beta} \times \frac{\tilde{\omega}_{i-1}}{\alpha_{i-1}}$
    $\beta = \tilde{\beta}$
    $\underline{\mathbf{q}}_i = \underline{\mathbf{r}}_{i-1} - \tilde{\omega}_i(\underline{\mathbf{q}}_{i-1} - \alpha_{i-1}\underline{\mathbf{v}}_{i-1})$
    $\underline{\mathbf{v}}_i = \underline{\underline{\mathbf{A}}}_i\underline{\mathbf{q}}_i$
    $\tilde{\omega}_i = \frac{\tilde{\beta}}{(\underline{\mathbf{p}}, \underline{\mathbf{v}}_i)}$
    $\underline{\mathbf{s}} = \underline{\mathbf{r}}_{i-1} - \tilde{\omega}_i\underline{\mathbf{v}}_i$
    $\underline{\mathbf{t}} = \underline{\underline{\mathbf{A}}}\underline{\mathbf{s}}$
    $\alpha_i = \frac{(\underline{\mathbf{t}}, \underline{\mathbf{s}})}{(\underline{\mathbf{t}}, \underline{\mathbf{t}})}$
    $\underline{\mathbf{u}}_i = \underline{\mathbf{u}}_{i-1} + \alpha_i\underline{\mathbf{s}} + \tilde{\omega}_i\underline{\mathbf{q}}_i$
    $\underline{\mathbf{r}}_i = \underline{\mathbf{s}} - \alpha_i\underline{\mathbf{t}}$
    If $\frac{\|\underline{\mathbf{r}}_i\|}{\|\underline{\mathbf{b}}_i\|} \leq \varepsilon$ then stop
End do loop
```

4.3. *The multigrid method*

The basic idea behind the multigrid method can be understood from the point of view of error wavelength analysis. If the error in an approximate solution can be considered as a wavelength of long, medium or short range, then the use of Fourier analysis allows the verification that short wavelength modes of the error decay faster on finer meshes than a longer wavelength and *vice versa*, i.e. that the medium and long ranges decay faster on coarser meshes. Using this fact, it is possible to construct a series of meshes which are interconnected and solve across them to decrease the various ranges of wavelength across this hierarchy of meshes in a way which turn out to be very efficient.

Before describing the algorithm used in this investigation, it is necessary to introduce some new notation. The notation of Wesseling [14] has been adopted in which $\{G^k\colon k = 1, K\}$ be a sequence of increasingly finer meshes with $k = 1$ being the coarsest level and $k = K$ corresponding to the finest level. Furthermore, let P and R be intermesh transfer operators, where P stands for *Prolongation* and R stands for *Restriction*, such that P interpolates the vector on a coarse mesh on to a fine mesh, and R gathers values from a finer mesh onto a coarser mesh.

If the discretised problem at each level of mesh is denoted by

$$\mathbf{L}^k(\mathbf{u}^k) = \mathbf{b}^k \tag{21}$$

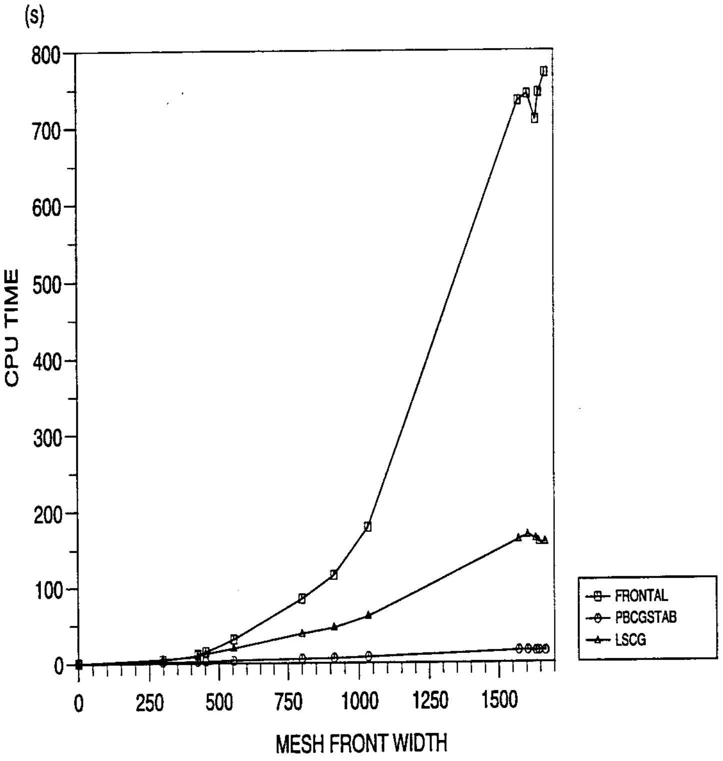


Figure 13. Comparison of CPU time on a DEC 3000 Alpha workstation vs. mesh front width for Stoke's problem using the frontal solver, the LSCG solver and preconditioned BCGSTAG of Van Der Vorst.

Then the basic recursive algorithm is comprised of two main routines and can be written as given by Wesseling [14]:

```
MG($u^K$, $u_0^K$)
1.0 $f^K = b^K$
2.0 do $i = 1$, ntg
3.0 MG01($u^K$, $u_0^K$, $f^K$, . . . , $K$)
4.0 $u_0^K = u^K$
5.0 end do
```

In MG00, the routine MG01 is given by:

```
3.00 MG01 ($u^k$, $u_0^k$, $f^k$, . . . , $k$)
3.01 if $k = 1$ then
3.02 $S(u^k, u_0^k, f^k, v, k)$
3.03 else
3.04 $S(u^k, u_0^k, f^k, v, k)$
3.05 $r^k = f^k - L^k(u^k)$
3.06 Choose $u_0^{k-1}$ and $s^{k-1}$
3.07 $f^{k-1} = L^{k-1}(u_0^{k-1}) + s^{k-1}R_k^{k-1}r^k$
3.08 MG01($u^{k-1}$, $u_0^{k-1}$, $f^{k-1}$, . . . , $k-1$)
3.09 $u^k = u^k + s^{k-1}P_k^{k-1}(u^{k-1} - u_0^{k-1})$
3.10 $S(u^k, u_0^k, f^k, v, k)$
3.11 Endif
```

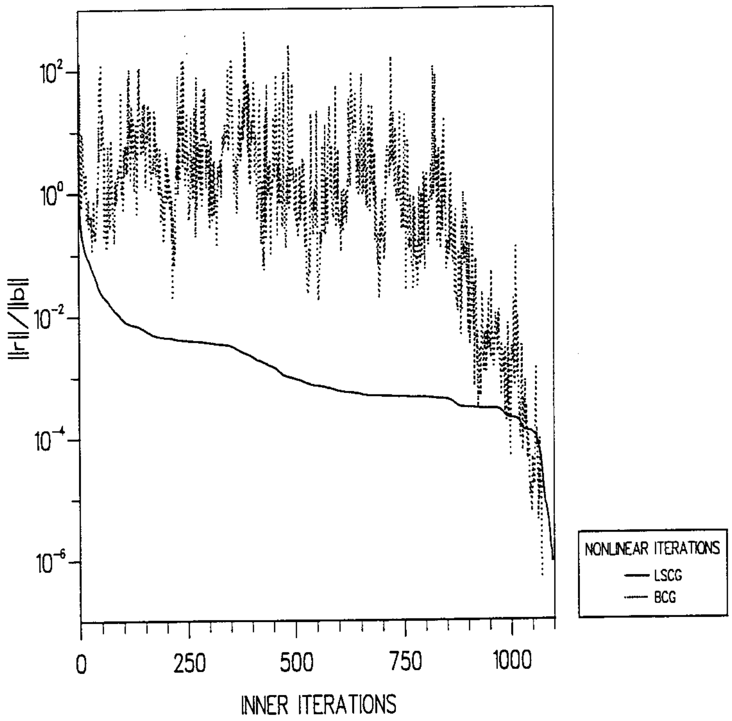


Figure 14. Reduction of the norm of the residual with BCG and LSCG for the turbulent flow between parallel plates test case on a given mesh.

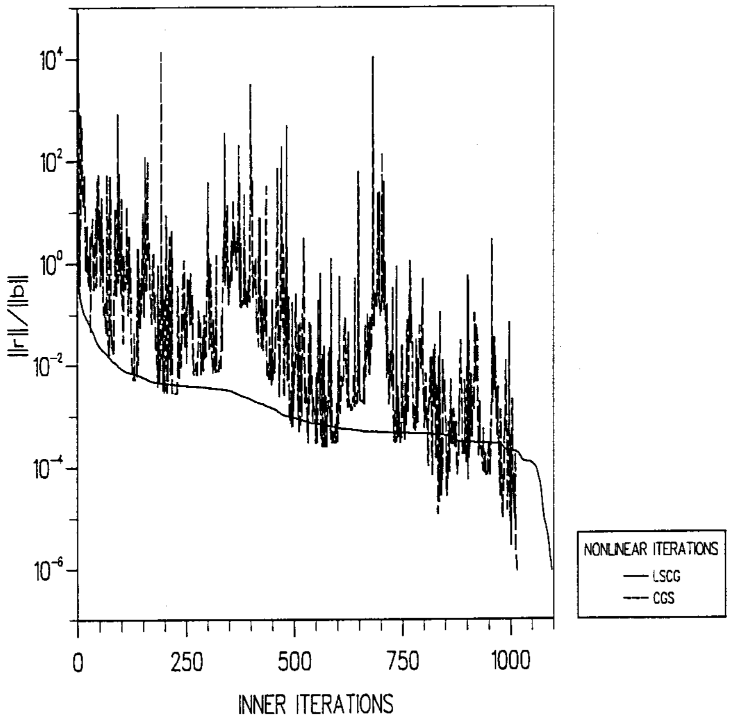


Figure 15. Reduction of the norm of the residual with CGS and LSCG for the turbulent flow between parallel plates test case on a given mesh.

In MG00, the solution procedure commences on the finest mesh, proceeds to move up to the coarse mesh and, finally, ends up back on the fine grid. Due to this movement, this algorithm is said to use a *V-cycling* schedule. The number of times that the V-cycle is performed is pre-set by the user as *ntg*.

In MG01, K is the finest level of mesh, u_0 is an initial estimate to the solution at the level of mesh indicated by its superscript and s^{k-1} is a relaxation parameter. In this investigation, its value was taken as 1.0 so that the algorithm corresponded with Brandt's original FAS (full approximation scheme) algorithm [5].

A natural selection for the prolongation operator, P, in the FEM is to use the element shape functions. For the restriction operator, R, the method of *simple injection*, which consists of using values at overlapping nodes which correspond to the finer mesh. As described in Section 3, the type of element used is the nine-noded quadrilateral element. An element centred subdivision was used to create the mesh hierarchy, as shown in Figure 1.

Associated with each level of mesh is a *smoothing* algorithm (matrix solving technique), $\mathbf{S}(\mathbf{u}, \mathbf{u}_0, \mathbf{f}, \nu, \mathrm{k})$, where $\mathbf{u}_0$ is an initial estimate, $\mathbf{f}$ is the right-hand side vector, $\mathbf{u}$ is the vector of unknowns, ν is the maximum number of inner iterations of the solver to use at the current level of mesh, k. The choice of solver for $\mathbf{S}$ was taken to be one of the iterative solvers described in the previous section, i.e. LSCG, BCG, CGS or BCGSTAB.

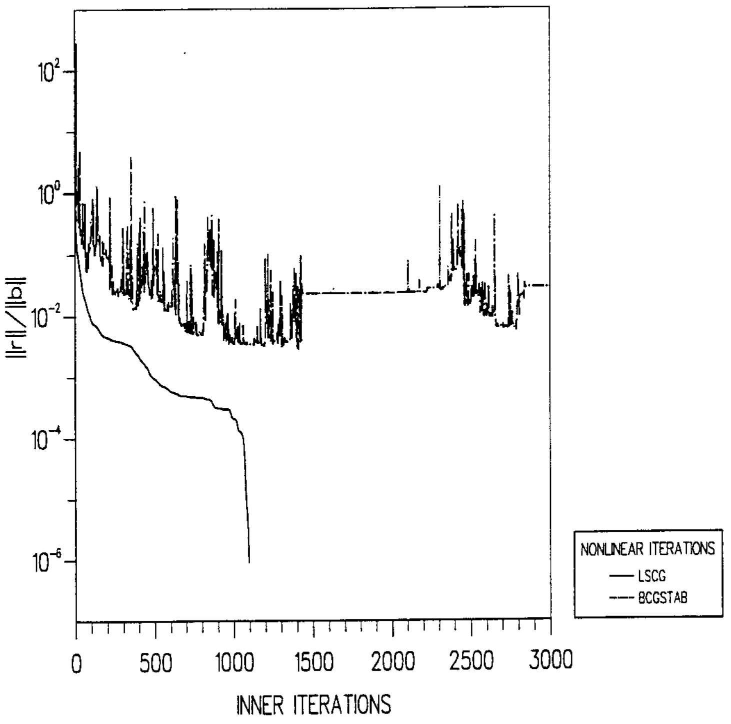


Figure 16. Reduction of the norm of the residual with BCGSTAB and LSCG for the turbulent flow between parallel plates test case on a given mesh.

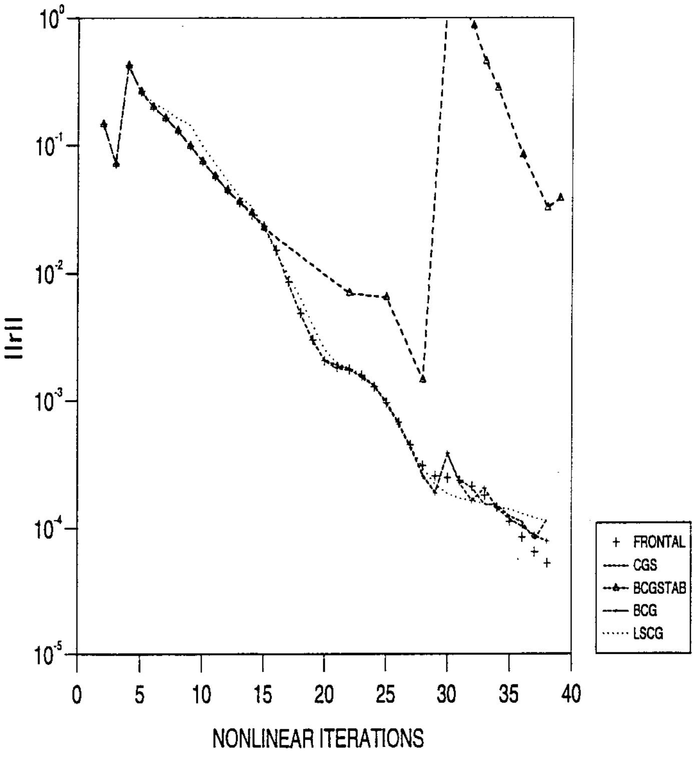


Figure 17. Reduction of the norm of the residual vs. non-linear iterations for the turbulent flow between parallel plates test case on a given mesh.

5. NUMERICAL EXAMPLES AND ASSESSMENT

5.1. Comparison with experiment using the frontal solver

5.1.1. Turbulent flow between parallel plates. The geometry and boundary conditions used for this example are shown in Figure 2(a). The Reynolds numbers (*Re*) examined ranged from approximately 15000 to 30000. An example of the type of mesh used to solve the equations is shown in Figure 2(b). In Figure 3, the horizontal velocity, *u*, can be seen to accurately predict the experimental results of Laufer [28] and Hussain–Reynolds [29]. Figure 4 shows the comparison of the turbulence kinetic energy, *k*, to the experimental results.

5.1.2. Turbulent flow over a backward facing step for Re = 3025. The geometry and boundary conditions used for this example are shown in Figure 5(a). An example of the type of mesh used to solve the equations is shown in Figure 5(b). In Figure 6, the horizontal velocity, *u*, shows the comparison with the experimental results of Briard [30] and Atkins [31]. Figure 7 shows the comparison of the turbulence kinetic energy, *k*, from the same experimental results. Figure 8 shows the velocity vectors for this example, and Figure 9 shows a comparison of the pressure distributions along the lower wall.

5.2. Assessment of the iterative solvers

5.2.1. Stoke's equation in a rectangular domain. A standard finite element solution of Stoke's equation was coded to initialise the implementation of the iterative solvers described in the previous section. These were then tested upon the geometry shown in Figure 10(a), with the boundary conditions given in the same figure. The solution procedure in this example allowed structured and unstructured meshes to be used as shown in Figure 10(b) and (c), respectively. The values of the stream function, ψ, are shown as contours in Figure 10(d). For a given mesh, the reduction of the norm of the residual was used as an indicator of the rate of convergence, as shown in Figure 11. Simple preconditioning was also implemented in the form of diagonal preconditioning. It was found that while there was not much difference between CGS, BCGSTAB and BCG, all of these were found to be considerably faster than LSCG (see Figure 12). Preconditioned BCGSTAB was found to be the most efficient solver for this problem followed by a 10% difference with unpreconditioned BCGSTAB. Considerable time saving was found by using the iterative solvers over the frontal solver (see Figure 13).

5.2.2. Turbulent flow between parallel plates. A series of meshes were constructed with an increasing number of elements by subdividing each element into four new elements. The iterative solvers were then used to solve the matrix equations on each of these meshes, the results were correlated and CPU solution times in seconds were obtained on a DEC Alpha workstation using 96 Mbytes of main memory.

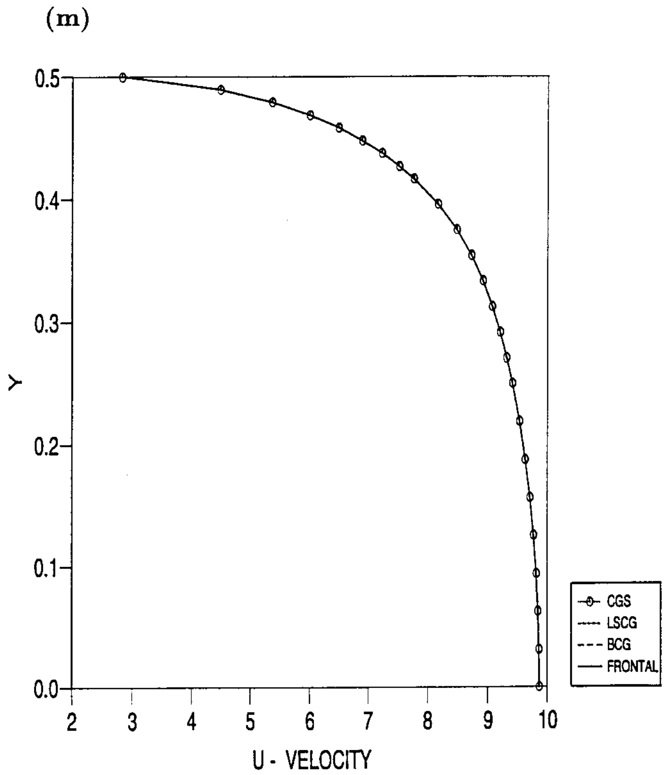


Figure 18. *u* velocity distribution for the various converging solvers.

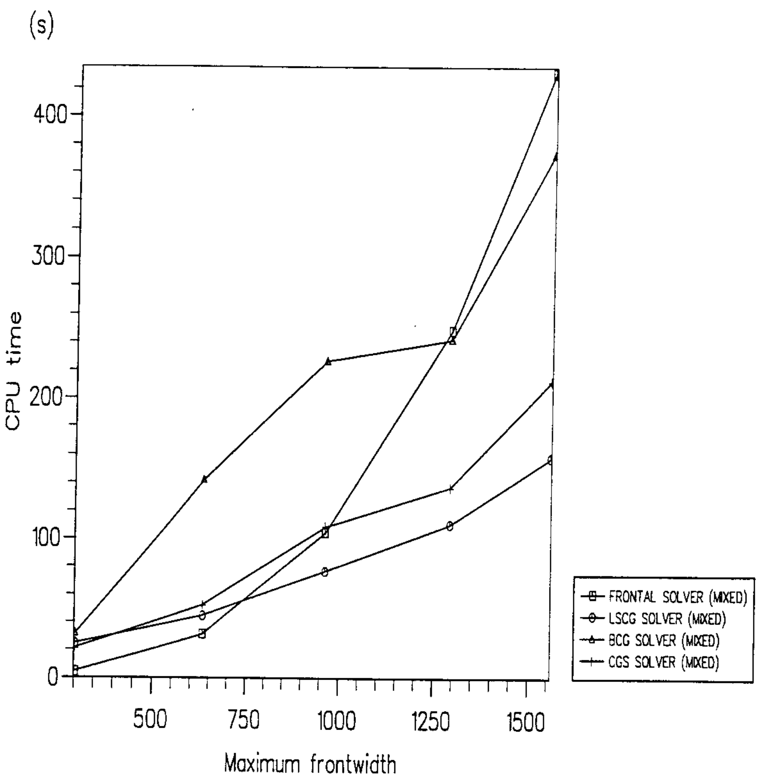


Figure 19. CPU time per non-linear iteration vs. front width for the various converging solvers.

Figure 14 compares the reduction of the norm of the residual vector, $\|\underline{\mathbf{r}}_0\|/\|\underline{\mathbf{b}}\|$, during a given non-linear iteration using the LSCG solver with the BCG solver. Figure 15 shows the comparison of LSCG with the CGS, and Figure 16 shows the comparison of LSCG with BCGSTAB. The inner tolerance of the iterative solvers was set to 10^{-6}.

The monotonic decreasing path of the residual with LSCG (which can be predicted from theory) is clear in these figures. Unsurprising as well are the erratic paths of descent of BCG and CGS in comparison with LSCG. However, a surprising result was that an almost equal number of iterations were required by these solvers than the LSCG solver at each non-linear iteration. Another surprising result was that the BCGSTAB solver failed to reduce the residual below a 10^{-2} drop, thus causing the divergence of the non-linear solution.

The drop in the norm of the residual versus the non-linear iterations is shown in Figure 17, demonstrating that convergence to the desired tolerance occurs at around the 10^{-4} mark. Also highlighted is the divergence occurring when BCGSTAB was used. Figure 18 shows that there is no difference in the *u* velocity profile when the different solvers are used. Figure 19 shows CPU time used by the different solvers plotted against the maximum mesh front width. The following points should be noted:

1. The frontal solver shows its characteristic (but undesirable) quadratic increase in CPU time with the mesh front width,

2. The iterative solvers show a flatter CPU growth rate,
3. The LSCG solver is the most efficient solver in this case, allowing up to 64% time saving over the frontal solver at the largest front width,
4. CGS allowed up to a 52% saving over the frontal solver,
5. BCG allowed up to a 16% saving over the frontal solver.

5.3. Assessment of the multigrid solvers

Figure 20 shows one of the three level hierarchy of meshes used to solve the turbulent flow in a channel test problem with the multigrid algorithm. The multigrid algorithm implemented used a V-cycling schedule with the FAS of Brandt [5]. The solver used at each level of grid was the LSCG solver. Figure 21 shows how the rate of convergence on a single grid can be improved by using a second and third level of mesh multigrid solver. In this figure, the single grid solver (1 GRID SG in legend) uses 100 inner iterations at each non-linear iteration, and the result is that these are not enough to guarantee convergence. However, when the multigrid coarse grid correction is added to the solution procedure, and 100 iterations are performed on the second level of grid, maintaining the same number of iterations on the first level of grid, it is possible to obtain convergence. Furthermore, if a third level is added to the multigrid procedure, with 500 and then 1000 iterations at this level, and still using the same number of iterations at the other levels, then the effect of the multigrid is to further improve the convergence rate. In terms of the CPU time involved, Figure 22 shows how the second

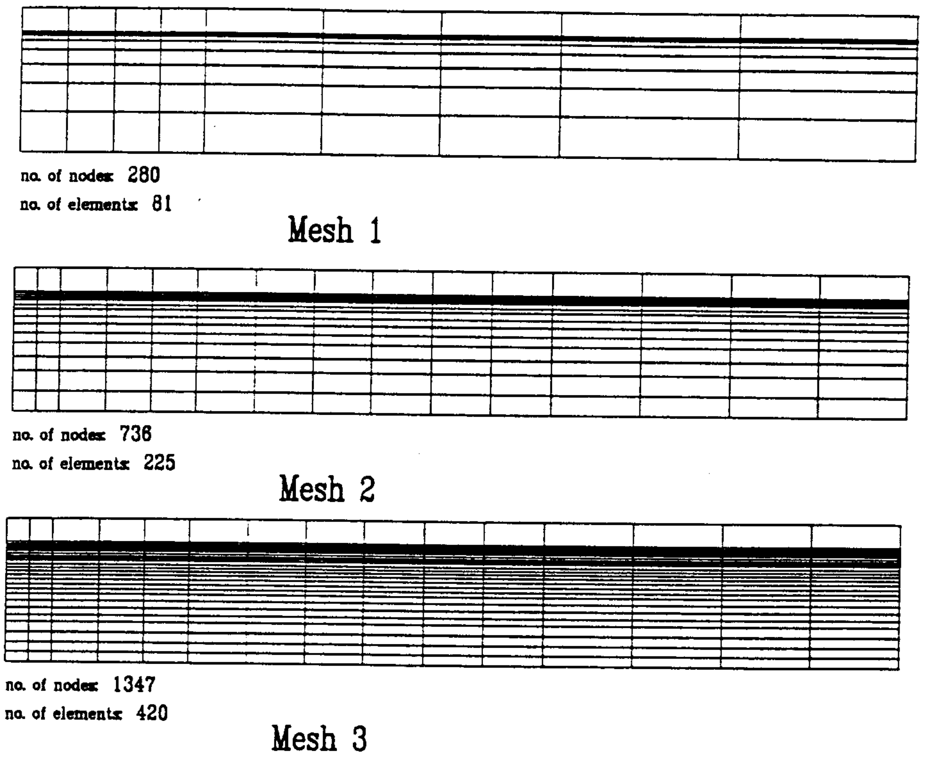


Figure 20. Typical hierarchy of meshes used in the multigrid solution.

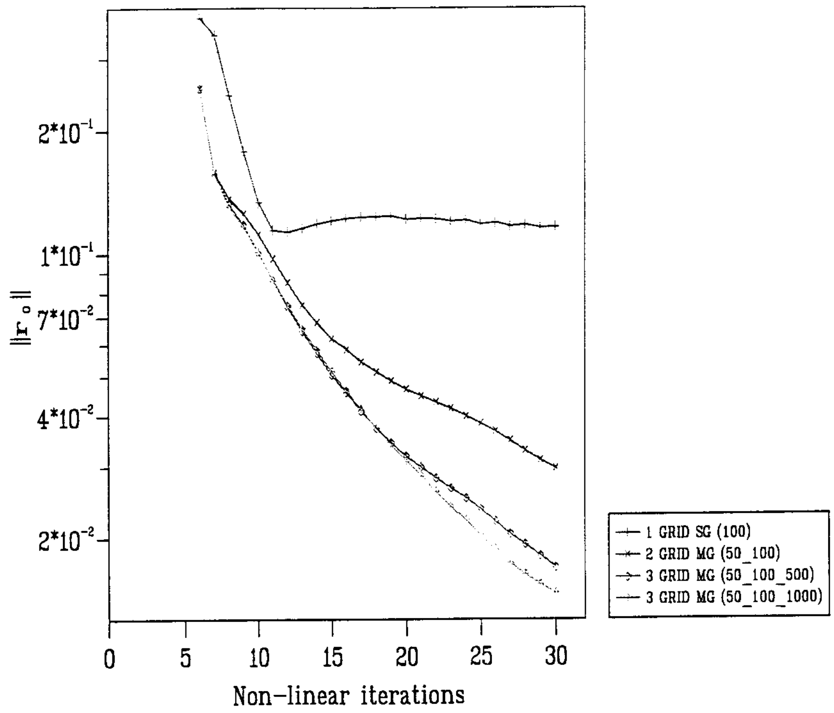


Figure 21. Performance of single grid vs. 2- and 3-mesh multigrid solvers as a function of the residual and the number of non-linear iterations.

level of mesh multigrid algorithm can be used to half the CPU time of the single grid solution and that the third level of mesh algorithm can be used to decrease this by a further 7%.

6. CONCLUSIONS

It has been shown that iterative methods of the type used here can greatly reduce the solution time of the matrix equations resulting form the finite element solution of turbulent fluid flow when compared with the commonly adopted frontal solution technique. It has also been demonstrated that the multigrid method can further accelerate the solution time of these iterative methods by up to 57%.

Further research is needed to find ways of creating a more 'user-friendly' application of the multigrid method than the one developed during the course of this investigation, possibly via the inclusion of automatic mesh generation techniques, subsequently leading to adaptive procedures.

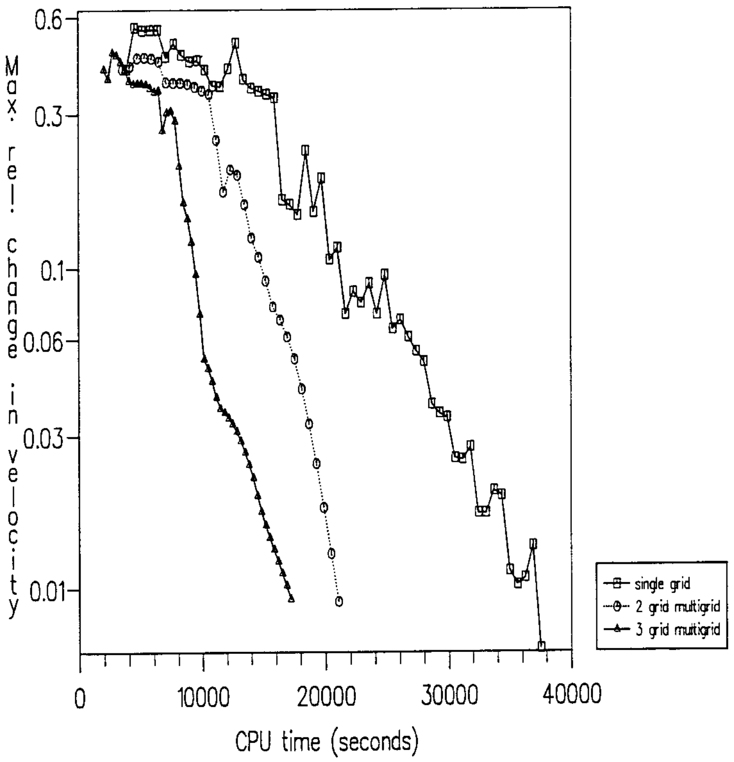


Figure 22. Performance of single grid vs. 2- and 3-mesh multigrid solvers as a function of the maximum relative change in the velocity and the CPU time in seconds on the DEC 3000 Alpha.

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support given by ABB Gas Turbine Development, without which this research would not have been possible. In particular the authors thank Professor Cedric Taylor for supervising the work and at ABB to Drs Harasgama, Xia and Fitzimmons.

REFERENCES

1. M. Hestenes and E. Stiefel, 'Methods of conjugate gradients for solving linear systems of equations', *J. Res. Nat. Bur. Stan.*, **49**, 409–435 (1952).
2. R. Fletcher, 'Conjugate gradient methods for indefinite systems', in *Lecture Notes in Mathematics*, Springer, Berlin, 1976, pp. 73–89.
3. P. Sonneveld, 'CGS, a fast Lancosz-type solver for non-symmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **10**, 36–52 (1976).
4. H. Van Der Vorst, 'BCGSTAB: a fast and smoothly converging variant of BCG for the solution of non-symmetric linear systems' *SIAM. J. Sci. Stat. Comput.*, **18**, 631–644 (1992).
5. A. Brandt, 'Diagonally inverted lower-upper factored implicit multigrid scheme for the 3D Navier–Stokes equations', *AIAA J.*, **28**, 1642–1649 (1977).
6. P. Hood, 'A finite element solution of the Navier–Stokes equations for incompressible contained flow', *M.Sc. Thesis*, University of Wales, Swansea, 1970.
7. A. Chorin, 'Numerical solution of the Navier–Stokes equations', *Math. Comput.*, **23**, 341–354 (1968).
8. S. Tuan and M. Olson, 'Primitive variables versus stream function finite element solutions of the Navier–Stokes equations', *Proc. 2nd Int. Conf. Finite Elements in Flow Problems*, Italy, 1976.

9. G. Schneider and G. Raithby, 'Finite element analysis of incompressible fluid flow incorporating equal order pressure and velocity interpolation', C. Taylor and K. Morgan (eds), *Computer Methods in Fluids*, Pentech Press, 1979, pp. 49–84.
10. S. Patankar *et al*., 'A calculation procedure for heat, mass and momentum transfer in 3D parabolic flows', *Int. J. Heat Mass Transf*., **15**, 1784–1806 (1972).
11. P. Gresho *et al*., 'On the theory of semi-implicit projection methods for incompressible flow via a FEM that also introduces a nearly consistent mass matrix: Part II Implementation', *Int. j. numer. methods fluids*, **11**, 621–660 (1990).
12. C. Taylor and P. Hood, 'Navier-Stokes equations using mixed interpolation', *Proc. Int. Symp. on FEM in Flow Problems*, January 7–11, University of Wales, Swansea, 1974, pp. 121–132.
13. O. Zienkiewicz and R. Taylor, *The Finite Element Method, Basic Formulation and Linear Problems*, vol. 1, McGraw-Hill, New York, 1989, pp. 324–327.
14. P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, New York, 1991.
15. B. Irons, 'A frontal solution program for finite element analysis', *Int. j. numer. methods eng*., **10**, (1970).
16. N. Lavery, 'Iterative and multigrid methods for the FE solution of turbulent and incompressible flow', *Ph.D. Thesis*, University of Wales, Swansea, 1996.
17. D. Howard, 'Numerical techniques for the simulation of three-dimensional swirling flow', *Ph.D. Thesis*, University of Wales, Swansea, 1988.
18. C. Johnson, *Numerical Solution of Partial Differential Equations by the FEM*, Cambridge University Press, Cambridge, 1990.
19. R. Fedorenko, 'The speed of convergence of the one iterative process', *USSR Comput. Math. Math. Phys*., **4**, 227–235 (1964).
20. A. Brandt, 'Multigrid techniques: 1984 guide with applications to fluid dynamics', *GMD Studien No. 85*, Gesellschaft fur Mathematik und Dynamics, St. Augustin, Germany, 1984.
21. J. Yokota, 'Diagonally inverted lower-upper factored implicit multigrid scheme for the 3D Navier–Stokes equations', *AIAA J*., **28**, 1642–1649 (1990).
22. X. Bai and L. Fuchs, 'Multigrid computation of turbulent reacting flows with thermal radiation', *Proc. Int. Conf. Laminar and Turbulent Flow*, University of Wales, Swansea, 1993, pp. 1337–1348.
23. R. Lonsdale, 'An algebraic multigrid solver for the Navier–Stokes equations on unstructured meshes', *Int. j. numer. methods heat and fluid flow*, referred (1991).
24. R. Webster, 'An algebraic multigrid solver for the Navier–Stokes equations', *Int. j. numer. methods fluids*, referred (1995).
25. J. Xia and C. Taylor, 'A multigrid solution based on the FEM for the Navier–Stokes equations', *Eng. Comput*., **9**, 469–475 (1992).
26. S. Wille, 'A non-linear adapted tri-tree multigrid solver for the finite element formulation o the Navier–Stokes equations', *Int. j. numer. methods fluids*, to be published (1995).
27. J. Rance, 'Flow and heat transfer in rotating channels', *Ph.D. Thesis*, University of Wales, Swansea, 1989.
28. J. Laufer, 'Investigation of turbulent flow in a two-Dimensional channel', *NACA report 1053*, 1951.
29. A. Hussain and W. Reynolds, 'Measurements in fully developed channel flow', *Trans. ASME, J. Fluids Eng*., **97**, 568–580 (1975).
30. M. Denham, P. Briard and M. Patric, 'A directionally sensitive laser anenometer for velocity measurements in highly turbulent flows', *J. Phys. Eng. Sci. Instrum*., **8**, 681–683 (1975).
31. D. Atkins *et al*., 'Numerical prediction of separated flows', *Int. j. numer. methods eng*., **15**, 129–144 (1980).