INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **26**: 959–975 (1998)

NODAL OPERATOR SPLITTING ADAPTIVE FINITE ELEMENT ALGORITHMS FOR THE NAVIER–STOKES EQUATIONS

S.Ø. WILLE*

Faculty of Engineering, Oslo College, Cort Adelersgate 30, N-0254 Oslo, Norway

SUMMARY

Solution algorithms for solving the Navier–Stokes equations without storing equation matrices are developed. The algorithms operate on a nodal basis, where the finite element information is stored as the co-ordinates of the nodes and the nodes in each element. Temporary storage is needed, such as the search vectors, correction vectors and right hand side vectors in the conjugate gradient algorithms which are limited to one-dimensional vectors. The nodal solution algorithms consist of splitting the Navier–Stokes equations into equation systems which are solved sequencially. In the pressure split algorithm, the velocities are found from the diffusion–convection equation and the pressure is computed from these velocities. The computed velocities are then corrected with the pressure gradient. In the velocity–pressure split algorithm, a velocity approximation is first found from the diffusion equation. This velocity is corrected by solving the convection equation. The pressure is then found from these velocities. Finally, the velocities are corrected by the pressure gradient. The nodal algorithms are compared by solving the original Navier–Stokes equations. The pressure split and velocity–pressure split equation systems are solved using ILU preconditioned conjugate gradient methods where the equation matrices are stored, and by using diagonal preconditioned conjugate gradient methods without storing the equation matrices.

KEY WORDS: grid adaption; grid iterations; iterative solvers; operator splitting; finite elements

1. INTRODUCTION

The advantage of explicit time marching schemes for solving partial differential equations is that the prediction of the future solution from earlier solutions is done without having to solve a system of equations. Therefore, computer memory for storing the equation matrices is not required and it is possible to solve larger problems with significantly more degrees of freedom.

Explicit time marching schemes are extensively used for hyperbolic equation systems, while implicit time schemes are most frequently used for solving parabolic equation systems. Implicit algorithms offer a higher degree of stability, which is required when diffusion is included in the flow equations. However, implicit algorithms usually require more work and more storage, as an equation system has to be solved.

In the present work, explicit and implicit time marching schemes are combined to solve the Navier–Stokes equations. The conjugate gradient method is used to solve the parabolic diffusion equation and the Laplace equation is used for the pressure implicitly. The forward Euler method is used to solve the convection equation.

* Correspondence to: Faculty of Engineering, Oslo College, Cort Adelersgate 30, N-0254 Oslo, Norway; E-mail: s.o.wille@iu.hioslo.no;www: www.iu.hioslo.no/ ~ sowille

CCC 0271–2091/98/080959–17$17.50

Received May 1997
Revised July 1997

The most complicating factor in solving the Navier–Stokes equations is how to handle the pressure term. A variety of approaches have been investigated. Penalty methods have proved to be unstable and produce pressure checkerboard, but the mixed interpolation method [1] appears to be the most stable, robust and accurate approach [2].

An operator splitting algorithm presented by Goda [3,4] consists of splitting the equations. The velocities are first found by using the pressure obtained at the previous step, then the new pressure is found from a modified Poisson equation, and in the last step, the velocities are corrected by the corresponding new pressure gradient. The numerical schemes applied to the splitting algorithm have been both explicit and implicit in time. Utnes [5,6] used a second-order explicit Runge–Kutte method for the time derivatives of the velocities and solved the modified Poisson equation by a direct solver. Ruas [7] and Goldberg [8] also implicitly solved for the velocities and used old velocities in forming the non-linear matrix coefficients. The equation systems were solved iteratively by conjugate gradient methods. An advantage of the splitting algorithm, is that equal order, linear basis function can be applied to approximate both pressure and velocity.

The new algorithm developed in the present work contains additional operator splitting. The velocity equation is split in a diffusion equation and a convection equation. In this algorithm, four equation systems are solved sequencially. These equations are the diffusion equation, the convection equation, the pressure equation and the mass equation. The diffusion equation and the pressure equation are both symmetric and positive definite Laplace equations, while the convection equation and the mass equation are pure hyperbolic equations.

The characteristics of the different algorithms, which are compared with respect to efficiency and stability, are summarized below.

(i) Coupled algorithm; assembled equation system; coupled node fill in ILU preconditioned conjugate gradient solver.
(ii) Pressure split algorithm; assembled equation system; ILU preconditioned conjugate gradient solver.
(iii) Pressure split algorithm; nodal equation system; diagonal preconditioned conjugate gradient solver.
(iv) Velocity–pressure split algorithm; assembled equation system; ILU preconditioned conjugate gradient solver.
(v) Velocity–pressure split algorithm; nodal equation system; diagonal preconditioned conjugate gradient solver.

Quad-tree grid generation has been applied successfully by Kallinderis [9] and Greaves [10–12]. These authors used the hierarchic structure of the quad tree for recursive subdivisions. As the quad-tree grid has achieved a satisfactory refined grid, a triangular mesh is constructed from the rectangular elements in the quad-tree. In the present work the tri-tree grid generation method is successfully applied [13,18].

2. TRI-TREE ADAPTIVE GRID GENERATION

The tri-tree grid generation is based on triangles in two dimensions and tetrahedra in three dimensions [13]. The refinements of triangles are shown in Figure 1. The information of the tree structures is contained in records associated with each triangle as shown in Figure 2. These records contain the level of refinement, the node numbers and pointers to the divisions associated with the triangular structures.

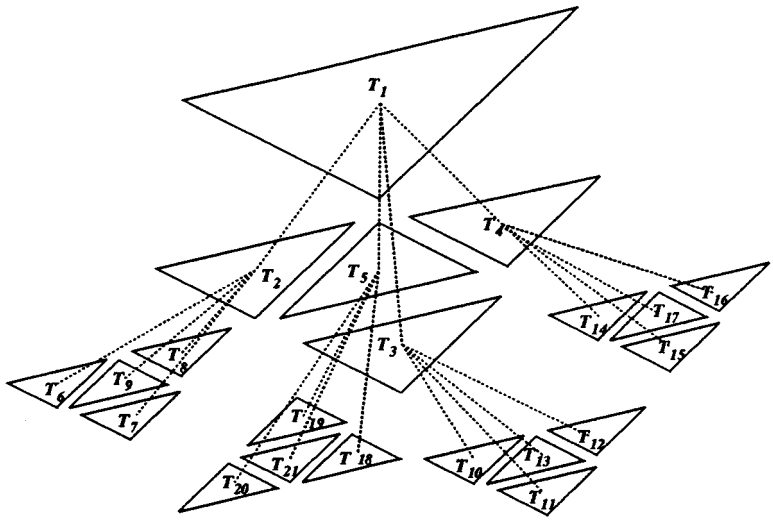

Figure 1. The figure shows the two-dimensional tri-tree structure for subsequent divisions of a triangle into 16 triangles.

The four terminal leaves of the tree in Figure 1 are discarded in one level of recoarsement of the entire tree. Between each recoarsement level, a tri-tree element can not be involved in more than one recoarsement. The procedure is performed once more to obtain the next level of recoarsement.

Although the tri-tree elements are discarded with respect to the finite element grid generation, the discarded tri-tree elements are marked and kept in the tri-tree structure. The reverse operation, the refinement operation, is thus performed on the same tree structure by reinserting

1	1	p1	p2	p3	2	3	4	5	0
2	-2	p1	p4	p5	5	0	0		1
3	-2	p4	p2	p6	0	5	0		1
4	-2	p5	p6	p3	0	0	5		1
5	-2	p6	p4	p5	2	4	3		1

Figure 2. The figure shows the information in the tri-tree structure in two dimensions. This information is contained in a record consisting of nine integers. The first integer describes the level of refinement. When this integer is negative, it indicates a terminal triangle. The next three integers are the indices to the corners of the present division. If the refinement level integer is positive, the next four integers are pointing to the triangles into which the triangle is refined. If the triangle is terminal, the following three integers are pointers to the neighboring triangles. If one of these integers is zero the triangle has no neighbor in that direction. The last index is pointing to the parent triangle.

the discarded tri-tree elements into the tri-tree. The recoarsement–refinement algorithm is, therefore, performed on an existing tri-tree structure and does not require additional memory or file storage.

3. THE NAVIER–STOKES EQUATIONS

3.1. The coupled algorithm

The non-linear Navier–Stokes equations are given by

$$\begin{aligned} &\rho \frac{\partial \mathbf{v}}{\partial \mathrm{t}} - \mu \nabla^2 \mathbf{v} + \rho \mathbf{v} \cdot \nabla \mathbf{v} + \nabla p = 0 \quad \text{in} \quad \Omega, \\ &-\nabla \cdot \mathbf{v} = 0 \quad \text{in} \quad \Omega, \end{aligned} \tag{1}$$

where $\mathbf{v}$ is the velocity vector, p is the pressure and μ is the viscosity coefficient. The first equation is the equation of motion which contains a time, diffusion, convection and pressure term. The second equation is the equation of continuity. A minus sign is introduced into the continuity equation, in order to obtain the same sign for the pressure gradient as for the continuity equation in the finite element formulation.

In the finite element formulation, the velocities are approximated by quadratic basis functions and the pressure is approximated by linear basis functions on each element [1]. Note the quadratic polynomials N_i and the linear polynomial L_i. Then, by the Galerkin residual method and integration by parts, the second-order finite element formulation of the Navier–Stokes equation system becomes,

$$\begin{aligned} \mathbf{F}_{\mathbf{v}} &= \int_{\Omega} \left[\rho N_i \frac{\partial \mathbf{v}}{\partial t} + \mu \nabla N_i \cdot \nabla \mathbf{v} + \rho N_i \mathbf{v} \cdot \nabla \mathbf{v} - \nabla N_i p \right] \mathrm{d}\Omega - \int_{\delta\Omega} \left[\mu N_i \frac{\partial \mathbf{v}}{\partial n} - N_i p \right] \mathrm{d}\delta\Omega = 0, \\ \mathbf{F}_{\mathbf{p}} &= -\int_{\Omega} L_i \nabla \cdot \mathbf{v} \, \mathrm{d}\Omega = 0. \end{aligned} \tag{2}$$

There are several methods of linearizing this equation system. The common linearization techniques involve computation of gradients or approximate gradients as the Newton method. The Newton linearization method is a global method of linearization.

The Navier–Stokes equations have one non-linear term, the convective acceleration, which requires a non-linear iterative solution procedure. The non-linear algorithm chosen is the Newton method, which is known to have second-order convergence rate. The Navier–Stokes equations are then differentiated with respect to the unknowns, and a linear equation system must be solved at each Newton iteration.

$$\begin{bmatrix} \dfrac{\partial \mathbf{F}_{\mathbf{v}}^{\mathbf{n}}}{\partial \mathbf{v}} & \dfrac{\partial \mathbf{F}_{\mathbf{v}}^{\mathbf{n}}}{\partial \mathrm{p}} \\ \dfrac{\partial \mathbf{F}_{\mathbf{p}}^{\mathbf{n}}}{\partial \mathbf{v}} & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{v}^{\mathbf{n}+1} \\ \Delta \mathbf{p}^{\mathbf{n}+1} \end{bmatrix} = - \begin{bmatrix} \mathbf{F}_{\mathbf{v}}^{\mathbf{n}} \\ \mathbf{F}_{\mathbf{p}}^{\mathbf{n}} \end{bmatrix}, \tag{3}$$

where

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta\mathbf{v}^{n+1}, \tag{4}$$

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \Delta\mathbf{p}^{n+1}. \tag{5}$$

If the initial solution $\mathbf{v}^0$ and p^0 which is chosen, is close enough to the final solution, convergence of the non-linear equation system is guaranteed.

3.2. The pressure split algorithm

The original Navier–Stokes equation can be reformulated into three equations which describe the fluid flow. In the reformulated version below, there is one excessive equation, which simplifies the numerical solution algorithm.

$$\begin{aligned}
&\rho\frac{\partial\mathbf{v}}{\partial t} - \mu\nabla^2\mathbf{v} + \rho\mathbf{v}\cdot\nabla\mathbf{v} + \nabla p = 0 \quad \text{in} \quad \Omega,\\
&\nabla^2 p - \rho\nabla\cdot\frac{\partial\mathbf{v}}{\partial t} = 0 \quad \text{in} \quad \Omega,\\
&\rho\frac{\partial\mathbf{v}}{\partial t} + \nabla p = 0 \quad \text{in} \quad \Omega.
\end{aligned} \tag{6}$$

The fine element formulation of the pressure split equations becomes,

$$\begin{aligned}
\mathbf{F}_\mathbf{v} &= \int_\Omega \left[\rho L_i\frac{\partial\mathbf{v}}{\partial t} + \mu\nabla L_i\cdot\nabla\mathbf{v} + \rho L_i\mathbf{v}\cdot\nabla\mathbf{v} - \nabla L_i p\right]\mathrm{d}\Omega - \int_{\delta\Omega}\mu L_i\frac{\partial\mathbf{v}}{\partial n}\,\mathrm{d}\delta\Omega = 0,\\
\mathbf{F}_\mathbf{p} &= \int_\Omega \left[\nabla L_i\nabla p + \rho L_i\nabla\cdot\frac{\partial\mathbf{v}}{\Delta t}\right]\mathrm{d}\Omega - \int_{\delta\Omega} L_i\frac{\partial p}{\partial n}\,\mathrm{d}\delta\Omega = 0,\\
\mathbf{F}_\mathbf{m} &= \int_\Omega \left[\rho L_i\frac{\partial\mathbf{v}}{\partial t} + \nabla p\right]\mathrm{d}\Omega = 0.
\end{aligned} \tag{7}$$

The three equations, $\mathbf{F}_\mathbf{v}$, $\mathbf{F}_\mathbf{p}$ and $\mathbf{F}_\mathbf{m}$ are then solved sequencially by the non-linear Newton algorithm.

$$\frac{\partial\mathbf{F}_\mathbf{v}^{\,n}}{\partial\mathbf{v}}\Delta\mathbf{v}^{n+1} = -\mathbf{F}_\mathbf{v}^{\,n}\frac{\partial\mathbf{F}_\mathbf{p}^{\,n}}{\partial\mathbf{v}}\Delta\mathbf{p}^{n+1} = -\mathbf{F}_\mathbf{p}^{\,n}\frac{\partial\mathbf{F}_\mathbf{m}^{\,n}}{\partial\mathbf{v}}\Delta\mathbf{v}^{n+1} = -\mathbf{F}_\mathbf{m}^{\,n}. \tag{8}$$

By replacing the pressure with $d^n = p^n - p^{n-1}$ [8], the Newton formulations of the pressure split the equations are,

$$\begin{aligned}
&\int_\Omega\left[\rho L_i\frac{L_j}{\Delta t} + \mu\nabla L_i\nabla L_j + \rho L_i(\nabla\mathbf{v}^n L_j + \mathbf{v}^n\nabla L_j)\right]\mathrm{d}\Omega\Delta\mathbf{v}^{n+1}\\
&= -\int_\Omega\left[\mu\nabla L_i\cdot\nabla\mathbf{v}^n + \rho L_i\mathbf{v}_\mathrm{n}\cdot\nabla\mathbf{v}^n + \nabla L_i p^n\right]\mathrm{d}\Omega,\\
&\int_\Omega\nabla L_i\nabla L_j\,\mathrm{d}\Omega\Delta p^{n+1} = -\int_\Omega\left[\rho L_i\frac{\nabla\cdot\mathbf{v}^n}{\Delta t} + \nabla L_i\nabla d^n\right]\mathrm{d}\Omega,\\
&\int_\Omega\rho L_i\frac{Lj}{\Delta t}\,\mathrm{d}\Omega\Delta\mathbf{v}^{\mathbf{n}+\mathbf{1}} = -\int_\Omega\nabla d^n\,\mathrm{d}\Omega.
\end{aligned} \tag{9}$$

3.3. The velocity–pressure split algorithm

The original Navier–Stokes equation can be reformulated into four equations which describe the fluid flow. In the reformulated version below, there is one excessive equation, which simplifies the numerical solution algorithm.

$$\begin{aligned}
&\rho \frac{\partial \mathbf{v}}{\partial t} - \mu \nabla^2 \mathbf{v} + \nabla p = 0 \quad \text{in} \quad \Omega,\\
&\rho \frac{\partial \mathbf{v}}{\partial t} + \rho \mathbf{v} \cdot \nabla \mathbf{v} = 0 \quad \text{in} \quad \Omega,\\
&\partial \nabla^2 p - \rho \nabla \cdot \frac{\partial v}{\partial t} = 0 \quad \text{in} \quad \Omega,\\
&\rho \frac{\partial \mathbf{v}}{\partial t} + \nabla p = 0 \quad \text{in} \quad \Omega.
\end{aligned} \tag{10}$$

The fine element formulation of the velocity–pressure split equations becomes,

$$\begin{aligned}
\mathbf{F}_\mathbf{v} &= \int_\Omega \left[\rho L_i \frac{\partial \mathbf{v}}{\partial t} + \mu \nabla L_i \cdot \nabla \mathbf{v} - \nabla L_i p \right] \mathrm{d}\Omega - \int_{\delta\Omega} \mu L_i \frac{\partial \mathbf{v}}{\partial n} \, \mathrm{d}\delta\Omega = 0,\\
\mathbf{F}_\mathbf{c} &= \int_\Omega \left[\rho L_i \frac{\partial \mathbf{v}}{\partial t} + \rho L_i \mathbf{v} \cdot \nabla \mathbf{v} \right] \mathrm{d}\Omega = 0,\\
\mathbf{F}_\mathbf{p} &= \int_\Omega \left[\rho L_i \nabla \cdot \frac{\partial \mathbf{v}}{\Delta t} + \nabla L_i \nabla p \right] \mathrm{d}\Omega - \int_{\delta\Omega} L_i \frac{\partial p}{\partial n} \, \mathrm{d}\delta\Omega = 0,\\
\mathbf{F}_\mathbf{m} &= \int_\Omega \left[\rho L_i \frac{\partial \mathbf{v}}{\partial t} + \nabla p \right] \mathrm{d}\Omega = 0.
\end{aligned} \tag{11}$$

The four equations, $\mathbf{F}_\mathbf{v}$, $\mathbf{F}_\mathbf{c}$, $\mathbf{F}_\mathbf{p}$ and $\mathbf{F}_\mathbf{m}$ are then solved by the non-linear Newton algorithm.

$$\begin{aligned}
&\frac{\partial \mathbf{F}_\mathbf{v}^{\,n}}{\partial \mathbf{v}} \Delta \mathbf{v}^{n+1} = -\mathbf{F}_\mathbf{v}^{\,n} \frac{\partial \mathbf{F}_\mathbf{c}^{\,n}}{\partial \mathbf{v}} \Delta \mathbf{v}^{n+1} = -\mathbf{F}_\mathbf{c}^{\,n},\\
&\frac{\partial \mathbf{F}_\mathbf{p}^{\,n}}{\partial \mathbf{v}} \Delta \mathbf{p}^{n+1} = -\mathbf{F}_\mathbf{p}^{\,n} \frac{\partial \mathbf{F}_\mathbf{m}^{\,n}}{\partial \mathbf{v}} \Delta \mathbf{v}^{n+1} = -\mathbf{F}_\mathbf{m}^{\,n}.
\end{aligned} \tag{12}$$

Replacing the pressure with $d^n = p^n - p^{n-1}$ [8], to achieve divergence free flow, the Newton formulations of the pressure split equations are,

$$\begin{aligned}
&\int_\Omega \left[\rho L_i \frac{L_j}{\Delta t} + \mu \nabla L_i \nabla L_j \right] \mathrm{d}\Omega \Delta \mathbf{v} = \int_\Omega \left[\mu \nabla L_i \cdot \nabla \mathbf{v}^n + \nabla L_i p^n \right] \mathrm{d}\Omega,\\
&\int_\Omega \rho L_i \frac{L_j}{\Delta t} \, \mathrm{d}\Omega \Delta \mathbf{v} = -\int_\Omega \rho L_i \mathbf{v}^\mathrm{n} \cdot \nabla \mathbf{v}^n \, \mathrm{d}\Omega,\\
&\int_\Omega \nabla L_i \nabla L_j \, \mathrm{d}\Omega \Delta p = -\int_\Omega \left[\rho L_i \frac{\nabla \cdot \mathbf{v}^n}{\Delta t} + \nabla L_i \nabla d^n \right] \mathrm{d}\Omega,\\
&\int_\Omega \rho L_i \frac{Lj}{\Delta t} \, \mathrm{d}\Omega \Delta \mathbf{v} = -\int_\Omega \nabla d^n \, \mathrm{d}\Omega.
\end{aligned} \tag{13}$$

Table I. Computation parameters

Re	Corner nodes	Elements	Δt	Adaption
100	550	1024	500	
200	608	1146	70	0.0
400	1021	1972	30	0.9
800	3061	6047	10	1.2
1600	13 002	25 913	1	6.9

The table shows the grid and time parameters: the number of elements, the number of nodes, the time step and the time used for adapting the grid to the solution

Let n_d be the spatial dimension, then the exact integrals above can easily be computed using the formula [14],

$$\int_\Omega L_i^\alpha L_j^\beta L_k^\gamma = \frac{\alpha!\beta!\gamma!}{(\alpha+\beta+\gamma+n_d)!}\, n_d!\Omega.$$

4. BOUNDARY CONDITIONS

The boundary conditions at external boundaries where either $\mathbf{v}$ is specified or $\partial\mathbf{v}/\partial n = 0$, will cancel the boundary integrals $\int_{\delta\Omega} \mu L_i\, \partial\mathbf{v}^n \backslash \partial n\, \mathrm{d}\delta\Omega$ in Equation (11). The pressure in Equation (11) is replaced by $d^n = p^n - p^{n-1}$. Then the normal derivative $\partial d/\partial n = 0$ and the boundary integral $\int_{\delta\Omega} L_i\, \partial d^n \backslash \partial n\, \mathrm{d}\delta\Omega$ cancel.

The advantage of using the non-linear Newton formulation for solving linear equations is that the boundary conditions for the correction introduced into the equation system are always zero, while the actual boundary value is inserted in the initial solution vector.

The pressure computations are based on the continuity equation for each element. Therefore the final pressure has to be calculated from the Poisson equation with boundary conditions. The Poisson equation is derived from the differentiation of the Navier–Stokes equations and substitution of the continuity equation.

$$\nabla^2 p + \rho\nabla(\mathbf{v}\cdot\nabla\cdot\mathbf{v}) = 0 \quad \text{in} \quad \Omega. \tag{14}$$

The finite element formulation and integration by parts result in,

$$\mathbf{F_P} = \int_\Omega [\nabla L_i \nabla p + \rho\nabla L_i(\mathbf{v}\cdot\nabla\mathbf{v})]\, \mathrm{d}\Omega - \int_{\delta\Omega} L_i \frac{\partial p}{\partial n}\, \mathrm{d}\delta\Omega. \tag{15}$$

By substitution of the velocity boundary conditions for external boundaries, $\partial\mathbf{v}/\partial n = 0$, and Newton formulation, the equation to be solved becomes,

$$\int_\Omega [\nabla L_i \nabla L_j\, \mathrm{d}\Omega\Delta p] = -\left[\int_\Omega [\nabla L_i \nabla p^n + \rho\nabla L_i(\mathbf{v}\cdot\nabla\mathbf{v})]\, \mathrm{d}\Omega - \int_{\delta\Omega} L_i \frac{\partial p}{\partial n}\, \mathrm{d}\delta\Omega\right]. \tag{16}$$

The pressure boundary condition $\partial p/\partial n$ is obtained directly from the Navier–Stokes equations and expressed by the velocity terms. In general, the normal derivative of the pressure is given by

$$\frac{\partial p}{\partial x} = -\rho \frac{\partial u}{\partial t} + \mu \nabla^2 u, \qquad \frac{\partial p}{\partial y} = -\rho \frac{\partial v}{\partial t} + \mu \nabla^2 v, \qquad \frac{\partial p}{\partial z} = -\rho \frac{\partial w}{\partial t} + \mu \nabla^2 w, \tag{17}$$

where

$$\mathbf{v} = [u, v, w] \qquad \frac{\partial p}{\partial n} = \left[\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z}\right] \cdot \mathbf{n}. \tag{18}$$

In the driven cavity problem, $\partial p/\partial n = 0$ for all boundaries. For flow where the velocity is constant along the inlet boundary, $\mu \nabla^2 \mathbf{v} = 0$. For flow in tubes where there is a parabolic flow profile, $u = u_0(1 - r^2/R^2)$ at the inlet, $\mu\, \partial^2 u/\partial r^2 = -\mu u_0\, 2/R^2$.

5. NUMERICAL SOLUTION ALGORITHMS

In the mixed formulation of the Navier–Stokes equations, the non-linear equation system is Newton formulated and the matrix coefficients are assembled. The storage scheme is sparse and described in detail in Dahl and Wille [2]. The corresponding linear equation system of the Newton formulation of the Navier–Stokes equations is solved using the non-symmetric CGSTAB conjugate gradient algorithm with coupled node fill in preconditioning.

In the pressure split algorithm, the velocity is found using the Navier–Stokes equations, $\mathbf{F_v}$ in Equation (7), with ILU preconditioned CGSTAB for the assembled equation matrix and with diagonal preconditioned CGSTAB in the nodal algorithm.

The pressure is found from the Poisson equation, $\mathbf{F_P}$, by ILU preconditioned, symmetric conjugate gradients for the assembled equation system and diagonal preconditioned, symmetric conjugate gradients for the non-assembled equation system.

The velocity correction equation $\mathbf{F_m}$ in Equation (7) is solved by diagonal preconditioned symmetric conjugate gradients for the non-assembled equation system and the ILU preconditioned symmetric conjugate gradients for the assembled equation system. When the velocity and pressure equations, $\mathbf{F_v}$ and $\mathbf{F_p}$ are not assembled, the mass matrix in $\mathbf{F_m}$ is simply lumped and the velocity correction is found by inverting the lumped mass vector.

In the velocity–pressure split algorithm, all equation systems to be solved are symmetric and positive definite. The equations $\mathbf{F_p}$ and $\mathbf{F_m}$ are solved as in the pressure split algorithm. The velocity is found from $\mathbf{F_v}$ in Equation (11), in the same way as in the pressure split algorithm, except that the symmetric conjugate gradient algorithm is used instead of CGSTAB. The velocity correction equation $\mathbf{F_c}$, due to convection, is solved in the same way as the velocity correction equation due to the pressure, $\mathbf{F_m}$.

Table II. Coupled algorithm

Re	Linear	Solution time	Error
100	73	141	$3 \cdot 10^{-3}$
200	60	146	$7 \cdot 10^{-4}$
400	127	455	$4 \cdot 10^{-3}$
800	238	2 957	$5 \cdot 10^{-3}$
1600	500	19 127	$3 \cdot 10^{-2}$

The table shows the number of linear iterations, the solution time and the error estimate for the coupled algorithm

Table III. Time adaptation

Re	Δt	Max Re_e	Max Co_e	Max Dp_e
200	620	0.98	2.51	9.16
400	290	0.99	1.84	4.30
600	110	0.99	1.73	10.81
800	60	0.97	1.26	5.90
1000	40	0.98	2.08	3.93

The table shows the maximum Reynolds number, the maximum Courant number and the maximum diffusion parameter before divergence when incrementing the time step for different Reynolds numbers

6. SOLUTION ADAPTION

There are three important parameters in the solution algorithm for the Navier–Stokes equations. These parameters are the Reynolds number (Re), the Courant number (Co) and the diffusion parameter (Dp).

$$Re = \frac{\rho \|\mathbf{v}\cdot\nabla\mathbf{v}\|}{\mu\|\nabla^2\mathbf{v}\|}, \qquad \mathrm{Co} = \frac{\rho\|\mathbf{v}\cdot\nabla\mathbf{v}\|}{\rho\|\partial\mathbf{v}/\partial t\|}, \qquad \mathrm{Dp} = \frac{\mu\|\nabla^2\mathbf{v}\|}{\rho\|\partial\mathbf{v}/\partial t\|}. \tag{19}$$

The Reynolds number is defined as the ratio of convection to diffusion. The Courant number is defined as the ratio of convection to acceleration and the dissipation parameter is defined as the ratio of diffusion to acceleration. The Reynolds number reflects the degree of non-linearity in the equation system. The Courant number indicates the degree of hyperbolicity and the diffusion parameter indicates the degree of parabolicity in the equation system. The Reynolds number, Courant number and the diffusion parameter can be computed for each element.

6.1. Grid adaption

The element Reynolds number Re_e is computed for each tri-tree element from the expression given below. Let L_i^c be the linear basis function evaluated at the geometrical center of the element. Then the different parameters become

$$Re_e = \frac{\Sigma_i L_i^c \left\| \rho \int_\Omega N_i \mathbf{v}\cdot\nabla\mathbf{v}\, d\Omega \right\|}{\Sigma_i L_i^c \left\| \mu \int_\Omega \nabla L_i \cdot \nabla\mathbf{v}\, d\Omega \right\|} < \epsilon_{Re}. \tag{20}$$

Numerical experiments such as those of Wille [14,15] have shown that $\epsilon_{Re} < 10$ in two spatial dimensions and $\epsilon_{Re} < 30$ in three dimensions, in order to obtain a converged solution for the Navier–Stokes equations. In the present work, the element Reynolds number limit is $\epsilon_{Re} = 1$ for refinement and recoarsement of the grid.

6.2. Time adaption

The element Corant number Co_e and the element diffusion parameter Dp_e are computed for each tri-tree element.

$$Co_e = \frac{\Sigma_i L_i^c \left\| \rho \int_\Omega L_i \mathbf{v} \cdot \nabla \mathbf{v} \, d\Omega \right\|}{\Sigma_i L_i^c \left\| \rho \int_\Omega L_i \frac{\partial \mathbf{v}}{\partial t} \, d\Omega \right\|} < \epsilon_{Co}, \tag{21}$$

$$Dp_e = \frac{\Sigma_i L_i^c \left\| \mu \int_\Omega \nabla L_i \nabla \mathbf{v} \, d\Omega \right\|}{\Sigma_i L_i^c \left\| \rho \int_\Omega L_i \frac{\partial \mathbf{v}}{\partial t} \, d\Omega \right\|} < \epsilon_{Dp}. \tag{22}$$

For explicit time schemes, it has been shown theoretically that the time marching scheme remains stable if $\epsilon_{Co} < 1$ and $\epsilon_{Dp} < 1/2$ [16]. These values have been derived for whole geometries, where characteristic length and mean velocity are applied in the derivations. As the diffusion equation is solved implicitly, the diffusion parameter is not important for convergence. The element Courant number has been found experimentally to be in the range 1.0–2.5 when divergence occurs (Table III). In the present work, the Courant number limit is chosen to be 0.5. The length of the time step is computed from

$$\Delta t < 0.2 \Delta t_0 / \max(Co_e), \tag{23}$$

where Δt_0 is the time step in the computation of the element Courant number, Co_e.

7. EXPERIMENTS

The numerical algorithms are tested for the driven cavity flow problem shown in Figure 6. The density of the fluid is $\rho = 1000$ and the viscosity is $\mu = 0.001$. For the coupled algorithm, (1), the convergence criterion for the linear iterations is $\epsilon_L = 0.001$ and the non-linear convergence criterion is $\epsilon_N = 0.0001$. In order to ensure a minimum number of linear iterations, the conjugate gradient algorithms are forced to do at least $N_{L_{min}} = 5$ iterations. The maximum number of linear iterations to be executed is $N_{L_{max}} = 100$, which ensures that the conjugate gradient algorithms are stopped when the required convergence criterion is not reached. Similar criteria are imposed on the non-linear Newton iterations. The maximum number of non-linear Newton iterations is set to $N_{N_{max}} = 5$. There are therefore two possibilities for stopping the linear and non-linear iterations. The iterations are either stopped when the convergence criterion is reached or when the number of iteration limits is reached. The time step in the coupled algorithm 1 is infinite, $\Delta t = \cdot 10^{30}$.

Table IV. Pressure split algorithm, full mass matrix

Re	Assembled algorithm			Nodal algorithm		
	Linear iterations	Solution time	Error	Linear iterations	Solution time	Error
100	615	69	$9 \cdot 10^{-4}$	698	255	$9 \cdot 10^{-4}$
200	750	90	$9 \cdot 10^{-4}$	664	260	$9 \cdot 10^{-4}$
400	1500	304	$1 \cdot 10^{-3}$	1532	1051	$1 \cdot 10^{-3}$
800	1505	979	$2 \cdot 10^{-3}$	2750	3467	$2 \cdot 10^{-3}$
1600	1560	4253	$3 \cdot 10^{-3}$	2779	15 959	$3 \cdot 10^{-2}$

The table shows the number of linear iterations, solution time in seconds and the obtained accuracy of the solution for the assembled and nodal pressure split algorithm with full mass matrices

Table V. Velocity–pressure split algorithm, full mass matrix

Re	Assembled algorithm			Nodal algorithm		
	Linear iterations	Solution time	Error	Linear iterations	Solution time	Error
100	920	86	$9 \cdot 10^{-4}$	1306	180	$9 \cdot 10^{-4}$
200	1300	133	$9 \cdot 10^{-4}$	1443	224	$9 \cdot 10^{-4}$
400	2000	347	$1 \cdot 10^{-3}$	2503	614	$1 \cdot 10^{-3}$
800	2011	1156	$2 \cdot 10^{-3}$	2616	2089	$7 \cdot 10^{-3}$
1600	2226	4766	$5 \cdot 10^{-3}$	3631	11 336	$3 \cdot 10^{-2}$

The table shows the number of linear iterations, solution time in seconds and the obtained accuracy of the solution for the assembled and nodal velocity–pressure split algorithm with full mass matrices

In the pressure split algorithm 6, the convergence criteria for the linear conjugate gradient iterations $\epsilon_L = 0.0001$. The limiting number of linear conjugate gradient iterations is $N_{L_{min}} = 5$ and $N_{L_{max}} = 50$. In the non-linear velocity Equation (6) one non-linear iteration is always executed. The number of outer split iterations is always $N_{S_{max}} = 100$.

The convergence and stop criteria for the velocity–pressure split algorithm are the same as for the pressure split algorithms.

7.1. Computational parameters

The computation parameters are shown in Table I. These parameters are characteristic for all numerical algorithms. As the solution adapted grids are generated during the computations in each algorithm, the grid and the grid parameters may vary slightly because the computations may have reached slightly different levels of convergence. The length of the time steps are the same for both of the operator split algorithms.

7.2. Coupled algorithm

The coupled algorithm has been thoroughly investigated in previous work [2,17,18]. For comparative reasons, the coupled algorithm is applied to the same test examples as the operator split algorithms. The results of the simulations for the coupled algorithm 1 are shown in Table II.

Table VI. Pressure split algorithm, lumped mass matrix

Re	Assembled algorithm			Nodal algorithm		
	Linear iterations	Solution time	Error	Linear iterations	Solution time	Error
100	400	53	$9 \cdot 10^{-4}$	427	213	$9 \cdot 10^{-4}$
200	500	70	$9 \cdot 10^{-4}$	431	228	$9 \cdot 10^{-4}$
400	1000	235	$9 \cdot 10^{-4}$	1022	926	$1 \cdot 10^{-4}$
800	1000	682	$1 \cdot 10^{-3}$	2955	3080	$9 \cdot 10^{-3}$
1600	1082	2997	$3 \cdot 10^{-3}$	2151	15 726	$1 \cdot 10^{-2}$

The table shows the number of linear iterations, solution time in seconds and the obtained accuracy of the solution for the assembled and nodal pressure split algorithm with lumped mass matrices.

Table VII. Velocity–pressure split algorithm, lumped mass matrix

Re	Assembled algorithm			Nodal algorithm		
	Linear iterations	Solution time	Error	Linear iterations	Solution time	Error
100	400	46	$9 \cdot 10^{-4}$	504	86	$9 \cdot 10^{-4}$
200	600	64	$9 \cdot 10^{-4}$	780	138	$9 \cdot 10^{-4}$
400	1000	196	$1 \cdot 10^{-3}$	1054	552	$1 \cdot 10^{-3}$
800	1001	611	$1 \cdot 10^{-2}$	2465	1991	$9 \cdot 10^{-2}$
1600	1059	3045	$6 \cdot 10^{-3}$	2268	8998	$1 \cdot 10^{-2}$

The table shows the number of linear iterations, solution time in seconds and the obtained accuracy of the solution for the assembled and nodal velocity-pressure split algorithm with lumped mass matrices.

7.3. Time adaption

The maximum possible time step which can be used without divergence is given in Table III. The computations are carried out using the velocity–pressure split algorithm. For each Reynolds number, time steps in increments of ten are applied until divergence occurs. The largest time step before divergence is then used for the computation of the Courant number and the diffusion parameter. The table shows that the maximum Courant number for convergence is in the range 1.2–2.5. The corresponding diffusion parameter is between 3.0 and 10.9. The important parameter in the present numerical algorithms is the Courant number. The updates due to pressure correction in the pressure split algorithm 6 and the velocity–pressure split algorithm 10 are performed explicitly. In the velocity–pressure split algorithm 10, the convective correction is also explicitly updated. In the present investigation, the diffusion equation for both algorithms is solved implicitly. The diffusion therefore does not impose any convergence limitations. The actual time step in the present investigation is chosen so the maximum element Courant number $Co_e < 0.2$

7.4. Operator split algorithms

Table IV shows the results of the simulations of cavity flow by the pressure split algorithm. The results on the left side of the table are for assembled equation systems and the conjugate gradient equation solvers are preconditioned by incomplete LU factorization. The results on the right side of Table IV are for the nodal algorithm and diagonal precondition-

Table VIII. Matrix storage

Re	Coupled assembled	Split assembled	Split nodal
100	$185 \cdot 10^3$	$16 \cdot 10^3$	0
200	$286 \cdot 10^3$	$18 \cdot 10^3$	0
400	$482 \cdot 10^3$	$30 \cdot 10^3$	0
800	$1075 \cdot 10^3$	$88 \cdot 10^3$	0
1000	$4655 \cdot 10^3$	$420 \cdot 10^3$	0

The table shows the number of coefficients and pointers to rows and columns in the matrices for the assembled algorithms. The nodal algorithms allocate no matrix storage.

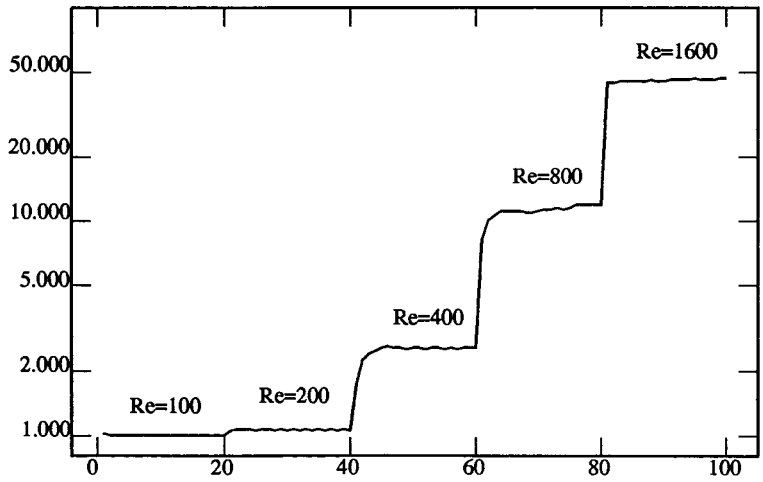

Figure 3. The figure shows the number of elements (logarithmic units) for each Reynolds number. The grid is iterated 20 times for each Reynolds number with respect to solution adaption.

ing of the equation system. The mass matrix in Equation (6) is assembled and the corresponding equation system is also solved by diagonal preconditioning of the conjugate gradient method. The table shows that the assembled algorithm is the most efficient of the assembled and the nodal algorithms, both in terms of number linear iterations and computational time.

Table V shows results of similar simulations for the velocity–pressure split algorithm. The results in the table show that the assembled algorithm is the most efficient in terms of number of total linear iterations as well as in computational time.

The results of the simulations of the velocity–pressure split algorithm with lumped mass matrices are shown in Tables VI and VII. These algorithms are faster than those with full mass matrices and corresponding solutions of the equation system. The mass lumping, however, has only minor effects on the convergence rate and the accuracy of the solutions.

The main advantages of the nodal algorithms are that no equation matrices are stored. Thus, problems can be investigated with more degrees of freedom and increased accuracy. The earnings in computational storage increase the computational times to some extent, but not dramatically. For the pressure split algorithm, the assembled equation matrix algorithm is approximately five times faster than the corresponding nodal algorithm. For the velocity–pressure split algorithm this factor is between two and three.

Comparing the two nodal algorithms, it appears that the velocity–pressure split algorithm is faster than the pressure split algorithm. The superiority of the velocity–pressure split algorithm is due to the fact that only linear matrix coefficients have to be generated during the matrix–vector multiplication in the conjugate gradient solvers.

7.5. Matrix storage

Table VIII shows the number of matrix coefficients which are stored in the computer memory for the different algorithms. The coupled algorithms require most storage, while the assembled operator split algorithms require a factor of ten less storage. The corresponding nodal algorithm does not need any storage for matrix coefficients.

7.6. Adaptive grid iterations

The tri-tree algorithm for refinements and recoarsements of finite element grids is explored. The refinement–recoarsement algorithm not only provides an accurate solution at certain parts of the grid, but has a major influence on the actual finite element equation system [19]. The refinements of the grid lead to a more symmetric and linear equation matrix. The recoarsements will ensure that the grid is not finer than necessary, for preventing divergence in an iterative solution procedure. The refinement–recoarsement algorithm is a dynamic procedure and the grid is adapted to the instant solution.

In the tri-tree multigrid algorithm, the solution from a coarser grid is scaled relative to the increase in the velocity boundary condition for the finer grid. In order to have a good start vector for the solution of the finer grid, the global Reynolds number or velocity boundary condition should not be subject to large changes. For each grid and each velocity solution, the element Reynolds number is computed and used as the grid adaptive indicator during the grid adaption procedure. The runtime expenses for grid adaption are very small compared with the computational time for solving the equation system (Table III). The grid can therefore be adapted several times during the iterations for obtaining a solution at a specific Reynolds number with only a minor increase in computational time.

Figure 3 shows the number of elements in the grid for each Reynolds number. The grid is refined if the element Reynolds number $Re_e > 1.0$ and recoarsed if the recoarsed element Reynolds number remains $Re_e < 1.0$.

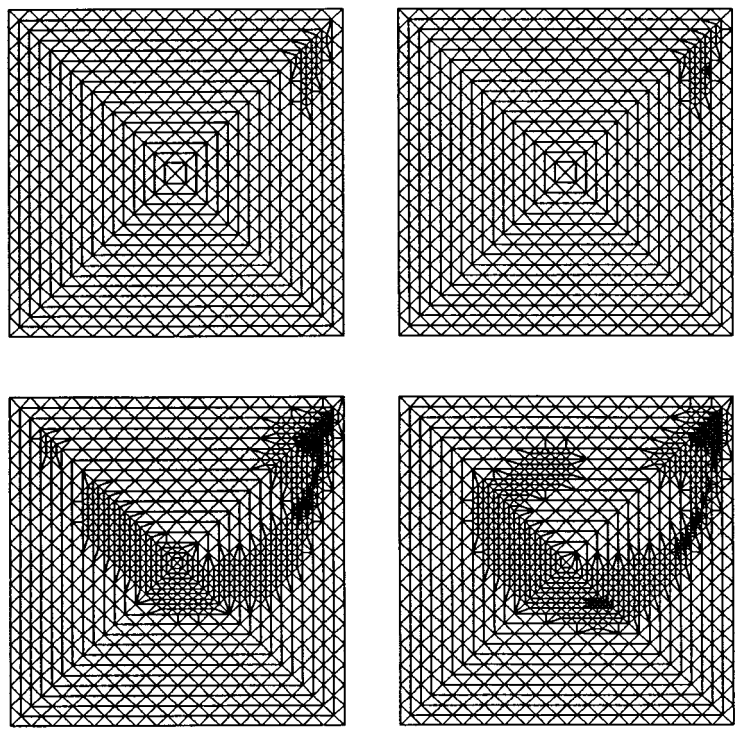

Figure 4. The figure shows the solution adapted grids obtained by the nodal velocity–pressure split algorithm. The grids to the left are the initial grids for Reynolds number 200 (upper) and Reynolds number 400 (lower). The grids to the right are the final grids for obtaining the converged solution. Ten grid adaptions are executed for each Reynolds number.

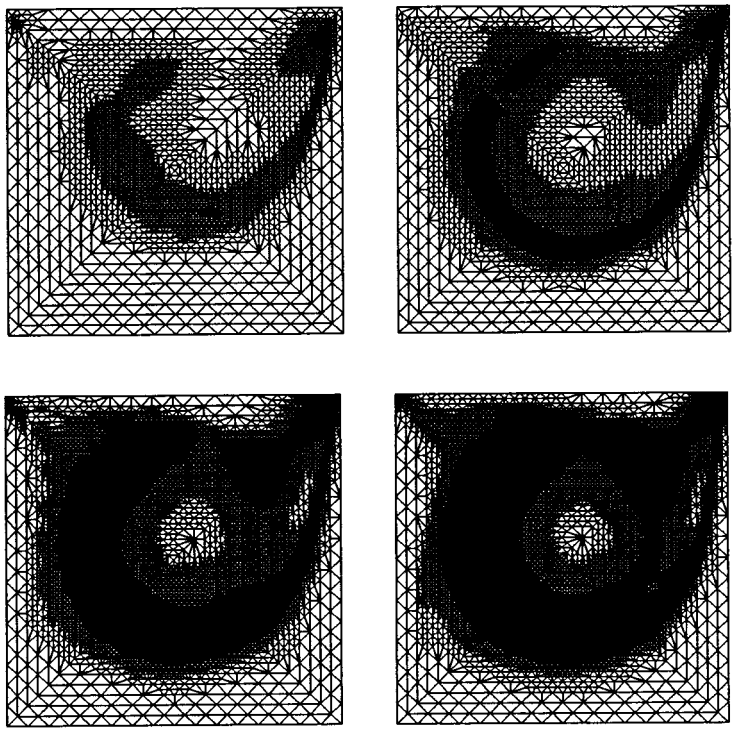

Figure 5. The figure shows the solution adapted grids obtained by the nodal velocity–pressure split algorithm. The grids to the left are the initial grids for Reynolds number 800 (upper) and Reynolds number 1600 (lower). The grids to the right are the final grids for obtaining the converged solution. Ten grid adaptions are executed for each Reynolds number.

Figures 4 and 5 show the initial and final grids for the adaptive refinements during the solution procedure for Reynolds numbers 200, 400, 800 and 1600. The figures clearly show that the adapted grids are changing as the solution converges during the solution procedure. The computations are executed with the velocity–pressure split algorithm (see Figure 6).

8. DISCUSSIONS

The Navier–Stokes equations may be considered as a composition of the diffusion equation and the convection equation. The solution of the diffusion equation is relatively smooth, while the solution of the convection equation is due to rapid changes in boundary conditions which propagate in the computational domain.

The traditional methods for solving the diffusion equations have been implicit, which implies that an equation system must be solved, either by a direct or an iterative equation solver. In most cases, in order to speed up the iterative solvers and stabilize the equation system, the equation system has to be preconditioned. Both the direct solver and the most efficient preconditioners require an assembled equation matrix. The storage of the assembled equation matrix is therefore a limiting factor for the size of the equation system which can be solved.

The convection equations have previously been solved successfully by explicit time marching schemes, such as the forward Euler method and the Runge–Kutta methods. The advantage of

explicit methods is that no equation systems need to be solved and therefore, no equation matrices have to be stored.

The velocity–pressure split algorithm consists of two mass equations and two Laplace equations. Provided that the convection and pressure terms are made sufficiently small by local grid adaption, the mass equations can be solved by simple mass lumping and inversion of the lumped mass vector.

The Laplace equations for diffusion and pressure can be solved by a symmetric preconditioned conjugate gradient method. ILU preconditioning is used in the assembled algorithm and diagonal preconditioning is used in the nodal algorithm.

Two nodal algorithms have been presented. Both algorithms are operator splitting algorithms. These algorithms utilize computation of the matrix coefficients when needed in the conjugate gradient equation solvers. Matrices do not need to be stored, which creates an advantage, in that the size of the problems to be investigated can be increased by several orders.

The present investigation must be considered as an initial demonstration of the ability of the operator splitting algorithms. Further investigations will include optimization of the parameters at Δt, the number of necessary maximum linear iterations and the number of non-linear iterations within each time step. These optimizations will be carried out with respect to a reduction in computational time.

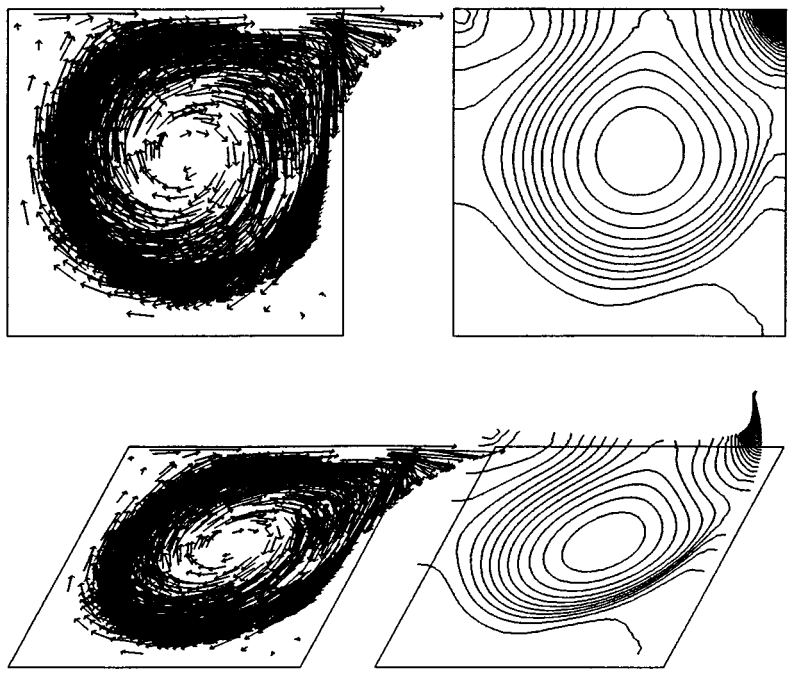

Figure 6. The figure shows the velocity vectors (left) and the pressure isobars (right) for Reynolds number 1600. The solution is computed using the nodal velocity–pressure split algorithm.

ACKNOWLEDGMENTS

The author is grateful to Demosthenes Skipitaris for skilled computer advice and to Trygve Svoldal for valuable suggestions and corrections of the manuscript. The project has been supported by The Norwegian Research Council, grant no. NN2461K for partial financing of the computer runtime expenses.

REFERENCES

1. C. Taylor and P. Hood, 'A numerical solution of the Navier–Stokes equations using the finite element technique', in *Computers and Fluids*, Pergamon, Oxford, **1**, 73–100 (1973).
2. O. Dahl and S.Ø. Wille, 'An ILU preconditioner with coupled node fill in for iterative solution of the mixed finite element formulation of the 2D and 3D Navier–Stokes equations', *Int. j. numer. methods fluids*, **15**, 525–544 (1992).
3. K. Goda, 'A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows', *J. Comp Phys.*, **30**, 76–95 (1979).
4. J. Kim and P. Moin, 'Application of a fractional-step method to incompressible Navier–Stokes equations', *J. Comp. Phys.*, **59**, 308–323 (1985).
5. T. Utnes, G. Moe and K.J. Eidsvik, 'Dynamic separation over bluff bodies predictions based upon Reynolds equations'. *J. Hyd. Res.*, **33**, 219–242 (1995).
6. G. Ren and T. Utnes, 'A finite element solution of the time-dependent Navier–Stokes equations using a modified velocity correction method', *Int. j. numer. methods fluids*, **17**, 349–364 (1993).
7. V. Ruas and J. Zhu, 'Decoupled solution of the velocity–vorticity system for two-dimensional viscous incompressible flow', *C.R. Akad Sci. Paris*, **t.318**, 293–298 (1994).
8. D. Goldberg, 'Applications de la methode des projections au calcyl par elements finis d'ecoulements tridimensionnels de fluides visqueux incompressible', *These de Doctorat l'Universite Paris VI.*, 1994.
9. Y. Kallinderis, 'Adaptive hybrid prismatic/tetrahedral grids', *Int. j. numer. methods fluids*, **20**, 1023–1037 (1992).
10. D.M. Greaves and A.G.L. Borthwick, 'Computation of flow past a square cylinder using adaptive hierarchical mesh', *Proc. 14th Int. Conf. on Offshore Mechanic and Arctic Engineering*, Copenhagen, Denmark, 1995.
11. D.M. Greaves, A.G.L. Borthwick, R. Eatock Taylor and G.X. Wu, 'Analysis of wave-body interactions using adaptive finite element meshes', *Proc. 10th Int. Workshop on Water Waves and Floating Bodies*, Oxford, UK, 1995.
12. K.F.C. Yiu, D.M. Greaves, S. Cruz, A. Saalehi and A.G.L. Borthwick, 'Quadtree grid generation: information handling, boundary fitting and CFD applications', *Comput. Fluids*, **25**, 759–769 (1996).
13. S.Ø. Wille, 'A structured tri-tree search method for generation of optimal unstructured finite element grids in two and three dimensions', *Int. j. numer. methods fluids*, **14**, 861–881 (1992).
14. S.Ø. Wille, 'The three dimensional prolonged adaptive unstructured finite element multigrid method for the Navier–Stokes equations', *Int. j. numer. methods fluids*, in press, 1996.
15. S.Ø. Wille, 'A local predictive convection–diffusion refinement indicator for the tri-tree adapted finite element multigrid algorithm of the Navier–Stokes equations', *Comput. Methods Appl. Mech. Eng.*, in press, 1996.
16. Z.U.A. Warsi, *Fluid Dynamics, Theoretical and Computational Approaches*, CRS Press, London, 1993.
17. S.Ø. Wille, 'A non linear adaptive full tri-tree multigrid method for the mixed finite element formulation of the Navier–Stokes equations', *Int. j. numer. methods fluids*, in press, 1995.
18. S.Ø. Wille, 'The prolonged adaptive multigrid method for finite element Navier–Stokes equations', *Comput. Methods Appl. Mech. Eng.*, **138**, 227–271 (1996).
19. S.Ø. Wille, 'Adaptive linearization and grid iterations with the tri-tree multigrid refinement–recoarsement algorithm for the Navier–Stokes equations', *Int. j. numer. methods fluids*, in press, 1996.

