INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **41**:977–996 (DOI: 10.1002/fld.477)

Parallel ILU preconditioning, *a priori* pivoting and segregation of variables for iterative solution of the mixed finite element formulation of the Navier–Stokes equations

S. Ø. Wille[1,2,*,†], Ø. Staff[1] and A. F. D. Loula[2]

[1]*Faculty of Engineering, Oslo University College, Norway*
[2]*Departamento de Matematica Aplicada e Computacional, Laboratório Nacional de Computação Cientifica, LNCC/CNPq, Petropolis, RJ, Brazil*

SUMMARY

A parallel ILU preconditioning algorithm for the incompressible Navier–Stokes equations has been designed, implemented and tested. The computational mesh is divided into N subdomains which are processed in parallel in different processors. During ILU factorization, matrices and vectors associated with the nodes on the interface between the subdomains are communicated to the equation matrices to the adjacent subdomain. The bases for the parallel algorithm are an appropriate node ordering scheme and a segregation of velocity and pressure degrees of freedom. The inner nodes of the subdomain are numbered first and then the nodes on the interface between the subdomains. To avoid division by zero during the ILU factorization, the equations corresponding to the velocity degrees of freedom are assembled first in the global equation matrix, followed by the equations corresponding to the pressure degrees of freedom. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: parallel computation; *a priori* pivoting; segregation of variables; ILU preconditioning; CGStab; sparse solvers; Navier–Stokes equations

1. INTRODUCTION

Large-scale simulations on single processors are usually limited by CPU time and the size of the central memory. Parallel multiprocessors computation is frequently used to overcome such limits. In this work a parallel finite element solver based on ILU preconditioning, *a priori* pivoting and segregation of variables are designed for the stationary Navier–Stokes system in

* Correspondence to: S. Ø. Wille, Faculty of Engineering, Oslo University College, Cort Adelersgate 30, N-0254 Oslo, Norway.
† E-mail: sowille@iu.hio.no

Contract/grant sponsor: CNPq, Conselho Nacional de Desenvolvimento Cintífico e Technologico—Brazil; contract/grant number: 300520/98-0
Contract/grant sponsor: NFR, The Norwegian Research Council; contract/grant number: 131 523/410

Received 26 May 2002
Revised 16 October 2002

its velocity–pressure formulation. An arbitrary number of processors can be used to perform large-scale simulations with reduced CPU time.

A mixed finite element approximation is considered with C^0 Lagrangian interpolation for both fields, quadratic for velocity and linear for pressure, yielding an indefinite, non-symmetric and non-linear global system. Newton's method is employed to solve the non-linear system iteratively. To avoid division by zero during the ILU factorization of the linearized system, a segregation of variable technique is applied. The equations corresponding to the velocity degrees of freedom are assembled first in the global matrix, and at last the equations corresponding to the pressure degrees of freedom which are the cause of the indefiniteness [1, 2]. During the parallel computation, adaptive meshing [3, 4] is applied to improve the finite element approximation and to improve the convergence performance of the iterative solver [5].

The global domain is divided into a number of subdomains. Due to the adaptation of the mesh with respect to the solution the total number of elements increases with the increasing Reynolds number and a new domain decomposition is required to balance the charge among the processors. Approximately, equal number of elements is distributed to each satellite processor.

Each satellite processor assembles the matrix corresponding to one subdomain and performs the ILU decomposition [2]. The parallel processing [6] is based on iterative equation solvers for non-symmetric systems [7–9].

In the parallel implementation of the ILU preconditioned algorithm the associated processors communicate matrices and vectors corresponding to the nodes on the interface between adjacent subdomains.

The improved computational performance of the proposed parallel solver is due to the adaptive meshing technique, ILU preconditioner, *a priori* pivoting and the segregation of variables [10–13].

In the present algorithm the node ordering is essential. A different, but functionally related approach is presented in References [14, 15]. These algorithms introduce globally defined preconditioners by mixing the clustered element-by-element preconditioning concept with incomplete factorization methods.

2. THE NAVIER–STOKES EQUATIONS

The model problem is the stationary Navier–Stokes system

$$\rho\mathbf{u}\cdot\nabla\mathbf{u} - \mu\nabla^2\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega \subset R^2 \tag{1}$$

$$-\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega \tag{2}$$

with homogeneous Dirichelet boundary conditions

$$\mathbf{u} = \mathbf{0} \quad \text{on } \Gamma = \partial\Omega \tag{3}$$

in which $\mathbf{u}$ is the velocity vector, p is the pressure, ρ is the density and μ is the viscosity coefficient. The first equation is the equation of motion which contains convection, diffusion and pressure gradient terms. The second equation is the equation of continuity.

A variational formulation of the Navier–Stokes system reads: Find the velocity $\mathbf{u} \in U = H_0^1(\Omega) \times H_0^1(\Omega)$ and the pressure $p \in Q = L(\Omega)/R$ such that

$$a(\mathbf{u}, \mathbf{u}, \mathbf{v}) + b(p, \mathbf{v}) = f(\mathbf{v}) \quad \forall \mathbf{v} \in U \tag{4}$$

$$b(q, \mathbf{u}) = 0 \quad \forall q \in Q \tag{5}$$

with

$$a(\mathbf{u}, \mathbf{w}, \mathbf{v}) = \int_\Omega [\mathbf{u} \cdot \nabla \mathbf{w} \cdot \mathbf{v} + \mu \nabla \mathbf{u} \cdot \nabla \mathbf{v} - p \nabla \cdot \mathbf{v}] \, \mathrm{d}\Omega \tag{6}$$

$$b(q, \mathbf{u}) = -\int_\Omega \nabla \cdot \mathbf{u} q \, \mathrm{d}\Omega \tag{7}$$

$$f(\mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} \, \mathrm{d}\Omega \tag{8}$$

We will consider mixed finite element formulation with both velocities and pressure approximated using C^0 Lagrangian interpolations with quadratic basis functions for velocities and linear basis functions for pressure on each element. Denoting $U_\mathrm{h} \subset U$ and $Q_\mathrm{h} \subset Q$, the corresponding C^0 Lagrangian finite element spaces for velocity and pressure fields, leads to the following Galerkin approximation of the Navier–Stokes equations:

$$a(\mathbf{u}_\mathrm{h}, \mathbf{u}_\mathrm{h}, \mathbf{v}_\mathrm{h}) + b(p_\mathrm{h}, \mathbf{v}_\mathrm{h}) = f(\mathbf{v}_\mathrm{h}) \quad \forall \mathbf{v}_\mathrm{h} \in U_\mathrm{h} \tag{9}$$

$$b(q_\mathrm{h}, \mathbf{u}_\mathrm{h}) = 0 \quad \forall q_\mathrm{h} \in Q_\mathrm{h} \tag{10}$$

3. FILL-IN RULES FOR INCOMPLETE GAUSSIAN FACTORIZATION

The fill-in rule for ILU preconditioner of iterative solvers for general algebraic equations is that fill-in is accepted at locations in the equation matrix where the magnitude of the coefficients is above a predefined limit. The order of fill-in is then determined by the size of this limit. This fill-in rule is not applicable for finite element equations as the pressure matrix is initially zero and would therefore never receive fill-ins. A different fill-in rule for finite element equations is then introduced as follows.

First-order fill-in: Fill-in is accepted at locations in the global matrix where the nodes belong to the same element.

Left-part of Figure 1 highlights three nodes: one in the lower-left corner, the centre node and the middle node at the upper edge. The right part of the figure shows the nodes where the degrees of freedom in the equation matrix receive fill-in during the factorization.

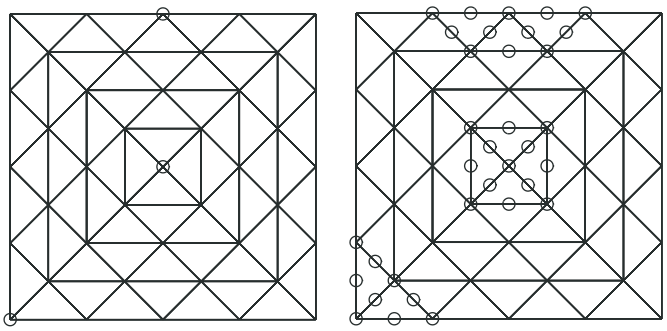


Figure 1. The figure to the left shows three nodes in mesh. The figure to the right indicate the locations of first order fill-in during factorization.

By applying this fill-in rule, the desired fill-in at the pressure locations in the continuity equations will take place. Thus, if nodal numbering, and thereby also the elimination order for the Gaussian factorization, is chosen carefully, the zeros at the pressure locations in the continuity equation will introduce no problems during the incomplete elimination.

4. PROCESSOR COMMUNICATION

The logical architecture considered is shown in Figure 2. The control processor can communicate with all satellites, and the satellites can communicate with their logical neighbours and the control processor. Communication primitives are restricted to point-to-point message passing.

The finite element computations are initiated in the control processor. The only function of the control processor is to initiate the computations. The control processor transmits the information needed by each satellite processor. When the finite element computations have been started, the satellite processors communicate with their neighbours in order to update matrices and vectors when needed. There is no communication concerning the computations between the control processor and any of the satellite processor after the initiation.

The implementation is based on a model where each satellite runs in a process on a standard computer networked with the other computers over Ethernet connected through a switch. Figure 3 shows the connection between any two processors, with each process divided into two threads running on the same processor. This is clearly a superset of the required logical architecture, and subdividing into two threads provides the added benefit of asynchronous communication between processors. When sending a message, the thread doing the actual calculations will queue the data in a fairly lightweight $O(1)$ operations, while the heavier $O(N)$ task of actually sending the message is left for the communicating thread to do in parallel with other calculations. When receiving a message, the asynchronous nature of the threads provides a benefit if the data was received while the processor was busy doing other work. Then, receiving a message is reduced to getting a pointer to data already in the process' address space, which is a fairly low overhead operation performed in fixed time. Figure 4

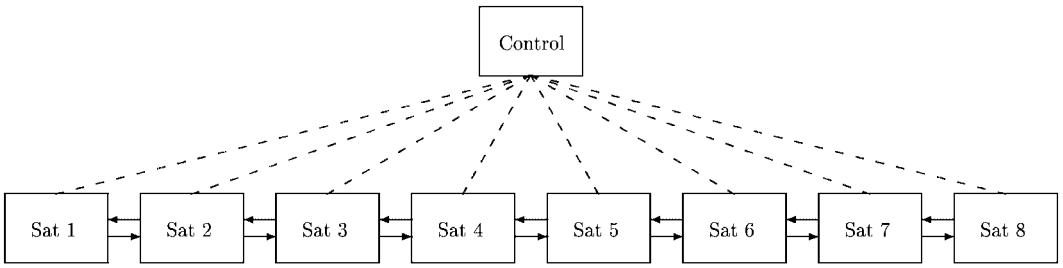


Figure 2. The configuration of a control processor and eight satellite processors. The control processor initiates the satellite processors. Each satellite processor has a two way communication to the neighbour satellite processors.

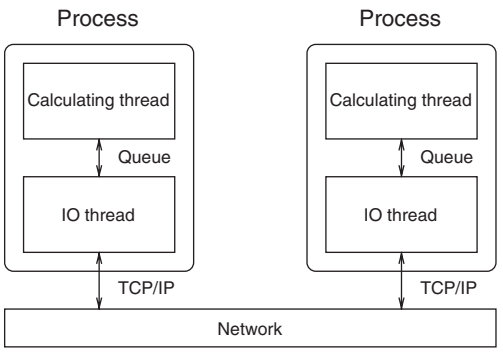


Figure 3. Two random processors (master or satellite) connected through a network. Each process is threaded, and the 'queue' communication is a constant overhead asynchronous operation while the TCP communication is a synchronous linear overhead operation.

shows a simplistic example, roughly corresponding to vector elimination for two satellites, of the difference between synchronous and asynchronous message passing. It is clear that the asynchronous mode has the potential to make some message operations extremely cheap.

As with everything else, added complexity, in this case that of running multiple threads, adds overhead. However, for the architecture at hand, this overhead is orders of magnitude lower than the time spent sending even fairly short (only several bytes long) messages. Additionally, the algorithm presented here is taylored to large messages, rendering the added overhead negligible.

5. NODE ORDERING IN SATELLITE MESHES

The basis for the parallel ILU preconditioning is the node ordering sequence. The local numbering of nodes for three subdomains is shown in Figure 5. The nodes in each subdomain

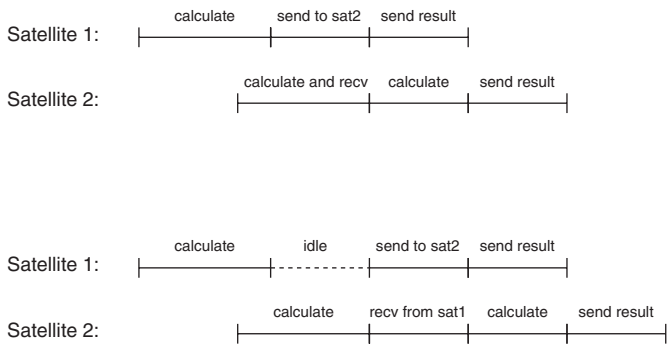


Figure 4. The potential difference between synchronous and asynchronous message passing. In the synchronous case, satellite 1 must wait for satellite 2 to finish calculating before the message can be sent. When passing the message asynchronously, satellite 2 will receive a message from satellite 1 while calculating, and when needing the data, and the message passing overhead is reduced to a simple constant time dequeue operation.

are numbered from the centre towards the periphery, and except for the nodes at the upper and lower interfaces, the inner nodes, are numbered first. Then the nodes at the upper interface are numbered and at last the nodes at the lower interface.

6. SEGREGATION OF VARIABLES

The segregation of the velocity and pressure degrees of freedom is illustrated in Figure 6. The velocity degrees of freedom for the inner nodes of the subdomain are assembled before the pressure degrees of freedom for the inner nodes. The degrees of freedom for the interface are then assembled by segregating the variables in the same fashion, first the velocity degrees of freedom, then the pressure degrees of freedom for the lower interface, and finally the velocity and pressure degrees of freedom for the upper interface.

7. STRUCTURE OF SATELLITE MATRICES

The numbering of nodes in each subdomain as shown in Figure 5 creates a finite element submatrix to be processed in each satellite processor as shown in Figure 6. The A matrices correspond to velocity degrees of freedom. The B matrices are the coupling matrices between velocity and pressure, while the C matrices correspond to the continuity equations. The pressure matrices P are initially equal to zero due to the absence of the pressure in the continuity equations. The index I refers to the inner nodes, the index R refers to the nodes on the upper interface and the index T refers to the nodes on the lower interface. The matrix A_{II} is thus the velocity matrix corresponding to the inner nodes. The matrix B_{II} is the coupling matrix between the velocities and pressures for the inner nodes. The matrix C_{II} contains the continuity equations for the inner nodes. The pressure matrix for the inner nodes is P_{II}. The matrices A_{IR}, B_{IR} are coupling velocity–velocity and velocity–pressure matrices between the

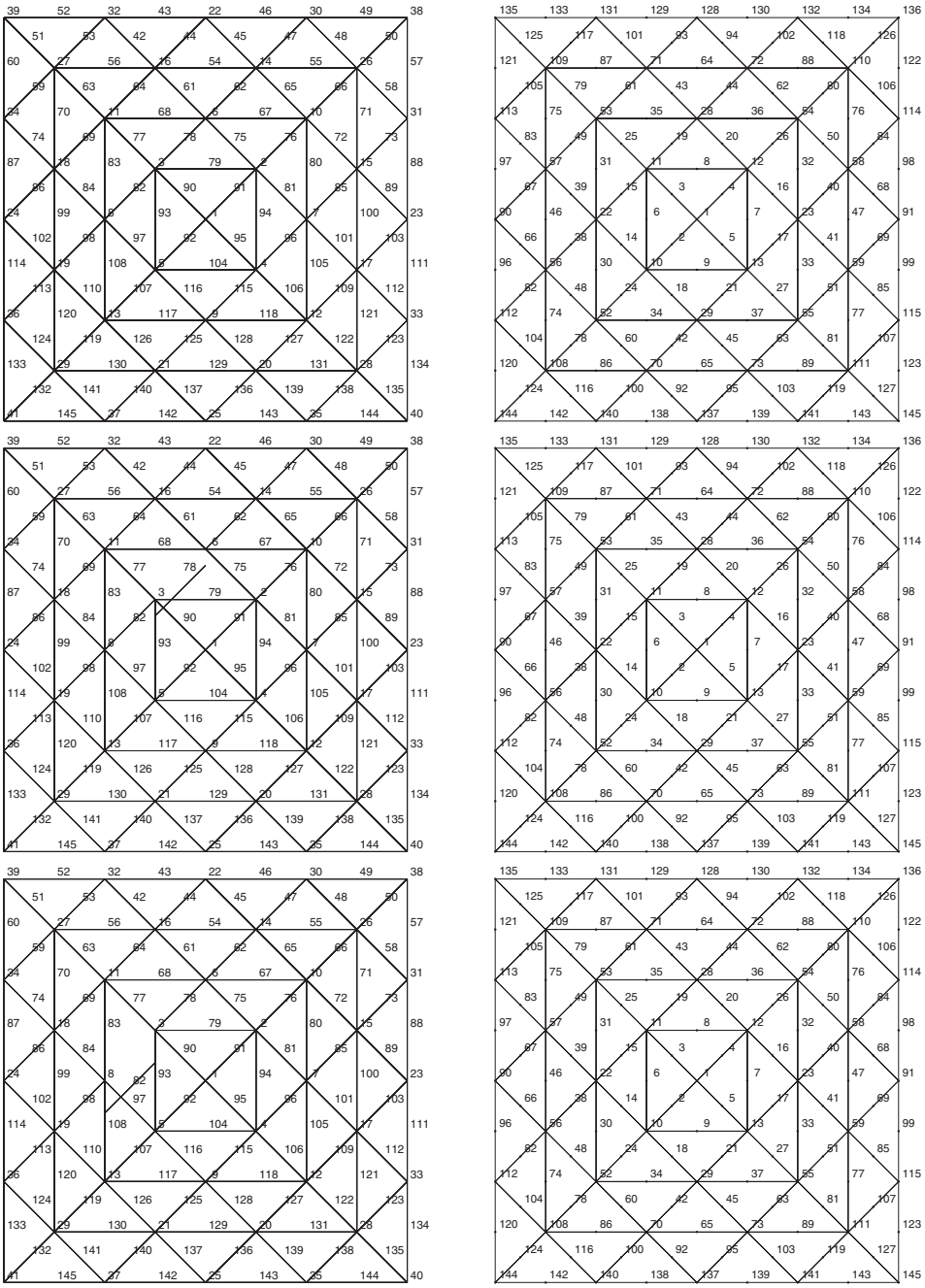


Figure 5. A mesh which is divided into three subdomains for parallel processing. The node numbering of each subdomain is divided in interior, receiving and transmitting nodes. The interior nodes are nodes 1–127. The receiving nodes are the nodes 128–136 and the transmitting nodes are the nodes 137–145. The interfaces between adjacent subdomains are the receiving and transmitting nodes.

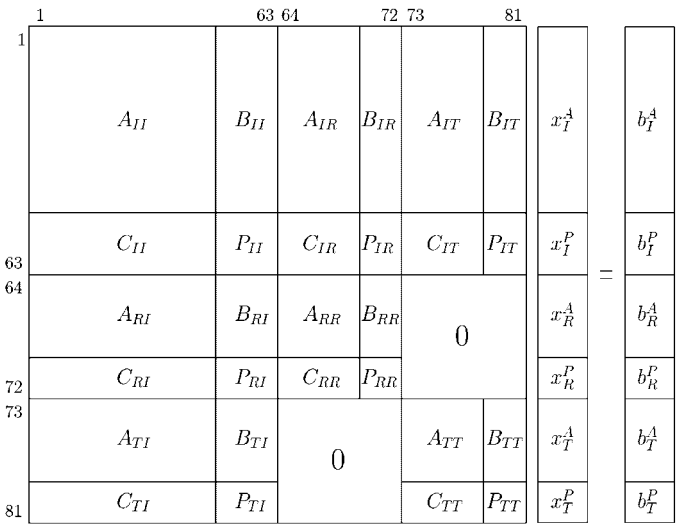
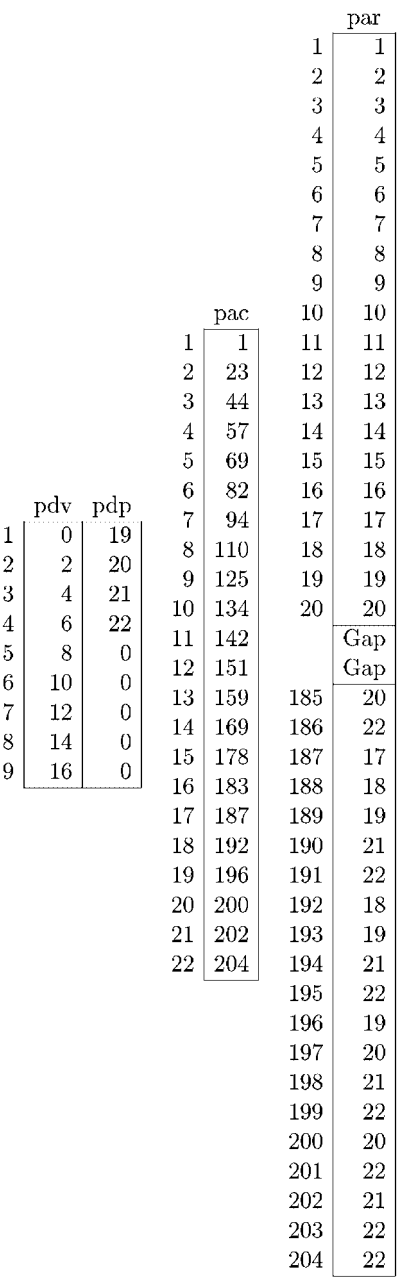


Figure 6. Finite element matrix of each subdomain. The matrix A_{II} corresponds to the internal degrees of freedom, the matrices A_{RR} A_{TT} corresponds to the degrees of freedom on the interface to adjacent subdomains. The zero matrices, **0** originate because there are no coupling between the nodes of the two interfaces.

inner nodes and nodes on the upper interface. The matrix C_{IR} is the continuity equations for the coupling of inner nodes and nodes on the upper interface. The matrices A_{IT}, B_{IT} and C_{IT} are corresponding matrices for the coupling between the inner nodes and the lower interface. The matrices for the degrees of freedom for the reverse coupling, the coupling between the nodes on the upper and lower interfaces, are contained in the matrices A_{RI}, B_{RI}, C_{RI}, P_{RI} and A_{TI}, B_{TI}, C_{TI}, P_{TI}, respectively. The matrices A_{RR}, B_{RR}, C_{RR}, P_{RR} correspond to the degrees of freedom of the nodes on the upper interface and the matrices A_{TT}, B_{TT}, C_{TT}, P_{TT} correspond to the degrees of freedom of the nodes on the lower interface. The two **0** matrices arise from the absence of coupling between the degrees of freedom of the nodes on the upper and lower interfaces.

The element matrices described above contain a large number of coefficients which are zero. A sparse storage structure, which is shown in Figure 7 is therefore used. The sparse storage structure in Figure 7 is for two-dimensional simulations. The pointer, $pdv[ni]+i$, where ni is the node number, $i=1$ and 2 are the u-velocity and the v-velocity, respectively, is pointing to the position in matrix column vector pac where the beginning of a row starts in the row vector par. The vector pac contains the address, $pdv[nj]+j$, when the degree of freedom is present in the equation matrix. The matrix coefficients are contained in a vector with the same size as the vector pac. The addresses of the pressure coefficients are obtained in a similar way.

Figure 8 shows the sparse pattern of the coefficients in the finite element equation matrix, which corresponds to each submesh, with node ordering given in Figure 5. The upper diagonal matrix shows the sparse pattern of the degrees of freedom of the internal nodes. The



	pdv	pdp
1	0	19
2	2	20
3	4	21
4	6	22
5	8	0
6	10	0
7	12	0
8	14	0
9	16	0

	pac
1	1
2	23
3	44
4	57
5	69
6	82
7	94
8	110
9	125
10	134
11	142
12	151
13	159
14	169
15	178
16	183
17	187
18	192
19	196
20	200
21	202
22	204

	par
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
	Gap
	Gap
185	20
186	22
187	17
188	18
189	19
190	21
191	22
192	18
193	19
194	21
195	22
196	19
197	20
198	21
199	22
200	20
201	22
202	21
203	22
204	22

Figure 7. Sparse storage structure of the finite element matrix of each subdomain. The vector *pac* contain pointers to *par*, the location of the beginning of each row. The vector *par* contains the nodes above the diagonal which are present in each row.

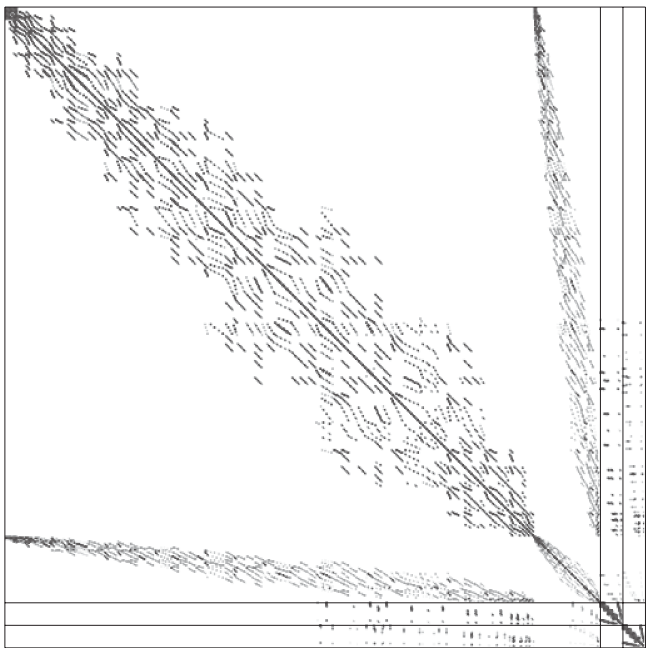


Figure 8. Sparse pattern of the finite element equation matrix of each subdomain.

small middle diagonal matrix shows the sparse pattern of the matrix which corresponds the upper boundary of the submesh. The small lower diagonal matrix shows the sparse pattern of the matrix which corresponds to the lower boundary of the submesh. In the matrices for internal nodes, upper boundary nodes and lower boundary nodes, the pressure gradient is assembled the right strip and the continuity equations is assembled in the lower strip. The coupling between internal nodes and upper boundary nodes is the first right and the first lower rectangle. The coupling between internal nodes and lower boundary nodes is the second right and the second lower rectangle. The **0** matrices in Figure 6 are also shown in Figure 8, where it is clearly seen that there is no coupling between the nodes on the upper and lower interfaces.

8. PARALLEL NUMERICAL ALGORITHM

The algorithms below show the elimination of each subdomain element matrix, Section 8.1, the vector elimination, Section 8.2, and vector substitution, Section 8.3, of the equation system in Figure 6. The superscript n denotes the satellite processor or subdomain number. The total number of satellite processors is MPR. These three algorithms are all executed in parallel on each satellite processor, except for some short events in the communication among the satellite processors.

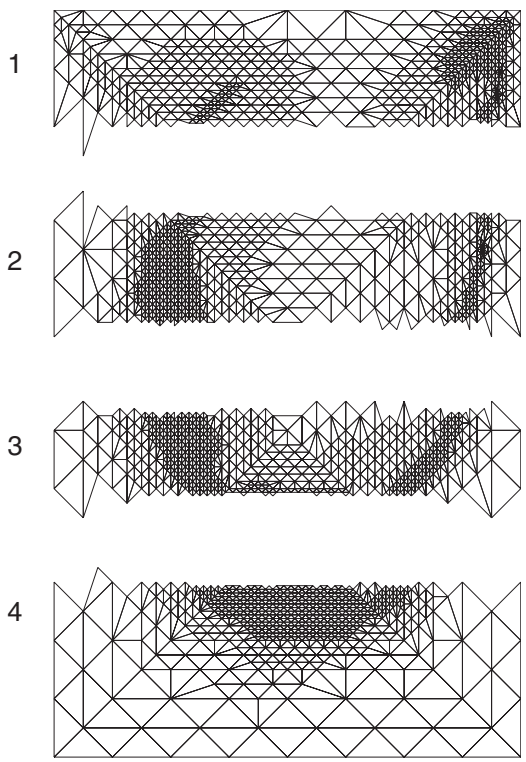


Figure 9. Two-dimensional mesh divided in four subdomains for Reynolds number 400. The mesh has been adaptively refined during the computations.

The elimination of the element matrix corresponding to the inner degrees of freedom, starts with the elimination of $\mathbf{A}_{\mathrm{II}}^{n}$ and $\mathbf{P}_{\mathrm{II}}^{n}$. During the elimination of $\mathbf{A}_{\mathrm{II}}^{n}$ sufficient fill-ins have occurred in $\mathbf{P}_{\mathrm{II}}^{n}$, so division by zero during the elimination of $\mathbf{P}_{\mathrm{II}}^{n}$ is avoided. At this point there will be no more contributions to the matrix $\mathbf{A}_{\mathrm{TT}}^{n}$ and $\mathbf{P}_{\mathrm{TT}}^{n}$ during the further elimination due to the two $\mathbf{0}$ matrices. Then the matrices $\mathbf{A}_{\mathrm{TT}}^{n-1}$ and $\mathbf{P}_{\mathrm{TT}}^{n-1}$ from the preceding subdomain is added into the matrices $\mathbf{A}_{\mathrm{RR}}^{n}$ and $\mathbf{P}_{\mathrm{RR}}^{n-1}$ of the present subdomain. The last step in the forward elimination, is the elimination of the matrices $\mathbf{A}_{\mathrm{RR}}^{n}$ and $\mathbf{P}_{\mathrm{RR}}^{n}$, which correspond to the degrees of freedom at the upper interface. During the elimination of $\mathbf{A}_{\mathrm{RR}}^{n}$, again fill-ins have occurred in $\mathbf{P}_{\mathrm{RR}}^{n}$, so the elimination of $\mathbf{P}_{\mathrm{RR}}^{n}$ creates no problems. The exceptions in the algorithm are for the first and the last subdomains. Since the first subdomain, $n=1$ has no preceding subdomain, the addition of the matrices $\mathbf{A}_{\mathrm{TT}}^{n-1}$ and $\mathbf{P}_{\mathrm{TT}}^{n-1}$ are omitted. For the last subdomain, the matrices $\mathbf{A}_{\mathrm{TT}}^{n}$ and $\mathbf{P}_{\mathrm{TT}}^{n}$ are also eliminated.

The forward elimination of the right-hand side, algorithm 8.2, follow the same path. First, the right-hand sides, $\mathbf{b}_{\mathrm{I}}^{\mathbf{A}^n}$ and $\mathbf{b}_{\mathrm{I}}^{\mathbf{P}^n}$, corresponding to the inner degrees of freedom are eliminated. The vectors $\mathbf{b}_{\mathrm{T}}^{\mathbf{A}^{n-1}}$ and $\mathbf{b}_{\mathrm{T}}^{\mathbf{P}^{n-1}}$ corresponding to the degrees of freedom at the lower interface of the preceding subdomain is added into the vectors $\mathbf{b}_{\mathrm{R}}^{\mathbf{A}^n}$ and $\mathbf{b}_{\mathrm{R}}^{\mathbf{P}^n}$ corresponding to the degrees of freedom at the upper interface of the present subdomain. The right-hand sides, $\mathbf{b}_{\mathrm{R}}^{\mathbf{A}^n}$ and $\mathbf{b}_{\mathrm{R}}^{\mathbf{P}^n}$,

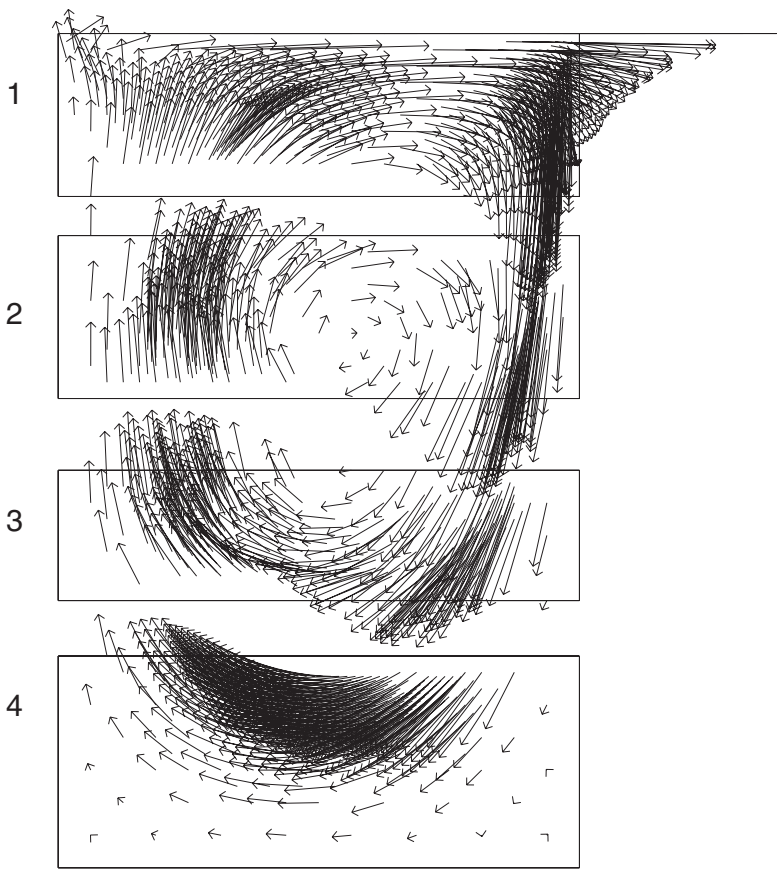


Figure 10. Two-dimensional velocity vector solution for the driven cavity flow for Reynolds number 400, computed on four satellite processors.

corresponding to the degrees of freedom at the upper interface is eliminated. The exceptions in the algorithm are again for the first and the last subdomains. For the first subdomain there is no contribution from the preceding subdomain so the addition of $\mathbf{b}_{\mathrm{T}}^{\mathbf{A}^{n-1}}$ and $\mathbf{b}_{\mathrm{T}}^{\mathbf{P}^{n-1}}$ into $\mathbf{b}_{\mathrm{R}}^{\mathbf{A}^{n}}$ and $\mathbf{b}_{\mathrm{R}}^{\mathbf{P}^{n}}$ is skipped. For the last subdomain, the elimination of $\mathbf{b}_{\mathrm{T}}^{\mathbf{A}^{n}}$ and $\mathbf{b}_{\mathrm{T}}^{\mathbf{P}^{n}}$, corresponding to the nodes of the lower edge, is performed.

The backward substitution algorithm, Section 8.3, has an exception for the last subdomain, $n = \mathrm{MPR}$. For the last subdomain, there is an additional substitution of $\mathbf{b}_{\mathrm{T}}^{\mathbf{P}^{n}}$ and $\mathbf{b}_{\mathrm{T}}^{\mathbf{A}^{n}}$, corresponding to the degrees of freedom at the lower edge. For all subdomains there is a backward substitution of the vectors $\mathbf{b}_{\mathrm{R}}^{\mathbf{P}^{n}}$ and $\mathbf{b}_{\mathrm{R}}^{\mathbf{A}^{n}}$, corresponding to the degrees of freedom at the upper interface. Then $\mathbf{b}_{\mathrm{T}}^{\mathbf{P}^{n}}$ is replaced by $\mathbf{b}_{\mathrm{R}}^{\mathbf{P}^{n+1}}$ and $\mathbf{b}_{\mathrm{T}}^{\mathbf{A}^{n}}$ is replaced by $\mathbf{b}_{\mathrm{R}}^{\mathbf{A}^{n+1}}$, since the solution for the degrees of freedom for the lower interface is already found in the proceeding subdomain. At last, the degrees of freedom corresponding to the inner degrees of freedom in each subdomain are found by the substitution of $\mathbf{b}_{\mathrm{I}}^{\mathbf{P}^{n}}$ and $\mathbf{b}_{\mathrm{I}}^{\mathbf{A}^{n}}$.

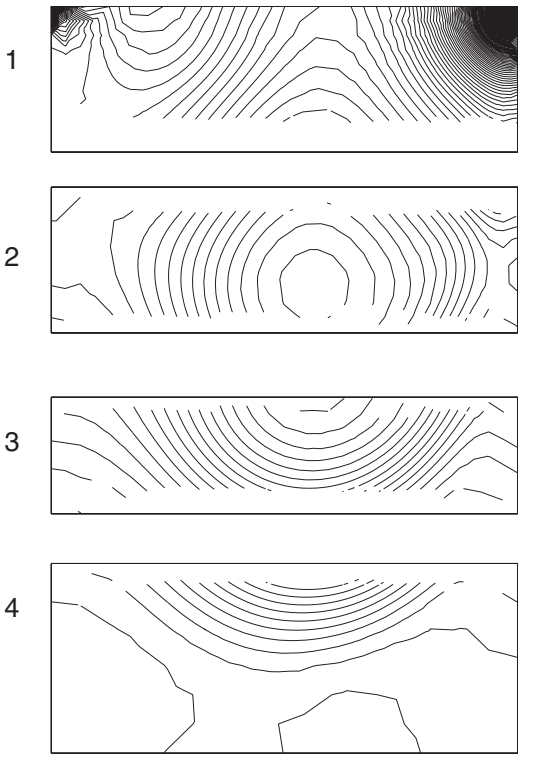


Figure 11. Two-dimensional pressure isobar solution for the driven cavity flow for Reynolds number 400, computed on four satellite processors.

8.1. Matrix elimination

	Eliminate	$\mathbf{A}_{\mathrm{II}}^{n},\ \mathbf{P}_{\mathrm{II}}^{n}$
if $(n>1)$	$\mathbf{A}_{\mathrm{RR}}^{n}=\mathbf{A}_{\mathrm{RR}}^{n}+\mathbf{A}_{\mathrm{TT}}^{n-1},$	$\mathbf{P}_{\mathrm{RR}}^{n}=\mathbf{P}_{\mathrm{RR}}^{n}+\mathbf{P}_{\mathrm{TT}}^{n-1}$
	Eliminate	$\mathbf{A}_{\mathrm{RR}}^{n},\ \mathbf{P}_{\mathrm{RR}}^{n}$
if $(n=\mathrm{MPR})$	*Eliminate*	$\mathbf{A}_{\mathrm{TT}}^{n},\ \mathbf{P}_{\mathrm{TT}}^{n}$

8.2. Vector elimination

	Eliminate	$\mathbf{b}_{\mathrm{I}}^{\mathbf{A}^n},\ \mathbf{b}_{\mathrm{I}}^{\mathbf{P}^n}$
if $(n>1)$	$\mathbf{b}_{\mathrm{R}}^{\mathbf{A}^n}=\mathbf{b}_{\mathrm{R}}^{\mathbf{A}^n}+\mathbf{b}_{\mathrm{T}}^{\mathbf{A}^{n-1}},$	$\mathbf{b}_{\mathrm{R}}^{\mathbf{P}^n}=\mathbf{b}_{\mathrm{R}}^{\mathbf{P}^n}+\mathbf{b}_{\mathrm{T}}^{\mathbf{P}^{n-1}}$
	Eliminate	$\mathbf{b}_{\mathrm{R}}^{\mathbf{A}^n},\ \mathbf{b}_{\mathrm{R}}^{\mathbf{P}^n}$
if $(n=\mathrm{MPR})$	*Eliminate*	$\mathbf{b}_{\mathrm{T}}^{\mathbf{A}^n},\ \mathbf{b}_{\mathrm{T}}^{\mathbf{P}^n}$

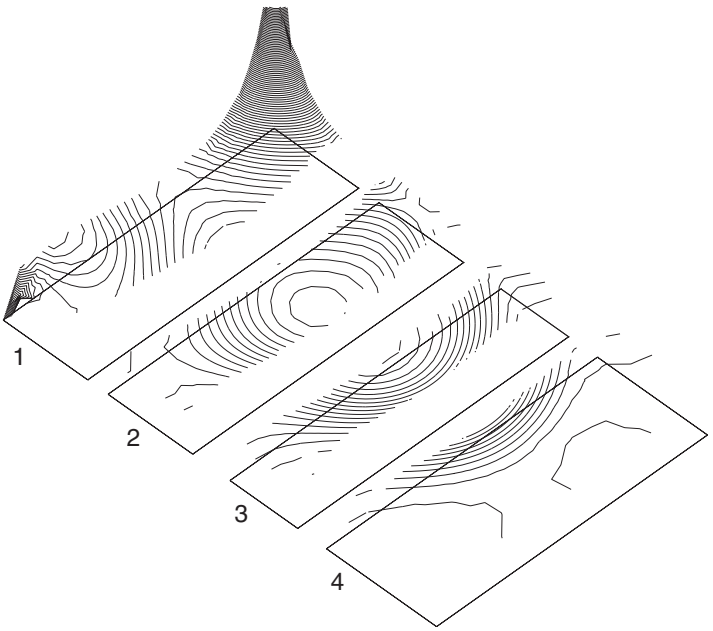


Figure 12. Two-dimensional pressure isobar solution for the driven cavity flow for Reynolds number 400, computed on four satellite processors.

8.3. *Vector substitution*

$$
\begin{array}{lll}
\textit{if } (n = \mathrm{MPR}) & \textit{Substitute} & \mathbf{b}_{\mathrm{T}}^{\mathbf{P}^n},\ \mathbf{b}_{\mathrm{T}}^{\mathbf{A}^n} \\
 & \textit{Substitute} & \mathbf{b}_{\mathrm{R}}^{\mathbf{P}^n},\ \mathbf{b}_{\mathrm{R}}^{\mathbf{A}^n} \\
\textit{if } (n < \mathrm{MPR}) & \mathbf{b}_{\mathrm{T}}^{\mathbf{P}^n} = \mathbf{b}_{\mathrm{R}}^{\mathbf{P}^{n+1}}, & \mathbf{b}_{\mathrm{T}}^{\mathbf{A}^n} = \mathbf{b}_{\mathrm{R}}^{\mathbf{A}^{n+1}} \\
 & \textit{Substitute} & \mathbf{b}_{\mathrm{I}}^{\mathbf{P}^n},\ \mathbf{b}_{\mathrm{I}}^{\mathbf{A}^n}
\end{array}
$$

9. NUMERICAL EXPERIMENTS

The test problem for the exploration of the solution algorithm is the driven cavity problem. An initial cube with corners in the co-ordinates in $(\pm 0.5, \pm 0.5, \pm 0.5)$, is divided in 12 tetrahedra, each with one node in the centre of the cube and the three other nodes on one of the surfaces on the cube. The cavity is then refined by the algorithm given by Reference [4], where each tetrahedron is divided in eight new tetrahedra. This refinement procedure is continued until the wanted degree of refinement is obtained. The computational domain is then divided in the four submeshes shown in Figure 9.

The finite elements are sorted with respect to a point far along the central axis in the y-direction. The far point is chosen far enough to obtain a compact node numbering within each layer of nodes. This ordering ensures that there are no jumps in no numbering between node layers.

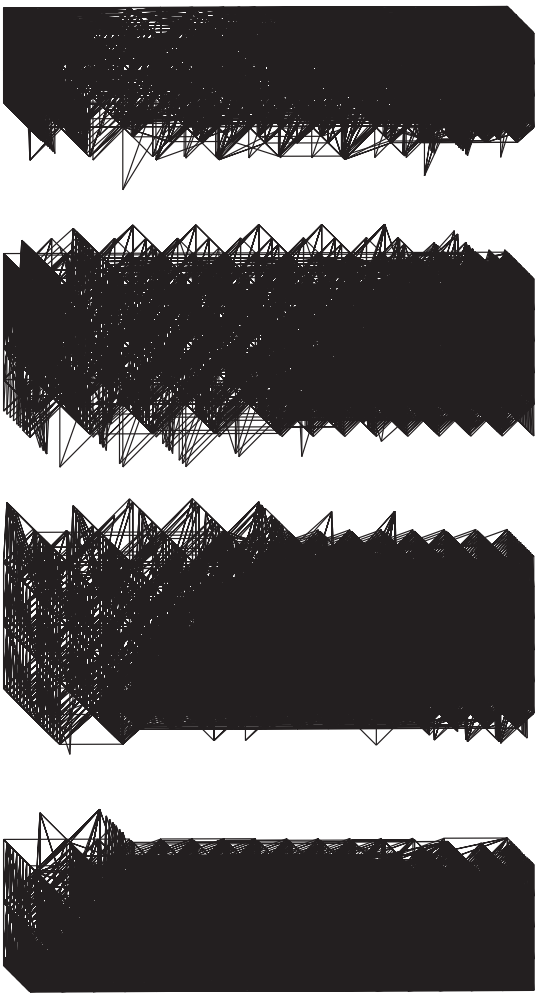


Figure 13. Three-dimensional mesh divided in four subdomains for Reynolds number 400. The mesh has been adaptively refined during the computations.

The boundary conditions for the driven 3D cavity are $\mathbf{u} = [u, v, w] = [umax, 0, 0]$ one the upper surface of the cavity and the no slip condition, $\mathbf{u} = [0, 0, 0]$ on the other surfaces. The Reynolds number

$$Re = \frac{umax\, d}{\mu} \tag{11}$$

is increased or decreased by corresponding increase or decrease of *umax*. A reasonably good initial guess of the solution vector is obtained by scaling of the solution for the previous

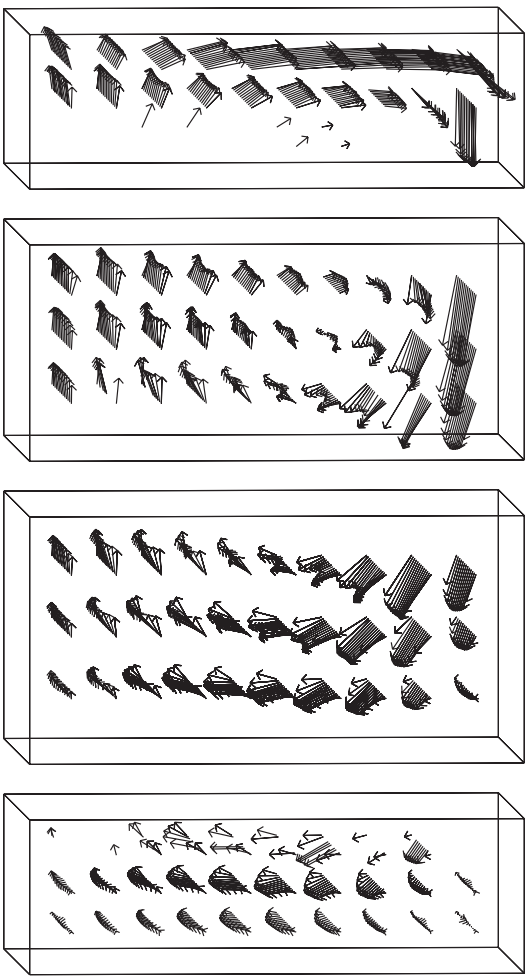


Figure 14. Three-dimensional velocity vector solution for the driven cavity flow for Reynolds number 400 computed on four satellite processors.

Reynolds number by the ratio

$$\mathbf{u}_n = \mathbf{u}_{n-1} \frac{umax_n}{umax_{n-1}} \tag{12}$$

where the n denotes the transition from one Reynolds number to the next.

The number of Newton iterations is fixed to 5 and the convergence criterion for the linear iterations is the norm of residual normalized by *umax* and equal to

$$\frac{|r|}{umax} < \varepsilon_L = 10^{-5} \tag{13}$$

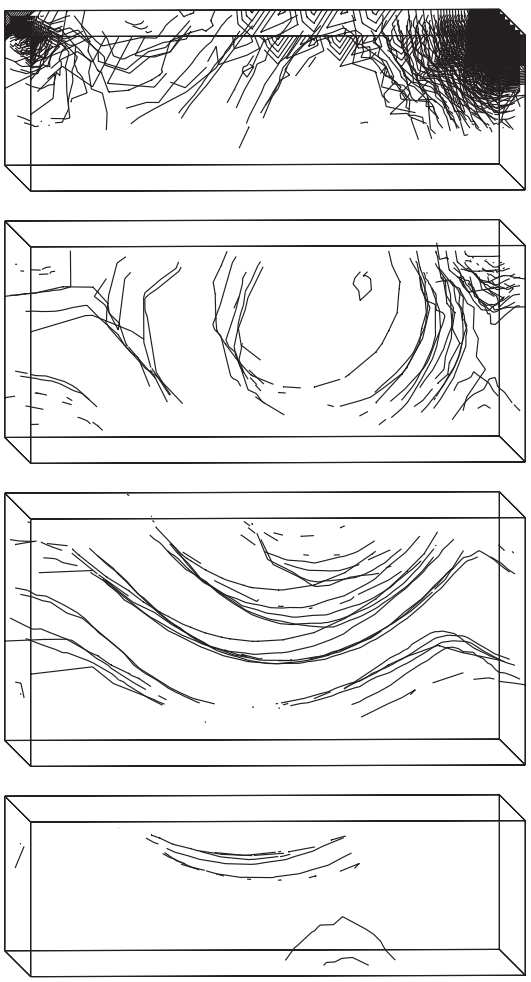


Figure 15. Three-dimensional pressure isobar solution for the driven cavity flow for Reynolds number 400 computed on four satellite processors.

In the parallel processing, $N/4$ elements are distributed to each satellite processor in chronological order. The mesh is adaptively refined and coarsened with the ratio of the convection to the diffusion, the Element Reynolds number, as refinement–recoarsement indicator. Figure 10 shows the velocity vector field computed by four parallel processors and Figure 11 shows the pressure isobars. In Figure 12, the pressure isobars are displayed slightly rotated.

Figure 13 shows the three-dimensional mesh for parallel processing of a three-dimensional driven cavity. The three-dimensional velocity field is displayed in Figure 14 and the pressure isobars are shown in Figure 15.

Figures 16 and 17 show the initialization time of the preconditioner, i.e. is the time for constructing the preconditioner in each satellite processor, for Reynolds number 400 and 800, respectively, as a function of number of satellites. As seen from the figures, the initialization

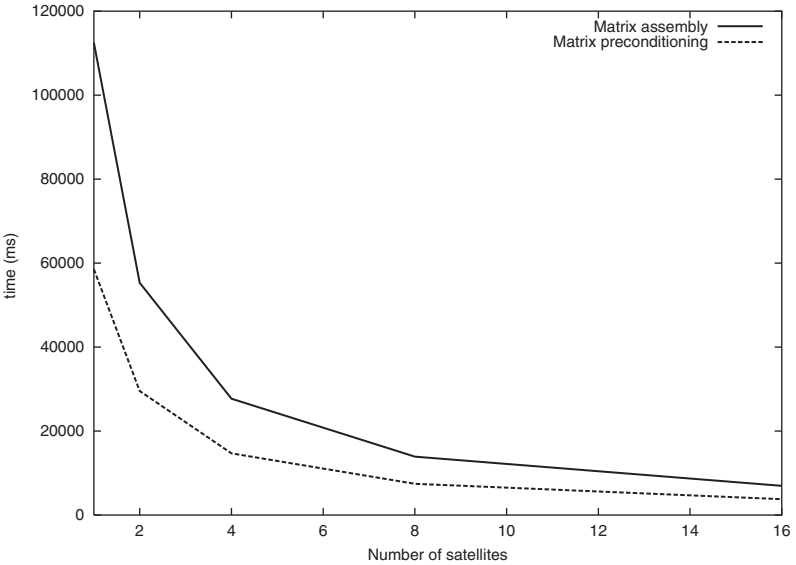


Figure 16. Initialization time for Reynolds number 400.

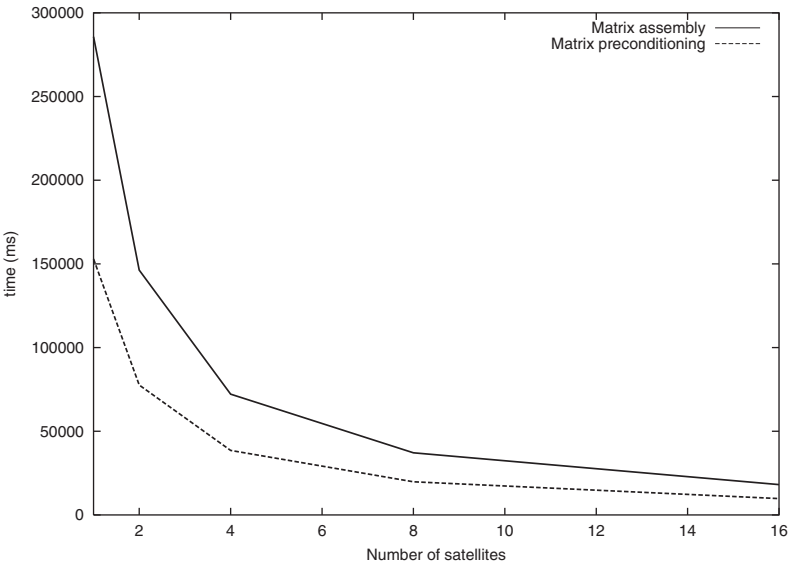


Figure 17. Initialization time for Reynolds number 800.

time decreases as more satellites are added. Figures 18 and 19 show the communication time, computation time and the total iteration time for Reynolds number 400 and 800. The iteration time is the time used in each satellite processor, which is approximately equal and coincides for all the satellite processors. The communication time is the time used in the

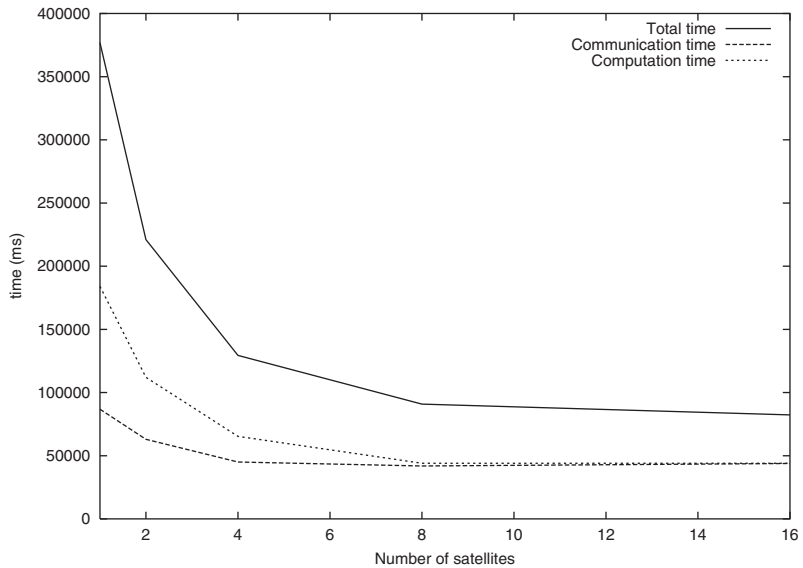


Figure 18. Iteration time for Reynolds number 400.

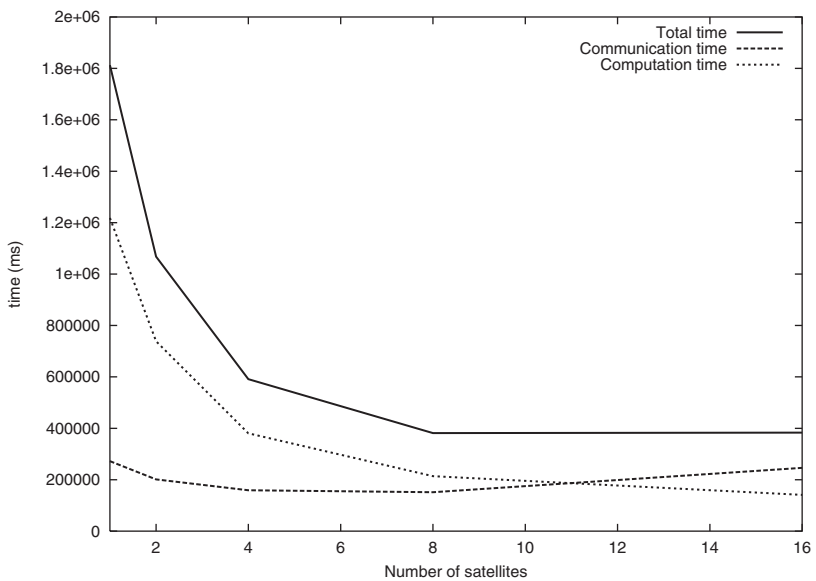


Figure 19. Iteration time for Reynolds number 800.

communication between two processors, which is also approximately equal and coincides for the communication between the satellite processors. The total time is therefore the sum of the iteration time, communication time and the time used by the control processor. The time curves show to flatten versus number of satellite processors. For few finite elements, the

initialization time for the parallel computations and communications is dominant. When the amount of data communicated is less than the package size, the constant package size in the communication will also contribute to the flattening of the communication time. The time curves show that there is no increase in efficiency followed by an increase in the number of satellite processor above 8 for the present mesh.

10. DISCUSSION

Parallel algorithms for the generation of finite element matrices, the matrix–vector product and the full matrix ILU preconditioner have been developed. The behaviour of the algorithms have been explored for computations in two and three dimensions. The efficiency of the algorithms is demonstrated by the measurements of the communication time and computation time for various Reynolds numbers and number of satellite processor. The total solution time for solving specific problems have been shown to decrease with increasing number of satellite processor.

REFERENCES

1. Dahl O, Wille SØ. 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. *International Journal for Numerical Methods in Fluids* 1992; **15**:525–544.
2. Wille SØ, Loula AFD. A priori pivoting in incomplete Gaussian preconditioning for iterative solution of mixed finite element formulation of the Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 2001; **29**:3735–3747.
3. Greaves DM, Borthwick AGL. Hierarchical tree-based finite element mesh generation. *International Journal for Numerical Methods in Engineering* 1999; **45**:447–471.
4. Wille SØ. A structured tri-tree search method for generation of optimal unstructured finite element grids in two and three dimensions. *International Journal for Numerical Methods in Fluids* 1992; **14**:861–881.
5. Wille SØ. An adaptive unstructured tri-tree iterative solver for mixed finite element formulation of the Stokes equations. *International Journal for Numerical Methods in Fluids* 1996; **22**:899–913.
6. Wille SØ, Skipitaris D. A dynamic adaption algorithm for grid, time and satellite processors for nodal finite elements formulations of Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1998; **161**:215–228.
7. Sonneveld P. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 1987; **10**:36–52.
8. Vorst HA van der, Sonneveld PA. More smoothly converging variant of CG-S. *Report 90-50*, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands. ISSN 0922-5641. 1990.
9. Meijerink JA, Vorst HA van der. An iterative solution method for linear systems of which the coefficient matrix is a symmetric *M*—Matrix. *Mathematics of Computation* 1977; **31**:148–162.
10. Tezduyar TE, Behr M, Mittal S, Johnsen AA. Computation of unsteady incompressible flows with the stabilized finite element methods—space-time formulations, iterative strategies and massively parallel implementations. In *New Methods in Transient Analysis*, Smolenski P *et al.* (eds). AMD-vol. 143. ASME: New York, 1992; 7–24.
11. Barragy E, Carey GF, Geijn R van der. Parallel performance and scalability for block preconditioned finite-element (P) solution of viscous flow. *International Journal for Numerical Methods in Engineering* 1995; **38**:1535–1554.
12. Silva RS, Almeida RC, Galeao ACN, Coutinho A. Iterative local solvers for distributed Krylow-Schwarz method applied to convection–diffusion problems. *Computer Methods in Applied Mechanics and Engineering* 1997; **149**:353–362.
13. Howard D, Connolley WM, Rollett JS. Unsymmetric conjugate gradient methods and sparse direct methods in finite element flow simulation. *International Journal for Numerical Methods in Fluids* 1990; **10**:925–945.
14. Kalro V, Tezduyar TE. Parallel iterative computational methods for 3D finite element flow simulations. *Computer Assisted Mechanics and Engineering Sciences* 1998; **25**:173–183.
15. Tezduyar TE, Osawa Y. Methods for parallel computation of complex flow problems. *Parallel Computing* 1999; **5**:2039–2066.

