Mathematical Modelling and Analysis
Volume 4, 1999, pages 124–134

PARALLEL COMPUTATION OF ROTATING FLOWS

L.K. LUNDIN[1], V.A. BARKER[2] and J.N. SØRENSEN[3]

[1] *Los Alamos National Laboratory/CIC-3*

MS B256, Los Alamos, NM 87545, USA

[2] *Department of Mathematical Modelling, Technical University of Denmark*

[3] *Department of Energy Engineering, Technical University of Denmark*

DK-2800 Lyngby, Denmark

E-mail: [1]lkl@lanl.gov, [2]vab@imm.dtu.dk

E-mail: [3]jns@et.dtu.dk

Received September 30, 1999

ABSTRACT

This paper deals with the simulation of 3-D rotating flows based on the velocity-vorticity formulation of the Navier-Stokes equations in cylindrical coordinates. The governing equations are discretized by a finite difference method. The solution is advanced to a new time level by a two-step process. In the first step, the vorticity at the new time level is computed using the velocity at the previous time level. In the second step, the velocity at the new time level is computed using the new vorticity. We discuss here the second part which is by far the most time-consuming. The numerical problem is that of solving a singular, large, sparse, over-determined linear system of equations, and the iterative method CGLS is applied for this purpose. We discuss some of the mathematical and numerical aspects of this procedure and report on the performance of our software on a wide range of parallel computers.

1. INTRODUCTION

Rotating flows play an important role in nature as well as in industry. In nature they are found, for example, in cyclones and tornados. In industry they are used in combustion chambers to mix fuel and air, and in cyclone-separators where the rotation contributes to the dissolution of phases of different density. Rotating flows have a number of interesting characteristics, such as the ability to maintain inertia waves and the propagation of vortex break-down.

The physical system considered in this paper is that of a cylinder of radius

R and height H containing a fluid with kinematic viscosity ν. A no-slip boundary condition applies everywhere on the cylinder surface. The motion of the fluid depends on the initial motion of the fluid and the motion of the fluid on the boundary of the cylinder. Let D denote the interior of the cylinder and ∂D its boundary. A cylindrical coordinate system is placed in the cylinder with origin at the center of the bottom surface. Thus the typical point of D is (r, θ, z), where $0 \leq r < R$, $0 \leq \theta < 2\pi$ and $0 < z < H$.

Our mathematical model consists of the Navier-Stokes equations in the velocity-vorticity formulation,

$$\frac{\partial \vec{\omega}}{\partial t} + \underline{\nabla} \times (\vec{\omega} \times \vec{v}) = \frac{1}{R_e} \underline{\nabla} \times \underline{\nabla} \times \vec{\omega}, \tag{1.1}$$

$$\underline{\nabla} \times \vec{v} = \vec{\omega}, \tag{1.2}$$

$$\underline{\nabla} \cdot \vec{v} = 0 \tag{1.3}$$

for $(r, \theta, z) \in D$, $t > 0$. Here $R_e = Ru_0/\nu$, where u_0 is a problem-dependent characteristic speed. The unknowns of the problem are the velocity vector $\vec{v}(r, \theta, z, t) \in \mathbb{R}^3$ and the vorticity vector $\vec{\omega}(r, \theta, z, t) \in \mathbb{R}^3$. The components of $\vec{v}$ and $\vec{\omega}$ are denoted $\vec{v} = (v_r, v_\theta, v_z) = (U, V, W)$ and $\vec{\omega} = (\omega_1, \omega_2, \omega_3)$, respectively. It is assumed that $\vec{v}$ (and hence $\vec{\omega}$ via (1.2) is known everywhere in D for $t = 0$ and everywhere on ∂D for $t > 0$.

Our procedure for solving problem (1.1)-(1.3) is to advance the solution from time level t_n to t_{n+1} in two steps: First, the vorticity vector $\vec{\omega}^{n+1}$ is found by solving (1.1) with given velocity vector $\vec{v}^n$. Then, the velocity vector $\vec{v}^{n+1}$ is found by solving the system (1.2)-(1.3) with given vorticity vector $\vec{\omega}^{n+1}$. Since the second step has proven to be the most time consuming by far, we henceforth confine our attention to that.

2. THE DISCRETE SYSTEM

Equations (1.2)-(1.3) in cylindrical coordinates are:

$$\frac{\partial (rU)}{\partial r} + \frac{\partial V}{\partial \theta} + r \frac{\partial W}{\partial z} = 0, \tag{2.1}$$

$$\frac{\partial W}{\partial \theta} - r \frac{\partial V}{\partial z} = r\omega_1, \tag{2.2}$$

$$\frac{\partial U}{\partial z} - \frac{\partial W}{\partial r} = \omega_2, \tag{2.3}$$

$$\frac{\partial (rV)}{\partial r} - \frac{\partial U}{\partial \theta} = r\omega_3. \tag{2.4}$$

Viewing this system as an independent problem, the correct boundary condition is the specification of the normal component of the velocity vector on ∂D. (See [4]). Consequently, the tangential velocity component on ∂D is part of

the solution. Hence we have a possible conflict, since the boundary condition for problem (1.1)-(1.3) also determines the tangential velocity component on ∂D. Experience has shown that the following approach to discretizing the Cauchy-Riemann equations is effective [5]:

- The correct tangential velocity component on ∂D is used in the discretization of (2.1).
- Difference equations arising from (2.2)-(2.4) are not used in those cases where one or more grid points belong to ∂D.

To discretize the cylinder we divide it into cells with vertices

$$(r_i, \theta_j, z_k) = (i\Delta r, j\Delta\theta, k\Delta z)$$

for $i = 0, 1, \ldots, N_1$, $j = 0, 1, \ldots, N_2 - 1$, $k = 0, 1, \ldots, N_3$, where $\Delta r = R/N_1$, $\Delta\theta = 2\pi/N_2$, and $\Delta z = H/N_3$. (See Fig. 1.)

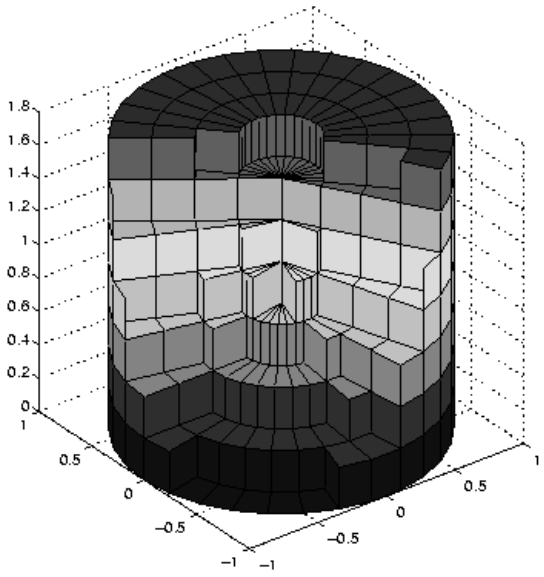

Figure 1. A portion of the discretized cylinder.

The cell vertices define the grid that is used in the discretization of (2.1)-(2.4). In introducing a notation for the unknown velocity components it is necessary to distinguish between grid points off the center axis ($i > 0$) and those on the axis ($i = 0$). In the first case the typical velocity components are denoted $(U_{i,j,k}, V_{i,j,k}, W_{i,j,k})$. In the second case the radial and tangential

components of the velocity vector are not defined. Here we express the velocity vector in terms of Cartesian coordinates

$$\vec{v}_{r=0} = (U_0, V_0, W_0)$$

and discretize these by

$$\vec{v}_{r=0} = (U_{0\ k}, V_{0\ k}, W_{0\ k}),\ k = 0, 1, \ldots, N_3.$$

In the difference equations that involve grid points at $r = 0$ there is a need for the components of the horizontal velocity in the directions θ and $\theta+\pi/2$. These are expressed by

$$\begin{aligned} U_{0,j,k} &= U_{0\ k}\cos\theta_j + V_{0\ k}\sin\theta_j, \\ V_{0,j,k} &= V_{0\ k}\cos\theta_j - U_{0\ k}\sin\theta_j, \end{aligned}$$

where $j = 0, 1, \ldots, N_2 - 1,\ k = 0, 1, \ldots, N_3$. (See Fig. 2.)

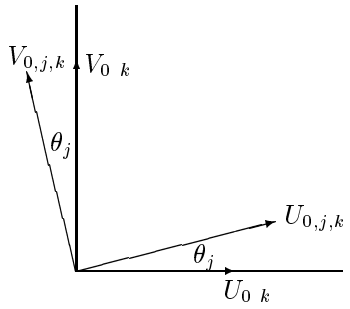

Figure 2. Horizontal velocity at $i = 0$.

Space limitations prohibit a complete description of the difference equations here, (see [2] and [3] for details), but the basic ideas are the following: Equation (2.1) is discretized on every cell by balanced finite difference approximations that yield a truncation error of second order with respect to the step sizes. To illustrate, the discretization of the first term in (2.1) with respect to the cell in Fig. 3 is

$$\begin{aligned} \frac{\partial(rU)}{\partial r} \approx \frac{1}{4\Delta r}\{&(rU)_{i+1,j,k} + (rU)_{i+1,j,k+1} + (rU)_{i+1,j+1,k+1} \\ &+(rU)_{i+1,j+1,k} - (rU)_{i,j,k} + (rU)_{i,j,k+1} + (rU)_{i,j+1,k+1} + (rU)_{i,j+1,k}\} \end{aligned}$$

where, for example, $(rU)_{i+1,j,k} = r_{i+1}U_{i+1,j,k}$. Cells which touch the center axis are 5-sided and require special treatment. Regarding equations (2.2)-(2.4), each of these is discretized on just one of the cell surfaces. For example,

(2.2) is discretized on the surface $r = r_i$ (see Fig. 3) as follows:

$$\begin{aligned} &\frac{1}{\Delta\theta}\left\{\frac{1}{2}\left[W_{i,j+1,k} + W_{i,j+1,k+1}\right] - \frac{1}{2}\left[W_{i,j,k} + W_{i,j,k+1}\right]\right\} \\ &\quad -\frac{r_i}{\Delta z}\left\{\frac{1}{2}\left[V_{i,j,k+1} + V_{i,j+1,k+1}\right] - \frac{1}{2}\left[V_{i,j,k} + V_{i,j+1,k}\right]\right\} \\ &= \frac{r_i}{4}\left[(\omega_1)_{i,j,k} + (\omega_1)_{i,j,k+1} + (\omega_1)_{i,j+1,k+1} + (\omega_1)_{i,j+1,k}\right]. \end{aligned}$$

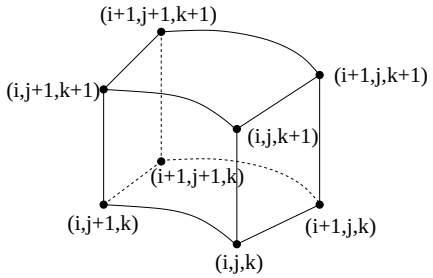

Figure 3. A typical cell.

The total set of difference equations constitutes an overdetermined linear algebraic system which, after an ordering of the unknowns and equations, can be expressed in the matrix-vector form

$$A\mathbf{x} = \mathbf{b}. \tag{2.5}$$

Let N denote the number of grid points in the cylinder. The dimensions of A, $\mathbf{x}$ and $\mathbf{b}$ are, respectively, $m \times n$, $n \times 1$ and $m \times 1$, where $m \approx 4N$ and $n \approx 3N$. Our chosen orderings of unknowns and equations proceed upward through the cylinder, level by level, producing a matrix A with the sparsity pattern shown in Fig. 4. This matrix can be partitioned in a form involving only four distinct blocks: A_1, A_2, A_3 and A_4. Blocks A_1 and A_4 appear only at the ends of the main diagonal, while block rows consisting of A_2 and A_3 are repeated between these.

An important property of A is that when N_2 is even then its null space

$$N(A) = \{\mathbf{x} \in R^n | A\mathbf{x} = \mathbf{0}\}$$

has dimension $N_1 - 2$. Further, for each of the values $i = 2, 3, \ldots, N_1 - 1$ there is a vector in null(A) with the nonzero components

$$W_{i,j,k} \quad = \quad \begin{cases} 1, & j+k \text{ even}, \\ -1, & j+k \text{ odd}. \end{cases}$$

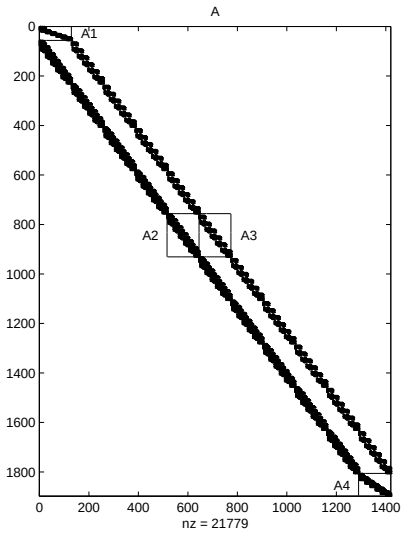

Figure 4. Sparsity pattern of the matrix A.

These vectors comprise an orthogonal basis of $N(A)$.

Overdetermined systems are generally inconsistent; i.e., they have no solution. However, (2.5) is "almost" consistent in the sense that the minimum least-squares error

$$e = \min_{\mathbf{x} \in R^n} \| \mathbf{b} - A\mathbf{x} \| . \tag{2.6}$$

where $\| \mathbf{y} \| = (\mathbf{y}^T \mathbf{y})^{1/2}$, is at the level of the truncation errors in the difference approximations used above. This is a consequence of the fact that problem (2.1)-(2.4) is itself consistent, being satisfied by the solution of problem (1.1)-(1.3).

To obtain a completely consistent algebraic problem we replace (2.5) by its normal system,

$$A^T A\mathbf{x} = A^T \mathbf{b}, \tag{2.7}$$

$A^T A$ is a symmetric semipositive definite matrix of order n. This system has the solution set

$$S(A, \mathbf{b}) = \{\mathbf{x} \in R^n | \mathbf{x} = \widehat{\mathbf{x}} + \mathbf{y},\ \mathbf{y} \in N(A)\},$$

where $\widehat{\mathbf{x}}$ is the unique solution of (2.7) orthogonal to $N(A)$. It is easy to show that $\| \mathbf{b} - A\mathbf{x} \| = e$ for all $\mathbf{x} \in S(A, \mathbf{b})$, where e is given by (2.6).

To obtain a well-formulated problem we need a criterion for selecting precisely one vector from $S(A, \mathbf{b})$. An analysis, which we omit, shows that if $\mathbf{x}$ is taken to be the grid values of a given, smooth velocity field (U, V, W), then the component of $\mathbf{x}$ in $N(A)$ goes to zero very rapidly as the grid is refined. On

the basis of this observation we identify $\widehat{\mathbf{x}}$ as the physically relevant solution of (2.7).

3. A CONJUGATE GRADIENT METHOD

The normal system (2.7) may be solved iteratively by the conjugate gradient method, a procedure that has several formulations. The one adopted here, CGLS ([1]), is shown below. The floating-point operation counts are given for each step:.

a)	$p^{(k+1)} = s^{(k)} + \beta_k p^{(k)}$	$2n$ FLOPs
b)	$q^{(k+1)} = Ap^{(k+1)}$	$32n$ FLOPs
c)	$\alpha_{k+1} = \frac{\gamma_k}{q^{(k+1)T} q^{(k+1)}}$	$\frac{8}{3}n$ FLOPs
d)	$r^{(k+1)} = r^{(k)} - \alpha_{k+1} q^{(k+1)}$	$\frac{8}{3}n$ FLOPs
e)	$x^{(k+1)} = x^{(k)} + \alpha_{k+1} p^{(k+1)}$	$2n$ FLOPs
f)	$s^{(k+1)} = A^T r^{(k+1)}$	$32n$ FLOPs
g)	$\gamma_{k+1} = s^{(k+1)T} s^{(k+1)}$	$2n$ FLOPs
h)	$\beta_{k+1} = \frac{\gamma_{k+1}}{\gamma_k}$	1 FLOPs

The iterations are initialized by setting $\beta_1 = 0$, $p^{(1)} = \vec{0}$, $r^{(1)} = b - Ax^{(1)}$ and, with $k = 0$, executing steps f) and g).

Regarding the choice of the initial vector $\mathbf{x^{(1)}}$, when the conjugate gradient method is applied to (2.7) then the solution it finds is the vector in $S(A, \mathbf{b})$ with the same null space component as $\mathbf{x^{(1)}}$. Since we want to find the vector in $S(A, \mathbf{b})$ with no null space component, an obvious choice of initial vector is $\mathbf{x^{(1)}} = \mathbf{0}$. Recall, however, that the computational problem under discussion is part of a time-stepping procedure. Since the velocity at the new time level will typically be close to the velocity at the previous time level, one would like to use the previous velocity as $\mathbf{x^{(1)}}$ in the new CGLS computation. Now this is possible provided only that the velocity computed at the first time level $t = t_1$ has no null space component.

4. PERFORMANCE

The procedure outlined above is the basis for a computer program that has been run over a wide range of machines. For implementation details see [3]. As pointed out in [5], the rate of convergence of the CGSL iterations can be significantly improved by equilibrating the rows of (2.5), a variant we denote rCGLS. Further, another variant, rCGLS-1, is sometimes effective in reducing communication costs of distributed computing. (See [3]).

For our experiments and simulations we have used the computers shown in the table below. Figs. 5 and 6 show the parallel speedup achieved on a number of computers. We have obtained the speed-ups by measuring for each computer the elapsed wall-clock time, T_n, spent performing 100 iterations of rCGLS using n processors. The parallel speed-up is then computed as T_1/T_n.

In all experiments the problem size is $(N_r, N_\theta, N_z) = (63, 66, 129)$ leading to a matrix with dimensions app. $1.5 \cdot 10^6 \times 2.0 \cdot 20^6$ and about 700 distinct non-zeros. Calculations are done in single precision[1].

Table 1.

Computer	Short Name	CPU's	Peak [MFLOP/s]	T_n
Fujitsu VPP700	VPP	32	70400	1.00*
Cray T3E/450	T3E450	64	57600	2.75*
Cray T3E/300	T3E300	64	38400	3.67
NEC SX-4	NEC	16	35200	1.97*
SGI Origin 2000/250	O2K250	64	32000	1.41*
Cray T3D/150	T3D	64	19200	6.73
IBM SP-2 SC/120	SP120	32	15360	4.08
SGI Origin 2000/195	O2K195	32	12480	2.68
Convex SPP-2000	SPP	16	11520	7.78
IBM SP-2 Thin/67	SPthin	32	8512	8.62*
SGI PC R8000	R8K	16	5760	5.84
IBM SP-2 Wide/67	SPwide	16	4256	8.57
SGI PC R4000	R4K	16	4000	11.26
Meiko CS-2	CS2	14	1400	22.67
SGI PC R10000	R10K	6	2328	23.90
CRAY T90	T90	1	1920	65.54
SUN HPC450	SUN	2	1000	45.67
CRAY C92A	C90	1	960	29.02
HP 9000/180	HP	1	720	166.24
IBM SP-2 SC/135	SP135	1	540	49.10
CRAY Y-MP	YMP	1	330	62.72
Pentium PRO/200	PC	1	200	370.76

Each of the timings is the fastest of rCGLS and rCGLS-1. T_n in the table is normalized to T_n on the fastest computer, the VPP-32. The actual FLOP-rate obtained on this computer is 9012 MFLOP/s. Timings T_n obtained with rCGLS-1 are marked with * in the table.

Since the maximum number of processors per computer varies from 6 to 64, the results are shown in two plots: one with measurements with at most 16 processors and another with measurements with at least 16 processors.

Some comments:

- With SPthin the measurements on 8, 16 and 32 processors are with rCGLS-1. For the NEC this is the case for 14 and 16 processors. With the NEC there is no speed-up from 14 to 16 processors with rCGLS because the work is too small compared with the time spent in the inner product communication. With our largest test case, $(N_r, N_\theta, N_z) = (127, 256, 193)$, the speed-up on NEC-16 increases to 80% of the number of processors. With

[1]Single-precision on a Cray and NEC is equal to double precision on the other tested computers.

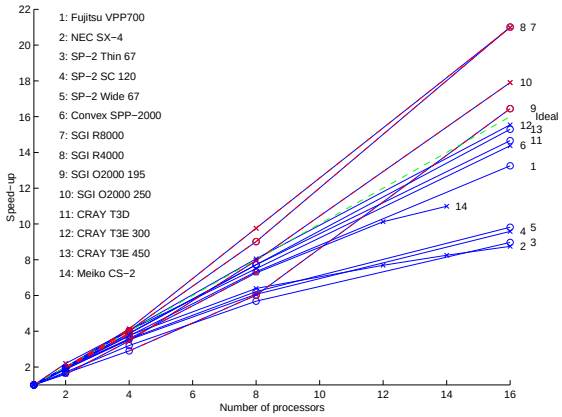

Figure 5. Parallel speed-up, at most 16 processors.

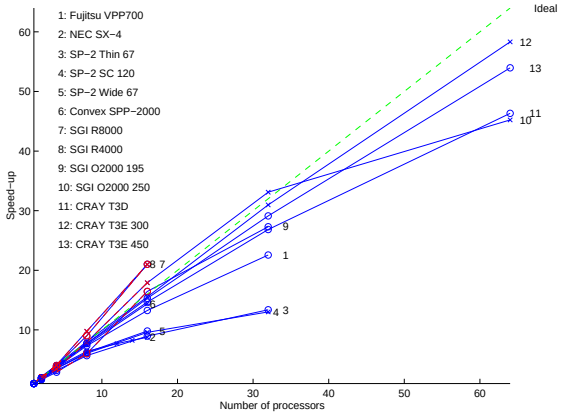

Figure 6. Parallel speed-up, at least 16 processors.

this resolution the speed-up on the VPP-32 increases to from 76% to 86% of the number of processors.

- R4K, R8K, O2K195 and O2K250 exhibit super-linear speed-up even with this rather large system. This happens because each CPU has a much larger cache than the other computers and because of the low storage requirement for A. There is however no super-linear speed-up with our larger test cases.
- In the described test case the load-balance is only optimal with 2^p processors. This explains why the speed-up of the NEC and the CS2[2] at 12 and 14 processors is low.
- The speed-up of the IBM SP-2 (all three variants) is quite low. This may be because the SP-2 nodes compute quite fast. The timings for 1 processor

[2]Tests are conducted with one CPU per node.

vary with less than 1%, while the timings with 32 processors vary with more than 20%. We therefore believe the relatively poor speed-up is also caused by the fact that the inter-processor communication in our SP-2 experiments is competing for the bandwidth of the inter-processor connection against the communication between processors of other users.

- With T3D and T3E450 the speed-up from 32 to 64 processors is modest and with O2K250 the same speed-up is poor. With 64 processors there are only two levels per processor, and therefore not much computation to overlap with the communication. This reveals a weakness of the chosen parallelization strategy: with a massively parallel computer a large number of levels is necessary to obtain a good speed-up. There us no speed-up curve for the T3E300 because this machine does not have enough memory to run the code on a single processor. The speed-up from 2 to 64 processors is however 29.7 which is 92.7% of a perfect speed-up. Since the processors of this computer operate at a lower clock than those of the T3E450, the communication overhead is relatively smaller, which is why rCGLS-1 is not faster on this machine.

It is common to measure the parallel scalability of an algorithm by measuring the scaled speed-up. Scaled speed-up is obtained by increasing the number of processors while keeping the amount of work per processor constant. With a perfectly parallel algorithm such an experiment should produce timings that are independent of the number of processors. An easy way to construct such an experiment for our problem would be to assign a fixed number of k-levels to each processor and then simply increase the vertical resolution along with the number of processors. However, due to the special cases of our Cauchy-Riemann equations at the first and last k-levels this approach would result in an uneven amount of work per processor, making it difficult to analyze scaled speed-up. We have therefore not pursued this.

REFERENCES

[1] A. Björck. *Numerical Methods for Least Squares Problems.* SIAM, 1996.

[2] L. K. Lundin. Computing the velocity of a rotating flow. *Parallel Computing*, **24** (14), 1998, 2021 – 2034.

[3] L. K. Lundin. *Parallel Computation of Rotating Flows.* (Ph.D. thesis), Technical Report IMM-PHD-1998-49, Department of Mathematical Modelling, Technical University of Denmark, 1998.

[4] C. G. Speziale. On the advantages of the vorticity-velocity formulation of the equations of fluid dynamics. *J. Comp. Phys.*, **73** (2),1987, 476 – 480.

[5] S. N. Sørensen, W. Z. Shen and M. O. L. Hansen. Vorticity-velocity formulation of the 3D Navier-Stokes equations in cylindrical coordinates. *J. Comp. Phys.* , (submitted).

LYGIAGRETIEJI BESISUKANČIŲ SKYSČIŲ MODELIAVIMO ALGORITMAI

L.K. LUNDIN, V.A. BARKER, J.N. SORENSEN

Darbe sprendžiamas trimatis Navje-Stokso uždavinys, kai lygtys formuluojamos cilindrinėje koordinačių sistemoje, o nežinomaisiais yra greičio komponentės ir sūkurys. Diferencialinės lygtys aproksimuojamos baigtinių skirtumų metodu. Vieną algoritmo žingsnį sudaro du etapai. Pirmajame etape panaudodami senas greičio komponenčių reikšmes apskaičiuojame sūkurio reikšmę naujuoju laiko momentu. Antrajame etape apskaičiuojamos naujos greičio komponenčių reikšmės. Straipsnyje didžiausias dėmesys skiriamas antrajam etapui, kadangi šios algoritmo dalies realizacija reikalauja daugiausia skaičiavimų. Sprendžiama perpildyta tiesinių lygčių sistema, kurios matrica yra siguliari, reta ir didelės dimensijos. Naudojamas CGLS iteracinis metodas. Aptariamas lygiagretusis algoritmas ir pateikiami rezultatai skaičiavimo eksperimentų, kurie buvo atlikti su įvairaus tipo lygiagrečiaisiais kompiuteriais.