Parallel and Distributed Computing Practices
Volume 5, Number 3, pp. 321-329

ISSN 1097-2803
© 2002 Nova Science Publishers, Inc.

NUMERICAL SCHEMES FOR FLUID FLOW AND HEAT TRANSFER IN MEDICAL SIMULATIONS*

MARJAN ŠTERK†, ROMAN TROBEC†, AND MATEJ PRAPROTNIK‡

Abstract. Incompressible fluid flow is governed by the Navier-Stokes equation, which, together with the diffusion and continuity equations, forms a coupled system of partial differential equations that have to be solved to simulate the fluid dynamics. We describe a finite difference scheme and boundary conditions used to solve the system of partial differential equations on general 3-dimensional domains with explicit integration in time. The most computationally intensive part is the pressure equation that requires the solution of a sparse linear system in each time-step of the simulation. Various iterative methods for the solution of the linear system are tested and compared among which the multigrid method outperforms others. Some test examples are given to prove the validity of the simulation results. The paper concludes with an analysis of parallel computational complexity of the SOR method and parallelization strategy for the multigrid method.

Key words. fluid flow, heat transfer, finite difference, sparse linear systems, parallel computation

AMS subject classifications. 65N06, 65Y05

1. Introduction.

Realistic simulations of physical phenomena require an increasing number of details in the mathematical model used, e.g. inhomogeneous domains of general geometry. Medical simulations, for example heart cooling during a surgery, include flow of blood through vessels, heat transfer through living tissues, metabolism of muscles that produce additional heat, etc. A human heart is a body of irregular shape. Heart tissue is inhomogeneous, composed of different substances [13]. The real physical situation can be modeled by a set of coupled partial differential equations (PDEs) that describe the temperature and the velocity in the domain of interest as a function of place and time.

The basic equation that governs the heat transfer is known as the *heat conduction equation*, which is derived for example in [8]. For isotropic substances it can be expressed as

$$\frac{\partial T}{\partial t} = \frac{1}{\rho c_p}\nabla \cdot (\lambda(\mathbf{r})\nabla T) - (\mathbf{v}\cdot\nabla)T\ , \tag{1.1}$$

where T is the temperature, ρ, c_p, and λ are the substance constants, $\mathbf{r}$ is the position vector and $\mathbf{v}$ the velocity.

The equation (1.1) describes for example the heat transfer of an isolated organ. But usually the organ of interest is surrounded by moving air, cooling water and other tissues. Fluid flow can be modeled by the *Navier-Stokes equation* [2, 7]

$$\frac{\partial \mathbf{v}}{\partial t} = \frac{\eta}{\rho}\nabla^2\mathbf{v} - \frac{\nabla p}{\rho} - (\mathbf{v}\cdot\nabla)\mathbf{v} + \mathbf{a}\ . \tag{1.2}$$

In the case of non-uniform temperature, the density ρ of liquid is also non-uniform, resulting in a buoyancy acceleration $\mathbf{a} = \frac{\rho(T)-\rho_0}{\rho_0}\mathbf{g}$, where ρ_0 is the mean density and $\mathbf{g}$ the gravitational acceleration.

Initial values are needed for $\mathbf{v}$, T and p. Boundary conditions for a fixed wall prescribe that the liquid velocity and the normal component of pressure gradient are zero. Transient conditions for boundaries between two substances, e.g. water and air, require continuous temperature field [2, 8].

The *continuity equation* for a mass flow has to be valid in the whole simulated domain [2]. For an incompressible fluid it can be expressed as

$$\nabla\cdot\mathbf{v} = 0\ . \tag{1.3}$$

*This research was partially funded under Grant No. **P0-503-0106** by the Ministry of Education, Science and Sports of Slovenia.

†Department of Digital Communication and Computer Networks, Jožef Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia (`roman.trobec@ijs.si`).

‡National Institute of Chemistry, Hajdrihova 19, SI-1001 Ljubljana, Slovenia.

To couple this equation with (1.2) we used the scheme due to Hirt and Cook [6, 2]. The new velocities are calculated from (1.2) by the explicit Euler's integration

$$\mathbf{v}^{(t+\Delta t)} = \mathbf{v}^{(t)} + \Delta t \frac{\partial \mathbf{v}}{\partial t}\left(\mathbf{v}^{(t)}, T^{(t)}, p^{(t+\Delta t)}\right) = \mathbf{v}^* - \frac{\Delta t}{\rho}\nabla p_\Delta \ ,$$

denoting the pressure correction in each time-step by p_Δ, i.e. $p^{(t+\Delta t)} = p^{(t)} + p_\Delta$, and writing $\mathbf{v}^*$ for everything on the right hand side of (1.2) that is not related to p_Δ. We obtain, applying (1.3) to $\mathbf{v}^{(t+\Delta t)}$

$$\nabla^2 p_\Delta = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{v}^*. \tag{1.4}$$

Equation (1.4) has a standard Poisson form and determines p_Δ for each time-step.

The rest of the paper is organized as follows. In the next section the procedure for the numerical solution of the above system of PDEs is introduced and its numerical stability is analyzed. Section 3 focuses on the solution of the pressure correction equation (1.4) and its computational complexity. In Section 4, the simulation results are shown for two instructive test examples. The paper concludes with a discussion on parallelization and speedup on a computer cluster.

2. Numerical Solution. The coupled system of PDEs introduced in Section 1 is solved with finite differences using explicit Euler's method [5]. From equations (1.2), (1.4), and (1.1) the following integration sequence is derived:

Step 1: $\mathbf{v}^* = \mathbf{v}^{(t)} + \Delta t\left[\frac{\eta}{\rho}\nabla^2\mathbf{v}^{(t)} - \left(\mathbf{v}^{(t)}\cdot\nabla\right)\mathbf{v}^{(t)} + \mathbf{g}\frac{\rho(T^{(t)}) - \rho_0}{\rho_0} - \frac{1}{\rho}\nabla p^{(t)}\right]$,

Step 2: p_Δ is calculated from $\mathbf{v}^*$ using (1.4),

Step 3: $\mathbf{v}^{(t+\Delta t)} = \mathbf{v}^* - \frac{\Delta t}{\rho}\nabla p_\Delta$,

Step 4: $p^{(t+\Delta t)} = p^{(t)} + p_\Delta$,

Step 5: $T^{(t+\Delta t)} = T^{(t)} + \Delta t\left[\frac{\lambda}{c_p\rho}\nabla^2 T^{(t)} - \left(\mathbf{v}^{(t+\Delta t)}\cdot\nabla\right)T^{(t)}\right]$.

It will be shown that great care must be taken when modeling the system, choosing finite difference schemes, boundary conditions and time-step.

2.1. Model Data and Boundary Conditions. The model domain is discretized to cubes of size $\Delta x \times \Delta x \times \Delta x$ and described by pressure, temperature and velocity. A 2-dimensional example is shown in Figure 2.1. Each cube contains either fluid (shown in white) or non-fluid (i.e. solid, shown in gray). For each fluid cube, the temperature and pressure are defined in the middle of the cube (integer coordinates in Figure 2.1). Each velocity component is defined on faces perpendicular to that component [2], i.e. at points with fractional x, y or z coordinate for v_x, v_y and v_z, respectively. For solid cubes, the pressure is undefined and the velocity is 0 on all six faces. Border solid cubes also have constant temperatures.

Only incompressible fluid flow is simulated at this stage. To approximate the effect of ambient air flow on heat transfer, the air temperatures are only calculated according to (1.1) in a thin layer next to other substances. The rest of the air remains at constant ambient temperature.

2.2. Finite Difference Schemes. The finite difference schemes for the diffusion equation and inhomogeneous substances are described in detail in [9]. The forward difference formula is used for all time derivatives because explicit Euler's method is used for integration [4]. The described mesh offset enables the use of central differences for most spatial derivatives. However, some terms require careful analysis. For example, using the central difference for the term $(\mathbf{v}\cdot\nabla)\mathbf{v}$ of the Navier-Stokes equation introduces oscillations into the solution because the velocity correction at odd points only

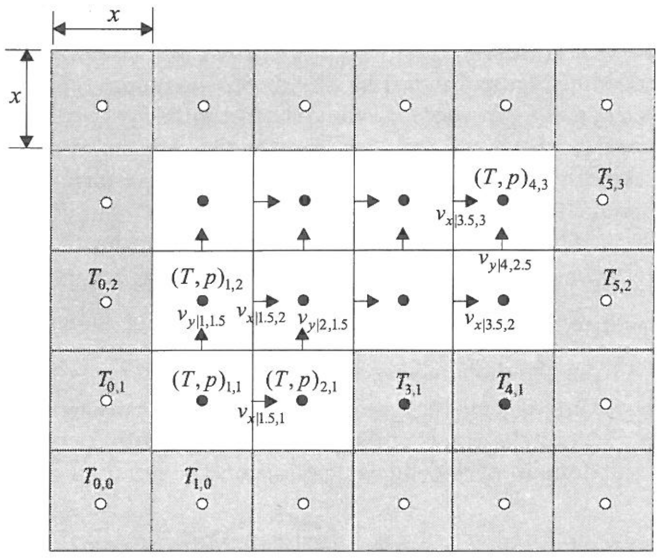

FIG. 2.1. *Temperature/pressure and velocity mesh.*

depends on the velocities at even points, and vice versa. Instead, we use the upwind formula [5]

$$\left.\frac{\partial v_y}{\partial x}\right|_{x,y+\frac{1}{2},z} = \begin{cases} \dfrac{v_{y|x+1,y+\frac{1}{2},z} - v_{y|x,y+\frac{1}{2},z}}{\Delta x}, & v_{x|x,y+\frac{1}{2},z} < 0, \\ \dfrac{v_{y|x,y+\frac{1}{2},z} - v_{y|x-1,y+\frac{1}{2},z}}{\Delta x}, & v_{x|x,y+\frac{1}{2},z} \geq 0, \end{cases} \tag{2.1}$$

shown graphically in Figure 2.2.

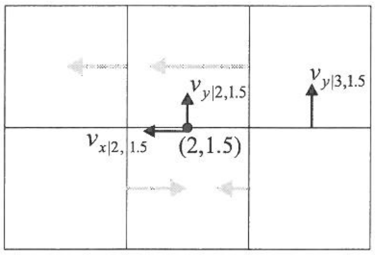

FIG. 2.2. *To calculate $\frac{\partial v_y}{\partial x}$ at the black point, v_x at that point is first interpolated from the neighboring v_x values (shown in gray). The value obtained is negative (the fluid flows to the left), therefore central and right values of v_y are used in the upwind formula (2.1).*

Special schemes have to be used at boundaries for some derivatives, for example $\left.\frac{\partial^2 v_x}{\partial y^2}\right|_{(3.5,3)}$ in Figure 2.1. The boundary condition $v_{x|3.5,3.5} = 0$ is approximated by setting $v_{x|3.5,4} = -v_{x|3.5,3}$. The formula for the approximation of $\frac{\partial^2 v_x}{\partial y^2}$ at the mesh point $\left(x+\frac{1}{2}, y, z\right)$ with a solid cube above it is thus

$$\begin{aligned} \left.\frac{\partial^2 v_x}{\partial y^2}\right|_{x+\frac{1}{2},y,z} &= \frac{v_{x|x+\frac{1}{2},y+1,z} - 2v_{x|x+\frac{1}{2},y,z} + v_{x|x+\frac{1}{2},y-1,z}}{\Delta x^2} \\ &= \frac{-3v_{x|x+\frac{1}{2},y,z} + v_{x|x+\frac{1}{2},y-1,z}}{\Delta x^2}. \end{aligned} \tag{2.2}$$

2.3. Stability Conditions. The space-step Δx is usually affected by the required spatial resolution of a specific application. The time-step Δt must be chosen accordingly in order to maintain the stability restrictions. Independent stability analysis of PDE parts was applied, which is simple but only gives an estimation of the required stability conditions. Exact analysis is not needed in practice because the time-step is also limited by the desired accuracy. Experiments show that in our case accuracy limits the time-step more severely than stability.

The propagation terms $(\mathbf{v}\cdot\nabla)\mathbf{v}$ and $(\mathbf{v}\cdot\nabla)T$ require the domain of dependence stability conditions, i.e. $\Delta t \le \frac{\Delta x}{v_{\max}}$. All other terms can be rewritten in a matrix form $\mathbf{y}' = f(t,\mathbf{y}) = J\mathbf{y}$, where $\mathbf{y}$ is the vector of all dependent variables in the discretized system. The Euler's method is stable if all eigenvalues of ΔtJ lie in the complex plane within the disk of radius 1 centered at -1 [5].

In the case of a system consisting of $k\times k\times k$ cubes, J contains, among others, rows of the form

$$C\cdot[\ldots,0,1,0,\ldots,0,1,0,\ldots,0,1,-6,1,0,\ldots,0,1,0,\ldots,0,1,0,\ldots], \tag{2.3}$$

where the distances of 1's from the diagonal are 1, k and k^2. C is either $\frac{\lambda}{\rho c_p \Delta x^2}$ or $\frac{\eta}{\rho\Delta x^2}$. Other rows that are not as critical regarding eigenvalues are not considered. Applying the Gerschgorin theorem we obtain an estimation that all the eigenvalues of ΔtJ lie within disks of radii $6\Delta tC$ centered at $-6\Delta tC$. Together with the domain of dependence condition the required stability conditions are

$$\Delta t \le \frac{\Delta x}{v_{max}}, \quad \Delta t < \frac{\rho c_p \Delta x^2}{6\lambda}, \quad \Delta t < \frac{\rho \Delta x^2}{6\eta}. \tag{2.4}$$

The maximum allowed time-step for various substances is shown in Figure 2.3. The time-step restrictions for solid substances do not depend on $v_{\max}$. Different time-steps are used for air and for the rest of the substances because the difference in restrictions is large.

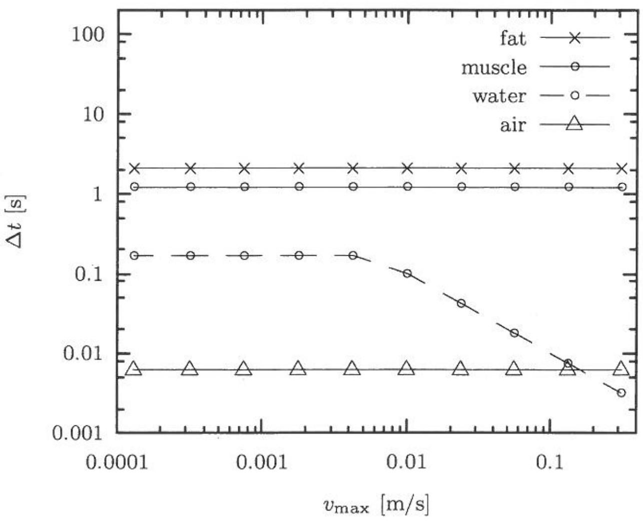

FIG. 2.3. *Time-step stability restrictions (2.4) as a function of fluid velocity for various substances discretized to $\Delta x = 1$ mm.*

3. Pressure Correction Equation. Step 2 of the integration sequence described in Section 2 is the most computationally demanding part of the simulation because a system of k^3 linear equations resulting from (1.4) must be solved in each time-step.

3.1. Linear Equations. The finite difference form of (1.4) for a cube with 6 fluid neighbours is

$$\begin{aligned} 6p_{\Delta|x,y,z} - p_{\Delta|x-1,y,z} - p_{\Delta|x+1,y,z} - \\ -p_{\Delta|x,y-1,z} - p_{\Delta|x,y+1,z} - \\ -p_{\Delta|x,y,z-1} - p_{\Delta|x,y,z+1} = \frac{\rho\Delta x^2}{\Delta t}(\nabla\cdot\mathbf{v}^*)|_{x,y,z}\ . \end{aligned} \tag{3.1}$$

Disregarding solid cubes, the linear system would be 7-diagonal with rows the same as in (2.3).

The discrete Neumann boundary conditions state that $\frac{\partial p_\Delta}{\partial n} = 0$ on the faces of all solid cubes. For example for a solid cube centered at $(x, y, z-1)$ the boundary condition on its upper face $\left(x, y, z-\frac{1}{2}\right)$ is $\frac{\partial p_\Delta}{\partial z} = 0$. Using the central difference approximation we obtain

$$\left.\frac{\partial p_\Delta}{\partial z}\right|_{x,y,z-\frac{1}{2}} = \frac{p_{\Delta|x,y,z} - p_{\Delta|x,y,z-1}}{\Delta x} = 0 \Rightarrow p_{\Delta|x,y,z} = p_{\Delta|x,y,z-1}\,. \tag{3.2}$$

Both $p_{\Delta|x,y,z}$ and $p_{\Delta|x,y,z-1}$ are thus removed from (3.1) in this case. In general, by numbering all the liquid cubes in the domain $1 \ldots N$ we get the linear system $A\mathbf{u} = \mathbf{b}$, where $\mathbf{u} = \{p_{\Delta|x,y,z}\}$, $\mathbf{b}$ is the same as in (3.1) and

$$A = [a_{i,j}]_{N\times N}\ ;\ a_{i,j} = \begin{cases} s & i = j, \text{cube } i \text{ has } s \text{ fluid neighbours,} \\ -1 & i \neq j, \text{cubes } i \text{ and } j \text{ are neighbours,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The matrix A is symmetric with the sum of rows equal to 0 and $\operatorname{rank}(A) = N-1$. Also $\sum b_i = 0$ so the system has infinitely many solutions, any of which is to be found. One of the solutions is chosen by setting an arbitrary unknown to 0 and removing the corresponding equation. The system becomes non-singular and positive definite.

3.2. Iterative Solution of the Sparse System. The matrix A is the same for all time-steps, seemingly making LU factorization an appropriate approach for the solution of the linear system. However, fill-in causes that the computational complexity of forward/backward substitution $O(\beta N)$ ($N = k^3$ and $\beta = k^2$ for a 3-D cubic domain) becomes prohibitive for large systems [5].

Iterative methods start with an initial approximation to the solution $u^{(0)}$, refining it through successive iterations, each of which has the same complexity as matrix-vector multiplication [5]. Because A is sparse with at most 7 non-zero elements per row, the complexity of each iteration step is $O(N)$. We tested three iterative methods: SOR, preconditioned conjugate gradient and multigrid.

3.2.1. SOR Method. The successive over-relaxation (SOR) method is a well-known stationary iterative method based on the Gauss-Seidel method. It requires a relaxation parameter ω, which in our case was obtained experimentally. $\omega = 1$ gives the Gauss-Seidel method while using ω_{optimal} greatly improves the convergence [14].

3.2.2. Conjugate Gradient. Conjugate gradient (CG) is an iterative method based on optimization. A search direction orthogonal to all previous search directions is found in each iteration, after which line search is performed [5]. Convergence can be further improved by using preconditioning, i.e. pre-multiplying the system by a matrix M^{-1} that approximates A^{-1}. We tested incomplete Cholesky preconditioning, which was obtained by performing Cholesky factorization on A but only allowing non-zero values to be written to the same places as in A [10]. This reduced the system condition number approximately by a factor of 10.

3.2.3. Multigrid. The Gauss-Seidel method reduces the high-frequency components of the error vector quickly, i.e. it smoothes the solution. However, many iterations are needed to reduce the low-frequency components. It is therefore beneficial to do parts of the calculation on coarser grids, where these components become high-frequency and are thus easily reduced. The solution is then interpolated to the original fine grid, where a few additional iterations are performed to obtain the final solution [1]. This idea is the basis of multigrid methods, which are generally regarded as best suited to the problems of this sort.

We implemented the full MG that starts with the coarsest grid and zigzags its way up to the original finest grid, as shown in Figure 3.1. The core of the method are the restriction and interpolation operators, which are used for transitions to coarser and finer grids, respectively. If these are chosen correctly, the number of iterations becomes independent of the domain size because error components are smoothed on all ranges of grid scales [15].

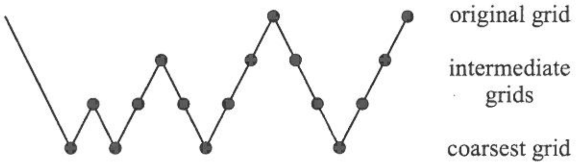

FIG. 3.1. *The full multigrid method consists of restriction (descending lines), interpolation(ascending lines) and applying the smoother (dots).*

3.3. Comparison of Convergence and Complexity. Figure 3.2 shows the comparison of tested iterative methods. The simulated domain was a closed water-filled box of size $(k \times k \times k)$ mm with left wall warmer than the others, which caused water circulation. The domain was discretized to 1 mm. The left graph shows the number of iterations needed to solve the linear system in the first time-step as a function of system size. The right graph shows the total computation time for a simulation of 1 s of real time. The computation times are closer for various methods than iteration numbers because faster converging methods require more FP operations per iteration and because solving the linear system represents only a part of the simulation. The multigrid method significantly outperforms others only for $k > 30$.

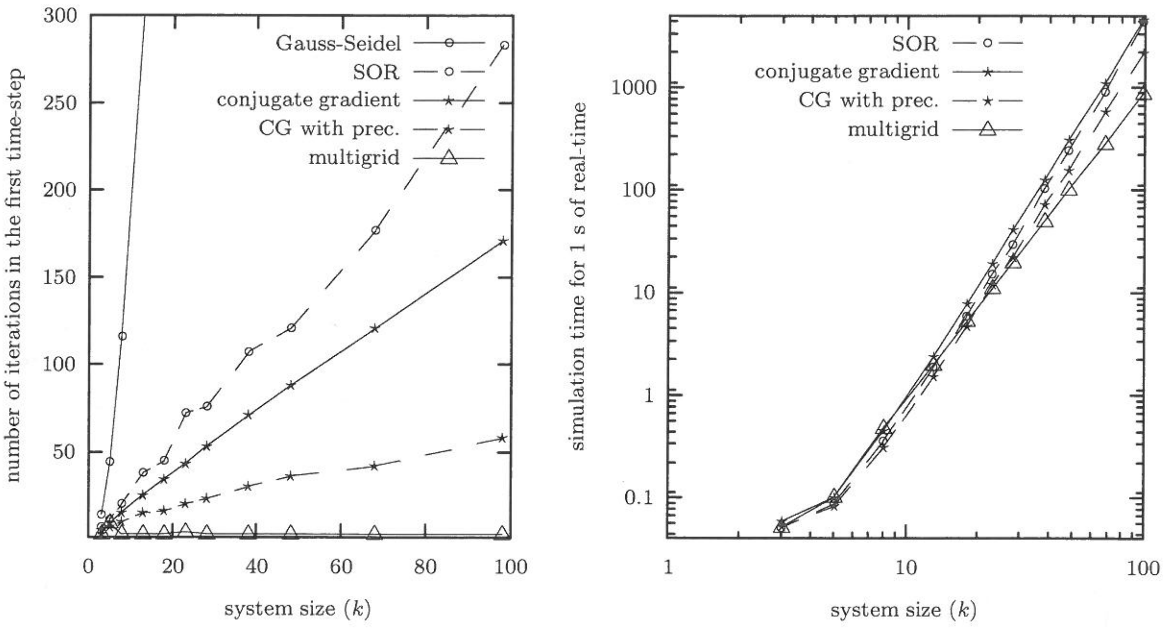

FIG. 3.2. *Comparison of iterative methods.*

4. Simulation Examples. Figure 4.1 shows the simulation results for a closed box with warm left wall discretized to 2 mm. Figure 4.2 shows the temperatures after 10 minutes of simulated topical heart cooling. The model of the heart was derived from the Visible Human Dataset [12]. The heart is partially submerged into cold liquid, which gradually cools it. If the fluid flow is neglected, the core of the liquid stays very cold due to lack of circulation and thus cools the heart more effectively than in the reality.

5. Parallel Implementation. The simulation was parallelized using 1-dimensional domain decomposition, which is appropriate for small numbers of processors [3]. Steps 1, 3, 4, and 5 from Section 2 are essentially local matrix-vector multiplications using communication between processors for exchanging borders of each sub-domain. The parallel execution time is proportional to the number of points N in the domain and is scalable well with the number of processors p, so the

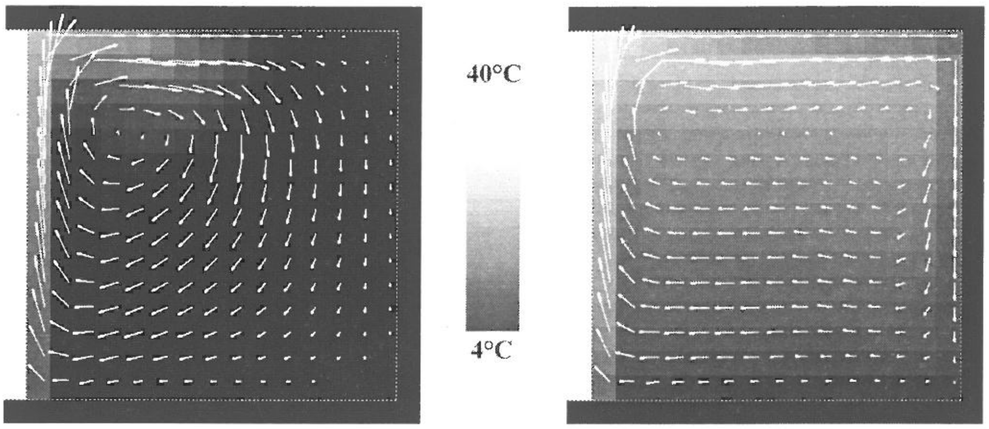

FIG. 4.1. *Central cross-section of a closed box with warmer left wall. The left and right figures show the temperatures (shown in grayscale) and velocities (shown with arrows) after 10 s (left) and 5 min (right). The maximal velocities are about 5.2 mm/s.*

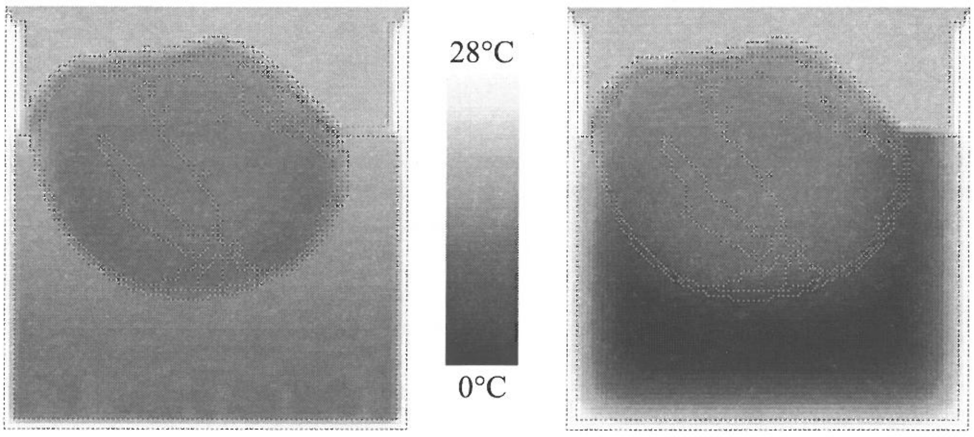

FIG. 4.2. *Temperatures in human heart after 10 minutes of topical cooling with fluid flow simulated (left) and with fluid flow neglected (right).*

parallel complexity is $O(N/p)$. For step 2, the SOR method was parallelized using the red-black ordering of grid points. Because new values at black points depend only on old values at red points and vice versa, the SOR iteration can be run in parallel in the same way as the rest of the calculations [4].

The speedups were tested on a small cluster consisting of 8 AMD Athlon 750 computers with two 100 Mb/s Ethernet ports per computer and connected in a ring. The mpich library [11] was used for communication. Figure 5.1 shows the speedup of the simulation of the closed box of size $k \times k \times k$ with warmer left wall, using SOR method to solve the linear system. Similar speedup is expected on larger systems because there is only a small difference between $k = 38$ and $k = 68$.

Although we have not yet parallelized the multigrid method, we can speculate on a speedup it will achieve. Measurements revealed that the sum of computation time spent in steps 1, 3, 4 and 5 and on the finest grid of step 2 amounts to more than 85% of total computation time. This part will achieve similar speedup as in Figure 5.1 because it can be parallelized in the same way. Further 10% is spent on the grid with halved resolution, which will still achieve a good speedup. The last 5% is spent on coarser grids and can only be efficiently parallelized for very large systems because of the increased communication/computation ratio.

6. Conclusions. The presented paper focuses on the implementation issues of the solution of a complex coupled system of PDEs. The chosen discretization method with numerical scheme and its

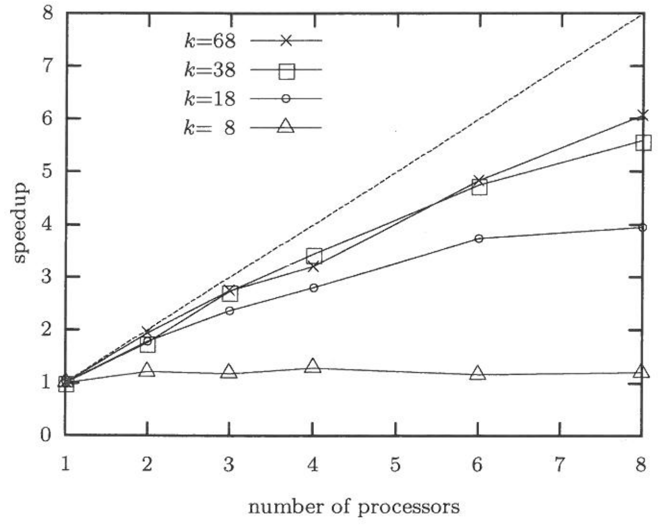

FIG. 5.1. *Speedup of the simulation using SOR method.*

stability has been analyzed in detail. The Poisson equation for pressure correction was transformed into a system of linear equations. The comparison of methods for its solution showed that the multigrid method outperforms others by a factor of 3 and more on the test example. On the other hand, for small systems simpler methods can be faster.

The SOR method and the explicit part of the calculation were parallelized and speedups were measured. It was shown that simple 1-dimensional decomposition works well enough. Some expectations on the speedup of the multigrid method were given.

In our future work we plan to finalize the parallelization of multigrid and replace explicit Euler integration with a second-order accurate implicit scheme, which will allow the use of larger timesteps, but will also require the solution of a linear system for velocity and temperature. It should be analyzed whether overall performance will improve. Our long-term goal is close-to-real-time simulation of inhomogeneous 3-dimensional domains composed of $N > 10^6$ points in order to open the possibility of personalized treatment in medicine.

REFERENCES

[1] A. BRANDT, *Multi-level adaptive solutions to boundary value problems*, Math. Comput., 31 (1977), pp. 333–390.

[2] C. A. J. FLETCHER, *Computational Techniques for Fluid Dynamics*, Springer Verlag, 1988.

[3] G. C. FOX, M. A. JOHNSON, G. A. LYZENGA, S. W. OTTO, J. K. SALMON, AND D. W. WALKER, *Solving Problems on Concurrent Proceessors, Volume 1: General Techniques and Regular Problems*, Prentice-Hall International, 1988.

[4] G. GOLUB AND J. M. ORTEGA, *Scientific Computing - An Introduction with Parallel Computing*, Academic Press Inc., Boston, 1993.

[5] M. T. HEATH, *Scientific Computing: An Introductory Survey, 2nd Ed.*, WCB/McGraw-Hill, 2002.

[6] C. W. HIRT AND J. L. COOK, *Calculating three-dimensional flows around structures*, J. Comput. Phys., 10 (1972), pp. 324–340.

[7] I. KUŠČER AND A. KODRE, *Mathematik in Physik und Technik*, Springer-Verlag, Berlin, 1993.

[8] M. N. ÖZISIK, *Finite Difference Methods in Heat Transfer*, CRC Press, Boca Raton, 1994.

[9] M. PRAPROTNIK, M. ŠTERK, AND R. TROBEC, *A new explicit numerical scheme for nonlinear diffusion problems*, in Parallel Numerics '02, Theory and Applications, Jožef Stefan Institute and University of Salzburg, 2002.

[10] C. SHEN AND J. ZHANG, *Parallel two level block ilu preconditioning techniques for solving large sparse linear systems*, Parallel Computing, 28 (2002), pp. 1451–1475.

[11] M. SNIR, S. OTTO, S. HUSS-LEDERMAN, D. WALKER, AND J. DONGARRA, *MPI: The Complete Reference*, The MIT Press, 1996.

[12] R. TROBEC, G. PIPAN, P. TRUNK, AND J. MOČNIK, *Spatial heart model derived from VHD*, in Bioimages for Europe '99, 2nd International Workshop of the Visible Human Dataset, Milan, 1999.

[13] R. TROBEC, B. SLIVNIK, B. GERŠAK, AND T. GABRIJELČIČ, *Computer simulation and spatial modelling in heart surgery*, Computers in Biology and Medicine, 4 (1998), pp. 393–403.

[14] M. VAJTERŠIC, *Algorithms for Elliptic Problems. Efficient sequential and Parallel Solvers*, Kluwer Academic Publishers, 1993.

[15] P. WESSELING, *An Introduction to Multigrid Methods*, John Wiley and Sons, 1991.

Edited by: Marian Vajteršic
Received: September 27th, 2002
Accepted: July 1st, 2003