Acta Cybernetica **15** (2001) 173–184.

Parallel Simulation of Spiral Waves in Reacting and Diffusing Media

E. M. Ortigosa*, L. F. Romero* and J. I. Ramos*

Abstract

The propagation of the spiral waves in excitable media is governed by the non-linear reaction-diffusion equations. In order to solve these equations in the three-dimensional space, two methods have been implemented and parallelized on both shared- and distributed- memory computers. These implicit methods linearize the equations in time, following alternate directions in the first case (ADI), and using the Crank-Nicolson discretization in the second case. A linear system of algebraic equations has been obtained and it has been solved using direct methods in the ADI technique, while in the second case has been used the conjugated gradient (CG) method. An optimized version of the CG algorithm is presented here, in which the largest efficiency has been obtained.

1 Introduction

Reaction-diffusion equations are ubiquitous in biology, combustion, ecology, etc., because of their relevance in pattern formation, ignition and extinction phenomena, etc. [1-3]. Many studies of these equations are related with equations for activators and inhibitors in one or two spatial dimensions, e.g. the Belousov-Zhabotinskii, Brusselator and Oregonator models, with and without extinction [3]. Of special interest to the study presented in this paper are the analytical and numerical analyses of the propagation of spiral waves in two-dimensional domains, where it has been observed that these waves have a periodic pattern in the absence of heterogeneities. They may exhibit breathing motions in the presence of obstacles or may simply be extinguished by means of the activation of a control parameter in a sufficiently large region of the computational domain. In non-homogeneous media, spiral waves are characterized by steep gradients in space and relaxation-type oscillations whose accurate simulation demands small spatial and temporal steps.

In three dimensions, there have been very few analytical and numerical studies of spiral waves, presumably because of both the large difficulties in examining wave propagation in three-dimensional space and the cost of such simulations [4]. As a

*Departamento de Arquitectura de Computadores, Campus de Teatinos, Universidad de Málaga, E-29071, Spain, e-mail: eva,felipe@ac.uma.es

consequence, filament models based on Fréchet formulae and differential geometry have been developed; these models are analogous to those of vortex filaments in theoretical fluid mechanics.

In this work, three-dimensional simulations of the propagation of spiral waves in a cubic volume, without obstacles, and in the presence and absence of extinction sources are presented. These numerical simulations have been carried out by means of both time linearized and non-linearized techniques with and without operator splitting, i.e., with and without approximate factorization of the three-dimensional operator into one-dimensional ones. The numerical methods employed in the discretization of the governing partial differential equations have been implemented in a parallel fashion in both shared- and distributed-memory computers, and their performance is reported in this paper.

2 Governing equations

Consider the following system of reaction-diffusion equations (Belousov-Zhabotinskii model):

$$\frac{\partial \mathbf{U}}{\partial t} = \mathbf{D}\left(\frac{\partial^2 \mathbf{U}}{\partial x^2} + \frac{\partial^2 \mathbf{U}}{\partial y^2} + \frac{\partial^2 \mathbf{U}}{\partial z^2}\right) + \mathbf{S}(\mathbf{U}). \tag{1}$$

where $\mathbf{D}$ is a diagonal diffusivity tensor, $\mathbf{U} = (u, v)^T$, t is time, x, y and z denote spatial coordinates, $\mathbf{S}$ is a non-linear term,

$$\mathbf{S} = \left(\frac{u(u-1) - (fv + \phi)\frac{u-q}{u+q}}{\epsilon}, u - v\right)^T, \tag{2}$$

T denotes transpose, f, q and ϵ are constants, and ϕ is a control parameter which can be a function of the space and/or time. The equation (1) has been solved in a cube of side equal to 15 non-dimensional units. Unless otherwise stated, $f = 1.4$, $q = 0.002$, $\epsilon = 0.01$, $\phi = 0$ and the diagonal terms of the diffusivity tensor are $d_u = 1$ and $d_v = 0.6$.

Discretizing the time variable in equation (1) by means of a Crank-Nicolson method, one can obtain a non-linear elliptic equation, which can be solved at each time step. The second-order spatial derivatives were discretized by means of three-point, second order accurate finite difference methods that provide a large system of algebraic equations at each time step. The Newton-Raphson method was employed to solve the resulting non-linear system of algebraic equations. If a single iteration of Newton-Raphson method is used, this method is known as Time-Linearization method.

In addition to these techniques, an approximate factorization of the three-dimensional operator into a sequence of one-dimensional ones (here referred to as ADI) was also used. This technique reduces the solution of the linear elliptic equation in three dimensions to the solution of one-dimensional, linear, two-point boundary value problems in each spatial dimension, but introduces second-order

(in time) approximate factorization errors which can be eliminated in an iterative way. On the other hand, the approximate factorization errors are of the same order of magnitude as those introduced by the time discretization but may be large where the norm of the Jacobian matrix of the source terms is large, i.e., at the edges of the spiral wave.

In the next section some details about these methods are presented.

2.1 ADI method

The discrete operators corresponding to equation (1) can be written as

$$\begin{aligned}
\left(\mathbf{I} - \frac{k}{2\Delta x^2}\mathbf{D}\delta_x^2 - \frac{1}{2}k\mu_1\mathbf{J}^n\right)\Delta\mathbf{U}^{**} = \\
\mathbf{D}\left(\frac{k}{\Delta x^2}\delta_x^2 + \frac{k}{\Delta y^2}\delta_y^2 + \frac{k}{\Delta z^2}\delta_z^2\right)\mathbf{U}^n + k\mathbf{S}^n, \\
\left(\mathbf{I} - \frac{k}{2\Delta y^2}\mathbf{D}\delta_y^2 - \frac{1}{2}k\mu_2\mathbf{J}\right)\Delta\mathbf{U}^{*} = \Delta\mathbf{U}^{**}, \\
\left(\mathbf{I} - \frac{k}{2\Delta z^2}\mathbf{D}\delta_z^2 - \frac{1}{2}k\mu_3\mathbf{J}\right)\Delta\mathbf{U} = \Delta\mathbf{U}^{*},
\end{aligned} \tag{3}$$

where the approximate factorization errors have been neglected, k is the time step, Δx is the spatial step size in the x direction, $0 \leq \mu_i \leq 1$, $i = 1, 2, 3$, $\mu_1 + \mu_2 + \mu_3 = 1$, the superscript n denotes the n-th time level, $\mathbf{J}$ denotes the Jacobian matrix of the mapping $\mathbf{U} \rightarrow \mathbf{S}(\mathbf{U})$, $\delta_x^2 v_i = v_{i+1} - 2v_i + v_{i-1}$, and $\Delta\mathbf{U} = \mathbf{U}^{n+1} - \mathbf{U}^n$.

2.2 Crank-Nicolson methods

If the system of equations (1) is solved by means of a Crank-Nicolson method and a time linearization, one linear system of algebraic equations is obtained.

$$\begin{aligned}
\left(\mathbf{I} - \mathbf{D}\left(\frac{k}{2\Delta x^2}\delta_x^2 + \frac{k}{2\Delta y^2}\delta_y^2 + \frac{k}{2\Delta z^2}\delta_z^2\right) - \frac{1}{2}k\mathbf{J}^n\right)\Delta\mathbf{U} = \\
\mathbf{D}\left(\frac{k}{\Delta x^2}\delta_x^2 + \frac{k}{\Delta y^2}\delta_y^2 + \frac{k}{\Delta z^2}\delta_z^2\right)\mathbf{U}^n + k\mathbf{S}^n.
\end{aligned} \tag{4}$$

A comparison between the numerical results obtained with equations (3) and (4), will indicate the magnitude of the approximate factorization errors of the ADI method. The (iterative) conjugate gradient method have been used for the resolution of the system (4), because of the magnitude and dispersion of the system.

The choice of a good preconditioner for the coefficients matrix is the most important factor that influence on the speed of convergence of the CG. We have tested the Jacobi (J), block Jacobi (BJ), incomplete Cholesky (IC) and incomplete-block

Cholesky (IBC) preconditioners in our studies. In the first and third preconditioners, two dependent variables per node have been used to form 2x2 blocks; calculations have also been performed without any preconditioner (NP), but the number of iterations of the CG method becomes extremely large and can be more expensive than a direct method. In Table 1, the average number of iterations required by each method to converge is presented as a function of the grid size; the values shown in this table correspond to $k = 0.0004$ and 20 time steps (n.a.: not available).

Table 1: Average number of iterations of CG for convergence.

Grid	NP	J	BJ	IC	IBC
51^3	5.95	2.25	2.25	2.3	2.20
101^3	6.00	3.1	3.20	3.2	3.1
201^3	n.a.	5.05	5.15	5.2	n.a.

The above table shows that the incomplete-block Cholesky factorization is the most efficient preconditioner; however, the cost associated with this incomplete factorization is larger than the associated with the decrease in the number of iterations[1]. For these reasons, a Jacobi preconditioner has been used in all the simulations presented below.

Another main issue when solving linear systems of algebraic equations is the ordering of the equations. There are two criteria for ordering: the original differential order equation and the grid point where the discretization takes place. Depending on the approach selected, two different orderings are obtained, named blocking of the equations and blocking of the variables respectively. Blocking of the equations results in a small number (2 in our problem) of very large blocks, while blocking of the variables per node results in more small blocks (2×2). Here, we have tested both types of blocking and found that blocking of the variables results in faster simulations due mainly to a 3% and 5% reduction in primary and secondary cache misses respectively.

3 Parallel implementation

3.1 Parallelization of ADI

This technique has been implemented on an Origin 2000, using a shared memory model (*openMP* libraries). The dynamic block cartesian decomposition (DBCD) has been used for parallel implementation. In this technique, each processor has a contiguous grid block, but the decomposition changes as the one-dimensional

[1] In order to decrease the factorization cost, calculations have been performed using a frozen preconditioner for several time steps. Results showed that the number of iterations of the CG method increase substantially due to the large relational speeds of the spiral wavers considered in this paper.

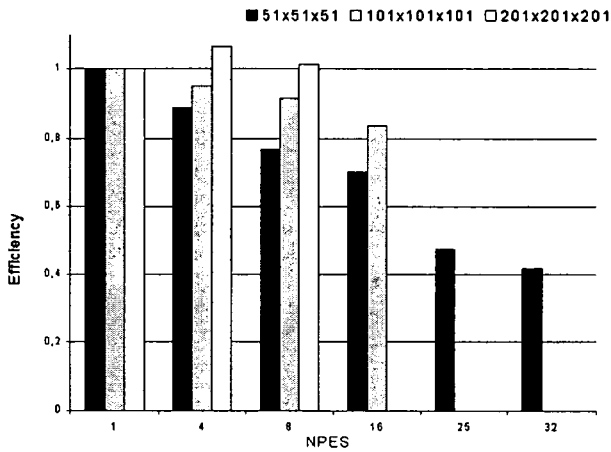

Figure 1: Efficiency of the ADI method as a function of the number of processors (NPES) and the grid size.

operator is changed, i.e., it changes according to the spatial direction. As stated in [5], the optimal solution is reached when the data is partitioned in the z-direction while solving the algebraic equations in the x- and y-directions, whereas a partition in the x- or y-directions may be employed when solving the algebraic equations in the z-direction. The main drawback of this technique is that a lot of accesses are required to remote data before the solution in the z direction is obtained. This produces a high number of cache misses.

A possible alternative is the Bruno-Capello algorithm [5] that partitions the domain in blocks of $\sqrt{n} \times \sqrt{n} \times \sqrt{n}$ in such a manner that there is no more than one block per processor in each plane coordinate; however, this algorithm imposes some limitations to the parallel system because of its requirements for a number of processors equal to the square of a natural number.

Results in Figure 1 indicate that the parallel efficiency of DBCD is quite close to 1, even with a high number of processors. This ideal behavior is owed in great measure to the regularity of the data, and the consequent efficiency of the compiler in the inclusion of directives for the premature search of cache blocks (*prefetching*). Moreover, as will be illustrated bellow, the main drawback of the ADI method is its sequential execution time compared with that of the Crank-Nicolson method.

3.2 Parallelization of Crank-Nicolson

First, the implementation of Crank-Nicolson method combined with the CG (CN-CG) method has been performed using a shared-memory model. In each time step it is necessary to generate the coefficients matrix and the independent term vector

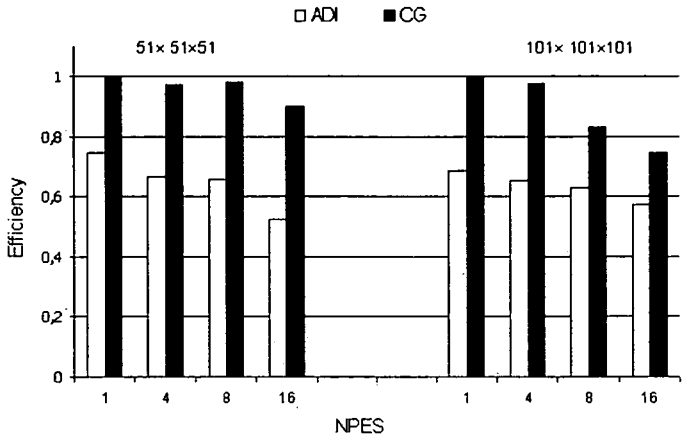

Figure 2: Efficiency of the ADI and CG methods as a function of the number of processors and the grid size.

(rhs), and to solve the system. In all these stages we have used a data-parallel model based on a grid partition along the z axis that coincides with a block partition of both the matrix and the vector rhs.

The parallelization of the CG method is very simple since every vector operation of this algorithm can be parallelized separately, i.e., each processor executes scalar operations on a subset of the components of a vector, and a block partition of the vectors is sufficient to obtain a good performance.

The CG method contains two inner products, three $\alpha\mathbf{x} + \mathbf{y}$ operations and one matrix vector product whose computations require at least three synchronization points, and a communication step in order to obtain the final result. It is not possible to merge messages and those operations imposes severe limitations to the parallel efficiency of the CG algorithm. However, an optimized rearrangement of CG for parallel computations can be found in [6]. Unfortunately, this optimized method is only useful when an incomplete factorization is employed as a preconditioner. As it have been shown above, this kind of preconditioner does not result in any acceleration in the convergence of the problem considered here.

On the other hand, another methods based on the CG not requiring the use of incomplete preconditioner have been described. In particular we have considered a model (Aykanat, Özgüner, F. and Scott, D.S. [7], also Chronopoulos and Gear [8]) that reduces the number of synchronization points and reduction messages to one per iteration, performing $2n$ additional floating operations. In this method we have introduced automatic *prefetching*, in order to reduce the cost produced by the accesses to remote memory. Experiments with this modified CG method have shown that there is only a small increase in the sequential computational time but its parallel performance is excellent.

Figure 2 shows the efficiency of ADI and CG as a function of both the number of processors and the grid size using a shared-memory model, and indicates that the efficiency of ADI is smaller than that of CG, although the differences decrease, as the grid is refined. However, the fact that the computational time for the sequential version is smaller for the CG method, has been critical in the decision to continue our work with the Crank-Nicolson-CG model.

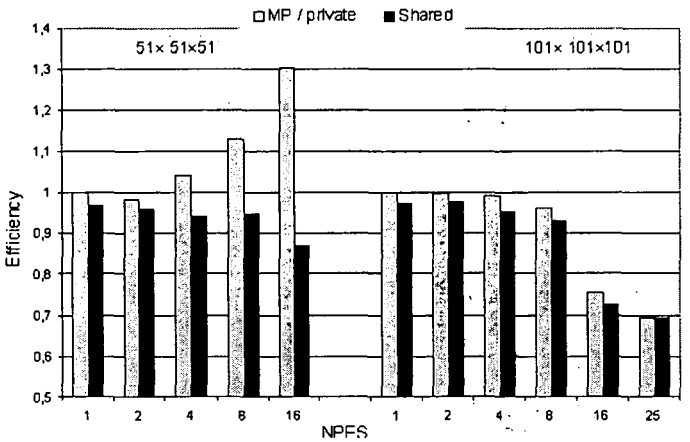

Figure 3: Efficiency of the CG methods as a function of the number of processors with shared-memory and message-passing models with private memory.

4 Optimization of Crank-Nicolson method

One of the main aims of this work is to perform a comparison between accessing in advance to cache blocks using *prefetching* and using asynchronous messages in the message-passing model (overlapped with computations) on shared- and distributed-memory computers. Therefore, we have implemented the conjugated gradient method on an Origin2000 computer, using both private memory and the *shmem* libraries for communications.

In Figure 3 we compare the efficiency for the two parallel implementations of the CG, being better for the message passing model. In the best case, an efficiency of 130% was obtained using 16 processors in a 51 point grid.

This result can be justified by the fact that the communication of large blocks of data is more efficient than having many cache misses in a shared memory model.

This difference becomes noticeable in thick grids where the access to remote memories is more frequent, since the relationship communications ($O(n^2)$)/computations ($O(n^3)$) is higher. Figure 3 also shows that the efficiency of both implementations of the CG, for the 101 point grid, in a shared memory

computer (Origin2000) is degraded as the grid size increases, because in these cases the capacity of the cache memory is insufficient for all the data.

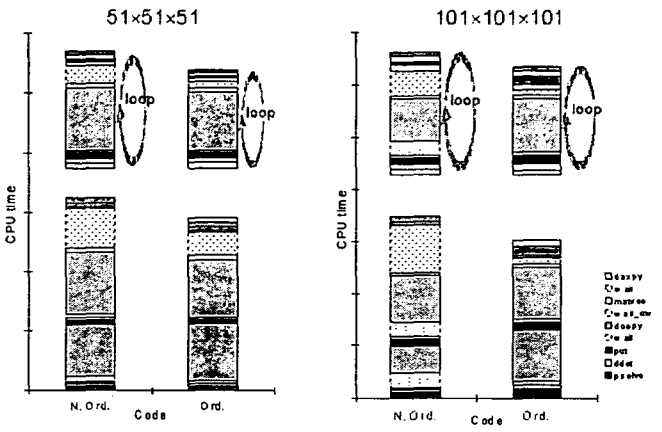

Figure 4: CPU times per processor for the non ordering and ordering codes.

An important performance loss can be observed for the $101 \times 101 \times 101$ grid when the number of processors is very high. Although this loss can be justified by the high cost of the remote accesses in proportion to the computations that each processors performs, we consider very important for our problem to obtain good performance under such conditions (very fine grids and higher number of processors). A detailed analysis (of the origin of this performance loss) shows that, for example, with a 101 point grid and 16 processors, almost the 30% of the execution time in message-passing the processors are waiting for the data. Therefore, the sending of asynchronous messages does not produce the expected results, because, in the most of cases, the sent data are required immediately in the following operation, and there are no intermediate computations that hide the latency of these messages.

The conjugate gradient algorithm of Gear et al., has been modified based on a reordering heuristic of the computations and communications [9]. We have moved several calculations to those places where the latencies of the messages can be hidden and the delays have been almost eliminated. This algorithm is based mainly on delaying the calculation of the solution, $\Delta \mathbf{U}$ (equation 4), from each iteration to the following one. When the convergence is reached, an update of $\Delta \mathbf{U}$ (to obtain the solution of the system) is required. This update is performed directly on the solution $\mathbf{U}$. After the conjugate gradient, a message with the bound values of $\mathbf{U}$ is sent to the neighbor processors and is overlapped with the update of $\Delta \mathbf{U}$. Figure 4 shows the execution time for each processor for each conjugated gradient routines (non ordered and ordered codes), as well as the latency time (dotted area). The bottom part of the graphic corresponds to the initialization part of the iterative

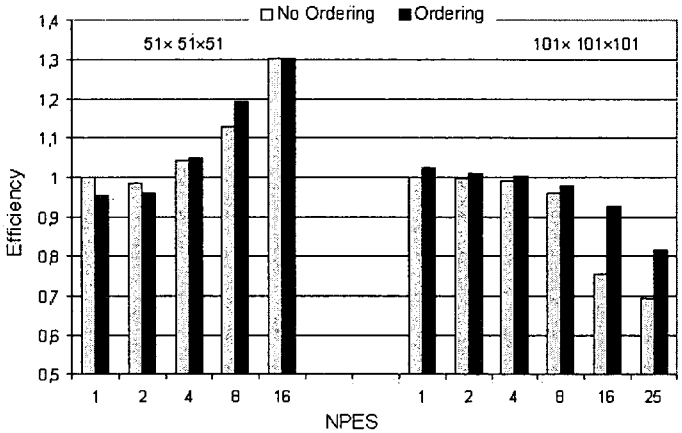

Figure 5: Efficiency comparison between the non-ordering and ordering codes.

algorithm, and the upper part corresponds to the inner loop (therefore it has to be repeated according to the number of iterations). In Figure 5 it can be seen how the reordering allows to reduce significantly the latency time, decreasing about a 15% the CPU time and increasing in an equivalent way the efficiency. This improvement is higher for a 101 point grid, where the communications cost is very important when the number of processors increases.

5 Physical Results

Calculations have been performed 1) without extinction, i.e., $\phi = 0$, 2) with an extinction barrier, i.e., $\phi = 0.2$ for $-0.3 \leq y \leq 0.3$ and $\phi = 0$ otherwise, and 3) with a localized extinction zone which is a cube with center located at $(0, 0, 0)$ and length equal to 1.65 where $\phi = 0.2$. In all cases, homogeneous Neumann boundary conditions were employed at the faces of the computational domain, $k = 0.0004$, and the calculations were performed until $t = 60$, starting from initial conditions corresponding to a wedge for u. In all the calculations considered in this paper, it has been observed that the solution became periodic at about $t = 25, 30$, and 28 for the first, second and third cases, respectively. In the first case, i.e., without extinction, it has been shown that the spiral wave rotates around an axis parallel to the z-direction which passes through the center of the computational domain and remains anchored there, the u solution is symmetric about the $z = 0$ plane, and the rotational speed of the spiral wave decreases as it approaches the planes $z = -7.5$ and $z = 7.5$. Some sample results illustrating the u solution at $t = 41.6$ are presented in Figure 6 which indicates that, at this time, the spiral wave is nearly absent near the bottom and top planes, whereas the influence of the initial conditions can still be observed near these planes. The u-solution also shows that at

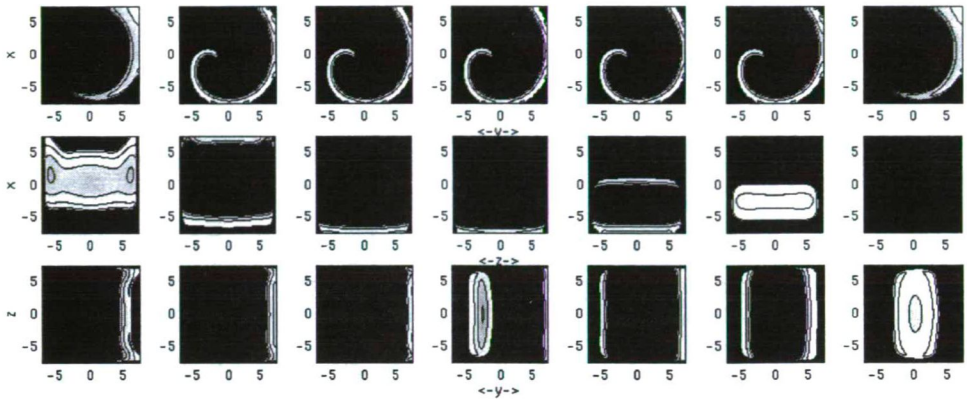

Figure 6: u-solution for a 101×101×101 grid at $t = 41.6$. From left to right: a) top: k=100, 84, 68, 51, 34, 18, 2, b) middle: j=100,84, 68, 51, 34, 18, 2, c) bottom: i=100, 84, 68, 51, 34, 18, 2.

$t = 41.6$, near to the top and bottom planar boundaries, the spiral wave is thicker and much shorter than that near $z = 0$.

In the second case, the extinction barrier is a slab parallel to the $x - z$ plane where the value of ϕ is different from zero; therefore, the source term for u decreases exponentially whenever $u > q$ by an amount that is proportional to ϕ, i.e., the spiral wave may be extinguished as indicated in Figure 7. The u- solution in $x = constant$ planes indicates that almost planar fronts propagate initially from the left to the right boundaries but, on encountering the extinction barrier, they decelerate and emerge from this barrier until they reach the right boundary.

In the third case, the results are similar to those of the first one except for the fact that the tip of the spiral wave rotates around the extinction source, and the wave is shorter and has a thicker tip. In the first case, the tip is anchored on the vertical axis passing through the center of the cube, while, in the third one, the anchoring point describes a trajectory which corresponds to the projection of the extinction cube into the $x - y$ plane.

6 Conclusions

Two numerical methods (ADI and Crank-Nicolson) for the numerical solution of three-dimensional reaction-diffusion equations corresponding to spiral wave propagation in excitable media have been parallelized in shared- and distributed-memory computer. The parallelization of the approximate factorization technique has been carried out with a dynamic block cartesian decomposition (DBCD) and its efficiency is quite close to one, even with a high number of processors. The parallelization of the Crank-Nicolson method has been performed by means of an optimization of the conjugate gradient method where the latency times have been almost elimi-

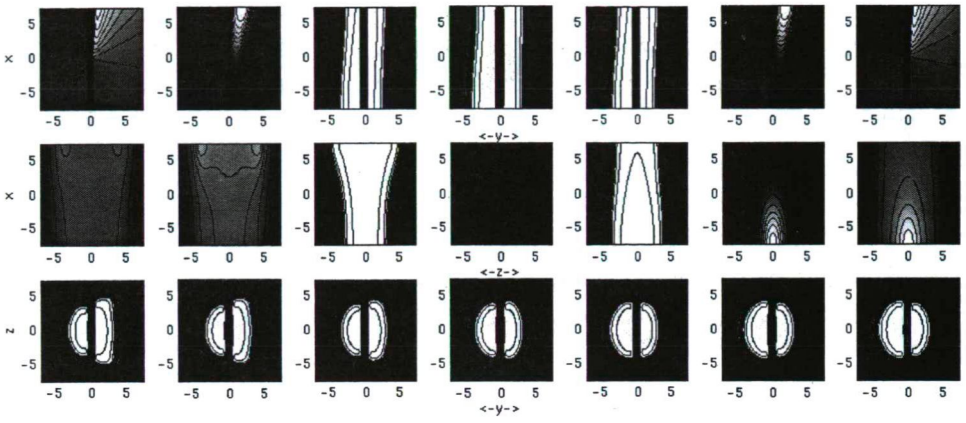

Figure 7: u-solution for a 101×101×101 grid at $t = 41.6$. From left to right: a) top: k=100, 84, 68, 51, 34, 18, 2, b) middle: j=100, 84, 68, 51, 34, 18, 2, c) bottom: i=100, 84, 68, 51, 34, 18, 2.

nated in a message-passing programming model. As a result, we obtain efficiencies close to the ideal, even with a higher number of processors. The scalability of our model should allow maintenance of the efficiency with proportional increments of the problem size and the number of processors.

This work has its natural continuation in the implementation of the mentioned optimization in a shared-memory environment, where the cost of remote accesses will decrease using the manual inclusion of *prefetching* directives. The objectives of a future work will be a comparison of both models and a extrapolation to the ADI method of these ideas.

References

[1] Murray, J.D. *Mathematical Biology*, Springer-Verlag: New York, 1989.

[2] Williams, F.A. *Combustion Theory*, second edition, Addison-Wesley: New York, 1985.

[3] Holden, A.V., Markus, M. and Othmer, H.G. (eds). *Nonlinear Wave Propagation in Excitable Media*, Plenum Press: New York, 1991.

[4] Keener, J. and Sneyd, J. *Mathematical Physiology*, Springer-Verlag: New York, 1998.

[5] Van der Wijngaart, R.F. Efficient Implementation of a 3-Dimensional ADI Method on the iPSC/860, *Supercomputing'93*, pp. 102–111, 1993.

[6] Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and Van de Vorst, H., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM: Philadelphia, 1994.

[7] Aykanat, C., Özgüner, F. and Scott, D.S. Vectorization and parallelization of the conjugate gradient algorithm on hypercube-connected vector processors, *Microprocess. Microprogram.*, 29, pp. 67–82, 1990.

[8] Chronopoulos, A. T. and Gear C. W. S-step iterative methods for symmetric linear systems, *J. comput. Appl. Math.*, 25, pp. 153–168, 1989.

[9] Romero, L.F., Ortigosa, E.M. and Ramos, J.I. Parallel Strategies for the VMEC Program, *J. Parallel Computing*, to appear.