# Highly parallel alternating directions algorithm for time dependent problems

Maria Ganzha[*], Krassimir Georgiev[†], Ivan Lirkov[†], Svetozar Margenov[†] and Marcin Paprzycki[*]

[*]*Systems Research Institute, Polish Academy of Science, ul. Newelska 6, 01-447 Warsaw, Poland*
[†]*Institute of Information and Communication Technologies, Bulgarian Academy of Sciences Acad. G. Bonchev, bl. 25A, 1113 Sofia, Bulgaria*

**Abstract.** In our work, we consider the time dependent Stokes equation on a finite time interval and on a uniform rectangular mesh, written in terms of velocity and pressure. For this problem, a parallel algorithm based on a novel direction splitting approach is developed. Here, the pressure equation is derived from a perturbed form of the continuity equation, in which the incompressibility constraint is penalized in a negative norm induced by the direction splitting. The scheme used in the algorithm is composed of two parts: (i) velocity prediction, and (ii) pressure correction. This is a Crank-Nicolson-type two-stage time integration scheme for two and three dimensional parabolic problems in which the second-order derivative, with respect to each space variable, is treated implicitly while the other variable is made explicit at each time sub-step. In order to achieve a good parallel performance the solution of the Poison problem for the pressure correction is replaced by solving a sequence of one-dimensional second order elliptic boundary value problems in each spatial direction. The parallel code is implemented using the standard MPI functions and tested on two modern parallel computer systems. The performed numerical tests demonstrate good level of parallel efficiency and scalability of the studied direction-splitting-based algorithm.

**Keywords:** Navier-Stokes, time splitting, ADI, incompressible flows, pressure Poisson equation, parallel algorithm
**PACS:** 02.60.Cb, 02.60.Lj, 02.70.Bf, 07.05.Tp, 47.10.ad, 47.11.Bc

## INTRODUCTION

The objective of this article is to analyze the parallel performance of a new fractional time stepping technique, based on a direction splitting strategy, developed to solve the incompressible Navier-Stokes equations.

Projection schemes were first introduced in [1, 2] and they have been used in Computational Fluid Dynamics (CFD) for the last forty years. During these years, these techniques have been evolving, but the main paradigm, consisting of decomposing vector fields into a divergence-free part and a gradient, has been preserved (see [3] for a review of projection methods). In terms of computational efficiency, projection algorithms are far superior to the methods that solve the coupled velocity-pressure system. This feature makes them the most popular techniques in the CFD community for solving the unsteady Navier-Stokes equations. The computational complexity of each time step of the projection methods is that of solving one vector-valued advection-diffusion equation, plus one scalar-valued Poisson equation with the Neumann boundary conditions. Note that, for large scale problems, and large Reynolds numbers, the cost of solving the Poisson equation becomes dominant.

The alternating directions algorithm proposed in [4] reduces the computational complexity of the action of the incompressibility constraint. The key idea is to modify the standard projection approach, in which the vector fields are decomposed into a divergence-free part plus a gradient part. This variation of the projection methods has been proved to be very efficient for solving variable density flows [see, for instance 5, 6]. In the new method the pressure equation is derived from a perturbed form of the continuity equation, in which the incompressibility constraint is penalized in a negative norm induced by the direction splitting. The standard Poisson problem for the pressure correction is replaced by series of one-dimensional second-order boundary value problems. This technique is proved to be stable and convergent [for details see 4]. Furthermore, a very brief initial assessment, found in [4], indicates that the new approach should be efficiently parallelizable. The aim of this paper is to experimentally investigate this claim on two distinct parallel computers, for two and three dimensional problems.

## STOKES EQUATION

Let us start by defining the problem to be solved. We consider the time-dependent Navier-Stokes equations on a finite time interval $[0,T]$, and in a rectangular domain $\Omega$. Since the nonlinear term in the Navier-Stokes equations does not interfere with the incompressibility constraint, we focus our attention on the time-dependent Stokes equations, written in terms of velocity $\mathbf{u}$ and pressure $p$:

$$\begin{cases} \mathbf{u}_t - \nu\Delta\mathbf{u} + \nabla p = \mathbf{f} & \text{in} \quad \Omega\times(0,T) \\ \nabla\cdot\mathbf{u} = 0 & \text{in} \quad \Omega\times(0,T) \\ \mathbf{u}|_{\partial\Omega} = 0, \quad \partial_n p|_{\partial\Omega} = 0 & \text{in} \quad (0,T) \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, \quad p|_{t=0} = p_0 & \text{in} \quad \Omega \end{cases}, \tag{1}$$

where the $\mathbf{f}$ is a smooth source term, the $\nu$ is the kinematic viscosity, and the $\mathbf{u}_0$ is a solenoidal initial velocity field with a zero normal trace. In our work, we consider homogeneous Dirichlet boundary conditions on the velocity.

To solve thus described problem, we discretize the time interval $[0,T]$ using a uniform mesh. Finally, let $\tau$ be the time step used in the algorithm.

## Singular perturbation analysis

The Chorin-Temam algorithm is a singular perturbation of the equation (1):

$$\begin{cases} \partial_t\mathbf{u}_\varepsilon - \nu\Delta\mathbf{u}_\varepsilon + \nabla p_\varepsilon = \mathbf{f} & \text{in} \quad \Omega\times(0,T), \\ -\tau\Delta p_\varepsilon + \nabla\cdot\mathbf{u}_\varepsilon = 0 & \text{in} \quad \Omega\times(0,T), \\ \mathbf{u}_\varepsilon|_{\partial\Omega} = 0, \quad \partial_n p_\varepsilon|_{\partial\Omega} = 0 & \text{in} \quad (0,T), \\ \mathbf{u}_\varepsilon|_{t=0} = \mathbf{u}_0, \quad p_\varepsilon|_{t=0} = p_0 & \text{in} \quad \Omega, \end{cases} \tag{2}$$

where the perturbation parameter $\varepsilon := \tau$.

We depart from the equation (2) by considering the following alternative $\mathscr{O}(\tau)$ perturbation of the equation (1):

$$\begin{cases} \partial_t\mathbf{u}_\varepsilon - \nu\Delta\mathbf{u}_\varepsilon + \nabla p_\varepsilon = \mathbf{f} & \text{in} \quad \Omega\times(0,T), \\ \tau A p_\varepsilon + \nabla\cdot\mathbf{u}_\varepsilon = 0 & \text{in} \quad \Omega\times(0,T), \\ \mathbf{u}_\varepsilon|_{\partial\Omega} = 0, \quad p_\varepsilon \in D(A) & \text{in} \quad (0,T), \\ \mathbf{u}_\varepsilon|_{t=0} = \mathbf{u}_0, \quad p_\varepsilon|_{t=0} = p_0 & \text{in} \quad \Omega, \end{cases} \tag{3}$$

where the operator $A$ and its domain $D(A)$ are such that the bilinear form $a(p,q) := \int_\Omega qAp\mathrm{d}x$ satisfies the following properties: $a$ is symmetric, and $\|\nabla q\|_{L^2}^2 \leq a(q,q), \forall q \in D(A)$.

## Fractional step technique

We now construct a fractional step technique approximating equation (3) by using the alternating direction strategies.

- *Pressure predictor*. The algorithm is initialized by setting $p^{-\frac{1}{2}} = 0$, and for $n \geq 0$ we set $p^{*,n+\frac{1}{2}} = p^{n-\frac{1}{2}}$.
- *Velocity update*. We update the velocity by using a direction splitting technique proposed by Douglas. The algorithm is initialized by setting $\mathbf{u}^0 = \mathbf{u}_0$, and for $n \geq 0$ the velocity is updated as follows:

$$\begin{aligned} \frac{\boldsymbol{\xi}^{n+1} - \mathbf{u}^n}{\tau} - \nu\Delta\mathbf{u}^n + \nabla p^{*,n+\frac{1}{2}} &= \mathbf{f}|_{t=(n+\frac{1}{2})\tau}, & \boldsymbol{\xi}^{n+1}|_{\partial\Omega} &= 0, \\ \frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\xi}^{n+1}}{\tau} - \frac{\nu}{2}\frac{\partial^2(\boldsymbol{\eta}^{n+1} - \mathbf{u}^n)}{\partial x^2} &= 0, & \boldsymbol{\eta}^{n+1}|_{\partial\Omega} &= 0, \\ \frac{\boldsymbol{\zeta}^{n+1} - \boldsymbol{\eta}^{n+1}}{\tau} - \frac{\nu}{2}\frac{\partial^2(\boldsymbol{\zeta}^{n+1} - \mathbf{u}^n)}{\partial y^2} &= 0, & \boldsymbol{\zeta}^{n+1}|_{\partial\Omega} &= 0, \\ \frac{\mathbf{u}^{n+1} - \boldsymbol{\zeta}^{n+1}}{\tau} - \frac{\nu}{2}\frac{\partial^2(\mathbf{u}^{n+1} - \mathbf{u}^n)}{\partial z^2} &= 0, & \mathbf{u}^{n+1}|_{\partial\Omega} &= 0, \end{aligned} \tag{4}$$

The two-dimensional version of the algorithm is obtained by omitting the last sub-step and setting $\mathbf{u}^{n+1} = \boldsymbol{\zeta}^{n+1}$.

- *Pressure update*. The pressure is updated by solving $Ap^{n+\frac{1}{2}} = -\frac{1}{\tau}\nabla\cdot\mathbf{u}^{n+1}$ with the direction splitting operator $A := (1-\partial_{xx})(1-\partial_{yy})(1-\partial_{zz})$ supplemented with appropriate boundary conditions. This is achieved as follows:

$$\begin{aligned} \varphi - \partial_{xx}\varphi &= -\frac{1}{\tau}\nabla\cdot\mathbf{u}^{n+1}, & \partial_x\varphi|_{\partial\Omega} &= 0, \\ \psi - \partial_{yy}\psi &= \varphi, & \partial_y\psi|_{\partial\Omega} &= 0, \\ p^{n+\frac{1}{2}} - \partial_{zz}p^{n+\frac{1}{2}} &= \psi, & \partial_z p^{n+\frac{1}{2}}|_{\partial\Omega} &= 0, \end{aligned} \tag{5}$$

## Higher-order variants

It is well known that the time accuracy of the Chorin-Temam scheme is limited. To improve the accuracy of the method we use higher-order versions of the method by using a collection of incremental schemes. We introduce the following alternative $\mathscr{O}(\tau^2)$ perturbation of the equation (2), which allows for the direction splitting:

$$\begin{cases} \partial_t\mathbf{u}_\varepsilon - \nu\Delta\mathbf{u}_\varepsilon + \nabla p_\varepsilon = \mathbf{f} & \text{in} \quad \Omega\times(0,T), \\ \tau A\phi_\varepsilon + \nabla\cdot\mathbf{u}_\varepsilon = 0 & \text{in} \quad \Omega\times(0,T), \\ \tau\partial_t p_\varepsilon = \phi_\varepsilon - \chi\nu\nabla\cdot\mathbf{u}_\varepsilon & \qquad \phi_\varepsilon\in D(A) \\ \mathbf{u}_\varepsilon|_{\partial\Omega} = 0, \quad \partial_n\phi_\varepsilon|_{\partial\Omega} = 0 & \text{in} \quad (0,T), \\ \mathbf{u}_\varepsilon|_{t=0} = \mathbf{u}_0, \quad p_\varepsilon|_{t=0} = p_0 & \text{in} \quad \Omega, \end{cases} \tag{6}$$

where $\chi\in[0,1]$ is an adjustable parameter.

## PARALLEL ALTERNATING DIRECTIONS ALGORITHM

Let us now describe the proposed parallel solution method. Authors of [4] introduced an innovative fractional time stepping technique for solving the incompressible Navier-Stokes equations, based on a direction splitting strategy. They used a singular perturbation of the Stokes equation with the perturbation parameter $\tau$. The standard Poisson problem for the pressure correction was replaced by series of one-dimensional second-order boundary value problems.

### Formulation of the scheme

The scheme used in the algorithm is composed of the following parts: (i) pressure prediction, (ii) velocity update, (iii) penalty step, and (iv) pressure correction. Let us now describe an algorithm that uses the direction splitting operator

$$A := \left(1-\frac{\partial^2}{\partial x^2}\right)\left(1-\frac{\partial^2}{\partial y^2}\right)\left(1-\frac{\partial^2}{\partial z^2}\right).$$

- *Pressure predictor*. The algorithm is initialized by setting $p^{-\frac{1}{2}} = p^{-\frac{3}{2}} = p_0$. Next, for all $n\geq 0$, a pressure predictor is computed as follows
$$p^{*,n+\frac{1}{2}} = 2p^{n-\frac{1}{2}} - p^{n-\frac{3}{2}}. \tag{7}$$
- *Velocity update*. The velocity field is initialized by setting $\mathbf{u}^0 = \mathbf{u}_0$, and for all $n\geq 0$ the velocity update is computed by solving the following series of one-dimensional problems

$$\begin{aligned} \frac{\boldsymbol{\xi}^{n+1}-\mathbf{u}^n}{\tau} - \nu\Delta\mathbf{u}^n + \nabla p^{*,n+\frac{1}{2}} &= \mathbf{f}^{n+\frac{1}{2}} = \mathbf{f}|_{t=\left(n+\frac{1}{2}\right)\tau}, \qquad \boldsymbol{\xi}^{n+1}|_{\partial\Omega} = 0 \\ \frac{\boldsymbol{\eta}^{n+1}-\boldsymbol{\xi}^{n+1}}{\tau} - \frac{\nu}{2}\frac{\partial^2(\boldsymbol{\eta}^{n+1}-\mathbf{u}^n)}{\partial x^2} &= 0, \qquad \boldsymbol{\eta}^{n+1}|_{\partial\Omega} = 0 && (8) \\ \frac{\boldsymbol{\zeta}^{n+1}-\boldsymbol{\eta}^{n+1}}{\tau} - \frac{\nu}{2}\frac{\partial^2(\boldsymbol{\zeta}^{n+1}-\mathbf{u}^n)}{\partial y^2} &= 0, \qquad \boldsymbol{\zeta}^{n+1}|_{\partial\Omega} = 0 && (9) \\ \frac{\mathbf{u}^{n+1}-\boldsymbol{\zeta}^{n+1}}{\tau} - \frac{\nu}{2}\frac{\partial^2(\mathbf{u}^{n+1}-\mathbf{u}^n)}{\partial z^2} &= 0, \qquad \mathbf{u}^{n+1}|_{\partial\Omega} = 0. && (10) \end{aligned}$$

- *Penalty step*. The intermediate parameter $\phi$ is approximated by solving $A\phi = -\frac{1}{\tau}\nabla\cdot\mathbf{u}^{n+1}$. This is done by solving the following series of one-dimensional problems:

$$\begin{array}{rll} \theta - \theta_{xx} = & -\frac{1}{\tau}\nabla\cdot\mathbf{u}^{n+1}, & \theta_x|_{\partial\Omega} = 0, \\ \psi - \psi_{yy} = & \theta, & \psi_y|_{\partial\Omega} = 0, \\ \phi - \phi_{zz} = & \psi, & \phi_z|_{\partial\Omega} = 0, \end{array} \tag{11}$$

- *Pressure update*. The pressure is updated using the parameter $\chi = \in [0, \frac{1}{2}]$.

$$p^{n+\frac{1}{2}} = p^{n-\frac{1}{2}} + \phi - \chi\nu\nabla\cdot\frac{\mathbf{u}^{n+1}+\mathbf{u}^{n}}{2} \tag{12}$$

### Parallel algorithm

In the proposed algorithm, we use a rectangular uniform mesh combined with a central difference scheme for the second derivatives for solving equations (8)–(11). Thus the algorithm requires only the solution of tridiagonal linear systems. The parallelization is based on a decomposition of the domain into rectangular sub-domains. Let us associate with each such sub-domain a set of integer coordinates $(i_x, i_y, i_z)$, and identify it with a given processor. The linear systems, generated by the one-dimensional problems that need to be solved in each direction, are divided into systems for each set of unknowns, corresponding to the internal nodes for each block that can be solved independently by a direct method. The corresponding Schur complement for the interface unknowns between the blocks that have an equal coordinate $i_x$, $i_y$, or $i_z$ is also tridiagonal and can be therefore easily inverted directly. The overall algorithm requires only exchange of the interface data, which allows for a very efficient parallelization with an efficiency comparable to that of an explicit schemes.

## EXPERIMENTAL RESULTS

Recall, that the aim of this paper is to experimentally verify the claim found in [4] that the proposed approach has good potential for parallelization. To this effect we have solved the problem (1) in the domain $\Omega = (0,1)^d$, $d = 2,3$, for $t \in [0,2]$ with Dirichlet boundary conditions. The discretization in time was done with time step $10^{-2}$. The parameter in the pressure update sub-step was $\chi = \frac{1}{2}$, and the kinematic viscosity was $\nu = 10^{-3}$. The discretization in space used mesh sizes $h_x = \frac{1}{n_x-1}$, $h_y = \frac{1}{n_y-1}$, and $h_z = \frac{1}{n_z-1}$. Thus, the equation (8) resulted in linear systems of size $n_x$, the equation (9) resulted in linear systems of size $n_y$, and the equation (10) in linear systems of size $n_z$. The total number of unknowns in the discrete 2D problem was $600\,n_x n_y$, while in the 3D problem it was $800\,n_x n_y n_z$.

To solve the problem, a portable parallel code was designed and implemented in C, while the parallelization has been facilitated using the MPI library [7, 8]. In the code, we use the LAPACK subroutines DPTTRF and DPTTS2 [see 9] for solving tridiagonal systems of equations resulting from equations (8), (9), (10), and (11) for the unknowns corresponding to the internal nodes of each sub-domain. The same subroutines are used to solve the tridiagonal systems with the Schur complement.

The parallel code has been tested on a cluster computer system (Sooner), located in the Oklahoma Supercomputing Center (OSCER), and on the IBM Blue Gene/P machine at the Bulgarian Supercomputing Center. In our experiments, times have been collected using the MPI provided timer and we report the best results from multiple runs. In the following tables, we report the elapsed time $T_c$ in seconds using $c$ cores, the parallel speed-up $S_c = T_1/T_c$, and the parallel efficiency $E_c = S_c/c$.

Tables 1 and 2 show the results collected on the Sooner for the 2D and 3D problems. It is a Dell Xeon E5405 (“Harpertown”) quad core Linux cluster. It has 486 Dell PowerEdge 1950 III nodes, and two quad core processors per node. Each processor runs at 2 GHz. Processors within each node share 16 GB of memory, while nodes are interconnected with a high-speed InfiniBand network (for additional details concerning the machine, see `http://www.oscer.ou.edu/resources.php`). We have used an Intel C compiler, and compiled the code with the following options: “-O3 -march=core2 -mtune=core2.”

The physical memory on a single node of Sooner (16GB) is not large enough for solving the 2D problem with $n_x = 8000$, and $n_y = 16000$. For solving problems of these sizes, virtual memory was used, and this is the reason for larger execution times on 1–8 cores.

**TABLE 1.** Execution time for solving of 2D problem on Sooner.

| $n_x$ | $n_y$ | Cores | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
| 1000 | 1000 | 93.0 | 46.9 | 24.4 | 17.0 | 6.3 | 2.6 | 1.1 | 0.8 | 0.5 | 0.5 | 0.5 |
| 1000 | 2000 | 186.4 | 95.6 | 55.2 | 40.2 | 16.9 | 6.5 | 2.6 | 1.3 | 0.8 | 0.7 | 0.8 |
| 2000 | 2000 | 384.3 | 197.3 | 112.9 | 85.0 | 40.1 | 17.3 | 6.7 | 3.1 | 1.4 | 1.2 | 0.8 |
| 2000 | 4000 | 793.2 | 394.9 | 227.1 | 171.0 | 84.5 | 40.6 | 17.4 | 7.0 | 3.6 | 2.0 | 1.5 |
| 4000 | 4000 | 1739.0 | 848.0 | 464.5 | 346.3 | 170.7 | 85.7 | 41.0 | 18.4 | 7.8 | 3.9 | 2.3 |
| 4000 | 8000 | 4006.5 | 1817.0 | 955.3 | 696.9 | 343.2 | 171.7 | 85.7 | 42.5 | 19.5 | 8.3 | 10.1 |
| 8000 | 8000 | 10007.0 | 4240.1 | 2185.1 | 1605.1 | 694.8 | 347.8 | 173.2 | 90.7 | 44.8 | 21.8 | 47.2 |
| 8000 | 16000 | 29138.7 | 10952.4 | 6117.1 | 4693.7 | 1567.6 | 697.6 | 345.0 | 182.4 | 93.3 | 48.1 | 127.2 |
| 16000 | 16000 | | | | | 4679.1 | 1608.2 | 703.9 | 374.7 | 187.9 | 99.0 | 302.9 |

**TABLE 2.** Execution time for solving of 3D problem on Sooner.

| $n_x$ | $n_y$ | $n_z$ | Cores | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
| 120 | 120 | 120 | 397.3 | 197.2 | 99.2 | 70.6 | 36.4 | 13.3 | 6.3 | 2.9 | 1.8 | 1.3 | 1.3 |
| 120 | 120 | 240 | 825.7 | 423.2 | 271.9 | 213.3 | 106.8 | 37.1 | 13.7 | 7.2 | 3.9 | 2.2 | 1.6 |
| 120 | 240 | 240 | 1809.1 | 904.5 | 598.5 | 493.2 | 252.5 | 108.4 | 37.9 | 16.5 | 8.3 | 4.2 | 3.3 |
| 240 | 240 | 240 | 3811.3 | 1965.5 | 1255.4 | 1105.9 | 498.0 | 217.9 | 75.0 | 39.1 | 16.7 | 9.6 | 4.9 |
| 240 | 240 | 480 | 7811.1 | 4030.9 | 2657.4 | 2356.9 | 1075.2 | 494.4 | 220.6 | 113.4 | 43.8 | 20.1 | 10.5 |
| 240 | 480 | 480 | 16898.5 | 8442.2 | 5869.7 | 4938.2 | 2376.7 | 1079.6 | 504.0 | 265.0 | 118.8 | 46.0 | 20.7 |
| 480 | 480 | 480 | | | | | 4963.8 | 2381.5 | 1122.9 | 515.7 | 233.6 | 86.8 | 44.4 |
| 480 | 480 | 960 | | | | | | 4842.0 | 2387.6 | 1114.4 | 553.6 | 237.5 | 124.5 |
| 480 | 960 | 960 | | | | | | | 5007.6 | 2436.1 | 1156.5 | 532.8 | 288.0 |
| 960 | 960 | 960 | | | | | | | | 5068.8 | 2510.2 | 1209.3 | 543.1 |

The obtained execution times confirm that the communication time between processors is larger than the communication time between cores within one processor. Also, the execution time for solving one and the same discrete problem decreases with increasing the number of cores, which shows that the communication in our parallel algorithm is mainly local.

Tables 3 and 4 present execution time collected on the IBM Blue Gene/P machine at the Bulgarian Supercomputing Center, for 2D and 3D problems. It consists of 2048 compute nodes with quad core PowerPC 450 processors (running at 850 MHz). Each node has 2 GB of RAM. For the point-to-point communications a 3.4 Gb 3D mesh network is used. Reduction operations are performed on a 6.8 Gb tree network (for more details, see `http:`

**TABLE 3.** Execution time for solving of 2D problem on IBM Blue Gene/P.

| | $n_x$ | 1000 | 1000 | 2000 | 2000 | 4000 | 4000 | 8000 | 8000 | 16000 |
|---|---|---|---|---|---|---|---|---|---|---|
| Cores | $n_y$ | 1000 | 2000 | 2000 | 4000 | 4000 | 8000 | 8000 | 16000 | 16000 |
| 1 | | 681.7 | 1415.6 | 2768.5 | 5565.9 | 11424.7 | | | | |
| 2 | | 329.0 | 690.6 | 1408.8 | 2809.5 | 5656.4 | 11619.1 | | | |
| 4 | | 164.6 | 335.6 | 709.7 | 1472.8 | 2886.4 | 5803.2 | 11907.5 | | |
| 8 | | 81.3 | 167.6 | 344.2 | 720.3 | 1468.6 | 2926.7 | 5892.0 | 12067.4 | |
| 16 | | 41.8 | 85.5 | 173.7 | 353.2 | 743.4 | 1539.9 | 3019.1 | 6066.0 | 12423.4 |
| 32 | | 20.4 | 42.1 | 84.6 | 174.1 | 356.5 | 745.9 | 1514.6 | 3030.7 | 6090.4 |
| 64 | | 10.5 | 21.8 | 42.4 | 85.9 | 174.2 | 353.8 | 747.9 | 1547.7 | 3040.7 |
| 128 | | 5.6 | 10.7 | 21.0 | 43.1 | 85.8 | 175.7 | 360.4 | 751.1 | 1529.3 |
| 256 | | 2.9 | 5.7 | 10.8 | 22.3 | 43.7 | 88.2 | 177.8 | 360.5 | 759.2 |
| 512 | | 1.8 | 3.1 | 6.0 | 11.3 | 22.2 | 44.7 | 88.8 | 180.1 | 369.0 |
| 1024 | | 1.0 | 1.9 | 3.3 | 6.3 | 11.8 | 23.8 | 45.5 | 90.8 | 181.9 |
| 2048 | | 0.9 | 1.3 | 2.4 | 3.8 | 7.4 | 13.0 | 24.9 | 47.8 | 94.9 |
| 4096 | | 0.7 | 1.0 | 1.6 | 2.7 | 4.5 | 7.8 | 14.3 | 27.0 | 51.0 |

**TABLE 4.** Execution time for solving of 3D problem on IBM Blue Gene/P.

| | $\mathbf{n_x}$ | 120 | 120 | 120 | 240 | 240 | 240 | 480 | 480 | 480 | 960 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathbf{n_y}$ | 120 | 120 | 240 | 240 | 240 | 480 | 480 | 480 | 960 | 960 |
| **Cores** | $\mathbf{n_z}$ | 120 | 240 | 240 | 240 | 480 | 480 | 480 | 960 | 960 | 960 |
| 1 | | 1623.6 | 3248.3 | 6582.4 | | | | | | | |
| 2 | | 769.5 | 1601.5 | 3264.9 | 6638.6 | | | | | | |
| 4 | | 370.3 | 763.1 | 1621.7 | 3318.0 | 6634.4 | | | | | |
| 8 | | 177.5 | 371.1 | 782.0 | 1662.5 | 3320.4 | 6717.8 | | | | |
| 16 | | 115.2 | 176.0 | 351.4 | 793.7 | 1647.1 | 3355.8 | 6783.0 | | | |
| 32 | | 45.4 | 117.9 | 178.8 | 382.5 | 787.4 | 1663.9 | 3384.2 | 6769.8 | | |
| 64 | | 23.2 | 45.8 | 117.7 | 184.7 | 383.8 | 804.8 | 1700.4 | 3390.2 | 6847.1 | |
| 128 | | 12.5 | 24.4 | 48.5 | 123.1 | 190.0 | 377.8 | 829.6 | 1721.6 | 3497.0 | 6986.4 |
| 256 | | 7.0 | 13.8 | 26.5 | 52.1 | 147.7 | 195.6 | 406.7 | 834.2 | 1746.2 | 3512.0 |
| 512 | | 3.7 | 7.0 | 13.6 | 26.8 | 51.7 | 127.1 | 199.7 | 409.6 | 847.4 | 1768.8 |
| 1024 | | 2.7 | 5.0 | 9.6 | 15.5 | 30.0 | 58.4 | 135.3 | 212.2 | 430.9 | 892.9 |
| 2048 | | 1.9 | 3.6 | 5.8 | 9.3 | 17.9 | 32.6 | 62.2 | 166.2 | 226.1 | 493.2 |
| 4096 | | 1.5 | 2.3 | 3.7 | 6.2 | 11.2 | 19.4 | 37.2 | 70.9 | 186.8 | 311.1 |

**TABLE 5.** Speed-up for solving of 2D problem.

| $\mathbf{n_x}$ | $\mathbf{n_y}$ | **Cores** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
| | | | | | | **Sooner** | | | | | |
| 1000 | 1000 | 1.98 | 3.81 | 5.46 | 14.67 | 36.04 | 84.73 | 121.69 | 184.46 | 170.30 | 192.20 |
| 1000 | 2000 | 1.95 | 3.38 | 4.64 | 11.04 | 28.54 | 71.20 | 147.11 | 227.46 | 272.04 | 244.87 |
| 2000 | 2000 | 1.95 | 3.40 | 4.52 | 9.58 | 22.19 | 57.33 | 125.09 | 270.75 | 328.98 | 471.20 |
| 2000 | 4000 | 2.01 | 3.49 | 4.64 | 9.38 | 19.52 | 45.61 | 112.48 | 258.14 | 405.30 | 532.66 |
| 4000 | 4000 | 2.05 | 3.74 | 5.02 | 10.19 | 20.28 | 42.37 | 94.44 | 223.78 | 440.61 | 739.74 |
| 4000 | 8000 | 2.20 | 4.19 | 5.75 | 11.67 | 23.33 | 46.75 | 94.34 | 205.44 | 481.72 | 397.86 |
| 8000 | 8000 | 2.36 | 4.58 | 6.23 | 14.40 | 28.77 | 57.79 | 110.38 | 223.31 | 458.67 | 211.87 |
| 8000 | 16000 | 2.66 | 4.76 | 6.21 | 18.59 | 41.77 | 84.46 | 159.72 | 312.20 | 606.34 | 229.16 |
| | | | | | | **IBM Blue Gene/P** | | | | | |
| 1000 | 1000 | 2.07 | 4.14 | 8.38 | 16.29 | 33.45 | 64.90 | 121.70 | 234.70 | 375.71 | 647.92 |
| 1000 | 2000 | 2.05 | 4.22 | 8.45 | 16.56 | 33.63 | 64.89 | 132.01 | 249.18 | 452.59 | 724.39 |
| 2000 | 2000 | 1.97 | 3.90 | 8.04 | 15.94 | 32.74 | 65.21 | 131.56 | 255.28 | 457.45 | 835.70 |
| 2000 | 4000 | 1.98 | 3.78 | 7.73 | 15.76 | 31.97 | 64.82 | 129.11 | 249.57 | 490.29 | 887.07 |
| 4000 | 4000 | 2.02 | 3.96 | 7.78 | 15.37 | 32.04 | 65.58 | 133.14 | 261.13 | 513.36 | 964.43 |

`//www.scc.acad.bg/`). We have used the IBM XL C compiler and compiled the code with the following options: "-O5 -qstrict -qarch=450d -qtune=450".

The memory of one node of IBM supercomputer is substantially smaller than on Sooner (2 GB vs. 16 GB) and is not enough for solving 2D problem with even $n_x = 4000$, and $n_y = 8000$; as well as 3D problem with $n_x = n_y = n_z = 240$. We solved these problems on two and more cores in the SMP mode using a single core per processor.

While the obtained parallel performance is quite satisfactory, we believe that it can be further improved. To achieve this goal, we plan to develop a mixed MPI/OpenMP code and to use the nodes of the Sooner with 8 OpenMP processes per node (and 4 OpenMP processes per node of the Blue Gene). This modified code should also allow us to run efficiently on the upcoming machines with 10-core Intel processors (and future computers with ever increasing number of cores per processor). Furthermore, we plan to synchronize the decomposition of the computational domain into sub-domains with the topology of the compute nodes in the Blue Gene connectivity network. In such way we will minimize the communication time in the parallel algorithm.

To complete analysis of the experimental performance data, Tables 5 and 6 show the obtained speed-up while the parallel efficiency is depicted in Tables 7 and 8. Obviously, the reported performance data is limited to the cases for which we were able to solve the problem on a single node (see the explanations about the memory limitations, above).

**TABLE 6.** Speed-up for solving of 3D problem.

| $\mathbf{n_x}$ | $\mathbf{n_y}$ | $\mathbf{n_z}$ | **Cores** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
| | | | | | | | **Sooner** | | | | | |
| 120 | 120 | 120 | 2.01 | 4.00 | 5.63 | 10.92 | 29.95 | 63.09 | 138.47 | 217.22 | 304.06 | 313.48 |
| 120 | 120 | 240 | 1.95 | 3.04 | 3.87 | 7.73 | 22.28 | 60.13 | 114.60 | 211.74 | 368.55 | 509.28 |
| 120 | 240 | 240 | 2.00 | 3.02 | 3.67 | 7.16 | 16.68 | 47.71 | 109.54 | 219.01 | 424.45 | 545.52 |
| 240 | 240 | 240 | 1.94 | 3.04 | 3.45 | 7.65 | 17.50 | 50.80 | 97.45 | 227.97 | 397.11 | 775.29 |
| 240 | 240 | 480 | 1.94 | 2.94 | 3.31 | 7.26 | 15.80 | 35.41 | 68.88 | 178.18 | 387.76 | 738.90 |
| 240 | 480 | 480 | 2.00 | 2.88 | 3.42 | 7.11 | 15.65 | 33.53 | 63.76 | 142.45 | 364.03 | 808.79 |
| | | | | | | | **IBM Blue Gene/P** | | | | | |
| 120 | 120 | 120 | 2.11 | 4.38 | 9.15 | 14.09 | 35.75 | 69.87 | 129.82 | 230.43 | 435.96 | 594.70 |
| 120 | 120 | 240 | 2.03 | 4.26 | 8.75 | 18.46 | 27.54 | 70.88 | 132.86 | 235.95 | 465.97 | 646.38 |
| 120 | 240 | 240 | 2.02 | 4.06 | 8.42 | 18.73 | 36.82 | 55.95 | 135.78 | 250.18 | 482.82 | 687.72 |

**TABLE 7.** Parallel efficiency for solving of 2D problem.

| $\mathbf{n_x}$ | $\mathbf{n_y}$ | **Cores** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
| | | | | | | | **Sooner** | | | | | | |
| 1000 | 1000 | 0.992 | 0.952 | 0.683 | 0.917 | 1.126 | 1.324 | 0.951 | 0.721 | 0.333 | 0.188 | | |
| 1000 | 2000 | 0.975 | 0.845 | 0.580 | 0.690 | 0.892 | 1.113 | 1.149 | 0.889 | 0.531 | 0.239 | | |
| 2000 | 2000 | 0.974 | 0.851 | 0.565 | 0.599 | 0.693 | 0.896 | 0.977 | 1.058 | 0.643 | 0.460 | | |
| 2000 | 4000 | 1.004 | 0.873 | 0.580 | 0.586 | 0.610 | 0.713 | 0.879 | 1.008 | 0.792 | 0.520 | | |
| 4000 | 4000 | 1.025 | 0.936 | 0.628 | 0.637 | 0.634 | 0.662 | 0.738 | 0.874 | 0.861 | 0.722 | | |
| 4000 | 8000 | 1.102 | 1.048 | 0.719 | 0.730 | 0.729 | 0.731 | 0.737 | 0.802 | 0.941 | 0.389 | | |
| 8000 | 8000 | 1.180 | 1.145 | 0.779 | 0.900 | 0.899 | 0.903 | 0.862 | 0.872 | 0.896 | 0.207 | | |
| 8000 | 16000 | 1.330 | 1.191 | 0.776 | 1.162 | 1.305 | 1.320 | 1.248 | 1.220 | 1.184 | 0.224 | | |
| | | | | | | | **IBM Blue Gene/P** | | | | | | |
| 1000 | 1000 | 1.036 | 1.036 | 1.048 | 1.018 | 1.045 | 1.014 | 0.951 | 0.917 | 0.734 | 0.633 | 0.361 | 0.247 |
| 1000 | 2000 | 1.025 | 1.054 | 1.056 | 1.035 | 1.051 | 1.014 | 1.031 | 0.973 | 0.884 | 0.707 | 0.537 | 0.349 |
| 2000 | 2000 | 0.983 | 0.975 | 1.005 | 0.996 | 1.023 | 1.019 | 1.028 | 0.997 | 0.893 | 0.816 | 0.562 | 0.434 |
| 2000 | 4000 | 0.991 | 0.945 | 0.966 | 0.985 | 0.999 | 1.013 | 1.009 | 0.975 | 0.958 | 0.866 | 0.708 | 0.507 |
| 4000 | 4000 | 1.010 | 0.990 | 0.972 | 0.960 | 1.001 | 1.025 | 1.040 | 1.020 | 1.003 | 0.942 | 0.749 | 0.621 |

A super-linear speed-up is observed on up to 128 cores of the Blue Gene. There are at least two reasons for the higher speed-up: the processors on supercomputer are slower than on Sooner and the communication is faster (due to special, extra, networking used in the Blue Gene). It is also worthy observing that as the problem size increases, the parallel efficiency on the supercomputer increases as well (e.g. on 4096 cores, it raises from 25% to 62% for the 2D problem). This shows the overall parallel robustness of the approach under investigation.

Finally, we have decided to compare both computer systems. To this effect, computing times obtained on both parallel systems are shown in the left side of Figure 1 and the obtained speed-up is presented in the right side of the same Figure. As indicated above, due to the slower processors, the execution times obtained on the Blue Gene/P are substantially larger than that on the Sooner. At the same time, the parallel efficiency obtained on the supercomputer is better. The main reason of this can be related to the superior performance of the networking infrastructure of the Blue Gene.

**TABLE 8.** Parallel efficiency for solving of 3D problem.

| $n_x$ | $n_y$ | $n_z$ | Cores | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
| | | | **Sooner** | | | | | | | | | | | |
| 120 | 120 | 120 | 1.007 | 1.001 | 0.703 | 0.691 | 0.936 | 1.025 | 1.082 | 0.849 | 0.594 | 0.306 | | |
| 120 | 120 | 240 | 0.975 | 0.759 | 0.484 | 0.483 | 0.709 | 0.939 | 0.902 | 0.827 | 0.720 | 0.497 | | |
| 120 | 240 | 240 | 0.997 | 0.753 | 0.457 | 0.446 | 0.520 | 0.757 | 0.886 | 0.853 | 0.829 | 0.533 | | |
| 240 | 240 | 240 | 0.967 | 0.757 | 0.430 | 0.477 | 0.545 | 0.795 | 0.766 | 0.888 | 0.776 | 0.757 | | |
| 240 | 240 | 480 | 0.970 | 0.734 | 0.413 | 0.453 | 0.493 | 0.552 | 0.543 | 0.695 | 0.757 | 0.722 | | |
| 240 | 480 | 480 | 1.002 | 0.715 | 0.424 | 0.441 | 0.485 | 0.520 | 0.500 | 0.552 | 0.711 | 0.790 | | |
| | | | **IBM Blue Gene/P** | | | | | | | | | | | |
| 120 | 120 | 120 | 1.055 | 1.096 | 1.143 | 0.881 | 1.117 | 1.092 | 1.014 | 0.900 | 0.851 | 0.581 | 0.420 | 0.267 |
| 120 | 120 | 240 | 1.014 | 1.064 | 1.094 | 1.154 | 0.861 | 1.108 | 1.038 | 0.922 | 0.910 | 0.631 | 0.439 | 0.340 |
| 120 | 240 | 240 | 1.008 | 1.015 | 1.052 | 1.171 | 1.151 | 0.874 | 1.061 | 0.977 | 0.943 | 0.672 | 0.557 | 0.435 |

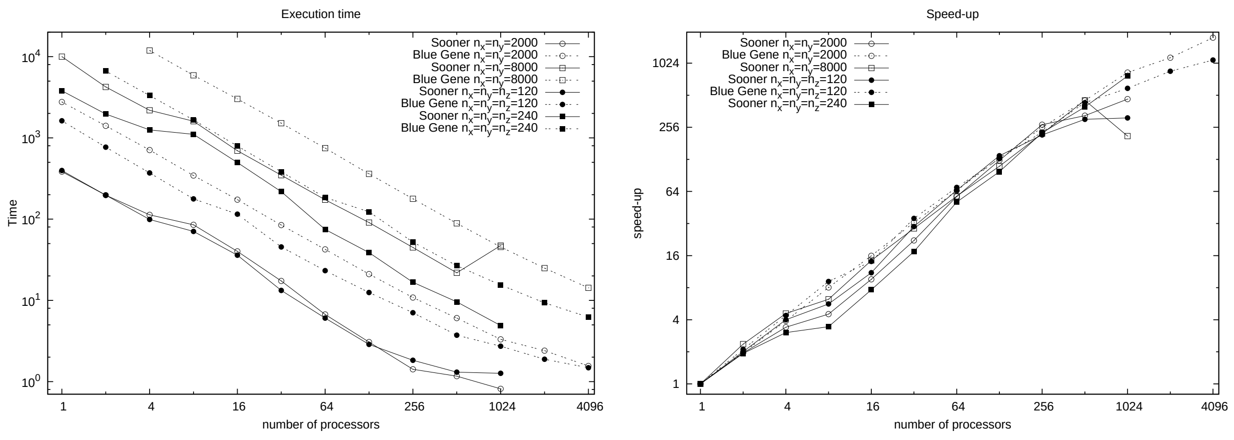

**FIGURE 1.** Execution time and speed-up for 2D problem $n_x = n_y = 2000, 8000$, 3D problem $n_x = n_y = n_z = 120, 240$

## ACKNOWLEDGMENTS

Computer time grants from the Oklahoma Supercomputing Center (OSCER) and the Bulgarian Supercomputing Center (BGSC) are kindly acknowledged. K. Georgiev, I. Lirkov, and S. Margenov were partially supported by grants DO02-147 and DPRP7RP-02/13 of the Bulgarian NSF. Work presented here is a part of the Poland-Bulgaria collaborative grant “Parallel and distributed computing practices”.

## REFERENCES

1. A. J. Chorin, *Math. Comp.* **22**, 745–762 (1968).
2. R. Temam, *Arch. Rat. Mech. Anal.* **33**, 377–385 (1969).
3. J.-L. Guermond, P. Minev, and J. Shen, *Comput. Methods Appl. Mech. Engrg.* **195**, 6011–6054 (2006).
4. J.-L. Guermond, and P. Minev, *Comptes Rendus Mathematique* **348**, 581–585 (2010).
5. J.-L. Guermond, and A. Salgado, *Journal of Computational Physics* **228**, 2834–2846 (2009).
6. J.-L. Guermond, and A. Salgado, *Comptes Rendus Mathematique* **346**, 913–918 (2008).
7. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI: The Complete Reference*, Scientific and engineering computation series, The MIT Press, Cambridge, Massachusetts, 1997, second printing.
8. D. Walker, and J. Dongarra, *Supercomputer* **63**, 56–68 (1996).
9. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users’ Guide*, SIAM, Philadelphia, 1999, third edn.