Martin Lazar

Parallel implementation of an algorithm of the first-order projection method for solving viscous incompressible fluid flow

In: Jan Chleboun and Petr Přikryl and Karel Segeth (eds.): Programs and Algorithms of Numerical Mathematics, Proceedings of Seminar. Dolní Maxov, June 6-11, 2004. Institute of Mathematics AS CR, Prague, 2004.
pp. 143–148.

Persistent URL: http://dml.cz/dmlcz/702787

PARALLEL IMPLEMENTATION OF AN ALGORITHM OF THE FIRST-ORDER PROJECTION METHOD FOR SOLVING VISCOUS INCOMPRESSIBLE FLUID FLOW *

Martin Lazar

Abstract

This paper concerns a parallel algorithm of the first-order projection method for solving the Navier-Stokes equations. We describe the process of parallelization and a particular implementation of the algorithm on a computational cluster. Finally, we compare used iterative methods and discuss the obtained parallel efficiency.

1. Problem formulation

The flow of a viscous incompressible fluid is described by the Navier-Stokes equations

$$\partial_t \boldsymbol{u} - \nu \Delta \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p = \boldsymbol{f} \tag{1}$$

and the equation of continuity

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2}$$

where $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t)$ is the velocity, $\nu = \text{const.}$ is the viscosity, $p = p(\boldsymbol{x}, t)$ is the pressure, and $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}, t)$ is a given function.

We add initial and boundary conditions

$$\begin{aligned} \boldsymbol{u}_{|\boldsymbol{x}\in\partial\Omega} &= 0, \\ \boldsymbol{u}_{|t=0} &= \boldsymbol{u}_0. \end{aligned} \tag{3}$$

We are looking for functions $\boldsymbol{u}$ and p that satisfy the above equations in some appropriate sense.

2. Numerical method

The numerical method we use proceeds in two phases. First, we carry out time-discretization using the projection method. Then we apply the finite-difference method in space at each time level to get the approximate values of the solution at the mesh points.

*This research has been partially supported by the Grant Agency of the Czech Republic under grant No. 201/04/1503.

2.1. Time discretization: the first-order projection method

The projection method discretizes Eqs. (1) and (2) in time. A key feature of the projection method is decoupling the computation of velocity and pressure. The transition to the time level t_{n+1} is described by the following procedure (see [2], [3] for details):

1. calculation of an auxiliary velocity $\hat{\boldsymbol{u}}$ at t_{n+1} (evolution step):

$$\hat{\boldsymbol{u}}^{n+1} = \boldsymbol{u}^n + \tau[\nu\Delta\boldsymbol{u}^n + \boldsymbol{f}^n - (\boldsymbol{u}^n \cdot \Delta)\boldsymbol{u}^n], \tag{4}$$

   where $\tau = t_{n+1} - t_n$;

2. calculation of the pressure p at t_{n+1} by solving the Neumann problem for the Poisson equation:

$$\begin{array}{c} \tau\Delta p^{n+1} = \nabla \cdot \hat{\boldsymbol{u}}^{n+1}, \\ \tau\frac{\partial p^{n+1}}{\partial \boldsymbol{n}} = \hat{\boldsymbol{u}}^{n+1} \cdot \boldsymbol{n}, \quad \boldsymbol{x} \in \partial\Omega; \end{array} \tag{5}$$

3. calculation of the new velocity $\boldsymbol{u}$ at t_{n+1} (projection step):

$$\begin{array}{c} \boldsymbol{u}^{n+1} = \hat{\boldsymbol{u}}^{n+1} - \tau\nabla p^{n+1}, \\ \boldsymbol{u}^{n+1} = 0, \quad \boldsymbol{x} \in \partial\Omega. \end{array} \tag{6}$$

2.2. Space discretization: finite-difference method

The Neumann problem (5) is solved by the finite-difference method. We choose a grid covering the domain Ω and denote

$$\begin{array}{rcl} \hat{\boldsymbol{U}}_i & \approx & \hat{\boldsymbol{u}}(\boldsymbol{x}_i, t_{n+1}), \\ P_i & \approx & p(\boldsymbol{x}_i, t_{n+1}). \end{array}$$

Then, after replacing the derivatives in the PDE by the respective differential quotients we get (in $\mathbf{R}^2$) the grid equations

$$2(\frac{k}{h} + \frac{h}{k})P_{\mathrm{m}} - \frac{k}{h}P_{\mathrm{w}} - \frac{k}{h}P_{\mathrm{e}} - \frac{h}{k}P_{\mathrm{n}} - \frac{h}{k}P_{\mathrm{s}} = \frac{hk}{\tau}(D_x\hat{V}_m + D_y\hat{W}_m), \tag{7}$$

where h, k are space steps and $\mathrm{w}, \mathrm{e}, \mathrm{n}, \mathrm{s}$ denote the points west, east, north, and south from the central point denoted by m, $\hat{\boldsymbol{U}}_i = [\hat{V}_i, \hat{W}_i]^T$, $D_x\hat{V}_m = \frac{1}{2h}(\hat{V}_e - \hat{V}_w)$, and $D_y\hat{W}_m = \frac{1}{2k}(\hat{W}_n - \hat{W}_s)$. This equation has to be modified in the boundary nodes according to the Neumann boundary condition.

If we put

$$\mathbf{P} = [P_1, P_2, \cdots]^T,$$

$$\mathbf{F} = [\cdots, \frac{hk}{\tau}(D_x\hat{V}_m + D_y\hat{W}_m), \cdots]^T,$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{D} & \mathbf{E} & & & \\ \mathbf{E} & \mathbf{D} & \mathbf{E} & & \\ & \mathbf{E} & \mathbf{D} & \mathbf{E} & \\ & & \ddots & \ddots & \ddots \\ & & & \mathbf{E} & \mathbf{D} \end{pmatrix},$$

where $\mathbf{F}$ is the right-hand side vector from (7), $\mathbf{D}$ is the tridiagonal matrix which has $2(\frac{h}{k} + \frac{k}{h})$ on the diagonal and $-\frac{k}{h}$ off the diagonal, and $\mathbf{E}$ is a diagonal matrix with $-\frac{h}{k}$ on the diagonal, we can see that the second step of our procedure (solution of the Poisson equation) yields the system of grid equations $\mathbf{AP} = \mathbf{F}$. This system is singular, its null space being constant functions. We solve it by an iterative method with preconditioning. The null space is removed from the resulting vector whenever the preconditioner is applied, see [1].

3. Parallel algorithm

The aim of parallelization is to speed up the computation using more computational units (*processors*) connected together through a computer network (*computational cluster*). The parallelization of our algorithm has been made by the *domain decomposition method*.

The domain Ω is divided into subdomains approximately of the same size in such a way that each subdomain is assigned to one processor. The computation of auxiliary velocity $\hat{\boldsymbol{u}}$ in the first step of the projection method (evolution step) can be performed by all the processors simultaneously. Each processor computes only the values in the nodes belonging to its own subdomain. To compute the derivatives on the boundary of a subdomain it is necessary that each processor stored also the values of velocity in the boundary nodes of the subdomains of the neighbouring processors (see Fig. 1).

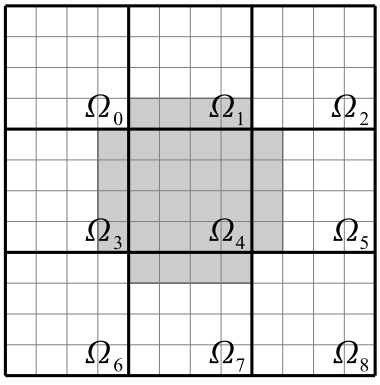

Fig. 1: *Division of Ω into the subdomains. The grid points needed for the computation of derivatives in Ω_4 are marked in gray.*

The computation of the new velocity $\boldsymbol{u}$ in the third step of the projection method (projection step) is similar to the computation of $\hat{\boldsymbol{u}}$.

The system of grid equations $\mathbf{AP} = \mathbf{F}$ from the discretization of the Poisson equation in the second step of the projection method can be solved by an arbitrary method for solving linear systems. Because of sparsity of the matrix A, it is useful to use a parallel iterative method with preconditioning. An overview of tested methods is shown in Table 1. For details about the tested methods and their parallel implementation see [6].

GMRES	Generalized Minimum Residual (with restart after 30 iterations)
CGS	Conjugate Gradient Squared
BiCGStab	Bi-Conjugate Gradient Squared Stabilized
TFQMR	Transpose Free Quasi Minimal Residual
SYMMLQ	Symmetric LQ method
BJacobi	Block Jacobi preconditioning (from left)
ASM	Additive Schwarz Method preconditioning (from left)

Tab. 1: *Tested iterative methods (above) and preconditioners (below).*

4. Numerical results

The program was written in programming language C using the Portable, Extensible Toolkit for Scientific Computation (PETSc, see [1]). All tests were done on cluster Lyra at the Department of Mathematic of Faculty of Applied Sciences at University of West Bohemia in Pilsen. This cluster consists of six[1] SMP nodes having two Pentium III 450 MHz processors connected together through Myrinet network. The relative tolerance for the iterative solvers was 10^{-5} for all the tests.

CPUs	GMRES +BJacobi	BiCGS +BJacobi	CGS (no precond.)	TFQMR (no precond.)	SYMMQR + ASM
1	193.6	176.6	212.3	196.3	150.3
2	107.5	93.4	115.6	106.2	86.3
3	86.6	69.4	78.1	72.7	64.6
4	53.9	46.1	57.0	50.6	39.5
5	50.9	36.6	41.6	41.3	36.8
6	39.0	31.4	37.9	35.0	26.9
7	34.7	26.9	34.7	29.9	25.7
8	31.1	23.7	33.1	28.3	20.4

Tab. 2: *Computation time in seconds for computing 100 time layers on grid of 100×100 nodes.*

[1] At testing time, two nodes were out of function, so all tests was done only on four nodes.

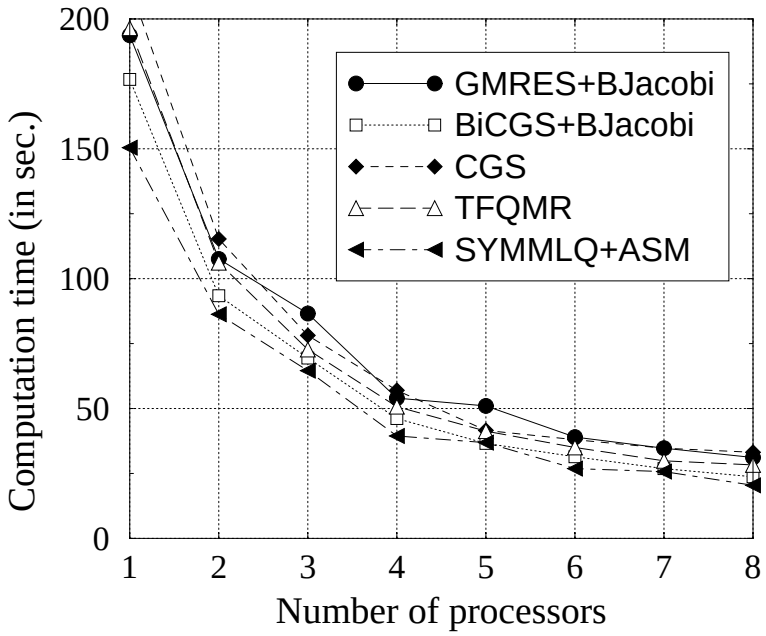

Fig. 2: *Computation time for computing 100 time layers on grid of 100×100 nodes.*

The results of the speed tests are in Table 2 and Fig. 2. We observed that:

- CGS with both preconditioners broke down in 40% of the tests approximately, SYMMLQ with ASM broke down in majority of the tests. Other combinations of iterative solvers and preconditioners did not break down.
- ASM and BJacobi were very similar regarding the number of iterations and the computational time.
- GMRES without preconditioning was approximately 2.6 times slower then with BJacobi or ASM preconditioner. SYMMQR was approximately 1.8 times slower and BiCGS was 1.3 times slower then with a preconditioner. TFQMR and CGS without preconditioner were not slower then the variant with a preconditioner.
- Parallel speedup (ratio of the computing time of the serial and the parallel algorithm) is very close to the ideal speedup, see Fig. 3.

5. Conclusion

Our parallel efficiency (ratio of the parallel speedup and the number of the processors used) is very close to one. We can compare it with the result from [4] where parallel direct solvers were tested on the Lyra cluster. There, the parallel efficiency was about $\frac{1}{3}$. We can see that the projection method on the structured grid with an iterative solver is a very well parallelizable task.

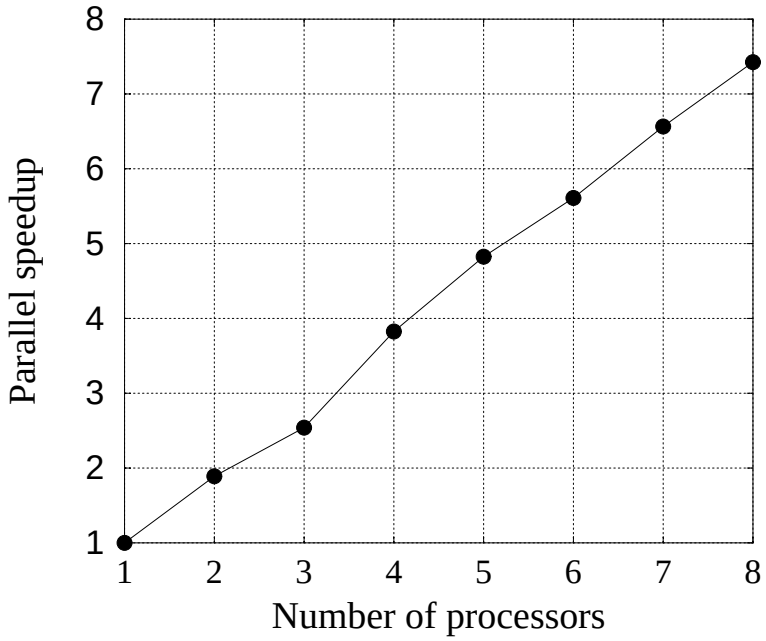

Fig. 3: *Speedup (ratio of the computing time of serial and parallel algorithm).*

References

[1] S. Balay et al.: *PETSc users manual.* Technical Report ANL-95/11 – Revision 2.2.0, Argonne National Laboratory, 2004.

[2] M. Brandner: *Numerical modeling of dynamics of unblended viscous incompressible fluid.* PhD thesis. University of West Bohemia, Pilsen, 2000. (In Czech.)

[3] E. Weinan, J.-G. Liu: *Projection method I: Convergence and numerical boundary layers.* SIAM J. Numer. Anal. **32**, 1995.

[4] M. Lazar: *Linear algebra libraries for massive parallel computing.* Diploma thesis. University of West Bohemia, Pilsen, 2001. (In Czech.)

[5] S. Míka, P. Přikryl: *Numerical methods for solving partial differential equations.* University of West Bohemia, Pilsen, 1995. (In Czech.)

[6] Y. Saad: *Iterative methods for sparse linear systems, 2nd edition.* SIAM, Philadelphia, 2003.