A Parallel ADI Method for a Nonlinear Equation Describing Gravitational Flow of Ground Water

I.V. Schevtschenko

Rostov State University
Laboratory of Computational Experiments on Super Computers
34/1 Communistichesky avenue, Apt. 111
344091, Rostov-on-Don, Russia
ishevtch@uic.rnd.runnet.ru

Abstract. The aim of the paper is an elaboration of a parallel alternating-direction implicit, or ADI, method for solving a non-linear equation describing gravitational flow of ground water and its realization on a distributed-memory MIMD-computer under the MPI message-passing system. Aside from that, the paper represents an evaluation of the parallel algorithm in terms of relative efficiency and speedup. The obtained results show that for reasonably large discretization grids the parallel ADI method is effective enough on a large number of processors.

Keywords: gravitational flow of ground water, finite difference method, Peaceman-Rachford difference scheme, parallel ADI method, conjugate gradient method.

1 Introduction

The past decades a substantial progress has been made in mathematical description of water flow and pollutant transport processes. Many today's mathematical models are available to predict admixtures migration in ground water under diverse conditions of processes progress. Approximation of such models generates large systems of linear algebraic or differential equations that demands utilizing modern supercomputers proposing powerful computational resources to solve large problems in various fields of science. In particular, by solving the equation of pollutant transport in ground water it is necessary to know a level of ground water in a water-bearing stratum described by the balance mass equation. In this paper we consider a parallel solution of that equation for approximation of which the finite difference method is used. The solution of the problem is found with the aid of the ADI method, in particular, with using Peaceman-Rachford difference scheme [8]. We exploit natural parallelism of the difference scheme to apply it to distributed-memory MIMD-computers.

An outline of the paper is as follows. Section 2 introduces to general formulation and numerical approximation of the original mathematical model, represents

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 904–910, 2001.

some results on accuracy and stability of the used difference scheme and substantiates applicability of the conjugate gradient (CG) method to solving systems of linear algebraic equations (SLAEs) generated by Peaceman-Rachford difference scheme. Section 3 is represented by a parallel realization of the ADI method. In the same place we evaluate the parallel algorithm in terms of relative efficiency and speedup. Finally, in section 4, we give our conclusions.

2 General Formulation and Numerical Approximation

One of the existing non-linear models describing gravitational flow of ground water in an anisotropic element of a water-bearing stratum Ω can be represented in the form [2]

$$a^* \frac{\partial h}{\partial t} = \sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left(\rho h k \frac{\partial h}{\partial x_i} \right) + \rho(v_{(x)} + v_{(y)}). \tag{1}$$

Here $x = x_1$, $y = x_2$; ρ is the water density, $h(x, y, t)$ is the level of ground water, k is the filtrational coefficient, $v_{(x)}$ and $v_{(y)}$ are the filtration velocities from below and from above of the water-bearing stratum respectively. Parameter $a^* > 0$ depends on physical characteristics of the water-bearing stratum.

For equation (1) we can define the initial condition

$$h(x, y, t = 0) = h_0(x, y),$$

where $h_0(x, y)$ is a given function and Dirichlet boundary value problem

$$h|_{\partial\Omega} = f(x, y).$$

Here $f(x, y)$ is a function prescribed on the boundary of the concerned field.

Approximation of equation (1) bases on Peaceman-Rachford difference scheme, where along with the prescribed grid functions $h(x, y, t)$ and $h(x, y, t+\tau)$ an intermediate function $h(x, y, t+\frac{\tau}{2})$ is introduced. Thus, passing (from n time layer to $n+1$ time layer) is performed in two stages with steps 0.5τ, where τ is a time step.

Let us introduce a grid of a size $M \times N$ in a simply connected area $\Omega = [0, a] \times [0, b]$ with nodes $x_i = i\Delta x$, $y_j = j\Delta y$, where $i = 1, 2, \ldots M$, $j = 1, 2, \ldots N$, $\Delta x = \frac{a}{M}$, $\Delta y = \frac{b}{N}$. By denoting $h = h^n$, $\bar{h} = h^{n+\frac{1}{2}}$, $\hat{h} = h^{n+1}$, $h_x = \frac{h_{i+1,j} - h_{ij}}{2}$, $h_y = \frac{h_{i,j+1} - h_{ij}}{2}$, $h_{\bar{x}} = \frac{h_{ij} - h_{i-1,j}}{2}$, $h_{\bar{y}} = \frac{h_{ij} - h_{i,j-1}}{2}$, $h_{\underline{x}} = \frac{h_{i+1,j} + h_{ij}}{2}$, $h_{\underline{y}} = \frac{h_{i,j+1} + h_{ij}}{2}$, $h_{\tilde{x}} = \frac{h_{i-1,j} + h_{ij}}{2}$, $h_{\tilde{y}} = \frac{h_{i,j-1} + h_{ij}}{2}$ let us write out the difference approximation for equation (1)

$$\begin{cases} a^* \frac{\bar{h}_{ij} - h_{ij}}{0.5\tau} = \frac{\rho_{\underline{x}} k_{\underline{x}} h_{\underline{x}} \bar{h}_x - \rho_{\tilde{x}} k_{\tilde{x}} h_{\tilde{x}} \bar{h}_{\tilde{x}}}{\Delta x} + \frac{\rho_{\underline{y}} k_{\underline{y}} h_{\underline{y}} h_y - \rho_{\tilde{y}} k_{\tilde{y}} h_{\tilde{y}} h_{\bar{y}}}{\Delta y} + \phi, \\ a^* \frac{\hat{h}_{ij} - \bar{h}_{ij}}{0.5\tau} = \frac{\rho_{\underline{x}} k_{\underline{x}} h_{\underline{x}} \bar{h}_x - \rho_{\tilde{x}} k_{\tilde{x}} h_{\tilde{x}} \bar{h}_{\tilde{x}}}{\Delta x} + \frac{\rho_{\underline{y}} k_{\underline{y}} \bar{h}_{\underline{y}} \hat{h}_y - \rho_{\tilde{y}} k_{\tilde{y}} \bar{h}_{\tilde{y}} \hat{h}_{\bar{y}}}{\Delta y} + \phi. \end{cases} \tag{2}$$

Here $\phi = \rho_{ij}(v_{(x)ij} + v_{(y)ij})$. The difference approximation of the initial condition and Dirichlet boundary value problem can be represented as

$$h|_{t=0} = h_{(0)ij}, \quad h|_{\partial\Omega} = f_{ij}. \tag{3}$$

By addressing stability investigation of equation (2) we formulate the following lemma

Lemma 1 *Peaceman-Rachford difference scheme for equation (1) with Dirichlet's boundary conditions at $a^* > 0$ is stable.*

Concerning the difference scheme (2) it can be noted that it has second approximation order [9] both in time and in space.

By using natural regulating of unknown values in the computed field let us reduce difference problem (2), (3) to the necessity of solving SLAEs $A_k u_k = f_k, k = 1, 2$ with special matrices. The coefficient matrices $A_k, k = 1, 2$ are not constant here. The obtained SLAEs had been scaled, i.e. the elements of the coefficient matrices and RHSs: $A_k = (a_{ij})_k^{MN}, f_k, k = 1, 2$ had the following form

$$\hat{a}_{ij} = \frac{a_{ij}}{\sqrt{a_{ii}a_{jj}}}, \ \hat{f}_i = \frac{f_i}{a_{ii}}, \ i, j = 1, 2, \ldots, MN$$

and solved with the CG method [9] afterwards. The selection of the CG method is based on its acceptable calculation time in comparison with simple iteration, Seidel, minimal residual and steepest descent methods [5].

To proceed, we note that from previous lemma we can infer the appropriateness of using the CG method since

$$A_k = (A_k)^T, \ A_k > 0, \ k = 1, 2.$$

3 Algorithm Parallel Scheme

Before passage to the description of the parallel algorithm we would like to say a few words about the computational platform on which the algorithm has been run and the library with the help of which it has been realized.

The computational system nCube 2S is a MIMD-computer of hypercubic architecture. The number of computational nodes is 2^n, $n \leq 13$. These nodes are unified into communication scheme of a multidimensional cube with maximum length of a communication line equaled to n. Such a communication scheme allows to transmit messages fast enough (channel capacity is 2.5 MBytes/s) irrespective of computational process since each node has a communication coprocessor aside from a computational processor. At our disposal we had the described system in reduced version: 64 nodes with peak performance of 128 MFlops and 2048 MBytes of memory.

Relative to the paradigm of message passing it can be noted that it is used widely on certain classes of parallel machines, especially those with distributed

memory. One of representatives of this conception is MPI (Message Passing Interface) [6]. As we can see from the title, MPI destines for supporting parallel applications to work in terms of the message passing system and allows to use its functions in C/C++ and Fortran 77/90. Besides, amongst a number of books devoted to various aspects of using MPI we can mention, for instance, the following [3], [4], [7].

The parallel algorithm for solving equation (2), as mentioned above, bases on natural parallelism which is suggested by Peaceman-Rachford difference scheme. Using the ADI method gives an opportunity to exploit any method to solve SLAEs obtained on $n + \frac{1}{2}$ and $n + 1$ time layers. In our case we use the CG method. Along with it, application of Peaceman-Rachford difference scheme to equation (1) allows to find numerical solution of the SLAEs on each time layer independently, i.e. irrespective of communication process. The main communication loading lies on connection between two time layers. Thus, one step of the algorithm to be executed requires two interchanges of data at passage to $n + \frac{1}{2}$ and $n + 1$ time layers.

As mentioned before, the ADI method generates two SLAEs with special matrices. One of those matrices obtained on $n+\frac{1}{2}$ time layer is a band tridiagonal matrix and consequently can be transformed by means of permutation of rows to a block tridiagonal matrix, while the second one, obtained on $n+1$ time layer, is a block tridiagonal matrix primordially.

Taking into account aforesaid one step of the parallel algorithm for solving equation (2) with SLAEs $A_k X_k = B_k,\ k = 1, 2$ can be represented in the following manner

1. Compute B_1 on $n + \frac{1}{2}$ time layer.
2. Make the permutation of vectors $X_1^{(0)}$, B_1, where $X_1^{(0)}$ is an initial guess of the CG method on $n + \frac{1}{2}$ time layer.
3. Solve equation $A_1 X_1 = B_1$ on $n + \frac{1}{2}$ time layer with the CG method.
4. Compute B_2 on $n + 1$ time layer step partially, i.e. without the last item of the second equation (2).
5. Make the permutation of vectors $X_2^{(0)} = X_1$, B_2, where $X_2^{(0)}$ is an initial guess of the CG method on $n + 1$ time layer.
6. Compute the missing item so as the computation of B_2 on $n + 1$ time layer has been completed.
7. Solve equation $A_2 X_2 = B_2$ on $n + 1$ time layer with the CG method.
8. Set $X_1^{(0)} = X_2$ and go to point 1 to do the next step of the algorithm.

Let us consider the described algorithm in more detail. Suppose, we have p processors and it is to solve a system of a size $M \times N$. We proceed from the assumption that $\left\{\frac{M}{p}\right\} = 0$ and $\left\{\frac{N}{p}\right\} = 0$, where $\{x\}$ is a fractional part of number x, i.e. vectors $X_k^{(0)},\ B_k,\ k = 1, 2$ are distributed uniformly.

First step of the algorithm is well-understood while the second one claims more attention. Let vectors $X_1^{(0)}$, B_1 be matrices (which are distributed in the rowwise manner) consist of elements of corresponding vectors, then to solve

equation $A_1X_1 = B_1$ on $n + \frac{1}{2}$ time layer with the CG method in parallel we need to transpose the matrix corresponding to vector $B_1 = \{b_1, b_2, \ldots, b_{MN}\}$

$$\begin{pmatrix} b_1 & b_2 & \ldots & b_N \\ b_{N+1} & b_{N+2} & \ldots & b_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ b_{(M-1)N+1} & b_{(M-1)N+2} & \ldots & b_{MN} \end{pmatrix} \to \begin{pmatrix} b_1 & b_{N+1} & \ldots & b_{(M-1)N+1} \\ b_2 & b_{N+2} & \ldots & b_{(M-1)N+1} \\ \ldots & \ldots & \ldots & \ldots \\ b_N & b_{2N} & \ldots & b_{MN} \end{pmatrix}$$

and the matrix corresponding to vector $X_1^{(0)}$. Of course, such a transposition requires transmission of some sub-matrices ($\frac{M}{p} \times \frac{N}{p}$ size) to the corresponding processors. Thus, the number of send/receive operations $C_{s/r}$ and the amount of transmitted data C_t (in element equivalent) are

$$C_{s/r} = 2p(p-1), \quad C_t = 2M\left(N - \frac{N}{p}\right).$$

Further, in accordance with the algorithm to avoid extra communications we compute vector B_2 partially and then permute vectors $X_2^{(0)} = X_1$, B_2 as above. Afterwards, we complete the computation of vector B_2 (its missing item) and solve equation $A_2X_2 = B_2$ on $n+1$ time layer with the CG method in parallel. By resuming aforesaid one step of the algorithm to be run requires

$$C_{sr} = 4p(p-1), \quad C_t = 4M\left(N - \frac{N}{p}\right),$$

$$C_c = \frac{N}{p}\left((25m-7)\left(I_{CG}^{n+\frac{1}{2}} + I_{CG}^{n+1}\right) + 90M - \frac{8M}{p} - 26\right) + 24M + 4.$$

Here $I_{CG}^{n+\frac{1}{2}}$ and I_{CG}^{n+1} are the number of iterations of the CG method in solving equation (2) on $n+\frac{1}{2}$ and $n+1$ time layers, and C_c is a computational complexity of the algorithm.

At this we finish the description of the parallel algorithm and consider some test experiments all of which are given for one step of the algorithm and for $I_{CG}^{n+\frac{1}{2}} = I_{CG}^{n+1} = 1$. The horizontal axis, in all the pictures, is $2^p, p = 0, 1, \ldots, 6$.

By following [10] let us consider relative efficiency and speedup

$$S_p = \frac{T_1}{T_p}, \quad E_p = \frac{S_p}{p},$$

where T_p is a time to run a parallel algorithm on a computer with p processors ($p > 1$), T_1 is a time to run a sequential algorithm on one processor of the same computer.

As we can see from figure 1 relative speedup and efficiency are satisfactory even for a grid of $N = M = 512$ size.

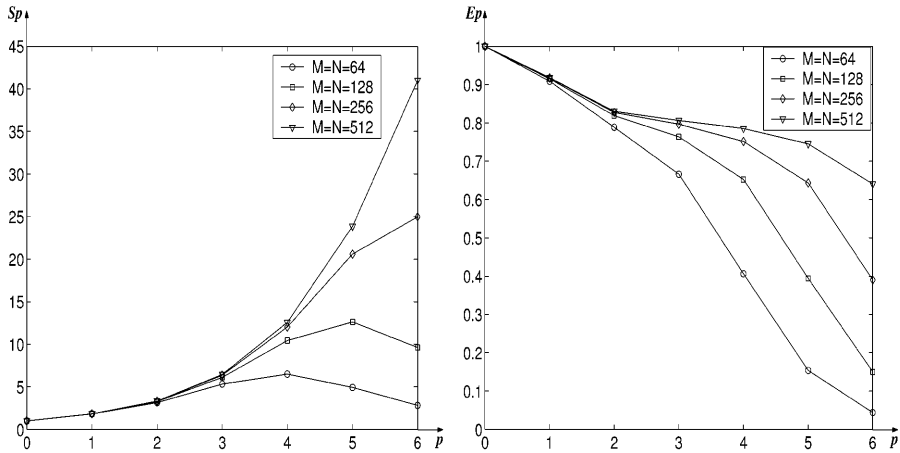

Fig. 1. Relative speedup (to the left) and efficiency (to the right) of the algorithm at various grid sizes.

4 Conclusion

In conclusion we would like to say a few words about further work which will be aimed at elaboration of a parallel ADI method for solving the following equation

$$a^* \frac{\partial h}{\partial t} = \sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left(\rho h k \frac{\partial h}{\partial x_i} \right) + \sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left(\rho h k \frac{\partial \zeta}{\partial x_i} \right) + \rho \left(v_{(x)} + v_{(y)} \right),$$

which is one of the base equations in solving the problem of gravitational flow of ground water.

References

1. R. Barrett, M. Berry, T.F. Chan, J. Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, Henk Van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, http://www.netlib.org/templates/Templates.html
2. J. Bear, D. Zaslavsky, S. Irmay, Physical principles of water percolation and seepage. UNESCO, (1968)
3. I. Foster, Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering, Addison-Wesley Pub. Co., (1995)
4. W. Gropp, E. Lusk, A. Skjellum, T. Rajeev, Using MPI: Portable Parallel Programming With the Message-Passing Interface, Mit press, (1999)
5. L.A. Krukier, I.V. Schevtschenko, Modeling Gravitational Flow of Subterranean Water. Proceedings of the Eighth All-Russian Conference on Modern Problems of Mathematical Modeling, Durso, Russia, September 6-12, RSU Press, (1999), 125-130

6. *MPI: A Message-Passing Interface Standard, Message Passing Interface Forum*, (1994)
7. P. Pacheco, Parallel Programming With MPI, Morgan Kaufamnn Publishers, (1996)
8. D. Peaceman and J. H.H. Rachford, The numerical solution of parabolic and elliptic differential equations. J. Soc. Indust. Appl. Math., No.3 (1955), 28-41
9. A.A. Samarskii and A.V. Goolin, Numerical Methods, Main Editorial Bord for Physical and Mathematical Literature, (1989)
10. Y. Wallach, Alternating Sequential/Parallel Processing, Springer-Verlag, (1982)