16th INTERNATIONAL DAAM SYMPOSIUM
"Intelligent Manufacturing & Automation: Focus on Young Researchers and Scientists"
19 – 22nd October 2005

PARALLEL SOLUTION OF IMPLICIT NUMERICAL SCHEME FOR OPEN CHANNEL FLOW EQUATIONS

Kranjčević, L.; Družeta, S. & Čarija, Z.

***Abstract**: Open channel flow equations are solved by implicit numerical scheme. Implicit numerical scheme requires considerable computational effort consequent upon the need to solve block tridiagonal system of equations per each time step, and thus parallel solution is employed. Research on different linear solvers has become increasingly complex, and this trend is likely to accentuate because of the growing need to design efficient sparse matrix algorithms for modern parallel computers.*

***Key words:** Parallel computation, open channel flow equations, direct linear solver parallelization*

1. INTRODUCTION

Upon investigation of efficient ways to solve systems of open channel flow and analysis of different numerical schemes, parallelization represents one logical step further in that direction. Since implicit numerical scheme is unbound by Courant Friedrichs Lewy (CFL) condition allowing in certain type of problems large time steps and thus reaching the solution very fast, aditional speed up is gained by parallelization of the numerical scheme. Implicit methods solve for the unknowns in a domain simultaneously, requiring solution of simultaneous algebraic equations per each time step. Given the hardware limitations of two processor computer and one dimensional finite difference problem a special type of direct linear solver is constructed and tested.

2. PARALLEL SOLUTION OF THE LINEAR SYSTEM

The one-dimensional St. Venant system of governing equations, based on conservation of mass and conservation of momentum for unsteady flow in a nonprismatic channel of arbitrary cross section, can expressed in vector form reads:

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial}{\partial x}\mathbf{F}(\mathbf{w}) = \mathbf{S}(x, \mathbf{w})$$

$$\mathbf{w} = \begin{pmatrix} A \\ Q \end{pmatrix}, \mathbf{F}(\mathbf{w}) = \begin{pmatrix} Q \\ \dfrac{Q^2}{A} + gI_1(x, A) \end{pmatrix}$$

$$\mathbf{S}(x, \mathbf{w}) = \begin{pmatrix} q_L \\ gI_2(x, A) + u_x q_L + gA(S_b - S_f) \end{pmatrix}$$

in which ***w*** is a vector of flow variables, ***F*** represents flux vector, and ***S*** source term vector. Further on t is time; x is the horizontal distance along the channel; A is the wetted cross-sectional area; Q is disharge and g is the gravitational acceleration. The friction slope is represented by S_f, the bed slope by S_b, and the hydrostatic pressure force by I_1, and the pressure force due to longitudinal channel width variation by I_2.

Direct consequence of one-dimensional open channel model solution with implicit numerical scheme is the system of linear equations that needs to be solved in each time step. For solution of one-dimensional mathematical model of that type, employment of finite difference discretization method is a logical choice. Finite difference method produces block tridiagonal linear system of equations with block matrices [2 x 2], that is to be solved in each time step:

$$\mathbf{A}\mathbf{w}_{i-1}^{n+1} + \mathbf{B}\mathbf{w}_{i}^{n+1} + \mathbf{C}\mathbf{w}_{i+1}^{n+1} = \mathbf{D}$$

where **A, B, C** are block matrices that are elements of a global matrix of the system and **D** represents RHS vector containing source term, some data from previous time level t^n and eventually, imposed boundary conditions. For brevity, treatment of boundary conditions in implicit case will not be discussed here. Matrices *A,B,C* are of order *n=2* and solution block vector *X* and right hand side vector *D* have length 2 since mathematical model consists of two partial differential equations.

Linear solver parallelization can lead into two basic directions:

- iterative parallel linear solver with different overrelaxation techniques or with different preconditioners;
- direct solver.

Two different aspects of the iterative solution of a linear system are distinguished. The first one is the particular acceleration technique that is used to construct a new approximation for the solution with information from previous approximations. This leads to specific iteration methods, such as SOR, Conjugate Gradients, etc. The second aspect is preconditioning of a given system to one that can be more efficiently parallelized. Second aspect is much more employed and usualy combined with different domain partitioners (CHACO, METIS, PARMETIS). These principles are embedded in widely used programs for solution of linear systems such as AZTEC, PETSc, BlockSolve95, ScaLAPACK, PLAPACK, pARMS, PARDISO etc.

For the direct solution of tridiagonal matrices there is not much that can be done in parallel if we do not want to do additional work by rearranging the order of computation. The different parallel techniques for the factorization of tridiagonal matrices fall into four classes: twisted factorization, recursive doubling, cyclic reduction and divide and conquer algorithm. Last three of the four mentioned methods are numerically too expensive whether because parallelization is too fine grained thus yielding too much communication or because parallel algorithms are much more computationaly expensive than simple serial Gaussian elimination and therefore impractical. Since on the hardware side test computer available was two-processor Intel Xeon workstation, parallel technique choosen was a version of twisted factorization method or also known as *Burn At Both Ends (BABE)* algorithm. The degree of parallelism of that type of procedure is clearly two. MPI communication protocol was used in this test.

This parallel algorithm amounts to starting Gauss Jacobi procedure from top-down and from bottom-up simultaneously. Parallel method is presented by pictures Fig. 1, Fig. 2. where *p1* represents master process, and *p2* worker process. Solution

process commences by communication where master process (*p1*) sends worker (*p2*) its share of system matrix i.e. bottom *nw* rows of matrix, and master clearly keeps *nm* rows (upper part) considering $n = nm + nw$. *nw* and *nm* need not be equal necesarilly, since load balancing technique might be employed. Master starts computation by normalizing block rows from top-down turning block *B* into identity matrix *I* and then eliminating block *A* underneath the current row. Worker starts from bottom-up normalizing current block row by turning block *B* into identity matrix *I* and then eliminating block *C* from upper row (Fig. 1.)

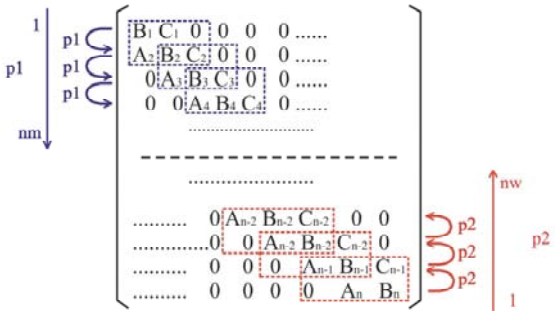

Fig. 1. First half of the solution process

After *nm* i.e. *nw* steps of normalization and *nm-1* and *nw-1* elimination steps block matrix two processes meet and then miminal communication is needed, as shown in Fig. 2. That communication *contact* represents the only sinchronization point in whole procedure and processes then turn their ways back. Worker starts eliminating matrices *A* on the way down, and master eliminates matrices *C* on the way back up.

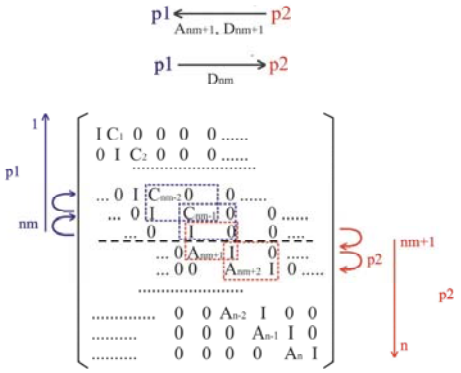

Fig. 2. Sinchronization and second half of the calculation process

After all the eliminations, block matrix is left with only identity matrices *I* on diagonal i.e. turning into identity matrix itself. Solution vector *X* is preserved in the right hand side vector *D*. Mathematicaly, this parallel Gauss-Jordan procedure matrix system

$$T\,X = D$$

is multiplied by vector product from the left with matrix T^{-1}

$$T^{-1}T\,X = T^{-1}D$$

$$X = T^{-1}D\,.$$

With one final communication message in solution procedure, master receives *nw* block rows of vector *D* containing solutions calculated by worker.

Since the goal of this paper is analysis of the parallel algorithm only, results obtained by calculation won't be presented. Test problem employed in this paper was chosen to emphasise parallel efficency of the code presented. Test case is an idealized dam-break flow in a rectangular, L=1 m long, frictionless channel with variable bottom topography which represents bump represented by some function B(x) situated in the middle part of the channel. Imaginary dam is located at x=0.5 m and it separates reservoir *h=1*m and the tail water part *h=0.5*m. At time t=0 s dam is removed instantly, and water is released into the downstream side in the form of a shock wave. Implicit numerical method is used with CFL number set to CFL=3. Boundary conditions are reflective walls on both sides causing shock wave to reflect back and forth. After computational time T=3.5 s shock wave travels the lenght of the channel approximately 10 times, calculation is stopped and wall time measured.

Method	CPU TIME [s]
Gaussian elimination serial	14,83
Gauss Jordan serial	15,2
Gauss Jordan parallel	7,849

Table 1. Run times comparison

5. CONCLUSION

Parallel algorithm was successfully constructed and implemented in calculation of given physical problem. Parallel code showed excellent parallel efficency Table 1. Even though direct methods are rarely paralelized, in this case it was justified considering the hardware given. Furthermore, it is important to notice rapid increase in number of two processor machines entering lately even the field of home PCs. Dual Intel Xeon parallel computer as being parallel computer with shared memory, allows also usage of OPEN MP communication protocol. Even though OPEN MP is easier to program having simpler *fork-join* parallelization principles with implicit communication directives, MPI protocol was chosen because of the portability reasons. MPI protocol allows the implementation of the algorithm on the both shared and distributed memory parallel computers (for instance LAN connected PC-c). Major reason why communication needs to be minimized lays in fact that in distributed memory parallel environment communication stands for major latency cause.

6. REFERENCES:

Saad, Y., Iterative Methods for Sparse Linear Systems, Copyright 2000 by Y. Saad, 2000.

Bermudez, A.,Vazquez,M.E., "Upwind methods for hyperbolic conservation laws with source terms", Comput.&Fluids 23(8),1049, 1994.

Garcia-Navarro, P., Priestly, A., Alcrudo, F., "An Implicit Method for Water Flow Modeling in Channel and Pipes", Journal of Hydraulic Research, Vol. 32, No.5, 1994.

Pacheco, P.S., Ming, W.C., MPI User Guide in FORTRAN

Rabenseifner, R., Resch. M. Hardware Architectures and Parallel Programming Models An Introduction, Parallel Programming Workshop, Stuttgart 2004.

Dongarra, J.J., Eijkhout, V. Numerical linear algebra algorithms and software, Journal of Computational and Applied Mathematics, Vol 123, Issues1-2, 2000. pag. 489-514