Parallel Incomplete Factorization of 3D NC FEM Elliptic Systems

Yavor Vutov

Institute for Parallel Processing, Bulgarian Academy of Sciences

Abstract. A new parallel preconditioner for solution of large scale second order 3D FEM elliptic systems is presented. The problem is discretized by rotated trilinear non-conforming finite elements. The algorithm is based on application of modified incomplete Cholesky factorisation (MIC(0)) to a locally constructed modification B of the original stiffness matrix A. The matrix B preserves the robustness of the pointwise factorisation and has a special block structure allowing parallelization. The performed numerical tests are in agreement with the derived estimates for the parallel times.

1 Introduction

We consider the model elliptic boundary value problem:

$$\begin{aligned} Lu \equiv -\nabla \cdot (a(x)\nabla u(x)) &= f(x) \text{ in } \Omega, \\ u &= 0 \text{ on } \Gamma_D, \\ (a(x)\nabla u(x)) \cdot n &= 0 \text{ on } \Gamma_N, \end{aligned} \tag{1}$$

where $\Omega = [0,1]^3 \subset \mathbb{R}^3$, $\Gamma_D \cup \Gamma_N = \partial\Omega$ and $a(x)$ is a symmetric and positive definite coefficient matrix. The problem is discretized using non-conforming finite elements method (FEM). The resulting linear algebraic system is assumed to be large. The stiffness matrix A is symmetric and positive definite. For large scale problems, the preconditioned conjugate gradient (PCG) method is known to be the best solution method [1].

The recent efforts in development of efficient solution methods for non-conforming finite element systems is inspired by their importance for various applications in scientific computations and engineering [10,3,9]. The goal of this study is to develop a new parallel PCG solver for the arising 3D FEM elliptic systems. A locally modified approximation of the global stiffness matrix is proposed allowing for: a) a stable MIC(0) (modified incomplete Cholesky) factorization; and b) a scalable parallel implementation. The considered non-conforming FEM and MIC(0) factorization are robust for problems with possible jumps of the coefficients

The algorithm is based on the experience in developing such kind of algorithms for 2D problems using conforming FEM elements on skewed meshes [8] and non-conforming rotated bilinear FEM elements [5,9,6].

The rotated trilinear non-conforming finite elements on hexahedrons are used for the numerical solution of (1).

We assume that $\Omega^h = w_1^{h_1} \times w_2^{h_2} \times w_3^{h_3}$ is a decomposition of the computational domain $\Omega \subset \mathbb{R}^3$ into hexahedrons. The degrees of freedom are associated with the midpoints of the sides. The standard computational procedure leads to the linear system of equations $Ax = b$, where the stiffness matrix A is sparse, symmetric and positive definite.

The rest of this paper is organised as follows. Section 2 describes the element-by-element construction of the preconditioner. Section 3 contains the parallel implementation details and estimates of parallel times. Some results from numerical experiments, are presented in Section 4. Short concluding remarks are given at the end.

2 Preconditioning strategy

We use PCG algorithm with preconditioner based on MIC(0) factorization. The MIC(0) factorization of a real sparse symmetric matrix A has the form:

$$C_{MIC(0)}(A) = (X - L)\, X^{-1} \left(X - L^T\right). \tag{2}$$

Here $(-L)$ is the strictly lower triangular part of A, and X is a diagonal matrix. Since we are going to use $C_{MIC(0)}$ as a preconditioner, we are interested in the case when $X > 0$. This holds when A is a M matrix. More details about MIC(0) factorization can be found in [7,4].

To solve the system with preconditioner (2) one have to solve one system with lower triangular matrix, one with upper triangular and one with diagonal matrix. The solution of the systems with triangular matrices is based on recursive computations. That is why the PCG algorithm with MIC(0) factorization as a preconditioner is inherently sequential. To construct a parallel MIC(0) solver, we introduce a locally constructed approximation B of the original stiffness matrix A.

Following the standard FEM assembling procedure we write A in the form $A = \sum_{e \in \omega_h} L_e^T A_e L_e$, where A_e is the element stiffness matrix, L_e stands for the restriction mapping of the global vector of unknowns to the local one corresponding to the current element e. Let us consider the following approximation B_e of A_e.

$$A_e = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix} \quad B_e = \begin{bmatrix} b_{11} & 0 & a_{13} & a_{14} & a_{15} & a_{16} \\ 0 & b_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & b_{33} & 0 & a_{35} & a_{36} \\ a_{41} & a_{42} & 0 & b_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & b_{55} & 0 \\ a_{61} & a_{62} & a_{63} & a_{64} & 0 & b_{66} \end{bmatrix}$$

The local numbering follows the pairs of the opposite nodes of the reference element. On fig. 1 (a) and (b) are shown the connectivity patterns of the matrices

A_e and B_e. The modification actually removes the links between the degrees of freedom on pairs of opposite sides. The diagonal entries of B_e are modified to hold the rowsum criteria. Assembling the locally defined matrices B_e we get the global one $B = \sum_{e \in \omega_h} L_e^T B_e L_e$,. The sparsity structure of the matrices A and B

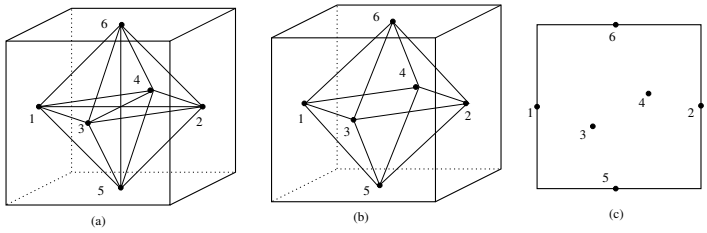

Fig. 1. (a) Connectivity pattern of A_e;(b) Connectivity pattern of B_e; (c) 2D projection of the finite element used in further figures.

is illustrated by Figure 2. Lexicographic node numbering is used. The important property of the matrix B is that its diagonal blocks are diagonal matrices.

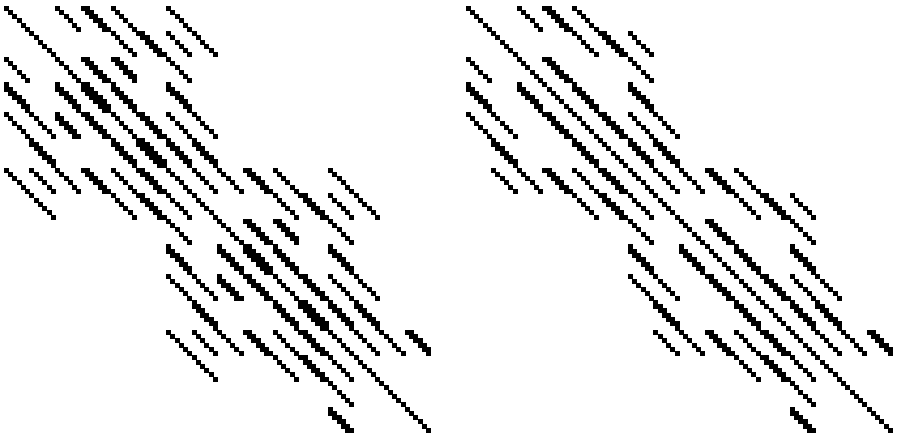

Fig. 2. Sparsity pattern of the matrices A (on the left) and B (on the right), for the division of Ω into 2x2x6 hexahedrons. Non-zero elements are drawn as black dots.

It can be shown that matrices A and B are spectrally equivalent with an uniform estimate for the condition number $\kappa(B^{-1}A)$. We can now introduce the preconditioner $\mathcal{C}$ for A which is defined as a MIC(0) factorization of B, that is, $\mathcal{C} = \mathcal{C}_{MIC(0)}(B)$. This needs of course B to allow for a stable MIC(0) factorization which has been analysed in [5] for the 2D case. The diagonal blocks of B allow a parallel implementation of the resulting PCG algorithm.

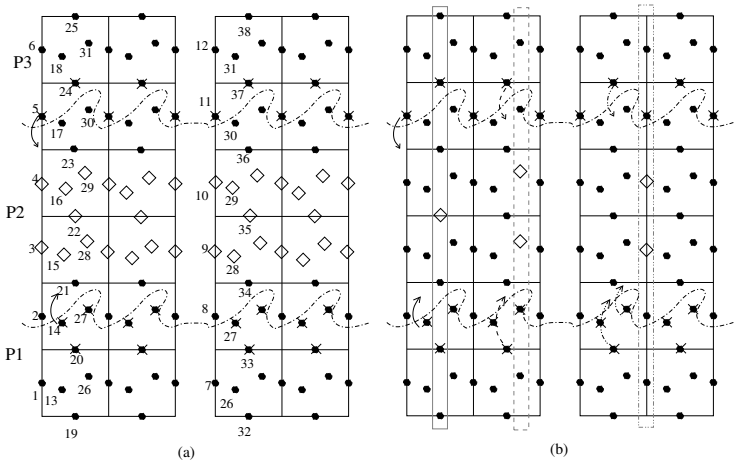

Fig. 3. Data distribution: $n_p = 3$, $n_1 = 2$, $n_2 = 2$, $n_3 = 6$. Communication scheme for matrix-vector multiplication (a), and for the solution of systems with lower triangular matrices (b)

3 Parallel implementation

Let us have $n_p \leq n_3 + 1$ processors denoted by P_1, P_2, ..., P_{n_p}. We assume that the domain is decomposed into $n_1 \times n_2 \times n_3$ non-conforming hexahedral elements. On fig. 3 is illustrated the partitioning of the domain into finite elements. A 2D projection on each "slice" of finite elements is used (see fig. 2 (c)). Since the "slices" have common sides, the nodes belonging to those sides appear twice (once on each 2D projection). Each vertical line of nodes corresponds to one of the diagonal blocks of the matrices A and B. The corresponding blocks have a varying size of n_3 or $n_3 + 1$. The total number of these blocks is $k = n_1(3n_2 + 1) + n_2$. To handle the system with the preconditioner $C_{MIC(0)}(B)w \equiv (X - L)\, X^{-1} \left(X - L^T\right) w = v$ one has to solve systems $\tilde{L}y \equiv (X - L)y = v$, $X^{-1}z = y$ and $\tilde{L}^T w = z$, L is the strictly lower triangular part of the matrix B. The triangular systems are solved using standard forward or backward recurrences. This can be done in k stages. Within stage i the block y_i is computed. Since the blocks $\tilde{L}_{ii}$, are diagonal, computations of each component of y_i are independent of each other and can be performed in parallel. That is the reason to distribute the entries of y_i among all the processors as shown on fig. 3. Therefore each processor P_j receives a strip of the computational domain. These strips are almost equally sized. Elements of all vectors and rows of all matrices which participate in PCG algorithm are distributed in the same manner. The processor P_i is responsible for the local computations on its strip. Let us see now how are performed the operations in the PCG algorithm, and what kind of communications are required.

Each PCG iteration consists of one solution of a system with the preconditioning matrix $C_{MIC(0)}(B)$, one matrix vector multiplication with the orig-

inal matrix A, two inner products, and three linked vector triads of the form $v := \alpha v + u$. The number of operations for one iteration of the PCG algorithm is $\mathcal{N}_{it}^{PCG} \approx 27N$, $N = 3n_1n_2n_3 + n_1n_2 + n_1n_3 + n_2n_3$.

For the triads, each processor calculates its part of the vector v and no communication is required. After computing the inner products corresponding to their parts of the vectors, the processors have to perform one global reduction operation to sum up the final result.

To obtain the components of the matrix-vector multiplication Av for which the processor P_i is responsible, it needs to receive from the processors P_{i-1} and P_{i+1} some components of the vector v. The number of these components is $4n_1n_2+n_1+n_2$. Because of the even distribution of nodes that lay on the splitting planes between the strips, half of that number is to be received by each of the processors P_{i-1} and P_{i+1}. On fig. 3 (a), with sign $\times$ are marked ellements to be transferred to P_2. While these communications are in progress the components of Av which do not depend on the components of v in the neighbouring processors can be computed. These components are marked with sign $\diamond$.

Let us go back to the solution of the preconditioner system (see (2)). The solution of a system with a diagonal matrix is trivial and does not require any communications. As we saw, the solution of the triangular systems can be done in k stages. On each stage, the part of the solution, corresponding to one vertical line of nodes, is computed. After each stage, the processors have to exchange some components in order the computations in the next stage to be performed. Three different patterns of transfers are required. They are illustrated with differently dashed lines and arrows. Transfer of one or two components between each pair of nodes in both directions is required per each vertical line. Again, computations for the inner components, marked with sign $\diamond$, can be overlapped with the communications.

Estimation of the parallel times is derived under the following assumptions: a) the execution of M arithmetic operations on one processor takes time $T = Mt_a$, where t_a is the average unit time to perform one arithmetic operation on a single processor, b) the time to transfer M data elements between two neighbouring processors can be approximated by $T^{comm} = t_s + Mt_c$, where t_s is the start-up time and t_c is the incremental time for each of the M elements to be transferred, and c) send and receive operations between each pair of neighbouring processors can be done in parallel. We get the following expressions for the communication times:

$$T^{comm}(C^{-1}v) \approx 6n_1n_2(t_s + 2t_c), \quad T^{comm}(Av) \approx t_s + 2n_1n_2t_c.$$

The above communications are completely local. The inner product needs one broadcasting and one gathering global communication but they do not contribute to the leading terms of the total parallel time. The parallel properties of the algorithm do not depend on the number of iterations, so it is enough to evaluate the parallel time per iteration, and use it in the speedup and efficiency analysis. As the computations are almost equally distributed among the processors, assuming there is no overlapping of the communications and computations one

can write for the total time on n_p processors:

$$T^{it}_{n_p} \approx \frac{27Nt_a}{n_p} + T^{comm}(C^{-1}v) + T^{comm}(Av)$$
$$= \frac{27(n_1n_2 + n_3(3n_1n_2 + n_1 + n_2))t_a}{n_p} + t_s + 2n_1n_2(3t_s + 7t_c)$$

What we can observe is that the communication time is practically independent of the number of processors. The situation changes if we have overlapping of computations and communications. One can expect in that case some reduction of the time $T^{it}_{n_p}$. However, with increasing of n_p this effect will weaken. The overlapping can notably reduce the influences of networks' latencies and slow speeds. This is true only if the amount of the computations overlapped is big enough. Of course, there is always an overhead for the communications related to the communication calls – buffering, addresses computations, etc. The speedup $S_{n_p} = T_1/T_{n_p}$ will grow with n_3 and it will grow even if n_1 and n_2 grow as n_3 up to the theoretical limit $S_{n_p} = n_p$. However, typically, for the real life parallel systems $t_s \gg t_c$ and $t_s \gg t_a$, and we could expect good speedups and efficiencies $E_{n_p} = S_{n_p}/n_p$ only when $n_3 \gg n_p\, t_s/t_a$.

4 Numerical Tests

The presented algorithm is coded in C++ using the MPI library. Two different reorderings of the calculations are performed to improve the overlapping of the computations and communications: 1) when computing Ax and solving the triangular systems, first are performed the computations which do not depend on the values stored in neighbouring processors. 2) some of the vector operations of the PCG algorithm are performed simultaneously with the solution of the preconditioner. It is observed that the later optimisation also improves the cache utilisation.

The experiments are performed on two parallel platforms. These platforms are referenced further as "A" and "B". Platform "A" is an "IBM SP Cluster 1600" made of 64 nodes p5-575 interconnected with a pair of connections to the Federation HPS (High Performance Switch). Each p5-575 node contains 8 SMP processors Power5 at 1.9GHz and 16GB of RAM. Platform "B" is a "Cray XD1" cabinet, fully populated with 72 2-way nodes, totally 144 AMD Opteron processors at 2.4GHz. Each node has 4GB of memory. The CPUs are interconnected with the Cray RaidArray network.

We consider the model Poisson equation in a unit cube with homogeneous Dirichlet boundary conditions assumed on the right side of the domain. The partitioning is uniform, let $n_1 = n_2 = n_3 = n$. The size of the discrete problem is $N = 3(n^3 + n^2)$. A relative stopping criterion $\frac{(C^{-1}r^i, r^i)}{(C^{-1}r^0, r^0)} < 10^{-9}$ is used in the PCG algorithm, where r^i stands for the residual at the i-th iteration step.

The mesh size parameter n, the total number of unknowns N, the iteration count n_{it} and the sequential times T_A and T_B on platforms "A" and "B" respectively are presented in Table 1. The experiment with $n = 255$ was not run

Table 1. Number of iterations and sequential times in seconds

n	N	n_{it}	T_A	T_B
31	92 256	22	1.2	0.7
63	762 048	31	10.7	7.4
127	6 193 536	44	105.6	78.4
255	49 939 200	63	1098.6	834.0

on platform "B" due to lack of enough physical memory available on a single node. The corresponding time in the table is obtained by a simple extrapolation. It is used later to estimate the speedups and the efficiencies of the tests on 4 and 8 processors. One can observe that: 1) the iteration count grows as $O(n^{1/2}) = O(N^{1/6})$; 2) the time per iteration is proportional to N.

In the Table 2. are presented the speedups and efficiencies of the performed parallel tests, n_p is the number of processors used. With $S^X_{n_p}$ and $E^X_{n_p}$ are denoted speedup and efficiency on platform "X" using n_p processors. One can see

Table 2. Parallel speedups and efficiencies

n	n_p	$S^A_{n_p}$	$S^B_{n_p}$	$E^A_{n_p}$	$E^B_{n_p}$
31	2	1.18	1.39	0.59	0.70
	4	1.00	1.61	0.25	0.40
	8	0.96	1.98	0.12	0.25
63	2	1.30	1.59	0.65	0.79
	4	1.83	2.00	0.44	0.50
	8	2.15	2.46	0.27	0.33
127	2	1.60	1.74	0.80	0.87
	4	2.30	2.70	0.57	0.67
	8	2.22	3.79	0.28	0.47
255	2	1.64	-	0.82	-
	4	2.63	2.91	0.65	0.72
	8	4.06	4.09	0.51	0.51

that for a given number of processors the speedup and efficiency grow with the problem size. Conversely for fixed n, the speedup and the efficiency decrease with the number of processors. For small ratios n/n_p they are still far from the theoretical upper bounds $S_{n_p} \leq n_p$ and $E_{n_p} \leq 1$. Unfortunately the ratio n/n_p is strongly bounded in real-life computations. When we increasing n the total memory requirements increase as n^3 and one has to choose n_p sufficiently large in order to fit the required for the computations vectors in the memory available on each node. As a result the ratio n/n_p decreases with the increase of n. But with the inevitable increase of the amount of RAM available on each node in

the future, the ratio n/n_p will improve. And so will increase the applicability of the proposed algorithm.

5 Conclusions

A new parallel MIC(0) preconditioner for 3d elliptic problems is proposed and studied. Estimates for the parallel times have been derived. The presented numerical tests are in a good agreement with the theoretical results. The future plans are addressed to improvement of the parallel efficiency based on a more appropriate modification B of the original stiffness matrix A [2]. Its block structure will be with larger diagonal blocks on the diagonal corresponding to mesh nodes of the entire plane. As a result, the number of communication steps in the solution of the preconditioner will decrease and more computations could be overlapped with the communications.

Acknowledgements

This work has been performed under the EC Project HPC-EUROPA RII3-CT-2003-506079. The author has also been supported by the EC INCO Grant BIS-21++ 016639/2005.

References

1. O. Axelsson, Iterative Solution Methods, *Cambridge University Press* (1995)
2. P. Arbenz, S. Margenov, Parallel MIC(0) preconditioning of 3D nonconforming FEM systems, Iterative Methods, Preconditioning and Numerical PDEs, Processings (2004), 12-15.
3. D.N. Arnold and F. Brezzi, Mixed and nonconforming finite element methods: implementation, postprocessing and error estimates, *RAIRO, Model. Math. Anal. Numer.*, **19** (1985), 7–32.
4. R. Blaheta, Displacement decomposition - incomplete factorization preconditioning techniques for linear elasticity problems, *Numer. Lin. Alg. Appl.*, **1** (1994), 107–126.
5. G. Bencheva, S. Margenov, Parallel incomplete factorization preconditioning of rotated bilinear FEM systems, *J. Comp. Appl. Mech.*, **4(2)** (2003), 105-117.
6. G. Bencheva, S. Margenov, J. Star, MPI Implementation of a PCG Solver for Nonconforming FEM Problems: Overlapping of Communications and Computations, *Technical Report 2006-023, Uppsala Univerty*
7. I. Gustafsson, Stability and rate of convergence of modified incomplete Cholesky factorization methods, *Report 79.02R. Dept. of Comp. Sci., Chalmers University of Technology*, Goteborg, Sweden, 1979
8. I. Gustafsson, G. Lindskog, On parallel solution of linear elasticity problems: Part I: Theory, *Numer. Lin. Alg. Appl.*, **5** (1998), 123–139.
9. R.D. Lazarov, S.D. Margenov, On a two-level parallel MIC(0) preconditioning of Crouzeuz-Raviart non-conforming FEM systems, *Springer Lecture Notes in Computer Science*, Vol. 2542 (2003), 192-201.
10. R. Rannacher, S. Turek, Simple nonconforming quadrilateral Stokes Element, *Numerical Methods for Partial Differential Equations*, 8(2) (1992), pp. 97–112.