ELSEVIER

Mathematics and Computers in Simulation 50 (1999) 247–254

MATHEMATICS AND COMPUTERS IN SIMULATION

www.elsevier.nl/locate/matcom

MPI parallel implementation of CBF preconditioning for 3D elasticity problems[1]

Ivan Lirkov*, Svetozar Margenov

Central Laboratory for Parallel Processing, Bulgarian Academy of Sciences, Acad.G.Bonchev Str., Bl.25A, 1113 Sofia, Bulgaria

Abstract

The numerical solution of 3D linear elasticity equations is considered. The problem is described by a coupled system of second-order elliptic partial differential equations. This system is discretized by trilinear brick finite elements. The PCG iterative method is used for solving the large-scale linear algebraic systems arising after the FEM discretization of the problem. Displacement decomposition technique is applied at the first step to construct a preconditioner using the decoupled block-diagonal part of the original matrix. Then circulant block-factorization is used for preconditioning of the obtained block-diagonal matrix.

New construction of a parallel algorithm for the discussed preconditioning method is proposed. The theoretical part of this study includes analysis of the execution time on various parallel architectures and asymptotic estimates of the parallel speedup and the parallel efficiency. The parallel performance estimates indicate that the proposed algorithm will be especially efficient on coarse-grain parallel systems, which is also confirmed by the numerical experiments. A portable MPI parallel code is developed. Numerical tests on three symmetric multiprocessor systems: SUN Enterprise 3000, SUN SPARCstation 10 and Origin 2000 are presented. The reported speedup and parallel efficiency illustrate well the features of the proposed method and its implementation.

Keywords: Parallel algorithms; PCG method; Preconditioner; Circulant matrix

1. Introduction

When a 3D elasticity system is discretized using the finite element method (FEM), the problem is reduced to a linear system $K\mathbf{x} = \mathbf{f}$. Here, the stiffness matrix K is large sparse and symmetric positive definite. The conjugate gradient (CG) type methods are recognized as the most cost-effective way to solve problems of this type [1]. To accelerate the iteration convergence a preconditioner M is combined with the CG algorithm. The theory of the preconditioned CG (PCG) method says that M is considered

* Corresponding author. E-mail: ivan@cantor.bas.bg

[1]This research has been supported by the Copernicus CP 940820 (HIPERGEOS) Project and by the Bulgarian NSF Grant I-701/97.

0378-4754/99/$20.00
PII: S0378-4754(99)00084-1

as a good preconditioner if it reduces significantly the condition number $\kappa(M^{-1}K)$, and at the same time, if the inverse matrix vector product $M^{-1}v$ can be efficiently computed for a given vector v. A third important aspect should be added to the above two, namely, the requirement for efficient implementation of the PCG algorithm on recent parallel computer systems.

A new high performance and parallel efficient PCG algorithm for 3D linear elasticity problems, implementing displacement decomposition circulant block factorization (DD CBF), is proposed and studied in this paper. The message passing interface (MPI) [9,10] standard is used to develop a portable parallel code. The paper is organized as follows. Section 2 is devoted to a brief description of the elasticity equations as well as of the benchmark problem under consideration. In Section 3, we focus on the construction of the DD CBF preconditioner. The parallel complexity of the algorithm is analyzed in the Section 4. Parallel numerical tests on SGI Origin 2000, SUN SPARCstation 10 and SUN Enterprise 3000 can be found in Section 5. Concluding remarks and some outlook about the parallel performance of the developed MPI FEM code are given in Section 6.

2. Elasticity equations and the benchmark problem

Let $\mathscr{B}$ be an elastic body occupying a bounded polyhedral domain $\Omega \subset \mathbb{R}^3$, impose Dirichlet/Neumann boundary conditions on $\Gamma_D \cup \Gamma_N \equiv \partial\Omega$. We denote the displacement vector by $\mathbf{u} = [u_1, u_2, u_3]^T$, the stress tensor by $\underline{\sigma}(\mathbf{u}) = [\sigma_{ij}(\mathbf{u})]$ and the strain tensor by $\underline{\epsilon}(\mathbf{u}) = [\epsilon_{ij}(\mathbf{u})]$. Without any restrictions we could assume that the Dirichlet boundary conditions are homogeneous. Then the following variational formulation of the problem holds: find $\mathbf{u} \in (H_0^1(\Omega))^3 (H_0^1(\Omega) = \left\{ v \in H^1(\Omega) : v_{|_{\Gamma_D}} = 0 \right\}$ is the standard Sobolev space), such that:

$$a_\Omega(\mathbf{u}, \mathbf{v}) = F(\mathbf{v}), \quad \forall \mathbf{v} \in (H_0^1(\Omega))^3,$$

where

$$a_\Omega(\mathbf{u}, \mathbf{v}) \equiv \int_\Omega \left[\lambda \operatorname{div} \mathbf{u} \operatorname{div} \mathbf{v} + \mu \sum_{i,j=1}^{3} \epsilon_{ij}(\mathbf{u}) \epsilon_{ij}(\mathbf{v}) \right] \mathrm{d}x.$$

The Lame coefficients λ and μ depend on the Young's modulus E and on the Poisson ratio ν. The bilinear form $a_\Omega(\mathbf{u},\mathbf{v})$ is symmetric and coercive. Then the related discrete variational problem is: find $u_h \in V^h(\Omega) \subset (H_0^1(\Omega))^3$, such that:

$$a_\Omega(u_h, v_h) = F(v_h), \quad \forall v_h \in V^h(\Omega).$$

In this study $V^h(\Omega)$ is the FEM space of piecewise trilinear functions. The latter problem is equivalent to the linear system:

$$K\mathbf{x} = \mathbf{f}, \tag{1}$$

where $\mathbf{x} = [x_i]^T$ is the vector of the nodal unknowns x_i, $i = 1, 2, \ldots, N$. The PCG method is used to solve Eq. (1).

In what follows, we restrict our considerations to the case $\Omega = [0, x_1^{\max}] \times [0, x_2^{\max}] \times [0, x_3^{\max}]$, where the boundary conditions on each of the sides of Ω are of a fixed type. The benchmark problem from [4]

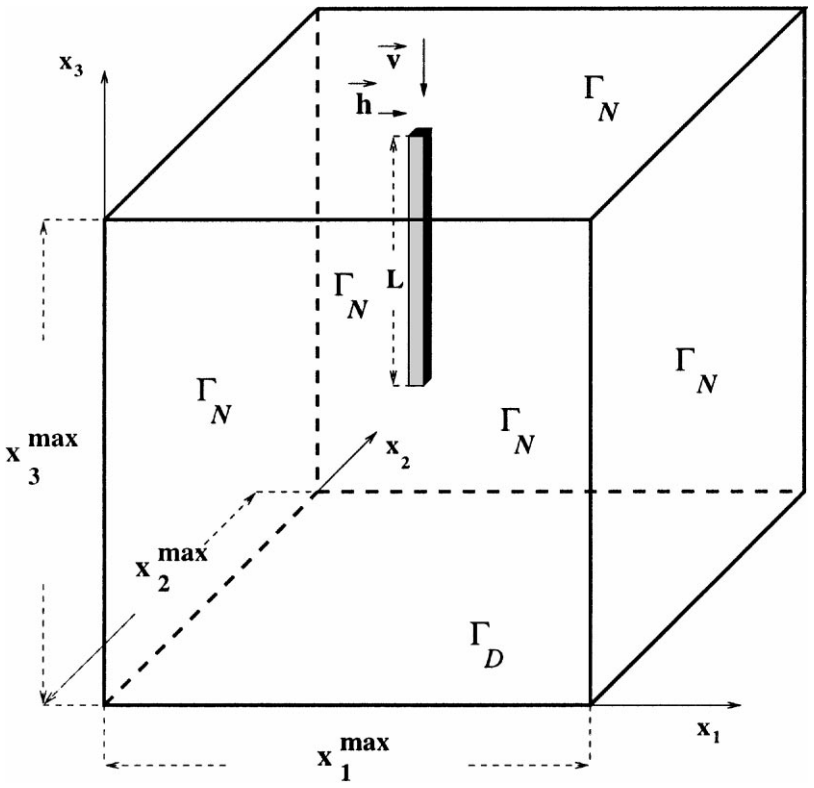

Fig. 1. Benchmark problem.

is used in the reported numerical tests. This benchmark represents the model of a single pile in a homogeneous sandy clay soil layer (see Fig. 1). An uniform grid is used with n_1, n_2 and n_3 grid points along the coordinate directions. Then the stiffness matrix K can be written in a 3×3 block form where the blocks K_{ij} are sparse block-tridiagonal matrices of a size $n_1n_2n_3 = N/3$.

3. DD CBF preconditioning

First, let us recall that a $m\times m$ circulant matrix C has the form $(C_{k,j}) = (c_{(j-k) \bmod m})$. Each circulant matrix can be factorized as $C = F\Lambda F^*$, where Λ is a diagonal matrix of the eigenvalues of C, and F is the Fourier matrix $F = (1/\sqrt{m})\{e^{2\pi(jk/m)i}\}_{0\le j,k\le m-1}$. Here i stands for the imaginary unit.

3.1. A displacement decomposition-based preconditioner

There are a lot of works dealing with preconditioning iterative solution methods for the FEM elasticity systems. Axelsson and Gustafson [2] constructed their preconditioners based on the point-ILU factorization of the displacement decoupled block-diagonal part of the original matrix. This approach is known as displacement decomposition (see, e.g., [3]).

To define the displacement decomposition preconditioner M_{DD} of the matrix K, we introduce the auxiliary Laplace equation $-u_{x_1x_1} - u_{x_2x_2} - u_{x_3x_3} = f$, with boundary conditions corresponding to the considered coupled elasticity problem. Let us primarily assume, that this Laplace equation is discretized by the same brick finite elements as the original problem, and let K_0 be the obtained stiffness matrix.

The following Korn's inequality gives the theoretical background of the displacement decomposition methods [2]:

$$\kappa(M_{\mathrm{DD}}^{-1}K) = \mathcal{O}\left(\frac{1}{1-2\nu_{\max}}\right), \tag{2}$$

where $\nu_{\max} = \max_{\Omega}\nu$ and

$$M_{\mathrm{DD}} = \mathrm{diag}(K_0, K_0, K_0). \tag{3}$$

The next step in our construction is to substitute in Eq. (3), K_0 by A_0, where A_0 stands for the Laplace stiffness matrix corresponding to linear finite elements or, which is equivalent in the case under consideration, to a seven point finite difference stencil. This step is motivated by the more simple/sparse structure of A_0 as well as by the spectral equivalence

$$\kappa(A_0^{-1}K_0) = \mathcal{O}(1). \tag{4}$$

3.2. Circulant block factorization

The CBF preconditioning technique (see [7]) incorporates the circulant approximation into the framework of the *LU* block factorization. It was recently introduced and analyzed in [5] for the model Dirichlet boundary value problem:

$$-(a(x_1,x_2,x_3)u_{x_1})_{x_1} - (b(x_1,x_2,x_3)u_{x_2})_{x_2} - (c(x_1,x_2,x_3)u_{x_3})_{x_3} = f(x_1,x_2,x_3)$$

in $\Omega = [0, x_1^{\max}] \times [0, x_2^{\max}] \times [0, x_3^{\max}]$. Let us assume (as in the previous section) that Ω is discretized by a uniform grid with n_1, n_2 and n_3 grid points along the coordinate directions, and that a standard (for such a problem) seven-point finite difference (FEM) approximation is used. The related stiffness matrix $A^{(d)}$ can be written in the block-form $A^{(d)} = \mathrm{tridiag}(-A_{i,i-1}^{(d)}, A_{i,i}^{(d)}, -A_{i,i+1}^{(d)})$, $\quad i = 1, 2, \ldots, n_1$, where $A_{i,i}^{(d)}$ is a block-tridiagonal matrix corresponding to the ith x_1-plane, and the off-diagonal blocks are diagonal matrices. Now, CBF preconditioner M_{CBF} is defined as follows:

$$M_{\mathrm{CBF}} = \mathrm{tridiag}(-C_{i,i-1}, C_{i,i}, -C_{i,i+1}), \quad i = 1, 2, \ldots, n_1. \tag{5}$$

Here $C_{i,j} = BC(A_{i,j}^{(d)})$ is a block-circulant approximation of the corresponding block $A_{i,j}^{(d)}$. The relative condition number of the CBF preconditioner for the model (Laplace) 3D problem is studied in [5] for $n_1 = n_2 = n_3 = n$, where the following estimate is derived:

$$\kappa(M_{\mathrm{CBF}}^{-1}A_0) \leq 4n. \tag{6}$$

Now, let us denote by M_0 the CBF preconditioner for A_0, the matrix introduced in the previous subsection. At the last step of our construction we substitute in Eq. (3), K_0 by M_0, and get the DD CBF

preconditioner defined by:

$$M_{\mathrm{DD\ CBF}} = \mathrm{diag}(M_0, M_0, M_0).$$

The estimate of the condition number of the preconditioned matrix:

$$\kappa(M_{\mathrm{DD\ CBF}}^{-1} K) = \mathcal{O}\left(\frac{n_{\max}}{1 - 2\nu_{\max}}\right) \tag{7}$$

follows straightforwardly from Eqs. (2),(4) and (6).

Remark 1. *We have observed in the performed numerical tests that a diagonal scaling of K improves the convergence rate of the iterative method in the case of problems with jumping coefficients.*

4. Analysis of the parallel complexity

We assume that the computations and communications are not overlapped and therefore, the execution time of the parallel implementation is the sum of the computation time and the communication time. We shall use in our analysis standard models for the arithmetic and communication times [8]. First, assuming no arithmetic vectorization, the execution of M arithmetic operations on one processor takes time $T_a = M \times t_a$, where t_a is the average unit time to perform one arithmetic operation on one processor. Second, the local communication time to transfer M data from one processor to its neighbor is approximated by $T_{\mathrm{local}} = t_s + M \times t_c$, where t_s is the start-up time and t_c is the incremental time necessary for each of all M words to be sent. Finally, we consider the following two quantities, which are of special interest for various computer architectures, namely:

- $b(p)$: broadcasting a number from one processor to all others, where there are p processors in the computer system;
- $g(M,p)$: gathering p data packets, each packet with M/p words, in one processor from all others.

We discuss three distributed memory architecture models: ring, square grid and hypercube, and shared memory model. The shared memory architecture is interpreted assuming that each two processors in the system can be considered as neighbors. The corresponding broadcasting and gathering times are given in the table below:

	$b(p)$	$g(M,p)$
ring	$(t_s + t_c)(p/2)$	$pt_s + Mt_c$
grid	$(t_s + t_c)\sqrt{p}$	$2\sqrt{p}t_s + M(1 + (1/\sqrt{p}))t_c$
hypercube	$(t_s + t_c)\log p$	$\log p t_s + M(1 - (1/p))t_c$
shared memory	$\lceil \log p \rceil (t_s + t_c)$	$(p-1)(t_s + (M/p)t_c)$

We consider broadcasting from the processor in the center of the square grid to obtain an optimal time. This is not essential for the algorithm itself.

We can now estimate the total execution time T_{PCG} for one PCG iteration with the described DD CBF preconditioner. Each iteration consists of one matrix vector multiplication with the matrix K, solution

of one system of equations with the preconditioner, two inner products and three linked triads (a vector updated by a vector multiplied by a scalar). Consequently:

$$T_{\mathrm{PCG}}(p) = T_{\mathrm{mult}} + T_{\mathrm{prec}} + 2T_{\mathrm{inn-prod}} + 3T_{\mathrm{triads}}.$$

.For simplicity we will assume that the mesh sizes are equal being exact power of two, i.e., $n_1 = n_2 = n_3 = n = 2^l$. Then:

$$T_{\mathrm{mult}} = 243\frac{n^3}{p}t_{\mathrm{a}} + 4(t_{\mathrm{s}} + 3n^2 t_{\mathrm{c}}), \qquad T_{\mathrm{inn-prod}} = 6\frac{n^3}{p}t_{\mathrm{a}} + g(p,p) + b(p),$$

$$T_{\mathrm{triads}} = 6\frac{n^3}{p}t_{\mathrm{a}}, \qquad T_{\mathrm{prec}} = 6\frac{n^2}{p}T_{\mathrm{FFT}}(n) + 36\frac{n^3}{p}t_{\mathrm{a}} + 2g\left(3\frac{n^3}{p}, p\right).$$

Here $T_{\mathrm{FFT}}(n)$ is the time for execution of FFT on a given n-vector on one processor. If we use 2-radix algorithm, then $T_{\mathrm{FFT}}(n) = 5n\log n t_{\mathrm{a}}$. Combining the above, we obtain the following estimates:

$$T_{\mathrm{PCG}}(p) = 3(103 + 10\log n)\frac{n^3}{p}t_{\mathrm{a}} + 4\left(t_{\mathrm{s}} + 3n^2 t_{\mathrm{c}}\right) + 2g\left(3\frac{n^3}{p}, p\right) + 2g(p,p) + 2b(p).$$

Let us consider in some more details the parallel execution time for the shared memory model. We have:

$$T_{\mathrm{PCG}}(p) = 2(p + \lceil\log p\rceil + 1)t_{\mathrm{s}} + 2\left[3(p-1)\frac{n^3}{p^2} + 6n^2 + \lceil\log p\rceil\right]t_{\mathrm{c}} + 3(103 + 10\log n)\frac{n^3}{p}t_{\mathrm{a}}.$$

Taking the leading terms of the above expression we get the approximation:

$$T_{\mathrm{PCG}}(p) \approx 2pt_{\mathrm{s}} + 6\left(1 - \frac{1}{p}\right)\frac{n^3}{p}t_{\mathrm{c}} + 3(103 + 10\log n)\frac{n^3}{p}t_{\mathrm{a}}. \tag{8}$$

To analyze the relative speedup S_{p} and the relative efficiency E_{p}, where $S_{\mathrm{p}} = (T(1)/T(p)) \leq p$ and $E_p = S_p/p \leq 1$, we apply Eq. (8) and obtain:

$$S_{\mathrm{p}} \approx \frac{3(103 + 10\log n)}{2(p^2/n^3)(t_{\mathrm{s}}/t_{\mathrm{a}}) + 6(1 - (1/p))(t_{\mathrm{c}}/t_{\mathrm{a}}) + 3(103 + 10\log n)}p. \tag{9}$$

Obviously, for the DD CBF preconditioner, $\lim_{n\to\infty} S_{\mathrm{p}} = p$ and $\lim_{n\to\infty} E_{\mathrm{p}} = 1$, i.e., the algorithm is asymptotically optimal. More precisely, if $\log n \gg (p^2/n^3)(t_{\mathrm{s}}/t_{\mathrm{a}}) + (t_{\mathrm{c}}/t_{\mathrm{a}})$, then E_{p} is close to 1. Unfortunately, the start-up time t_{s} is usually much larger than t_{a}, and for relatively small n the first term of the denominator in Eq. (9) is dominating. In such case the efficiency could be much smaller than 1.

5. Parallel tests of the DD CBF preconditioning FEM code

The developed parallel MPI C code was tested on three parallel machines. We report here the elapsed time T_{p} on p processors, the speedup $S_{\mathrm{p}} = T_1/T_{\mathrm{p}}$, and the parallel efficiency $E_{\mathrm{p}} = S_{\mathrm{p}}/p$. The benchmark problem was already described in Section 2.

Table 1
Parallel time (in seconds), speedup and parallel efficiency

	n_1	n_2	n_3	p	T_p	S_p	E_p
Origin 2000							
All	32	32	32	1	213.326		
				2	116.553	1.830	0.915
Per iteration				1	1.817		
				2	0.993	1.830	0.915
SUN SPARCstation 10							
All	32	32	32	1	1120.900		
				2	600.175	1.868	0.934
				4	351.283	3.191	0.798
Per iteration				1	9.263		
				2	5.002	0.798	0.926
				4	2.895	3.199	0.800
SUN Enterprise 3000							
	32	32	32	1	412.370		
				2	202.645	2.035	1.017
				4	102.005	4.043	1.011
				8	55.3532	7.450	0.931
	48	48	48	1	2516.900		
				2	1238.110	2.033	1.016
				3	834.714	3.015	1.005
				4	648.159	3.883	0.971
				6	420.440	5.986	0.998
				8	317.082	7.938	0.992
	64	64	32	1	2322.830		
				2	1142.210	2.034	1.017
				4	579.867	4.006	1.001
				8	298.219	7.789	0.974

We show in Table 1 results obtained on SGI Origin 2000 machine with two 188 MHz processors and 128 Mb main memory. From the machines we have tested our code on, this one has the fastest processors. The idea of this set of numerical data is to see what is the behavior of the algorithm on a very coarse-grain system where the influence of the communications is minimal. The parallel efficiency is above 90% which confirms our general expectations.

The results in Table 1 are produced on a SUN SPARCstation 10 with four 130 MHz processors and 192 Mb main memory. One can see again the well expressed high degree of parallelism of the algorithm. The observed better efficiency for $p = 2$ (compare the results on Origin 2000 and SUN SPARCstation 10) is due to the relatively slower processors of this SUN configuration.

In Table 1 (SUN Enterprise 3000), we present results of experiments executed on SUN Ultra-Enterprise Symmetric Multiprocessor with eight 167 MHz processors and 1 Gb main memory. The size of the main memory of this machine allows to get results for finer meshes. As expected, the parallel efficiency increases with the size of the discrete problems.

Remark 2. *There exist at least two more reasons for the high efficiency reported above: (a) the network parameters start-up time and time for transferring of single word are relatively small for the*

multiprocessor machines; (b) there is also some overlapping between the computations and the communications in the algorithm.

6. Conclusions

The DD CBF preconditioning algorithm, presented in this paper, possesses strongly expressed parallel structure with well-balanced local communications. The performed numerical tests clearly demonstrate the high level of parallel efficiency of the developed parallel code obtained on different models of symmetric multiprocessors with up to eight processors. The speedup and the parallel efficiency increase with the size of the discrete problem. As it was shown above, the achieved parallel efficiency is above 80% for the considered real-life large-scale discrete problems.

The use of the MPI standard is a key component in the development of concurrent computing environment in which applications and tools can be transparently ported between different computers. The reported new MPI code is portable on both distributed and shared memory systems, as well as on clusters of workstations.

The DD CBF code provides new effective tools for computer simulation of real-life engineering problems with 10^5–10^6 unknowns in realistic time on a class of coarse-grain parallel computer systems with currently increasing cost-efficiency.

Remark 3. *The reported code enables possibility to solve very large-scale problems on distributed memory parallel computers. The decomposition strategy allows to treat efficiently such problems, where the size is only limited by the total sum of the distributed memory. This has been already confirmed (see [6]) by the performed tests with our CBF code for 2D elliptic problems.*

References

[1] O. Axelsson, Iterative Solution Methods, Cambridge University Press, Cambridge, 1994.
[2] O. Axelsson, I. Gustafsson, Iterative methods for the solution of the Navier equations of elasticity, Comp. Meth. Appl. Mech. Eng. 15 (1978) 241–258.
[3] R. Blaheta, Displacement decomposition incomplete factorization preconditioning techniques for linear elasticity problems, Num. Lin. Alg. Appl. 1 (1994) 107–128.
[4] A. Georgiev, A. Baltov, S. Margenov, Hipergeos Benchmark Problems Related to Bridge Engineering Applications, PR HG CP 94-0820-MOST-4.
[5] I. Lirkov, S. Margenov, Conjugate gradient method with circulant block-factorization preconditioners for 3D elliptic problems, in: Proceedings of the Workshop No. 3 Copernicus 94-0820 'HiPerGeoS' project meeting, Rožnov pod Radhoštěm 1996.
[6] I. Lirkov, S. Margenov, M. Paprzycki, Benchmarking performance of parallel computers using a 2D elliptic solver, in: O. Iliev, M. Kaschiev, Bl. Sendov, P. Vassilevski (Eds.), Recent Advances in Numerical Methods and Applications, World Scientific, Singapore, (1999) 464–472.
[7] I. Lirkov, S. Margenov, P. Vassilevski, Circulant block-factorization preconditioners for elliptic problems, Computing 53(1) (1994) 59–74.
[8] Y. Saad, M.H. Schultz, Data communication in parallel architectures, Parallel Comput. 11 (1989) 131–150.
[9] M. Snir, St. Otto, St. Huss-Lederman, D. Walker, J. Dongara, MPI: The Complete Reference Scientific and Engineering Computation Series, The MIT Press, Cambridge, MA, 1997, Second printing.
[10] D. Walker, J. Dongara, MPI: a standard message passing interface, Supercomputer 63 (1996) 56–68.