PROJECT REPORT HG CP 94–00820–MOST–6.2

Tests of the parallel FEM codes on multiprocessor computers. Final requirements for parallel performance of FEM code

Krassimir Georgiev Ivan Lirkov Svetozar Margenov
BAS, Sofia, Bulgaria

Abstract

This report follows the research plan of the HIPERGEOS Project COP–94 00820. Tests of the developed in Center of Informatics and Computer Technology MPI parallel FEM codes on a different multiprocessor systems and distributed memory computers for 3D simulation of pile systems are presented. The considered benchmark is derived from the real-life engineering practice of MOST XXI Ltd (Bulgaria).

The problem is described by a coupled system of elliptic partial differential equations of second order, with strongly varying coefficients. This system is discretized by trilinear parallelepipedal finite elements.

The Preconditioned Conjugate Gradient (PCG) iterative solution is used for solving of large-scale linear systems arising from the Finite Element Method (FEM) discretization of the 3D elasticity equations. Two different preconditioning techniques are used. First approach consist of applying the Domain Decomposition method as an outer iteration solver where the Block-Size Reduction Block-Incomplete LU (BSR BILU) PCG algorithm is used as an *inner* iterative procedure. The second approach is based on the Displacement Decomposition where a Circulant Block-Factorization preconditioner is used for solving of the decoupled block-diagonal part of the original coefficient matrix. Both methods are highly parallelizable.

The numerical experiments reported bellow are obtained using three symmetric multiprocessor systems: SUN Enterprise 3000 (a workstation with 8×167 MHz Ultra SPARC processors), SUN SPARCstation 20 (a workstation with 4×100 MHz processors) and Origin 2000 (a workstation with 2×188 MHz SGI processors) as well as the parallel computer with distributed memory IBM SP Power 2 (with 32×135 MHz RS/6000 processors). A set of tables and figures illustrate the performed parallel numerical tests. They well demonstrate the parallel properties of the proposed methods and algorithms as well as the robustness of the developed codes.

1 Introduction

We are concerned with the numerical solution of 3D elasticity systems. After discretization, such problems are reduced to the solution of linear systems of the form $K\mathbf{x} = \mathbf{F}$ where K is sparse symmetric positive definite matrix. These linear systems are solved by the Preconditioned Conjugate Gradient method.

Portable parallel finite element codes for 3D linear elasticity problems implementing respectively overlapping domain decomposition and circulant block factorization preconditioners are developed. The Message Passing Interface (MPI) standard both for shared and distributed memory parallel computers is used. The Project Report is organized as follows. Section 2 is devoted to a description of the elasticity equations and the benchmark problem under consideration. In section 3 we focus on the overlapping domain decomposition approach and on the developed parallel solver. The last part of this section contains numerical tests on two models of SUN symmetric multiprocessor systems and on the distributed memory parallel computer IBM SP. A description of the displacement decomposition circulant block factorization, the parallel code using this preconditioning technic and numerical tests on SGI Origin 2000, SUN SPARCstation 20 and SUN Enterprise 3000 can be found in Section 4. The conclusions, requirements and some outlook for the parallel performance of the developed parallel MPI FEM codes are given in the final Section 5.

2 Elasticity equations and the benchmark problem

Let $\mathcal{B}$ be an elastic body occupying bounded polyhedral domain $\Omega \subset I\!R^3$, impose Dirichlet/Neumann boundary conditions on $\Gamma \equiv \partial\Omega$. We denote by $\mathbf{u} = [u_1, u_2, u_3]^T$ the displacement vector, by $\mathbf{F} = [f_1, f_2, f_3]^T$ the vector of the body forces, by $\underline{\sigma} = (\sigma_{ij})(\mathbf{u})$ the stress tensor and by $\underline{\epsilon}(\mathbf{u}) = (\epsilon_{ij}(\mathbf{u}))$ the strain tensor (the deformation associated with $\mathbf{u}$).

The variational formulation in displacements of this problem is as follows:

Find: $\mathbf{u} \in (H_0^1(\Omega))^3$, such that

$$a_\Omega(\mathbf{u},\mathbf{v}) \equiv \int_\Omega \left[\lambda\ div\mathbf{u}\ div\mathbf{v}\ +\ \mu \sum_{i,j=1}^{3} \epsilon_{ij}(\mathbf{u})\ \epsilon_{ij}(\mathbf{v}) \right] dx = F(\mathbf{v}), \qquad \forall\, \mathbf{v} \in (H_0^1(\Omega))^3,$$

where $H_0^1(\Omega)$ is the standard Sobolev space and

$$F(\mathbf{v}) = \sum_{i=1}^{3} \int_\Omega f_i v_i dx.$$

The Lame coefficients λ and μ depend on the modulus of elasticity (Young's modulus) - E and on the contraction ratio (Poisson ratio) - ν.

The second order bilinear form $a_\Omega(\mathbf{u},\mathbf{v})$ is symmetric and coercive.

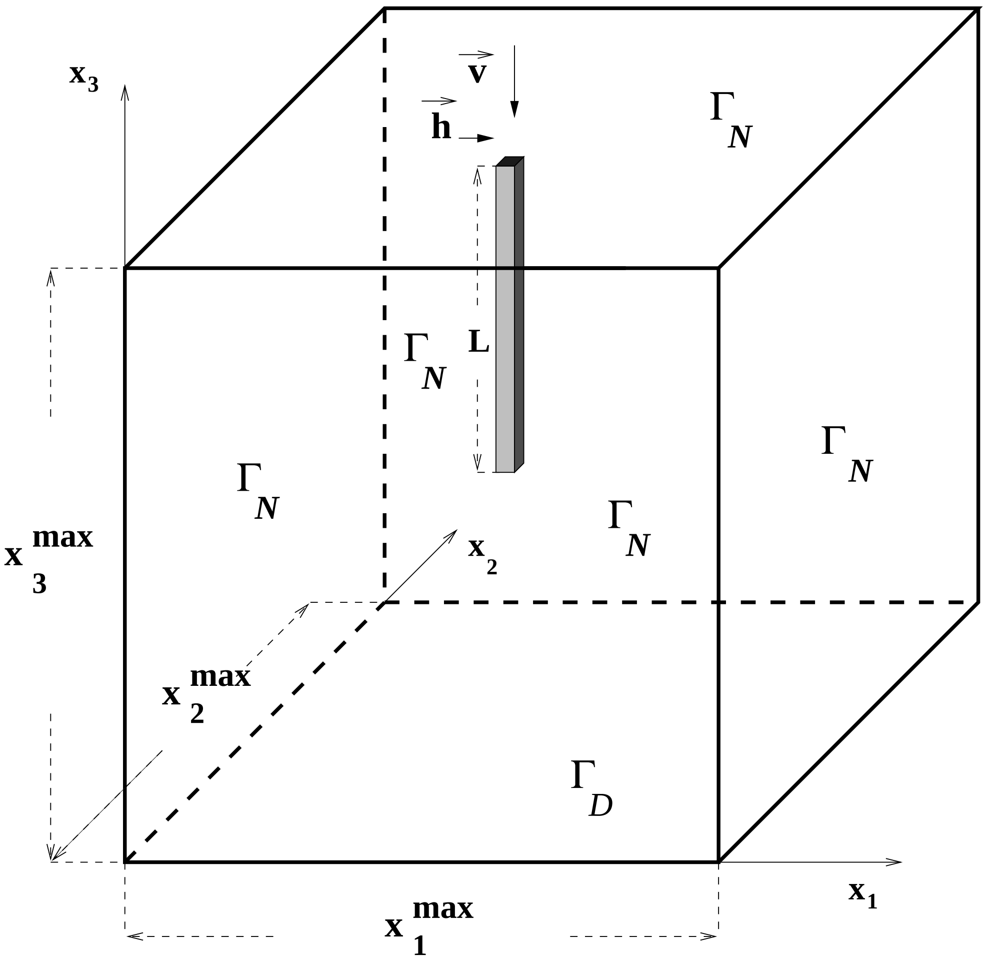

Figure 1: Benchmark 1.

Benchmark 1 as formulated by MOST XXI (see [8]) is used in the presented in this report tests. Single pile in a homogeneous sandy clay soil layer is considered. The computational domain is the parallelepiped (see Fig. 1)

$$\Omega \,=\, [0, x_1^{max}] \times [0, x_2^{max}] \times [0, x_3^{max}].$$

The boundary conditions are as follows:

- Homogeneous Dirichlet boundary conditions on the *bottom side* of the parallelepiped $x_3 = 0$;
- Homogeneous Neumann boundary conditions on the *vertical sides* and on the *top side* of the parallelepiped

  $$x_1 = 0, \quad x_1 = x_1^{max}, \quad x_2 = 0, \quad x_2 = x_2^{max}, \quad x_3 = x_3^{max}$$

  i.e the stresses are equal to zero, excluding the top cross section of the pile, where the load from the upper construction is transfered;

- Non-homogeneous Neumann boundary conditions on the *top cross section* of the pile.

The discrete variational problem is:

Find: $u_h \in V^h(\Omega) \subset (H_0^1(\Omega))^3$, such that

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

where $V^h(\Omega)$ is the FEM space of piecewise trilinear functions. After the discretization we get the linear system

$$K\,\mathbf{x} \;=\; \mathbf{F}, \tag{1}$$

where $\mathbf{x} = [x_i]^T$ is the vector of unknowns — x_i, $i = 1, 2, \ldots, N$, $\mathbf{F}$ — the right-hand side is the vector of components $F_i = \mathbf{F}(\Phi_i)$, and K is the stiffness matrix.

To solve the linear system of equations (1) the preconditioned conjugate gradient method is used [1].

<u>PCG — method:</u>

- $M\mathbf{U}_0 \;=\; \mathbf{F}$ – *initial iterate*;
- $\mathbf{r}_0 \;=\; \mathbf{F} \;-\; K\mathbf{U}_0$ — *initial defect*;

for $\quad k = 0, 1, \ldots \quad$ *until convergence*

- $M\mathbf{z}_k \;=\; \mathbf{r}_k$;
- $\beta_k \;=\; \dfrac{(\mathbf{r}_k, \mathbf{z}_k)}{(\mathbf{r}_{k-1}, \mathbf{z}_{k-1})}, \qquad \beta_0 \;=\; 0$;
- $\mathbf{p}_k \;=\; \mathbf{z}_k \;+\; \beta_k \mathbf{p}_{k-1}$ — *next search direction*;
- $\alpha_k \;=\; \dfrac{(\mathbf{r}_k, \mathbf{z}_k)}{(K\mathbf{p}_k, \mathbf{p}_k)}$;
- $\mathbf{U}_{k+1} \;=\; \mathbf{U}_k \;+\; \alpha_k \mathbf{p}_k$ — *next iterate*;;
- $\mathbf{r}_{k+1} \;=\; \mathbf{r}_k \;-\; \alpha_k K \mathbf{p}_k$ — *next quasi-residual*;

3 Domain decomposition preconditioning

In the developed parallel code the Schwartz algorithm based on overlapping sub-domains is used [4].

The triangulation of the computational domain Ω is done in the following way. The domain is first divided into non-overlapping substructures $\Omega_i, i = 1, 2, ..., P$, where P is equal to 1, 2, 4 or 8 (see the case $P = 4$ on Fig. 2). All the substructures Ω_i are further divided into parallelepipedal elements with size $h_1 \times h_2 \times h_3$.

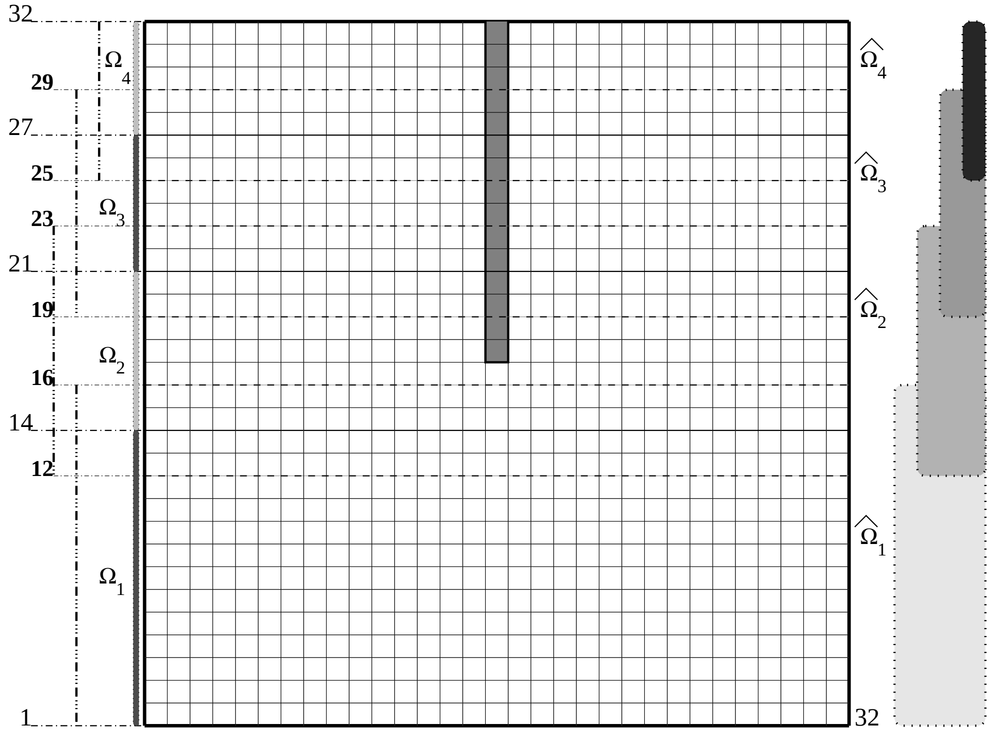

Figure 2: The four sub-domain case. Cross section $x_2 = \frac{x_2^{max}}{2}$. The overlap is $2h_3$.

We extend each sub-domain Ω_i to a larger region $\widehat{\Omega}_i$ with an overlap of $2h_3$ (see Fig. 2).

We also assume that $\partial\widehat{\Omega}_i$ does not cut through any element. We make the same construction for the substructures that are near the boundary of the domain Ω except that we cut off the part of $\widehat{\Omega}_i$ that is outside of Ω.

Let $\widehat{K_i}$ be a local stiffness matrix according to the sub-domain $\widehat{\Omega}_i$. We define restriction maps R_i and extension maps R_i^T, as follows. Let the global stiffness matrix K is of order $N = 3n_1n_2n_3$, where n_1, n_2 and n_3 are the numbers of the nodes in x_1, x_2 and x_3 directions. Let us denote with $\widehat{N_i}$ the number of the unknowns corresponding to the interior with respect to $\widehat{\Omega}_i$, nodes. For each sub-domain $\widehat{\Omega}_i$ the restriction matrix R_i is of size $N \times \widehat{N_i}$ with entries 1's and 0's , taking into account the indexes of the nodes belonging to $\widehat{\Omega}_i$. This matrix restricts a vector $\mathbf{x}$ of length N to vector $R_i\mathbf{x}$ of length $\widehat{N_i}$. Thus, the local stiffness matrix $\widehat{K_i}$ corresponding to the sub-domain $\widehat{\Omega}_i$ is

$$\widehat{K_i} \;=\; R_i\, K\, R_i^T.$$

Let us note that in the case of many sub-domains, the additive Schwartz algorithm is a generalization of the well known two-sub-domains case with a preconditioner

$$M_{add}^{-1} = \sum_{i=1}^{P} R_i^T \;\widehat{K_i}^{-1}\; R_i.$$

Indeed, if we have computed the iterate (k), we get the $(k+1)$st by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \left(\sum_{i=1}^{P} R_i^T \ \widehat{K_i}^{-1} \ R_i \right) \left(\mathbf{F} - K \ \mathbf{x}^{(k)} \right)$$

As $R_i^T \ \widehat{K_i}^{-1} \ R_i \ \mathbf{y}$ can be computed in parallel for the different sub-domains, this leads to a coarse grain parallelism. Taking into account the definition of the restriction matrix R_i (and the extension or interpolation matrix R_i^T) we may conclude that it is not necessarily to store these matrices.

So defined preconditioner M_{add} can be viewed as a straightforward generalization of the standard block-Jacobi preconditioner. The convergence rate of this preconditioner deteriorates when the number of sub-domains increases.

3.1 The Parallel Solver

In the Domain Decomposition preconditioning code the system $M_{add}\mathbf{z}_k = \mathbf{r}_k$ in the Preconditioning Conjugate Gradient algorithm is solved as follows.

1. $\hat{\mathbf{r}}_k^i$ vector of a length $\widehat{N_i}$; entries are equal to those of $\mathbf{r}_k$ for the components corresponding to the mesh-points in $\widehat{\Omega_i}$ and 0 elsewhere;
2. $\hat{\mathbf{z}}_k^i \ : \ \widehat{K_i}\hat{\mathbf{z}}_k^i = \hat{\mathbf{r}}_k^i$;
3. $\mathbf{z}_k^i$ vector of a length N; an extension of $\hat{\mathbf{z}}_k^i$ with 0 entries outside $\hat{\Omega}_i$;
4. $\mathbf{z}_k = \sum_{i=1}^{P} \mathbf{z}_k^i$

All four steps of this algorithm are performed in parallel. The coefficient matrices of the systems of linear equations

$$\widehat{K_i}\hat{\mathbf{z}}_k^i = \hat{\mathbf{r}}_k^i \tag{2}$$

are symmetric positive definite block matrices and the preconditioned conjugate gradient method with Block-Size Reduction Block-ILU preconditioner proposed by Chan and Vassilevski [5] (see also: [3] and [6]) is used to solve the problems in each sub-domain.

When parallel computers (with shared or distributed memory) are used for an implementation of domain decomposition algorithms, as a rule the different sub-domains are mapped to individual processors. The problem which has to be solved is the communications between the different processors. The Message Passing Interface (MPI) standard [7, 10, 13] is used.

At this Fortran version of the code realizing our algorithm we have used:

- **blocking send** (MPI_SEND) and **blocking and unblocking receive** (MPI_RECV and MPI_IRECV) operations in a **standard mode** from the *Point-to-Point communications*;

- MPI_BARRIER for **barrier synchronization**, MPI_BCAST for **broadcasting** data, MPI_GATHER and MPI_GATHERV (a vector variant of MPI_GATHER) for **gathering** data, MPI_ALLREDUCE for **reduction** operations from the *Collective communications*;

- MPI_WTIME for **timing** (*elapsed time*) the programs or sections of the codes.

An outline of the code is following.

```
include 'mpif.h'
call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, subdomains, ierr)
call MPI_GET_PROCESSOR_NAME(name, lname, ierr)
```

$\Downarrow$

$\boxed{\text{Input data}}$

$\Downarrow$

Generation of the stiffness matrix and the right-hand-side, factorizing diagonal blocks and computing entries needed for the Sherman-Morrison-Woodbury formula for each sub-domain in parallel. *No communications at this stage.*

$\Downarrow$

Solving the system (1) via PCG method with DD preconditioner

Computing the initial iterate in parallel
PCG method with BSR-BILU preconditioner.
Each processor consists of its own part of the vector of the initial iterate.
No communications at this stage.

****

Computing the initial defect in parallel.
This stage consist of matrix–vector product and addition of vectors and therefore
each processor send/receive a part of the vectors corresponds to the overlapping regions to/from his neighbors.
After that the processors work in parallel and each processor consists of its own part of the vector of the initial defect.

MPI communication primitives used:

MPI_SEND, MPI_IRECV, MPI_WAITALL

****

Computing the initial quasi-defect in parallel
PCG method with BSR-BILU preconditioner.
Each processor consists of its own part of the vector of the initial quasi-defect.
No communications at this stage.

****

Iteration cycle:

Dot products, vectors additions and matrix–vector product.

MPI communication primitives used:

MPI_SEND, MPI_IRECV, MPI_WAITALL, MPI_ALLREDUCE, MPI_SUM.

The computing of the next search direction, which leads to solving systems of equations, is done in parallel in each sub-domain without any communications (PCG method with BSR-BILU preconditioner).

$\Downarrow$

$\boxed{\text{Output data}}$
$\Downarrow$

```
call MPI_FINALIZE(ierr)
```

3.2 Parallel tests of the DD preconditioning FEM code

Parallel numerical tests of the FEM code realizing the overlapping domain decomposition approach for the benchmark problem (single pile in a weak soil media) have been performed on symmetric multiprocessor computers SUN Enterprise 3000 (up to eight processors) and SUN SPARC 20 (up to four processors) as well as on the parallel computer with distributed memory IBM SP Power 2 (up to eight processors). The *speed up* ($S_p = T_1/T_p$, where T_p is the *elapsed time* for running the code on p processors measured by MPI routine MPI_WTIME) and *parallel efficiency* ($E_p = S_p/p$) obtained during the numerical experiments can be found in the next tables.

n_1	n_2	n_3	p	S_p	E_p
32	32	32	2	1.06	0.53
			4	4.53	1.13
			8	8.74	1.09

(a) SUN Enterprise 3000

n_1	n_2	n_3	p	S_p	E_p
32	32	32	2	1.59	0.80
			4	3.87	0.97
			8	8.36	1.05

(b) IBM SP Power 2

n_1	n_2	n_3	p	S_p	E_p
32	32	32	2	1.07	0.54
			4	3.63	0.91

(c) SUN SPARCstation 20

Table 1: Domain Decomposition preconditioning

In general, the computational time decreases when the number of the processors (sub-domains) increases and the obtained parallel efficiency shows very good parallel features of the considered preconditioner. In the case, when the whole computational domain is divided in two sub-domains, the elapsed time decreases slightly and it is far from the proportionality to the number of the processors used. The reason is that in this case, the sub-domain where the pile is located is relatively large and the number of the inner iterations (PCG-BSR-BILU) is only approximately two times less than in the algorithm without subdividing. Taking into account that one needs to perform some domain decomposition iterations (outer iterations) we get the explanation of the bad results for this particular case (first row in the tables).

The super-linear speed up which can be seen in some of the runs is a result of at least following two phenomena: (a) the stronger *memory locality* increases the role of the cash memories (b) the number of inner PCG iterations decreases when the number of subdomains/processors increases; as a result the contribution of the communications to the total elapsed time decreases.

The developed code is a portable code for parallel computers with distributed memory, symmetric multiprocessor computers and clusters of workstations with MPI installation.

4 Displacement decomposition circulant block factorization preconditioning

Let us remind that a circulant matrix C has the form $(C_{k,j}) = \left(c_{(j-k) \bmod m}\right)$, where m is the size of C. Let us also denote for any given coefficients $(c_0, c_1, \ldots, c_{m-1})$ by $C = (c_0, c_1, \ldots, c_{m-1})$ the circulant matrix

$$\begin{bmatrix} c_0 & c_1 & c_2 & \ldots & c_{m-1} \\ c_{m-1} & c_0 & c_1 & \ldots & c_{m-2} \\ \vdots & \vdots & \vdots & & \vdots \\ c_1 & c_2 & \ldots & c_{m-1} & c_0 \end{bmatrix}.$$

Any circulant matrix can be factorized as

$$C = F \Lambda F^*,$$

where Λ is a diagonal matrix containing the eigenvalues of C, and F is the Fourier matrix

$$F = \frac{1}{\sqrt{m}} \left\{ e^{2\pi \frac{jk}{m} \mathbf{i}} \right\}_{0 \le j,k \le m-1}. \tag{3}$$

Here $\mathbf{i}$ stands for the imaginary unit.

4.1 Displacement decomposition

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

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

$$\kappa(M_{DD}^{-1} K) = \mathcal{O}\left(\frac{1}{1 - 2\nu_{\max}} \right), \tag{4}$$

where $\nu_{\max} = \max_\Omega \nu$, $\nu \in (0, 1/2)$ is the Poisson ratio and

$$M_{DD} = \begin{pmatrix} K_0 & 0 & 0 \\ 0 & K_0 & 0 \\ 0 & 0 & K_0 \end{pmatrix}. \tag{5}$$

The next step in the construction used here is to substitute K_0 by A_0 in (5), where A_0 stands for the Laplace stiffness matrix corresponding to linear finite elements or seven point finite difference stencil discretization. This is motivated by the more simple structure of A_0 as well as by the spectral equivalence

$$\kappa(A_0^{-1}K_0) = \mathcal{O}(1). \tag{6}$$

4.2 Circulant block factorization

The CBF preconditioning technique (see [12]) incorporates the circulant approximations into the framework of the LU block factorization.

Let us consider the 3D elliptic problem:

$$-(a(x_1,x_2,x_3)u_{x_1})_{x_1} - (b(x_1,x_2,x_3)u_{x_2})_{x_2} - (c(x_1,x_2,x_3)u_{x_3})_{x_3} = f(x_1,x_2,x_3) \tag{7}$$

on the parallelepiped $[0,x_1^{max}] \times [0,x_2^{max}] \times [0,x_3^{max}]$ with Dirichlet boundary condition. If the domain is discretized by uniform grid with n_1, n_2 and n_3 grid points along the coordinate directions, and if a standard (for such a problem) seven-point FDM (FEM) approximation is used, then the stiffness matrix K admits a block-tridiagonal structure. The matrix K can be written in the block-form

$$K = tridiag(-K_{i,i-1}, K_{i,i}, -K_{i,i+1}) \qquad i = 1,2,\ldots,n_1,$$

where $K_{i,i}$ are block-tridiagonal matrices which correspond to one x_1-plane and the off-diagonal blocks are diagonal matrices.

Now, we apply the general CBF preconditioning approach and construct the preconditioner M_{CBF} in the form:

$$M_{CBF} = tridiag(-C_{i,i-1}, C_{i,i}, -C_{i,i+1}) \qquad i = 1,2,\ldots n_1. \tag{8}$$

Here $C_{i,j} = Block - Circulant(K_{i,j})$ is a block-circulant approximation of the corresponding block $K_{i,j}$. The relative condition number of the CBF preconditioner for the model (Laplace) 3D problem is studied in [11] for $n_1 = n_2 = n_3 = n$, where the following estimate is derived:

$$\kappa(M_0^{-1}A_0) \leq 4n. \tag{9}$$

We use M_{CBF} as preconditioner for the matrix A_0. In this way we get the DD CBF preconditioner for the stiffness matrix K of the problem (1) defined by

$$M_{DD\ CBF} = \begin{pmatrix} M_0 & 0 & 0 \\ 0 & M_0 & 0 \\ 0 & 0 & M_0 \end{pmatrix}.$$

The estimate of the condition number of this preconditioner directly follows from (4), (6) and (9).

$$\kappa(M_{DD\ CBF}^{-1}K) = \mathcal{O}\left(\frac{n_{\max}}{1-2\nu_{\max}}\right). \tag{10}$$

Remark 1 *A diagonal scaling of K is used to improve the convergence rate in the performed tests with the considered benchmark problem.*

4.3 Parallel tests of the DD CBF preconditioning FEM code

The parallel MPI C code was tested on three parallel machines. We report here the speed-up S_p and the parallel efficiency E_p obtained on the considered multiprocessor computers. The benchmark problem described in section 2 is used.

Table 2(a) shows the results obtained on SGI Origin 2000 machine with 2 processors $\times$ 188 MHz and 128 Mb main memory. This is the fastest from the used machines. The parallel efficiency is close to 1 and this fact confirms that the preconditioner is highly parallelizable.

n_1	n_2	n_3	p	S_p	E_p
32	32	32	1	1	1
			2	1.800	0.900

(a) Origin 2000

n_1	n_2	n_3	p	S_p	E_p
32	32	32	2	1.993	0.997
			4	3.945	0.986

(b) SUN SPARCstation 20

n_1	n_2	n_3	p	S_p	E_p
32	32	32	2	1.968	0.984
			4	3.795	0.949
			8	5.833	0.729
48	48	48	2	1.979	0.989
			3	2.948	0.983
			4	3.857	0.964
			6	5.170	0.862
			8	6.400	0.800
64	64	32	2	1.996	0.998
			4	3.945	0.986
			8	6.949	0.869

(c) SUN Enterprise 3000

Table 2: Displacement Decomposition Circulant Block-Factorization preconditioning

The results in Table 2(b) are produced on a SUN SPARCstation 20 with 4 processors $\times$ 100 MHZ and 128 Mb main memory. The observed higher efficiency is a direct result from the relatively slower processors.

In Table 2(c) we show the results of the experiments executed on SUN Ultra-Enterprise Symmetric Multiprocessor with 8 processors $\times$ 167 MHz and 1 Gb main memory. The results are almost the same on 2 and 4 processors as in the Table 2(b). We draw attention to the fact that we measure the elapsed times and an additional reason for the lower parallel efficiency on 8 processors is that the results are obtained in multiuser mode. The size of the main memory of this machine allows to get results for finer meshes. As it was expected, the parallel efficiency increases with the size of the discrete problems.

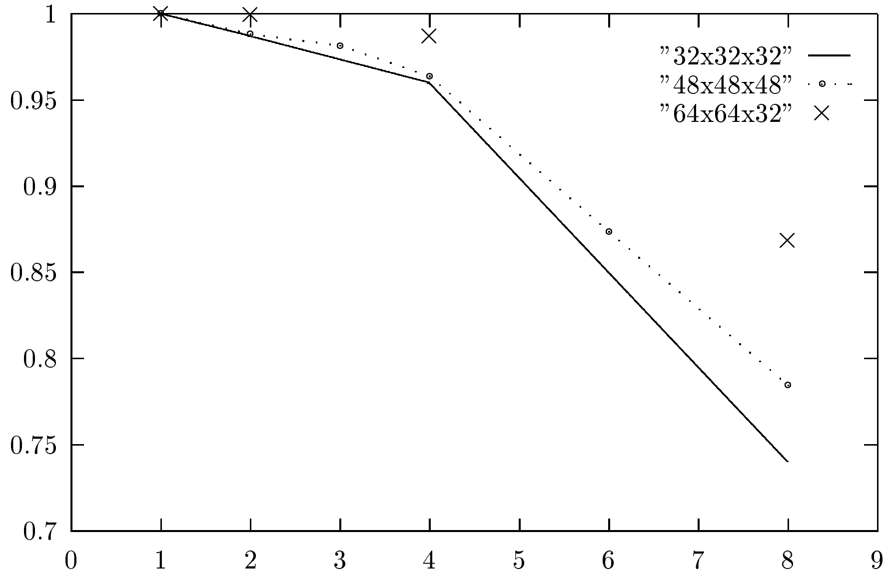

Figure 3: Parallel efficiency per iteration on SUN Enterprise 3000

Figure 3 illustrates the parallel efficiency for problems with different discretization parameters. The higher parallel efficiency for larger problems is well expressed.

There exist at least two more reasons for the high efficiency reported above:

- the network parameters *start-up time* and *time for transferring of single word* are relatively small for the multiprocessor machines;
- there is also some overlapping between the computations and the communications in the algorithm.

5 Final requirements for parallel performance of the FEM codes

The following conclusions are derived from the performed and analyzed numerical tests with the developed new parallel codes:

- The realized Domain Decomposition (DD) and Displacement Decomposition Circulant Block-Factorization (DD CBF) methods posses 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 codes obtained on different models of symmetric multiprocessors and distributed memory parallel computers with up to eight processors. For the considered benchmark problem, the highest efficiency is achieved on four processors systems.

- The speed-up and the parallel efficiency increase with the size of the discrete problem. As it has been shown above, the achieved parallel efficiency is above 80% for real-life large-scale discrete problems.
- The clusters of workstations can be considered as distributed memory computer systems where the main difference relates to the network parameters (the start-up time and the time for transferring of single word) which are relatively larger.
- The use of the Message Passing Interface (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 developed new parallel FEM codes are portable on distributed memory parallel computers and shared memory symmetric multiprocessors as well as on clusters of workstations.
- The developed codes provide 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.
- The reported codes enable opportunity for solving of very large-scale problems on distributed memory parallel computers. The decomposition strategy allows to treat efficiently such problems, where the size is limited only by the total sum of the distributed memory.

References

[1] O. Axelsson. *Iterative solution methods.* Cambridge University Press, 1994.

[2] O. Axelsson and I. Gustafsson. Iterative methods for the solution of the Navier equations of elasticity. *Comp.Meth.Appl.Mech.Eng.*, 15:241–258, 1978.

[3] T.F. Chan, S.D. Margenov, and P.S. Vassilevski. Performance of block–ILU factorization preconditioners based on block–size reduction for 2D elasticity systems. *SIAM J.Sci.Comput.*, to appear.

[4] T.F. Chan and T.P. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

[5] T.F. Chan and P.S. Vassilevski. A framework for block-ILU factorizations using block-size reduction. *Math.Comp.*, 64(209):129–156, 1995.

[6] T.F. Chan and P.S. Vassilevski. Convergence analysis of block-ILU factorization algorithms based on block–size reduction. CAM report, UCLA, to appear in Math. Comp.

[7] Message Passing Interface Forum. MPI: A message-passing interface standard. *International Journal of Supercomputer Applications*, 8(3/4), 1994. Special issue on MPI.

[8] Asen Georgiev, Angel Baltov, and Svetozar Margenov. Hipergeos benchmark problems related to bridge engineering applications. PROJECT REPORT HG CP 94–0820–MOST–4.

[9] K. Georgiev. Implementation of the additive overlapping domain decomposition method to 3D elasticity problems. *Mathematica Balkanica*, 10:419–433, 1997.

[10] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface.* The MIT Press, Cambridge, Massachusetts, 1996. Third printing.

[11] I. Lirkov and S. Margenov. Conjugate gradient method with circulant block-factorization preconditioners for 3D elliptic problems. In *Proceedings of the Workshop # 3 Copernicus 94-0820 "HIPERGEOS" project meeting, Rožnov pod Radhoštěm*, 1996.

[12] I. Lirkov, S. Margenov, and P.S. Vassilevski. Circulant block-factorization preconditioners for elliptic problems. *Computing*, 53(1):59–74, 1994.

[13] M. Snir, St. Otto, St. Huss-Lederman, D. Walker, and J. Dongara. *MPI: The Complete Reference.* Scientific and engineering computation series. The MIT Press, Cambridge, Massachusetts, 1997. Second printing.