Computer Assisted Mechanics and Engineering Sciences, 4: 69–93, 1997.

Domain decomposition PCG methods for serial and parallel processing

M. Papadrakakis and S. Bitzarakis
Institute of Structural Analysis and Aseismic Research
National Technical University of Athens, Zografou Campus, Athens 157 73

(Received May 27, 1996)

In this paper two domain decomposition formulations are presented in conjunction with the preconditioned conjugate gradient method (PCG) for the solution of large-scale problems in solid and structural mechanics. In the first approach the PCG method is applied to the global coefficient matrix, while in the second approach it is applied to the interface problem after eliminating the internal d.o.f. For both implementations a Subdomain-by-Subdomain (SBS) polynomial preconditioner is employed based on local information of each subdomain. The approximate inverse of the global coefficient matrix or the Schur complement matrix, which acts as the preconditioner, is expressed by a truncated Neumann series resulting in an additive type local preconditioner. Block type preconditioning, where full elimination is performed inside each block, is also studied and compared with the proposed polynomial preconditioning.

1. Introduction

Element-by-element (EBE) methods were motivated by the intrinsic characteristic of finite element implementations in which global data structure can be stored and maintained at the local (element) level. Fox and Stanton [1] and Fried [2] were among the first to point out that the assembly of the coefficient matrix is not essential and that the matrix-vector product required for the conjugate gradient (CG) iterative method can be obtained by performing operations at the element level. Hughes and co-workers [3] and Nour-Omid and Parlett [4] were the first to address the problem of preconditioning for the solution of differential equations based on EBE implementations. The primary purpose of the EBE applications was to keep storage requirements to a minimum, but the advent of vector and parallel computers has given to this approach a new interest especially in large-scale 3D problems. Ref. [5] reviews a wide range of preconditioning techniques for solving large-scale problems in computational mechanics.

In this work a subdomain-by-subdomain (SBS) formulation proposed in Ref. [6] is further improved and its performance is investigated in a number of test cases. Two domain decomposition implementations are considered. The first approach is based on a multi-element group partitioning of the entire domain in which the dominant matrix-vector operations of the coefficient matrix and the preconditioner are performed on SBS basis. In the second approach the PCG algorithm is applied to the interface problem after eliminating the internal d.o.f.. For this Schur complement implementation a SBS preconditioner is proposed based on the local Schur complement information of each subdomain.

The approximate inverse of the global coefficient matrix or the Schur complement matrix which acts as a preconditioner is expressed by a truncated Neumann series and an additive composition of the preconditioning matrix by its local contributions is achieved. In order to improve the efficiency of the preconditioner, in terms of number of operations, a rejection by magnitude criterion is adopted. Thus, the off-diagonal terms of the preconditioning matrix that influence to a lesser extend the outcome of its multiplication with the residual vector are rejected. A compact storage scheme which avoids storing unnecessary terms in the preconditioning matrix is adopted.

An important feature of this SBS preconditioner is its local additive nature which is inherently parallelizable without requiring complicated manipulations or any colouring schemes. Such implementation may be envisaged as mapping one or more subdomains onto each processor, while independent preconditioning matrices could be formed and operated on each processor. Block type preconditioning, where full elimination is performed inside each block, is also studied and compared with the proposed polynomial preconditioning.

The performance of the preconditioning strategies presented are tested on a number of numerical examples, while parallel tests are performed on a Silicon Graphics Challenge XL computer with 16 MIPS R4400 processors.

2. A GLOBAL SUBDOMAIN-BY-SUBDOMAIN FORMULATION

The assembling of the coefficient matrix $\mathbf{A}$ in the finite element method can be expressed as

$$\mathbf{A} = \sum_{j=1}^{s} \tilde{\mathbf{A}}^{(j)}, \tag{1}$$

in which $\tilde{\mathbf{A}}^{(j)}$ is the so called macro-element matrix of the subdomain given by $\mathbf{P}^{(j)^{\mathrm{T}}}\mathbf{A}^{(j)}\mathbf{P}^{(j)}$, where $\mathbf{P}^{(j)}$ is the global connectivity matrix of subdomain j which maps the global to local d.o.f., $\mathbf{A}^{(j)}$ is the subdomain coefficient matrix and s is the number of subdomains. The global unknown variables can be related to the unassembled global unknown variables by the transformation $\mathbf{x}^e = \mathbf{P}\mathbf{x}$, where $\mathbf{x}^e$ is defined from the subdomain unknown variables in global d.o.f. as $\mathbf{x}^e = [\mathbf{x}^{(1)}\ \mathbf{x}^{(2)}\ \dots\ \mathbf{x}^{(s)}]^{\mathrm{T}}$ and $\mathbf{P} = [\mathbf{P}^{(1)}\ \mathbf{P}^{(2)}\ \dots\ \mathbf{P}^{(s)}]^{\mathrm{T}}$. Similarly, the unassembled global right-hand side vector can be defined as $\mathbf{f}^e = [\mathbf{f}^{(1)}\ \mathbf{f}^{(2)}\ \dots\ \mathbf{f}^{(s)}]^{\mathrm{T}}$ and $\mathbf{f} = \mathbf{P}^{\mathrm{T}}\mathbf{f}^e$.

2.1. The SBS-PCG algorithm

Based on the previous definitions the subdomain-by-subdomain preconditioned conjugate gradient algorithm (SBS-PCG) for solving $\mathbf{A}\mathbf{x} = \mathbf{f}$ with preconditioning matrix $\mathbf{C}$ may be stated as follows:

Initialise:

$$\mathbf{x}_0 = \mathbf{0}, \quad \mathbf{r}_0 = \mathbf{f},$$

$$\mathbf{z}_0 = \mathbf{C}^{-1}\mathbf{r}_0,$$

$$\mathbf{d}_0 = \mathbf{z}_0$$

for $m = 0, 1, \dots$

Coefficient matrix — vector operation:

$$\{\mathbf{d}^e\}_m = \mathbf{P}\mathbf{d}_m = [\mathbf{d}^{(1)}\ \mathbf{d}^{(2)}\ \dots\ \mathbf{d}^{(s)}]_m^{\mathrm{T}},$$

$$\{\mathbf{u}^{(j)}\}_m = \mathbf{A}^{(j)}\{\mathbf{d}^{(j)}\}_m, \quad j = 1, 2, \dots, s,$$

$$\{\mathbf{u}^e\}_m = [\mathbf{u}^{(1)}\ \mathbf{u}^{(2)}\ \dots\ \mathbf{u}^{(s)}]_m^{\mathrm{T}},$$

$$\mathbf{u}_m = \mathbf{P}^{\mathrm{T}}\{\mathbf{u}^e\}_m.$$

Vector operations:

$$\alpha_m = \frac{\mathbf{r}_m^{\mathrm{T}}\mathbf{z}_m}{\mathbf{d}_m^{\mathrm{T}}\mathbf{u}_m}, \tag{2}$$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \alpha_m \mathbf{d}_m ,$$

$$\mathbf{r}_{m+1} = \mathbf{r}_m - \alpha_m \mathbf{u}_m .$$

Convergence check:

$$\texttt{if} \quad \|\mathbf{r}_{m+1}\| \leq \varepsilon \|\mathbf{r}_0\| \quad \texttt{stop, else continue}$$

Preconditioning matrix — vector operation:

$$\mathbf{z}_{m+1} = \mathbf{C}^{-1} \mathbf{r}_{m+1} .$$

Vector operations:

$$\beta_m = \frac{\mathbf{r}_{m+1}^{\mathrm{T}} \mathbf{z}_{m+1}}{\mathbf{r}_m^{\mathrm{T}} \mathbf{z}_m} ,$$

$$\mathbf{d}_{m+1} = \mathbf{z}_m + \beta_m \mathbf{d}_m$$

end for

3. SCHUR COMPLEMENT DOMAIN DECOMPOSITION

Let us consider a finite element domain D subdivided into s non-overlapping subdomains (D_j, $j = 1, ..., s$). The mesh nodes which are common to the subdomain interfaces define a global interface noted by N_b. Numbering first the nodal point unknowns within D_j not belonging to N_b and last the interface nodes N_b, results in an arrow pattern for the global coefficient matrix. Thus, the equations of equilibrium $\mathbf{Ax} = \mathbf{f}$ for the displacement based finite element method can be written as

$$\begin{bmatrix} \mathbf{A}_{ii}^{(1)} & \mathbf{0} & & \mathbf{0} & \tilde{\mathbf{A}}_{ib}^{(1)} \\ \mathbf{0} & \mathbf{A}_{ii}^{(1)} & & \mathbf{0} & \tilde{\mathbf{A}}_{ib}^{(2)} \\ & & \ddots & & \vdots \\ \mathbf{0} & \mathbf{0} & & \mathbf{A}_{ii}^{(s)} & \tilde{\mathbf{A}}_{ib}^{(s)} \\ \tilde{\mathbf{A}}_{ib}^{(1)\mathrm{T}} & \tilde{\mathbf{A}}_{ib}^{(2)\mathrm{T}} & \dots & \tilde{\mathbf{A}}_{ib}^{(s)\mathrm{T}} & \mathbf{A}_{bb} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_i^{(1)} \\ \mathbf{x}_i^{(2)} \\ \vdots \\ \mathbf{x}_i^{(s)} \\ \mathbf{x}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_i^{(1)} \\ \mathbf{f}_i^{(2)} \\ \vdots \\ \mathbf{f}_i^{(s)} \\ \mathbf{f}_b \end{Bmatrix} \tag{3}$$

with

$$\mathbf{A}_{bb} = \sum_{j=1}^{s} \tilde{\mathbf{A}}_{bb}^{(j)} \quad \text{and} \quad \mathbf{f}_b = \sum_{j=1}^{s} \tilde{\mathbf{f}}_b^{(j)} \tag{4}$$

and

$$\tilde{\mathbf{A}}_{bb}^{(j)} = \mathbf{P}_b^{(j)\mathrm{T}} \mathbf{A}_{bb}^{(j)} \mathbf{P}_b^{(j)} , \quad \tilde{\mathbf{A}}_{ib}^{(j)} = \mathbf{A}_{ib}^{(j)} \mathbf{P}_b^{(j)} , \quad \tilde{\mathbf{f}}_b^{(j)} = \mathbf{P}_b^{(j)\mathrm{T}} \mathbf{f}_b^{(j)} . \tag{5}$$

$\mathbf{A}_{ii}^{(j)}$, $\mathbf{A}_{bb}^{(j)}$ correspond to the internal and the interface d.o.f. of the subdomain D_j respectively, while $\mathbf{A}_{ib}^{(j)}$ corresponds to the interaction of the internal and the interface d.o.f.. $\tilde{\mathbf{A}}_{bb}^{(j)}$, $\tilde{\mathbf{A}}_{ib}^{(j)}$ and $\tilde{\mathbf{f}}_b^{(j)}$ are the macro-element matrices and vector respectively and $\mathbf{P}_b^{(j)}$ is the global interface connectivity matrix of the subdomain j which maps the local interface d.o.f. to global interface d.o.f.

The concept behind the domain decomposition approach is to reduce the initial system of equations to a system comprising only the interface d.o.f. Applying a static condensation to the subdomain internal d.o.f., Eq. (3) can be transformed into the following problem for the interface unknowns:

$$\left(\mathbf{A}_{bb} - \sum_{j=1}^{s} \tilde{\mathbf{A}}_{ib}^{(j)^{\mathrm{T}}} \mathbf{A}_{ii}^{(j)^{-1}} \tilde{\mathbf{A}}_{ib}^{(j)}\right) \mathbf{x}_b = \mathbf{f}_b - \sum_{j=1}^{s} \tilde{\mathbf{A}}_{ib}^{(j)^{\mathrm{T}}} \mathbf{A}_{ii}^{(j)^{-1}} \mathbf{f}_i^{(j)} \tag{6}$$

or

$$\mathbf{S}\mathbf{x}_b = \hat{\mathbf{f}}_b\,, \tag{7}$$

where

$$\mathbf{S} = \sum_{j=1}^{s} \tilde{\mathbf{S}}^{(j)} = \sum_{j=1}^{s} \mathbf{P}^{(j)^{\mathrm{T}}} \mathbf{S}^{(j)} \mathbf{P}^{(j)}\,, \tag{8}$$

$$\mathbf{S}^{(j)} = \mathbf{A}_{bb}^{(j)} - \mathbf{A}_{ib}^{(j)^{\mathrm{T}}} \mathbf{A}_{ii}^{(j)^{-1}} \mathbf{A}_{ib}^{(j)}\,, \tag{9}$$

$$\hat{\mathbf{f}}_b = \mathbf{f}_b - \sum_{j=1}^{s} \mathbf{P}_b^{(j)^{\mathrm{T}}} \mathbf{A}_{ib}^{(j)^{\mathrm{T}}} \mathbf{A}_{ii}^{(j)^{-1}} \mathbf{f}_i^{(j)}\,. \tag{10}$$

Matrix $\mathbf{S}$ is the Schur complement of $\mathbf{A}_{bb}$ in $\mathbf{A}$ and each subdomain matrix $\mathbf{S}^{(j)}$ corresponds to a local static condensation operator or a local Schur complement.

3.1. The Schur complement PCG algorithm

The matrix $\mathbf{S}$ is generally dense, and therefore a direct method for solving system (7) could be prohibitively expensive if N_b is large. The usual implementation for solving the Schur complement matrix equations is to apply PCG methods. In applying the PCG method one needs to evaluate the matrix vector product $\mathbf{S}\{\mathbf{d}_b\}_m$ for the direction vector $\{\mathbf{d}_b\}_m$ at iteration m. In this case, $\mathbf{S}$ does not need to be explicitly assembled. Instead, each matrix-vector product can be efficiently performed using subdomain-by-subdomain sparse matrix-vector products and local forward and backward substitutions.

The global unknown interface variables can, in this case, be related to the unassembled global unknown interface variables by the transformation $\mathbf{x}_b^e = \mathbf{P}_b \mathbf{x}_b$, where $\mathbf{x}_b^e$ is defined from the subdomain unknown interface variables in global d.o.f. as $\mathbf{x}_b^e = [\mathbf{x}_b^{(1)}\ \mathbf{x}_b^{(2)}\ \ldots\ \mathbf{x}_b^{(s)}]^{\mathrm{T}}$, and $\mathbf{P}_b = [\mathbf{P}_b^{(1)}\ \mathbf{P}_b^{(2)}\ \ldots\ \mathbf{P}_b^{(s)}]^{\mathrm{T}}$. Similarly, the unassembled global interface right-hand side vector can be defined as $\hat{\mathbf{f}}_b^e = [\hat{\mathbf{f}}_b^{(1)}\ \hat{\mathbf{f}}_b^{(2)}\ \ldots\ \hat{\mathbf{f}}_b^{(s)}]^{\mathrm{T}}$ and $\hat{\mathbf{f}}_b = \mathbf{P}_b{}^{\mathrm{T}} \hat{\mathbf{f}}_b^e$.

The implementation of the PCG algorithm in the domain decomposition formulation of Eq. (3) based on the Schur complement equations (7) may be realised as follows:

Initialise:

$$\{\mathbf{x}_b\}_0 = \mathbf{0}\,, \quad \{\mathbf{r}_b\}_0 = \hat{\mathbf{f}}_b\,,$$

$$\{\mathbf{z}_b\}_0 = \mathbf{C}_s^{-1} \{\mathbf{r}_b\}_0\,,$$

$$\{\mathbf{d}_b\}_0 = \{\mathbf{z}_b\}_0$$

for $m = 0, 1, \ldots$

Coefficient matrix — vector operation:

$$\{\mathbf{d}_b^e\}_m = \mathbf{P}_b\{\mathbf{d}_b\}_m = \left[\mathbf{d}_b^{(1)}\ \mathbf{d}_b^{(2)}\ \dots\ \mathbf{d}_b^{(s)}\right]_m^{\mathrm{T}},$$

$$\left\{\mathbf{u}_b^{(j)}\right\}_m = \mathbf{S}^{(j)}\left\{\mathbf{d}_b^{(j)}\right\}_m, \quad j = 1, 2, \dots, s,$$

$$\{\mathbf{u}_b^e\}_m = \left[\mathbf{u}_b^{(1)}\ \mathbf{u}_b^{(2)}\ \dots\ \mathbf{u}_b^{(s)}\right]_m^{\mathrm{T}},$$

$$\{\mathbf{u}_b\}_m = \mathbf{P}_b^{\mathrm{T}}\{\mathbf{u}_b^e\}_m.$$

Vector operations:

$$\alpha_m = \frac{\{\mathbf{r}_b\}_m^{\mathrm{T}}\{\mathbf{z}_b\}_m}{\{\mathbf{d}_b\}_m^{\mathrm{T}}\{\mathbf{u}_b\}_m}, \tag{11}$$

$$\{\mathbf{x}_b\}_{m+1} = \{\mathbf{x}_b\}_m + \alpha_m\{\mathbf{d}_b\}_m,$$

$$\{\mathbf{r}_b\}_{m+1} = \{\mathbf{r}_b\}_m - \alpha_m\{\mathbf{u}_b\}_m.$$

Convergence check:

`if` $\|\{\mathbf{r}_b\}_{m+1}\| \le \varepsilon\,\|\{\mathbf{r}_b\}_0\|$ `stop, else continue`

Preconditioning matrix — vector operation:

$$\{\mathbf{z}_b\}_{m+1} = \mathbf{C}_s^{-1}\{\mathbf{r}_b\}_{m+1}.$$

Vector operations:

$$\beta_m = \frac{\{\mathbf{r}_b\}_{m+1}^{\mathrm{T}}\{\mathbf{z}_b\}_{m+1}}{\{\mathbf{r}_b\}_m^{\mathrm{T}}\{\mathbf{z}_b\}_m},$$

$$\{\mathbf{d}_b\}_{m+1} = \{\mathbf{z}_b\}_m + \beta_m\{\mathbf{d}_b\}_m$$

end for

Once the interface variables $\mathbf{x}_b$ are determined, the internal variables $\mathbf{x}_i^{(j)}$ can be solved from

$$\mathbf{x}_i^{(j)} = -\mathbf{A}_{ii}^{(j)^{-1}}\left(\tilde{\mathbf{A}}_{ib}^{(j)}\mathbf{x}_b - \mathbf{f}_i\right). \tag{12}$$

The matrix-vector product $\mathbf{S}^{(j)}\left\{\mathbf{d}_b^{(j)}\right\}_m$ can be reformulated as a subdomain problem with non-zero Dirichlet boundary conditions $\mathbf{d}_b^{(j)}$:

$$\begin{bmatrix} \mathbf{A}_{ii}^{(j)} & \mathbf{A}_{ib}^{(j)} \\ \mathbf{A}_{bi}^{(j)} & \mathbf{A}_{bb}^{(j)} \end{bmatrix} \begin{Bmatrix} \mathbf{d}_i^{(j)} \\ \mathbf{d}_b^{(j)} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{u}_b^{(j)} \end{Bmatrix}, \tag{13}$$

where $\mathbf{u}_b^{(j)}$ represents the interface tractions that correspond to the displacements $\mathbf{d}_b^{(j)}$. After eliminating $\mathbf{d}_i^{(j)}$ the second equation becomes

$$\mathbf{u}_b^{(j)} = \mathbf{S}^{(j)}\mathbf{d}_b^{(j)}, \tag{14}$$

consequently

$$\mathbf{u}_b = \sum_{j=1}^{s}\mathbf{P}_b^{(j)^{\mathrm{T}}}\mathbf{u}_b^{(j)}. \tag{15}$$

4. AN ADDITIVE PRECONDITIONER

The implementation of algorithms (2) or (11) require the solution of the systems $\mathbf{C}\mathbf{z}_m = \mathbf{r}_{bm}$ or $\mathbf{C}_s\{\mathbf{z}_b\}_m = \{\mathbf{r}_b\}_m$ at each iteration, where $\mathbf{C}$ and $\mathbf{C}_s$ are the preconditioning matrices of the coefficient and Schur complement matrices respectively. It is apparent that the more $\mathbf{C}$ or $\mathbf{C}_s$ resembles $\mathbf{A}$ or $\mathbf{S}$, the more the ellipticity of the transformed coefficient-matrix is reduced and the convergence rate of PCG is improved.

4.1. The global SBS additive preconditioner

A SBS preconditioner based on polynomial preconditioning techniques is proposed in Ref. [6]. A more general formulation for approximating the inverse of $\mathbf{A}$ as well as $\mathbf{S}$ is given hereinafter. In order to establish approximate expressions for $\mathbf{A}^{-1}$ we define the auxiliary matrix $\bar{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{\mathrm{T}}$, where $\mathbf{B} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$ is a block diagonal matrix of $\mathbf{A}$ and $\mathbf{L}$ is the complete factor of $\mathbf{B}$. Then, the inverse of $\bar{\mathbf{A}}$ is expressed as

$$\bar{\mathbf{A}}^{-1} = (\mathbf{I} - \mathbf{V})^{-1} \tag{16}$$

with

$$\mathbf{V} = \mathbf{L}^{-1}(\mathbf{B} - \mathbf{A})\mathbf{L}^{-\mathrm{T}}. \tag{17}$$

For $|\mathbf{V}| < 1$ the inverse of $(\mathbf{I} - \mathbf{V})$ can be written by the polynomial expansion

$$(\mathbf{I} - \mathbf{V})^{-1} \approx \eta_0\mathbf{I} + \eta_1\mathbf{V} + \eta_2\mathbf{V}^2 + \ldots + \eta_k\mathbf{V}^k, \tag{18}$$

where η_k are properly selected coefficients for accelerated convergence.

Following the same definition for the subdomain quantities of $\mathbf{V}$ as for $\mathbf{A}$ with $\mathbf{V}^{(j)} = \mathbf{L}^{-1}(\mathbf{B}^{(j)} - \mathbf{A}^{(j)})\mathbf{L}^{-\mathrm{T}}$ the inverse expression (18) may be written as

$$(\mathbf{I} - \mathbf{V})^{-1} \approx \eta_0\mathbf{I} + \eta_1\sum_{j=1}^{s}\tilde{\mathbf{V}} + \eta_2\left(\sum_{j=1}^{s}\tilde{\mathbf{V}}^{(j)}\right)^2 + \ldots + \eta_k\left(\sum_{j=1}^{s}\tilde{\mathbf{V}}^{(j)}\right)^k, \tag{19}$$

An approximation to the inverse can be constructed by approximating the power of the sum by the sum of the power as follows:

$$(\mathbf{I} - \mathbf{V})^{-1} \approx \eta_0\mathbf{I} + \eta_1\sum_{j=1}^{s}\tilde{\mathbf{V}} + \eta_2\sum_{j=1}^{s}\tilde{\mathbf{V}}^{(j)^2} + \ldots + \eta_k\sum_{j=1}^{s}\tilde{\mathbf{V}}^{(j)^k}, \tag{20}$$

Thus, the inverse preconditioning matrix

$$\mathbf{C}^{-1} = \mathbf{L}^{-T}(\mathbf{I} - \mathbf{V})\mathbf{L}^{-1}. \tag{21}$$

may be written

$$\mathbf{C}^{-1} = \eta_0\mathbf{B}^{-1} + \sum_{j=1}^{s}\tilde{\mathbf{R}}^{(j)}. \tag{22}$$

Based on the above formulation the operation $\mathbf{z}_{m+1} = \mathbf{C}^{-1}\mathbf{r}_{m+1}$ of algorithm (2) becomes:

$$\begin{aligned}
\mathbf{z}_{m+1} &= \eta_0\mathbf{B}^{-1}\mathbf{r}_{m+1}, \\
\{\mathbf{r}^e\}_{m+1} &= \mathbf{P}\mathbf{r}_{m+1}, \\
\{\mathbf{z}^{(j)}\}_{m+1} &= \tilde{\mathbf{R}}^{(j)}\{\mathbf{r}_b{}^{(j)}\}_{m+1}, \quad j = 1, 2, \ldots, s, \\
\{\mathbf{z}^e\}_{m+1} &= [\mathbf{z}^{(1)} \ \ldots \ \mathbf{z}^{(s)}]^{\mathrm{T}}_{m+1}, \\
\mathbf{z}_{m+1} &= \mathbf{P}^{\mathrm{T}}\{\mathbf{z}^e\}_{m+1} + \bar{\mathbf{z}}_{m+1}.
\end{aligned} \tag{23}$$

4.2. The Schur complement SBS additive preconditioner

The additive preconditioning matrix implemented in the Schur complement formulation follows the same lines of the SBS preconditioner. The inverse of $\bar{\mathbf{S}} = \mathbf{D}_s^{-1/2}\mathbf{S}\mathbf{D}_s^{-1/2}$ is expressed as

$$\bar{\mathbf{S}}^{-1} = (\mathbf{I} - \mathbf{V})^{-1} \tag{24}$$

with

$$\mathbf{V} = \mathbf{D}_s^{-1/2}(\mathbf{D}_s - \mathbf{S})\mathbf{D}_s^{-1/2}, \tag{25}$$

where

$$\mathbf{D}_s = \sum_{j=1}^{s} \mathbf{P}_b^{(j)^{\mathrm{T}}} \mathbf{D}^{(j)} \mathbf{P}_b^{(j)} \tag{26}$$

is the assembled global diagonal matrix with the diagonal entries of $\mathbf{S}$ and $\mathbf{D}^{(j)}$ is the diagonal matrix with the diagonal entries of $\mathbf{S}^{(j)}$.

Following the previous derivation the preconditioning matrix may now be written as

$$\mathbf{C}_s^{-1} = \mathbf{D}_s^{-1/2}(\mathbf{I} - \mathbf{V})\mathbf{D}_s^{-1/2}, \tag{27}$$

or

$$\mathbf{C}_s^{-1} = \eta_0 \mathbf{D}_s^{-1} + \sum_{j=1}^{s} \tilde{\mathbf{R}}_s^{(j)} \tag{28}$$

Thus the operation $\{\mathbf{z}_b\}_{m+1} = \mathbf{C}_s^{-1}\{\mathbf{r}_b\}_{m+1}$ of the algorithm (11) becomes:

$$\begin{aligned}
\{\bar{\mathbf{z}}_b\}_{m+1} &= \eta_0 \mathbf{D}_s^{-1}\{\mathbf{r}_b\}_{m+1}, \\
\{\mathbf{r}_b^e\}_{m+1} &= \mathbf{P}_b\{\mathbf{r}_b\}_{m+1}, \\
\{\mathbf{z}_b^{(j)}\}_{m+1} &= \tilde{\mathbf{R}}_s^{(j)}\{\mathbf{r}_b^{(j)}\}_{m+1}, \quad j = 1, 2, \ldots, s, \\
\{\mathbf{z}_b^e\}_{m+1} &= [\mathbf{z}_b^{(1)} \ \ldots \ \mathbf{z}_b^{(s)}]_{m+1}^{\mathrm{T}}, \\
\{\mathbf{z}_b\}_{m+1} &= \mathbf{P}_b^{\mathrm{T}}\{\mathbf{z}_b^e\}_{m+1} + \bar{\mathbf{z}}_{m+1}.
\end{aligned} \tag{29}$$

4.3. The polynomial coefficients η_i

Polynomial preconditioning consists of choosing a polynomial $p(\bar{\mathbf{A}})$ or $p(\bar{\mathbf{S}})$ to replace $\bar{\mathbf{A}}^{-1}$ or $\bar{\mathbf{S}}^{-1}$. The simplest polynomial expansion of equation (18) is given by the Neumann series with $\eta_0 = \ldots = \eta_k = 1 : p(\mathbf{H}) = \mathbf{I} + \mathbf{V} + \mathbf{V}^2 + \ldots$, where $\mathbf{H}$ is $\bar{\mathbf{A}}$ or $\bar{\mathbf{S}}$. More efficient polynomials can be obtained by bringing the spectrum of the preconditioned matrix as close as possible to that of the identity. One way to achieve this is to minimise the residuals $\|\mathbf{I} - p(\mathbf{H})\mathbf{H}\|$, where $\|\cdot\|$, where represents the L_2-norm.

According to Refs. [7, 8] this can be achieved by defining the best polynomial which minimises

$$\max_{\mu \in [a,b]} |1 - \mu p(\mu)|, \tag{30}$$

where $[a, b]$ is some interval containing the eigenvalues of H, with $0 < a < b$. The Chebychev polynomials or alternatively a least squares minimisation procedure may be implemented to solve this problem. The latter approach takes the form: *Find p that minimises* $\|1 - \mu p(\mu)\|_w$, *where* $\|\cdot\|_w$, *is the* L_2*-norm on* $[a, b]$ *with respect to some weight function* $w(\mu)$. It has been observed in Ref. [8]

that least squares polynomials tend to perform better than those based on the Chebychev ones, in that they lead to a better overall clustering of the spectrum.

Following the derivation of Ref. [8] the parameters a and b are computed as the Gershgorin estimates $\mu_{\min}$ and $\mu_{\max}$ of the extreme eigenvalues of the matrix $\mathbf{H}$. However, since the Gershgorin lower bound a may be non positive even when $\mathbf{H}$ is positive definite and the upper bound b is usually overestimated, the least squares polynomials are thus defined in the interval $[0, \bar{b}]$, where $\bar{b} = \rho b$ and ρ is a user defined reduction factor. The computed first two polynomials take the form

$$p_1(\mu) = -\frac{5}{4}\bar{b} + \mu\,, \tag{31}$$

$$p_2(\mu) = \frac{7}{8}\bar{b}^2 - \frac{7}{4}\bar{b}\mu + \mu^2 \tag{32}$$

and the coefficients in equation (18) correspond to the coefficients of μ in equations (31) and (32), respectively. Thus, for the first order approximation $\eta_0 = -5\bar{b}/4$, $\eta_1 = 1$ and for the second order approximation $\eta_0 = 7\bar{b}^2/8$, $\eta_1 = -7\bar{b}/4$, $\eta_2 = 1$.

4.4. Computational aspects

The preconditioning approach based on the polynomial expansion of the inverse $\bar{\mathbf{A}}$ or $\bar{\mathbf{S}}$ has the following features: A first or second order expansion is used in this study. The multiplications $\mathbf{R}^{(j)}\{\mathbf{r}^{(j)}\}_m$ or $\mathbf{R}_s^{(j)}\{\mathbf{r}_b^{(j)}\}_m$ are performed explicitly rather than implicitly. An implicit multiplication would involve extended computations in each iteration. When this multiplication is performed explicitly $\mathbf{R}^{(j)}$ or $\mathbf{R}_s^{(j)}$ are formed only once and stored in single precision arithmetic before the iterative procedure begins.

In order to further reduce the operations involved in this multiplication a rejection by magnitude criterion is adopted for the off-diagonal terms of $\mathbf{R}^{(j)}$ or $\mathbf{R}_s^{(j)}$ by comparing their values with the corresponding assembled diagonal terms:

$$(R_{kl})^2 \leq \psi D_k^{-1} D_l^{-1}\,. \tag{33}$$

This criterion controls the terms to be retained in $\mathbf{R}^{(j)}$ or $\mathbf{R}_s^{(j)}$ according to their magnitude. D_k is the global Schur complement component of the diagonal entries of $\mathbf{S}$ at the k-th d.o.f. The control parameter ψ takes values $0 < \psi < 1$.

In both SBS and Schur complement PCG algorithms the subdomain contributions are considered to be contributions by groups of elements depending on the partition of the domain into a number of subdomains. This subdivision on the domain level is equivalent with an overlapping block partitioning in which each block corresponds to the assembled coefficient matrix of each subdomain with or without condensation of the internal d.o.f. Thus, operations are performed at the level of each subdomain j $(j = 1, ..., s)$ and the formation of the global coefficient matrix or Schur complement matrix is avoided.

The replacement of $\left(\sum_{j=1}^{s} \tilde{\mathbf{V}}^{(j)}\right)^k$ by $\sum_{j=1}^{s} \tilde{\mathbf{V}}^{(j)^k}$ aims in reducing the multiplications between subdomains in order to keep operations and communication overheads to a minimum and facilitate the parallel implementation of the method.

In the Schur complement PCG algorithm the Schur complement of each subdomain is formed with a modified decomposition algorithm proposed by Han and Abel [9]. This matrix decomposition generates a rectangular matrix $\mathbf{W}$ by performing forward and backward substitutions according to

$$\mathbf{A}_{ii}^{(j)}\mathbf{W}^{(j)} = \mathbf{A}_{ib}^{(j)}\,. \tag{34}$$

Then the subdomain Schur complement follows from

$$\mathbf{S}^{(j)} = \mathbf{A}_{bb}^{(j)} - \mathbf{W}^{(j)\mathrm{T}}\mathbf{W}^{(j)} . \tag{35}$$

This approach requires about half the computations of the conventional approach but it needs to store $\mathbf{W}^{(j)}$ in full. When memory limitations exist, the $\mathbf{S}^{(j)}$ may not be formed and the matrix-vector multiplications can be performed implicitly.

5. Block diagonal preconditioning

A subdivision of the basic global equations $\mathbf{Ax} = \mathbf{f}$, on the algebraic level, corresponding to the subdivision of the variables in r subsets $\mathbf{x} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_r]^{\mathrm{T}}$ gives the diagonal submatrices $\mathbf{A}_{11}$, $\mathbf{A}_{22}$, ..., $\mathbf{A}_{rr}$ which may be decomposed by Cholesky factorization according to $\mathbf{A}_{ii} = \mathbf{L}_{ii}\mathbf{L}_{ii}^{\mathrm{T}}$. Defining $\bar{\mathbf{L}}$ to be of the block diagonal form $\bar{\mathbf{L}} = [\mathbf{L}_{11}\ \mathbf{L}_{22}\ \ldots\ \mathbf{L}_{rr}]$, the preconditioning matrix is then given by $\mathbf{B} = \bar{\mathbf{L}}\bar{\mathbf{L}}^{\mathrm{T}}$. Then, the transformed coefficient matrix $\bar{\mathbf{A}} = \bar{\mathbf{L}}^{-1}\mathbf{A}\bar{\mathbf{L}}^{-\mathrm{T}}$ has unit block diagonal submatrices and $\bar{\mathbf{L}}_{ii}^{-1}\mathbf{A}_{ij}\bar{\mathbf{L}}_{jj}^{-\mathrm{T}}$ are the off diagonal submatrices.

Based on the above definitions the CG algorithm with a block diagonal preconditioner on the algebraic level performs iterations on the transformed space $\bar{\mathbf{r}}_j = \mathbf{L}_{jj}^{-1}\mathbf{b}$, $\bar{\mathbf{x}} = \mathbf{L}_{jj}^{\mathrm{T}}\mathbf{x}$, $j = 1, \ldots, r$. After the convergence check $\|\bar{\mathbf{r}}_{m+1}\| \leq \varepsilon_0 \|\bar{\mathbf{r}}_0\|$ is satisfied the required subvectors of $\mathbf{x}$ are computed by $\mathbf{x}_j = \mathbf{L}_{jj}^{\mathrm{T}}\bar{\mathbf{x}}_j$. For this implementation vector $\{\mathbf{u}^{(j)}\}_m$ of the PCG algorithm is computed as follows :

$$\begin{aligned}
\{\mathbf{z}_j\}_m &= \mathbf{L}_{jj}^{-\mathrm{T}}\{\mathbf{d}_j\}_m , \quad j = 1, \ldots, r , \\
\{\mathbf{u}_j\}_m &= \{\mathbf{d}_j\}_m + \mathbf{L}_{jj}^{-\mathrm{T}} \left(\sum_{j \neq k} \{\mathbf{z}_k\}_m \right) , \quad j = 1, \ldots, r , \\
\mathbf{u}_m &= [\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_r]_m^{\mathrm{T}} .
\end{aligned} \tag{36}$$

The two extreme cases of the block-type preconditioning correspond to the diagonal scaling, when the order of the blocks is 1×1, and to the standard Cholesky factorization where there is one block of order $n \times n$. As the block size increases, the convergence rate is improving but as more fill-in is occurring the computation time per iteration is increasing. Thus, the size of the block should be combined with small amount of fill-in during the block factorizations. Block-type factorization preconditioners have been proposed in Refs. [10, 11] among others. In our implementation the block-type preconditioning is combined with the domain decomposition concept where the multiplication $\mathbf{A}_{jk}\{\mathbf{z}_k\}_m$ is performed on the subdomain level and not on the global level.

6. Numerical results

The numerical algorithms proposed in the previous sections are applied to three structural analysis test problems. Example 1 is a clamped orthogonal plate with a central point load. Due to symmetry the quarter of the plate is discretised with a mesh of 32×32 quadrilateral elements (3,008 d.o.f.). The element aspect ratio (ratio of the largest dimension to the smallest) is taken 4 in order to introduce ill-conditioning to the stiffness matrix. Example 2 is a cantilever beam subjected to transverse end loading. The beam is discretised with a mesh of 96×24 quadrilateral plane stress elements giving 4,800 d.o.f.. The element aspect ratio is also taken to be 4. Example 3 is a parabolic cylindrical shell, shown in Figure 1, clamped in its straight edges with a loading along its longitudinal axis of symmetry. Two discretizations are tested for the one quarter of the shell: In Test case 1 a 12×24 mesh of 9-node Lagrangian elements results in 5,734 d.o.f., while in Test case 2 the corresponding descretization is 24×48 elements resulting in 22,990 d.o.f.. The element aspect ratio is approximately

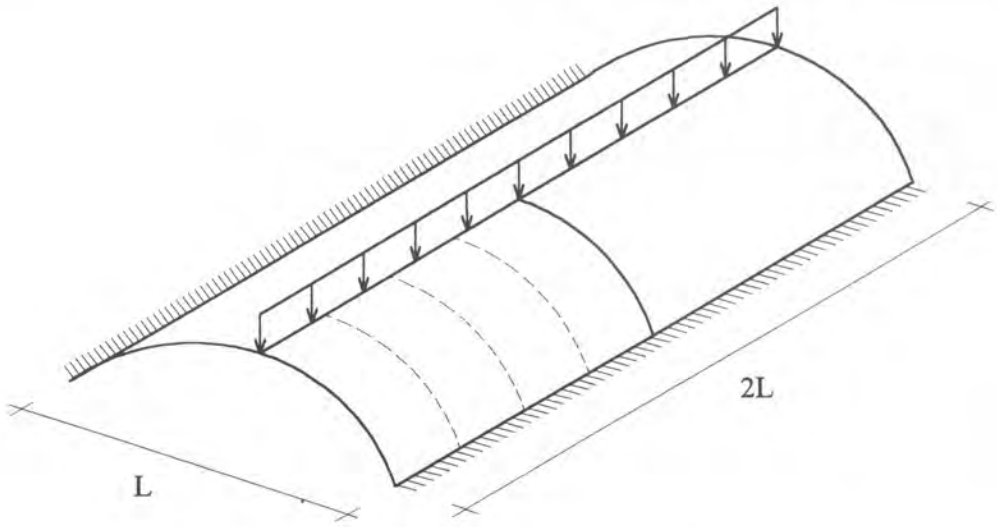

Fig. 1. Geometric representation of the shell example

one and the width over the radius of curvature is 0.0125. When not explicitly stated a second order expansion is used for the polynomial preconditioning. The numerical results for Examples 1, 2 and Test 1 of Example 3 are performed on a SG Indigo R4000 workstation, while the numerical results for Test case 2 of Example 3 are performed on a SG Challenge XL computer with 16 MIPS R4400 processors.

Figure 2 gives a comparison between the global block type and the global SBS preconditioning for the plate example divided with one way dissection into 8 subdomains. In the latter case the factorized operator of equation (21) is taken $\mathbf{L} = \mathbf{D}^{1/2}$, where $\mathbf{D}$ is the diagonal matrix with the diagonal entries of $\mathbf{A}$. It can be observed that as the block size increases the convergence rate is improving, but as more fill-in is occurring the computation time is not substationally decreasing. In order for the block type preconditioner to be effective the size of the block has to be relatively large but this produces excessive storage requirements due to extensive fill-in. For this reason the diagonal form for $\mathbf{L}$ is adopted in all subsequent tests of the global SBS-PCG implementations.

Figure 3 shows the influence of the stiffness matrix handling on the total CPU time of the global SBS-PCG for different number of subdomains for the plate example. It can be observed that the skyline storage of the matrices has a substantial influence on the computing time due to the increased operations with zero terms on the matrix-vector multiplications. The compact storage, on the other hand, seems not to be affected by the number of subdivisions of the entire domain. The remaining tests have been performed with a compact storage handling of the stiffness matrix.

Figures 4 and 5 depict the performance of the global SBS preconditioner with respect to the rejection factor ψ which controls the non-zero terms to be retained in matrix $\mathbf{R}^{(j)}$ of equation (22) and the reduction factor ρ of the Gershgorin estimate $\mathbf{b}$ for the evaluation of polynomial coefficients η_i. The fixed coefficients used in Ref. [6] correspond to the values $\eta_0 = \eta_1 = 1$, $\eta_2 = 1/4$. Figure 6 shows the influence of the number of subdomains on the performance of the method for the cantilever beam.

Figure 7 depicts the performance of Diagonal and global SBS preconditioners for different values of the rejection factor ψ ($\rho = 0.50$) for the shell example (Test case 1) divided into 3, 6 and 12 subdomains. It should be mentioned that the fixed coefficients ($\eta_0 = \eta_1 = 1$, $\eta_2 = 1/4$) in the polynomial expansion of the inverse matrix did not achieve convergence for this example. Figures 8 and 9 depict the performance of the SBS preconditioner with respect to the rejection factor ψ. In Figure 8 the SBS preconditioner is implemented on the global coefficient matrix, while in Figure 9 the SBS preconditioner is implemented on the Schur complement. In both cases a subdivision with

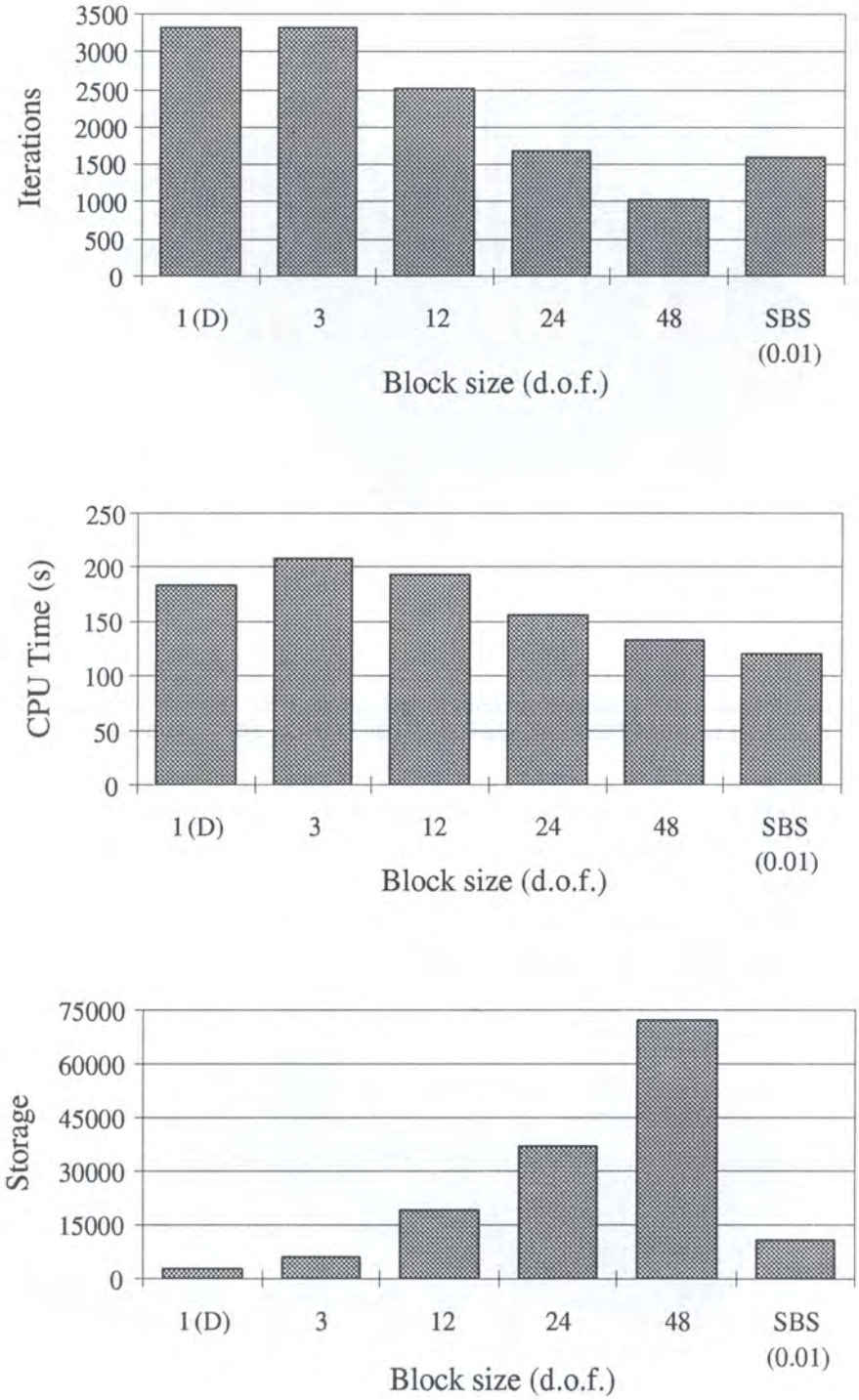

Fig. 2. Example 1: Performance of Diagonal (D), Block Diagonal and Global SBS ($\psi = 0.01$) preconditioners. Division into 8 subdomains (Storage refers to non-zero terms of preconditioner. Stiffness matrix non-zero terms: 28,966)

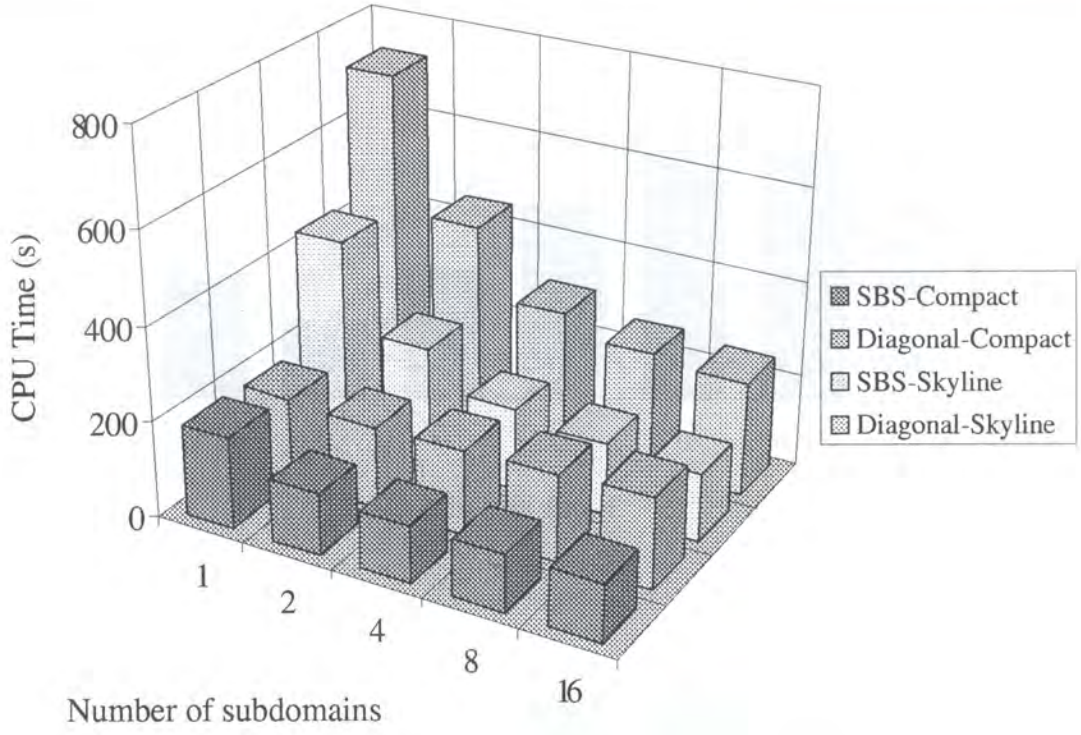

Fig. 3. Example 1: Influence of the stiffness matrix storage handling for different number of subdomains on the performance of PCG for Diagonal (D) and Global SBS preconditioners

16 subdomains has been used. The subdivision adopted is the one way dissection shown in Figure 1.

Figures 10 and 11 show the performance of the SBS-PCG algorithm implemented on the global and Schur complement levels. Table 1 depicts the computing time required for the subtasks of the Schur complement SBS-PCG implementation. Storage requirements are presented in Tables 2 and 3. A compact storage scheme is used for the stiffness matrices of the subdomains for the global implementation, while a skyline storage is adopted for the Schur complement implementation.

Table 1. Example 3 — Test case 1:
Computing time allocation for the Schur complement SBS-PCG implementation

Number of subdomains		2	3	4	6	8	12
Diagonal	Condensation	67.50	33.40	25.15	18.00	15.10	11.85
	$\mathbf{C}_s$ formulation	—	—	—	—	—	—
	PCG iterations	16.50	22.60	31.20	48.50	69.15	110.85
SBS-1st order	Condensation	67.50	33.40	25.15	18.00	15.10	11.85
	$\mathbf{C}_s$ formulation	0.40	0.50	0.60	0.85	1.10	1.60
	PCG iterations	10.50	14.85	19.85	31.40	43.55	73.00
SBS-2nd order	Condensation	67.50	33.40	25.15	18.00	15.10	11.85
	$\mathbf{C}_s$ formulation	22.50	34.70	31.85	41.30	52.25	74.65
	PCG iterations	9.95	13.95	18.75	29.00	40.60	66.55

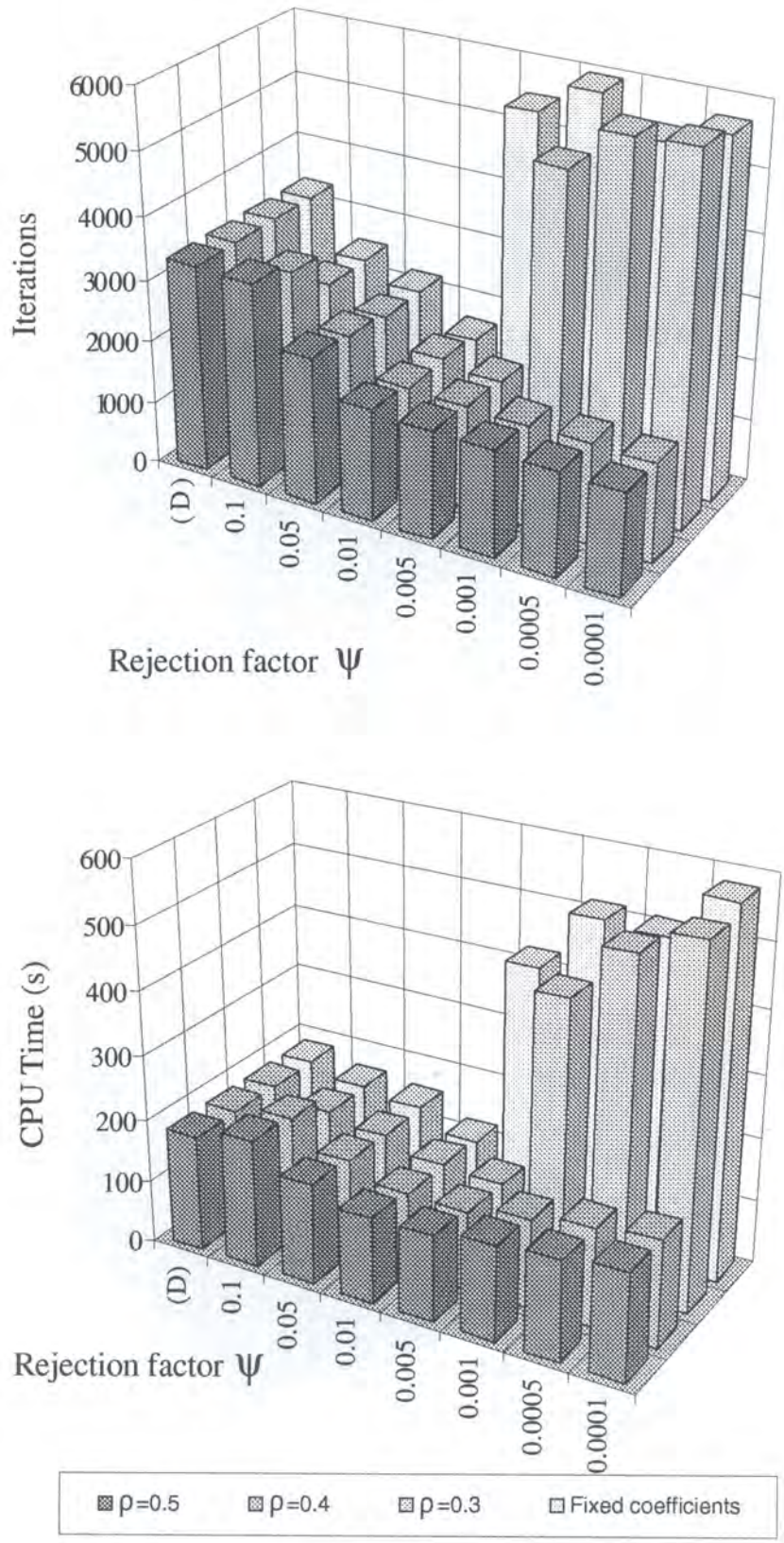

Fig. 4. Example 1: Performance of Diagonal (D) and Global SBS preconditioners for different rejection factors ψ and reduction factors ρ of the Gershwin estimator b. Divisions into 8 subdomains

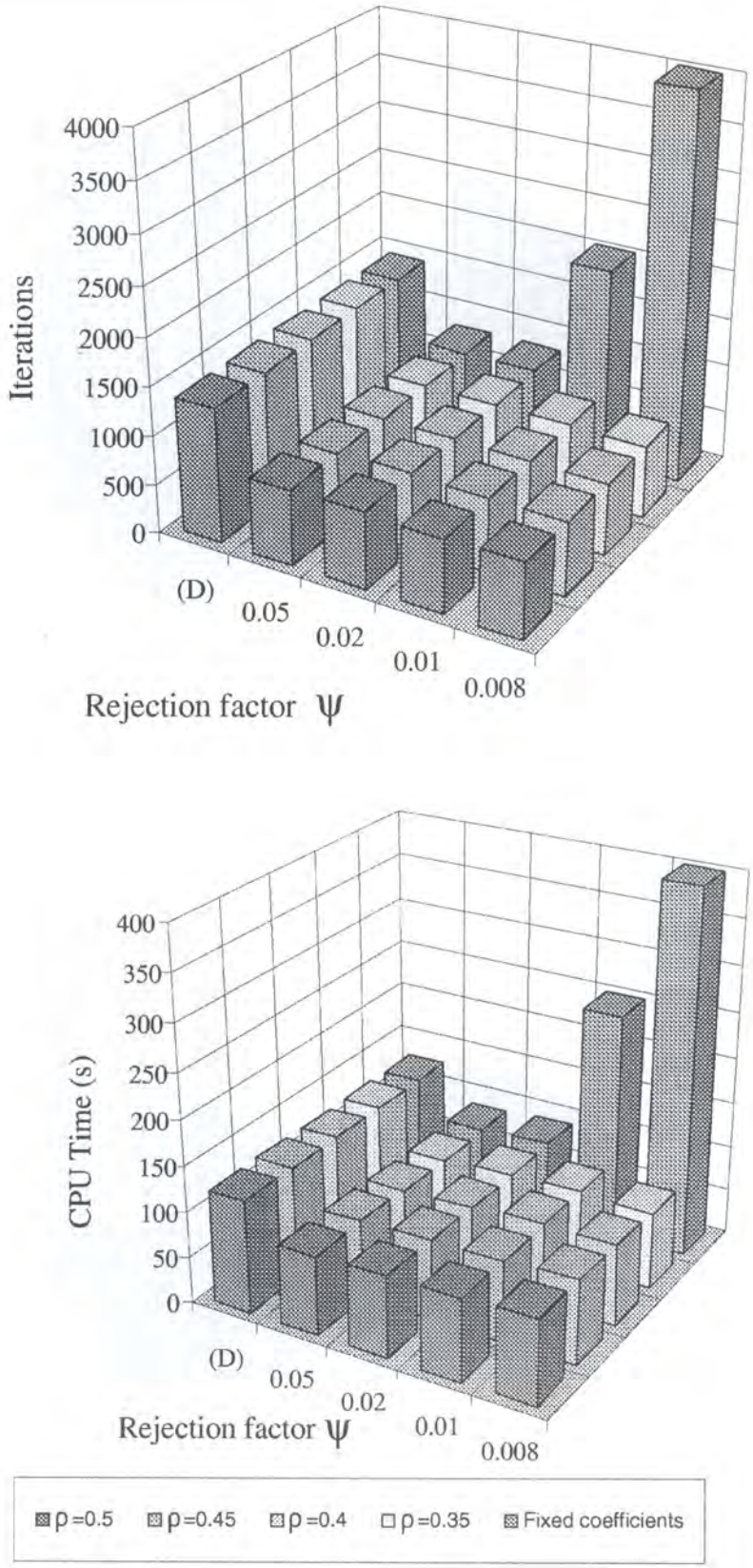

Fig. 5. Example 1: Performance of Diagonal (D) and Global SBS preconditioners for different rejection factors ψ and reduction factors ρ of the Gershwin estimator b. Divisions into 8 subdomains

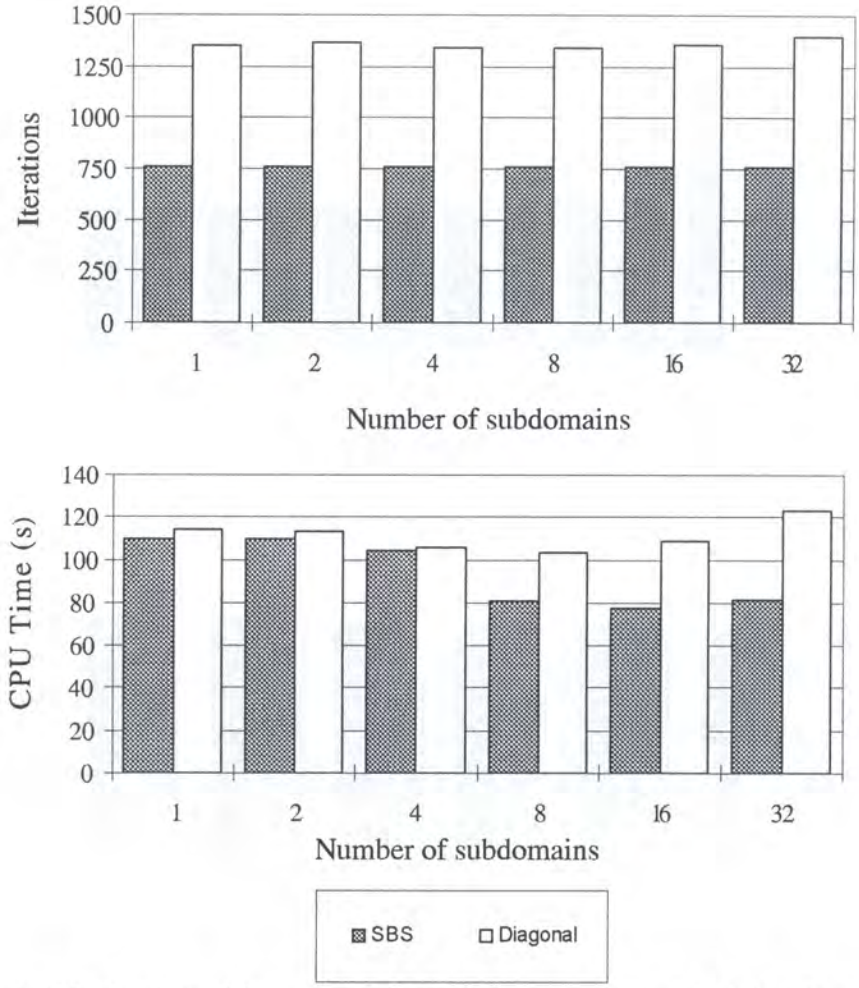

Fig. 6. Example 2: Performance of Diagonal and Global SBS preconditioners for different number of subdomains

Table 2. Test 1: Storage requirements for the global SBS-PCG implementation in 4-byte words

Number of subdomains	Total coefficient matrix storage (compact scheme)	Diagonal Preconditioning	SBS Preconditioning (1st order)	SBS Preconditioning (2nd order)
1	409,676	11,468	50,159	53,238
2	413,076	11,468	50,601	53,984
3	416,090	11,468	51,600	55,207
4	419,512	11,468	52,599	56,431
6	426,356	11,468	54,597	58,879
8	433,208	11,468	56,595	61,327
12	446,936	11,468	60,591	66,223

Figures 12 and 13 illustrate the performance of the parallel implementation of the SBS-PCG method. A comparison between global SBS and Schur complement formulation is depicted. More detailed computing times are depicted in Table 4, while storage requirements are shown in Tables 5 and 6. Finally, Figures 14 and 15 show the speedup factors achieved for different implementations when dedicated or multi-user computing modes are employed and the number of subdomains are kept fixed independently of the number of processors.

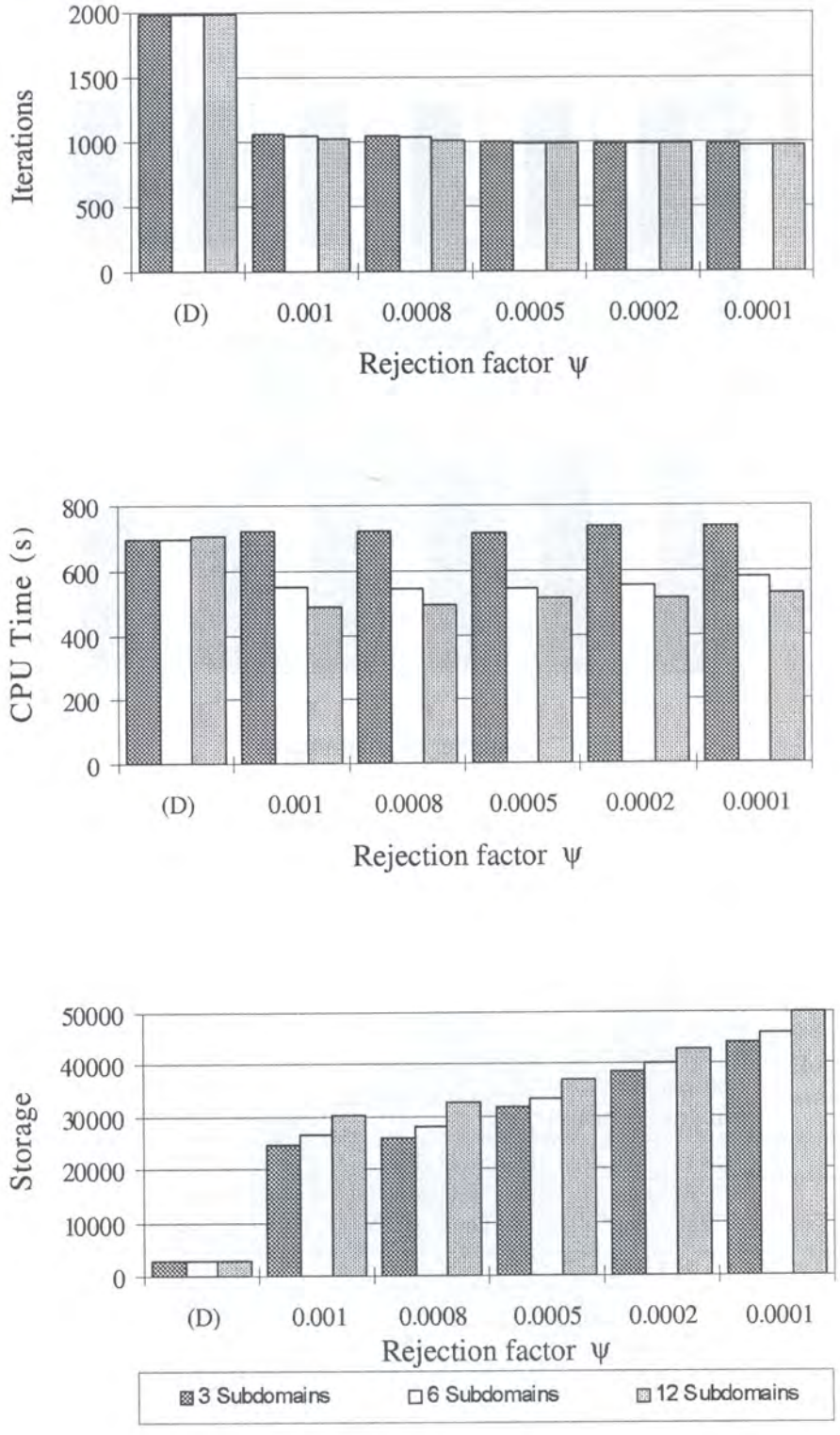

Fig. 7. Example 3 — Test case 1: Performance of Diagonal (D) and Global SBS preconditioners for different rejection factors ψ ($\rho = 0.50$). Divisions into 3, 6, 12 subdomains (Storage refers to non-zero terms of the preconditioner. Stiffness matrix non-zero terms: 208,045, 213,468 and 223,468 respectively

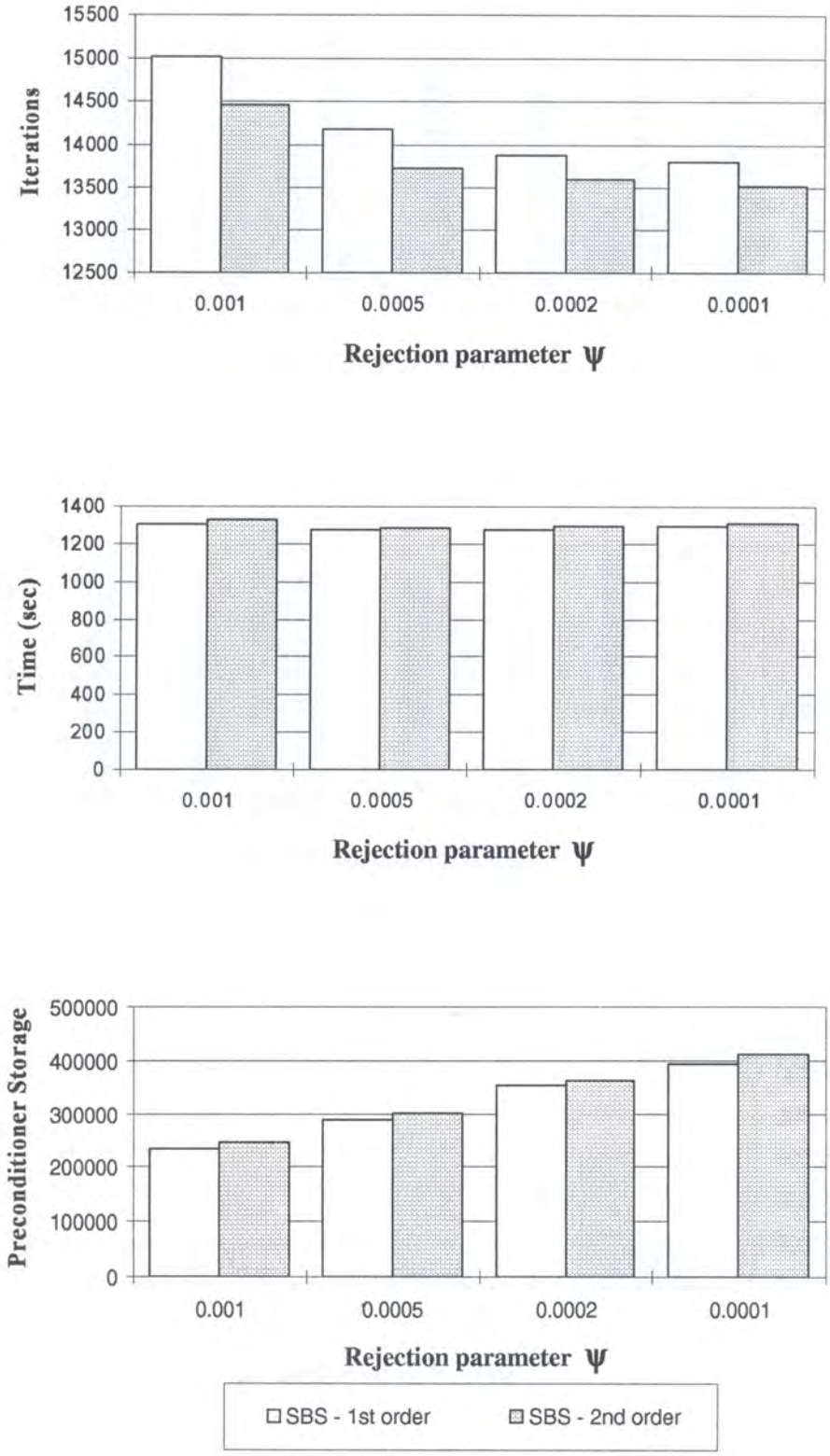

Fig. 8. Example 3 — Test case 1: Performance of Global SBS-PCG implementations for different rejection factors ψ

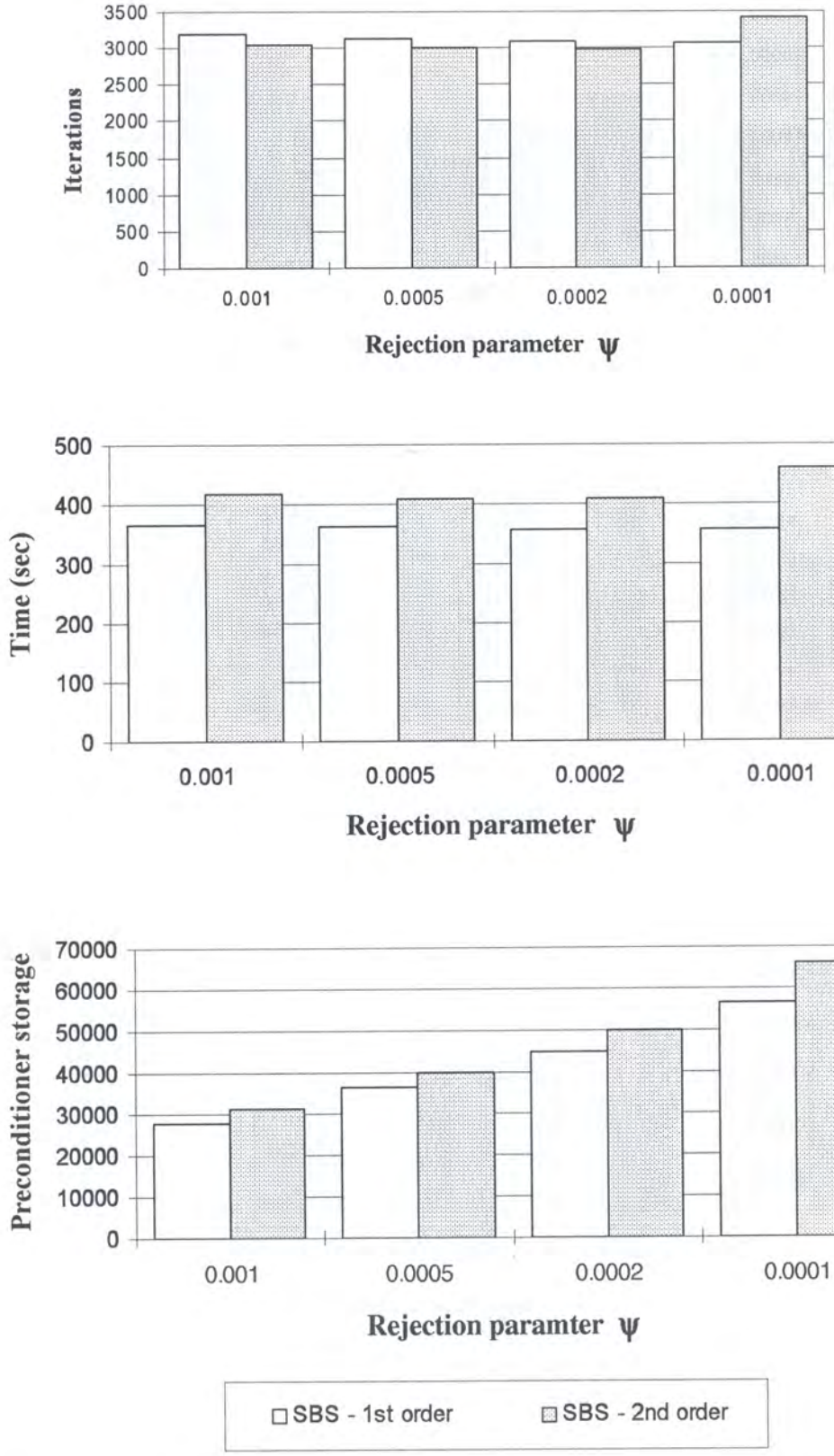

Fig. 9. Example 3 — Test case 2: Performance of Schur complement SBS-PCG implementations for different rejection factors ψ

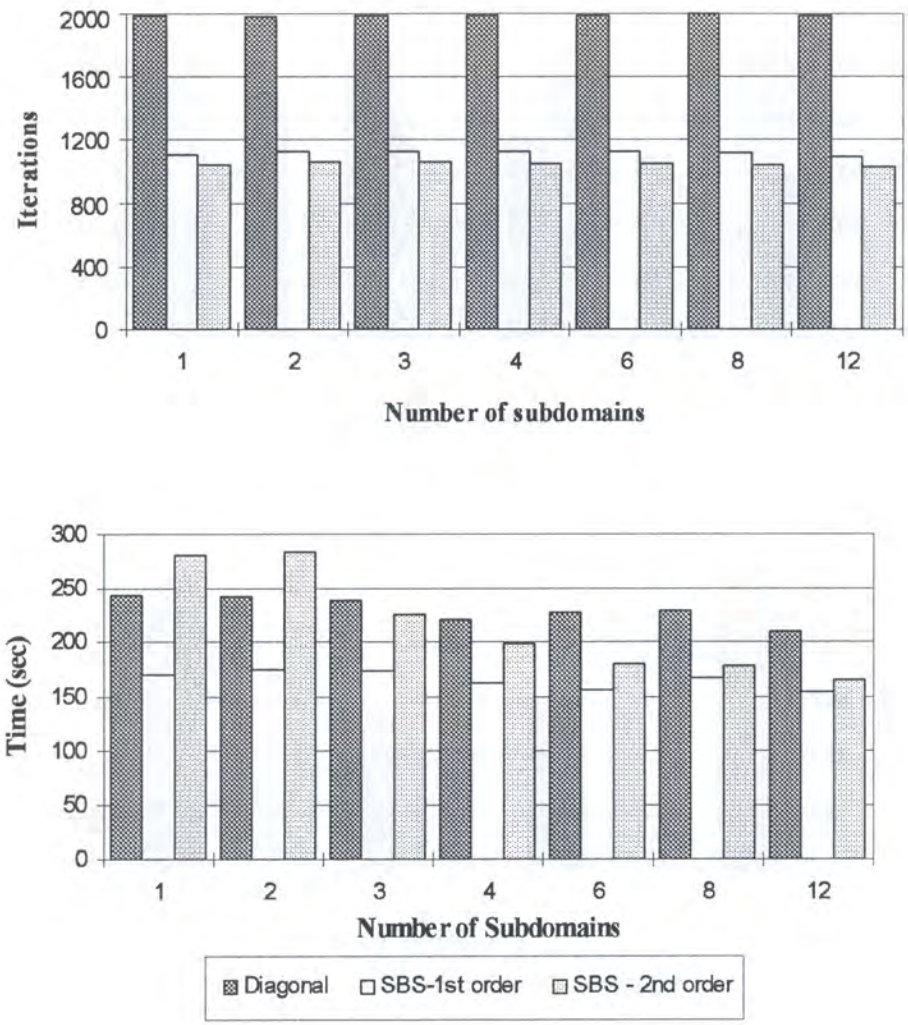

Fig. 10. Example 3 — Test case 2: Performance of Global SBS-PCG implementations for different number of subdomains

Table 3. Test 1: Storage requirements for the Schur complement SBS-PCG implementation in 4-byte words

Number of subdomains	Total Schur complement storage (skyline scheme)	Diagonal Preconditioning	SBS Preconditioning (1st order)	SBS Preconditioning (2nd order)
2	159,636	888	3,210	3,504
3	213,006	1,118	4,450	4,989
4	266,952	1,348	5,678	6,455
6	375,420	1,808	8,076	9,367
8	484,176	2,268	10,455	12,351
12	701,976	3,188	15,577	19,154

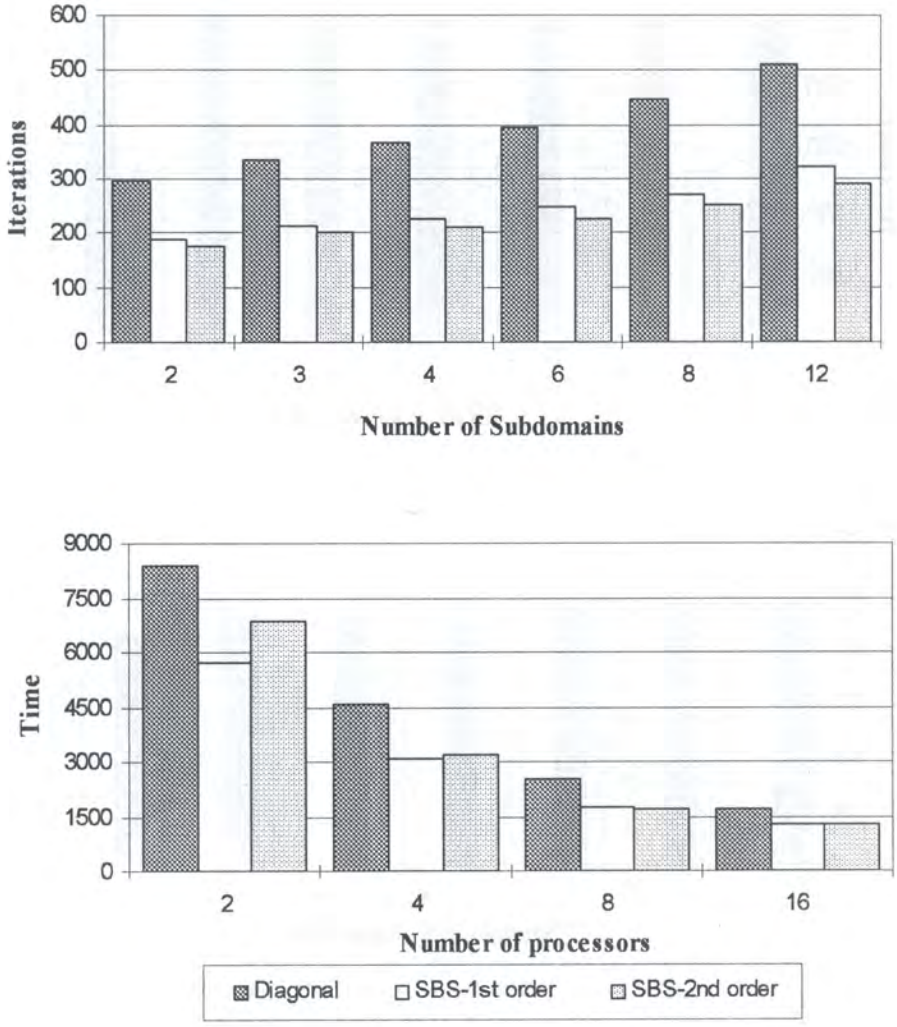

Fig. 11. Example 3 — Test case 2: Performance of Schur complement SBS-PCG implementations for different number of subdomains

Table 4. Example 3 — Test case 2:
CPU time allocation for the Schur complement SBS-PCG implementation

Number of subdomains		2	4	8	16
Diagonal	Condensation	450.00	135.00	43.00	16.00
	$\mathbf{C}_s$ formulation	—	—	—	—
	PCG iterations	8.90	124.80	240.00	525.00
SBS-1st order	Condensation	450.00	135.00	43.00	16.00
	$\mathbf{C}_s$ formulation	0.10	0.50	0.55	0.65
	PCG iterations	5.90	78.30	151.50	340.50
SBS-2nd order	Condensation	450.00	135.00	43.00	16.00
	$\mathbf{C}_s$ formulation	6.45	62.45	62.50	66.00
	PCG iterations	5.70	72.50	145.00	328.90

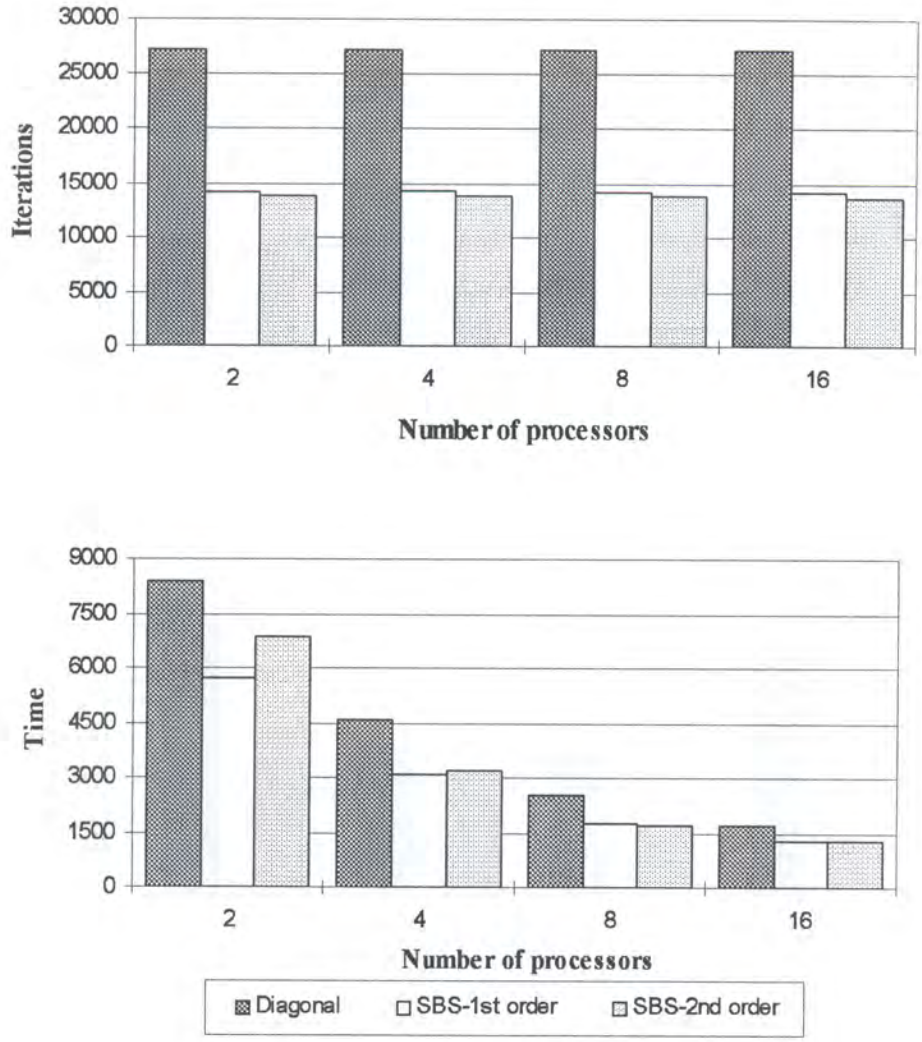

Fig. 12. Example 3 — Test case 2: Performance of Global SBS-PCG implementations for different number of subdomains

Table 5. Example 3 — Test case 2:
Storage requirements for the global SBS-PCG implementation in 4-byte words

Number of subdomains	Total coefficient matrix storage (compact scheme)	Diagonal Preconditioning	SBS Preconditioning (1st order)	SBS Preconditioning (2nd order)
2	1,710,808	45,980	260,436	264,151
4	1,722,440	45,980	264,744	269,397
8	1,750,392	45,980	273,360	279,889
16	1,806,248	45,980	290,592	300,873

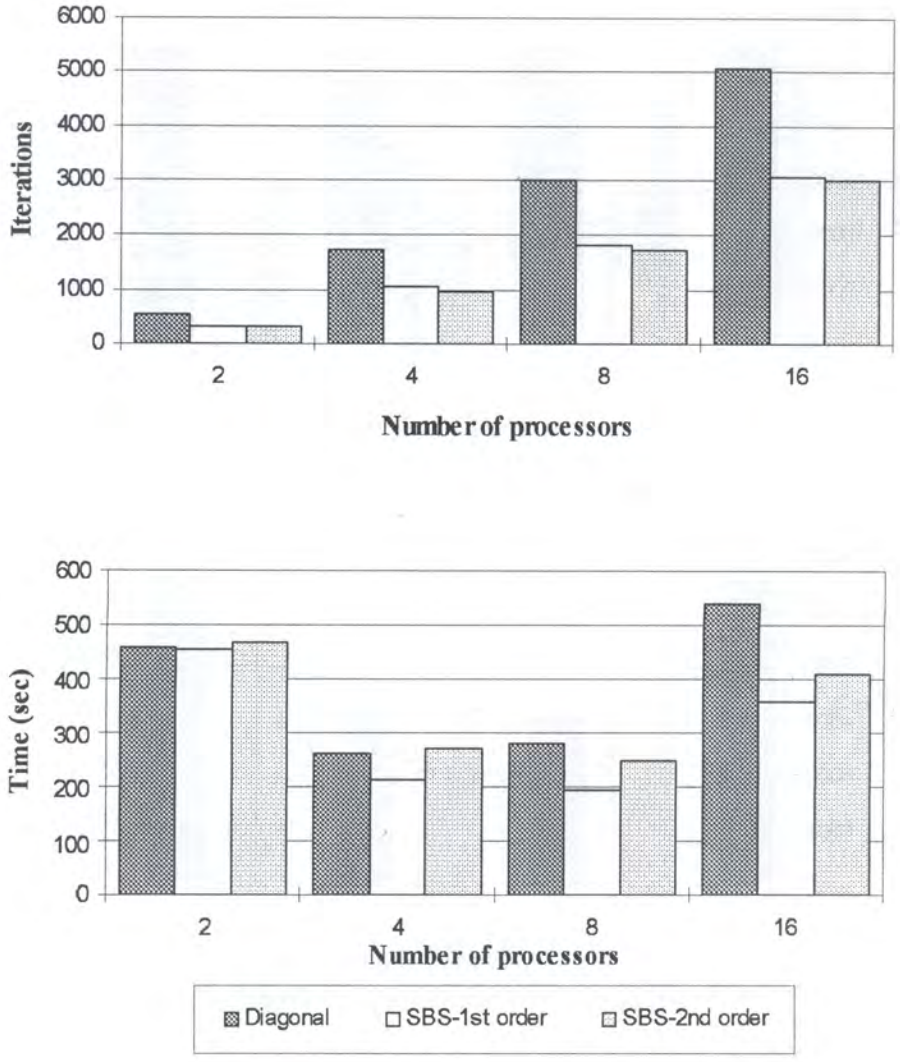

Fig. 13. Example 3 — Test case 2: Performance of Schur complement SBS-PCG implementations for different number of subdomains

Table 6. Example 3 — Test case 2: Storage requirements for the Schur complement SBS-PCG implementation in 4-byte words

Number of subdomains	Total Schur complement storage (skyline scheme)	Diagonal Preconditioning	SBS Preconditioning (1st order)	SBS Preconditioning (2nd order)
2	56,882	476	2,920	4,682
4	283,934	1,428	11,138	9,826
8	738,038	3,332	22,668	25,685
16	1,646,246	7,140	56,346	40,091

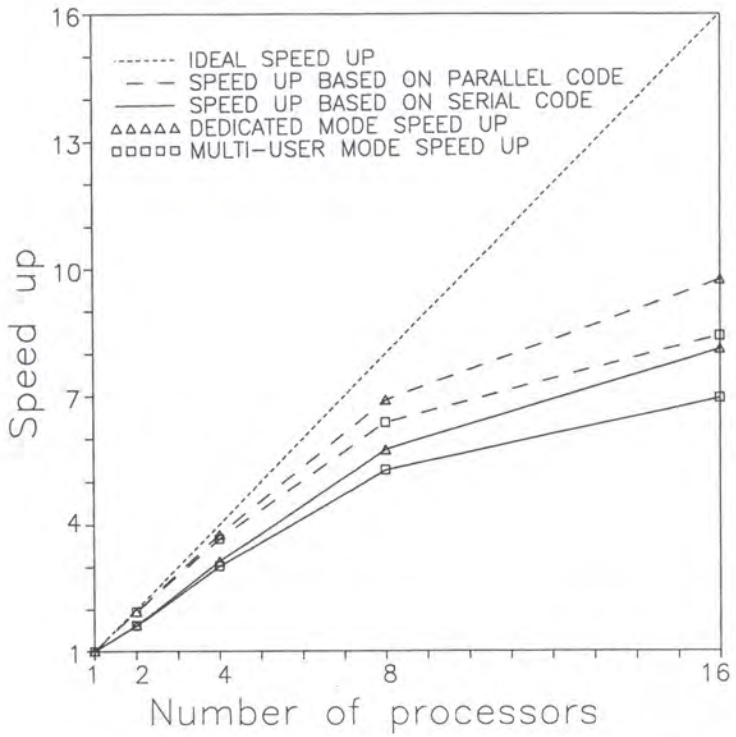

Fig. 14. Example 3 — Test case 2: Speed up factors of Schur complement SBS-PCG implementations for different number of subdomains

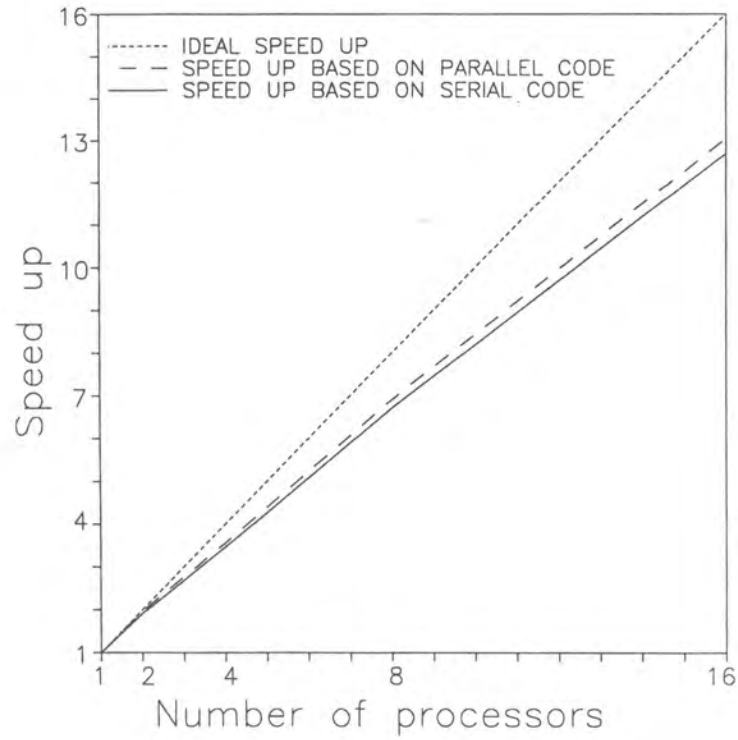

Fig. 15. Example 3 — Test case 2: Speed up factors of global SBS-PCG implementations for different number of subdomains

7. Concluding remarks

Based on the results of the numerical examples presented the following remarks may be drawn:

1. The efficiency of the block type diagonal preconditioning is severely affected by the size of the block. Despite an improvement on the convergence rate of the PCG method, with increasing block sizes, the handling of large preconditioning matrices imposes an overhead on the computing time per iteration and increases substantially the storage requirements. Small block sizes, on the other hand, appear not to improve the convergence properties of the method. For these reasons the diagonal form $\mathbf{B} = \mathbf{D}$ is adopted for the additive polynomial preconditioning proposed.

2. The fixed coefficients in the truncated Neumann expansion are more sensitive on the rejection factor ψ, which determines the non-zero terms to be retained in the preconditioning matrix, than the corresponding coefficients obtained by the least squares minimization procedure and the Gershgorin upper bound estimate. It was also found that the use of preconditioner with fixed coefficients did not achieve convergence for the shell examples.

3. The two parameters that affect the quality of the polynomial preconditioning proposed is the rejection factor ψ and the reduction factor ρ applied to the Gershgorin estimate. The optimum values of ψ were found in the range of 0.05 to 0.001, while the optimum value for ρ was found around 0.5.

4. The quality of the polynomial preconditioner is also affected by the number of terms retained in the truncated Neumann expansion. It was found that although the number of PCG iterations are indifferent to the first or second order expansion, the storage requirements and CPU time per iteration are in favour of the first order expansion.

5. The use of single precision arithmetic for storing and handling the preconditioning matrices gives a substantial improvement on the storage requirements without affecting the convergence properties of the PCG method.

6. The improvement achieved with the proposed polynomial type preconditioning over the diagonal preconditioning depends on the ellipticity of the problem. For ill-conditioned problems the superiority of the polynomial preconditioning is more evident. The number of iterations are almost constantly dropped to half, the total CPU time reduction ranges between 65% and 80%, while the additional storage demands are insignificant.

7. The two formulations presented with regard to the treatment of the coefficient matrix, i.e. on the global level or on the Schur complement level, appear to have their own merits in different implementations. For large number of processors the global SBS formulation appears to be more promising, since its numerical scalability is optimum because the convergence properties of the PCG are not affected by the number of subdomains, while its parallel scalability is very satisfactory. The Schur complement SBS formulation, on the other hand, is more competitive when implemented in a coarse-grained environment. The Schur complement formulation performs better when the size of the interface is kept to a minimum and the internal problems are large.

8. The performance of the methods is also affected by the computational environment they are implemented. It was found that when operated on a dedicated environment the speedup factors achieved were up to 30% higher than when operated on a multi-user mode.

ACKNOWLEDGEMENTS

This work has been partially supported by HC&M/9203390 of the European Union and by PENED/91ED105, PAVE 92 BE297 research projects of the General Secretariat of Science and Technology of Greece.

REFERENCES

[1] R.L. Fox, E.L. Stanton. Developments in structural analysis by direct energy minimization. *AIAA J.*, **6**: 1036–1042, 1968.

[2] I. Fried. More on gradient iterative methods in finite element analysis. AIAA J., **7**: 565–567, 1969.

[3] T.J.R. Hughes, J. Winget, I. Levit, T. Tezduyar. New alternating direction procedures in finite element analysis based upon EBE approximate factorizations. In: S.N. Atluri, N. Perrone, eds., *Computer Methods in Nonlinear Solids and Structural Mechanics*, AMD Vol. 4, 75–109. ASME, New York, 1983.

[4] B. Nour-Omid, B. N. Parlett. Element preconditioning using splitting techniques. *SIAM J. Sci. Stat. Comput.*, **6**: 761–770, 1985.

[5] M. Papadrakakis, Solving large-scale linear problems in solid and structural mechanics. In: M. Papadrakakis, ed., *Solving Large Problems in Mechanics*, pp. 1–37. J. Wiley, 1993.

[6] M. Papadrakakis, N. Bitoulas, K. Hatjikonstantinou. An effective superelement-by-superelement method for large finite element computations. *Computing Systems in Engineering*, **2**: 535–540, 1991.

[7] O.G. Johnson, C.A. Michelli, G. Paul. Polynomial preconditioners for conjugate gradient calculations. *SIAM J. Numer. Anal.*, **20**: 362–376, 1983.

[8] Y. Saad. Practical use of polynomial preconditionings for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.*, **6**: 865–881, 1985.

[9] T.Y. Han, J.F. Abel. Substructure condensation using modified Cholesky decomposition. *Int. J. Num. Meth. Eng.*, **20**: 1959–1964, 1984.

[10] P. Concus, G.H. Golub, G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.*, **6**: 220–252, 1985.

[11] A. Jennings, G.M. Malik. The solution of sparse linear equations by the conjugate gradient method. *Int. J. Num. Meth. Eng.*, **12**: 141–158, 1978.