Technische Universität Chemnitz-Zwickau

DFG-Forschergruppe "SPC" · Fakultät für Mathematik

Matthias Pester Sergej Rjasanow

A Parallel Preconditioned Iterative Realization of the Panel Method in 3D

Preprint-Reihe der Chemnitzer DFG-Forschergruppe
"Scientific Parallel Computing"

SPC 94_18 **November 1994**

A Parallel Preconditioned Iterative Realization of the Panel Method in 3D

Matthias Pester

Technical University of Chemnitz-Zwickau, Faculty of Mathematics, D-09107 Chemnitz, Germany

and

Sergej Rjasanow

University of Kaiserslautern, Dept. of Mathematics, P.O.B. 3049, D-67653 Kaiserslautern, Germany

The parallel version of precondition iterative techniques is developed for matrices arising from the panel boundary element method for three-dimensional simple connected domains with Dirichlet boundary conditions. Results were obtained on an nCube-2 parallel computer showing that preconditoned iterative methods are very well suited also in three-dimensional case for implementation on a MIMD computer and that they are much more efficient than usual direct solution techniques.

KEY WORDS boundary value problem, boundary element method, preconditioning, iterative method, Fast Fourier Transform, parallel algorithm

1. Introduction

In this paper we consider a numerical solution of the three-dimensional Dirichlet boundary value problem for the Laplace equation by the panel method.

This method leads to an algebraic system of linear equations with a full dense, large order and, in general, nonsymmetric matrix [1], [26]. The generation of the Boundary Element Method (BEM) matrix $A \in \mathbb{R}^{N\times N}$ generally can be realized very efficiently on a MIMD computer ([3], [16]) using $O(N^2)$ arithmetical operations and therefore the major remaining problem is to construct an efficient solution strategy.

Usually, the Gaussian elimination algorithm is used for the numerical solution of such systems leading to a number $O(N^3)$ of arithmetical operations.

Fortunately, there are some special classes of three-dimensional domains leading to matrices with some additional properties, by using also special discretization techniques, of course (see [10], [11], [14], [19]).

It was shown in [14] that the Galerkin BEM on a rotational domain leads to a so called *circulant-block matrix*. The theory and some effective numerical algorithms with circulant-block matrices are presented in [19]. The solution of a system of linear equations involving a circulant-block matrix consists of an $O(N^2)$ amount of preparation work independent of the right-hand side of the system, and of the solution itself leading to $O(N^{3/2}\ln N)$ arithmetical operations. Since the usual matrix-vector multiplication requires $O(N^2)$ arithmetical operations, the circulant-block matrices can be used as a preconditioner in some iterative solution procedure. Iterative algorithms for full dense matrices, where each step of iteration requires one or two matrix-vector multiplications, some scalar products, some vector additions and the solution of the preconditioning system, involve a very high level of parallelism only if the preconditioning technique fits to the parallel realization.

In this paper we shall show that the preconditioning based on the algorithms with circulant-block matrices fullfills all requirements to the efficient parallel procedure and can be used together with some iterative schemes, e. g. usual Gradient Method (GM) or modern BiCGSTAB method [23] for the numerical solution of the BEM systems.

The paper is organized as follows: In Section 2 we describe the panel method for a three-dimensional boundary integral equation. Section 3 deals with the matrices arising from the rotational domains which have a special circulant-block structure. The iterative methods and a complete parallel solution of the problem are discussed in Section 4. Finally, we present the numerical results and draw some conclusions.

2. Panel method for the boundary integral equation

In this paper we consider the integral equation of the first kind for an unknown function $v(x)$:

$$\frac{1}{4\pi}\int_\Gamma \frac{v(x)}{|x-y|}\mathrm{d}F_x = f(y), \qquad x, y \in \Gamma \subset I\!R^3 \tag{2.1}$$

Here, Γ is the sufficiently smooth boundary of a three-dimensional bounded, simply connected domain Ω. Let Γ be given by the parametric representation

$$\Gamma = \left\{x : x \in I\!R^3, x = x(t,z), 0 \le t < 1, 0 \le z \le 1\right\}, \tag{2.2}$$

where the function $x(t,z)$ is 1-periodical in t:

$$x(t,z) = x(t+1,z)$$

The main properties of the operator $\mathcal{A}$ with

$$(\mathcal{A}v)(y) = \frac{1}{4\pi}\int_\Gamma \frac{v(x)}{|x-y|}\mathrm{d}F_x$$

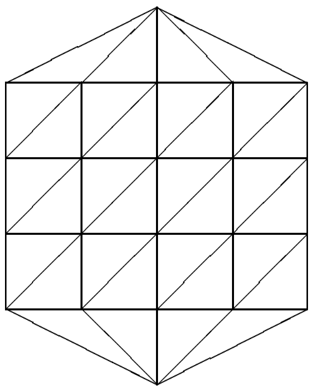

Figure 1. Discretization of the parameter domain

are (see [8], [15], [26]):

1. $\mathcal{A}$ defines a continuous bijective mapping

$$\mathcal{A} : I\!H^{-1/2}(\Gamma) \rightarrow I\!H^{1/2}(\Gamma)$$

2. the equation (2.1) has a unique solution $v(x) \in I\!H^{-1/2}(\Gamma)$ for all $f(y) \in I\!H^{1/2}$,

where $I\!H^s(\Gamma)$ is the Sobolev space on the boundary Γ. One of the most simple methods for the numerical solution of the equation (2.1) bases on the following two discretization ideas:

1. The boundary Γ will be replaced by the union Γ_h of the plane triangles (panels) Γ_j:

$$\Gamma \sim \Gamma_h = \bigcup_{j=1}^{N} \Gamma_j \tag{2.3}$$

2. The unknown function $v(x)$ will be approximated by a piecewise constant function $v_h(x)$:

$$v_h(x) = v_j \quad \forall x \in \Gamma_j \tag{2.4}$$

In order to define the discretization (2.3) of the boundary Γ we divide the parameter domain $[0,1) \times [0,1)$ into triangles using the nodes:

$$\{(t_k, z_l) = (h_t(k-1), h_z(l-1))\,,\ k = 1, \ldots, n,\ l = 1, \ldots, m+1\} \tag{2.5}$$

where $h_t = 1/n$ and $h_z = 1/m$ (see Figure 1). The total number of triangles used by this discretization is:

$$N = n + 2(m-2)n + n = 2n(m-1).$$

The panel method for the equation (2.1) leads to:
Find v_h of the form (2.4) such that the collocation equations

$$\sum_{j=1}^{N} v_j \int_{\Gamma_j} \frac{\mathrm{d}F_x}{|x-y_i|} = f(y_i) \tag{2.6}$$

are satisfied for all $i = 1, \ldots, N$.
The points y_i are the collocation points and can be chosen (e. g.) as:

$$y_i = \frac{1}{3}\left(x_i^{(1)} + x_i^{(2)} + x_i^{(3)}\right), \quad i = 1, \ldots, N,$$

where $x_i^{(1)}$, $x_i^{(2)}$ and $x_i^{(3)}$ are images on the boundary Γ of the corresponding points in the parameter domain by the discretization (2.5). The equations (2.6) can be replaced by the algebraic system

$$Av = b,\ A \in \mathbb{R}^{N\times N},\ v, b \in \mathbb{R}^N \tag{2.7}$$

with

$$a_{ij} = \int_{\Gamma_j} \frac{\mathrm{d}F_x}{|x-y_i|}, \quad b_i = f(y_i), \quad i, j = 1, \ldots, N. \tag{2.8}$$

Generally, the Matrix A of the system (2.7) is a non-symmetric, full dense matrix without any additional properties which can be used for some effective solution techniques. All elements a_{ij} in (2.8) can be computed analytically.

3. Panel method on a rotational domain

Now, we shall prove that the matrix A arising from a rotational domain has a circulant-block structure. Let Γ be given by the parametric representation

$$\Gamma = \left\{ x : x \in \mathbb{R}^3,\ x = \begin{pmatrix} R(z)\cos 2\pi t \\ R(z)\sin 2\pi t \\ z \end{pmatrix},\ 0 \le t < 1,\ 0 \le z \le 1 \right\} \tag{3.9}$$

where $R(0) = R(1) = 0,\ R(z) > 0,\ z \in (0,1)$. The numbering of the panels has an evident influence on the structure of the matrix. In order to get a circulant-block structure of the system matrix we start numbering at the panels of the strip

$$S_1 = \{(t,z),\ 0 \le t < 1,\ 0 \le z \le h_z\}$$

where we can find only one kind of triangles.
In the strips

$$S_l = \{(t,z),\ 0 \le t < 1,\ z_l \le z \le z_{l+1}\}, \quad l = 2, \ldots, m-1$$

there are two kinds of triangles (see Figure 1). Finally, we number the panels in the last strip

$$S_m = \{(t,z),\ 0 \le t < 1,\ 1 - h_z \le z \le 1\}\,.$$

Therefore the numbering of the panels Γ_j is the following:

$1 \le j \le n$ – panels in the strip S_1 for $t_k, k = 1, \ldots, n$;
$(2l-3)n + 1 \le j \le (2l-2)n$ – panels of the *"first kind"* in the strips $S_l, l = 2, \ldots, m-1$;
$(2l-2)n + 1 \le j \le (2l-1)n$ – panels of the *"second kind"* in the strips $S_l, l = 2, \ldots, m-1$;
$(2m-3)n + 1 \le j \le (2m-2)n$ – panels in the strip S_m.

The number of panels and hence the number of unknowns becomes

$$N = 2(m-1)n.$$

Lemma 3.1. *The matrix of the system (2.7) for the rotational domain (3.9) using the above numbering of unknowns has the circulant-block structure:*

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1,2(m-1)} \\ \vdots & & \vdots \\ A_{2(m-1),1} & \cdots & A_{2(m-1),2(m-1)} \end{pmatrix}, \qquad A_{ij} \in \mathbb{R}^{n \times n},$$

$$A_{ij} \text{ – circulant},\ i,j = 1, \ldots, 2(m-1).$$

Proof Let $a_{(i_1,i_2)(j_1,j_2)}$ be an element (i_1, j_1) of the matrix A_{i_2,j_2}, $1 \le i_2, j_2 \le 2(m-1)$; $1 \le i_1, j_1 \le n$.
We have to prove that

$$a_{(i_1+1,i_2)(j_1+1,j_2)} = a_{(i_1,i_2)(j_1,j_2)}, \quad i_1, j_1 = 1, \ldots, n-1; \tag{3.10}$$
$$a_{(i_1+1,i_2)(1,j_2)} = a_{(i_1,i_2)(n,j_2)}, \quad i_1 = 1, \ldots, n-1; \tag{3.11}$$

for all $i_2, j_2 = 1, \ldots, 2(m-1)$.

An element $a_{(i_1,j_1)(i_2,j_2)}$ is defined by (2.8):

$$a_{ij} = \int_{\Gamma_j} \frac{\mathrm{d}F_x}{|x - y_i|}, \quad b_i = f\,(y_i)\,, \quad \text{where } j = (j_2 - 1)n + j_1,\ i = (i_2 - 1)n + i_1.$$

Now we consider the orthogonal matrix $Q(\varphi)$

$$Q(\varphi) = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad Q(\varphi)Q(\varphi)^{\top} = I$$

for $\varphi = 2\pi/n$. Since

$$\left\{ y : \ y = Q\left(\frac{2\pi}{n}\right) x,\ x \in \Gamma_j \right\} = \begin{cases} \Gamma_{j+1}, & j_1 < n, \\ \Gamma_{j+1-n}, & j_1 = n; \end{cases}$$

$$Q\left(\frac{2\pi}{n}\right) y_i = \begin{cases} y_{i+1}, & i_1 < n, \\ y_{i+1-n}, & i_1 = n; \end{cases}$$

$$|x - y_i| = \left| Q\left(\frac{2\pi}{n}\right)(x - y_i) \right|$$

we obtain (3.10) and (3.11) for the corresponding i_1 and j_1. Therefore, each matrix $A_{(i_1,i_2)(j_1,j_2)}$ is a circulant. ∎

The most important property of the circulant-block matrices is that a system of linear equations with such a kind of matrix can be solved by an amount of arithmetical operations of $O\left(n \cdot m^3\right) = O\left(N^2\right)$ (if $m \sim n$), and therefore such matrices can be used as a preconditioner for more general systems.

We will now give a short description of the properties of circulant-block matrices and formulate the algorithm for the solution of the linear systems. For more detail we refer to [14], [19].

Each circulant matrix C can be written as a polynomial of the simplest nontrivial circulant J:

$$C = C(J) = \sum_{l=1}^{n} c_{1l} J^{l-1}, \quad J = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}. \tag{3.12}$$

The circulant matrices are diagonalizable by the Fourier matrix F:

$$C = n^{-1} F \Lambda F^*, \quad f_{kl} = \omega_k^{l-1}, \quad \omega_k = e^{i\frac{2\pi}{n}(k-1)}, \quad i^2 = -1 \tag{3.13}$$

$$\Lambda = \operatorname{diag}\left(\lambda_1, \ldots, \lambda_n\right) = \operatorname{diag}(Fc)$$

where $\lambda_i, i = 1, \ldots, n$, are the eigenvalues of the matrix C and c denotes its first row $\left(c_{11}, \ldots, c_{1n}\right)^\top$. Hence, each circulant-block matrix $A = A(J)$ may be written in the form (see (3.12)):

$$A = A(J) = \begin{pmatrix} a_{11}(J) & \cdots & a_{1m}(J) \\ \vdots & \ddots & \vdots \\ a_{m1}(J) & \cdots & a_{mm}(J) \end{pmatrix} \tag{3.14}$$

where $a_{kl}(J)$ are the polynomials of degree $n-1$. This property (3.14) gives the following (see [19])

Lemma 3.2. *If all matrices*

$$A\left(\omega_k\right) = \begin{pmatrix} a_{11}\left(\omega_k\right) & \cdots & a_{1m}\left(\omega_k\right) \\ \vdots & \ddots & \vdots \\ a_{m1}\left(\omega_k\right) & \cdots & a_{mm}\left(\omega_k\right) \end{pmatrix} \in \mathbb{C}^{m \times m}$$

are diagonalizable and

$$\begin{aligned} A(\omega_k)\, x_l(\omega_k) &= \lambda_l(\omega_k)\, x_l(\omega_k)\,; \\ A^*(\omega_k)\, y_l(\omega_k) &= \overline{\lambda_l(\omega_k)} y_l(\omega_k)\,; \\ y_l(\omega_k)^* x_j(\omega_k) &= n\delta_{lj}\,, \quad (\delta_{lj} \text{ - Kronecker symbol}) \\ & \quad l = 1, \ldots, m,\ k = 1, \ldots, n, \end{aligned} \tag{3.15}$$

then the matrix $A(J)$ is also diagonalizable and

$$\begin{aligned} A(J)\,(x_l(\omega_k) \otimes f_k) &= \lambda_l(\omega_k)\, x_l(\omega_k) \otimes f_k, \\ A(J)^{\top}\,(y_l(\omega_k) \otimes f_k) &= \overline{\lambda_l(\omega_k)} y_l(\omega_k) \otimes f_k; \\ & \quad l = 1, \ldots, m,\ k = 1, \ldots, n; \end{aligned} \tag{3.16}$$

where $\otimes$ denotes the Kronecker product. ■

The property (3.17) implies:

$$A(J) = \frac{1}{n} \sum_{k,l} \lambda_l(\omega_k)\,(x_l(\omega_k) \otimes f_k)\,(y_l(\omega_k) \otimes f_k)^*\,. \tag{3.17}$$

The solution of the system of linear equations

$$Ay = b, \quad A \in \mathbb{R}^{N \times N},\ y, b \in \mathbb{R}^N \tag{3.18}$$

involving a circulant-block matrix can be obtained now as

$$y = A^{-1}(J)b = \frac{1}{n} \sum_{k,l} \frac{1}{\lambda_l(\omega_k)}\,(x_l(\omega_k) \otimes f_k)\,(y_l(\omega_k) \otimes f_k)^*\, b \tag{3.19}$$

and computed with the help of

Algorithm 1

1. $C := (F^*B)^{\top} \in \mathbb{C}^{n \times m}$
2. $Y := 0$
3. for $l = 1, \ldots, m$

3.1. $D_l := \operatorname{diag}(d_1, \ldots, d_n)\,, \qquad d_k := \dfrac{y_l^*(\omega_k)\, C e_k}{n\lambda_l(\omega_k)}$

3.2. $Y := Y + \operatorname{Re}\left(F D_l X_l^{\top}\right),$

where $B = \left(b_1 \vdots \cdots \vdots b_m\right) \in \mathbb{R}^{n \times m}$ is the right-hand side of (3.18),

$Y = \left(y_1 \vdots \cdots \vdots y_m\right) \in \mathbb{R}^{n \times m}$ is the solution of (3.18),

$X_l = (x_l(\omega_1), \ldots, x_l(\omega_n)) \in \mathbb{C}^{m \times n},\ l = 1, \ldots, m$

and e_k is the k-th column of the identity matrix $I \in \mathbb{R}^{m \times m}$.

We need only $O\left(m^2 n \ln n\right)$ arithmetic operations for the entire alorithm if we use the Fast Fourier Transform (see [2]) twice at steps 1 and 3.2.

The preparation step is the numerical solution of n complete eigenvalue problems (3.15) and requires $O\left(n \cdot m^3\right)$ arithmetic operations. To solve these problems we transform the matrices $A\left(\omega_k\right)$ to Hessenberg form using orthogonal Householder matrices and apply the QR method afterwards (see [20], [5], [12]).

It is necessary to remark at this place that we have to solve the problems (3.15) only once before starting the iterations, if we use a circulant-block matrix as a preconditioner.

4. Parallel iterative solution of the problem

Since in general the matrix of the system (2.7) is non-symmetric we cannot use the Conjugate Gradient Method for the iterative solution. For our numerical tests we use the classical Gradient Method

Algorithm 2

1. $y_0 \in \mathbb{R}^n$;
   $r_0 = Ay_0 - b, \qquad w_0 = B^{-1} r_0$
2. for $k = 0, 1, 2, \ldots$
   $$y_{k+1} = y_k - \alpha_{k+1} w_k, \qquad \alpha_{k+1} = \frac{(Aw_k, r_k)}{(Aw_k, Aw_k)}$$
   $$r_{k+1} = r_k - \alpha_{k+1} A w_k, \qquad w_{k+1} = B^{-1} r_{k+1}$$

and the modern BiCGSTAB (see [23], [24])

Algorithm 3

1. $y_0 \in \mathbb{R}^n$;
   $r_0 = Ay_0 - b, \qquad w_0 = B^{-1} r_0$
   $s_0 = w_0$
2. for $k = 0, 1, 2, \ldots$
   $$u = r_k - \alpha_{k+1} A s_k$$
   $$v = w_k - \alpha_{k+1} B^{-1} A s_k, \qquad \alpha_{k+1} = \frac{(r_0, w_k)}{(r_0, B^{-1} A s_k)}$$
   $$y_{k+1} = y_k - \alpha_{k+1} s_k - \gamma_{k+1} v, \qquad \gamma_{k+1} = \frac{(u, Av)}{(Av, Av)}$$
   $$r_{k+1} = u - \gamma_{k+1} A v, \qquad w_{k+1} = B^{-1} r_k + 1$$
   $$s_{k+1} = w_{k+1} + \beta_{k+1}\left(s_k - \gamma_{k+1} B^{-1} A s_k\right), \quad \beta_{k+1} = \frac{\alpha_{k+1}}{\gamma_{k+1}} \frac{(r_0, w_{k+1})}{(r_0, w_k)}$$

The parallel realization of the problem, once the rotational domain for the preconditioning is chosen, consists of the following steps:

1. computing the matrix A according to (2.8);
2. solution of the eigenvalue problems (3.15) for the preconditioning;

3. iterative solution using the Algorithm 2 or 3 and the Algorithm 1 for the preconditioning step $Bw_{k+1} = r_{k+1}$.

The generating of the matrix A can be done fully parallel if the geometrical information is available for all processors $p = 0, \dots, P-1$, where each of the processors computes a number N/P of rows of the matrix A (Fig. 2).

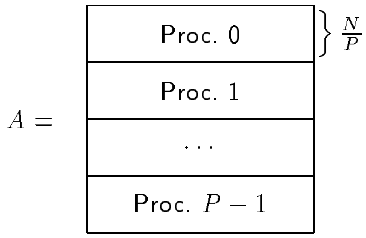

Figure 2. Matrix computation on P processors

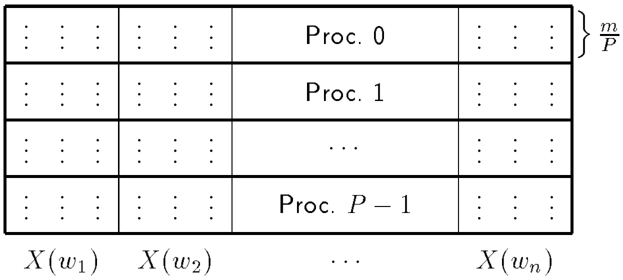

Figure 3. Distribution of the eigenvector matrices

The numerical solution of the eigenvalue problems (3.15) can also be done in parallel. Here each processor has either to solve a number n/P of problems or just to "wait" if $n < P$ (where only the "first" n processors have to solve one problem each). For the parallel realization of the Algorithm 1 it is necessary that each of the processors has a certain part of the matrices of the eigenvectors which are distributed as shown in Figure 3.

For convenience the eigenvalues $\lambda_l(w_k)$ should be stored on each processor. Therefore, the corresponding transfer of the data is necessary in this step.

Within the iterative solution of the system (2.7) one step of the iteration requires one (Algorithm 2) or two (Algorithm 3) matrix-vector multiplications, some vector additions, scalar products and one solution of an appropriate preconditioning system

$$Bw_{k+1} = r_{k+1}.$$

Since the parallel realization of all those operations except the preconditioning is the same as in the case of two-dimensional problems, we refer to [16] and will discuss only the Algorithm 1 here.

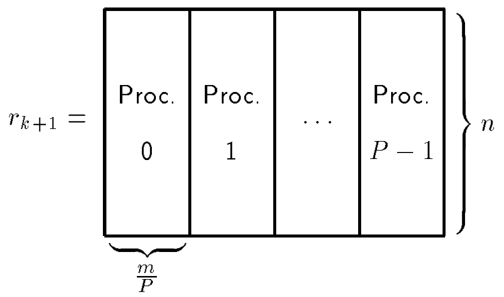

Figure 4. Distribution of the right-hand side

The Algorithm starts with the right-hand side r_{k+1} distributed on the P processors as shown in Figure 4. Each processor computes the corresponding part of the matrix C using the FFT.

In step 3.1. each processor can only compute a part of each scalar product

$$y_l^*(w_k) \cdot Ce_k, \; k = 1, \ldots, n,$$

corresponding to the locally stored m/P components per vector, and therefore a global exchange of data is necessary in this step to form the global sums over the local partial sums. The step 3.2. again can be done completely parallel where each processor has to execute m/P FFT's for the corresponding columns of the matrices $D_l X_l^\top$, $l = 1, \ldots, m$. Our numerical tests show that the above parallel realization of the Algorithm 1 is very well suited for a MIMD-Computer.

5. Numerical tests

For our numerical tests we use the domain given by the parametric representation

$$\Gamma = \left\{ x : x = \begin{pmatrix} R(z)\cos 2\pi t \\ R(z)\sin 2\pi t(2 - 1.5\sin 2\pi t) \\ z \end{pmatrix}, \; 0 \le z \le 1, \; 0 \le t < 1 \right\}$$

$$R(z) = \sqrt{z(1-z)}$$

as shown in Figure 5.

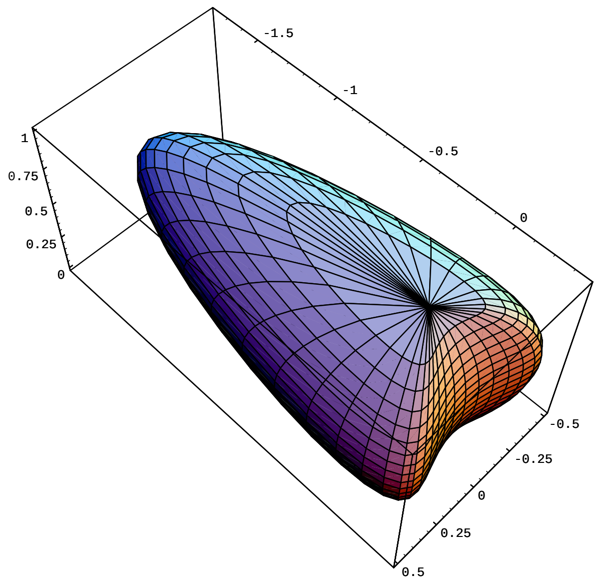

Figure 5. Test boundary Γ

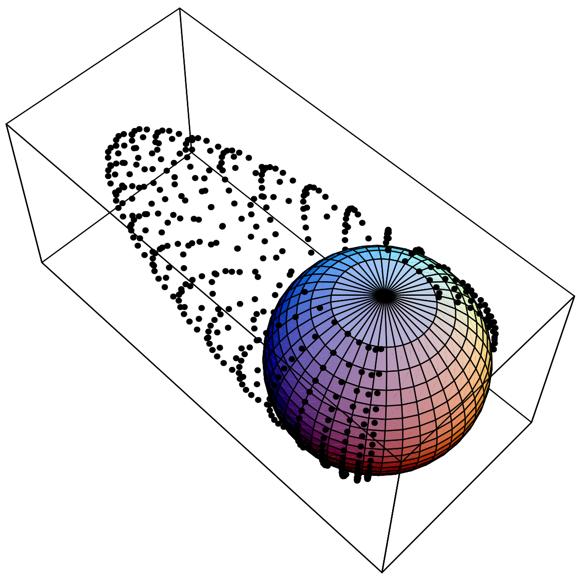

Figure 6. Preconditioning by the spherical surface

For the preconditioning we used the circulant-block matrix arising from the spherical surface

$$\left\{ x : x = \begin{pmatrix} R(z)\cos 2\pi t \\ R(z)\sin 2\pi t \\ z \end{pmatrix}, \; 0 \le z \le 1, \; 0 \le t < 1 \right\}, \quad R(z) = \sqrt{z(1-z)}$$

The tests were performed on a "nCube-2S" parallel computer with up to 64 processors and 8 MByte of local memory. For a different number of processors we get a corresponding maximum size of the problem. So we have a lack of fill-in in the Tables 1, 2, 3, caused by the computer capacity for large problems or by the algorithm itself for small problems and a large number of processors where no parallelization effect can be obtained.

Table 1. Time (in seconds) for generating the matrices

problem size			number of processors						
n	m	N	1	2	4	8	16	32	64
8	5	64	2.8	1.4	0.7	0.35			
16	5	128	11.1	5.6	2.8	1.4			
32	5	256	44.5	22.3	11.1	5.6			
32	9	512	178.1	89.4	44.7	22.4	11.2		
64	9	1024		357.2	178.8	89.5	44.8		
64	17	2048				357.7	178.9	89.6	44.7
128	17	4096						355.5	179.1

Table 2. Number of iterations for Algorithms 2 and 3: (a) preconditioned, (b) simple

problem size			Alg. 2		Alg. 3	
n	m	N	(a)	(b)	(a)	(b)
8	5	64	25	97	8	13
16	5	128	22	145	10	16
32	5	256	30	127	10	17
16	9	256	23	96	10	20
32	9	512	25	202	10	25
64	9	1024	32	395	9	26
32	17	1024	30	233	11	27
64	17	2048	28	617	12	31
32	33	2048	29	340	10	32
128	17	4096	29	747	9	36
64	33	4096	28	871	11	35

Table 3. Time for the solution by Algorithms 2 and 3 on different numbers of processors

problem size			Algorithm 2 – simple iteration						
n	m	N	1	2	4	8	16	32	64
32	5	256	25.7	13.6	7.6	5.0			
16	9	256	19.5	10.4	5.6	3.8	3.3		
32	9	512	160.1	82.9	43.2	24.1	15.5		
64	9	1024		625.2	329.7	172.5	93.3		
32	17	1024		373.8	190.4	99.1	54.9	35.0	
64	17	2048				1001.1	520.6	287.4	
32	33	2048				553.5	287.6	156.9	101.8
128	17	4096						1243.3	690.7
64	33	4096						1449.2	805.7

problem size			Algorithm 2 – preconditioned iteration						
n	m	N	1	2	4	8	16	32	64
32	5	256	10.1	5.7	3.6	2.7			
16	9	256	10.3	5.7	3.5	2.7	2.5		
32	9	512	32.9	17.5	9.8	6.4	4.9		
64	9	1024		70.7	37.8	21.8	14.4		
32	17	1024		79.2	42.1	24.3	16.0	12.7	
64	17	2048				67.7	39.8	26.8	
32	33	2048				83.5	48.42	32.5	26.1
128	17	4096						76.4	
64	33	4096						81.9	56.2

problem size			Algorithm 3 – simple iteration						
n	m	N	1	2	4	8	16	32	64
32	5	256	7.1	3.8	2.2	1.4			
16	9	256	8.2	4.4	2.5	1.6	1.3		
32	9	512	40.1	16.6	10.0	5.1	3.3	2.8	
64	9	1024		83.7	42.8	20.7	12.5	7.7	6.7
32	17	1024		86.8	49.2	24.9	12.9	8.3	6.9
64	17	2048				101.3	53.0	29.4	18.7
32	33	2048				104.5	47.9	30.3	16.9
128	17	4096						122.5	71.5
64	33	4096						119.1	67.8

problem size			Algorithm 3 – preconditioned iteration						
n	m	N	1	2	4	8	16	32	64
32	5	256	6.7	3.8	2.4	1.8			
16	9	256	8.9	4.9	3.1	2.3	2.1		
32	9	512	26.1	13.8	7.8	5.1	3.9		
64	9	1024		39.6	21.2	12.3	8.2		
32	17	1024		57.4	30.7	17.8	11.8	9.3	
64	17	2048				56.8	33.6	22.3	
32	33	2048				57.1	35.6	22.7	18.2
128	17	4096						47.7	
64	33	4096						64.1	44.2

Remark 1: It is necessary to remark that the iterative solution method presented here is not of optimal (in usual sense) order. The number of arithmetical operations required is of the capital order $O\left(h^{-4}\right)$ $(h \sim 1/n \sim 1/m)$. The main work remains in the multiplication of the system matrix with a vector, which should be done once for the Algorithm 2 and twice for the Algorithm 3 per iteration step. The solution of the eigenvalue problems (3.15) requires also $O\left(h^{-4}\right)$ arithmetical operations. This work should be done only in the preparation phase. The preconditioning system is solved via Algorithm 1 ($O(h^{-3} \ln h^{-1})$ operations). The decrease of the arithmetical work for the matrix-vector multiplication, investigated in [7] can also be used for our algorithms.

Remark 2: Here we consider only a model problem, the domain decomposition techniques ([9], [13], [21], [22]) seem to be unavoidable for more realistic and therefore more complicated geometries of the problem.

Remark 3: The numerical tests show clearly the superiority of the BiCGSTAB iterative procedure to the classical Gradient method. It can also be seen that the number of iterations for BiCGSTAB without preconditioning redoubles if the dimension of the problem increases 16 times. If we assume that the condition number of the system is of the capital order $\varkappa = O(h^{-1})$, then the number of iterations is proportional to $O(\sqrt{\varkappa})$ which is usual for CG-like methods.

Remark 4: As Table 1 shows, the parallel generation of the system matrix scales up with the problem size and the number of processors in a nearly optimal way. The behaviour of the parallel iterative solution depends on the discretization parameters n and m (Table 3) and the different way of distributing the data among the processors (see Figures 2,3,4). It is clear that there is in general no optimal balance between the processors. Especially the solution of the n full complex eigenvalue problems for the preconditioner leads to a potential disbalance because of the variety of the convergence speed. Hence, the speed-up is not optimal, but sufficient.

REFERENCES

1. P. K. Banerjee and R. Butterfield. *Boundary Element Methods in Engineering Science.* McGraw-Hill Book Company (UK) Limited, 1981.
2. J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, 19 : 297–301, 1965.
3. A. J. Davies. The boundary element method on a transputer network. In C. Brebbia and G. Gipson, editors, *Boundary Elements XIII*, pages 985–994. Computational Mechanics Publications, Southampton Boston, 1991.
4. A. J. Davies. The implementation of the boundary element method on a network of transputers. Technical report 241, Hatfiled Polytechnical NOC, 1991.
5. B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler. Matrix eigensystem routines - EISPACK guide extension. volume 51 of *Lect. Notes in Comp. Science.* Springer Verlag, 1977.
6. K. Georgiev and S. Margenov. Boundary element method realization on a systolic processor. In *Proceedings of the 10-th Symposium on Algorithms, Algorithms'89*, pages 91–94, Strbske Pleso, 1989.
7. W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math.*, 54: 463–491, 1989.
8. G. C. Hsiao and W. L. Wendland. A finite element method for some integral equations of the first kind. *J. Math. Anal. and Appl.*, 3(58): 449–481, 1977.
9. G. C. Hsiao and W. L. Wendland. Domain decomposition in boundary element methods. In R. G. et. al., editor, *Proc. of Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 41–49. SIAM, Philadelphia, 1991.
10. B. N. Khoromskij and W. L. Wendland. Spectral equivalent preconditioners for boundary equations in substructuring techniques. *East-West Journal of Numer. Math.*, 1(1): 1–26, 1992.
11. B. N. Khoromskij and E. P. Zhidkow. Boundary integral equations on special surfaces and their applications. *Sov. J. Numer. Anal. Math. Model.*, 2: 463–488, 1988.
12. W. Lang u. a. *Dokumentation zum Programmsystem MEIWEP.* TH Karl-Marx-Stadt, Sektion Mathematik, 1978.
13. U. Langer. Parallel iterative solution of symmetric coupled FE/BE-equations via domain decomposition. *Contemporary Math.*, 157: 335–344, 1994.
14. A. Meyer and S. Rjasanow. An effective direct solution method for certain boundary integral equations in 3D. *Math. Methods in Appl. Sciences*, 13: 45–53, 1990.
15. J. C. Nedelec and J. Planchard. Une methode variationelle d'element finis pour la resolution numerique d'un probleme exterieur dans $I\!R^3$. *R.A.J.R.O, R3*, pages 105–129, 1973.
16. M. Pester and S. Rjasanow. A parallel version of the preconditioned conjugate gradient method for boundary element equations. *Num. Lin. Alg. with Appl.*, 1994. to appear.
17. S. Rjasanow. CG-Verfahren für die Randintegralgleichungen. Preprint 61, TU Karl-Marx-Stadt, 1988.
18. S. Rjasanow. Vorkonditionierte iterative Auflösung von Randelementgleichungen für die Dirichlet–Aufgabe. Wiss. Schriftenreihe (7), TU Chemnitz, 1990.
19. S. Rjasanow. Effective algorithms with circulant-block matrices. *Linear Alg. Appl.*, 202: 55–69, 1994.
20. B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klebe, and C. B. Moler. Matrix eigensystem routines - EISPACK guide. volume 6 of *Lect. Notes in Comp. Science.* Springer Verlag, 1976.
21. O. Steinbach. Boundary element methods in domain decomposition methods. Bericht 94–10, DFG-Schwerpunkt Randelementmethoden, 1994.
22. O. Steinbach. Parallel iterative solvers for symmetric boundary element domain

decomposition methods. Bericht 94–7, DFG-Schwerpunkt Randelementmethoden, 1994.

23. C. H. Tong. A family of quasi-minimal residual methods for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 15(1): 89–105, 1994.
24. H. A. van der Vorst. BiCGSTAB: a fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 13(2): 631–644, 1992.
25. C. Van Loan. *Computational Frameworks for the Fast Fourier Transform.* SIAM, Philadelphia, 1992.
26. W. L. Wendland. Bemerkungen zu Randelementmethoden und ihren mathematischen und numerischen Aspekten. *Mitteilungen der GAMM, Heft 2*, pages 3–27, 1986.

Other titles in the SPC series:

93_1 G. Haase, T. Hommel, A. Meyer and M. Pester. Bibliotheken zur Entwicklung paralleler Algorithmen. May 1993.

93_2 M. Pester and S. Rjasanow. A parallel version of the preconditioned conjugate gradient method for boundary element equations. June 1993.

93_3 G. Globisch. PARMESH – a parallel mesh generator. June 1993.

94_1 J. Weickert and T. Steidten. Efficient time step parallelization of full-multigrid techniques. January 1994.

94_2 U. Groh. Lokale Realisierung von Vektoroperationen auf Parallelrechnern. March 1994.

94_3 A. Meyer. Preconditoning the Pseudo-Laplacian for Finite Element simulation of incompressible flow. February 1994.

94_4 M. Pester. Bibliotheken zur Entwicklung paralleler Algorithmen. (aktualisierte Fassung). March 1994.

94_5 U. Groh, Chr. Israel, St. Meinel and A. Meyer. On the numerical simulation of coupled transient problems on MIMD parallel systems. April 1994.

94_6 G. Globisch. On an automatically parallel generation technique for tetrahedral meshes. April 1994.

94_7 K. Bernert. Tauextrapolation – theoretische Grundlagen, numerische Experimente und Anwendungen auf die Navier-Stokes-Gleichungen. June 1994.

94_8 G. Haase, U. Langer, A. Meyer and S. V. Nepomnyaschikh. Hierarchical extension and local multigrid methods in domain decomposition preconditoners. June 1994.

94_9 G. Kunert. On the choice of the basis transformation for the definition of DD Dirichlet preconditioners. June 1994.

94_10 M. Pester and T. Steidten. Parallel implementation of the Fourier Finite Element Method. June 1994.

94_11 M. Jung and U. Rüde. Implicit Extrapolation Methods for Multilevel Finite Element Computations: Theory and Applications. June 1994.

94_12 A. Meyer and M. Pester. Verarbeitung von Sparse-Matrizen in Kompaktspeicherform (KLZ/KZU). June 1994.

94_13 B. Heinrich and B. Weber. Singularities of the solution of axisymmetric elliptic interface problems. June 1994.

94_14 K. Gürlebeck, A. Hommel and T. Steidten. The method of lumped masses in cylindrical coordinates. July 1994.

94_15 Th. Apel and F. Milde. Realization and comparison of various mesh refinement strategies near edges. August 1994.

94_16 Th. Apel and S. Nicaise. Elliptic problems in domains with edges: anisotropic regularity and anisotropic finite element meshes. August 1994.

94_17 B. Heinrich. The Fourier-finite-element method for Poisson's equation in axisymmetric domains with edges. August 1994.

Some papers can be accessed via anonymous ftp from server `ftp.tu-chemnitz.de`, directory `pub/Local/mathematik/SPC`. (Note the capital L in Local!)