Technische Universität Chemnitz-Zwickau

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

G. Globisch and S.V. Nepomnyaschikh*

The hierarchical preconditioning having unstructured grids

Preprint SFB393/97_11

Abstract

In this paper we present two hierarchically preconditioned methods for the fast solution of mesh equations that approximate 2D-elliptic boundary value problems on arbitrary unstructured quasi uniform triangulations. Based on the fictitious space approach the original problem can be embedded into an auxiliary one, where both the hierarchical grid information and the preconditioner by decomposing functions on it are well defined. We implemented the corresponding Yserentant preconditioned conjugate gradient method as well as the BPX–preconditioned cg–iteration having optimal computational costs. Several numerical examples demonstrate the efficiency of the artificially constructed hierarchical methods which can be of enormous importance in the industrial engineering, when often only the nodal coordinates and the element connectivity of the underlying (fine) discretization are available.

Key words: partial differential equations, parallel computing, multilevel methods, hierarchical preconditioning, finite element methods, automatical grid generation

* The work of this author is also supported by the Deutsche Forschergemeinschaft (DFG).

Preprint-Reihe des Chemnitzer SFB 393

SFB393/97_11 **April 1997**

Contents

Authors' address:

Dr. rer. nat. Gerhard Globisch
Faculty for Mathematics
Technical University Chemnitz-Zwickau
D - 09107 Chemnitz, Germany

e-mail: `gerhard.globisch@mathematik.tu-chemnitz.de`

http://www.tu-chemnitz.de/sfb393/people/globisch.html

Dr. Sergey V. Nepomnyaschikh
Computing Center
Siberian Branch Russian Academy of Sciences
6 Lavrentiev av.
Novosibirsk 630090, Russia

e-mail: `svnep@comcen.nsk.su`

1 Introduction

In the next section we introduce the 2D-boundary value problem of second order having formally selfadjoint differential operator. Our aim is the numerical solution of the problem by hierarchical methods, although, in practice, we have its unstructured discretization available only. For this, in section 3 we construct the structured auxiliary problem into which the original one can be embedded. We define the one-to-one correspondence between the nodes of the unstructured mesh and the nodes of the hierarchically discretized square defining the fictitious space, cf. also [27]. This approach is used for applying the fictitious space lemma to derive the corresponding spectral equivalence inequality describing the preconditioning property of the artificially constructed hierarchical preconditioner belonging to the auxiliary grid points. In section 4 we give a short survey of the underlying theory presented detailed e.g. in [23, 27]. In the mentioned papers the convergence rate of the iterative process was proved to be so fast as it is the case for the conventional hierarchical solution method, i.e., it is (nearly) independent of the mesh size. In section 5 we discuss the various aspects of the numerical implementation of the new hierarchical method. We do it in the case of the auxiliary Yserentant preconditioning as well as in the more important case of the artificial BPX-preconditioner. In section 6 we illustrate the efficient implementation of the two algorithms computing several 2D–potential problems. Moreover, in each case we compare our artificially constructed hierarchical iteration based on the canonically performed refinement of the coarse and structured user triangulation with this method using unstructured fine grids generated by an advancing front mesh generator. Finally, first numerical results of the parallel implementation of our approach are given, where the corresponding numerical analysis is yet under consideration. The iteration numbers are rather satisfactory although the comparison with the parallelized structured methods is avoided. The basis of the implementation of the unstructured parallel solvers is a non-overlapping domain decomposition data structure (see e.g. [13, 16, 29]) such that they are well-suited for parallel machines with MIMD architecture. Section 6 is also an impressive performance to demonstrate the practical importance of the designed methods. Often, in the industrial engineering boundary value problems have to be solved, where a (rather) fine mesh of the domain and the discretization concept are given sometimes already resulting in the corresponding system of equations, see e.g. [31]. But no fast hierarchical solver can be applied because nothing is known about the grid structure. Using our approach this bottleneck isn't any more. We only mention here that our method can be transfered to the 3D calculations of boundary value problems.

2 The description of the original problem

Let $\Omega \subset \mathbb{R}^2$ be a bounded plane domain with a piecewise smooth boundary Γ which belongs to the class C^2 and satisfies the Lipschitz condition, see [34]. We consider the elliptic boundary value problem

$$\left.\begin{array}{l} -\sum\limits_{i,j=1}^{2} \frac{\partial}{\partial x_i} a_{ij}(x) \frac{\partial u}{\partial x_j} + a_0(x)\, u = f(x), \quad x = (x_1, x_2)^T \in \Omega \\ u(x) = g_0(x), \quad x \in \Gamma_0 \\ \frac{\partial\, u}{\partial N} + \sigma(x) u = g_1(x), \quad x \in \Gamma_1\ . \end{array}\right\} \qquad (1)$$

Here the symbol $\partial/\partial N$ denotes the conormal derivative w.r.t. the outward normal. On the boundary Γ of the domain Ω both Dirichlet boundary conditions and Neumann boundary conditions are imposed. We have $\Gamma = \Gamma_0 \cup \Gamma_1$. We introduce the following subspaces of the Sobolov space $H^1(\Omega)$.

$$H^1(\Omega, \Gamma_0) = \{u \in H^1(\Omega) : u(x) = g_0(x),\ x \in \Gamma_0\}$$
$$\overset{\circ}{H}{}^1 \ = \{v \in H^1(\Omega) : v(x) = 0,\ x \in \Gamma_0\}$$

Let us suppose that the coefficient functions $a_{ij}(x)$, $i, j = 1, 2$, and the right–hand side $f(x)$ of the above boundary value problem are such that from (1) we may derive the symmetric and coercive bilinearform

$$a(u, v) = \int_\Omega (\sum_{i,j=1}^{2} a_{ij}(x) \frac{\partial u}{\partial x_j} \frac{\partial u}{\partial x_j} + a_0(x)\, uv\,)\, dx + \int_{\Gamma_1} \sigma(x)\, uv\, dx\ ,$$

and the continuous linear functional

$$l(v) = \int_\Omega f(x)\, v\, dx + \int_{\Gamma_1} g_1(x)\, v\, dx$$

which define the well known variational problem

$$u \in H^1(\Omega, \Gamma_0)\ : a(u, v) = l(v) \quad \text{for all} \ \ v \in \overset{\circ}{H}{}^1, \tag{2}$$

where we seek for the solution $u \in H^1(\Omega, \Gamma_0)$. Having this, as we know e.g. by [4], for the variational problem (2) there is an unique solution $u \in H^1(\Omega, \Gamma_0)$ which we want to compute numerically. Hereafter, for simplicity we may suppose $g_0(x)$ to be equal to zero.

Let a positive parameter h be fixed which is sufficiently small and let $\Omega^h = \cup_{i=1}^{M} \tau_i$ be a quasiuniform triangulation of the domain Ω. In practice, often the triangulation is rather fine and unstructured, i.e., the mesh data information is consisted of the nodal coordinates and the element connectivity only, see e.g. Figure 1. The quasi uniformity of the triangulation Ω^h means that there are positive constants l_1, l_2 and s independently of the discretization parameter h such that

$$l_1 h \le r_i \le l_2 h\ , \quad \frac{r_i}{\rho_i} \le s\ , \quad i = 1, \dots M\ ,$$

where r_i and ρ_i are the radii of the circumscribed and the inscribed circles for the triangles τ_i, respectively, see also [10]. Moreover, we also assume that the triangulation boundary Γ^h approximates the boundary $\Gamma = \partial\Omega$ with an error $O(h^2)$, see [23] for more details.

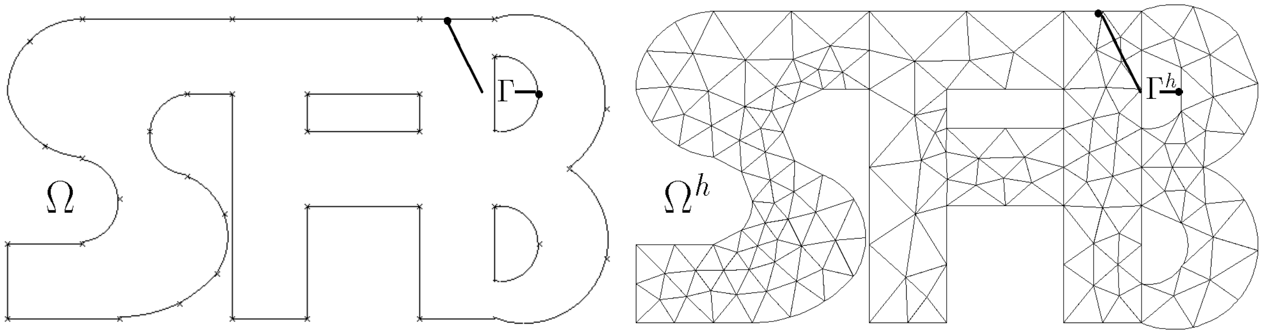


Figure 1: The domain Ω and its unstructured quasiuniform triangulation Ω^h.

For the triangulation Ω^h we define the space $H_h(\Omega^h)$ of real continuous functions which are linear on each triangle of Ω^h and vanish at the boundary part Γ_0^h. We extend these functions on $\Omega\backslash\Omega^h$ by zero. The solution u^h of the following projection problem

$$u^h \in H_h(\Omega^h) \ : a(u^h, v^h) = l(v^h) \quad \text{for all} \ \ v^h \in H_h(\Omega^h) \tag{3}$$

is called an approximate solution. Aspects of approximation have been thoroughly studied in [10, 28]. Each function $u^h \in H_h(\Omega^h)$ is put in standard correspondence with a real column vector $\underline{u} \in \mathbb{R}^N$ whose components are the values of the function u^h at the corresponding nodes of the triangulation Ω^h. Then, the problem (3) is equivalent to the solution of the system of mesh equations

$$\left.\begin{array}{l} A\underline{u} = \underline{f}\ , \quad \text{where we have:} \\ (A\underline{u}, \underline{v}) = a(u^h, v^h) \quad \text{for all} \ \ u^h, v^h \in H_h(\Omega^h)\ , \\ (\underline{f}, \underline{v}) = l(v^h) \quad \text{for all} \ \ v^h \in H_h(\Omega^h)\ . \end{array}\right\} \tag{4}$$

Here u^h and v^h are the corresponding prolongations of the vectors $\underline{u}$ and $\underline{v}$. The symbol $(\cdot,\cdot)$ denotes the Euclidian scalar product in $\mathbb{R}^N$.

The aim of this paper is to construct a symmetric positive definite preconditioning operator C for the problem (4) satisfying the spectral equivalence inequality

$$c_1(C\underline{u}, \underline{u}) \leq (A\underline{u}, \underline{u}) \leq c_2(C\underline{u}, \underline{u}) \quad \text{for all } \underline{u} \in \mathbb{R}^N\ , \tag{5}$$

where the positive constants c_1 and c_2 are independent of the discretization parameter h. Furthermore, jumping coefficients $a_{ij}(x)\, , i, j = 1, 2,$ may not essentially detoriate this fast convergence property of the corresponding solution method capitalizing from the above preconditioning. Clearly, the multiplication of a vector by C^{-1} should be easy to implement.

3 The construction of the auxiliary problem

The preconditioner C is constructed applying the method of fictitious space (see e.g. [25]) in two stages. At the first and interim stage we pass from the arbitrary unstructured triangulation Ω^h to an auxiliary structured non–hierarchical mesh, and, using this, at the second stage to the hierarchical mesh which is defined to be the hierarchical mesh of the square containing the original domain Ω. We note that the passage from an arbitrary triangulation to a structured mesh was earlier used in [24]. The preprint [27] includes the development of [23] for the case of locally refined grids. Other techniques for constructing the preconditioners on unstructured meshes were proposed in [5, 6, 9, 19, 20, 25, 32]. The definition of preconditioning operators having non-hierarchical grids was considered in [17].

In order to use the Lemma of fictitious space for analysing the artificially defined preconditioners we construct the discretized auxiliary space Π^h and the corresponding operators between Π^h and Ω^h as follows. We embed the domain Ω in a square Π, see Figure 2. Let K_i denote the union of triangles in the triangulation Ω^h which have the vertex z_i in common, and, let d_i be the maximum radius of the circles which may be inscribed into K_i. In the square Π we introduce an auxiliary rectangular grid Π^h having the step size $\bar{h}$ such that

$$\bar{h} < \frac{1}{2\sqrt{2}} \min_{i=1(1)N} (d_i)\ . \tag{6}$$

Now, let us fix $\bar{h} := l \cdot 2^{-J}$, where l is the length of the sides of Π and J is an appropriately choosen positive integer. Throughout the paper, speaking about the set Π^h and their subsets we identify $\bar{h}$ by h. We denote the nodes of the grid Π^h by $Z_{ij} = (x_i, y_j)$, $i, j = 1, 2, \ldots, L$. Using the cell diagonals we triangulate Π^h. For this see also Figure 2.

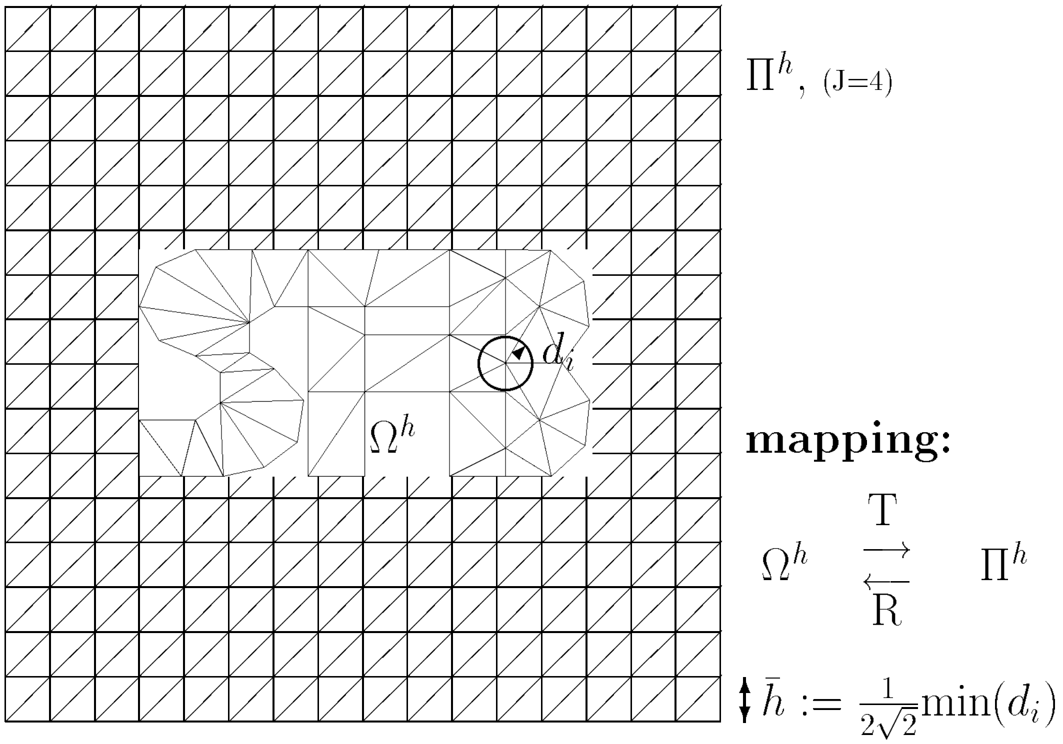


Figure 2: Embedding the unstructured mesh Ω^h into the auxiliary grid Π^h.

Let the cells of Π^h denoted by $D_{ij} = \{(x, y) : x_i \leq x < x_{i+1}\ , y_j \leq y < y_{j+1}\}$. Doing as given, we get $\Pi^h = \cup_{i,j=1}^{L}\ D_{ij}$.

Let Q^h be the minimum figure consisting of cells D_{ij} and containing Ω^h. Hence we have $\Omega^h \subset Q^h$. Hereafter, because of the above triangulation of Π^h the sets Π^h and Q^h and their subsets refer to triangulations as well. Let S^h be the set of the boundary nodes of Q^h. We subdivide the set S^h into two subsets S_0^h and S_1^h as follows. If $\bar{D}_{ij} \cap \Gamma_0 \neq \emptyset$, all the nodes of $D_{ij} \cap S^h$ are in S_0^h. Consequently we have $S_1^h = S^h \backslash S_0^h$.

Let $H_h(Q^h)$ be the space of the real continuous functions which are linear on the triangles of Q^h and vanish at the nodes of S_0^h.

Now, we define the projection operator $\mathcal{R} : H_h(Q^h) \rightarrow H_h(\Omega^h)$ and the extension operator $\mathcal{T} : H_h(\Omega^h) \rightarrow H_h(Q^h)$: For a given mesh function $U^h(Z_{ij}) \in H_h(Q^h)$ we define a function $u^h \in H_h(\Omega^h)$ as follows. Let z_l be a vertex in the triangulation Ω^h. Assuming that $z_l \in D_{ij}$ we put

$$u^h(z_l) = (\mathcal{R}U^h)(z_l) = U^h(Z_{ij}) .$$

The function u^h is equal to zero at the nodes $z_l \in \Gamma_0^h$. Let us define the expanded operator $R : H_h(\Pi^h) \rightarrow H_h(\Omega^h)$ to be the operator of restriction on Ω^h as follows.

$$(RU^h)(Z_{ij}) = U^h(Z_{ij}) \quad \text{for all } Z_{ij} \in Q^h .$$

Subdividing the nodes of Π^h into the nodes of Q^h including those of S^h and the remaining nodes and ordering them we obtain the matrix representation of R to be $R = (\mathcal{R}\ \ \mathbf{O})$, cf. also [1].

The definition of the operator $\mathcal{T}$ is given in the following. For the mesh function $u^h \in H_h(\Omega^h)$ we suppose a function $U^h \in H_h(Q^h)$. The function U^h is equal to zero at the nodes $Z_{ij} \in S_0^h$. At all of the other nodes the function U^h is defined as follows. If a cell D_{ij} contains a certain vertex z_l of the triangulation Ω^h we put

$$U^h(Z_{ij}) = (\mathcal{T}u^h)(Z_{ij}) = u^h(z_l)\ .$$

For each of the remaining nodes $Z_{ij} \in Q^h$ we find the closest vertex z_l of the triangulation Ω^h. In the case of several closest vertices we may choose any of them and using it we put the same as above. By the theorem of the extension of mesh functions given in [8] there exists the extension operator $T : H_h(\Omega^h) \to H_h(\Pi^h)$ which is also uniformly bounded w.r.t. the parameter h. This operator spreads Ω^h over Π^h.

Finally, in the space $H_h(Q^h)$ we define the operator A_Q to be

$$(A_Q U, V) = \int_{Q^h} ((\nabla U^h, \nabla V^h) + U^h V^h)\, dx\ , \tag{7}$$

for all $U^h, V^h \in H_h(Q^h)$ This is an auxiliary problem we did not discretize, but, we use the operator A_Q to make the application of the fictitious space lemma possible as it is done in the next section.

4 The application of the fictitious space lemma

Taking the conventions into account which we adopted in the previous section the preconditioning operator C in (5) can be constructed by means of the lemma of fictitious space, see also [24]. For convenience we give this lemma here.

Lemma 1

Let H_0 and H be Hilbert spaces with the scalar products $(\cdot,\cdot)_{H_0}$ and $(\cdot,\cdot)_H$, respectively. Let $A_0 : H_0 \to H_0$ and $A : H \to H$ be symmetric and positive definite continuous operators in the spaces H_0 and H. Suppose that $\mathcal{R}$ is a linear operator such that $\mathcal{R} : H \to H_0$ and $(A_0\mathcal{R}v, \mathcal{R}v)_{H_0} \le c_R(Av, v)_H$ is fulfilled for all $v \in H$. Moreover, there exists an operator $\mathcal{T}$ such that $\mathcal{T} : H_0 \to H$ for which the conditions $\mathcal{R}\mathcal{T}u_0 = u_0$ and $c_T(A\mathcal{T}u_0, \mathcal{T}u_0)_H \le (A_0u_0, u_0)_{H_0}$ are valid for all $u_0 \in H_0$. Here c_R and c_T are positive constants. Then

$$c_T(A_0^{-1}u_0, u_0)_{H_0} \le (\mathcal{R}A^{-1}\mathcal{R}^* u_0, u_0)_{H_0} \le c_R(A_0^{-1}u_0, u_0)_{H_0} \tag{8}$$

holds for all $u_0 \in H_0$. The operator $\mathcal{R}^*$ is the adjoint to $\mathcal{R}$ w.r.t. the scalar products $(\cdot,\cdot)_{H_0}$ and $(\cdot,\cdot)_H$, i.e., we have $\mathcal{R}^* : H \to H_0$ and $(\mathcal{R}^*u_0, v)_H = (u_0, \mathcal{R}v)_{H_0}$.

We note that for constructing and implementing the preconditioner, i.e., the operator $\mathcal{R}A^{-1}\mathcal{R}^*$, we only require the existence of the operator $\mathcal{T}$. Having the situation given in section 2 and 3, the role of the operator A_0 is played by the stiffness matrix A in (4). The finite dimensional space $H_h(\Omega^h)$ plays the role of the space H_0. The space $H_h(Q^h)$ is used to be the fictitious space. Thus, for the operator A may stand the A_Q.

Now, according to the above lemma, there exist positive constants $\underline{c}$ and $\overline{c}$ independent of the mesh size parameter h such that

$$\underline{c}(A^{-1}\underline{u}, \underline{u}) \le (\mathcal{R}A_Q^{-1}\mathcal{R}^*\underline{u}, \underline{u}) \le \overline{c}(A^{-1}\underline{u}, \underline{u})$$

is valid for all vectors $\underline{u} \in \mathbb{R}^N$. For the proof of this see also [27]. Hereafter we use the same designation for an operator and its matrix representation.
Considering (5) inversely, finally we get the following result which was proved in [27] taking distinct boundary conditions on Γ^h into account. There are positive constants c_1 and c_2 such that

$$c_1(A^{-1}\underline{u},\underline{u}) \leq (C^{-1}_{\Pi^h,\mathrm{bc}(\Gamma^h)} R^* \underline{u}, R^* \underline{u}) \leq c_2(A^{-1}\underline{u},\underline{u}) \tag{9}$$

is fulfilled for all vectors $\underline{u} \in \mathbb{R}^N$ belonging to the original discretization. In (9) the preconditioner $C^{-1}_{\Pi^h}$ is either the BPX–multilevel–preconditioner (see also [8]) or the Yserentant hierarchical preconditioner (see also [35]) which we may construct now on the structured hierarchical grid Π^h. As it was expected in the case of the artificial BPX–multilevel–preconditioner the constants c_1 and c_2 are independent of the auxiliary mesh size parameter h. Hence, the condition number of the operator $RC^{-1}_{\Pi^h}R^*A$ which we applied numerical within the cg–iteration process is of order $O(1)$. In the case of the artificially constructed Yserentant preconditioning we may have the condition number $\kappa(RC^{-1}_{\Pi^h}R^*A) = O((J+1)^2)$, where the index J indicates the depth of the artificially constructed hierarchical mesh Π^h. The numerical results given in section 6 illustrate the good convergence behaviour of the corresponding cg–methods impressively.

We implement the corresponding hierarchical preconditioners $C^{-1}_{\Pi^h}$ using the auxiliary grid Π^h as it is given in the next section.

5 Aspects of the numerical implementation

In this section we itemize and analyse the numerical operations which are additionally necessary for implementing the artificially constructed preconditioners within the corresponding conjugate gradient method.

In the case of the Yserentant preconditioning the correction vector $\underline{w}$ of the cg–iteration process is computed to be

$$\underline{w} = C^{-1}\underline{r} = \mathbf{Q}\,\mathbf{J}^{-1/2} \,||\, \mathbf{J}^{-1/2}\,\mathbf{Q}^T\underline{r} \tag{10}$$

where $\mathbf{Q}$ is the well known basis transformation between the usual finite element nodal basis and the hierarchical basis, see e.g. [13, 21, 35]. The symbol $\underline{r}$ denotes the residual vector and the matrix $\mathbf{J} = \mathrm{diag}(A)$ performes the Jacobi preconditioning as given.

In the case of the BPX–preconditioning we have

$$\underline{w} = C^{-1}\underline{r} = \sum_{j=0}^{J-1} \mathbf{Q}_j \,||\, [\mathbf{Q}_j^T\underline{r}] + \underline{r}\;, \tag{11}$$

where $\mathbf{Q}_j$ is the basis transformation belonging to the j-th level, $j = 0, \ldots, J-1$, see also [8, 13, 21]. Note that in this case the level–depending Jacobi preconditioning can also be applied.

The numerical implementation of our preconditioning methods $C^{-1}_{\Pi^h}$ means that we work in (10) and (11) using the longer vectors $\underline{v} = R^*\underline{r}$ consisting of either L^2 components belonging to the usual hierarchical list of the grid Π^h or the corresponding number of components defined by the hierarchical BPX-list of Π^h. We call the first artificially constructed hierarchical preconditioning "artYs" and the second method is epitomized by "artBPX". Let us note that in both cases we do not make use of any coarse grid solver as it is usually the case when the application of the hierarchical preconditioners is classically implemented. Now, we describe the following actions.

1. *The computation of the parameter* $\bar{h} < \frac{1}{2\sqrt{2}}\min(d_i)\,, i = 1, \ldots, N$:
Instead of calculating the maximum of the radii of inscribed circles w.r.t. the triangle set K_i having the point i in common, for all of the triangles $\tau_i \in \Omega^h\;, k = 1, \ldots, M$, we compute their three heights. Then, taking the minimum of them for defining d it must be divided by two to get the parameter $\bar{h}$. For this we need $M \cdot 3 \cdot 11 \cdot N \leq 99N$ operations.

2. *The definition of the auxiliary grid* Π^h *depending on* $\bar{h}$:
Using only the coordinates of the points at the boundary Γ^h we calculate $x_l = \min(x_1^i)\,,$ $x_r = \max(x_1^i)\,, y_b = \min(x_2^i)$ and $y_u = \max(x_2^i)\,, i = 1, 2, \ldots, \mathrm{card}(\Gamma^h)$. Then we have $l = \max(y_u - y_b\,, x_r - x_l)$ and $J = [\log(l/\bar{h})/\log(2)] + 1$. Thus, we arrive at $l = 2^J \cdot \bar{h}$ which is the length of the square Π and at $L^2 = (2^J + 1)^2$ which is the number of points in Π^h. The above symbol "$[\cdot]$" denotes the entier–operation Finally, we center Π^h with respect to Ω^h using x_l, y_b and l. The number of operations which is necessary for performing **2.** is negligible in comparison with the other efforts analysed in this section.

3. *The definition of the matrices* R^* *and* R*, respectively*:
For the matrix R^* we use a vector having L^2 components which are defined according to section 2. Speaking more detailed, for each point $k \in \Omega^h\,, k = 1, \ldots, N$, we are seeking for the cell $D_{ij}\,, i, j = 1, 2, \ldots, L$, containing the point k. Provided that the nodes of Π^h are numbered linewise from below to above and knowing L, for the point k the row and column indices i and j are computed. Then, we get the number k_p of a vertex in Π^h. We put k into R^* at the k_p-th position. The other components of R^* are set to be zero. We get

$$R^*(k_p) = \begin{cases} k, & \text{if} \quad k = 1, \ldots, N \\ 0, & \text{otherwise}\,, \end{cases}$$

where $k_p = 1, 2, \ldots, L^2$. The number of arithmetical operations for implementing the above calculation of the nodal point in Π^h uniquely assigned to the vertex in Ω is a total of approximately $7N$.

4. *The determination of the hierarchical list depending on the mesh* Π^h:
Let the points of the grid Π^h be numbered linewise from below to above throughout the structured hierarchical mesh Π^h having the depth J.
By the little subroutine called "locpoint$(k, L, L^2, J, j, fath_1, fath_2)$" for each input node $k\;, k = 1, 2, \ldots, L^2$, we compute the position j of k in the auxiliary hierarchical list and the two father nodes taking the mesh connectivity of the hierarchical grid Π^h of depth J into account. For this we need a total amount of approximately $6L^2 \sim C(\frac{h}{\bar{h}})N$ numerical operations, where the constant $C(\frac{h}{\bar{h}})$ depends on the given homogeneity of the unstructured grid Ω^h. Because the operation of the type $\mathbf{QQ}^T$ (see (10) and (11), respectively) can be reduced to the simple utilization of the corresponding hierarchical list we get the amount for this which is equivalent to $2 \cdot L^2$ in each iteration step of the cg–method. Although the memory size of $4L^2$ words for the auxiliary Yserentant hierarchical list is considerable we note that their determination performed only once at the beginning of the cg–iteration is really more effective than the utilization of nested differences of the coordinates of all of the nodes k of the grid Π^h which had to be used within each iteration step two times for doing the same as required by (10) and (11). The auxiliary Yserentant hierarchical list can be easily extended to the BPX–list of the grid Π^h calling the program "hb2bpx$(\cdot)$". The amount for implementing (11) is equivalent to those which was given above for (10), where, correspondingly, for the auxiliary BPX-list the memory size less than $8L^2$ words is necessary.

5. *Computing the diagonal matrix* $\mathbf{J}_{\Pi^h}$ *for also performing the Jacobi preconditioning*: At first we set the real vector $\mathbf{J}_{\Pi^h}(k)\,,k = 1,\ldots,L^2$, zero. Now, for all points k in the closure $\bar{\Omega}^h$ we want to compute a real number approximating the inverse of the corresponding main diagonal element of the auxiliary stiffnes matrix A_Q introduced by (7) in section 2. Then, we put this number at the k-th position of the vector. We implement this numerically for all of the triangles $\tau_i \in \Omega^h\,, i = 1,\ldots,M$, as follows.

Using the three vertices of the τ_i we define the minimum rectangular union of cells $D_{ij} \in \Pi^h$ encompassing the triangle. Now, for all vertices x of this cell union we perform the decision whether x is inside or outside the closure of the triangle $\tau_i\,, i = 1,\ldots,M$. In the case of interior points we mark them by putting the number of the corresponding triangle into the vector positions $\mathbf{J}_{\Pi^h}(k)$. In the case of a vertex which is located at an edge of the triangle τ_i we also mark this location by the number of the triangle, but, in addition to, specifically. The process for performing the inner/outer decision is considered in the following.

Let $P_i(t_i)\,, i = 1,2,3$, be the parametrizations of the three straight lines defined by the three edges of the triangle τ_i. Obviously, to get the i-th edge including both the start and the end vertex, we vary the real parameter t_i in the interval $[0,1]$. Fixing the vertex x we define the parametrized equation of the horizontal straight line to be $P(t_x) = x + t_x(1,0)^T$, where $t_x \in (-\infty,\infty)$. Cutting $P(t_x)$ and $P_i(t_i)\,, i = 1,2,3$, we count the number of positive and negative signs of the parameter t_x, respectively. We do this when the horizontal straight line has a non empty intersection with each of the straigh lines $P_i(t_i)\,, i = 1,2,3$, where the parameter t_i is in the open interval $(0,1)$. We get

$$\mathrm{card}\{\mathrm{sign}(t_x) : \mathrm{sign}(t_x) < 0 \ \text{ where: } \ (x + t_x(1,0)^T) \cap P_i \neq \emptyset\,, i = 1,2,3\,, t_i \in (0,1)\}\,,$$

$$\mathrm{card}\{\mathrm{sign}(t_x) : \mathrm{sign}(t_x) > 0 \ \text{ where: } \ (x + t_x(1,0)^T) \cap P_i \neq \emptyset\,, i = 1,2,3\,, t_i \in (0,1)\}\,.$$

Having the triangle domain it is easy to see that the two sets above have either an even or a odd cardinal number. In the case of the odd number the discrete vertex x is in the closure of the triangle τ_i, otherwise, it is outside. Naturally, if the horizontal line parameter t_x is equal to zero, we have $x \in \bar{\tau}_i$.

In the limit case, i.e., if one edge related straight line parameter t_i is equal to 0 or 1 we shift the horizontal test straight line orthogonally upward and downward using the shift vector $(0,\epsilon)^T$ and $(0,-\epsilon)^T$, respectively, where $\epsilon < \bar{h}$ is sufficiently small. Then, according to the above regulation, again testing the cut-behaviour we get the desired signs of the parameter t_x putting it into the decision set. This algorithm for performing the inner/outer decision can be generalized to the case of a domain which is bounded by arbitrary piecewisely parametrized boundary descriptions. In our case, taking all triangles of the unstructured grid Ω^h into account we approximately need the numerical amount $6M + 60L^2$ to do as described.

Let us note the following. By the special marking of all of the cell points of the grid Π^h that remained up to now unmarked and have the directly horizontal and/or vertical and/or inclined edge–connection with at least one marked interior point which is not located at the edge of an triangle we get the shape of the auxiliary domain Q^h closing Ω^h in the form of steps. This marking process runs globally throughout Π^h, i.e., step by step, for all the points $k\ \,, k = 1,2,\ldots,L^2$, in Π^h the corresponding (triangular) seven point star is considered making the above decision.

When the above marking was well done we may use the interim number entries in the vector $\mathbf{J}_{\Pi^h}$ to approximate real main diagonal values as follows, where we have the entries

$a_{ii}\,, i=1,\ldots,N$, of the stiffness matrix A of the original problem available.

$$\mathbf{J}_{\Pi^h}(k) = \begin{cases} a_{ii}\,, & \text{if } \; R^*(k) = i\,,\; i=1,\ldots,N \\ \tilde{a}_{kk}\,, & \text{if } \; R^*(k) = 0\;. \end{cases} \tag{12}$$

When the vector component $\mathbf{J}_{\Pi^h}(k)$ was marked by a triangle number belonging to k the value $\tilde{a}_{kk}$ can be computed to be e.g. the arithmetic mean

$$\tilde{a}_{kk} = \frac{1}{3}(a_{i_1i_1} + a_{i_2i_2} + a_{i_3i_3})\,, \tag{13}$$

where the values $a_{i_1i_1}\,, a_{i_2i_2}$ and $a_{i_3i_3}$ are the main diagonal entries of A belonging to the three nodes of the corresponding triangle number previously set into the k-th position of $\mathbf{J}_{\Pi^h}(k)$. When the component $\mathbf{J}_{\Pi^h}(k)$ was unmarked the zero value remains as it was set at the beginning. Moreover, we can define $\tilde{a}_{kk}$ to be the edge related weighted mean

$$\begin{aligned} \tilde{a}_{kk} = \; & \tfrac{s_{i_1i_3}+s_{i_2i_3}-s_{i_1i_2}}{s}\left(\tfrac{a_{i_1i_1}+a_{i_2i_2}}{2}\right) + \tfrac{s_{i_1i_2}+s_{i_2i_3}-s_{i_1i_3}}{s}\left(\tfrac{a_{i_1i_1}+a_{i_3i_3}}{2}\right) \; + \\ & \tfrac{s_{i_1i_2}+s_{i_1i_3}-s_{i_2i_3}}{s}\left(\tfrac{a_{i_2i_2}+a_{i_3i_3}}{2}\right)\,, \end{aligned} \tag{14}$$

where $s = s_{i_1i_2} + s_{i_1i_3} + s_{i_2i_3}$ and the magnitude $s_{i_mi_n}$ is the distance of the vertex $x \in \Pi^h$ in the closure of the triangle τ_i to the triangle edge which has the start point i_m and the end point $i_n\,, m,n = 1,2,3,\; m \neq n$. Choosing the above definition of the vector $\mathbf{J}_{\Pi^h}$, in comparison with the arithmetic mean definition the convergence behaviour of the method becomes hardly improved. For the above two opportunities to set real values into $\mathbf{J}_{\Pi^h}(k)$ we need less than approximately $3L^2$ and $(17+33)L^2$ numerical operations, resp. When we apply the artificially constructed BPX–preconditioner the vector $\mathbf{J}_{\Pi^h}(k)\,, k=1,\ldots,L^2$, can be extended to the corresponding BPX–length with negligible amount.

Thus, we may conclude as follows. In the case of "artYs" as well as in the case of "artBPX" the preconditioner has an optimal computational cost, i.e. the number of arithmetic operations required for their implementation is proportional to the number of unknowns in the problem.

If the problem (1) includes jumping coefficients $a_{ij}(x)\,, i,j = 1,2$, we perform the "outer" Jacobi–preconditioning that corresponds to the calculation $\underline{w} := \mathbf{J}^{-1/2}C^{-1}\mathbf{J}^{-1/2}\underline{r}$. Especially, if the ratio of the jumps are large, say e.g. greater than 100, the artificial preconditioning methods would fail without doing as given. In this case the diagonal matrix $\mathbf{J}_{\Pi^h}(k)$ between $\mathbf{Q}$ and $\mathbf{Q}^T$ has simply the entries 1 when the k were marked ($k = 1,2,\ldots,L^2$, see **5.**) and 0 otherwise. The above outside vector $\mathbf{J}^{-1/2}$ of length N contains the $-1/2$–root of the main diagonal of A.

To get the first parallel implementation of our methods we have done the following. Whereas for the parallelization of the classical hierarchical methods by the non overlapping domain decomposition the communication w.r.t. the correction values belonging to the coupling nodes is performed at the stage marked by the symbol "||" in (10) and (11), respectively, see e.g. [13, 14, 16, 21], in our case we can not use this approach. Embedding the p subdomain meshes $\Omega_s^h\,, s=1,\ldots,p$, arisen from the domain decomposition of the domain Ω into the corresponding auxiliary grids $\Pi_s^h\,, s=1,\ldots,p$, in general we get an overlapping union of the auxiliary regions. Having the described situation the corresponding vector types of the parallel cg-method can not be handled as it is well known up to now, see e.g. [21] and the references therein. To overcome the difficulties that occur when the first experiments in section 6.2 were made we perform one communication before applying $\mathbf{Q}^T$ and one communication after $\mathbf{Q}$ was completed. Hence, we accumulate the correction vector $\underline{w}_s$, $s = 1, \ldots, p$, before starting the next cg–step. By means of this strategy we get a parallelizable preconditioner. Thus, e.g. for the Yserentant hierarchical preconditioning we have

$$\underline{w}_s = || \; C_s^{-1} \; || \, \underline{r_s} \; := \; \sum_{s=1}^{p} H_s \, [R_s \; C_{\Pi_s^h}^{-1} \; R_s^*(\sum_{s=1}^{p} \; H_s^T \underline{r_s})] \; ,$$

where the accumulation matrices H_s symbolically handle the communication w.r.t. the residual vectors $\underline{r}_s$, $s = 1, \ldots, p$, distributed to p processors having L_s^2 components there.

As it is given in section 6.2, it seems that this approach gives rather bad iteration numbers when the auxiliary grids Π_s^h , $s = 1, \ldots, p$, do really overlap. The problem of the definition of the preconditioner in the case of the parallel cg–method such that the effect of the preconditioning is independent of the mesh size remains yet to be solved.

6 Numerical results

This section is devided into two subsections consisting of the numerical tests computing potential problems sequentially on a large HP workstation, and, in parallel using the GCPowerPlus multiprocessor computer, respectively.

The tables presented here contain the results for the cg–algorithm preconditioned by the artificially constructed Yserentant preconditioner "artYs" as well as by the arificially constructed BPX–preconditioner "artBPX" both computing the itemized test examples. The subcolumn marked by "struct. grid" means that we perform computations using a coarse structured initial grid successively refined canonically as the level depth J increases but embedded in the corresponding auxiliary grid Π^h consisting of L^2 points.[1] For comparison the subcolumn marked by "unstr. grid" contains the results belonging to really unstructured grids generated by the mesh generator given in [11] having (nearly) the same number N of degrees of freedom. Here, both the number of cg–iterations and the corresponding CPU–time (in sec) are given which were needed to get the relative error of the cg–iteration less than the previously defined accuracy $\epsilon = 10^{-4}$.[2] The relative error was measured in the $AC^{-1}A$-norm. In the first column indicating the depth J sometimes two numbers divided by the symbol "/" are given which differ from each other. Then, the first number belongs to the auxiliary grid depth due to the canonical refinement of the structured initial mesh and the second one is the depth of the auxiliary grid having some inhomogeneities causing the different J by means of the computation of the triangle heights. Naturally, here we have another number of L^2 given in the corresponding row below.

At the bottom of all of the tables the percentages of the CPU–time are given which were necessary for performing the operations indicated by R^*, R, and the preconditioning $C_{\Pi^h}^{-1}$ within the cg–iteration, respectively, where the third percentage includes also the amount of the cg–iteration itself. The percentages are measured on an average w.r.t. the given depths J of the auxiliary grids. Taking this percentages into account we finally discover that the artificially constructed hierarchical methods using only the nodal coordinates and the element connexion need the numerical effort which is approximately 1.6 times

[1] In every table changed, in the columns marked by "struct. grid", using scriptsize the added brackets include the iteration number and the corresponding CPU-time for the real structured hierarchical methods.

[2] In the given CPU-time neither the times for computing the hierarchical lists of the auxiliary grid Π^h and the step form approximation Q^h inside nor the time for considering the support of the corresponding grid functions w.r.t. the boundary conditions on Γ^h are incorporated. In practice this hidden amount does enlarge the real CPU-time substantially.

more than the effort of the original hierarchical approach having a lot of additional mesh data information to be input. Therefore the application of our new methods is a good practice, especially, for the industrial engineering.
We do not hide the following which we observed e.g. computing the examples **4.** and **5.** The more the unstructured meshes get lost their quasiuniformity the more the iteration numbers of the corresponding preconditioned cg–method increase. But this is in accordance with our theory. For the approach to get rid of the behaviour caused by locally refined grids see e.g. [27].

6.1 Sequential Computing

The results are computed by means of the HP 9000/889 K460-workstation using large memory size (1GigaByte) and on an average 7MFlop performance. The executable programs are called "pmhi.artYs.HPPA.px" in the case of the artificially performed Yserantant hierarchical preconditioning and "pmhi.artBPX.HPPA.px" in the BPX–case, respectively. The information about the software background of these packages including tools of the pre and postprocessing are contained e.g. in [1, 2].

1. *Preconditioning having the potential problem in the square:*

$$-\Delta u = 0 \quad \text{in} \quad \Omega = (0,4) \times (0,4)$$

$$u = \begin{cases} 0\,, & \text{on } \Gamma_{01} = \{x = (x_1, x_2)^T : x_1 = 0, x_2 < 1\} \cup \{x : x_2 = 0\,, 0 < x_1 \le 4\} \\ 1\,, & \text{on } \Gamma_{02} = \{x : x_1 = 0, 1 \le x_2 \le 4\}\,, \end{cases}$$

$$\text{where} \quad \Gamma_0 = \Gamma_{01} \cup \Gamma_{02}\,; \quad \text{and,} \quad \partial u/\partial N = 0 \quad \text{on} \quad \Gamma_1 = \partial\Omega \backslash \Gamma_0\,.$$

			artYs		artBPX	
J	N	L^2	struct. grid	unstr. grid	struct. grid	unstr. grid
3/4	25	81	[9 (0.00)] 13 (0.01)	11 (0.00)	[9 (0.00)] 9 (0.00)	9 (0.00)
4/5	81	289	[13 (0.01)] 19 (0.02)	15 (0.02)	[11 (0.01)] 12 (0.01)	11 (0.02)
5/6	289	1089	[17 (0.01)] 24 (0.03)	19 (0.07)	[13 (0.01)] 14 (0.02)	12 (0.05)
6/7	1089	4225	[20 (0.03) 27 (0.13)	22 (0.30)	[14 (0.03)] 15 (0.07)	13 (0.21)
7/8	4225	16641	[24 (0.14)] 30 (0.51)	25 (1.78)	[15 (0.10)] 16 (0.34)	15 (1.33)
8/9	16641	66049	[26 (0.79)] 32 (3.28)	28 (9.77)	[15 (0.51)] 16 (1.75)	16 (6.57)
9/10	66049	263169	[28 (4.36)] 33 (15.09)	26 (36.23)	[15 (2.56)] 16 (8.15)	17 (27.61)
10/11	263169	1050625	[29 (21.13)] 33 (64.89)	26 (143.62)	[15 (12.09)] 16 (33.83)	22 (137.73)
11/12	1050625	4198401	[29 (86.18)] 33 (255.49)	29 (681.89)	[15 (48.95)] 16 (139.99)	mem. ex.
12/–	4198401	16785409	[29 (348.20)] mem.ex.	mem. ex.	memory exceeded	
R^*:				24		22
R:				20		19
$C_{\prod^h}^{-1}$:				56		59

Table 1: #cg–iterations and CPU–times for the computing in the square

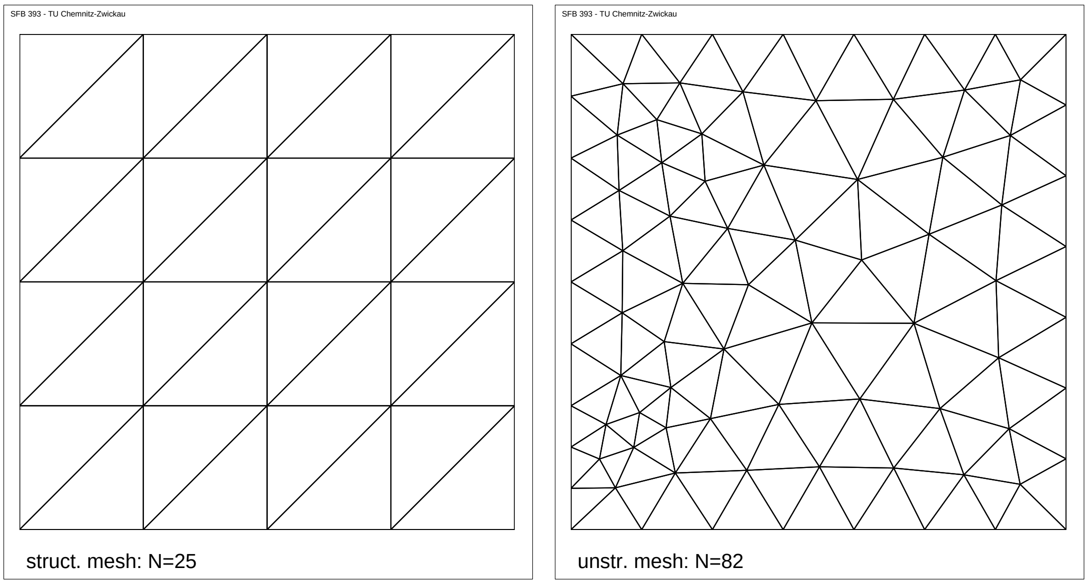


Figure 3: The structured mesh ($N = 25$) and the unstructured mesh ($N = 82$)

2. *Preconditioning having the potential problem in the club shaped domain:*

$$
\begin{aligned}
&-\Delta u = 0 \quad \text{in} \;\; \Omega = \{x : x_1^2 + x_2^2 < 1\} \\
&u = \begin{cases} \;\;\,1\,, & x \in \Gamma_{01} \text{ marked by (1) in Figure 4} \\ -1\,, & x \in \Gamma_{02} \text{ marked by (2) in Figure 4}\,, \end{cases} \\
&\partial u / \partial N = 0 \quad \text{on} \quad \Gamma_1 = \partial\Omega \setminus (\Gamma_{01} \cup \Gamma_{02})\,.
\end{aligned}
$$

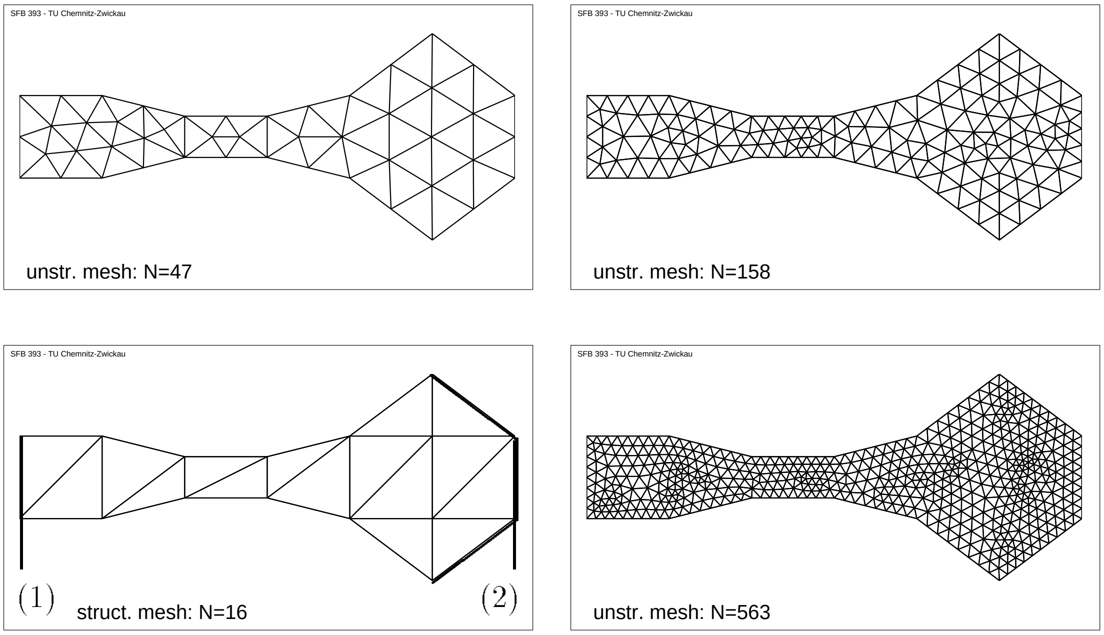


Figure 4: The structured mesh and a subsequence of the unstructured meshes

J	N	L^2	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
5	16	1089	[9 (0.00)] 10 (0.01)	10 (0.01)	[9 (0.00)] 9 (0.01)	9 (0.01)
6	47	4225	[15 (0.00)] 18 (0.06)	14 (0.04)	[14 (0.00)] 15 (0.05)	14 (0.05)
7	157	16641	[19 (0.01)] 21 (0.25)	19 (0.22)	[17 (0.01)] 18 (0.22)	16 (0.23)
8	569	66049	[22 (0.02)] 23 (1.61)	18 (1.17)	[19 (0.02)] 20 (1.52)	18 (1.42)
9	2161	263169	[26 (0.07)] 31 (10.47)	23 (7.27)	[19 (0.07)] 23 (8.18)	20 (6.90)
10	8417	1050625	[28 (0.36)] 35 (86.75)	26 (37.46)	[19 (0.27)] 26 (36.58)	23 (32.47)
11	33217	4198401	[30 (2.00)] 41 (201.23)	36 (128.46)	[19 (1.49)] 30 (240.35)	28 (161.91)
12	131969	16785409	[30 (11.01)] 43 (847.29)	40 (794.20)	[19 (7.04)] mem.ex.	mem. ex.
R^*:				18		15
R:				10		9
$C_{\Pi^h}^{-1}$:				72		76

Table 2: #cg–iterations and CPU–times for the computing in the club shaped domain

3. *Preconditioning having the problems (a) and (b) in the circular domain:*

$$-\mathrm{div}(a(x)\mathrm{grad}(u(x))) = 0 \quad \text{in} \quad \Omega = \{x : x_1^2 + x_2^2 < 1\}$$

where (a) : $a(x) = 1\,, \quad x \in \Omega\,,$

and $\quad$ (b) : $a(x) = \begin{cases} 1\,, & x \in \Omega_1 = \Omega \setminus \bar{\Omega}_2 \\ 10^6\,, & x \in \Omega_2 \text{ marked by (2) in Figure 5} \end{cases}$

$$u = \begin{cases} 100\,, & \text{on } \Gamma_{01} = \{x : \ x_1^2 + x_2^2 = 1\,, \ -1 \le x_1 \le -\frac{\sqrt{2}}{2}\,, \ 0 \le x_2 \le \frac{\sqrt{2}}{2}\ \} \\ 0\,, & \text{on } \Gamma_{02} = (\partial\Omega \backslash \Gamma_{01})\,. \end{cases}$$

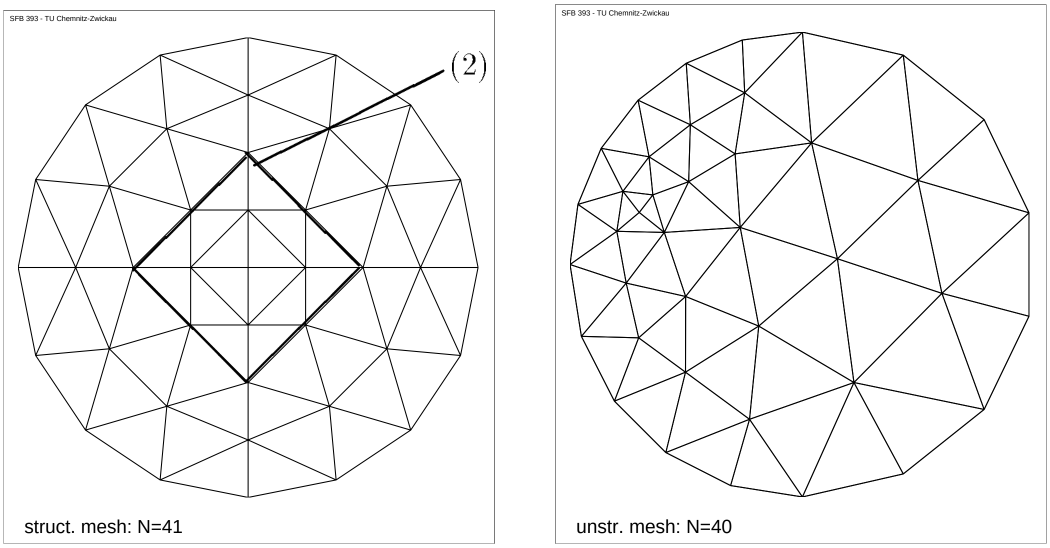


Figure 5: The structured mesh (N=41) and the unstructured mesh ($N = 40$)

J	N	L^2	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
3/4	13	81	[5 (0.00)] 5 (0.00)	4 (0.00)	[5 (0.00)] 5 (2.45)	5 (0.00)
4/5	41	289	[9 (0.00)] 12 (0.01)	9 (0.01)	[8 (0.00)] 10 (0.00)	9 (0.03)
5/6	145	1089	[14 (0.00)] 16 (0.02)	12 (0.04)	[11 (0.00)] 12 (0.01)	12 (0.04)
6/7	545	4225	[18 (0.02)] 15 (0.17)	15 (0.17)	[13 (0.01)] 13 (0.05)	13 (0.17)
7/8	2113	16641	[21 (0.06)] 21 (0.17)	19 (1.38)	[15 (0.05)] 15 (0.25)	14 (1.86)
8/9	8321	66049	[24 (0.32)] 24 (1.91)	23 (7.38)	[15 (0.22)] 16 (1.44)	15 (9.10)
9/10	33025	263169	[25 (1.70)] 28 (10.54)	26 (33.05)	[16 (1.25)] 19 (7.71)	17 (45.05)
10/11	131585	1050625	[26 (9.68)] 33 (53.20)	31 (154.38)	[16 (6.19)] 22 (38.16)	21 (126.79)
11/12	525313	4198401	[26 (41.19)] 40 (258.82)	38 (242.76)	[16 (27.21)] 23 (158.3)	mem. ex.
12/–	2099201	16785409	[26 (164.55)] 42 (1086.4)	mem. ex.	[16 (101.07)] mem.ex.	mem. ex.
R^*:				24		23
R:				22		21
$C_{\Pi^h}^{-1}$:				54		56

Table 3: #cg–it. and CPU–times for the homogeneous problem in the circular domain

J	N	L^2	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
3/4	13	81	[5 (0.00)] 6 (0.00)	5 (0.43)	[5 (0.00)] 6 (0.00)	6 (0.00)
4/5	41	289	[18 (0.00)] 25 (0.01)	27 (0.03)	[15 (0.00)] 20 (0.01)	20 (0.03)
5/6	145	1089	[33 (0.02)] 42 (0.04)	42 (0.14)	[23 (0.01)] 35 (0.05)	32 (0.11)
6/7	545	4225	[45 (0.04)] 63 (0.19)	54 (0.51)	[28 (0.04)] 50 (0.24)	44 (0.57)
7/8	2113	16641	[53 (0.15)] 37 (0.55)	33 (2.41)	[35 (0.12)] 31 (0.50)	29 (2.49)
8/9	8321	66049	[63 (0.81)] 50 (4.29)	42 (11.41)	[38 (0.57)] 38 (3.28)	36 (12.14)
9/10	33025	263169	[72 (4.93)] 66 (34.48)	59 (82.77)	[41 (3.24)] 49 (18.69)	45 (67.71)
10/11	131585	1050625	[80 (30.45)] 78 (130.71)	69 (413.29)	[47 (23.89)] 39 (65.54)	36 (205.01)
11/12	525313	4198401	[88 (243.07)] 93 (674.28)	85 (1755.6)	[51 (92.45)] 53 (368.83)	mem. ex.
12/–	2099201	16785409	[94 (620.26)] mem.ex.)	memory exceeded		

Table 4: #cg–iterations and CPU–times for the material problem in the circular domain

Obviously, this test example and also the first one have the peculiarity of the given jumping boundary condition in common. Naturally, the real unstructured meshes for computing the inhomogeneous problem were also generated by the advancing front algorithm given in [11], where the interfaces can be taken into account. To abbreviate this section we renounce to present the corresponding grids.

4. *Preconditioning having the problems (a) and (b) in the "SFB–domain":*

$$-\mathrm{div}(a(x)\mathrm{grad}(u(x))) = 0 \quad \text{in} \quad \Omega = SFB\,, \qquad \text{see Figure 1}\,,$$

$$\text{where (a): } a(x) = 1\,, \quad x \in \Omega = SFB$$

$$\text{and} \qquad \text{(b): } a(x) = \begin{cases} 1\,, & x \in S \\ 10^3\,, & x \in F \\ 10^6\,, & x \in B \end{cases}$$

$$u = x_1 + x_2 + 1 \quad \text{on} \quad \Gamma_0 \quad = \quad \text{exterior part of } \partial\Omega\,,$$

$$\partial u/\partial N = 0 \quad \text{on} \quad \Gamma_1 = \partial\Omega \setminus \Gamma_0 \quad = \quad 3 \text{ interior boundary pieces}\,.$$

For this problem the really unstructured mesh having $N = 163$ nodes was already shown in Figure 1. For completing the structured part of the tables below we used the initial mesh given in Figure 2 ($N = 50$) consecutively refining it canonically.

			artYs		artBPX	
J	N	L^2	struct. grid	unstr. grid	struct. grid	unstr. grid
5	50	1089	[7 (0.00)] 8 (0.00)		[8 (0.00)] 7 (0.02)	
6	157	4225	[15 (0.01)] 15 (0.03)	14 (0.03)	[16 (0.03)] 13 (0.07)	12 (0.04)
7	536	16641	[22 (0.03)] 23 (0.16)	18 (0.13)	[21 (0.04)] 20 (0.28)	13 (0.18)
8	1954	66049	[28 (0.10)] 25 (1.13)	24 (1.09)	[25 (0.10)] 24 (2.17)	17 (1.63)
9	7430	263169	[33 (0.39)] 34 (7.36)	37 (7.53)	[28 (0.45)] 31 (13.04)	27 (11.41)
10	28942	1050625	[38 (2.85)] 41 (44.75)	39 (58.12)	[30 (2.48)] 33 (56.27)	30 (48.64)
11	114206	4198401	[41 (14.93)] 46(170.39)	48 (212.75)	[31 (13.38)] 34 (241.84)	34 (219.28)
12	453694	16785409	[44 (67.55)] 70(1262.1)	83 (1470.6)	[32 (52.05)] mem.ex.	mem. ex.
R^*:				25		23
R:				23		20
$C_{\Pi^h}^{-1}$:				52		57

Table 5: #cg–it. and CPU–times for the homogeneous problem in the "SFB–domain"

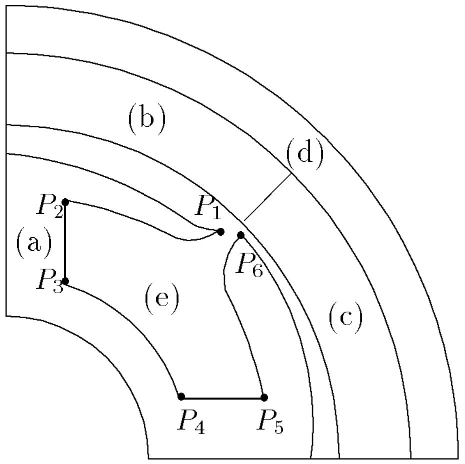
J	N	L^2	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
5	50	1089	[8 (0.00)] 9 (1.01)		[8 (0.02)] 8 (0.02)	
6	157	4225	[16 (0.01)] 16 (0.04)	17 (0.27)	[14 (0.03)] 15 (0.08)	14 (0.05)
7	536	16641	[22 (0.03)] 20 (3.56)	20 (0.16)	[20 (0.03)] 21 (0.31)	18 (0.18)
8	1954	66049	[27 (0.08)] 27 (5.55)	26 (1.18)	[24 (0.10)] 20 (1.47)	26 (1.91)
9	7430	263169	[33 (0.37)] 40 (9.66)	37 (9.68)	[26 (0.44)] 28 (10.57)	30 (17.97)
10	28942	1050625	[37 (2.55)] 56 (62.14)	59 (97.44)	[28 (2.29)] 45 (68.67)	40 (285.98)
11	114206	4198401	[40 (13.98)] 78(350.76)	88 (396.96)	[28 (11.39)] 60 (406.59)	54 (347.28)
12	453694	16785409	[43 (65.61)] 83(1496.6)	96 (1740.8)	[28 (45.73)] mem.ex.	mem. ex.
R^*:				25		23
R:				23		20
$C_{\Pi^h}^{-1}$:				52		57

Table 6: #cg–iterations and CPU–times for the material problem in the "SFB–domain"

Figure 6: The filled "SFB"–isoline picture delivered by our postprocessing

5. *Magnetic field computation in an electronic motor:*

This example is of important practical interest, cf. [12, 15] also for details. The domain Ω is the fourth of the cross section of an electronic motor the magnetic field computation must be calculated in. The following Figure 7 presents motor's geometry with its distinct material properties additionally connected with geometric peculiarities, which are causing solution's singularities in several indicated points P_i, $i = 1, \dots, 6$, see also [11] for more details.



abs. permeability : $\mu_0 = 1.257 * 10^{-6} Vs/Am$

rel. permeability and the given materials :

(a)	iron rotor	$\mu_r = 1694$
(b), (c)	permanent magnet	$\mu_r = 1.15$
(d)	sheet–metal shell	$\mu_r = 2488$
(e)	air gap	$\mu_r = 1$

Figure 7: The fourth-cross section of the electronic motor containing 4 materials

By means of Maxwell's laws the magnetic field problem defined on motor's cross section can be rewritten in the following variational formulation, cf. also [12, 15] :

Find the function $u \in \overset{o}{H}{}^1$ such that for all $v \in \overset{o}{H}{}^1$ holds :

$$\int_\Omega \frac{1}{\mu_0 \mu_r(x)} \nabla^T u \, \nabla v \, dx_1 \, dx_2 = \int_\Omega \frac{1}{\mu_0 \mu_r(x)} \left(\frac{\partial v}{\partial y} B_{0x_1} - \frac{\partial v}{\partial x} B_{0x_2} \right) dx_1 \, d_2 \ ,$$

where B_{0x_1} and B_{0x_2} denote the remanent inductions of the permanent magnet in x_1 and in x_2 direction, respectively.

Because of the complicate inner geometry no structured grids for discretizing this domain are available. Moreover, as it can be seen already in Figure 8 we hint at the fact that by the automatical mesh generator in [17] the unstructured mesh can be initially adapted to the given point singularities.

			artYs		artBPX	
J	N	L^2	struct. grid	unstr. grid	struct. grid	unstr. grid
8	469	66049		57 (4.92)		42 (3.14)
9	1831	263169		72 (20.25)		51 (19.32)
10	7237	1050625		87 (79.73)		63 (92.45)
11	28777	4198401		109 (650.67)		103 (657.80)
12	114769	16785409		155 (3699.0)		mem. ex.
R^*:				27		23
R:				23		20
$C_{\Pi^h}^{-1}$:				50		57

Table 7: #cg–iterations and CPU–times for the magnetic field problem

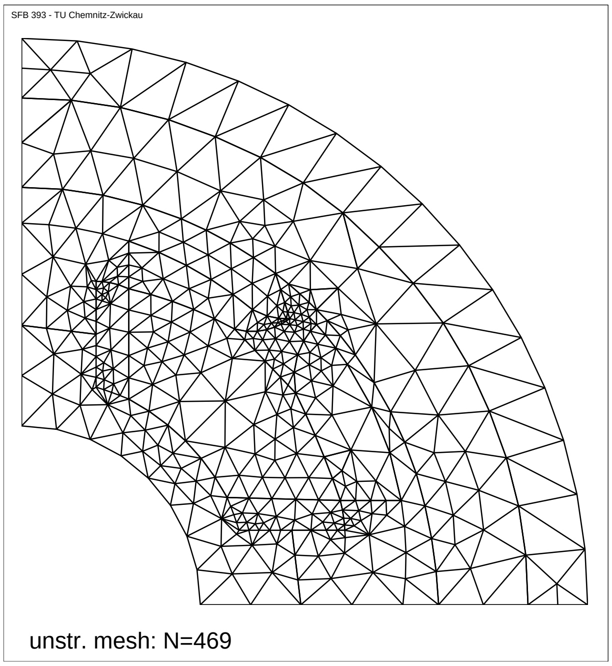


Figure 8: The adaptive mesh of the motor's fourth losing the quasiuniformity

For problems with inhomogeneous coefficient functions $a_{ij}(x)$, $i,j = 1,2$, having jumps we used the "outer" Jacobi–preconditioning in point **5.** of the previous section. Moreover, if all of the interfaces defined by the different material properties coincide with edges of the auxiliary mesh Π^h no difficulties occured when using the auxiliary defined "inner" Jacobi–preconditioning nevertheless. Hence, we may conclude that this "inner" Jacobi–preconditioner becomes inadequately disturbed when the interface approximation is performed in the form of steps. Therefore, we propose the shifting of appropriately choosen nodes of the domain Q^h to the interfaces to overcome the described difficulties in an other way which may be even more successful. Computing the inhomogeneous problems the weeker increasing of the iteration numbers starting at a certain stage of J (observed e.g. in Table 4) is due to the better approximation of the interfaces as it is made more precisely in the form of steps.

6.2 First results of the parallel computing

To get the results of the subsection we used the well known Parsytec parallel computer GCPowerPlus having 32MByte memory at each processor node and a peak performance of 80MFlop. The programs are called "pmhi.artYs.ppc.px" in the case of the artificially performed Yserantant hierarchical preconditioning and "pmhi.artBPX.ppc.px" in the BPX–case, respectively. For more details describing the related software tools see also [1, 2, 14]. The next three examples are computed using 16 processors in each case. The domain decompositions used to be the basis of the parallelization in the case of the "structured grid"–calculation are given according to the meshes presented in the left part of the Figures 3 and 4, respectively. Computing in the square we have the 16 subsquares consisting of the two initial triangles shown in Figure 3. Computing in the club shaped domain we have the 16 subtraingles given in the leftbelow of Figure 4 to be the subdomains for the DD–based parallelism. For the parallelization of the "real unstructured grid"–computations, especially in the case of the magnetic field calculation, we applied the FE–data distribution of the meshes after their generation was done by the parallel mesh generator in [11]. Here, the parameter L^2 is the sum $\sum_{s=1}^{p} L_s^2$ and for J holds $J = \max(J_s)$, $s = 1, \ldots, p$.

1. *The decomposed problem no.* **1.** *of the previous subsection:*

J	N	L^2	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
1/4	25	186	[10 (0.00)] 10 (1.07)	16 (0.34)	[9 (0.00)] 16 (0.13)	16 (1.10)
2/5	81	400	[13 (0.18)] 19 (1.72)	26 (0.69)	[11 (0.20)] 23 (0.43)	30 (2.05)
3/6	289	1296	[17 (0.22)] 24 (1.75)	41 (1.93)	[13 (0.26)] 31 (0.60)	40 (3.01)
4/7	1089	4624	[20 (0.28)] 28 (1.83)	55 (6.21)	[14 (0.30)] 42 (0.87)	53 (14.20)
5/8	4225	17424	[24 (0.36)] 31 (1.95)	70 (11.50)	[15 (0.37)] 53 (1.22)	69 (16.51)
6/9	16641	67600	[26 (0.47)] 34 (2.81)	109 (22.86)	[15 (0.47)] 73 (2.81)	75 (43.20)
7/–	66049	266256	[28 (1.00)] 35 (4.41)	mem. ex.	[15 (0.82)] 114 (11.63)	mem. ex.
8/–	263169	1056784	[29 (2.92)] 36 (12.15)	mem. ex.	[15 (2.00)] mem.ex.	mem. ex.
R^*:				25		23
R:				22		22
$C_{\Pi^h}^{-1}$:				53		55

Table 8: #cg–it. (CPU) for problem **1.** (16 subsquares and data distribution, resp.)

2. *The decomposed problem no.* **2.** *of the previous subsection:*

J	N	L^2	artYs struct. grid	artYs unstr. grid	artBPX struct. grid	artBPX unstr. grid
2/4	16	272	[10 (0.00)] 16 (0.07)	8 (0.00)	[6 (0.00) 6 (0.29)	12 (0.07)
3/3	47	848	[15 (0.19)] 22 (0.38)	23 (0.44)	[14 (0.24)] 19 (0.32)	25 (0.41)
4/6	157	2960	[19 (0.24)] 37 (0.67)	37 (1.24)	[17 (0.31) 38 (0.68)	26 (0.95)
5/7	569	11024	[22 (0.29)] 56 (1.18)	59 (4.19)	[19 (0.37) 55 (1.15)	42 (4.30)
6/8	2161	42512	[26 (0.37)] 90 (2.81)	87 (17.25)	[19 (0.41) 71 (2.31)	55 (12.76)
7/9	8417	166928	[28 (0.43)] 152 (10.10)	145 (111.75)	[19 (0.51) 84 (6.61)	mem. ex.
8/–	33217	661520	[30 (0.71)] 265(57.68)	mem. ex.	[19 (0.72)] 125 (32.59)	mem. ex.
9/–	131969	2633744	[31 (1.77)] 350(268.88)	memory exceeded		
R^*:				20		18
R:				11		10
$C_{\Pi^h}^{-1}$:				69		72

#cg–it. (CPU) for problem **2.** (16 subtriangles and data distribution, resp.)

3. *The parallel magnetic field computation in the fourth of the motor:*

			artYs		artBPX	
J	N	L^2	struct. grid	unstr. grid	struct. grid	unstr. grid
8	469	208591		67 (17.20)		62 (13.62)
9	1831	566735		115 (85.97)		93 (79.50)
10	7237	828815		165 (124.58)		mem. ex.
R^*:				25		21
R:				23		19
$C^{-1}_{\Pi^h}$:				52		60

Table 10: #cg–it. and CPU for problem **5.**, where data distribution was made

Finally, let us give the following remarks comparing the results of the three tables presented here. If all of the subdomain meshes Ω^h_s, $s = 1, \ldots, p$, into which the whole mesh Ω^h is decomposed coincide with the auxiliary square grids Π^h_s, $s = 1, \ldots, p$, the computation in parallel is very efficient as it was expected, see Table 8. Otherwise, the step form approximation of the coupling boundaries defined by the domains Q^h_s which are subsets of the overlapped grids Π^h_s, $s = 1, \ldots, p$, detoriates the convergence of the preconditioned parallel cg–method substantially. In comparison with the results of the parallel artYs–method the more bad iteration numbers of the parallel version of the artBPX are caused by the weekness that in the latter case up to now the communication is only performed w.r.t. the coupling nodes assigned to the finest level zone of the artificial BPX–list.
We are seeking for the remedy to recover the fast convergence of the artificially preconditioned cg–methods also in the general case of their parallelization.

References

[1] T. Apel, SPC-PM Po 3D — User's manual. Preprint SPC 95_33, TU Chemnitz-Zwickau, December 1995.

[2] T. Apel, F. Milde, and M. Theß, SPC-PM Po 3D — Programmer's manual. Preprint SPC 95_34, TU Chemnitz–Zwickau, December 1995.

[3] G.P. Astrachanzev, Fictitious domain method for the second-order elliptic equation with natural boundary conditions. Zh. Vychisl. Mat. Mat. Fiz., **18** (1978), pp. 118–125.

[4] J.-P. Aubin, *Approximation of Elliptic Boundary–value problems.* Wiley-Interscience, New York – London – Sydney – Toronto, 1972.

[5] R.E. Bank and J. Xu, An algorithm for coarsening unstructured meshes. Numer. Math, to appear.

[6] R.E. Bank and J. Xu, The hierarchical basis multigrid method and incomplete LU decomposition. Contemporary Mathematics, **180** (1994), pp. 163–174.

[7] F.A. Bornemann and H. Yserentant, A basic norm equivalence for the theory of multilevel methods. Numer. Math., **64** (1993), pp. 455–476.

[8] J.H. Bramble, J.E. Pasciak and J. Xu, Parallel multilevel preconditioners. Math. Comp., **55** (1990), pp. 1-22.

[9] T.F. Chan and B.F. Smith, Domain Decomposition and Multigrid algorithms for elliptic problems on unstructured meshes. Contemporary Mathematics, **180** (1994), pp. 175–190.

[10] Ph. Ciarlet, *The Finite Element Method for Elliptic Problems.* North–Holland, Amsterdam, 1977.

[11] G. Globisch, PARMESH – a parallel mesh generator. Parallel Computing, **21**, no. 3 (1995), pp. 509–524.

[12] G. Globisch, Robuste Mehrgitterverfahren für einige elliptische Randwertaufgaben in zweidimensionalen Gebieten. Technische Universität Chemnitz, Dissertation, Chemnitz, 1993.

[13] G. Haase, U. Langer and A. Meyer, Parallelisierung und Vorkonditionierung des CG–Verfahrens durch Gebietszerlegung. in: G. Bader, R. Rannacher, G. Wittum (eds.), Numerische Algorithmen auf Transputer–Systemen, Teubner-Skripten zur Numerik, Teubner–Verlag, Stuttgart 1992.

[14] G. Haase, T. Hommel, A. Meyer and M. Pester, Bibliotheken zur Entwicklung paralleler Algorithmen. Preprint SPC 95_20, TU Chemnitz-Zwickau, June 1995.

[15] B. Heise, Analysis of a fully discrete finite element method for a nonlinear magnetic field problem. SIAM J. Numer. Anal., **31**, no. 3 (1994), pp. 745–759.

[16] M. Jung, Parallelization of multi-grid methods based on domain decomposition ideas. Preprint SPC 95_27, TU Chemnitz-Zwickau, November 1995.

[17] R. Kornhuber and H. Yserentant, Multilevel methods for elliptic problems on domains not resolved by the coarse grid. in: D.E. Keyes, J. Xu (eds.), Domain decomposition for PDEs, Contemporary Mathematics, **180** (1994), pp. 49–60.

[18] A.M. Matsokin, Extension of mesh functions with norm-preserving. in: Variational Methods of Numerical Analysis, Comp. Centre, Siberian Branch of Acad. Sci. of the USSR, Novosibirsk, 1986, pp. 111–132 (in Russian).

[19] A.M. Matsokin, Solution of grid equations on non-regular grids. Preprint No. 738, Comp. Centre, Siberian Branch of Acad. Sci. of the USSR, Novosibirsk, 1987 (in Russian).

[20] A.M. Matsokin and S.V. Nepomnyaschikh, The fictitious domain method and explicit continuation operators, Zh. Vychisl. Mat. Mat. Fiz., **33** (1993), pp. 45–59.

[21] A. Meyer, A parallel preconditioned conjugate gradient method using domain decomposition and inexact solvers on each subdomain. Computing, **45** (1990), pp. 217–234.

[22] A. Meyer and M. Pester, Verarbeitung von Sparse-Matrizen in Kompaktspeicherform KLZ/KZU. Preprint SPC 94_12, TU Chemnitz-Zwickau, June 1994.

[23] S.V. Nepomnyaschikh, Fictitious space method on unstructured meshes. East–West J. Numer. Math., **3** (1995), no. 1, pp. 71–79.

[24] S.V. Nepomnyaschikh, Mesh theorems of traces, normalization of function traces and their inversion. Sov. J. Numer. Anal. Math. Model., **6** (1991), no. 3, pp. 223–242.

[25] S.V. Nepomnyaschikh, Method of splitting into subspaces for solving elliptic boundary value problems in complex-form-domains. Sov. J. Numer. Anal. Math. Model., **6** (1991), no. 2, pp. 151–168.

[26] S.V. Nepomnyaschikh, Optimal multilevel extension operators. Preprint SPC 95_3, TU Chemnitz-Zwickau, March 1995.

[27] S.V. Nepomnyaschikh, Preconditioning operators on unstructured grids. Preprint No. 178, Weierstraß–Institut für Angewandte Analysis und Stochastik, ISSN 0946–8633, Berlin 1995.

[28] L.A. Oganesyan and L.A. Ruchovets, *Variational Difference Methods for Solving Elliptic Equations.* Izdat. Akad. Nauk Arm. SSR, Erevan, 1979 (in Russian).

[29] P. Oswald, *Multilevel Finite Element Approximation: Theory and Application.* B.G. Teubner, Stuttgart, 1994.

[30] M. Pester and S. Rjasanow, A parallel version of the preconditioned conjugate gradient method. Preprint SPC 93_2, TU Chemnitz-Zwickau, June 1993. Journal of Num. Lin. Alg. with Appl. **6**(1) 1994.

[31] W. Queck (ed.), FEMGP (Finite Element Multigrid Package). Programmdokumentation, Technologieberatungszentrum Parallele Informationsverarbeitung GmbH (TBZ*PARIV), Bernsdorfers Str. 210–212, D–09126 Chemnitz, 1993.

[32] J. Xu, Iterative methods by space decomposition and subspace correction. SIAM Review, **34** (1992), no. 4, pp. 581–613.

[33] J. Xu, The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. (submitted to Computing).

[34] G.N. Yakovlev, On traces of piecewise smooth surfaces of functions from the space W_p^l. Mat. Sbornik **74** (1967), pp. 526–543

[35] H. Yserentant, On the multi–level splitting of finite element spaces. Numer. Math. **49** (1986), pp. 379–412.

Other titles in the SFB393 series:

96-01 V. Mehrmann, H. Xu. Chosing poles so that the single-input pole placement problem is well-conditioned. Januar 1996.

96-02 T. Penzl. Numerical solution of generalized Lyapunov equations. January 1996.

96-03 M. Scherzer, A. Meyer. Zur Berechnung von Spannungs- und Deformationsfeldern an Interface-Ecken im nichtlinearen Deformationsbereich auf Parallelrechnern. March 1996.

96-04 Th. Frank, E. Wassen. Parallel solution algorithms for Lagrangian simulation of disperse multiphase flows. Proc. of 2nd Int. Symposium on Numerical Methods for Multiphase Flows, ASME Fluids Engineering Division Summer Meeting, July 7-11, 1996, San Diego, CA, USA. June 1996.

96-05 P. Benner, V. Mehrmann, H. Xu. A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils. April 1996.

96-06 P. Benner, R. Byers, E. Barth. HAMEV and SQRED: Fortran 77 Subroutines for Computing the Eigenvalues of Hamiltonian Matrices Using Van Loans's Square Reduced Method. May 1996.

96-07 W. Rehm (Ed.). Portierbare numerische Simulation auf parallelen Architekturen. April 1996.

96-08 J. Weickert. Navier-Stokes equations as a differential-algebraic system. August 1996.

96-09 R. Byers, C. He, V. Mehrmann. Where is the nearest non-regular pencil? August 1996.

96-10 Th. Apel. A note on anisotropic interpolation error estimates for isoparametric quadrilateral finite elements. November 1996.

96-11 Th. Apel, G. Lube. Anisotropic mesh refinement for singularly perturbed reaction diffusion problems. November 1996.

96-12 B. Heise, M. Jung. Scalability, efficiency, and robustness of parallel multilevel solvers for nonlinear equations. September 1996.

96-13 F. Milde, R. A. Römer, M. Schreiber. Multifractal analysis of the metal-insulator transition in anisotropic systems. October 1996.

96-14 R. Schneider, P. L. Levin, M. Spasojević. Multiscale compression of BEM equations for electrostatic systems. October 1996.

96-15 M. Spasojević, R. Schneider, P. L. Levin. On the creation of sparse Boundary Element matrices for two dimensional electrostatics problems using the orthogonal Haar wavelet. October 1996.

96-16 S. Dahlke, W. Dahmen, R. Hochmuth, R. Schneider. Stable multiscale bases and local error estimation for elliptic problems. October 1996.

96-17 B. H. Kleemann, A. Rathsfeld, R. Schneider. Multiscale methods for Boundary Integral Equations and their application to boundary value problems in scattering theory and geodesy. October 1996.

96-18 U. Reichel. Partitionierung von Finite-Elemente-Netzen. November 1996.

96-19 W. Dahmen, R. Schneider. Composite wavelet bases for operator equations. November 1996.

96-20 R. A. Römer, M. Schreiber. No enhancement of the localization length for two interacting particles in a random potential. December 1996. to appear in: Phys. Rev. Lett., March 1997

96-21 G. Windisch. Two-point boundary value problems with piecewise constant coefficients: weak solution and exact discretization. December 1996.

96-22 M. Jung, S. V. Nepomnyaschikh. Variable preconditioning procedures for elliptic problems. December 1996.

97-01 P. Benner, V. Mehrmann, H. Xu. A new method for computing the stable invariant subspace of a real Hamiltonian matrix or Breaking Van Loan's curse? January 1997.

97-02 B. Benhammouda. Rank-revealing 'top-down' ULV factorizations. January 1997.

97-03 U. Schrader. Convergence of Asynchronous Jacobi-Newton-Iterations. January 1997.

97-04 U.-J. Görke, R. Kreißig. Einflußfaktoren bei der Identifikation von Materialparametern elastisch-plastischer Deformationsgesetze aus inhomogenen Verschiebungsfeldern. March 1997.

97-05 U. Groh. FEM auf irregulären hierarchischen Dreiecksnetzen. March 1997.

97-06 Th. Apel. Interpolation of non-smooth functions on anisotropic finite element meshes. March 1997

97-07 Th. Apel, S. Nicaise. The finite element method with anisotropic mesh grading for elliptic problems in domains with corners and edges.

97-08 L. Grabowsky, Th. Ermer, J. Werner. Nutzung von MPI für parallele FEM-Systeme. March 1997.

The complete list of current and former preprints is available via
`http://www.tu-chemnitz.de/sfb393/sfb97pr.html`.