Institut für Angewandte Analysis
und Stochastik

im Forschungsverbund Berlin e.V.

# Nyström's method and iterative solvers for the solution of the double layer potential equation over polyhedral boundaries

B. Kleemann, A. Rathsfeld

submitted: 1st February 1993

Institut für Angewandte Analysis
und Stochastik
Hausvogteiplatz 5-7
D – O 1086 Berlin
Germany

Preprint No. 36
Berlin 1993

---

*1991 Mathematics Subject Classification.* 45 L 10, 65 R 20, 65 F 10.
*Key words and phrases.* potential equation, Nyström's method, two–grid iteration.

This work was supported in part by Deutsche Forschungsgemeinschaft under grant number Pr 336/2–1.

Herausgegeben vom
Institut für Angewandte Analysis und Stochastik
Hausvogteiplatz 5-7
D – O 1086 Berlin

Fax: + 49 30 2004975
e-Mail (X.400): c=de;a=dbp;p=iaas-berlin;s=preprint
e-Mail (Internet): preprint@iaas-berlin.dbp.de

**Abstract.** In this paper we consider a quadrature method for the solution of the double layer potential equation corresponding to Laplace's equation in a three-dimensional polyhedron.We prove the stability for our method in case of special triangulations over the boundary of the polyhedron. For the solution of the corresponding system of linear equations, we consider a two-grid iteration and a further simple iteration procedure. Finally, we establish the rates of convergence and complexity and discuss the effect of mesh refinement near the corners and edges of the polyhedron.

## 1. Introduction.

One popular method for the solution of boundary value problems for elliptic differential equations consists in the reduction to boundary integral equations. For instance, the Dirichlet problem for Laplace's equation in a bounded and simply connected polyhedron $\Omega \subseteq \mathbf{R}^3$ or the Neumann problem for the same equation on $\mathbf{R}^3 \setminus \Omega$ can be reduced to the second kind integral equation $Ax = y$ over the boundary $S := \partial\Omega$ (cf. e. g. [22] ), where $A = I + 2W$ and

$$Wx(Q) := [1/2 - d_\Omega(Q)]x(Q) + \frac{1}{4\pi}\int_S \frac{n_P \cdot (Q-P)}{|P-Q|^3} x(P) d_P S, \tag{1.1}$$

$$d_\Omega(Q) := \lim_{\epsilon \to \infty} \frac{\mu(\{P \in \Omega : |P-Q| < \epsilon\})}{\mu(\{P \in \mathbf{R}^3 : |P-Q| < \epsilon\})}.$$

Here $n_P$ denotes the unit vector of the interior normal to $\Omega$ at $P$ and $\mu(Z)$ is the Lebesgue measure of $Z$. Note that, since the boundary $S$ is not smooth, $W$ is not compact. The kernel function $k(Q,P) := \frac{1}{4\pi} n_P \cdot (Q-P)|P-Q|^{-3}$ vanishes if $P$ and $Q$ lie on the same face of $S$. However, if $P$ and $Q$ tend to an edge point of $S$ and lie on different faces, then $k(Q,P)$ is of order $|P-Q|^{-2}$. Thus the kernel function of $W$ has a fixed singularity at the set of edge points.

For the numerical solution of $Ax = y$, various methods have been introduced. The first method was the so called panel method, i. e., piecewise constant collocation (cf.[33, 13, 34, 2]). This method has been proved to converge if $\Omega$ satisfies the condition introduced by Wendland in [33] (cf. (3.2)). Moreover, Kral and Wendland [16] (cf. also [17, 1]) have shown that the

panel method is stable for the case of certain rectangular domains $\Omega$. Arbitrary polyhedral domains have been considered in [26]. Elschner [9] has analysed the Galerkin method with piecewise polynomial trial functions over arbitrary polyhedrons, and the Galerkin method together with an approximation of the Lipschitz boundary by smooth surfaces has been investigated by Dahlberg and Verchota [7]. In [24] a quadrature method has been considered which is similar to the methods of Graham and Chandler [6], Kress [18], and Elschner [8] for the corresponding equation over polygonal boundaries. The advantage of quadrature methods consists in the full discretization of the equation. For other discretization schemes, a further discretization step is needed in order to compute the entries of the stiffness matrix. The analysis of this step, however, is still uncompleted. The first analysis of an iterative solution is due to Wendland [33] who has considered a discretized version of Neumann's iteration. Multi-grid methods for the solution of the arising linear systems have been analysed by Schippers [29, 30], Atkinson, Graham [4, 5], and the author [23] for the one-dimensional case as well as by Hebeker [12] and Atkinson [3] for the two-dimensional case (cf. also [11, 10]). The main point in the convergence proof is roughly speaking the following: Split the discretized operator into the sum of two parts, where the first part is the discretized double layer operator restricted to a neighbourhood of the set of edge points. After multiplying the whole discretized equation by the inverse of this first part, a second kind equation with discretized compact operator arises. For this situation, the well-known theory of multi-grid methods applies. The drawback of these methods, however, is that the first part of the discretized operator is to be inverted. Since this step means the inversion of a large matrix, no improvement of the asymptotic order of complexity can be achieved. Therefore, the authors of [30, 3] recommend better variants of multi-grid procedures for which no convergence proofs are known yet.

The aim of the present paper is to analyse a simple quadrature method which was mentioned in [24] without proof. In comparison with the methods investigated in [24] it is much easier since it is based on a different type of the so-called method of singularity subtraction. More exactly, in [24] the integral $\int k(Q,P)x(P)dP$ has been written as $\int k(Q,P)[x(P)-x(R)]dP + x(R)\int k(Q,P)dP$, and the quadrature rule has been applied to $\int k(Q,P)[x(P)-x(R)]dP$. The point $R$ has been chosen from the set of edge or corner points in an appropriate way depending on $Q$. In the quadrature method of the present paper we easily take $R = Q$. This choice enables us to prove stability without the so-called localization technique and to deduce the convergence of a simple two-grid iteration scheme analogously to the one-dimensional case (cf. [5, 23]). We remark that our subtraction technique may be advantageous also if additional terms with discontinuities along the diagonal are added to the kernel function. These additional terms arise if one considers domains with curved boundaries or boundary integral equations corresponding to the Helmholtz equation. In detail we shall describe the Nyström method and its iterative solution in Sect.2. The first iteration scheme is just the classical two-grid method (cf. [11, 10]), where Jacobi's iteration is used for the smoothing step and Nyström's interpolation for the restriction and prolongation. However, when applied to our double layer operator, it shows the same new features as detected in the case of one-dimensional boundaries (cf. Sect.4 and [4, 5, 23]). Namely, the convergence ratio of the iteration process does not tend to zero if the mesh size of the fine and coarse grid tends to zero. The convergence ratio rather depends on another property of the grids ( cf. (3.3)

and (4.2)). In particular, the ratio can be improved by cutting off a larger neighbourhood of the set of edge points (i.e., by choosing a strip with a larger $\epsilon$ in (2.2), where $\epsilon$ is still of the order of the mesh size). Beside the two-grid iteration for the solution of the arising linear system of equations, we also define an iteration scheme which corresponds to the Neumann series expansion. This iteration has been first introduced by Wendland [33] for the case of the panel method. In Sect.3 we prove the stability of the Nyström method. It turns out that our quadrature method is stable if a certain finite section procedure (cf. (2.2)) is convergent. For instance, if Wendland's condition (cf. [33] or (3.2)) is satisfied or if the corners of $\Omega$ are rectangular and graphs of Lipschitz functions, then this finite section procedure is convergent and our Nyström method is stable. We note that the quadrature method in [24] is based on a different type of finite section technique, and its stability can be proved also for rectangular non-Lipschitz corners. The convergence of the iteration schemes will be analysed in Sect.4. We shall show that the two-grid iteration is convergent if the Nyström method is stable. The Neumann iteration converges if Wendland's condition (cf. [33] or (3.2)) is satisfied. In Sect.5 we shall introduce special non-uniform triangulations and obtain the same asymptotic error estimates as in [24]. From these estimates we deduce the complexity[1], i.e., the number of necessary operations to compute the approximate solution with an error less than a prescribed small number. If $N$ is the number of linear equations of the Nyström method, then one needs $O(N^3)$ operations to solve the system of equations by Gaussian elimination. This order can be reduced to $O(N^2)$ if one applies the iteration scheme corresponding to the Neumann series expansion. However, for domains $\Omega$ with complicated geometry, $O(N^2)$ means an estimate by constant times $N^2$, where the constant may be large. The application of the two-grid iteration over quasi-uniform grids provides an order $O(N^2)$ with smaller constant. Unfortunately, for highly graded meshes, the two-grid method has the same asymptotic order of complexity as Gaussian elimination, and the number of necessary operation reduces by a constant factor, only. In other words, comparing the Nyström method over an "optimally" graded mesh solved by Gaussian elimination with the Nyström method over an almost uniform grid solved by two-grid iteration, we get the same asymptotic order of complexity. However, we expect the two-grid method to have a smaller constant in the complexity estimate. Comparing the two-grid method and Gaussian elimination for the same mesh grading parameter, the complexity of the two-grid iteration is less if $N$ is sufficiently large. In Sect. 6 we present numerical experiments in order to confirm our theoretical results. In particular, we compare our iterations with the Gaussian algorithm, the GMRES ([28, 32, 14, 31]), and the GMRES with two-grid preconditioner. Remark that throughout the paper we have chosen the composite midpoint rule as the quadrature formula. Higher order rules can be treated similarly (cf. [24]), but they do not improve the asymptotic order of convergence. In Sect. 7 we give the sequential code of the two-grid method. Finally, in Sect. 8 we discuss some aspects of the parallel implementation and compare the CPU-times of vector and parallel (SIMD) computations.

[1] Since we shall discuss iterative algorithms for linear systems where the size depends on the mesh refinement, the convergence order depending on the number of degrees of freedom is not sufficient to measure the quality of our methods.

## 2. The quadrature method and the iteration procedures.

**2. 1. Derivation of the Nyström method.** Let us start with the singularity subtraction. Taking into account that the constant function is an eigenfunction of $W$ corresponding to the eigenvalue $1/2$ (cf. [22], Sect.1.3), the equation $Ax = y$ may be written as

$$2x(Q) + \frac{1}{2\pi}\int_S \frac{n_P \cdot (Q-P)}{|Q-P|^3}[x(P) - x(Q)]d_P S = y(Q),\ Q \in S. \tag{2.1}$$

Now the next step in the discretization of (2.1) is the finite section method. We take a small strip $Str \subset S$ of width $\epsilon > 0$ around the edges and set $S_\epsilon = S \setminus Str$. We suppose that $\epsilon$ is of the same size as the diameters of the subdomains in the partition used for the quadrature rule. Instead of (2.1) we consider [2]

$$2x_\epsilon(Q) + \frac{1}{2\pi}\int_{S_\epsilon} \frac{n_P \cdot (Q-P)}{|Q-P|^3}[x_\epsilon(P) - x_\epsilon(Q)]d_P S = y(Q),\ Q \in S. \tag{2.2}$$

Now we take a triangulation $S_\epsilon = \cup_{i=1}^N S^i$ of $S_\epsilon$. We denote the centre of the triangle $S^i$ by $P^i$ and the surface measure of $S^i$ by $\mu(S^i)$. In order to get a quadrature method for (2.2) we replace the integral by a quadrature rule. Thus the Nyström method consists in solving

$$2x_N(Q) + \frac{1}{2\pi}\sum_{i=1}^N \frac{n_{P^i} \cdot (Q-P^i)}{|Q-P^i|^3}[x_N(P^i) - x_N(Q)]\mu(S^i) = y(Q),\ Q \in S. \tag{2.3}$$

As it is well known, the solution of (2.3) can be obtained in two steps. First one has to solve

$$2x_N(P^j) + \frac{1}{2\pi}\sum_{i=1}^N \frac{n_{P^i} \cdot (P^j-P^i)}{|P^j-P^i|^3}[x_N(P^i) - x_N(P^j)]\mu(S^i) = y(P^j),\quad j = 1,\dots,N, \tag{2.4}$$

where, taking into account that $n_{P^i} \cdot (P^j - P^i) = 0$ for $P^j$ and $P^i$ on the same face of $S$, we have set $n_{P^j} \cdot (P^j - P^j)|P^j - P^j|^{-3} := 0$. Using the solution $x_N(P^j)$ of (2.4), the solution $x_N$ of (2.3) is given by Nyström's interpolation

$$x_N(Q) = \frac{y(Q) - \frac{1}{2\pi}\sum_{i=1}^N \frac{n_{P^i} \cdot (Q-P^i)}{|Q-P^i|^3} x_N(P^i)\mu(S^i)}{2 - \frac{1}{2\pi}\sum_{i=1}^N \frac{n_{P^i} \cdot (Q-P^i)}{|Q-P^i|^3}\mu(S^i)},\ Q \in S. \tag{2.5}$$

[2] Note that the transition to this finite section equation will play an important role for the stability proof in Sect.3. Since we truncate the boundary before dividing it into subdomains, we get a smaller number of linear equations. The additional approximation error is of the same asymptotic order as the discretization error without finite section. However, numerical tests show stability and smaller errors for the Nyström method without finite section step.

The denominator in (2.5) is different from 0 (cf. the beginning of the proof to Theorem 3.1).

**2. 2. The two-grid method.** Now we need some notation in order to introduce the two-grid method. Let us follow [11] and denote the operators on the left-hand side of (2.2) and (2.3) by $A_\epsilon$ and $A_N$, respectively. Further, let $\chi$ stand for the characteristic function of $S_\epsilon$ and $\chi I$ for the operator of multiplication by $\chi$. Then $A_\epsilon$ and $A_N$ take the form (cf.(1.1),(2.2) and (2.3))

$$A_\epsilon = 2[1 - W(\chi)]I + 2W\chi I, \quad A_N = 2[1 - W_N(\chi)]I + 2W_N\chi I, \tag{2.6}$$

where

$$W_N(\chi x)(Q) := \frac{1}{4\pi}\sum_{i=1}^{N} \frac{n_{P^i}\cdot(Q-P^i)}{|Q-P^i|^3}\mu(S^i)x(P^i) \tag{2.7}$$

and $[1 - W_N(\chi)]I$ as well as $[1 - W(\chi)]I$ stand for the operators of multiplication by the functions $[1 - W_N(\chi 1)]$ and $[1 - W(\chi 1)]$, respectively. Beside the triangulation $S_\epsilon = \cup_{i=1}^{N} S^i$ we consider a coarse grid $S_\epsilon = \cup_{i=1}^{N_c} S_c^i$. Let $P_c^i$ stand for the midpoint of $S_c^i$. We denote the corresponding discretized double layer operator $W_N\chi I$ over the coarse grid by $W_{N_c}\chi I$. Our aim is to solve $A_N x_N = y$ by a two-grid iteration process. Therefore, we start with an arbitrary initial function $x^0$ (e. g. $x^0 = 0$) and compute the approximate solutions $x^i$ for $x_N$ by iteration. Suppose we have obtained $x^{i-1}$. Then $x^i$ will be determined as follows (cf.[11]): We start with the smoothing step

$$x' := \frac{1}{2}[1 - W_N(\chi)]^{-1}\{y - 2W_N(\chi x^{i-1})\} \tag{2.8}$$

and define the residual (defect)

$$d := y - 2[1 - W_N(\chi)]x' - 2W_N(\chi x'). \tag{2.9}$$

The correction term $c$ is the solution of the coarse grid equation[3]

$$A_{N_c}c := 2[1 - W_N(\chi)]c + 2W_{N_c}(\chi c) = d. \tag{2.10}$$

Finally, the approximate solution $x^i$ of the i-th iteration step is given by

$$x^i := x' + c. \tag{2.11}$$

[3] Note that we could have set also $A_{N_c} := 2[1 - W_{N_c}(\chi)]I + 2W_{N_c}\chi I$.

Let us note that there exists a faster algorithm for the computation of $x^i$ due to Atkinson (cf.[5]). Moreover, the algorithm becomes still faster if Nyström's interpolation in the prolongation of the coarse grid data to the fine grid is replaced by piecewise constant interpolation. We shall describe the details in item d) of Sect.6.3.

In Sect. 4 we shall show that $x^i \to x_N$ and $\|x^i - x_N\| \leq q^i \|x^0 - x_N\|$ holds with $0 < q < 1$, where $q$ depends only on the parameter $\epsilon_p$ appearing in the stability conditions for the quadrature method (cf.(3.3),(4.2)). Moreover, we derive the orders of complexity in Sect.5.2. Finally, we remark that there is a possibility to avoid the finite section step (2.2) without loosing the convergence estimates of Sect.4.1. Instead of throwing away the strip $Str$ we may choose the coarse grid equal to the fine grid over $Str$ (cf. the first remark following Theorem 4.1). In the case of one-dimensional boundaries this iteration is analysed in [23].

**2. 3. The Neumann iteration.** Since a coarse grid is used in (2.10), the computational work for the two-grid iteration is less than that for a direct method of solution of (2.4). This will be satisfactory if the finer triangulation is nearly uniform (cf. Sect.5.2). However, if we take into account the singular behaviour of the solution $x = (I + 2W)^{-1}y$ and that of the kernel function $k$ corresponding to the integral operator $W$, then we have to use a strongly non-uniform mesh. In this case the stability condition for the triangulation (cf.(3.3),(4.2)) implies that, from the point of view of asymptotic order, the number of triangles in the coarse grid is nearly equal to the number of triangles of the fine grid. Therefore, we consider another iteration process for the solution of (2.4). This iteration was proposed by Wendland [33] for the case of piecewise constant collocation and is nothing else than a discretized version of Neumann's iteration. We start with $x^0 = 0$, choose a parameter $\kappa > 1$ (cf. the remarks after Theorem 4.2) and write the equation $A_N x_N = y$ in the equivalent form $\{I + \frac{1}{\kappa}(A_N - \kappa I)\}x_N = \frac{1}{\kappa}y$. The iterative solutions are defined by $x^i := \frac{1}{\kappa}y - \frac{1}{\kappa}(A_N - \kappa I)x^{i-1}$, i.e.,

$$x^i := x^{i-1} + \frac{1}{\kappa}\left\{y - 2[1 - W_N(\chi)]x^{i-1} - 2W_N(\chi x^{i-1})\right\}, \quad i = 1, 2, \ldots . \tag{2.12}$$

In Sect.5.3 we shall prove that $x^i \to x_N$ and $\|x^i - x_N\| \leq Cq^i\|x^0 - x_N\|$ holds with $0 < q < 1$, where $q$ depends on the geometry of $\Omega$. Note that the two-grid iteration may be preferable if $q$ is close to 1.

## 3. The stability of method (2.3).

Let us consider the space $C(S)$ of continuous functions over $S$ supplied with the supremum norm $\|\cdot\|$. By $\|\cdot\|$ we also denote the operator norm on $C(S)$. We call (2.3) stable if $A_N$ is invertible and $\|A_N^{-1}\| \leq C = constant$ with $C$ independent of the partition $S_\epsilon = \cup S^i$. Let us introduce the constant

$$C_1 := \sup_{Q \in S} \frac{1}{4\pi} \int_S \frac{|n_P \cdot (P - Q)|}{|P - Q|^3} d_P S < \infty.$$

By $C$ let us denote a generic positive constant which varies from instance to instance.For

fixed $i$, let $E^i$ denote the union of all faces on $S$ which do not contain the triangle $S^i$. Further, set $rad\, S^i := \sup\{|P - P^i|, P \in S^i\}$ and let $C_{fs} > 0$. In order to prove stability, we shall assume:

(3.1) The operator $A_\epsilon$ is invertible and $\|A_\epsilon^{-1}\| \leq C_{fs}$.

Note that $A_\epsilon$ is the finite section operator defined by the left-hand side of (2.2). Thus assumption (3.1) means the stability of the finite section method in the second step of the discretization of $A$. This finite section method is analysed in [25], where it has been proved that (3.1) holds if and only if certain double layer operators defined over the tangent cones are invertible. Sufficient for (3.1) is that each corner is rectangular and the graph of a Lipschitz function [4]. Another sufficient condition is that

$$\limsup_{r \longrightarrow 0} \sup_{Q \in S} \frac{1}{2\pi} \int_{\{P \in S : |P-Q| \leq r\}} \frac{|n_P \cdot (Q-P)|}{|P-Q|^3} d_P S \; < \; 1. \tag{3.2}$$

This assumption was introduced also in [33] and can be checked using the fact that the integral of the double layer kernel is a solid angle (cf. [22], Sect.4.2.2). In particular, (3.2) and therewith (3.1) hold for convex domains $\Omega$. Now we have

THEOREM 3.1. *Suppose (3.1) holds. Then there is a $C$ and an $\epsilon_p$ $(1/2 > \epsilon_p > 0)$ depending on $C_{fs}$ and $C_1$ such that $\|A_N^{-1}\| \leq C$ holds whenever*

$$\frac{rad\, S^i}{dist(S^i, E^i)} \leq \epsilon_p, \quad i = 1, \ldots, N \tag{3.3}$$

*is satisfied.*

Assumption (3.3) can be satisfied by choosing an appropriate partition of $S_\epsilon$. For the proof of Theorem 3.1 we shall need the following

LEMMA 3.1.

*i) For any bounded function $x$ over $S$, there holds*

$$\|W\chi x - W_N \chi x\| \leq 5C_1\epsilon_p\|x\| + C_1 \max_{i=1,\ldots,N} \sup_{P \in S^i} |x(P) - x(P^i)|. \tag{3.4}$$

*ii) For any $x \in C(S)$,*

[4] A corner is called rectangular if there exist a neighbourhood $U$ of the corner point and an orthogonal coordinate system such that the corner point has the coordinates (0,0,0) and $U \cap S$ is contained in the union of the three coordinate planes. The corner is the graph of a Lipschitz function if there exist a neighbourhood $U$ of the corner point, an orthogonal coordinate system, and a Lipschitz function $\varphi$ defined over the x-y-plane such that $S \cap U$ coincides with $\{(x, y, \varphi(x, y)) : (x, y) \in \mathbf{R}^2\} \cap U$.

$$\max_{i=1,\dots,N} \sup_{P\in S^i} |W_N\chi x(P) - W_N\chi x(P^i)| \le C\sqrt{\epsilon_p}\|x\|. \tag{3.5}$$

*iii) We have* $\|W\chi I\| \le C_1$ *and* $\|W_N\chi I\| \le 5C_1$.

*Proof. ( of Theorem 3.1.)* First we observe that $2[1 - W(\chi)]$ and $2[1 - W_N(\chi)]$ are bounded from below or, equivalently, that the operators of multiplication by the inverse functions are bounded operators. Namely, $A_\epsilon = 2[1 - W(\chi)]I + W\chi I$ is invertible by (3.1) and the operator $W\chi I$ is compact.Thus the multiplication operator $2[1 - W(\chi)]I$ is Fredholm with index zero and hence invertible, i.e., $2[1 - W(\chi)]$ is bounded from below. From Lemma 3.1, i) with $x \equiv 1$, we get

$$\|W(\chi) - W_N(\chi)\| \le 5C_1\epsilon_p. \tag{3.6}$$

Thus, if $\epsilon_p$ is small enough, then

$$\|\frac{1}{2}[1 - W(\chi)]^{-1}2[W(\chi) - W_N(\chi)]\| \le 1/2$$

and $I + \frac{1}{2}[1 - W(\chi)]^{-1}2[W(\chi) - W_N(\chi)]I$ is invertible. Multiplying the last operator by $2[1 - W(\chi)]I$, we get that $2[1 - W_N(\chi)]I$ is invertible, too, and the function $2[1 - W_N(\chi)]$ is bounded from below. From the just mentioned arguments we even conclude

$$\|\frac{1}{2}[1 - W(\chi)]^{-1}\| \le C, \quad \|\frac{1}{2}[1 - W_N(\chi)]^{-1}\| \le C, \tag{3.7}$$

where the constant depends only on $C_{fs}$. Hence, for the stability of $A_N$, it suffices to derive the invertibility of $\frac{1}{2}[1 - W_N(\chi)]^{-1}A_N = I + \frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I$ from that of $\frac{1}{2}[1 - W(\chi)]^{-1}A_\epsilon = I + \frac{1}{2}[1 - W(\chi)]^{-1}2W\chi I$. This, however, follows as in [6]. Namely, let us introduce

$$T := \frac{1}{2}[1 - W(\chi)]^{-1}2W\chi I, \quad T_N := \frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I$$

and consider $\|(T - T_N)T_N\|$. We obtain

$$\begin{aligned}
&\| \left(\frac{1}{2}[1 - W(\chi)]^{-1}2W\chi I - \frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I\right) \frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I\| \\
&\le \|\frac{1}{2}[1 - W_N(\chi)]^{-1}2[W(\chi) - W_N(\chi)]\frac{1}{2}[1 - W(\chi)]^{-1}2W\chi I\frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I\| \\
&\quad + \|\frac{1}{2}[1 - W_N(\chi)]^{-1}2(W\chi I - W_N\chi I)\frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I\|.
\end{aligned} \tag{3.8}$$

The first term on the right-hand side is smaller than $C\epsilon_p$ by (3.7),(3.6), and Lemma 3.1,iii). The second term can be estimated by Lemma 3.1,i) and (3.7). We get

$$\begin{aligned} &\|\frac{1}{2}[1-W_N(\chi)]^{-1}2(W\chi I - W_N\chi I)\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\| \leq \\ &C\cdot 2\cdot\Big\{5C_1\epsilon_p\|\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\| + \\ C_1 \max_{i=1,\dots,N}\sup_{P\in S^i}&\left|\left(\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\right)(P)-\left(\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\right)(P^i)\right|\Big\}, \end{aligned} \tag{3.9}$$

where the term under the supremum can be estimated by

$$\begin{aligned} &\left|\left(\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\right)(P)-\left(\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\right)(P^i)\right| \leq \\ &\left|\left\{\frac{1}{2}[1-W_N(\chi)]^{-1}(P)-\frac{1}{2}[1-W_N(\chi)]^{-1}(P^i)\right\}2(W_N\chi x)(P)\right| + \\ &\left|\frac{1}{2}[1-W_N(\chi)]^{-1}(P^i)2\left\{(W_N\chi x)(P)-(W_N\chi x)(P^i)\right\}\right|. \end{aligned} \tag{3.10}$$

Together with (3.7), (3.6), Lemma 3.1,iii) and ii) we conclude

$$\begin{aligned} &\left|\left(\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\right)(P)-\left(\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\right)(P^i)\right| \\ &\leq C\sqrt{\epsilon_p}\|x\|. \end{aligned} \tag{3.11}$$

Thus (3.9) and (3.11) lead to

$$\begin{aligned} \|\frac{1}{2}[1-W_N(\chi)]^{-1}2(W\chi I - W_N\chi I)\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\| &\leq \\ C\,\|2(W\chi I - W_N\chi I)\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi x\| &\leq C\sqrt{\epsilon_p}\|x\|. \end{aligned} \tag{3.12}$$

Equations (3.8) and (3.12) imply

$$\|(T-T_N)T_N\| \leq C\sqrt{\epsilon_p}. \tag{3.13}$$

Taking into account that $(I+T)=\frac{1}{2}[1-W(\chi)]^{-1}A_\epsilon$ is invertible, we choose $\epsilon_p$ such that $\|(I+T)^{-1}\|\cdot\|(T-T_N)T_N\|\leq 1/2$ and obtain

$$\begin{aligned} \|[I+T+(T-T_N)T_N]^{-1}\| &\leq \|(I+T)^{-1}\|\,\left\|\left[I+(I+T)^{-1}(T-T_N)T_N\right]^{-1}\right\| \\ &\leq \|(I+T)^{-1}\|\cdot 2\leq C. \end{aligned} \tag{3.14}$$

Furthermore, the operator

$$[I+T+(T-T_N)T_N]^{-1}[I+T-T_N] \tag{3.15}$$

is a left inverse of $(I+T_N)$. Since $T_N$ is a finite range operator, $(I+T_N)$ is Fredholm and its index is zero. Thus (3.15) is the inverse of $(I+T_N)$, and we get $\|(I+T_N)^{-1}\| \leq C$. From $\frac{1}{2}[1+W_N(\chi)]^{-1}A_N = I+T_N$ and (3.7) we conclude $\|A_N^{-1}\| \leq C$. □

*Proof.* (*of Lemma 3.1,* i) ) For $Q \in S$, we get

$$W(\chi x)(Q) - W_N(\chi x)(Q) \tag{3.16}$$

$$\begin{aligned}
&= \frac{1}{4\pi}\int_{S_\epsilon} \frac{n_P\cdot(P-Q)}{|P-Q|^3}x(P)d_PS - \frac{1}{4\pi}\sum_{i=1}^{N}\frac{n_{P^i}\cdot(P^i-Q)}{|P^i-Q|^3}x(P^i)\mu(S^i)\\
&= \frac{1}{4\pi}\sum_{i=1}^{N}\int_{S^i}\left\{\frac{n_P\cdot(P-Q)}{|P-Q|^3}x(P) - \frac{n_{P^i}\cdot(P^i-Q)}{|P^i-Q|^3}x(P^i)\right\}d_PS\\
&= \frac{1}{4\pi}\sum_{i=1}^{N}\int_{S^i}\frac{n_P\cdot(P-Q)}{|P-Q|^3}\{x(P)-x(P^i)\}d_PS +\\
&\quad \frac{1}{4\pi}\sum_{i=1}^{N}\int_{S^i}\left\{\frac{n_P\cdot(P-Q)}{|P-Q|^3} - \frac{n_{P^i}\cdot(P^i-Q)}{|P^i-Q|^3}\right\}x(P^i)d_PS.
\end{aligned}$$

For $P, P^i \in S^i$, we conclude $n_P = n_{P^i}$ and $n_P\cdot(P-Q) = n_{P^i}\cdot(P^i-Q)$. Using this, we arrive at

$$\begin{aligned}
|W(\chi x)(Q) - W_N(\chi x)(Q)| &\leq C_1 \max_{i=1,\dots,N}\sup_{P\in S^i}|x(P)-x(P^i)| +\\
&\quad \sum_{i=1}^{N}\|x\|\frac{1}{4\pi}\int_{S^i}\frac{|n_P\cdot(P-Q)|}{|P-Q|^3}\left|1-\frac{|P-Q|^3}{|P^i-Q|^3}\right|d_PS\\
&\leq C_1 \max_{i=1,\dots,N}\sup_{P\in S^i}|x(P)-x(P^i)| +\\
&\quad C_1\|x\|\max\sup_{P\in S^i}\left|1-\frac{|P-Q|^3}{|P^i-Q|^3}\right|, \qquad (3.17)
\end{aligned}$$

where the last maximum is taken over all $i=1,\dots,N$ such that $S^i$ is not on the face of $S$ which contains $Q$. Furthermore,

$$\left|1-\frac{|P-Q|^3}{|P^i-Q|^3}\right| \leq \left\{1+\frac{|P-Q|}{|P^i-Q|}+\left(\frac{|P-Q|}{|P^i-Q|}\right)^2\right\}\left|1-\frac{|P-Q|}{|P^i-Q|}\right|$$

$$\leq \left\{1+\left[|1-\frac{|P-Q|}{|P^i-Q|}|+1\right]+\left[|1-\frac{|P-Q|}{|P^i-Q|}|+1\right]^2\right\}\left|1-\frac{|P-Q|}{|P^i-Q|}\right|, \tag{3.18}$$

$$\left|1-\frac{|P-Q|}{|P^i-Q|}\right| \leq \frac{|P^i-P|}{|P^i-Q|} \leq \frac{rad\, S^i}{dist(S^i,E^i)} \leq \epsilon_p < 1/2. \tag{3.19}$$

From (3.17)-(3.19) we conclude the validity of i). □

*Proof.* (*of Lemma 3.1,*ii) ) Let $\delta > 0$ and $S^{i,\delta} := \{P \in S_\epsilon \cap E^i : \textit{for all } Q \in S^i \textit{ there holds } |n_P \cdot (Q-P)|/|Q-P| \leq \delta\}$. Furthermore, let $Pl$ be an arbitrary plane not containing $P^i$ and consider $Pl^{i,\delta} := \{P \in Pl : |n_P \cdot (P^i-P)|/|P^i-P| \leq \delta\}$. Then the solid angle under which $Pl^{i,\delta}$ is seen from $P^i$ is just $2\pi\delta$. Hence, (cf.[15] or [22], Sect.4.2.2)

$$\frac{1}{4\pi}\int_{Pl^{i,\delta}} \frac{|n_P \cdot (P-P^i)|}{|P-P^i|^3} d_P S = \frac{1}{2}\delta. \tag{3.20}$$

Since $S^{i,\delta}$ is contained in less than $M$ planes ($M$ is the number of faces on $S$), we conclude

$$\frac{1}{4\pi}\int_{S^{i,\delta}} \frac{|n_P \cdot (P-Q)|}{|P-Q|^3} d_P S \leq \frac{M}{2}\delta \tag{3.21}$$

for $Q \in S^i$. Now, for $P \in S^i$, we get

$$\begin{aligned} W_N\chi x(P) - W_N\chi x(P^i) &= \frac{1}{4\pi}\sum_{k=1}^{N}\left\{\frac{n_{P^k}\cdot(P^k-P)}{|P^k-P|^3} - \frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P^i|^3}\right\} x(P^k)\mu(S^k) \\ &= I_1 + I_2, \end{aligned} \tag{3.22}$$

where $I_1$ is just the sum over the $k = 1,\ldots,N$ with $S^k \subset S_\epsilon \setminus S^{i,\delta}$. The second term $I_2$ is the sum over $k = 1,\ldots,N$ with $S^k \cap S^{i,\delta} \neq \emptyset$. If $R = R^k \in S^k \cap S^{i,\delta}$ and $Q \in S^k$, then

$$\begin{aligned} \frac{|n_Q\cdot(P^i-Q)|}{|P^i-Q|} &= \frac{|n_R\cdot(P^i-R)|}{|P^i-R|}\cdot\frac{|P^i-R|}{|P^i-Q|} \leq \delta\left(1+\frac{|R-Q|}{|P^i-Q|}\right) \\ &\leq \delta\left(1+\frac{rad\, S^k}{dist(S^k,E^k)}\right) \leq \delta(1+\epsilon_p) \leq 2\delta. \end{aligned} \tag{3.23}$$

Consequently, $S^k \subseteq S^{i,2\delta}$ and we get

$$|I_2| \leq \tag{3.24}$$
$$\|x\|\left\{\frac{1}{4\pi}\sum_{k:S^k\subseteq S^{i,2\delta}}\frac{|n_{P^k}\cdot(P^k-P)|}{|P^k-P|^3}\mu(S^k) + \frac{1}{4\pi}\sum_{k:S^k\subseteq S^{i,2\delta}}\frac{|n_{P^k}\cdot(P^k-P^i)|}{|P^k-P^i|^3}\mu(S^k)\right\}.$$

Analogously to the proof of i) we conclude (cf. also (3.21))

$$\frac{1}{4\pi}\sum_{k:S^k\subseteq S^{i,2\delta}}\frac{|n_{P^k}\cdot(P^k-P)|}{|P^k-P|^3}\mu(S^k)=\frac{1}{4\pi}\int_{\{\cup_{S^k\subseteq S^{i,2\delta}}S^k\}}\frac{|n_R\cdot(R-P)|}{|R-P|^3}d_RS+$$
$$\frac{1}{4\pi}\sum_{k:S^k\subseteq S^{i,2\delta}}\int_{S^k}\left\{\frac{|n_{P^k}\cdot(P^k-P)|}{|P^k-P|^3}-\frac{|n_R\cdot(R-P)|}{|R-P|^3}\right\}d_RS$$
$$\leq\frac{1}{4\pi}\int_{S^{i,2\delta}}\frac{|n_R\cdot(R-P)|}{|R-P|^3}d_RS+C_15\epsilon_p\leq\frac{M}{2}2\delta+C\epsilon_p. \tag{3.25}$$

Hence,

$$|I_2|\leq 2\,\|x\|\{M\delta+C\epsilon_p\}. \tag{3.26}$$

On the other hand, we get

$$|I_1|\leq\|x\|\frac{1}{4\pi}\sum_{k:S^k\subseteq S_\epsilon\setminus S^{i,\delta}}\frac{|n_{P^k}\cdot(P^k-P^i)|}{|P^k-P^i|^3}\left|\left\{\frac{\frac{n_{P^k}\cdot(P^k-P)}{|P^k-P|^3}}{\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P^i|^3}}-1\right\}\right|\mu(S^k), \tag{3.27}$$

$$\left\{\frac{\frac{n_{P^k}\cdot(P^k-P)}{|P^k-P|^3}}{\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P^i|^3}}-1\right\}=\frac{\frac{n_{P^k}\cdot(P^k-P)}{|P^k-P|^3}-\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P^i|^3}}{\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P^i|^3}}$$
$$=\frac{\frac{n_{P^k}\cdot(P^k-P)}{|P^k-P|^3}-\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P|^3}}{\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P^i|^3}}+\frac{\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P|^3}-\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P^i|^3}}{\frac{n_{P^k}\cdot(P^k-P^i)}{|P^k-P^i|^3}}$$
$$=\frac{n_{P^k}\cdot(P^i-P)}{n_{P^k}(P^k-P^i)}\cdot\frac{|P^k-P^i|^3}{|P^k-P|^3}+\left\{\frac{|P^k-P^i|^3}{|P^k-P|^3}-1\right\}. \tag{3.28}$$

If $\epsilon_p<\delta/2$ and $S^k, S^i$ lie on different faces of $S$, then $S^k\subseteq S_\epsilon\setminus S^{i,\delta}$ implies that there exists a point $Q\in S^i$ such that

$$|n_{P^k}\cdot(Q-P^k)|\ \geq\ \delta|Q-P^k|,$$

$$|n_{P^k}\cdot(P^i-P^k)| \geq |n_{P^k}\cdot(Q-P^k)|-|P^i-Q| \geq \left\{\delta-\frac{|P^i-Q|}{|Q-P^k|}\right\}|Q-P^k|$$
$$\geq \left\{\delta-\frac{rad\,S^i}{dist(S^i,E^i)}\right\}|Q-P^k| \geq \{\delta-\epsilon_p\}|Q-P^k| \geq \frac{1}{2}\delta|Q-P^k|.$$

This and (3.18),(3.19) yield

$$\left|\left\{\frac{\frac{n_{P^k}(P^k-P)}{|P^k-P|^3}}{\frac{n_{P^k}(P^k-P^i)}{|P^k-P^i|^3}}-1\right\}\right| \leq C\frac{|P^i-P|}{\delta|Q-P^k|}+C\epsilon_p$$

$$\leq C\frac{rad\,S^i}{\delta\,dist(S^i,E^i)}+C\epsilon_p \leq C\epsilon_p\left(\frac{1}{\delta}+1\right). \tag{3.29}$$

Now, analogously to (3.25), we conclude from (3.27),(3.29)

$$|I_1| \leq \|x\|C\left(\frac{1}{\delta}+1\right)\epsilon_p \leq C\{\frac{1}{\delta}+1\}\epsilon_p\|x\|. \tag{3.30}$$

Together with (3.22) and (3.26) we get

$$|W_N\chi x(P)-W_N\chi x(P^i)| \leq C\left\{\delta+\epsilon_p+\epsilon_p\left(\frac{1}{\delta}+1\right)\right\}\|x\|. \tag{3.31}$$

Let $\delta=\sqrt{\epsilon_p}$. Then we obtain

$$|W_N\chi x(P)-W_N\chi x(P^i)| \leq C\sqrt{\epsilon_p}\|x\|. \tag{3.32}$$

□

*Proof.* (*of Lemma 3.1,* iii) ). The estimate $\|W\chi\| \leq C_1$ follows from the definition of $C_1$. The second estimate is a consequence of the first and of i). □

## 4. The convergence of the iteration process .

**4. 1. The two-grid iteration.** Writing the steps (2.8)-(2.11) in one equation we arrive at

$$x^i = \left\{I-A_{N_c}^{-1}2W_N\chi I\right\}\frac{1}{2}[1-W_N(\chi)]^{-1}y+Op\ x^{i-1},$$
$$Op := A_{N_c}^{-1}2(W_N\chi I-W_{N_c}\chi I)\frac{1}{2}[1-W_N(\chi)]^{-1}2W_N\chi I$$

Obviously, $x_N$ is a fixed point of the iteration. Hence, we get from the last equation

$$(x^i - x_N) = Op\,(x^{i-1} - x_N). \tag{4.1}$$

However, the proof of Theorem 3.1 yields $\|A_{N_c}^{-1}\| \le C$ if

$$\frac{rad\, S_c^i}{dist(S_c^i, E_c^i)} \le \epsilon_p, \quad i = 1, \dots, N_c. \tag{4.2}$$

Here $E_c^i$ denotes the union of all faces of $S$ not containing $S_c^i$. We get

$$\begin{aligned}\|2(W_{N_c}\chi I - W_N\chi I)\frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I\| \le \\ \|2(W\chi I - W_N\chi I)\frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I\| + \\ \|2(W_{N_c}\chi I - W\chi I)\frac{1}{2}[1 - W_N(\chi)]^{-1}2W_N\chi I\|.\end{aligned}$$

Similarly to (3.12) we conclude that both terms on the right-hand side of the last inequality are less than $C\sqrt{\epsilon_p}$ if (3.3) and (4.2) hold. Thus $\|Op\| \le C\sqrt{\epsilon_p}$. If an arbitrary $q$ with $0 < q < 1$ is given and $\epsilon_p$ is small enough, then $C\sqrt{\epsilon_p} \le q < 1$, and from (4.1) we conclude

$$\|x^i - x_N\| \le q\|x^{i-1} - x_N\|.$$

THEOREM 4.1. *Suppose (3.1) holds and choose an arbitrary $q$ such that $0 < q < 1$. Then there is a small positive $\epsilon_p$ depending on $q$ such that the two-grid iteration (2.8)-(2.11) converges for any triangulations $S_\epsilon = \cup S^i, S_\epsilon = \cup S_c^i$ satisfying (3.3) and (4.2), respectively. Moreover, there holds*

$$\|x^i - x_N\| \le q^i\|x^0 - x_N\|. \tag{4.3}$$

REMARK . *Suppose no finite section step is performed and, nevertheless, the operators $A_N$ and $A_{N_c}$ are stable. Instead of the truncation, assume that the partitions $S = \cup_{i=1}^N S^i$ and $S = \cup_{i=1}^{N_c} S_c^i$ coincide over the strip $Str$ of width $\epsilon$. Define $E^i$ for $S^i \cap Str = \emptyset$ as in the beginning of Sect.3. For $S^i \cap Str \ne \emptyset$, let $E_0^i$ stand for the union of all faces on $S$ which do not contain $S^i$ and set $E^i := E_0^i \setminus Str$. Similarly, let us define $E_c^i$ and consider (3.3) and (4.2) with this new definition of $E^i$ and $E_c^i$. Then Theorem 4.1 remains true without the assumption (3.1). The proof is just the same since the restrictions of $W_N$ and $W_{N_c}$ to $Str$ coincide and thus these restrictions vanish in the difference $W_N - W_{N_c}$ and in the corresponding operator $Op$.*

REMARK . *Obviously, Theorem 4.1 remains valid if we assume the stability of the coarse grid operator $A_{N_c}$ instead of the assumption (3.1).*

**4. 2. The Neumann iteration.** For the iteration (2.12) we prove

THEOREM 4.2. *Suppose (3.2) holds. Then there is a* $q$, $0 < q < 1$ *and an* $\epsilon_p$, $0 < \epsilon_p < 1/2$ *such that, for any triangulation satisfying (3.3), the iteration (2.12) converges and there holds*

$$\|x^i - x_N\| \leq Cq^i \|x^0 - x_N\|. \tag{4.4}$$

*Proof.* Let us set $T_N := 2[\frac{1}{2} - W_N(\chi)]I + 2W_N\chi I$, $T := 2[\frac{1}{2} - W(\chi)]I + 2W\chi I$. Furthermore, let $\rho_1$ stand for the essential norm of the operator $2W$. It is well known that (3.2) implies $\rho_1 < 1$ (cf.[15, 33] or [22], Sect.4.2.3). Moreover, let $\rho_2$ denote the maximum of the absolute values of the eigenvalues of $2W$ which are different from 1. Then the well-known estimate $\rho_2 < 1$ follows e.g. from the proof of Theorem 12 in Sect.1.3 of [22]. Hence, $\rho := \max\{\rho_1, \rho_2\} < 1$, and we can choose $q$ with $\max\{\frac{2}{\kappa} - 1\ ,\ 1 + (\rho - 1)\frac{1}{\kappa}\} < q < 1$. Now it is not hard to verify that the iterative solutions of the Neumann iteration satisfy (4.1) with $Op := -\frac{1}{\kappa}\{T_N + (1 - \kappa)I\}$. Thus all what we need to show is that the spectral radius of the operator $Op$ is less than $q$. This, however, follows if we prove that the spectrum of $T_N$ is contained in the set $Z := \{z \in C : |z| < \rho + \delta\} \cup \{z \in C : |z - 1| < \delta\}$ for any sufficiently small $\delta > 0$. Thus we have to show the invertibility of $A_{\lambda,N} := \{I - \lambda T_N\}$ for $\lambda^{-1} \in C \setminus Z$. To get this we observe that $A_\lambda := \{I - \lambda T\}$ is invertible for $\lambda^{-1} \in C \setminus Z$. Namely, the spectrum of $2W$ is contained in $Z$ and the proofs of Theorems 2.1 and 3.2 of [25] yield the invertibility of the finite section operator $A_{\lambda,\epsilon}$. Now consider the Nyström approximate operator $A_{\lambda,N}$ for $A_\lambda$ and suppose (3.3) holds. Repeating the proof of Theorem 3.1, we conclude that, for sufficiently small $\epsilon_p$, the operator $A_{\lambda,N}$ is invertible for any $\lambda$ with $\lambda^{-1} \in C \setminus Z$. □

REMARK . *If the number* $\rho$ *from the last proof is known explicitly, then the optimal choice for* $\kappa$ *is* $\kappa = (3 - \rho)/2$. *Namely, in this case the lower bound* $\max\{\frac{2}{\kappa} - 1\ ,\ 1 + (\rho - 1)\frac{1}{\kappa}\}$ *of* $q$ *is minimal. If* $\rho$ *is unknown, then one should choose at least* $1 < \kappa \leq 3/2$.

REMARK . *As it has been mentioned in the last proof,* $2W$ *has a single eigenvalue* $\lambda = 1$ *and the rest of the spectrum is contained in* $\{z \in \mathbf{C} : |z| \leq \rho\}$. *Due to the singularity subtraction* 1 *is an eigenvalue of* $T_N$, *too. Moreover, let us choose any* $q$ *with* $\rho < q < 1$. *Analogously to [27], Sect.5 one can prove that the corresponding eigenspace is of dimension one and the absolute values of the other eigenvalues are less than* $q$ *provided the number* $\epsilon_p$ *in (3.3) is sufficiently small. We recommend the following iteration procedure: Start with* $x^1 = y/2$ *and set* $x^i = y - (A_N - I)x^{i-1}$, $i = 2, 3, \ldots$. *This iteration corresponds to the procedure (2.12), where* $x^0 = 0$, $\kappa = 2$ *for* $i = 1$, *and* $\kappa = 1$ *for* $i = 2, \ldots$. *The operator Op from (4.1) is equal to* $\frac{1}{2}(I - T_N)$ *for* $i = 1$ *and to* $-T_N$ *for* $i = 2, \ldots$. *Hence, the component of the error taken in the direction of the eigenfunction corresponding to eigenvalue* 1 *vanishes after the first step. Since this error component is zero and the other eigenvalues of* $T_N$ *have absolute values less than* $q$, *the iteration converges and* $\|x^i - x_N\| \leq Cq^i\|x^0 - x_N\|$. *In the numerical example of Sect.6.3 we have implemented this iteration.*

REMARK . *Let us suppose again that* $\epsilon_p$ *is sufficiently small and that all the eigenvalues of* $T_N$ *which are different from 1 are contained in* $\{z \in \mathbf{C} : |z| < q\}$. *In addition we suppose that* $T_N$ *is diagonalizable. Then (cf.[28]) the GMRES is convergent and* $\|x^i - x_N\| \leq Cq^i\|x^0 - x_N\|$.

## 5. Rates of convergence and complexity .

**5. 1. The error estimate and the complexity for Gaussian elimination.** In order to get rates of convergence we need more information about the triangulation. Following [24], we could introduce special triangulations for an arbitrary polyhedron $\Omega$. For the sake of simplicity, however, let us define the partitions only for the case of the cube $\Omega = (0,1) \times (0,1) \times (0,1)$.[5] Since the faces of the cube are squares, we can use rectangles in the partition of $S_\epsilon$. Moreover, by symmetry it is enough to define the partition over one face, e. g. over $F := [0,1] \times [0,1] \simeq [0,1] \times [0,1] \times \{0\}$. We choose the fixed grid parameters $\alpha \geq 1, i_0 \in \mathbf{Z}_+$ and define, for any integer $n > i_0$, the set $S_\epsilon = S_n$ and the partition $S_n = \cup_{i=1}^{N_n} S^i$ depending on $n$. To get these partitions over $F$, we first divide $F$ into $n$ "strips" $F_i, i = 0, \ldots, n-1$ parallel to the sides of $F$ with distances $\frac{1}{2}(\frac{i}{n})^\alpha$ to the boundary (cf. Fig.1):

$$(5.1) \quad F_i := \left\{(x,y) \in F : \frac{1}{2}(1 - (\frac{i+1}{n})^\alpha) < \max\{|\frac{1}{2} - x|, |\frac{1}{2} - y|\} < \frac{1}{2}(1 - (\frac{i}{n})^\alpha)\right\}.$$

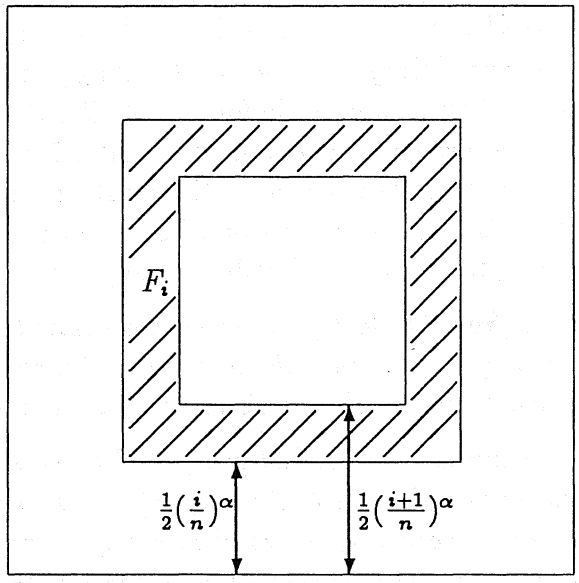


FIG. 1. *The strip $F_i$.*

The number $\alpha$ is the grading parameter. Choosing a high $\alpha$ results in very small strips near the sides of $F$, i.e., near the set of edge points. We introduce $\epsilon := (i_0/n)^\alpha$ and obtain $F \cap Str = \cup_{i=0}^{i_0-1} F_i$ and $S_n = S \setminus Str$. Thus the parameter $i_0$ defines the size of the strip which we cut off in the finite section step. Now we only have to divide the rest of the strips $F_i, i = i_0, \ldots, n-1$ into rectangles. In order to satisfy (3.3) for a small $\epsilon_p$ and to obtain a small number of subdivision domains, we divide $F_i$ into rectangles, where the lengths of the sides are nearly equal to the width of the strip. For symmetry reasons, it is enough to give this partition of $F_i$ over the part

[5] All the results of this section remain true if the boundary $S$ is locally the graph of a Lipschitz function, if the Nyström method is stable, and if the special grids introduced in Sect.4 of [24] are used.

$$F_i' := \left\{(x,y) \in \mathbf{R}^2 : \frac{1}{2}(\frac{i}{n})^\alpha < x < \frac{1}{2}(\frac{i+1}{n})^\alpha, \frac{1}{2}(\frac{i}{n})^\alpha < y < 1 - \frac{1}{2}(\frac{i}{n})^\alpha\right\}.$$

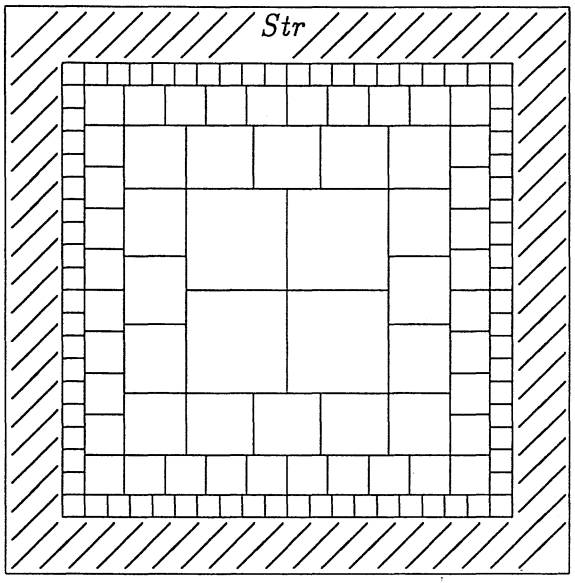


FIG. 2. *Partition of F*

We set $F_i' = \cup_{j=1}^k F_{i,j}$ with

$$F_{i,j} := \left\{(x,y) \in \mathbf{R}^2 : \frac{1}{2}(\frac{i}{n})^\alpha < x < \frac{1}{2}(\frac{i+1}{n})^\alpha, t_j < y < t_{j+1}\right\}, \tag{5.2}$$

$$t_1 = \frac{1}{2}(\frac{i}{n})^\alpha, t_2 = \frac{1}{2}(\frac{i+1}{n})^\alpha, t_k = 1 - \frac{1}{2}(\frac{i+1}{n})^\alpha, t_{k+1} = 1 - \frac{1}{2}(\frac{i}{n})^\alpha, \tag{5.3}$$

$$t_j := t_2 + (j-2)\frac{t_k - t_2}{k-2},\ j = 3,4,\ldots,k-1.$$

In order to get that the $F_{i,j}$ are almost squares, we choose $k$ such that

$$t_j - t_{j-1} = \frac{t_k - t_2}{k-2} = \frac{1 - (\frac{i+1}{n})^\alpha}{k-2} \approx \frac{1}{2}\left((\frac{i+1}{n})^\alpha - (\frac{i}{n})^\alpha\right), \tag{5.4}$$

$$k := 2 + \left[\frac{1 - (\frac{i+1}{n})^\alpha}{\frac{1}{2}((\frac{i+1}{n})^\alpha - (\frac{i}{n})^\alpha)}\right].$$

Now the partition $S_n = \cup_{i=1}^{N_n} S^i$ is the union over all rectangles $F_{i,j}$ for all strips $F_i$ and all faces of $S$ (cf. Fig.2).

Summing up the numbers $k$ from (5.4) for $i = i_0, \ldots, n-1$, we obtain that the number $N_n$ is of order

$$N_n \sim \begin{cases} n^\alpha & \text{if} \quad \alpha > 2 \\ n^2 \log n & \text{if} \quad \alpha = 2 \\ n^2 & \text{if} \quad 2 > \alpha. \end{cases} \tag{5.5}$$

Condition (3.3) is fulfilled for sufficiently large $i_0$ since

$$\frac{rad\, S^i}{dist(S^i, E^i)} \leq C \frac{\frac{1}{2}((\frac{i_0+1}{n})^\alpha - (\frac{i_0}{n})^\alpha)}{\frac{1}{2}(\frac{i_0}{n})^\alpha} \leq C \frac{\alpha(i_0+1)^{\alpha-1}}{i_0^\alpha} \leq C \frac{1}{i_0}. \tag{5.6}$$

In other words, if $i_0$ is large enough, then Theorem 3.1 implies the stability of the Nyström method (2.3) over the partition $S_n = \cup_{i=1}^{N_n} S^i$. On the other hand, a larger $i_0$ will lead to a larger error in the quadrature. Thus we choose $i_0$ to be the smallest number such that (2.3) is stable. Now suppose the right-hand side $y$ of our integral equation is continuous on $S$ and $C^\infty$ on each face of $S$. Then (cf.[22], Sect.5.1.4) the solution $x$ fulfills

$$\sup_{P \in S} |x(P)| \leq C, \quad |\nabla^l x(P)| \leq C\, dist(P, e)^{\delta - l}, \quad (l = 1, \ldots), \tag{5.7}$$

where $e$ is the edge nearest to $P$ and $0 < \delta < 1$ is a certain number. This number $\delta$ depends on the geometry of $S$. Its determination for arbitrary $S$ is a hard problem since it requires the computation of eigenvalues for certain boundary value problems over spherical domains.

THEOREM 5.1. *Let $\alpha \geq 1$. Suppose (3.1) holds and choose $i_0$ such that (3.3) is satisfied with $0 < \epsilon_p$, where $\epsilon_p$ is taken from Theorem 3.1. If the function $y$ is continuous on $S$ and $C^\infty$ on each face of $S$ and if $x_N$ is the solution of (2.3), then*

$$\sup_{i=1,\ldots,N_n} |x_N(P^i) - x(P^i)| \leq C \begin{cases} n^{-2} & \text{if} \quad \alpha > 2/\delta \\ n^{-2} \log n & \text{if} \quad \alpha = 2/\delta \\ n^{-\alpha\delta} & \text{if} \quad \alpha < 2/\delta. \end{cases} \tag{5.8}$$

*Here $\delta$ is the exponent appearing in (5.7).*

*Proof.* Let us define the piecewise interpolation projection $P_N$ by

$$P_N f(P) := \begin{cases} f(P^i) & \text{if } P \in S^i \text{ for an } i = 1, \ldots, N_n \\ 0 & \text{if } P \in Str \end{cases}.$$

Since the $W_N \chi f$ depends on $\{f(P^i)\}$ only, we get $W_N \chi I = W_N \chi P_N$ and, for the multiplication operator $2[1 - W_N(\chi)]I$, we arrive at $P_N 2[1 - W_N(\chi)]P_N = P_N 2[1 - W_N(\chi)]I$. Thus

$$A_N = 2[1 - W_N(\chi)]I + 2W_N \chi I =$$
$$P_N \{A_N|_{im\, P_N}\} P_N + (I - P_N)\{2[1 - W_N(\chi)]I\}(I - P_N) + (I - P_N)\{A_N|_{im\, P_N}\} P_N.$$

From this equation we conclude that $A_N$ is invertible if and only if $\{P_N A_N|_{im\,P_N}\}$ is invertible, and

$$\| \{P_N A_N|_{im\,P_N}\}^{-1} \| \leq C \|A_N^{-1}\|.$$

Using the boundedness of $\| \{P_N A_N|_{im\,P_N}\}^{-1} \|$, the proof of Theorem 5.1 follows analogously to that of Theorem 4.1 in [24]. The only difference is that instead of (4.5) in [24] one has to use

$$|x(P) - x(P^j)| \leq |P - P^j|^\delta$$

which follows from the formulae (5.7). Note that only the last inequality is used in the considerations following (4.5) of [24]. □

Now let us suppose we solve (2.4) using Gaussian elimination. Then the number of operations is of order $N_n^3$. In order to get an error smaller than a prescribed positive $\epsilon_C$, we have to choose (cf.(5.8)) $n \sim \epsilon_C^{-1/2}$ if $\alpha > 2/\delta$, $n \sim \epsilon_C^{-1/2} \log^{1/2}(\epsilon_C^{-1})$ if $\alpha = 2/\delta$ and $n \sim \epsilon_C^{-1/(\alpha\delta)}$ if $\alpha < 2/\delta$. Consequently, the number of complexity *comp*, i. e., the number of operations to achieve an error less than $\epsilon_C$ is given by (cf.(5.5))

$$comp \sim \begin{cases} \epsilon_C^{-6/(\alpha\delta)} & \text{if} \quad 1 \leq \alpha < 2 \\ \epsilon_C^{-3/\delta} \log^3(\epsilon_C^{-1}) & \text{if} \quad \alpha = 2 \\ \epsilon_C^{-3/\delta} & \text{if} \quad 2 < \alpha < 2/\delta \\ \epsilon_C^{-3/\delta} \log^{3/\delta}(\epsilon_C^{-1}) & \text{if} \quad \alpha = 2/\delta \\ \epsilon_C^{-3\alpha/2} & \text{if} \quad 2/\delta < \alpha. \end{cases} \tag{5.9}$$

**5. 2. The complexity of the two-grid method.** Now let us consider the two-grid iteration (2.8)-(2.11). We choose the coarse grid as follows: First we choose a fixed parameter $\eta > 1$. As above we give the grid only over one face of $S$, i.e., on the square $F$. We introduce the strips $F_{c,i}$ with distance $\eta_i/2$ to the boundary, where $\eta_i := (i_0/n)^\alpha \eta^i$, $i = 0, \ldots, i_1$. In other words we set

$$\begin{aligned} F_{c,i} &:= \left\{(x,y) \in F : \frac{1}{2}(1-\eta_{i+1}) < \max\{|\frac{1}{2} - x|, |\frac{1}{2} - y|\} < \frac{1}{2}(1-\eta_i)\right\}, \\ & \qquad i = 0, 1, \ldots, i_1, \; i_1 := \left[\alpha \frac{\log \frac{n}{i_0}}{\log \eta}\right], \; \eta_{i_1+1} := 1 \end{aligned} \tag{5.10}$$

and get a similar partition into strips as for the fine grid, only with a geometric grading. Since $\eta$ is a fixed number, the maximal width of the strips does not tend to zero if $n$ tends to infinity. Analogously to the fine grid, we define the further partition only over

$$F'_{c,i} := \left\{(x,y) \in \mathbf{R}^2 : \frac{1}{2}\eta_i < x < \frac{1}{2}\eta_{i+1}, \; \frac{1}{2}\eta_i < y < 1 - \frac{1}{2}\eta_i\right\}.$$

We set

$$F_{c,i,j} := \left\{(x,y) \in \mathbf{R}^2 : \frac{1}{2}\eta_i < x < \frac{1}{2}\eta_{i+1},\ t_{c,j} < y < t_{c,j+1}\right\},$$

$$t_{c,1} := \frac{1}{2}\eta_i,\ t_{c,2} := \frac{1}{2}\eta_{i+1},\ t_{c,k} := 1 - \frac{1}{2}\eta_{i+1},\ t_{c,k+1} := 1 - \frac{1}{2}\eta_i, \tag{5.11}$$

$$t_{c,j} := t_{c,2} + (j-2)\frac{t_{c,k} - t_{c,2}}{k-2},\ j = 3, \ldots, k-1, \tag{5.12}$$

where $k$ is chosen such that $F_{c,i,j}$ is nearly a square, i. e.

$$t_{c,j} - t_{c,j-1} = \frac{t_{c,k} - t_{c,2}}{k-2} = \frac{1-\eta_{i+1}}{k-2} \approx \left[\frac{1}{2}\eta_{i+1} - \frac{1}{2}\eta_i\right], \tag{5.13}$$

$$k := 2 + \left[\frac{1-\eta_{i+1}}{\frac{1}{2}\eta_{i+1} - \frac{1}{2}\eta_i}\right]. \tag{5.14}$$

Now the partition $S_n = \cup_{i=1}^{N_{c,n}} S_c^i$ is the union over all rectangles $F_{c,i,j}$ for all strips $F_{c,i}$ and all faces of $S$.

In every strip $F_{c,i}$ we have about $O(\{\frac{1}{2}(i_0/n)^\alpha \eta^{i+1} - \frac{1}{2}(i_0/n)^\alpha \eta^i\}^{-1})$ subdomains. Thus the number $N_{c,n}$ of all subdivision domains is of order $n^\alpha$. For $\eta$ sufficiently close to one, condition (4.2) is fulfilled since

$$\frac{rad\, S_c^i}{dist(S_c^i, E_c^i)} \le C \frac{\frac{1}{2}(\frac{i_0}{n})^\alpha \eta^{i+1} - \frac{1}{2}(\frac{i_0}{n})^\alpha \eta^i}{\frac{1}{2}(\frac{i_0}{n})^\alpha \eta^i} \le C\,(\eta - 1). \tag{5.15}$$

The LU-factorization of the coarse grid matrix corresponding to $A_{N_c}$ requires $O(N_{c,n}^3)$ operations and each iteration step of the two-grid algorithm requires $O(N_n^2)$. To obtain an error less than $\epsilon_C$ we need about $\log \epsilon_C^{-1}$ iterations. For the complexity of this iteration, we conclude from (5.5), (5.8), and $N_{c,n} \sim n^\alpha$ that

$$comp \sim \begin{cases} \epsilon_C^{-4/(\alpha\delta)} \log(\epsilon_C^{-1}) & \text{if} \quad 1 \le \alpha \le 4/3 \\ \epsilon_C^{-3/\delta} & \text{if} \quad 4/3 < \alpha < 2/\delta \\ \epsilon_C^{-3/\delta} \log^{3/\delta}(\epsilon_C^{-1}) & \text{if} \quad \alpha = 2/\delta \\ \epsilon_C^{-3\alpha/2} & \text{if} \quad 2/\delta < \alpha. \end{cases} \tag{5.16}$$

Note that the factor $\log(\epsilon_C^{-1})$ in the case $1 \le \alpha \le 4/3$ can be removed if we solve the Nyström equation over a sequence of meshes and start the iteration with the solution from a coarser grid as initial function. In the case of a one-dimensional boundary this variant is analysed by Atkinson and Graham [5]. We also note that in practical computations the coarse grid equation can be solved by another iteration method. This would lead to better orders of complexity.

**5. 3. The complexity of the Neumann iteration.** If we use the procedure proposed by Wendland, then (cf. Theorem 4.2) we need about $C\ \log(\epsilon_C^{-1})$ iterations in order to get an error less than $\epsilon_C$, where $C$ depends on the geometry of $S$, only. Hence, we need $C\ N_n^2 \log(\epsilon_C^{-1})$ operations and the complexity is given by (cf.(5.8),(5.5))

$$
(5.17) \qquad comp \sim \begin{cases} \epsilon_C^{-4/(\alpha\delta)} \log(\epsilon_C^{-1}) & \text{if} \quad 1 \le \alpha < 2 \\ \epsilon_C^{-2/\delta} \log^3(\epsilon_C^{-1}) & \text{if} \quad \alpha = 2 \\ \epsilon_C^{-2/\delta} \log(\epsilon_C^{-1}) & \text{if} \quad 2 < \alpha < 2/\delta \\ \epsilon_C^{-2/\delta} \log^{1+2/\delta}(\epsilon_C^{-1}) & \text{if} \quad \alpha = 2/\delta \\ \epsilon_C^{-\alpha} \log(\epsilon_C^{-1}) & \text{if} \quad 2/\delta < \alpha. \end{cases}
$$

Again, one factor $\log(\epsilon_C^{-1})$ can be dropped, if a sequence of partition is used (cf.the end of Section 5.2).

Summarizing the complexity results, we can say the following: If one uses Gaussian elimination for the solution of the system of linear equations, then the best choice of $\alpha$ is to take $\alpha$ a little bit larger than two or, roughly speaking, to set $\alpha = 2$. With this choice the complexity is reduced from $C\epsilon_C^{-6/\delta}$ for the uniform mesh with $\alpha = 1$ to $C\epsilon_C^{-3/\delta}$ for the graded mesh defined by $\alpha \in (2, \frac{2}{\delta})$. Especially, the optimal choice of $\alpha$ is independent of the parameter $\delta$ appearing in (5.7). The Neumann iteration with the same $\alpha$ results even in a complexity of about $C\epsilon_C^{-2/\delta}$. However, if the geometry of $S$ is complicated, then $C$ is large and the Neumann iteration is slow. In this case we recommend the two-grid iteration, where the convergence speed of the iteration can be improved by using appropriate meshes. This method together with the choice $\alpha = 4/3$ gives the same asymptotic rate as obtained by Nyström's method together with Gaussian elimination and $\alpha = 2$. The constant in the asymptotic estimate for the two-grid algorithm, however, seems to be much smaller. If we compare two-grid iteration and Gaussian elimination for the same $\alpha$ and for large $n$, then the complexity of the two-grid iteration is smaller since the LU-factorization is performed for a smaller system of equations.

## 6. Numerical examples.

**6. 1. The error of the Nyström method.** In the numerical examples of Sect.6 we shall consider the double layer potential equation over the boundary of three different domains $\Omega_i$, $i = 1, 2, 3$. The first one is the convex cube $\Omega_1 := [0,1] \times [0,1] \times [0,1]$. Beside this we consider the L-block $\Omega_2$, i.e., the direct product of the L-shaped domain $([0,1] \times [0,0.5]) \cup ([0.25,1] \times [0.5,1])$ in the x-z-plane multiplied by the interval $[0,1]$ in y-direction. Note that $\Omega_2$ is rectangular and locally the graph of a Lipschitz function. Our third domain $\Omega_3$ is the polyhedron of Fig.3, where we have chosen $P_1 := (0, 0, \frac{1}{\sqrt{2}})$, $P_2 := (0,0,0)$, $P_3 := (-1,-1,-\lambda)$, $P_4 := (1,-1,\lambda)$, $P_5 := (1,1,-\lambda)$, and $P_6 := (-1,1,\lambda)$ with $\lambda := 1$ [6]. This polyhedron $\Omega_3$ does not fulfil any sufficient condition for the stability of our Nyström method.

[6] In order to get a nice and clear picture, we have drawn $\Omega_3$ in Fig.3 with a smaller $\lambda$.

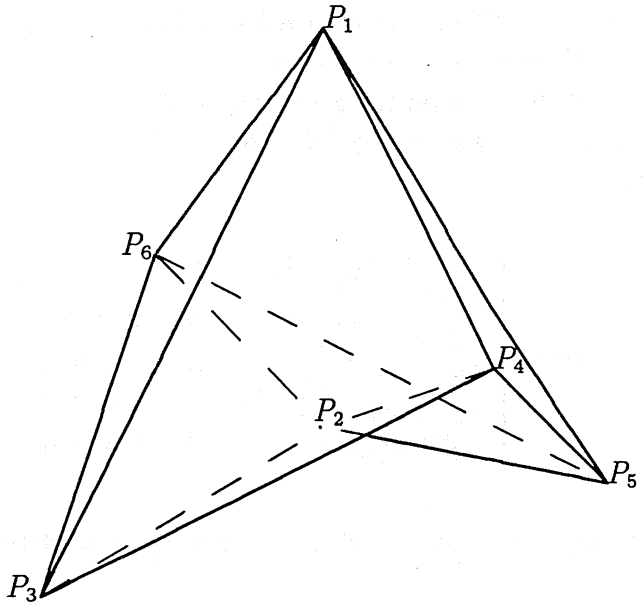


FIG. 3. *Polyhedron* $\Omega_3$

In our first example let us consider method (2.3) over $\Omega_2$ for the continuous but non-harmonic data $y(P) = 6|P - (1,0,0)|^{-2}$. The "supremum norm" error $SER = SER_n$ depending on the values $n, i_0$, and $\alpha$ is given in Tables 1 and 2. For simplicity, the supremum norm is just the maximum of the error taken over the six points (0.25,0,0.5), (0.25,0.5,0.5), (0.625,0,0.25), (0.25,0.1,0.5), (0.325,0,0.45), and (0.025,0,0.25). Moreover, to determine this error we have replaced the true solution by an extrapolation of the approximate values. By $EX = EX_n$ we denote the following estimate for the order of convergence:

$$EX := -\frac{\log SER_n - \log SER_{n-1}}{\log n - \log(n-1)}.$$

Thus $SER_n \sim n^{-EX}$. The mesh for the Nyström method in Tables 1 and 2 is defined analogously to Sect. 5.1. Note that the partition $S_\epsilon = \cup_{i=1}^{N_n} S^i$ of Sect.5.1 is determined by the parameters $\alpha, i_0$ and $n$, where $\alpha$ is the degree of mesh refinement near the edges. The integer $n$ denotes the total number of strips and $i_0$ the number of strips which are neglected in the quadrature rules. High values of $\alpha$ lead to small subdivision domains near the edges and to greater subdivision domains around the mid-points of the faces of $S = \partial\Omega_2$. In order to keep the last subdomains small, we have used an idea of Kress (cf. Equ.(2.2) in [18]) and have replaced[7] the function $\frac{i}{n} \mapsto (\frac{i}{n})^\alpha$ in the definition of the strips $F_i$ by a function $\frac{i}{n} \mapsto \varphi(\frac{i}{n})$, where $\varphi : [0,1] \to [0,1]$ behaves like $t \mapsto t^\alpha$ for $t$ near to 0 and $C_\varphi := \max_{0\le t\le 1} |\varphi'(t)|$ is as small as possible. Note that the maximal side length of the subdomains in the strip $F_i$ is equal to $\frac{1}{2}[\varphi(\frac{i+1}{n}) - \varphi(\frac{i}{n})]$ and can be estimated by $\frac{1}{2n} C_\varphi$. Especially, we have set

$$\varphi(t) := \frac{\{t[1+\lambda_1 - \lambda_1 t]\}^\alpha}{\{t[1+\lambda_1 - \lambda_1 t]\}^\alpha + \{(1-t)[1+\lambda_1 - \lambda_1(1-t)]\}}.$$

[7] If $n$ is very small and the strip *Str* is neglected, then the mesh after the replacement is nearly uniform. Therefore, we have used the original function $\frac{i}{n} \mapsto (\frac{i}{n})^\alpha$ for the computations in the Tables 3 and 8.

The parameter $\lambda_1$ is 0 for $\alpha = 1$, 0.105 for $\alpha = 4/3$, 0.28 for $\alpha = 2$, and 0.48 for $\alpha = 3$. Now, if $U$ is the solution of the Dirichlet problem in $\Omega_2$ with the boundary value $U(P) = 3|P - (1,0,0)|^{-2}$ $(P \in S)$, then $U$ admits the representation

$$U(Q) = \frac{1}{4\pi} \int_S \frac{n_P \cdot (Q - P)}{|Q - P|^3} x(P) d_P S, \tag{6.1}$$

where $x$ is the solution of the double layer potential equation $Ax = y$. Substituting $x_N$ into this representation formula and computing the integral via quadrature rule, we arrive at the formula

$$U_N(Q) := \frac{1}{4\pi} \sum_{i=1}^{N_u} \frac{n_{P_u^i} \cdot (Q - P_u^i)}{|Q - P_u^i|^3} x_N(P_u^i) \mu(S^i). \tag{6.2}$$

Here $S = \cup_{i=1}^{N_u} S_u^i$ denotes the uniform partition of Sect.5.1 defined with $\alpha = 1$ and $i_0 = 0$. The error $ERR := |U(Q) - U_N(Q)|$ for $Q = (0.5, 0.5, 0.5)$ is given in the last columns of Tables 1 and 2. If we compare the supremum norm errors in Tables 1 and 2 for $i_0 = 0$, large n, and equal numbers $N_n$, then we get smaller errors for large orders of mesh grading $\alpha$. The best approximate values for $U(0.5, 0.5, 0.5)$ correspond to the parameters $\alpha = 2$, $i_0 = 1$. Thus graded meshes lead to better approximations. However, since the convergence orders $EX$ oscillate, the effect of mesh grading is not so clear as expected in view of Theorem 5.1. For small $n$, $\alpha = 1$ is still a good choice. Moreover, we have no explanation for the high orders of convergence obtained in the case $i_0 = 0$. The first term in the asymptotics of the solution seems to be due to the edge singularity, i.e., the exponent $\delta$ appearing in (5.7) is equal to $2/3$. Thus $EX$ should tend to $2/3 \cdot \alpha$. The values $EX$, however, are close to $2/3 \cdot \alpha$ only for $i_0 > 0$. The smallest supremum norm errors are obtained in the case $i_0 = 0$ and we recommend to choose $i_0 = 0$ whenever the method (2.3) is stable with this choice. For the pointwise behaviour of the error, we refer to the paper [24].

Next let us consider the cube $\Omega_1$, $S := \partial\Omega_1$ and the harmonic function

$$U(Q) := \frac{1}{2} \left\{ \frac{1}{|Q - (1.5, 0.5, 0.5)|} + \frac{1}{|Q - (-0.5, 0.5, 0.5)|} \right\}, \quad Q \in \Omega_1. \tag{6.3}$$

Then $U(0.5, 0.5, 0.5) = 1$ and $U$ is given by (6.1), where $x$ is the solution of $Ax = y := 2U|_S$. Table 3 contains some results for the error $ERR := |U_N(0.5, 0.5, 0.5) - U(0.5, 0.5, 0.5)|$ depending on $\alpha$, $i_0$ and $n$. The choice $i_0 = 1$ yields smaller systems of equations (cf. $N_n$ in Table 3) and smaller errors $ERR$ in comparison to the choice $i_0 = 0$. We have observed this surprising result also for $\Omega_2$ but not for all boundary values $U|_S$.

**6. 2. Convergence of the two-grid iteration.** Let us consider $\Omega_2$ and the two-grid method (2.8)-(2.11), where the coarse grid is defined as in Sect.5.2. The choice $i_0 = 1$ will be sufficient to get a fast iteration procedure. By (5.15) we get $\epsilon_p = C(\eta - 1)$ and from Sect.4.1

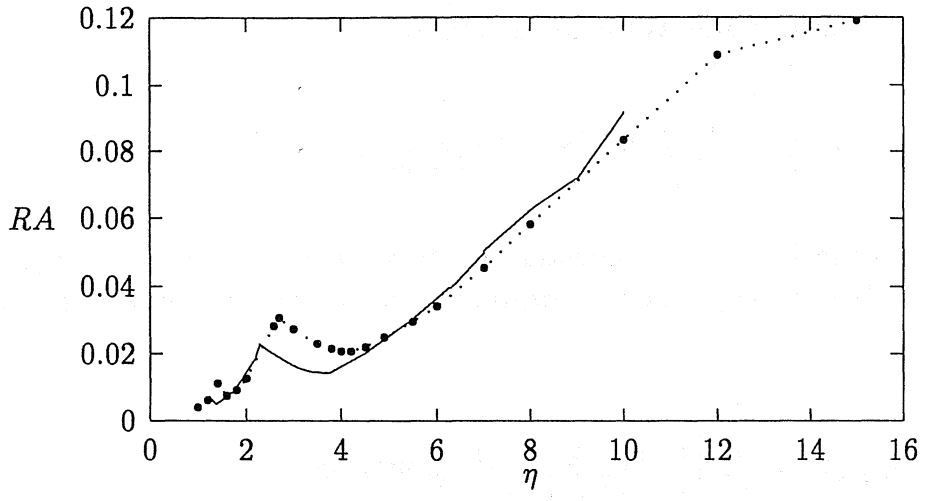


FIG. 4. *Convergence factor RA of the two-grid method, $\Omega_2$, Continuous line: $n = 6$, Dotted line: $n = 7$, Neglect of Str, $i_0 = 1$, $\alpha = 4/3$*

we conclude $\|Op\| < C\sqrt{\epsilon_p} = C\sqrt{\eta - 1}$. To confirm this dependence, let us introduce the estimate $RA$ for $\|Op\|$ by setting

$$RA := \sup_{i=2,\ldots} \frac{\|x^i - x_N\|}{\|x^{i-1} - x_N\|}. \tag{6.4}$$

$$\|x^i - x_N\| := \sup_{j=1,\ldots,N_n} |x^i(P^j) - x_N(P^j)|. \tag{6.5}$$

Some values for $RA$ are given in Fig.4. Indeed they show that $RA$ is smaller for smaller $(\eta - 1)$. We even get a satisfactory $RA$ if the coarse grid is obtained by dividing each face $F \setminus Str$ (cf. Sect.5.2.) into four equal squares. Since this choice leads to a small number $N_c$ and a small number of necessary operations in the iteration process, we have used this coarse grid for the computations in Table 8. If we do not neglect the strip $Str$, then the convergence factor $RA$ is much greater (equal to 0.4). On the other hand, if we do not neglect $Str$ but choose the coarse grid equal to the fine one over $Str$, then $RA$ is small again. Some of the values of $RA$ are presented in Fig.5. Again, we get a satisfactory $RA$ if the coarse grid is chosen equal to the fine one over $Str$ and equal to the four uniform squares over each face $F \setminus Str$. This choice has been used in the computations for the Tables 4 and 5. Note that, for this coarse grid, $\epsilon_p$ grows slightly if n becomes larger. In the computations for $\Omega_3$ (cf. the Tables 6 and 7) we have chosen $i_0 = 3$ as well as $\eta = 1.5$. For smaller $i_0$, the two-grid method over $\Omega_3$ diverges.

**6. 3. Comparison of several iteration procedures.** We have implemented the following algorithms for the solution of the system of equations (2.4).

a) GE: The first method is the Gaussian elimination (LU-factorization).

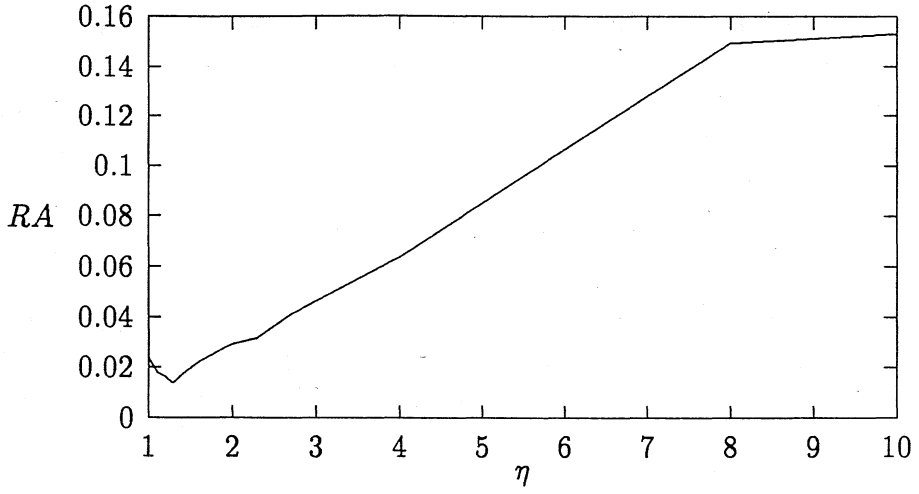


FIG. 5. *Convergence factor RA of the two-grid method, No neglect of Str,* $\Omega_2$, $i_0 = 1$, $\alpha = 4/3$, $n = 6$

b) GM: The second is the GMRES method (cf.[28]). Before applying this iteration, we multiply both sides of the equation by the inverse of the main diagonal of the matrix. The basis of the Krylov space is obtained by the method of Householder [32, 14, 31] and no restart is performed. A restart would result in slower convergence for $\Omega_1$ and $\Omega_2$ and in no convergence for $\Omega_3$.

c) TH: We consider the two-grid method of the form (2.8)-(2.11), where the coarse grid equation (2.10) is solved by GE. Note that this form of the two-grid iteration can be found for example in the book by Hackbusch [11].

d) TA: This is the method (2.8)-(2.11), where the coarse grid equation is solved by GE and the prolongation defined by Nyström's interpolation is replaced by piecewise constant interpolation. Note that piecewise linear prolongation would not improve the numerical results essentially. The use of Nyström's interpolation for the restriction and piecewise constant interpolation for the prolongation seems to be the most efficient variant of the two-grid iteration since only one multiplication by an $N \times N$-matrix is required. Moreover, we have computed the iterative solutions following the algorithm of Atkinson [4, 5]. Thus the iteration looks as follows: Start with the initial solution $x^0 := 0$ and suppose we are given the iterative solution $x^i$ over the node points of the fine grid. We compute the residual (defect) $r := \frac{1}{2}[1 - W_N(\chi)]^{-1}\{y - A_N x^i\}$ over the node points of the fine grid (multiplication of an $N \times N$-matrix times a vector). Then we determine $p := -2W_N(\chi r)$ over the node points of the coarse grid (multiplication of an $N_c \times N$-matrix times a vector). We solve the coarse grid equation $A_{N_c} d = p$ to get $d$ over the nodes of the coarse grid. Finally, we compute $x^{i+1} = x^i + r + d'$ over the nodes of the fine grid. Here $d'$ is the piecewise constant interpolation of $d$, i.e., $d'(P^i) = d(P_c^j)$ if the fine grid node $P^i$ belongs to the coarse grid subdomain $S_c^j$.

e) TA': This is the method TA with the only difference that the coarse grid equation is solved by GM.

f) PG: We consider the preconditioned GMRES method. Thus instead of solving $A_N x_N = y$ over the nodes of the fine grid, we solve $A^{(-1)}A_N x_N = A^{(-1)}y$, where $A^{(-1)}$ is an approximate matrix for the inverse of $A_N$. More exactly, $A^{(-1)}$ is the matrix which corresponds to one step of the iteration TA. .

g) PG': This is the same method as PG with the difference that $A^{(-1)}$ is the matrix which corresponds to one step of the iteration TA', i.e., the coarse grid equation in the two-grid preconditioner is solved by GM.

h) NE: This is the simple iteration (2.12) with $\kappa = 1$ and initial function $x^1 = y/2$.

i) JA: This is Jacobi's method.

In the Tables 4, 5, 6, and 7 we present the computing times $TI$ for the several methods applied to the solution of (2.4) over $\Omega_2$, $\Omega_3$, and $\Omega_1$. The time is given in CPU-seconds. The linear system of equations is solved up to an error of a tenth of the discretization error and the discretization error is the arithmetic mean of the errors $|x_N(P^i) - x(P^i)|$, $i = 1, \ldots, N$. Furthermore, the coarse grid equation in PG' is solved up to an error of $10^{-5}$ for $\Omega_3$ and $10^{-6}$ for $\Omega_2$. The coarse grid solution in TA' is computed up to $10^{-4}$ for $\Omega_3$ and up to a tenth of the discretization error for $\Omega_2$. The right-hand side is $y(P) = 6|P - (1,0,0)|^{-2}$ for $\Omega_2$ and $y(P) = 2|P - (1,0,\frac{1}{\sqrt{2}})|^{-1}$ for $\Omega_3$. Note that the number $RA$ for the algorithms GM, PG, and PG' is the geometric mean value of the ratios $\|x^i - x_N\| \,/\, \|x^{i-1} - x_N\|$. For GE, TH, TA, NE, and TA', the ratio $RA$ is defined as in (6.4). The number of iterations is denoted by $NI$. From the Tables 4, 5, 6, and 7 we learn that GM is the most effective solver. However, if $n$ is much higher than in the Tables 4-7, then we expect the methods TA' and PG' to be much faster than GM. We have tested higher values of $n$ for the domain $\Omega_1$. For $n = 11$ and $N_n = 2904$, GM requires 402 seconds whereas TA' only 372. The iteration PG does not seem to be an acceleration of the two-grid iteration though the convergence ratio $RA$ is the best for this algorithm . Numerical tests show that NE diverges for the domain $\Omega_3$.[8] For $\Omega_2$ it is quite fast and better than TA. Furthermore, let us mention that we have obtained similar results for $\alpha = \frac{4}{3}$ and $\alpha = 2$.

In our last numerical example let us consider $\Omega_1$ and neglect the strip $Str$. Let us consider the function (6.3) and choose $\alpha = 4/3$, $i_0 = 1$. In this situation $N_c = 24$ is already much smaller than $N$ and the two-grid algorithm should be the fastest. In Table 8 we present the CPU-time for the computation including JA, NE, GM, and TA. Indeed, it turns out that TA is faster. The number $RA$ for the algorithm GM is the geometric mean value of the ratios $\|x^i - x_N\| \,/\, \|x^{i-1} - x_N\|$ and the ratio (6.4) for JA, NE, and TA. Thus the convergence factors of the iteration procedures are smaller for TA and, therefore, a smaller number of iteration steps is needed. The method TA can be accelerated by choosing $\eta = 10$ (cf. the introduction of the coarse grid in Sect.5.2). Then, for e.g. $n = 11$, TA requires 299 seconds only. Note that the time for the computation of the matrices is greater than that for the iteration process provided that $n$ is large and that TA, GM, or TA' are applied. For instance, 244 seconds of the 318 for TA and 235 seconds of the 353 for GM (cf. Table 8) are needed in order to generate the matrices. All the calculations for the CPU-time (cf. Tables 4-8 )

[8] The reason for this seems to be that the actual number $\epsilon_p$ (cf. (3.3)) for the grids are not small enough. We conjecture that Theorem 4.2 remains true even if (3.2) is not satisfied.

have been performed in double-precision arithmetic on a VAX 4000-300.

## 7. Code for the two-grid method TA'.

In this section we shall describe the sequential code for method TA' (cf. Sect. 6.3 d) and e)). It is one of the fastest methods tested in Sect. 6.3 and the fastest variant of the two-grid method (2.8)-(2.11). The whole code is written in FORTRAN 77. It consists of a main program and some subroutines. The main program performs the following steps:

- input of parameters
- generation of the grids (subroutine GRID),
- computation of the matrices (subroutines MATRIX, DIAG),
- computation or input of the boundary values (subroutine DIRICH),
- two-grid iteration due to Atkinson (including subroutine GMRES for the solution of the coarse grid equation),
- output of the solution.

In the sequel the codes for the main program and the subroutines MATRIX and DIAG are given. For the subroutines DIRICH, GRID, and GMRES, only necessary input and output parameters are described.

Before we start with the code, let us mention that in the implementation of TA' there arise three different matrices A, C, and D. All these matrices are of the same structure. Let $P^j$ be the midpoint of $S^j$ and $n_P$ the unit vector of the interior normal to $\Omega$ at $P$ and set

$$w_{ij} := \frac{1}{2\pi} \frac{n_{P^j} \cdot (P^i - P^j)\mu(S^j)}{|P^i - P^j|^3}, \; i = 1, \ldots, N, j = 1, \ldots, N. \tag{7.1}$$

Then the restriction of operator $2W_N\chi I$ (cf. (2.7)) to the grid $\{P^i, i = 1, \ldots, N\}$ has the matrix representation $A = (w_{ij})_{i,j=1}^N$. If in (7.1) the points $P^j, P^i$, $i, j = 1 \ldots, N$ are replaced by the coarse grid points $P_c^j, P_c^i$, $i, j = 1 \ldots, N_c$ and the weights $\mu(S^j)$ by $\mu(S_c^j)$ (cf. Sect. 2.2), then the restriction of $2W_{N_c}\chi I$ to $\{P_c^i, i = 1, \ldots, N_c\}$ has the matrix representation $D = (w_{ij})_{i,j=1}^{N_c}$. Moreover, if in (7.1) the $P^j$ and $S^j$ remain and the $P^i$ are replaced by $P_c^i$, then we get the matrix $C = (w_{ij})_{i=1,\ldots,N_c, j=1,\ldots,N}$ which corresponds to the restriction of $2W_N\chi I$ acting from the fine grid to the coarse grid. The last restriction appears in the Nyström interpolation. The restriction of $A_N$ (cf. (2.6)) to the fine grid has the matrix representation $A + DIAG\_A$, where $DIAG\_A$ stands for a certain diagonal matrix. Note, that $DIAG\_A$ is just that diagonal matrix for which all row sums of $A + DIAG\_A$ are equal to 2. Similarly, the restriction of $A_{N_c}$ (cf. (2.10)) to the coarse grid has the representation $D + DIAG\_D$.

```
C***********************************************************************
C        main program
C***********************************************************************
C   double layer potential equation for Laplacian,
C   interior Dirichlet problem for a polyhedron,
C   calculation of the double layer weight function,
```

```
C   quadrature method (Nystroem's method) with singularity subtraction,
C   generation of a fine and a coarse grid,
C      subdivision into rectangles or triangles,
C      first each face of the polyhedron is divided into strips parallel
C      to the edges, then each strip is divided into nearly uniform
C      rectangles or triangles,
C      mesh grading perpendicular to the edges,
C      degree of grading for the fine mesh controlled by parameter ALPHA
C      (cf. (5.1)),
C      degree of grading for the coarse mesh controlled by parameter ETA
C      (cf. Sect. 5.2),
C      neglect of a small number of strips near the edges,
C      choose fine and coarse grid such that they coincide over
C      a small number of strips near to the edges,
C   application of midpoint rule as quadrature formula,
C   solution of the resulting linear system by the two grid method TA'
C      (cf. Sect. 6.3 d), e)),
C      two-grid iteration algorithm due to Atkinson,
C      prolongation by piecewise constant interpolation,
C      solution of the coarse grid equation by GMRES.
C*********************************************************************
C   NMAX1 = maximal number of subdomains of the fine grid
C   NMAX2 = maximal number of subdomains of the coarse grid
C*********************************************************************
      INTEGER NMAX1,NMAX2
      PARAMETER (NMAX1=3035, NMAX2=225)
      DOUBLE PRECISION
     *  A(NMAX1,NMAX1),C(NMAX2,NMAX1),D(NMAX2,NMAX2),
     *  PX(NMAX1),PY(NMAX1),PZ(NMAX1),
     *  NX(NMAX1),NY(NMAX1),NZ(NMAX1),
     *  RX(NMAX2),RY(NMAX2),RZ(NMAX2),
     *  NRX(NMAX2),NRY(NMAX2),NRZ(NMAX2),
     *  S(NMAX1),SR(NMAX2),
     *  X0(NMAX1),X1(NMAX1),X2(NMAX1),
     *  Y0(NMAX2),Y1(NMAX2),
     *  DIAG_A(NMAX1),DIAG_D(NMAX2),
     *  ALPHA,ETA,R0,R05,RM6
      INTEGER LITTN,I,J,J1,K,L,I0,NEGLEC,ITER,KRYL,IND(NMAX1)
      DATA R0,R05,RM6/0D0,0.5D0,1D-6/
C
C*********************************************************************
C   input: ALPHA  - grading parameter (cf. (5.1)),
C          LITTN  - n = number of strips parallel to the edges,
C          NEGLEC - number of neglected strips ,
C          I0     - number of strips where fine and coarse grid
```

```
C                   coincide,
C          ITER   - number of steps of the two-grid iteration,
C          KRYL   - maximal dimension of Krylov space for GMRES,
C          ETA    - grading parameter (cf. Sect. 5.2).
C***********************************************************************
C
      WRITE(6,*) ' INPUT: ALPHA, LITTN, NEGLEC, I0, ITER, KRYL, ETA'
      READ (5,*) ALPHA, LITTN, NEGLEC, I0, ITER, KRYL, ETA
C
C***********************************************************************
C   calculation of the
C     midpoints of the subdomains for the
C      -fine grid (PX, PY, PZ: x-, y-, z-coordinate),
C      -coarse grid (RX, RY, RZ: x-, y-, z-coordinate),
C     interior normals of the subdomains for the
C      -fine grid (NX, NY, NZ: x-, y-, z-coordinate),
C      -coarse grid (NRX, NRY, NRZ: x-, y-, z-coordinate),
C     quadrature weights of the
C      -fine grid (S),
C      -coarse grid (SR),
C     number of subdomains of the
C      -fine grid (K),
C      -coarse grid (L),
C     index array IND with the property:
C      J = IND(I), if point (PX(I),PY(I),PZ(I)) of fine grid belongs
C      to subdomain on coarse grid containing (RX(J),RY(J),RZ(J))
C***********************************************************************
C
      CALL GRID(ALPHA,ETA,LITTN,NEGLEC,I0,NMAX1,NMAX2,
     *          PX,PY,PZ,NX,NY,NZ,S,K,
     *          RX,RY,RZ,NRX,NRY,NRZ,SR,L,IND)
C
C***********************************************************************
C
      IF(K.GT.NMAX1 .OR. L.GT.NMAX2) THEN
         WRITE (6,*) 'Stop because NMAX1 or NMAX2 is exceeded!'
         STOP
      END IF
C
C***********************************************************************
C   determination of the matrices (7.1):
C     - A: fine grid matrix,
C     - C: matrix defining the transition of fine grid data
C          to the coarse grid,
C     - D: coarse grid matrix,
```

```
C   separate determination of the diagonals DIAG_A and DIAG_D
C*********************************************************************
C
      CALL MATRIX(NMAX1,NMAX1,K,K,A,PX,PY,PZ,NX,NY,NZ,S,PX,PY,PZ)
C
      CALL MATRIX(NMAX2,NMAX1,L,K,C,PX,PY,PZ,NX,NY,NZ,S,RX,RY,RZ)
C
      CALL MATRIX(NMAX2,NMAX2,L,L,D,RX,RY,RZ,NRX,NRY,NRZ,SR,RX,RY,RZ)
C
      CALL DIAG(NMAX2,NMAX1,L,K,C,DIAG_D)
C
      CALL DIAG(NMAX1,NMAX1,K,K,A,DIAG_A)
C
      DO I=1,L
         D(I,I) = DIAG_D(I)
      END DO
C*********************************************************************
C   calculation of Dirichlet boundary values, X0 contains
C   the doubled values
C*********************************************************************
      CALL DIRICH(NMAX1,K,X0,PX,PY,PZ)
C
C*********************************************************************
C   two-grid iteration TA' (cf. Sect. 6.3 d), e)) realised by loop 1000
C*********************************************************************
C
C   initial solution X1:
      DO I=1,K
         X1(I)=R0
      END DO
C
      DO 1000 J1=1,ITER
C
         IF(J1 .EQ. 1) THEN
            DO I=1,K
               X2(I) = X0(I)/DIAG_A(I)
            END DO
         ELSE
C
C   calculation of dotproducts TX(I) = A * X1 :
           DO I=1,K
              TX(I) = R0
           END DO
           DO J=1,K
              DO I=1,K
```

```
                TX(I) = TX(I) + A(I,J)*X1(J)
              END DO
            END DO
C
C   determination of the defect r=X2:
            DO I=1,K
               X2(I) = (X0(I) - DIAG_A(I)*X1(I) - TX(I))/DIAG_A(I)
            END DO
          END IF
C
C   determination of p=Y1 on the coarse grid:
          DO I=1,L
             Y1(I) = R0
          END DO
          DO J=1,K
             DO I=1,L
                Y1(I) = Y1(I) - C(I,J)*X2(J)
             END DO
          END DO
C
C  solution of the coarse grid equation by GMRES iteration:
C    input/output:
C      Y0     : initial solution and final solution.
C    input:
C      D      : matrix, L: order of D
C      Y1     : right-hand side of linear system
C      KRYL   : maximal dimension of Krylov subspace
C      RM6    : prescribed error
C    output:
C      IANZ   : number of iteration steps performed to reach an error
C               less than the prescribed error; if this is not possible
C               within KRYL steps, then IANZ=KRYL
C      TOLSCH: error estimate for the solution Y0
C
          DO I=1,L
             Y0(I) = Y1(I)*R05
          END DO
C
          CALL GMRES(L,NMAX2,D,Y1,KRYL,RM6,IANZ,TOLSCH,Y0)
C
C   new iterative solution X1: X1 + r + prolongation(d), d=Y0:
C     prolongation is done by piecewise constant interpolation
C     with the help of index array IND
C
          DO I=1,K
```

```
            X1(I) = X1(I) + X2(I) + Y0(IND(I))
          END DO
C
C   end of iteration
C
 1000 CONTINUE
C*********************************************************************
C   discrete solution of the double layer potential equation is X1
C*********************************************************************
      WRITE(6,*) ' solution of the double layer potential equation:'
      WRITE(6,*) (X1(I), I=1,K)
      STOP
      END
C.....................................................................
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

C*********************************************************************
C   file MATRIX.FOR
C*********************************************************************
C   generation of matrix (7.1)
C     input:
C         NMAX2: maximal number of rows of matrix 2W
C         NMAX1: maximal number of columns of matrix 2W
C         N2   : actual number of rows of the matrix,
C                dimension of RX, RY, RZ
C         N1   : actual number of columns of the matrix,
C                dimension of PX, PY, PZ, NX, NY, NZ, S
C         PX,PY,PZ: midpoints of the subdomains of one grid,
C                   x-, y- and z-coordinates
C         NX,NY,NZ: interior normals of the subdomains of this first grid,
C                   x-, y- and z-coordinates
C         S    : quadrature weights
C                (areas of the subdomains of the first grid)
C         RX,RY,RZ: midpoints of the subdomains of a second grid,
C                   x-, y- and z-coordinates
C     output:
C         A    : matrix (7.1)
C*********************************************************************
      SUBROUTINE MATRIX(NMAX2,NMAX1,N2,N1,A,PX,PY,PZ,NX,NY,NZ,S,
     *           RX,RY,RZ)
C*********************************************************************
      INTEGER NMAX1,NMAX2,N1,N2
      DOUBLE PRECISION PX(NMAX1),PY(NMAX1),PZ(NMAX1),S(NMAX1),
     *      NX(NMAX1),NY(NMAX1),NZ(NMAX1),RX(NMAX2),RY(NMAX2),RZ(NMAX2),
     *      A(NMAX2,NMAX1),X1,X2
```

```
      DOUBLE PRECISION R0,R2,RM15,PI
      INTEGER I,J
      DATA R0,R2,RM15 /0D0,2D0,1D-15/
      DATA PI / 0.31415926535897932384626433832795028D+01/
C
      DO J=1,N1
         DO I=1,N2
            X1 = NX(J)*(RX(I)-PX(J)) + NY(J)*(RY(I)-PY(J)) +
     *           NZ(J)*(RZ(I)-PZ(J))
            IF(ABS(X1) .LT. RM15) THEN
               A(I,J) = R0
            ELSE
               X2 = SQRT((RX(I)-PX(J))**2 + (RY(I)-PY(J))**2 +
     *              (RZ(I)-PZ(J))**2)
               A(I,J) = -S(J)*X1/(R2*PI*X2**3)
            END IF
         END DO
      END DO
C
      RETURN
      END
C.....................................................................
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

C**********************************************************************
C   file DIAG.FOR
C**********************************************************************
C   calculation of DIAG_A or DIAG_D (cf. comments after (7.1))
C     input:
C         NMAX2: maximal number of rows of matrix A
C         NMAX1: maximal number of columns of matrix A
C         L    : actual number of rows of the matrix
C         K    : actual number of columns of the matrix
C         A    : matrix (7.1), calculated by subroutine MATRIX
C     output:
C         DI(NMAX2): diagonal of the matrix
C**********************************************************************
      SUBROUTINE DIAG(NMAX2,NMAX1,L,K,A,DI)
C**********************************************************************
      INTEGER NMAX1,NMAX2,K,L
      DOUBLE PRECISION A(NMAX2,NMAX1),DI(NMAX2)
      DOUBLE PRECISION S,R0,R2
      INTEGER I,J
      DATA R0,R2 /0D0,2D0/
C
```

```
      DO I=1,L
         DI(I) = R0
      END DO
      DO J=1,K
         DO I=1,L
            DI(I) = DI(I) + A(I,J)
         END DO
      END DO
      DO I=1,L
         DI(I) = R2 - DI(I)
      END DO
C
      RETURN
      END
```

## 8. Implementation of the two-grid method on a massively parallel computer.

**8. 1. Analysis of storage and parallelization of the two-grid method.** Let us retain the notation of $A, C$, and $D$ from the last section and denote the matrix $(w_{ij})_{i=1,\ldots,N,j=1,\ldots,N_c}$ by $B$, where $w_{ij}$ is given by (7.1) after replacing $S^j$ by $S_c^j$ and $P^j$ by $P_c^j$. Suppose we are given the coarse grid data $\{u_j\}$. Then the determination of the Nyström interpolant (2.5) at the point $P = P^i$ requires the computation of the scalar product of the i-th row of $B$ times the vector $\{u_j\}$ as well as the computation of the i-th row sum:

$$v_i = \frac{y_i - \sum_{j=1}^{N_c} w_{ij} u_j}{2 - \sum_{j=1}^{N_c} w_{ij}}, \quad i = 1, \ldots, N. \tag{8.1}$$

If we replace $P^i$ by $P_c^i$ or $P_c^j$ by $P^j$, then a similar formula holds provided the entries $w_{ij}$ of $B$ are replaced by those of $A, C$, and $D$, respectively.

Each iteration step of the two-grid method (2.8)-(2.11) consists of the following operations:

- two Nyström interpolations of fine grid data to the fine grid,
- one Nyström interpolation of coarse grid data to the fine grid (prolongation),
- two Nyström interpolations of fine grid data to the coarse grid (restriction),
- the solution of a linear system on the coarse grid
- a multiplication (of complex manner).

Supposing that the different dense matrices $A$, $B$, $C$, and $D$ are stored in RAM, we need $(N^2 + 2NN_c + {N_c}^2) \times 8$ bytes of memory for arrays in double precision (16 decimals). Further let $N_c$ be small (say 0) and the available memory on our computer be large (say 128 megabytes). Then, nevertheless, $N$ is less than 4100 or even smaller. Depending on $\alpha, i_0$ and the domain $\Omega$, the number $n$ of strips on every face (cf. (5.1)) of $\Omega$ is limited at least by 14 which is obtained in the best case where $\alpha = 1$, strip $Str$ is neglected, and $\Omega = \Omega_1$. This number is to small to get a high accuracy for the approximate solution determined by (2.3).

Especially, this is not sufficient if we wish to determine the estimate $EX_n$ for the order of convergence (cf. Sect. 6.1 and Tables 1 and 2).

To overcome this limitation we have two possibilities:

1. compute the matrices once and write them to a disk or
2. compute the matrices every time when they are needed.

Because we want to treat problems with up to 100,000 subdomains on the fine grid the first way would require more than $10^5 \times 10^5 \times 8 = 80$ gigabytes disk storage. If a high speed disk would have the desired capacity and if the I/O disk rate is 10 megabytes per second (a high one), it would take 8,000 seconds (more than 2 hours) to transfer this amount of data once from memory to disk or vice versa. Therefore, let us choose the second way.

For the generation of the matrices $A$, $B$, $C$, and $D$, the distances of a fixed point to all the other points and some additional elementary operations are to be computed. The main part can be organized as a sequence of elementwise array operations. Therefore, both the Nyström interpolation and the matrix generation are suitable for a pipeline code. A massively parallel computer on which such a code would run very fast is the MP-1 (or newly the MP-2) computer from the MasPar Corporation.

**8. 2. Description of the massively parallel computer.** All code has been developed and tested on the MasPar MP-1216 at the Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR) of Stuttgart University. This computer has a fine grained, massively and data parallel architecture (Single-Instruction-stream Multiple-Data-stream = SIMD), cf. [21]. The SIMD architecture is a good match for applications involving massive amounts of data elements, all being processed in a parallel manner like in great parts of our algorithm. The MP-1216 at the IPVR has 16,384 processing elements arranged on a 128 by 128 grid and a total of 256 megabytes of high speed memory. Each processor contains 16 kilobytes of local memory and is able to communicate directly with any of its eight nearest neighbours via the fast X-Net or to use the Global Router for 1024 simultaneous connections to arbitrary processors. This configuration has a peak performance of about 550 megaflops (double precision).

Let us remark:

- On the MasPar-system of the IPVR the I/O rate is rather 1 megabyte per second than 10 megabytes per second. So one transfer from RAM to disk would rather take 22 than 2 hours.
- Some weeks ago MasPar has offered a high speed disk array with an I/O rate of 60 megabytes per second. Unfortunately, it is very expansive.

Thus our choice to generate any matrix once again at any time of occurrence is the only possible for a high number of strips.

The code is written in FORTRAN 90 because a FORTRAN 77 code for a serial and a vector computer has been established earlier. The FORTRAN 77 code has been translated into FORTRAN 90 by VAST-2 and then adapted by hand to get high performance. An important point for coding an algorithm in FORTRAN 90 for the MasPar system is the array mapping (cf. [20]). In general every individual array element is placed on one processor. A

vector (one-dimensional data array) is mapped onto the two-dimensional processor grid in a serpentine fashion, beginning with the first data element in the upper left corner of the processor grid, continuing with the first column of the processor grid, going to the second one and so forth until the last column is filled or the data array ends. If the number of entries of the vector is greater than that of processors in the processor grid, a next layer of the local memory of the processor grid is filled until all data elements are in the local memory of the processors. Two-dimensional arrays (matrices) are mapped to the processor grid in a way called cut-and-stack. Imagine a stamp as great as the processor grid and stamp the data matrix with this stamp beginning in the upper left corner of the matrix. (With a stamp of $128 \times 128$ processors a block of $128 \times 128$ data elements will be stamped.) Continue stamping to the right of the matrix until all data elements of the first 128 rows and possibly also some free places at the right hand side of the matrix are stamped. Then continue stamping a second stamp row and so forth until all matrix-elements are stamped. Enumerating the stamped blocks of the matrix in the order of stamping, we get the layers in the local memory where the data-blocks of the matrix are stored.

The following simple example of multiplying the elements of two arrays will show the data-parallel model and the use of parallel operations (cf. [19]). The data-parallel code reads as:

```
PARAMETER ( N = 16000 )
DOUBLE PRECISION A(N), B(N), C(N)
C = A * B
```

where N (i.e. 16,000) processors are performing the calculation in parallel. Its execution looks like:

a) load A into register1 on all processing elements
(i.e. A(1) $\Longrightarrow$ PE1, A(2) $\Longrightarrow$ PE2, ...),
b) load B into register2 on all processing elements
(i.e. B(1) $\Longrightarrow$ PE1, B(2) $\Longrightarrow$ PE2, ...),
c) multiply register1 with register2, store the result into register3 on all processing elements
d) store register3 into C on all processing elements
(i.e. PE1 $\Longrightarrow$ C(1), PE2 $\Longrightarrow$ C(2), ...).

Notice that there are no loops in the data-parallel code. The concept of one element per processor is expressed by omitting the loop and defining operations that act on each element of a data set. Moreover, note the implicit ordering of data, i.e., the elements A(1) and B(1) are defined to reside in the same processor's memory. If the vector B is to be multiplied by a constant, then at first this constant is to be broadcasted to all PEs and the execution a)–d) can be performed.

**8. 3. Parallel code.** Let us show and comment the parallel code for the time-consuming parts of the two-grid method (2.8)-(2.11). These parts are the matrix generation and the Nyström interpolation. Both are implemented in the same subroutine, first computing a part of the matrix and then using it immediately for the Nyström interpolation. To improve the performance, the algorithm is adapted to two different situations, namely, a small number of rows together with a large number of columns and vice versa.

At first we present the code MCNYFI, where the case of a small number of rows and a great

number of columns is implemented. We shall call it the row oriented code (ROC).

```
C***********************************************************************
C        file mcnyfi.for
C***********************************************************************
C        Generation of matrix C (cf. (7.1)) and the Nystroem interpolation
C        from fine grid data to the coarse grid (restriction).
C        This corresponds to a small number of rows and a large number
C        of columns of the matrix.
C     input:
C         NMAX2: maximal number of subdomains on the coarse grid
C         NMAX1: maximal number of subdomains on the fine grid
C         L    : actual number of subdomains on the coarse grid
C                = number of rows of the matrix,
C                dimension of RX, RY, RZ, X, Z
C         K    : actual number of subdomains on the fine grid
C                = number of columns of the matrix,
C                dimension of PX, PY, PZ, NX, NY, NZ, S, Y
C         PX,PY,PZ: midpoints of the subdomains of the fine grid,
C                   x-, y- and z-coordinates
C         NX,NY,NZ: interior normals of the subdomains of the fine grid,
C                   x-, y- and z-coordinates
C         S    : quadrature weights
C                (areas of the subdomains of the fine grid)
C         RX,RY,RZ: midpoints of the subdomains of the coarse grid,
C                   x-, y- and z-coordinates
C         Y    : vector which is to be interpolated
C         Z    : vector of the right-hand side
C     output:
C         X    : resulting vector
C
C***********************************************************************
      SUBROUTINE MCNYFI(NMAX2,NMAX1,L,K,PX,PY,PZ,NX,NY,NZ,S,RX,RY,RZ,
     1                  X,Y,Z)
C***********************************************************************
      INTEGER NMAX2,NMAX1,L,K
      DOUBLE PRECISION, INTENT (IN), ARRAY (NMAX2) ::
     *                                    RX, RY, RZ, Z
      DOUBLE PRECISION, INTENT (IN), ARRAY (NMAX1) ::
     *                                    PX, PY, PZ, S, NX, NY, NZ, Y
      DOUBLE PRECISION, INTENT (OUT), ARRAY (NMAX2) :: X
C
C     local variables:
C
      INTEGER I,J
```

```
      DOUBLE PRECISION UV1(K), UV2(K), UV3(K), UV4(K), W(L)
      DOUBLE PRECISION R0,R2,RM15,PI,PIT
      DATA R0,R2,RM15 /0D0,2D0,1D-15/
      DATA PI / 0.31415926535897932384626433832795028D+01/
C
      PIT = 1D0/(R2*PI)
C
C     generate one row UV4 of the matrix and
C     compute the scalar product X of UV4 and Y as well as
C     the row-sum W.
C
      DO I=1,L
C        broadcast the coordinates (RX,RY,RZ) of the points with index I
C        to the processing elements of all other points:
         UV1(:K) = RX(I)
         UV2(:K) = RY(I)
         UV3(:K) = RZ(I)
C
C        determine the differences in the coordinates of all points to
C        the broadcasted point:
         UV1(:K) = UV1(:K) - PX(:K)
         UV2(:K) = UV2(:K) - PY(:K)
         UV3(:K) = UV3(:K) - PZ(:K)
C
C        determine the numerator of equation (7.1):
         UV4(:K) = S(:K)*(NX(:K)*UV1(:K) + NY(:K)*UV2(:K) +
     1             NZ(:K)*UV3(:K))
C
C        determine the denominator of equation (7.1):
C        code is faster in this form than with immediate computation
C        of the power to 3/2:
         UV3(:K) = SQRT(UV1(:K)**2 + UV2(:K)**2 + UV3(:K)**2)**3
C
C        compute the ratio in equation (7.1) for those denominators
C        which are different from zero:
         WHERE ( UV3(:K) .LT. RM15 )
            UV4(:K) = R0
         ELSE WHERE
            UV4(:K) = (UV4(:K)) / (UV3(:K))
         END WHERE
C
C        perform the scalar product of the numerator of the
C        Nystroem interpolation step (2.5):
         X(I) = -PIT*DOTPRODUCT(UV4(:K),Y(:K))
C
```

```
C        determine the row sum of the denominator of (2.5):
         W(I) = -PIT*SUM(UV4(:K))
      END DO
C
C     perform the operation of equation (2.5):
      X(:L)=(Z(:L) - X(:L)) / (R2 - W(:L))
      RETURN
      END
```

After broadcasting the coordinates of one point to all processor elements (a less parallel operation), all further operations until the WHERE-statement are done in parallel, i.e., simultaneously on all processing elements. The WHERE-statement is the parallel version of the scalar IF-clause. The condition after the WHERE is checked on all processing elements, i.e., for all array elements simultaneously. The assignment statement following the logical expression is executed for all elements having the value .TRUE., and the assignment statement following the keyword ELSEWHERE is executed for all elements having the value .FALSE.. Consequently, this is done in two steps. The computation of the scalar product and the vector sum are done with the FORTRAN 90 intrinsic functions DOTPRODUCT and SUM, respectively. These functions are tuned for the MasPar system and realize a quite good performance.

Now we present the code for subroutine MBNYFI, where the case of a large number of rows and a small number of columns is implemented. We call this code the column oriented code (COC).

```
C***********************************************************************
C        file mbnyfi.for
C***********************************************************************
C        Generation of matrix B (cf. (7.1)) and the Nystroem interpolation
C        from the coarse grid data to the fine grid (prolongation).
C        This corresponds to a large number of rows and a small number
C        of columns of the matrix.
C     input:
C         NMAX1: maximal number of subdomains on the fine grid
C         NMAX2: maximal number of subdomains on the coarse grid
C         K    : actual number of subdomains on the fine grid
C                = number of rows of the matrix,
C                dimension of PX, PY, PZ, X, Z
C         L    : actual number of subdomains on the coarse grid
C                = number of columns of the matrix,
C                dimension of RX, RY, RZ, NRX, NRY, NRZ, SR, Y
C         RX,RY,RZ   : midpoints of the subdomains of the coarse grid,
C                      x-, y- and z-coordinates
C         NRX,NRY,NRZ: interior normals of the subdomains of the coarse grid,
C                      x-, y- and z-coordinates
C         SR   : quadrature weights
```

```
C                (areas of the subdomains of the coarse grid)
C         PX,PY,PZ  : midpoints of the subdomains of the fine grid,
C                     x-, y- and z-coordinates
C         Y     : vector which is to be interpolated
C         Z     : vector of the right-hand side
C     output:
C         X     : resulting vector
C
C*********************************************************************
      SUBROUTINE MBNYFI(NMAX1,NMAX2,K,L,RX,RY,RZ,NRX,NRY,NRZ,SR,PX,PY,PZ,
     1                  X,Y,Z)
C*********************************************************************
      INTEGER NMAX1,NMAX2,K,L
      DOUBLE PRECISION, INTENT (IN), ARRAY (NMAX1) ::
     *                                  PX, PY, PZ, Z
      DOUBLE PRECISION, INTENT (IN), ARRAY (NMAX2) ::
     *                                  RX, RY, RZ, SR, NRX, NRY, NRZ, Y
      DOUBLE PRECISION, INTENT (OUT), ARRAY (NMAX1) :: X
C
C     local variables:
C
      INTEGER I,J
      DOUBLE PRECISION R0,R2,RM15,PI,PIT
      DOUBLE PRECISION UV1(K), UV2(K), UV3(K), UV4(K), W(K)
      DATA R0,R2,RM15 /0D0,2D0,1D-15/
      DATA PI / 0.31415926535897932384626433832795028D+01/
C
      PIT = 1D0/(R2*PI)
C
C     generate one column UV4 of the matrix and
C     add the product of this column by an element of vector Y
C     to the actual scalar products X of all rows,
C     add this column to the actual sums W of all rows.
C
      X(:K) = R0
      W(:K) = R0
      DO J=1,L
C        determine the differences in the coordinates of all points to
C        the J-th point:
         UV1(:K) = PX(:K) - RX(J)
         UV2(:K) = PY(:K) - RY(J)
         UV3(:K) = PZ(:K) - RZ(J)
C
C        determine the numerator of equation (7.1):
         UV4(:K) = SR(J)*(NRX(J)*UV1(:K) + NRY(J)*UV2(:K) +
```

```
     1             NRZ(J)*UV3(:K))
C
C        determine the denominator of equation (7.1):
C        code is faster in this form than with immediate computation
C        of the power to 3/2:
         UV3(:K) = SQRT(UV1(:K)**2 + UV2(:K)**2 + UV3(:K)**2)**3
C
C        compute the ratio in equation (7.1) for those denominators
C        which are different from zero:
         WHERE ( UV3(:K) .LT. RM15 )
            UV4(:K) = R0
         ELSE WHERE
            UV4(:K) = (UV4(:K)) / (UV3(:K))
         END WHERE
C
C        actualise the sums and the scalar products of all rows
         X(:K) = X(:K) + UV4(:K)*Y(J)
         W(:K) = W(:K) + UV4(:K)
      END DO
C
C     perform the operation of equation (2.5):
      X(:K)=(Z(:K) + PIT*X(:K)) / (R2 + PIT*W(:K))
      RETURN
      END
```

After one column UV4 is computed, its values are added to the vectors X and W assigned on the DPU and containing the actual scalar products and row sums of all rows (starting value zero). Because L is small in comparison to K ($L \leq$ 4,000, $K \leq$ 100,000 ) long vector operations (additions and multiplications) are performed and the processing elements are working efficiently. Using the ROC instead, only a great number of short scalar products and row sums are to be evaluated. If L = 4,000, only about a fourth of the 16,384 processing elements would compute the row sum. Often L is less than 4,000 and only a much smaller number of PEs is involved. The COC is also used in the case that the number of rows and that of columns are equal. The ROC is not efficient in this case, too.

Table 9 shows the times effected by the use of different combinations of the two codes for the generation of $A$, $B$, and $C$. The example is computed for the case of L-block $\Omega_2$, no neglect of strip $Str, i_0 = 1, \alpha = 1, \eta =$ 1,000, and 3 Iterations. For $n = 18$, we get $N =$ 15,644, $N_c$ = 1,740.

On the MP-1216 two variants of the algorithm TH (cf. Sect. 6.3 c)) are implemented. TH together with matrices $B$, $C$, and $D$ in RAM will be called THG. Because the coarse grid solver GE has not been adapted to the parallel machine[9], only examples with a small number

[9] Note that the array mapping of the MasPar system is not very suitable for the implementation of GE. In contrary, the array mapping fits for algorithms like GMRES, where the main part consists of multiplications matrix times vector.

$N_c$ of coarse grid subdomains could be handled with THG. Usually, $N_c$ is small if strip $Str$ is neglected, $i_0 = 1$ and $\eta$ is large. The second variant of TH will be called THR. The only difference with respect to THG is that the coarse grid equation is solved by GM and that the coarse grid matrix $D$ is the only matrix stored in RAM. For the evaluation of the iterative GMRES solution, in every iteration step it is necessary to multiply the coarse grid matrix by a vector. This matrix times vector operation is also realized in parallel. The percentage of GMRES time in comparison to that of the whole method for the example in Table 9 is below 6% and will decrease for higher $n$. So we omit this parallel code. With algorithm THR it is possible to handle problems up to $N_c \leq 4{,}000$.

In Table 10 we present the CPU-times T2 for method THG on the MP-1216 and compare it with the CPU-times T1 on a Convex 210 with pipeline-processor which has a peak performance of about 50 megaflops. The times are given in CPU-seconds. On the Convex method TH is implemented in its original form, i.e., matrix $A$ is stored in RAM, too. Since the Convex at our institute has only a RAM of 60 megabytes, we must emphasize that problems with $N_n$ greater or equal to 2,500 are swapped to disk by the system and the computing time increases rapidly[10]. The example of the numerical test is computed for $\Omega_2, S = \partial\Omega$ and the function $U(P) = \sqrt{11/4}|P - (-1,0,0)|^{-1}$. The error ERR is defined as in Sect. 6.1 and RA as in (6.4). As parameters for the method we use $\alpha = 4/3$, neglect of strip $Str, i_0 = 1$ and 5 iterations are performed.

In Table 11 further CPU-times are presented for the same boundary value but another domain $\Omega$. Let $\Omega = \Omega_4$ be the rectangular domain of Fig.6 which is rectangular but locally not the graph of a Lipschitz function. T3 represents CPU-times for method THR on the MP-1216.

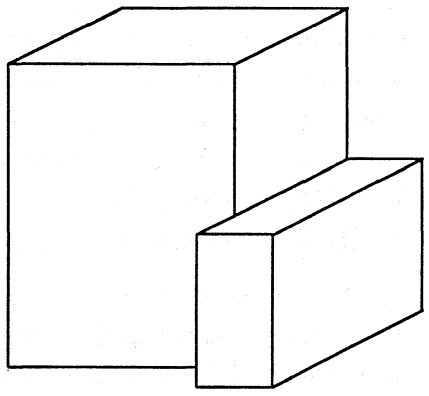

FIG. 6. *Domain* $\Omega_4$ *:=* $(0,0.75)\times(0.25,1)\times(0,1)\cup(0.75,1)\times(0,1)\times(0,0.5)$.

The tables clearly show that it is impossible to treat problems with up to 100,000 subdomains on the fine grid on a Convex 210 with only 64 megabytes of memory. Even the MP-1216 requires some hours to treat these problems. Finally, let us mention that, on the MasPar MP-1216 with a peak performance of 550 megaflops, our code reaches 234 megaflops. Namely,

[10] cf. the 6,649 seconds for $N_n$=2,786 in column $T1$ of Table 10 and the 1,016 seconds for $N_n$=3,274 in column $T1$ of Table 11.

for the case of $\Omega_2$, $\alpha = 4/3$, $i_0 = 1$, $N = 96,706$, $N_c = 242$, five iterations, and neglect of $Str$, the $2.07 \cdot 10^{12}$ arithmetic operations require a time of 2 hours 27 minutes and 10 seconds.

**Acknowledgement.** We would like to thank Dr.J.Reusch from MasPar Computer GmbH nearby Munich for the permanent help during the implementation of the parallel code on the MP-1216 at Stuttgart University. Further, we wish to thank R.Schlundt for placing the sequential GMRES code at our disposal.

## REFERENCES

[1] T.S.ANGELL, R.E.KLEINMAN AND J.KRAL, *Layer potentials on boundaries with corners and edges*, Casopis pro pestovani matematiky, 113 ,4 (1988), pp. 387-402.

[2] K.E.ATKINSON, *A survey of boundary integral equation methods for the numerical solution of Laplace's equation in three dimensions*, in Numerical solution of integral equations.(ed.M.Golberg) Plenum Press, New York, 1990.

[3] K.E.ATKINSON, *Two-grid methods for linear integral equations of the second kind on piecewise smooth surfaces in $R^3$*, Report No.14, University of Iowa, Department of Mathematics, Iowa City, 1991.

[4] K.E.ATKINSON AND I.G.GRAHAM, *An iterative variant of the Nyström method for boundary integral equations on non smooth boundaries*, in J.R.Whiteman (ed.), The mathematics of finite elements and applications VI, MAFELAP 1987, Academic Press, London, 1988, pp.297-303.

[5] K.E.ATKINSON AND I.G.GRAHAM, *Iterative solution of linear systems arising from the boundary integral method*, SIAM J.Sci.Statist.Comput. 13, No.3, pp.694-722.

[6] G.CHANDLER AND I.G.GRAHAM, *High order methods for linear functionals of solutions of second kind integral equations*, SIAM J. Numer. Anal., 25, (1988), pp. 1118-1137.

[7] B.E.J.DAHLBERG AND G.VERCHOTA, *Galerkin methods for the boundary integral equations of elliptic equations in non-smooth domains*, in: Proc. Conf. Boca Raton Fla. 1988, Providence 1990, pp. 39-60.

[8] J.ELSCHNER, *On spline approximation for a class of non-compact integral equations*, Math. Nachr., 146, (1990), pp. 271–321.

[9] J.ELSCHNER, *The double layer potential operator over polyhedral domains II: Spline Galerkin methods*, Math. Meth. Appl. Sci. 45, (1992), pp. 23–37.

[10] W.HACKBUSCH, *Integralgleichungen, Theorie und Numerik*, Teubner Studienbuecher Mathematik, B.G.Teubner Stuttgart, 1989.

[11] W.HACKBUSCH, *Multi-grid methods and applications*, Springer, Berlin, 1985.

[12] F.K.HEBEKER, *On the numerical treatment of viscous flows against bodies with corners and edges by boundary element and multi-grid methods*, Numer.Math., 52, (1988), pp. 81-99.

[13] R.E.KLEINMAN AND W.L.WENDLAND, *On Neumann's method for the Exterior Neumann problem for the Helmholtz equation*, Journal of Math. Anal. and Appl., 1, (1977), pp. 170-202.

[14] A.KIELBASINSKI AND H.SCHWETLICK, *Numerische lineare Algebra*, VEB Dt. Verlag d. Wiss., Berlin, 1988.

[15] J.KRAL, *Integral operators in potential theory*, LNM 823, Springer-Verlag, Berlin Heidelberg New York, 1980.

[16] J.KRAL AND W.L.WENDLAND, *On the applicability of the Fredholm–Radon method in potential theory and the panel method*, in Notes on Numerical Fluid Mechanics, vol. 21, Viehweg, 1988, pp. 120–136.

[17] J.KRAL AND W.WENDLAND, *Some examples concerning applicability of the Fredholm-Radon method in potential theory*, Aplikace matematiky, 31, (1986), pp. 293-308.

[18] R.KRESS, *A Nyström method for boundary integral equations in domains with corners*, Numer. Math., 58, (1990), pp. 145-161.

[19] MASPAR, *Data-Parallel Programming Guide*, MasPar Computer Corporation, Sunnyvale, California.

[20] MASPAR, *Fortran Reference Manual*, Software Version 1.1, August 1991, MasPar Computer Corporation, Sunnyvale, California.

[21] MASPAR, *System Overview*, March 1991, MasPar Computer Corporation, Sunnyvale, California.

[22] V.G.MAZYA, *Boundary integral equations*, in V.G.Mazya and S.M.Nikol'skiĭ (eds.) Encyclopaedia of

Math. Sciences, vol. 27, Analysis IV , Springer-Verlag, Berlin, Heidelberg 1991.

[23] A.RATHSFELD, *Iterative solution of linear systems arising from the Nyström method for the double layer potential equation over curves with corners*, Math. Meth. Appl. Sci. 15, (1992).

[24] A.RATHSFELD, *On quadrature methods for the double layer potential equation over the boundary of a polyhedron*, to appear.

[25] A.RATHSFELD, *On the finite section method for double layer potential equations over the boundary of a polyhedron*, Math.Nachr. 157, (1992), pp. 7-14.

[26] A.RATHSFELD, *The invertibility of the double layer potential operator in the space of continuous functions defined on a polyhedron. The panel method*, Appl. Anal. 45, (1992), pp. 135-177.

[27] S.REMPEL AND G.SCHMIDT, *Eigenvalues for spherical domains with corners via boundary integral equations*, Int. Equ. Op.Theory 14, (1991), pp. 229-250.

[28] Y.SAAD AND M.H.SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. 7, No. 3, (1986), pp. 856-869.

[29] H.SCHIPPERS, *Multi-grid methods for boundary integral equations*, Numer. Math., 46, (1985), pp. 351-363.

[30] H.SCHIPPERS, *Multi-grid methods in boundary element calculations*, in C.A.Brebbia, G.Kuhn, and W.L.Wendland (eds.) Boundary Elements IX, Springer-Verlag, Berlin, Heidelberg, New York, (1987), pp. 475-492.

[31] R.SCHLUNDT, *Iterative Verfahren für lineare Gleichungssysteme mit schwach besetzten Koeffizientenmatrizen*, Preprint, Inst. f. Angew. Anal. u. Stoch., (1993).

[32] H.F.WALKER, *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Stat. Comput. 9, No. 1, (1988), pp. 152-163.

[33] W.L.WENDLAND, *Behandlung von Randwertaufgaben im* $\mathrm{R}^3$ *mit Hilfe von Einfach- und Doppelschichtpotentialen*, Numer. Math., 11, (1968), pp. 380–404.

[34] W.L.WENDLAND, *Boundary methods and their asymptotic convergence*, in P.Filipi (ed.) Lecture Notes of the CISM Summer- School on "Theoretical Acoustics and Numerical Techniques", International Centre for Mechanical Sciences, Udine, Notes Nb.277, Springer-Verlag Wien, New York, (1983), pp. 137-216.

Institut für Angewandte Analysis und Stochastik
Hausvogteiplatz 5-7
Berlin
O-1086
Germany
kleemann@iaas-berlin.dbp.de
rathsfeld@iaas-berlin.dbp.de

| $\alpha$ | $i_0$ | $n$ | $N_n$ | $SER_n$ | $EX_n$ | $ERR$ | $\alpha$ | $i_0$ | $n$ | $N_n$ | $SER_n$ | $EX_n$ | $ERR$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 6 | 1688 | 0.01820 | 2.78 | 0.00116 | 1 | 1 | 4 | 406 | 0.1428 | 0.57 | 0.00104 |
| | | 8 | 3034 | 0.00181 | 8.03 | 0.00066 | | | 6 | 1164 | 0.1099 | 0.64 | 0.00121 |
| | | 10 | 4774 | 0.00595 | -5.34 | 0.00044 | | | 8 | 2316 | 0.0900 | 0.69 | 0.00111 |
| | | 12 | 6902 | 0.00740 | -1.20 | 0.00032 | | | 10 | 3856 | 0.0765 | 0.73 | 0.00100 |
| | | 14 | 9424 | 0.00694 | 0.42 | 0.00024 | | | 12 | 5790 | 0.0666 | 0.76 | 0.00090 |
| | | 16 | 12340 | 0.00596 | 1.15 | 0.00019 | | | 14 | 8118 | 0.0591 | 0.78 | 0.00082 |
| | | 18 | 15644 | 0.00497 | 1.54 | 0.00016 | | | 16 | 10834 | 0.0531 | 0.78 | 0.00075 |
| | | 20 | 19342 | 0.00413 | 1.76 | 0.00014 | | | 18 | 13944 | 0.0482 | 0.78 | 0.00069 |
| | | 22 | 22434 | 0.00345 | 1.87 | 0.00012 | | | 20 | 17448 | 0.0442 | 0.83 | 0.00064 |
| | | 24 | 27914 | 0.00292 | 1.91 | 0.00011 | | | 22 | 21340 | 0.0408 | 0.84 | 0.00059 |
| | | 26 | 32788 | 0.00251 | 1.91 | 0.00010 | | | 24 | 25626 | 0.0379 | 0.85 | 0.00055 |
| | | 28 | 38056 | 0.00218 | 1.89 | 0.00009 | | | 26 | 30306 | 0.0379 | 0.84 | 0.00052 |
| | | 30 | 43712 | 0.00192 | 1.87 | 0.00008 | | | 28 | 35374 | 0.0335 | 0.74 | 0.00049 |
| | | 32 | 49762 | 0.00170 | 1.84 | 0.00007 | | | 30 | 40836 | 0.0318 | 0.75 | 0.00046 |
| $\frac{4}{3}$ | 0 | 3 | 458 | 0.1151 | -0.31 | 0.0044 | $\frac{4}{3}$ | 1 | 6 | 1190 | 0.0794 | 0.91 | 0.00035 |
| | | 4 | 836 | 0.0727 | 1.61 | 0.00283 | | | 8 | 2406 | 0.0593 | 1.01 | 0.00040 |
| | | 5 | 1370 | 0.0450 | 2.15 | 0.00174 | | | 10 | 4070 | 0.0466 | 1.08 | 0.00038 |
| | | 6 | 2030 | 0.0265 | 2.90 | 0.00121 | | | 12 | 6230 | 0.0431 | 0.43 | 0.00035 |
| | | 7 | 2746 | 0.0141 | 4.08 | 0.00090 | | | 14 | 8884 | 0.0405 | 0.39 | 0.00031 |
| | | 8 | 3660 | 0.00801 | 4.25 | 0.00069 | | | 16 | 12006 | 0.0373 | 0.62 | 0.00028 |
| | | 9 | 4634 | 0.00517 | 3.72 | 0.00056 | | | 18 | 15564 | 0.0341 | 0.78 | 0.00025 |
| | | 10 | 5794 | 0.00149 | 11.80 | 0.00046 | | | 20 | 19660 | 0.0310 | 0.87 | 0.00022 |
| | | 11 | 7074 | 0.00120 | 2.29 | 0.00038 | | | 22 | 24228 | 0.0284 | 0.95 | 0.00020 |
| | | 12 | 8442 | 0.00161 | -3.35 | 0.00033 | | | 24 | 29256 | 0.0260 | 0.99 | 0.00018 |
| | | 13 | 10022 | 0.00126 | 3.04 | 0.00028 | | | 26 | 34792 | 0.0230 | 1.57 | 0.00016 |
| | | 14 | 11630 | 0.00126 | -0.04 | 0.00025 | | | 28 | 40788 | 0.0221 | 0.49 | 0.00015 |
| | | 15 | 13392 | 0.00134 | -0.83 | 0.00022 | | | 30 | 47364 | 0.0205 | 1.09 | 0.00014 |
| | | 16 | 15320 | 0.00116 | 2.19 | 0.00020 | | | 32 | 54324 | 0.0191 | 1.11 | 0.00013 |
| | | 17 | 17314 | 0.00103 | 1.84 | 0.00018 | | | | | | | |
| | | 18 | 19466 | 0.00090 | 2.55 | 0.00016 | | | | | | | |
| | | 19 | 21716 | 0.00079 | 2.40 | 0.00015 | | | | | | | |
| | | 20 | 24152 | 0.00069 | 2.56 | 0.00014 | | | | | | | |

TABLE 1

*Method (2.4), $\Omega_2$, Supremum norm error, Error of Dirichlet solution at (0.5,0.5,0.5)*

| $\alpha$ | $i_0$ | $n$ | $N_n$ | $SER_n$ | $EX_n$ | $ERR$ | $\alpha$ | $i_0$ | $n$ | $N_n$ | $SER_n$ | $EX_n$ | $ERR$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0 | 3 | 646 | 0.0755 | -1.80 | 0.00494 | 2 | 1 | 6 | 1384 | 0.0524 | 0.65 | 0.000531 |
| | | 4 | 1310 | 0.0388 | 2.31 | 0.00275 | | | 8 | 2982 | 0.0402 | 0.92 | 0.000291 |
| | | 5 | 2234 | 0.0209 | 2.77 | 0.00186 | | | 10 | 5320 | 0.0300 | 1.32 | 0.000186 |
| | | 6 | 3428 | 0.0189 | 0.55 | 0.00127 | | | 12 | 8522 | 0.0221 | 1.66 | 0.000134 |
| | | 7 | 4902 | 0.0165 | 0.90 | 0.00092 | | | 14 | 12526 | 0.0171 | 1.65 | 0.000104 |
| | | 8 | 6666 | 0.0096 | 4.01 | 0.00073 | | | 16 | 17380 | 0.0135 | 1.77 | 0.000086 |
| | | 9 | 8766 | 0.0077 | 1.93 | 0.00058 | | | 18 | 23158 | 0.0109 | 1.80 | 0.000075 |
| | | | | | | | | | 20 | 29906 | 0.0090 | 1.85 | 0.000066 |
| | | | | | | | | | 22 | 37596 | 0.0075 | 1.86 | 0.000059 |
| | | | | | | | | | 24 | 46276 | 0.0064 | 1.88 | 0.000055 |
| | | | | | | | | | 26 | 55940 | 0.0055 | 1.89 | 0.000051 |
| | | | | | | | | | 28 | 66640 | 0.0048 | 1.91 | 0.000048 |

TABLE 2

*Method (2.4), $\Omega_2$, Supremum norm error, Error of Dirichlet solution at (0.5,0.5,0.5)*

| $\alpha$ | $i_0$ | $n$ | $N_n$ | $N_u$ | $ERR$ | $\alpha$ | $i_0$ | $n$ | $N_n$ | $N_u$ | $ERR$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 2 | 96 | 96 | 0.014 | 2.5 | 1 | 2 | 24 | 96 | 0.00019 |
| | | 3 | 216 | 216 | 0.0063 | | | 3 | 144 | 216 | 0.00099 |
| | | 4 | 384 | 384 | 0.0036 | | | 4 | 408 | 384 | 0.00038 |
| | | 5 | 600 | 600 | 0.0023 | | | | | | |
| 4/3 | 0 | 2 | 120 | 96 | 0.014 | 3.5 | 1 | 2 | 24 | 96 | 0.0039 |
| | | 3 | 264 | 216 | 0.0059 | | | 3 | 192 | 216 | 0.0017 |
| | | 4 | 504 | 384 | 0.0034 | | | | | | |
| 4/3 | 1 | 2 | 24 | 96 | 0.0028 | | | | | | |
| | | 3 | 96 | 216 | 0.0082 | 3.5 | 2 | 3 | 24 | 216 | 0.0099 |
| | | 4 | 240 | 384 | 0.0081 | | | 4 | 144 | 384 | 0.0055 |
| | | 5 | 408 | 600 | 0.0072 | | | | | | |

TABLE 3

*Method (2.4), $\Omega_1$, Error of Dirichlet solution at the midpoint*

| | GE | | | NE | | | TH | | | TA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | 4 | 6 | 8 | 4 | 6 | 8 | 4 | 6 | 8 | 4 | 6 | 8 |
| $N$ | 736 | 1688 | 3034 | 736 | 1688 | 3034 | 736 | 1688 | 3034 | 736 | 1688 | 3034 |
| $N_c$ | | | | | | | 370 | 564 | 758 | 370 | 564 | 758 |
| $NI$ | | | | 27 | 31 | 35 | 3 | 4 | 5 | 4 | 5 | 6 |
| $RA$ | | | | 0.78 | 0.79 | 0.79 | 0.07 | 0.14 | 0.19 | 0.14 | 0.18 | 0.21 |
| $TI$ | 249 | 2914 | 16919 | 32 | 185 | 651 | 81 | 351 | 1147 | 63 | 255 | 726 |

TABLE 4

*CPU-times for several solvers, $\Omega_2$, $\alpha = 1$, No neglect of Str*

| | GM | | | TA' | | | PG | | | PG' | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | 4 | 6 | 8 | 4 | 6 | 8 | 4 | 6 | 8 | 4 | 6 | 8 |
| $N$ | 736 | 1688 | 3034 | 736 | 1688 | 3034 | 736 | 1688 | 3034 | 736 | 1688 | 3034 |
| $N_c$ | | | | 370 | 564 | 758 | 370 | 564 | 758 | 370 | 564 | 758 |
| $NI$ | 9 | 10 | 11 | 4 | 5 | 6 | 5 | 6 | 7 | 5 | 6 | 7 |
| $RA$ | 0.37 | 0.38 | 0.38 | 0.14 | 0.18 | 0.19 | 0.09 | 0.12 | 0.13 | 0.09 | 0.15 | 0.21 |
| $TI$ | 25 | 141 | 470 | 38 | 162 | 498 | 67 | 274 | 788 | 55 | 213 | 623 |

TABLE 5

*CPU-times for several solvers, $\Omega_2$, $\alpha = 1$, No neglect of Str*

| | GE | | | GM | | | TH | | | TA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | 5 | 7 | 9 | 5 | 7 | 9 | 5 | 7 | 9 | 5 | 7 | 9 |
| $N$ | 720 | 1344 | 2160 | 720 | 1344 | 2160 | 720 | 1344 | 2160 | 720 | 1344 | 2160 |
| $N_c$ | | | | | | | 696 | 1104 | 1488 | 696 | 1104 | 1488 |
| $NI$ | | | | 60 | 75 | 94 | 4 | 5 | 5 | 3 | 3 | 5 |
| $RA$ | | | | 0.99 | 0.98 | 0.95 | 0.22 | 0.35 | 0.23 | 0.17 | 0.23 | 0.25 |
| $TI$ | 224 | 1521 | 6286 | 111 | 440 | 1271 | 282 | 1072 | 2617 | 245 | 927 | 2285 |

TABLE 6

*CPU-times for several solvers, $\Omega_3$, $\alpha = 1$, No neglect of Str*

| | TA' | | | PG | | | PG' | | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | 5 | 7 | 9 | 5 | 7 | 9 | 5 | 7 | 9 |
| $N$ | 720 | 1344 | 2160 | 720 | 1344 | 2160 | 720 | 1344 | 2160 |
| $N_c$ | 696 | 1104 | 1488 | 696 | 1104 | 1488 | 696 | 1104 | 1488 |
| $NI$ | 3 | 3 | 5 | 3 | 3 | 5 | 3 | 3 | 5 |
| $RA$ | 0.17 | 0.23 | 0.25 | 0.12 | 0.13 | 0.23 | 0.12 | 0.13 | 0.23 |
| $TI$ | 330 | 984 | 2303 | 249 | 941 | 2319 | 389 | 1325 | 3556 |

TABLE 7

*CPU-times for several solvers, $\Omega_3$, $\alpha = 1$, No neglect of Str*

| | | JA | | | NE | | | GM | | | TA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $N_n$ | $NI$ | $RA$ | $TI$ | $NI$ | $RA$ | $TI$ | $NI$ | $RA$ | $TI$ | $NI$ | $RA$ | $TI$ |
| 2 | 24 | 3 | 0.22 | | 4 | 0.54 | | 2 | | | 1 | | |
| 4 | 240 | 8 | 0.55 | 2 | 4 | 0.35 | 2 | 3 | 0.07 | 2 | 2 | 0.02 | 2 |
| 8 | 1368 | 20 | 0.71 | 116 | 6 | 0.54 | 68 | 5 | 0.10 | 72 | 4 | 0.16 | 63 |
| 11 | 2928 | 34 | 0.78 | 772 | 9 | 0.56 | 368 | 6 | 0.11 | 353 | 5 | 0.22 | 318 |

TABLE 8

*CPU-times for JA, GE, GM and TA, $\Omega_1$, $\alpha = 4/3$, Neglect of Str*

| $C$ | $A$ | $B$ | overall time in seconds |
|---|---|---|---|
| ROC | ROC | ROC | 1,160 |
| ROC | ROC | COC | 660 |
| ROC | COC | COC | 475 |
| 11.2% | 65.9% | 4.7% | |

TABLE 9

*Method (2.8)-(2.11), THR, $\Omega_2$, Time using different combinations of codes for the generation of A, B, and C and the Nyström interpolation, percentage of the time for different codes in case of the most efficient combination.*

| $n$ | $N_n$ | $\eta - 1$ | $N_c$ | $ERR$ | $RA$ | $T1$ (method TH) | $T2$ (method THG) |
|---|---|---|---|---|---|---|---|
| 2 | 40 | 1,000 | 40 | 0.0030 | - | 0 | 12 |
| 3 | 180 | 1,000 | 40 | 0.0015 | 0.012 | 1 | 20 |
| 4 | 444 | 1,000 | 40 | 0.0023 | 0.043 | 5 | 33 |
| 5 | 818 | 1,000 | 40 | 0.0023 | 0.071 | 15 | 52 |
| 6 | 1,368 | 1,000 | 40 | 0.0022 | 0.106 | 49 | 80 |
| 7 | 2,014 | 1,000 | 40 | 0.0020 | 0.133 | 172 | 114 |
| 8 | 2,786 | 1,000 | 40 | 0.0018 | 0.160 | 6,649 | 155 |
| 16 | 13,966 | 1,000 | 40 | 0.0009 | - | - | 1,040 |
| 32 | 63,146 | 69 | 124 | 0.0004 | 0.300 | - | 5,394 |
| | | | | | | | $T3$ (method THR) |
| 40 | 101,392 | 98 | 124 | 0.0003 | 0.330 | - | 13,000 |

TABLE 10

*CPU-times for method TH on a Convex 210 and THG on a MP-1216, domain $\Omega_2$.*

| $n$ | $N_n$ | $\eta - 1$ | $N_c$ | $ERR$ | $RA$ | $T1$ | $T2$ | $T3$ (method THR) |
|---|---|---|---|---|---|---|---|---|
| 2 | 48 | 1,000 | 48 | 0.0149 | - | 0 | 45 | 98 |
| 3 | 212 | 1,000 | 48 | 0.0076 | 0.016 | 1 | 54 | 117 |
| 4 | 520 | 1,000 | 48 | 0.0045 | 0.038 | 7 | 70 | 141 |
| 5 | 964 | 1,000 | 48 | 0.0030 | 0.059 | 24 | 91 | 205 |
| 6 | 1,604 | 1,000 | 48 | 0.0022 | 0.073 | 66 | 120 | 221 |
| 7 | 2,364 | 1,000 | 48 | 0.0017 | 0.088 | 200 | 154 | 254 |
| 8 | 3,274 | 1,000 | 48 | 0.0014 | 0.114 | 1,016 | 196 | 312 |
| 16 | 16,394 | 12 | 344 | 0.00045 | 0.086 | - | - | 1,135 |
| 24 | 39,972 | 20 | 348 | 0.00024 | - | - | 2,960 | - |
| 32 | 74,110 | 25 | 436 | 0.00016 | 0.153 | - | - | 7,400 |

TABLE 11

*CPU-times for method TH on a Convex 210 and THG and THR on a MP-1216 for the same example as in the Table 9 but over $\Omega_4$.*

# Veröffentlichungen des Instituts für Angewandte Analysis und Stochastik

## Preprints 1992

1. D.A. Dawson and J. Gärtner: Multilevel large deviations.

2. H. Gajewski: On uniqueness of solutions to the drift–diffusion–model of semiconductor devices.

3. J. Fuhrmann: On the convergence of algebraically defined multigrid methods.

4. A. Bovier and J.-M. Ghez: Spectral properties of one–dimensional Schrödinger operators with potentials generated by substitutions.

5. D.A. Dawson and K. Fleischmann: A super–Brownian motion with a single point catalyst.

6. A. Bovier, V. Gayrard: The thermodynamics of the Curie–Weiss model with random couplings.

7. W. Dahmen, S. Prößdorf, R. Schneider: Wavelet approximation methods for pseudodifferential equations I: stability and convergence.

8. A. Rathsfeld: Piecewise polynomial collocation for the double layer potential equation over polyhedral boundaries. Part I: The wedge, Part II: The cube.

9. G. Schmidt: Boundary element discretization of Poincaré–Steklov operators.

10. K. Fleischmann, F. I. Kaj: Large deviation probability for some rescaled superprocesses.

11. P. Mathé: Random approximation of finite sums.

12. C.J. van Duijn, P. Knabner: Flow and reactive transport in porous media induced by well injection: similarity solution.

13. G.B. Di Masi, E. Platen, W.J. Runggaldier: Hedging of options under discrete observation on assets with stochastic volatility.

14. J. Schmeling, R. Siegmund–Schultze: The singularity spectrum of self-affine fractals with a Bernoulli measure.

15. A. Koshelev: About some coercive inequalities for elementary elliptic and parabolic operators.

16. P.E. Kloeden, E. Platen, H. Schurz: Higher order approximate Markov chain filters.

**17.** H.M. Dietz, Y. Kutoyants: A minimum–distance estimator for diffusion processes with ergodic properties.

**18.** I. Schmelzer: Quantization and measurability in gauge theory and gravity.

**19.** A. Bovier, V. Gayrard: Rigorous results on the thermodynamics of the dilute Hopfield model.

**20.** K. Gröger: Free energy estimates and asymptotic behaviour of reaction–diffusion processes.

**21.** E. Platen (ed.): Proceedings of the $1^{st}$ workshop on stochastic numerics.

**22.** S. Prößdorf (ed.): International Symposium "Operator Equations and Numerical Analysis" September 28 – October 2, 1992 Gosen (nearby Berlin).

**23.** K. Fleischmann, A. Greven: Diffusive clustering in an infinite system of hierarchically interacting diffusions.

**24.** P. Knabner, I. Kögel–Knabner, K.U. Totsche: The modeling of reactive solute transport with sorption to mobile and immobile sorbents.

**25.** S. Seifarth: The discrete spectrum of the Dirac operators on certain symmetric spaces.

**26.** J. Schmeling: Hölder continuity of the holonomy maps for hyperbolic basic sets II.

**27.** P. Mathé: On optimal random nets.

**28.** W. Wagner: Stochastic systems of particles with weights and approximation of the Boltzmann equation. The Markov process in the spatially homogeneous case.

**29.** A. Glitzky, K. Gröger, R. Hünlich: Existence and uniqueness results for equations modelling transport of dopants in semiconductors.

**30.** J. Elschner: The $h$-$p$–version of spline approximation methods for Mellin convolution equations.

**31.** R. Schlundt: Iterative Verfahren für lineare Gleichungssysteme mit schwach besetzten Koeffizientenmatrizen.

**32.** G. Hebermehl: Zur direkten Lösung linearer Gleichungssysteme auf Shared und Distributed Memory Systemen.

**33.** G.N. Milstein, E. Platen, H. Schurz: Balanced implicit methods for stiff stochastic systems: An introduction and numerical experiments.

**34.** M.H. Neumann: Pointwise confidence intervals in nonparametric regression with heteroscedastic error structure.

**35.** M. Nussbaum: Asymptotic equivalence of density estimation and white noise.