JOHANNES KEPLER UNIVERSITÄT LINZ

Comparison of Parallel Solvers for Nonlinear Elliptic Problems Based on Domain Decomposition Ideas

Bodo Heise
Department of Mathematics
Johannes Kepler University, A–4040 Linz, Austria

Michael Jung
Faculty for Mathematics
Technical University Chemnitz, D–09107 Chemnitz, Germany

Institutsbericht Nr. 494 August 1995

INSTITUT FÜR MATHEMATIK

A–4040 LINZ, ALTENBERGERSTRASSE 69, AUSTRIA

Comparison of Parallel Solvers for Nonlinear Elliptic Problems Based on Domain Decomposition Ideas

Bodo Heise* and Michael Jung†

Abstract

In the present paper, the solution of nonlinear elliptic boundary value problems (b.v.p.) on parallel machines with Multiple Instruction Multiple Data (MIMD) architecture is discussed. Especially, we consider electro–magnetic field problems the numerical solution of which is based on finite element discretizations and a nested Newton solver. For solving the linear systems of algebraic finite element equations in each Newton step, parallel conjugate gradient methods with a Domain Decomposition preconditioner (DD PCG) as well as parallelized global multigrid methods are applied. The implementation of the whole algorithm, i.e. the mesh generation, the generation of the finite element equations, the nested Newton algorithm, the DD PCG method and the global multigrid method, is based on a non–overlapping DD data structure.

The efficiency of the parallel DD PCG methods and the parallelized global multigrid methods, which are embedded in the nested Newton solver, are compared. Furthermore, the performance of the parallel nested Newton solver on different machines (GC Power Plus, Multicluster with transputers T805, and workstation cluster) is demonstrated by numerical results.

AMS subject classifications: 65N55, 65N22, 65N30, 78A30

Key words: Nonlinear partial differential equations; Parallel computing; Multigrid methods; Domain decomposition; Finite element methods; Magnetic field calculations

1 Introduction

Recently, numerical algorithms for Multiple Instruction Multiple Data (MIMD) parallel computers with message–passing principle have found growing interest. In particular, Domain Decomposition (DD) methods have become a standard tool for the numerical solution of boundary value problems (b.v.p.) for partial differential equations (cf. the proceedings of the international Symposia on "Domain Decomposition methods for partial differential equations" since 1987 [10, 5, 6, 11, 37, 45, 38]). The first aim of these methods is data partitioning. The more important second aim is the construction of possibly new highly parallel solvers. From the point of view of the solver, we distinguish so–called local methods from the global ones.

The local DD methods involve the solution of local subproblems in subdomains. As an example, we consider the non–overlapping DD methods, the main ideas of which are presented, e.g. in [3, 7, 8, 48, 49, 21, 22, 23, 39]. These methods have been applied successfully to the numerical solution of two–dimensional linear and nonlinear b.v.p.'s discretized by finite elements (FE) [19, 20, 30, 31] as well as to the a coupling of finite and boundary elements [33, 34, 14]. The inherent parallel data distribution allows parallel computation with a communication cost one order lower than the FE problem itself. Thus, computing on many processors can be performed with high efficiency [31, 33]. Obviously, it is advantageous to employ multigrid methods for the subproblems.

Another approach is to parallelize a well–known, e.g., iterative solver. Then, all properties of the method, in particular, convergence properties, are preserved. On the other hand, the efficiency

*Institute of Mathematics, Johannes Kepler University Linz, Altenberger Str. 69, A - 4040 Linz, Austria. e-mail: heise@numa.uni-linz.ac.at. This research has been supported by the German Research Foundation DFG within the Priority Research Programme "Boundary Element Methods" under the grant La 767/1-3.

†Faculty for Mathematics, Technical University Chemnitz, D - 09107 Chemnitz, Germany. e-mail: michael.jung@mathematik.tu-chemnitz.de

of the algorithm is determined by the communication overhead only. As an example, we consider the parallelization of global multigrid methods [41, 46, 42, 1, 2] where the 'quality' of the parallel algorithm depends heavily on how far the communication can be reduced.

The main idea of the approach of [35] is to apply the parallel DD data distribution of the local DD methods in a global multigrid method. It allows us to perform the multigrid interpolation and restriction operators without any communication. Thus, communication is needed in the smoother and the coarse–grid solver only. The ideas presented in [35] enable us to implement the Gauss–Seidel smoother with the same communication effort as it is required by the Jacobi smoother, but with better smoothing properties. Further, a Schur complement solver with preconditioning, originating from the local DD methods, is employed as coarse–grid solver.

The aim of the present paper is to compare local DD methods with global multigrid methods and to apply both of them to nonlinear b.v.p.'s using a parallel nested Newton approach. We will demonstrate that, if the global multigrid methods benefit from DD ideas, both solvers require a numerical effort of the same order of magnitude.

Although, the algorithms under consideration have been designed especially for MIMD multiprocessor systems, there is also interest in their implementation on workstation clusters. Therefore, we present results for a workstation cluster, too.

The rest of the paper is organized as follows. In Section 2, we formulate the nonlinear boundary value problem. In Section 3, we present the parallel nested Newton method and refer to theoretical results. Section 4 is devoted to the different solvers for the arising linear problems. We summarize results for the DD–PCG solver and describe some details of the parallelization of the global multigrid method. In Section 5, we present performance results for a model problem and a more complicated practical electro–magnetic field problem. Finally, we add some concluding remarks in Section 6.

2 Nonlinear electromagnetic field problems

Let us consider a two–dimensional nonlinear stationary magnetic field problem in a bounded domain $\Omega \subset \mathbf{R}^2$. The variational formulation of the problem can be written as follows:
Find $u \in V = H_0^1(\Omega)$ such that

$$a(u,v) = \langle f, v\rangle \quad \forall v \in V, \tag{1}$$

where

$$a(u,v) = \int_\Omega \nu(x, |\nabla u|)\ \nabla^\top u\ \nabla v\ dx\ ,$$

and

$$\langle f, v\rangle = \int_\Omega (\mathrm{S}v - \mathrm{H_{0y}}\frac{\partial v}{\partial \mathrm{x}} + \mathrm{H_{0x}}\frac{\partial v}{\partial \mathrm{y}})\ dx.$$

Here, the physical model has been developed from Maxwell's equations, cf. [28, 29] for details. We assume that Ω representing the cross–section of some electromagnetic device lies in the x–y–plane of the $\mathbf{R}^3$. Then the solution u is the z–component of the vector potential $\vec{\mathrm{A}}$. The z–component of the current density is represented by S, and the vector $\vec{\mathrm{H}}_0 = (\mathrm{H_{0x}}, \mathrm{H_{0y}}, 0)^\top$ describes the magnetization of permanent magnets. The nonlinearity of the problem is caused by the dependence of ν on the absolute value of the magnetic induction, $B = |\mathrm{rot}\vec{\mathrm{A}}| = |\nabla u|$.

We assume that $\bar{\Omega}$ consists of subdomains

$$\bar{\Omega} = \bigcup_{j=1}^{N_M} \bar{\hat{\Omega}}_j, \quad \text{with} \quad \hat{\Omega}_i \cap \hat{\Omega}_k = \emptyset \quad \forall i \neq k.$$

The $\hat{\Omega}_j$'s represent materials with different magnetic properties (iron, copper, air, permanentmagnetic materials) in the cross–section of an electromagnetic device. We suppose that the function ν depends on the locus $x \in \Omega$, but ν is always the same function of B in one of the $\hat{\Omega}_j$'s, i.e.

$$\nu(x,B) = \nu^{(j)}(B) \quad \text{if} \quad x \in \hat{\Omega}_j.$$

The function $\nu^{(j)}(B)$ is a constant, $\nu^{(j)}(B) \equiv \nu_1^{(j)}$, if the material in $\hat{\Omega}_j$ is not ferromagnetic (e.g., copper, air, vacuum). We formulate some properties of the functions $\nu^{(j)}$, $j = 1, \ldots, N_M$, which are justified by the physical model (cf. [26, 28, 27, 29]):

(A0) $\nu^{(j)}(z) = \nu_1^{(j)} = \text{ const.} \quad \forall z \in [0, z_1^{(j)}]$,

(A1) $\nu^{(j)}(z) \geq \nu_1^{(j)} \quad \forall z \geq 0$,

(A2) $\nu^{(j)}(z)$ is a monotonic increasing function,

(A3) $\lim_{z\to\infty} \nu^{(j)}(z) = \nu_\infty^{(j)}$, where $\nu_\infty^{(j)} = (\mu_0)^{-1}$ for ferromagnetic materials,

(A4) there exists $\nu^{(j)\prime}(z) \quad \forall\ z\ \geq\ 0$, and there exists a constant $M_1^{(j)}$ with $\nu^{(j)\prime}(z) \leq M_1^{(j)} \quad \forall z \geq 0$,

(A5) there exists a constant $M^{(j)}$ with $\nu^{(j)\prime}(z)\ z + \nu^{(j)}(z) \leq M^{(j)} \quad \forall z \geq 0$.

Let ν_1, M_1 and M be the global constants with $\nu_1 = \min_{j=1,\dots,N_M} \nu_1^{(j)}$, $M_1 = \max_{j=1,\dots,N_M} M_1^{(j)}$, and $M = \max_{j=1,\dots,N_M} M^{(j)}$. The following theorem has been proved in [29]:

THEOREM 1 *Let (A1), (A2), (A4), (A5) be fulfilled. Then the variational problem (1) has a unique solution $u \in V$.*

Proof. See [29]. ■

Standard finite element discretization with linear triangular elements has been discussed in [29]. Further, an algorithm for the interpolation of a pointwise given material function $\nu^{(k)}(.)$ is described therein. This cubic C^1-spline interpolation preserves the properties (A0), (A1), (A2), (A3), (A4), (A5) with slight modifications of the constants M and M_1. Thus, existence and uniqueness of the solution of the fully discrete problem have also been proved [29].

3 The parallel nested Newton method

3.1 Description of the algorithm

We assume that, as in the finite element substructuring technique, Ω is decomposed into p non-overlapping subdomains $\Omega_i, i = 1, \dots, p$, such that (cf. [17, 22, 23, 30, 33, 31, 34])

$$\bar{\Omega} = \bigcup_{i=1}^{p} \bar{\Omega}_i, \quad \Omega_i \cap \Omega_j = \emptyset \quad \text{for} \quad i \neq j, \tag{2}$$

i.e., the subdomains $\hat{\Omega}_k$ determined by the materials may be decomposed further. The latter decomposition can be achieved by applying the automatic, adaptive domain decomposition procedure described in [14, 12]. Thus, we have the possibility to provide a quasi-static load balance. The subdomains $\bar{\Omega}_i, i = 1, \dots, p$, are assigned to p processors of the MIMD computer. We further assume that in each subdomain $\bar{\Omega}_i$ there is a multilevel sequence of linear finite element discretizations such that this discretization process results in conform triangulations of Ω creating a nested sequence

$$V_1 \subset V_2 \subset \cdots \subset V_l \subset V = H_0^1(\Omega) \tag{3}$$

of spaces of linear finite elements, as it will be described in Subsection 4.2.

We obtain a sequence of variational problems for $q = 1, \dots, l$:
Find $u_q \in V_q \subset V$ such that

$$a_q(u_q, v_q) = \langle f_q, v_q \rangle \quad \forall v_q \in V_q. \tag{4}$$

and a sequence of equivalent nonlinear finite element equations

$$K_q \mathbf{u}_q = \mathbf{f}_q \quad q = 1, \dots, l, \tag{5}$$

with nonlinear operators $K_q : \mathbf{R}^{N_q} \longrightarrow \mathbf{R}^{N_q}$, solution vectors $\mathbf{u}_q \in \mathbf{R}^{N_q}$ and vectors $\mathbf{f}_q \in \mathbf{R}^{N_q}$.

An parallel algorithm for solving (5) with $q = l$, connecting a Newton-like method with the nested iteration and a suitable parallel solver, will be presented in the following. It requires the Fréchet derivative of K_q at a vector $\mathbf{v}_q$,

$$K_q'[\mathbf{v}_q] : \mathbf{R}^{N_q} \to \mathbf{R}^{N_q} \quad q = 1, \dots, l,$$

which is a linear operator and can be represented by the Jacobi matrix, cf. [27, 32, 30, 33, 31]. The result will be the approximate solution $\mathbf{u}_l^\star$ with relative accuracy ε (nested iteration ε).

The Parallel Nested Newton (PNN) algorithm described below includes both, the Newton method (NM) and the modified Newton method (MNM) which is sometimes recommended to save matrix generation time. Further, it includes the two parallel solvers for the linearized problems, namely the DD-preconditioned conjugate gradient method (PCG) and the parallelized global multigrid method (GMG), which are to be compared.

Algorithm PNN (Parallel Nested Newton)
Step 0
Initialization of the grid number:

(0.)(P) $q := 1$.

Step 1
Set the initial solution for grid q:

(1.1)(P) IF $q = 1$ THEN $\mathbf{u}_q^0 = \mathbf{0}$;

(1.2)(P) IF $q > 1$ THEN $\mathbf{u}_q^0 = \tilde{I}_{q-1}^q \mathbf{u}_{q-1}^\star$;
the initial solution is the interpolation of the best solution on grid $q-1$.

(1.3)(P) Initialize the Newton iteration number $j := 0$.

Step 2
Compute the initial Jacobi matrix and the defect vector

(2.)(P) $J_q^0 = K_q'[\mathbf{u}_q^0]$; $\quad \mathbf{d}_q^1 = \mathbf{f}_q - K_q \mathbf{u}_q^0$.

Step 3

(3.)(P) Choose a relaxation parameter τ_q^j with $0 < \tau_q^j \le 1$ and a relative accuracy parameter ε_{lin} with $0 < \varepsilon_{\text{lin}} < 1$.

Step 4

(4.)(S) Solve the linear defect system

$$J_q^j \mathbf{w}_q^{j+1} = \mathbf{d}_q^{j+1} \tag{6}$$

approximately (with relative accuracy ε_{lin})

(PCG) using a parallelized preconditioned conjugate gradient (PCG) solver where the preconditioning is realized via a Domain Decomposition (DD) method (local method),

or

(GMG) performing k iterations of a parallelized global multigrid method.

The result is $\tilde{\mathbf{w}}_q^{j+1}$.

Step 5
Correct the solution:

(5.)(P) $\mathbf{u}_q^{j+1} = \mathbf{u}_q^j + \tau_q^j \tilde{\mathbf{w}}_q^{j+1}$.

Step 6
Control the convergence (parameter c_τ is chosen a priori with $c_\tau < 1$):

(6.1)(P) Compute the new defect vector
$\mathbf{d}_q^{j+2} = \mathbf{f}_q - K_q \mathbf{u}_q^{j+1}$;

(6.2)(P) determine the matrix for the linearized problem which is to be solved next

(NM) i.e., compute the new Jacobi matrix (in the Newton method),
$J_q^{j+1} = K_q'[\mathbf{u}_q^{j+1}]$;

or

(MNM) use the matrix again (in the modified Newton method),
$J_q^{j+1} = J_q^j$.

(6.3)(C) Compute defect norms
$d_q^{j+1} = \|\mathbf{d}_q^{j+1}\|; \quad d_q^{j+2} = \|\mathbf{d}_q^{j+2}\|;$

(6.4)(P) IF $d_q^{j+2} \geq d_q^{j+1}$ THEN $\left(\begin{array}{l} \tau_q^j := \min\left\{ c_\tau \tau_q^j, \dfrac{\tau_q^j d_q^{j+1}}{d_q^{j+1} + d_q^{j+2}} \right\}; \\ \texttt{GOTO } \textit{Step 5} \end{array} \right);$

(6.5)(P) IF $d_q^{j+2} \leq \varepsilon d_q^1$ THEN $\left(\begin{array}{l} \mathbf{u}_q^\star := \mathbf{u}_q^{j+1}; \\ \texttt{IF } q < l \texttt{ THEN } (q := q+1; \texttt{ GOTO } \textit{Step 1}); \\ \texttt{IF } q = l \texttt{ THEN EXIT}; \end{array} \right);$

(6.6)(P) Perform a further Newton step:
$j := j + 1;$
GOTO *Step 3*.

In this description, (P) indicates that the step is performed completely in parallel, i.e., independently on the processors, provided that a DD data structure as described in Subsection 4.2 is applied. The solver (S) includes parallel independent parts, communication between processors handling neighbouring subdomains, and global communication. Note, (C) indicates that global communication is necessary. Obviously, the only additional communication (compared with solving a linear problem) is the computation of global defect norms.

3.2 Convergence results

In [27], convergence results have been proved for a damped, modified multigrid-Newton method. These results apply for the choice of (MNM) and (GMG) in the algorithm PNN given above, too. We will quote the main results here and refer to [27] for the details.

Assuming convergence and certain properties (e.g. symmetric smoothing, construction of coarse-grid operators by the Galerkin approach) which are fulfilled by many standard multigrid algorithms, see [27, Lemma 3] or [36], the multigrid method can be used as a preconditioner as it has been proved in [36, 40]. The spectral equivalence constants depend on the multigrid convergence factor η with respect to some vector norm and the number k of multigrid iterations only [36, 40].

As a consequence of the preconditioning properties of the multigrid method, the linear convergence of the damped, modified multigrid-Newton method on a fixed grid q is proved in [27, Thm. 2]. The convergence factor as well as the necessary damping parameters τ_q^j depend on η, k, and the material constants ν_1 and M only. Thus, the Parallel Nested Newton method with a damped, modified multigrid-Newton method converges globally [27, Coroll. 3]. The number of Newton iterations can be bounded a priori for all grids $q = 1, \ldots, l$. The convergence can be proved even for one multigrid cycle per Newton iteration ($k = 1$).

4 Parallelization strategy and data structure

4.1 Mesh generation

Let us consider the decomposition (2) of the domain Ω into p non–overlapping subdomains Ω_i. Since we want to use multigrid methods we have to generate a sequence of nested triangular meshes $\mathcal{T}_q$, $q = 1, 2, \ldots, l$, which is creating the sequence (3) of finite element spaces. The starting point for the construction of the coarsest mesh $\mathcal{T}_1$ is a decomposition of the coupling boundary $\Gamma_C = \bigcup_{i=1}^p \partial\Omega_i$ into so–called basic lines $\Gamma_{C,k}$ $(k = 1, 2, \ldots, k_C)$ with $\Gamma_C = \bigcup_{k=1}^{k_C} \bar{\Gamma}_{C,k}$, $\Gamma_{C,k} \cap \Gamma_{C,k'} = \emptyset$ for $k \neq k'$. We suppose that a basic line is either part of the intersection of a subdomain boundary with the boundary of the domain Ω, $\Gamma_{C,k} \subset \partial\Omega_i \cap \partial\Omega$, or part of the intersection of the boundaries of two neighbouring subdomains, $\Gamma_{C,k} \subset \partial\Omega_i \cap \partial\Omega_j$ $(i \neq j)$. On the basis of the decomposition of the coupling boundary, the code PARMESH [9] (cf. also [13]) generates in parallel finite element triangulations of the subdomains Ω_i, which result in an admissible triangulation of the whole domain Ω. In the present version of this mesh generator it is assumed that the basic lines are straight lines, arcs of a circle or parabolas. The finer triangulations $\mathcal{T}_q$, $q = 2, 3, \ldots, l$, are obtained by a successive refinement process, i.e. all triangles of the triangulation $\mathcal{T}_{q-1}$ are divided into four smaller subtriangles. Obviously, this refinement process can be performed in parallel.

In each triangulation the nodes are classified into three groups: the *cross-points* (vertices), i.e. the starting and end points of the basic lines, the *edge coupling nodes*, i.e. the nodes which are generated on the basic lines, and the *inner nodes*.

For the numbering of the nodes in each triangulation we use the following order: cross–points, edge coupling nodes on $\Gamma_{C,1}$, edge coupling nodes on $\Gamma_{C,2}$, ... , edge coupling nodes on Γ_{C,k_C}, inner nodes of Ω_1, inner nodes of Ω_2, ... , inner nodes of Ω_p.

4.2 Domain Decomposition data structure

The numbering of the nodes described in Subsection 4.1 induces the following block structure of the systems (6) of algebraic finite element equations

$$J\mathbf{w} = \begin{pmatrix} J_V & J_{VE} & J_{VI} \\ J_{EV} & J_E & J_{EI} \\ J_{IV} & J_{IE} & J_I \end{pmatrix} \begin{pmatrix} \mathbf{w}_V \\ \mathbf{w}_E \\ \mathbf{w}_I \end{pmatrix} = \begin{pmatrix} \mathbf{d}_V \\ \mathbf{d}_E \\ \mathbf{d}_I \end{pmatrix} . \tag{7}$$

Here, the indices "V", "E", and "I" correspond to the cross–points (vertices), the edge coupling nodes, and the inner nodes, respectively. For the sake of simplicity we have omitted the indices q and j.

For the description of the DD preconditioners (see Subsection 4.3) we use the block structure

$$J\mathbf{w} = \begin{pmatrix} J_C & J_{CI} \\ J_{IC} & J_I \end{pmatrix} \begin{pmatrix} \mathbf{w}_C \\ \mathbf{w}_I \end{pmatrix} = \begin{pmatrix} \mathbf{d}_C \\ \mathbf{d}_I \end{pmatrix} , \tag{8}$$

where the index "C" stands for the coupling nodes, i.e. for the cross–points and the edge coupling nodes, cf. [17, 22, 23, 31, 34]. Therefore, the matrices J_C and $J_{CI} = J_{IC}^T$ are defined by

$$J_C = \begin{pmatrix} J_V & J_{VE} \\ J_{EV} & J_E \end{pmatrix} \qquad \text{and} \qquad J_{CI} = \begin{pmatrix} J_{VI} \\ J_{EI} \end{pmatrix} .$$

The Jacobi matrices J and the right–hand sides $\mathbf{d}$ can be represented as sums of super–element (subdomain) Jacobi matrices J_i and super–element right–hand sides $\mathbf{d}_i$. With the $(N \times N_i)$ boolean matrices A_i mapping some overall vector $\mathbf{v} \in \mathbf{R}^N$ of nodal variables into the super–element vector $\mathbf{v}_i \in \mathbf{R}^{N_i}$ of variables associated with the subdomain $\bar{\Omega}_i$ only, we get

$$J = \sum_{i=1}^{p} A_i^T J_i \, A_i \qquad \text{and} \qquad \mathbf{d} = \sum_{i=1}^{p} A_i^T \mathbf{d}_i \ . \tag{9}$$

In each processor P_i only the corresponding super–element Jacobi matrix J_i and the super–element defect $\mathbf{d}_i$ are stored. A consequence of this storage is the following: If we want to know the values of the elements of the matrix J_V and the defect $\mathbf{d}_V$ or of the matrix J_E and the defect $\mathbf{d}_E$ we have to perform a summation over cross-points or over edge coupling nodes, respectively. Since each inner node belongs to one processor only, the processor P_i has the full information about the elements of the matrix $J_{I,i}$ and the defect $\mathbf{d}_{I,i}$. Resulting from the numbering strategy and the proposed storage of the matrices, J_I is a blockdiagonal matrix.

For the implementation of parallel solvers it is convenient to introduce two types of distribution of vectors to the processors P_i (see, e.g., [21, 24]). We say that the vector $\mathbf{v}$ is of *overlapping type*, if $\mathbf{v}$ is stored in processor P_i as $\mathbf{v}_i = A_i\mathbf{v}$. A vector $\mathbf{d}$ of *adding type* is stored in processor P_i as $\mathbf{d}_i = \sum_{i=1}^{p} A_i^T \mathbf{d}_i$. For example, in the linear solvers which we describe in the Subsections 4.3 and 4.4 the defect vectors are of adding type and the solution vectors are of overlapping type.

4.3 Parallel CG solver with DD preconditioning

The parallel CG algorithm with DD preconditioning for solving the systems (8) can be implemented in a standard way, cf. [22, 34]. It runs completely in parallel with the exception of the two scalar products, and the preconditioning. The DD preconditioner for J, i.e. the matrix C with

$$C = \begin{pmatrix} I_C & J_{CI}B_I^{-T} \\ O & I_I \end{pmatrix} \begin{pmatrix} C_C & O \\ O & C_I \end{pmatrix} \begin{pmatrix} I_C & O \\ B_I^{-1}J_{IC} & I_I \end{pmatrix} \tag{10}$$

contains three components, i.e., the preconditioners C_C and $C_I = \mathrm{diag}\,(C_{I,i})_{i=1,2,\ldots,p}$, and the regular basis transformation matrix $B_I = \mathrm{diag}\,(B_{I,i})_{i=1,2,\ldots,p}$, which can be adapted to the matrix J_q^j in a suitable way [22].

Here, we choose a multigrid V-cycle with one pre- and one postsmoothing step of Gauss-Seidel type in the symmetric Multiplicative Schwarz Method [18, 22] for C_I, and B_I is implicitly defined by hierarchical extension (formally $E_{IC} = -B_I^{-1}J_{IC}$) [25]. We apply a Schur complement preconditioner C_C following Bramble/Pasciak/Schatz (BPS)[3], which uses the idea of Dryja [7] on the coupling boundaries, and a global crosspoint system.

Spectral equivalence between J_q^j and C has been proved in [21, 22]. Together with the results of [3, 18, 25, 27] we can prove that the numerical effort spent for one Newton step on grid q is at most of order $\mathcal{O}(N_q \ln h_q^{-1} \ln\ln h_q^{-1} \ln\varepsilon_{\mathrm{lin}})$, i.e. almost optimal. Here h_q denotes the discretization parameter, such that $N_q = \mathcal{O}(h_q^{-2})$. We refer to [34, 31] for details.

4.4 Parallelized global multigrid method

In this Subsection, we will give a short description of an implementation of multigrid algorithms on parallel machines with MIMD architecture. The basis of the parallelization strategy is the non–overlapping DD data structure discussed in Subsection 4.2. We present the smoothing procedures being used, interpolation and restriction procedures, and some methods for solving the systems of algebraic finite element equations on the coarsest mesh. Especially, the parallelization of these procedures is analyzed. A detailed discussion of the parallel multigrid algorithm can be found in [35].

4.4.1 Smoothers of Gauss–Seidel type

In order to minimize communication, we construct Gauss–Seidel type smoothers which require the same communication as a Jacobi smoother in each iteration step, but with better smoothing properties. The smoothers are based on the block structure (7), one iteration step is defined as follows:

Let an initial guess $\mathbf{w}^{(k)}$ be given. The new approximate solution $\mathbf{w}^{(k+1)}$ will be computed in the following way:

$$J_V \mathbf{w}_V^{(k+1)} = \mathbf{d}_V - J_{VE}\mathbf{w}_E^{(k)} - J_{VI}\mathbf{w}_I^{(k)} \tag{11}$$

$$J_E \mathbf{w}_E^{(k+1)} = \mathbf{d}_E - J_{EV}\mathbf{w}_V^{(k+1)} - J_{EI}\mathbf{w}_I^{(k)} \tag{12}$$

$$(D_I + L_I)\mathbf{w}_I^{(k+1)} = \mathbf{d}_I - J_{IV}\mathbf{w}_V^{(k+1)} - J_{IE}\mathbf{w}_E^{(k+1)} - U_I\mathbf{w}_I^{(k)}\,. \tag{13}$$

Here, L_I, D_I, and U_I are a strict lower triangular matrix, a diagonal matrix, and a strict upper triangular matrix, respectively, with $J_I = L_I + D_I + U_I$. The application of the proposed smoothing procedure requires to solve the systems of algebraic equations (11) – (13). If we suppose that at least one edge coupling node is generated on each part $\Gamma_{C,k}$ $(k = 1, 2, \ldots, k_C)$ of the coupling boundary and there exists no edge in the triangulation $\mathcal{T}_1$ connecting nodes on two different parts $\Gamma_{C,k}$ and $\Gamma_{C,k'}$, then J_V is a diagonal matrix and J_E is a block-diagonal matrix with tridiagonal blocks. Using the numbering of the nodes described in Subsection 4.2, the matrix J_I is also a block-diagonal matrix with the blocks $J_{I,i}$. Thus, we get a block-diagonal structure of the matrix $(D_I + L_I)$, where the blocks are triangular matrices.

From the algebraic point of view, solving (11) is trivial. Because of the block structure of the matrix J_E the system of algebraic equations (12) decomposes into k_C tridiagonal systems of algebraic equations. For solving these systems of equations we use a standard Gauss algorithm for tridiagonal matrices (see, e.g., [47]). Relation (13) describes nothing but one iteration step of the point–wise Gauss–Seidel iteration applied to a system of equations with the matrix J_I.

For solving the systems (11) and (12) we need the matrices J_V and J_E in assembled form. As mentioned in Subsection 4.2 this assembly requires communication. More precisely, we have to perform one communication over the cross–points for getting the diagonal matrix J_V, and we need two communications over the edge coupling nodes for getting the diagonal and the subdiagonal of the symmetric matrix J_E. It is clear that this assembly must be performed only if we have computed a new Jacobi matrix on a given grid.

Because of the representation (9) of the matrices J_{VE}, J_{VI}, J_{EI}, and the representation of the vector $\mathbf{d}$, the right–hand sides of the systems (11) and (12) are vectors of adding type. Before we can solve the systems of equations (11) and (12) we have to convert the corresponding right–hand sides into vectors of overlapping type. These type conversions require one communication over the cross–points and one communication over the edge coupling nodes in each smoothing step. After the type conversion of the right–hand sides and the assembly of the matrices J_V and J_E, each processor P_i has stored the full information about those parts of the systems of equations (11) and (12) which correspond to the nodes associated with the subdomain $\bar{\Omega}_i$. Taking into consideration the structure of the matrices J_V and J_E we see that each processor can determine those components of the vectors $\mathbf{w}_V^{(k+1)}$ and $\mathbf{w}_E^{(k+1)}$ which are related to the subdomain $\bar{\Omega}_i$ without any additional communication.

Because of the block structure of the matrices $(D_I + L_I)$ and U_I the point–wise Gauss-Seidel procedure can be performed in parallel, i.e. it is applied to the systems of equations

$$(D_{I,i} + L_{I,i})\mathbf{w}_{I,i}^{(k+1)} = \mathbf{d}_{I,i} - J_{IV,i}\mathbf{w}_{V,i}^{(k+1)} - J_{IE,i}\mathbf{w}_{E,i}^{(k+1)} - U_{I,i}\mathbf{w}_{I,i}^{(k)} \tag{14}$$

simultaneously. Each processor P_i has the full information about the right–hand side and the matrix $D_{I,i}+L_{I,i}$ of the corresponding system of equations (14), and therefore, no communication is necessary.

In the convergence theory we need a symmetric multigrid operator (see also Section 3.2). Therefore, we define additionally an analogous Gauss-Seidel type smoother which works in reverse order.

4.4.2 Interpolation and restriction procedures

Since the boundary value problem is discretized with piecewise linear trial functions we use linear interpolation within the multigrid algorithm. The restriction operator is defined as the adjoint operator to the interpolation operator. In the implementation of the multigrid algorithm, the corrections are stored as vectors of overlapping type and the defects as vectors of adding type. Therefore, we have to map a vector of overlapping type into a vector of overlapping type in the interpolation procedure. In the restriction procedure, a vector of adding type is mapped into a vector of adding type. Therefore, both procedures do not need any communication.

4.4.3 Coarse–grid solvers

We utilize parallelized preconditioned conjugate gradient methods applied to the corresponding Schur complement system as coarse–grid solvers. Here, communication is required in the two scalar products and in the preconditioner, whereas all other operations are completely parallel. A nonstandard formulation of the preconditioned conjugate gradient method which minimizes the communication between the processors (see [43, 44]) is used. The advantage of this formulation is the fact that the two scalar products per iteration step can be computed immediately one by another such that only one start–up time is necessary.

As preconditioners may be used the diagonal part of the matrix J_C, BPX–preconditioners with a global cross–point system [4, 50], or BPS–preconditioners using ideas of Dryja [7] on the coupling boundaries and a global cross–point system (see also [3]). In the case of the BPX and the BPS preconditioners we need a hierarchy of partitions of the coupling boundary. In general, such a hierarchy does not exist for the coarsest mesh. Therefore, we generate a sequence of nested auxiliary meshes down to a coarsest mesh which contains the cross-points only, and we work after some transformations on these auxiliary meshes.

5 Numerical results

5.1 Implementation

The Parallel Nested Newton algorithm with both, the DD-PCG solver and the global multigrid method as linear solver, is implemented in the parallel code FEM$\bigotimes$BEM [13, 15, 35]. In this way, all parts of the PNN algorithm, such as the grid generation, the matrix generation and representation, and the defect computation, are identical for (PCG) and (GMG). Thus, both methods can be compared. The algorithm was mainly tested on the parallel systems Power Xplorer (with maximal 16 processors Power PC 601) and the GC-Power Plus (with maximal 128 processors Power PC 601) with the operating system Parix.

In the following we describe our choice of components and parameters in the PNN method and in the solvers for linear problems. Indeed, we have tested various combinations of them, and we present here both, the best choice for (PCG) and the best choice for (GMG) with respect to the total computing time.

Parallel Nested Newton:
For practical computations, the Newton method (NM) should be chosen. Here, the numerical effort for the repeated Jacobi matrix generation is of minor importance than the fast convergence. Note that the matrix generation can be done completely in parallel [30, 33, 31].

Further, we apply a linear interpolation $\tilde{I}^q_{q-1}, q = 2, \ldots, l$ and restrict the number of Newton iterations on the grids $q = 2, \ldots, l-1$ to $j \leq 2$, cf. [26, 27]. We set $c_\tau = 0.25$, and always try to have the relaxation parameter close to 1 as in [26, 27]. Indeed, on all the grids except the coarsest ($q = 1$), all Newton iterates have been accepted with $\tau^j_q = 1$. The parameter ε_{lin} can be adapted to the quadratic convergence speed of the Newton method [27]. For the presented examples, best results with respect to the total computing time have been achieved with $\varepsilon_{\text{lin}} = 0.01$.

DD preconditioned conjugate gradient method:
The components of the (PCG) algorithm are chosen as described in Subsection 4.3.

Global parallelized multigrid method:
In the (GMG) solver, we used a V-cycle with 2 pre- and 2 postsmoothing steps of the parallelized Gauss-Seidel type presented in Subsection 4.4. The maximal number of multigrid iterations has been restricted to 2, this number is doubled if the multigrid convergence rate affects the Newton rate essentially, cf. [26, 27]. With respect to the coarse grid solver, a preconditioned Schur complement CG solver turned out to be sufficient where best results have been achieved with a relative accuracy of 0.1 in the multilevel case ($q \geq 2$), and a DD based (BPS) preconditioner.

We compare the performance of the algorithms for an electric magnet as an academic test example, and for a direct current motor as a real-life example.

5.2 Electric magnet

An electric magnet with Dirichlet boundary conditions and a 50 cm × 50 cm iron core serves as a first test example for computations. In Figure 1, the initial grid for the domain Ω decomposed into 64 congruent square subdomains and level lines of the solution are shown. In Table 1, we compare the algorithms for different numbers of grids being used, i.e., for different discretizations, whereas Table 2 is to demonstrate the performance for different current densities, i.e., for different nonlinear effects. We distinguish the time for the solver from the time for generating the Jacobi matrices and defect vectors.

Further, we measure the processor time and the communication time for each processor separately. The communication time includes the waiting for input and output as well as the communication itself. Therefore, the minimal communication time of all processors is a measure for the communication overhead, cf. [16].

Although, in the global multigrid (GMG) a few seconds more are spent for communication, the global algorithm is faster. The reason can be found in the excellent convergence of the multigrid method, cf. also [35]. In all but four cases (the exceptions are for 40A/mm^2, grids 4, 5, 6) the criterion given by $\varepsilon_{\text{lin}} = 0.01$ is fulfilled after two or even one multigrid step.

The speed is always normalized with the total time of the PCG algorithm for the identical problem.

5.3 Electric machine

A direct current motor which is excited by permanent magnets serves as a real-life test example. In order to demonstrate the scale-up of our algorithms we have considered an additional test problem, a quarter of the original machine, so that we are in the position to compare problems with 16 and 64 subdomains.

The decomposition of the cross-section of the machine into 64 subdomains which are assigned to the 64 processors is shown in Figure 2 on the left-hand side. Here, the cross-points are marked. The

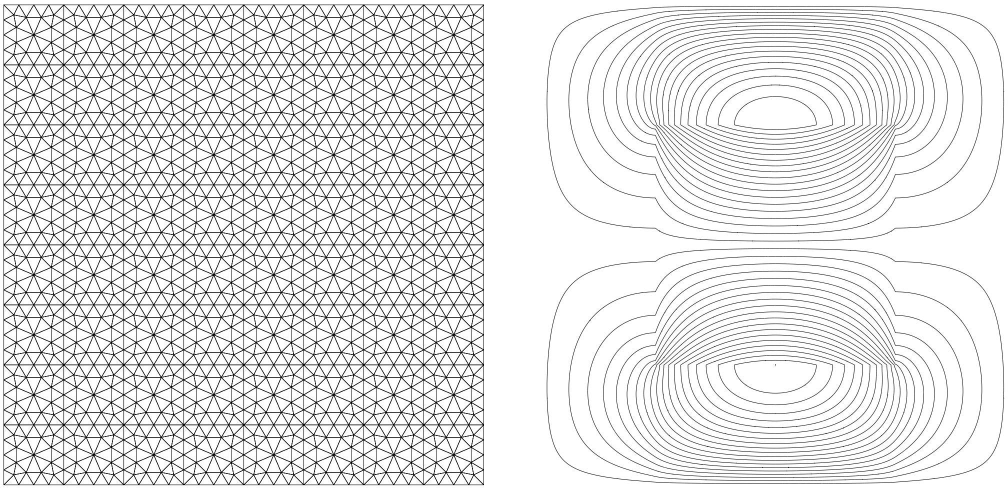

Figure 1: Initial grid for the electric magnet and equipotential lines of the solution for the current density $4\,\mathrm{A/mm^2}$

Table 1: Performance for different discretizations (electric magnet)

Solver	**PCG**		**GMG**	
Number of grids Number of unknowns	5 326 689	6 1 308 705	5 326 689	6 1 308 705
Newton iterations 1st grid CG iterations 1st grid	4 5,5,6,7	4 5,5,6,7	4 5,5,6,7	4 5,5,6,7
Newton iterations 2nd grid CG/MG iterations 2nd grid	2 6,6	2 6,6	2 1,2	2 1,2
Newton iterations 3rd grid CG/MG iterations 3rd grid	2 4,6	2 4,6	2 1,2	2 1,2
Newton iterations 4th grid CG/MG iterations 4th grid	2 4,7	2 4,7	2 1,2	2 1,2
Newton iterations 5th grid CG/MG iterations 5th grid	3 4,7,7	2 4,7	3 1,1,2	2 1,1
Newton iterations 6th grid CG/MG iterations 6th grid		3 3,7,8		3 1,1,2
Time (system generation) Time (solver)	4.3 10.1	14.6 26.8	4.3 8.2	14.4 12.0
Total time	14.4	41.4	12.5	26.4
Maximal processor time Minimal communication time	8.4 6.0	33.9 7.5	4.4 8.1	16.8 9.6
Speed (normalized)	1	1	1.15	1.57

Time in seconds, GC-Power Plus, 64 processors (subdomains); relative accuracy $\varepsilon = 10^{-6}$

Table 2: Performance for different nonlinear behaviour

Solver	PCG			GMG		
Current density S [A/mm^2]	4	40	400	4	40	400
Newton iter. 1st grid CG iterations 1st grid	4 5,5,6,7	10 5,5,5,6,5, 5,5,5,5,7	9 5,5,4,4, 4,4,4,5,6	4 5,5,6,7	10 5,5,5,6,5, 5,5,5,5,7	9 5,5,4,4, 4,4,4,5,6
Newton iter. 2nd grid CG/MG iterations 2nd grid	2 6,6	2 5,6	2 5,6	2 1,2	2 1,2	2 1,2
Newton iter. 3rd grid CG/MG iterations 3rd grid	2 4,6	2 5,7	2 5,6	2 1,2	2 1,2	2 1,2
Newton iter. 4th grid CG/MG iterations 4th grid	2 4,7	2 3,7	2 3,7	2 1,2	2 1,2	2 1,2
Newton iter. 5th grid CG/MG iterations 5th grid	2 4,7	2 3,8	2 3,7	2 1,1	2 1,2	2 1,2
Newton iter. 6th grid CG/MG iterations 6th grid	3 3,7,8	3 2,9,13	3 2,7,8	3 1,1,2	3 1,2,2	2 1,2
Time (generation) Time (solver)	14.6 26.8	15.8 34.7	15.8 27.8	14.4 12.0	15.6 16.9	13.2 13.4
Total time	41.4	50.5	43.6	26.4	32.5	26.6
Maximal processor time Minimal communication time	33.9 7.5	40.2 10.3	34.3 9.3	16.8 9.6	18.8 13.7	15.1 11.5
Speed (normalized)	1	1	1	1.57	1.55	1.64

Time in seconds, GC-Power Plus, 64 processors (subdomains); relative accuracy $\varepsilon = 10^{-6}$, 1 308 705 unknowns

coarse grid which is to be refined 4 or 5 times in the multilevel calculations is given on the right-hand side of Figure 2. We present equipotential lines in Figure 3.

We conclude that the global multigrid method is faster again, but the Domain Decomposition PCG method has the better scalability. Here, we denote the normalized ratio between problem size and time as scale-up; the quotient of scale-up and the number of processors is called scaled efficiency. For DD type methods, the scaled efficiency decreases only slightly with a growing number of processors involved in the solution of growing problems, cf. [32, 30, 33, 34, 31].

Further, in Figure 4 we present two bar graphs for the whole motor. They indicate the time proportion between communication and processing for the PCG (on the left hand side) and for the GMG (on the right hand side). Each bar shows the input time including waiting (left, grey), the output time including waiting (middle, black), and the processor time (right, white), in relation to the total time for each of the 64 processors. Obviously, the processor time differs for different load caused by the decomposition of the domain. Therefore, the processors having less load have to wait for the others.

5.4 Comparison in different parallel environments

To compare the performance on different multiprocessor systems, we carried out additional computations on a Multicluster 2 (with 16 processors, transputers T 805, using Parix) and on a cluster of 8 workstations SPARC 2 with PVM. In the latter case two processes are running on each workstation. In our code, all communication between processors is realized via a (virtual) hypercube topology using a library of standard communication routines [16]. This library has been implemented for many parallel systems including the above-mentioned. Therefore, we are able to run the identical parallel code in different multiprocessor environments.

We present the total computing times (quarter of the motor, 5 grids) in Table 4. From our results, we conclude that global multigrid is faster on all machines. The speed ratio has been calculated as the reciprocal of the total time ratio between GMG and PCG. It can be observed that the speed ratio is the highest on the transputer cluster where we have a good balance between processor power and

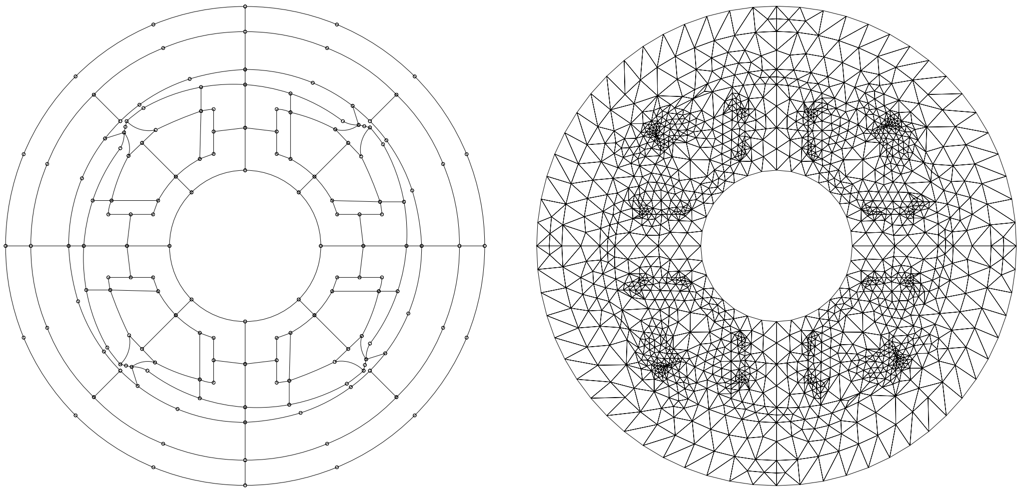

Figure 2: Decomposition and discretization (coarse grid) of the motor

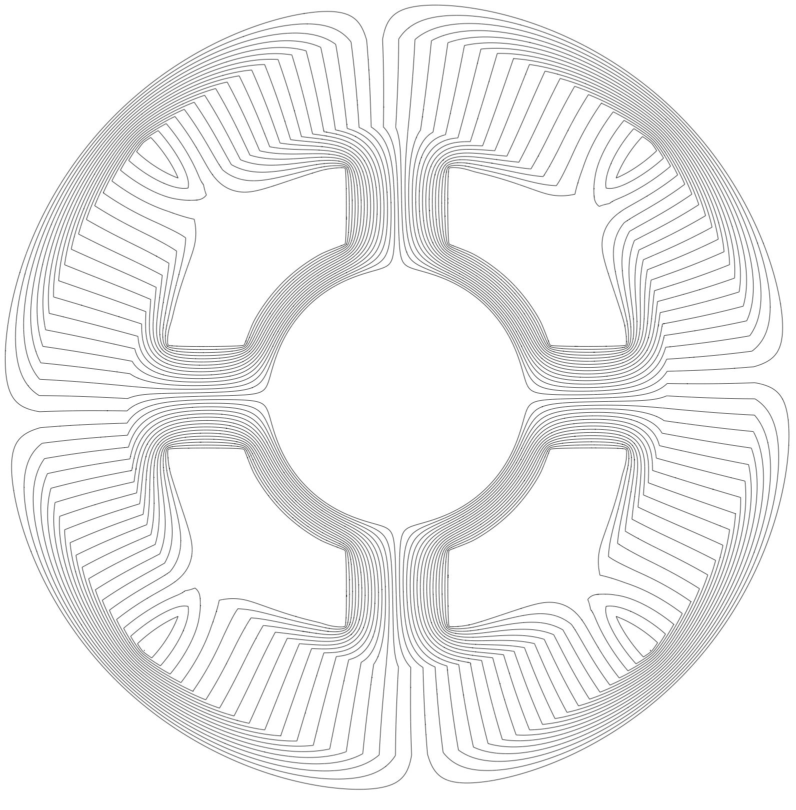

Figure 3: Equipotential lines of the magnetic vector potential for the motor

Table 3: Performance for the electric motor and a related test problem

Solver	**PCG**		**GMG**	
Example	1/4 machine	machine	1/4 machine	machine
subdomains (processors)	16	64	16	64
Number of unknowns	374 129	1 514 008	374 129	1 514 008
Newton iter. 1st grid	3	3	3	3
CG iterations 1st grid	6,11/16	6,11/15	6,11,12	6,11,11
Newton iter. 2nd grid	2	2	2	2
CG/MG iterations 2nd grid	9,10	9,11	1,2	1,2
Newton iter. 3rd grid	2	2	2	2
CG/MG iterations 3rd grid	10,12	10,12	1,2	2,2
Newton iter. 4rth grid	2	2	2	2
CG/MG iterations 4th grid	11,13	11,13	1,2	1,2
Newton iter. 5rth grid	2	2	2	2
CG/MG iterations 5th grid	11,15	11,15	1,2	1,2
Newton iter. 6th grid	2	2	2	2
CG/MG iterations 6th grid	12,16	13,16	1,2	1,2
Time (system generation)	15.8	17.3	15.0	16.5
Time (solver)	45.5	52.3	8.9	18.3
Total time	61.3	69.6	23.9	34.8
Maximal processor time	55.8	56.0	18.3	18.7
Minimal communication time	5.5	13.6	5.6	16.1
Scale-up (normalized)	1.0 $\rightarrow$	3.56	1.0 $\rightarrow$	2.78
Scaled efficiency (relative)	1.0 $\rightarrow$	0.89	1.0 $\rightarrow$	0.69

Time in seconds, GC-Power Plus, maximal 64 processors; relative accuracy $\varepsilon = 10^{-4}$

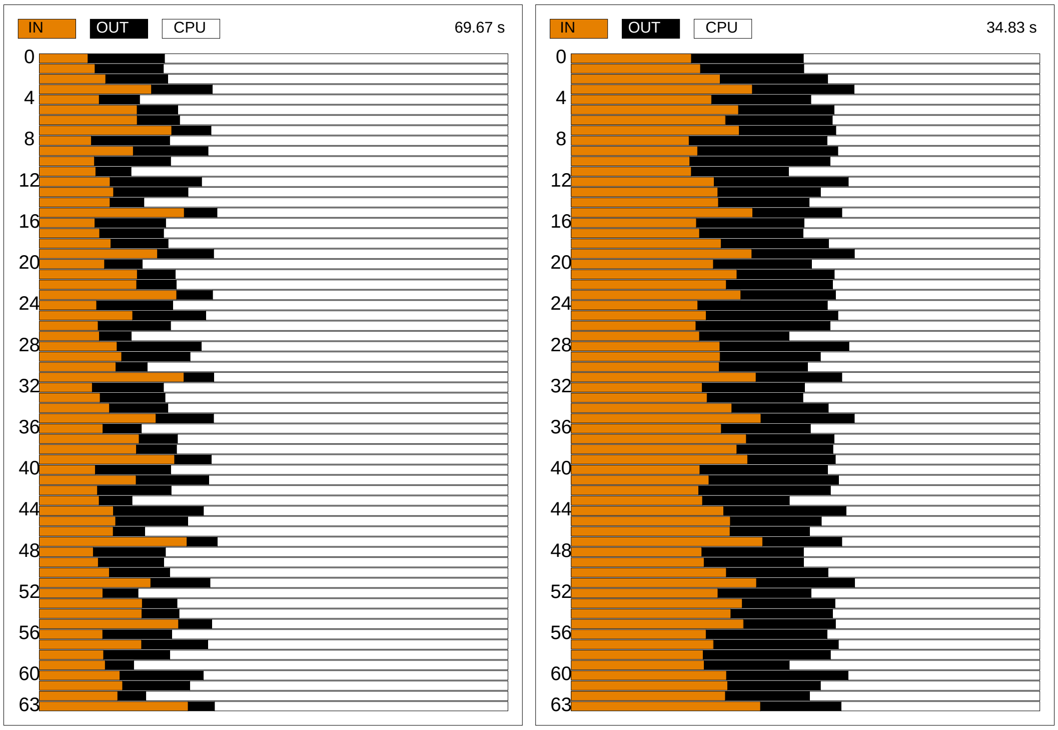

Figure 4: Communication and processor time (left: PCG, right: GMG)

communication power but quite slow processors.

If the Power Xplorer is compared with the transputer, the processor power has increased much more than the the communication power. Consequently, the speed ratio is lower.

In the workstation cluster, we have a slow communication network. Almost half of the total time of the PCG algorithm, and more than half of the total time of the GMG algorithm, is spent for communication.

Note that for this example the GMG method performs two Newton iterations on the fifth grid, while the PCG algorithm needs three iterations. Therefore both, the system generation time and the communication time, are lower for the global multigrid method.

Table 4: Performance for an identical problem in different parallel environments

System Processors	Power Xplorer Parix 16 * Power PC 601		Multicluster 2 Parix 16 * Transputer T805		Workstation cluster PVM 8 * SPARC 2	
Solver	PCG	GMG	PCG	GMG	PCG	GMG
Time (system generation)	5.0	4.0	61.5	46.9	63.6	51.8
Time (solver)	15.8	5.1	357.0	36.5	206.9	114.8
Total time	20.8	9.1	418.5	83.4	270.5	166.6
Maximal processor time	16.5	4.8	409.3	76.6	141.8	59.7
Minimal commun. time	4.3	4.3	9.2	6.8	128.7	106.9
Speed ratio	1 : 2.29		1 : 5.34		1 : 1.62	

Time in seconds, 16 subdomains, 93 377 unknowns; relative accuracy $\varepsilon = 10^{-4}$

6 Conclusions

We conclude that both methods, the parallel CG solver with DD preconditioning (PCG) and the parallel global multigrid method (GMG) can be applied in a nested Newton algorithm for solving nonlinear electromagnetic field problems very efficiently on MIMD parallel computers. In particular, using the presented global multigrid method which applies the data structure of the non-overlapping DD method and a DD coarse grid solver, we can solve a nonlinear problem faster than using the DD method itself. The time ratio depends on the relation between communication power and processor power of the multiprocessor system.

Our computations show that the communication time in the GMG method is approximately the same as in the PCG algorithm, but the quota of the total time is lower for the PCG. Further, we conclude from the computations that the PCG algorithm has the better scalability.

The Domain Decomposition ideas can be extended to three-dimensional problems. Therefore, it is very promising to extend both solvers, the CG with DD preconditioning and the global multigrid method based on the DD data distribution (and the DD coarse grid solver) to the solution of the discrete finite element equations of three-dimensional problems from magnetics as well as from, e.g. continuum mechanics or other fields of research.

References

[1] P. Bastian. *Parallele adaptive Mehrgitterverfahren.* PhD thesis, Universität Heidelberg, 1994.

[2] P. Bastian and G. Horton. Parallelization of robust multigrid methods: ILU factorization and frequency decomposition method. *SIAM J. Sci. Stat. Comput.*, 12(6):1457–1470, Nov. 1991.

[3] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring I – IV. *Mathematics of Computation*, 1986, 1987, 1988, 1989. 47, 103–134, 49, 1–16, 51, 415–430, 53, 1–24.

[4] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Mathematics of Computation*, 55(191):1 – 22, 1990.

[5] T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, editors. *Domain decomposition methods for partial differential equations*, Philadelphia, 1989. SIAM. Proc. of the 2nd International Symposium, Los Angeles, 1988.

[6] T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, editors. *Domain decomposition methods for partial differential equations*, Philadelphia, 1990. SIAM. Proc. of the 3rd International Symposium, Houston, 1989.

[7] M. Dryja. A capacitance matrix method for Dirichlet problems on polygonal regions. *Numerische Mathematik*, 39(1):51–64, 1982.

[8] M. Dryja and O. B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In: [6], pages 3–21 (1990).

[9] G. Globisch. PARMESH - a parallel mesh generator. *Parallel Computing*, 21(3):509–524, 1995.

[10] R. Glowinski, G. H. Golub, G. A. Meurant, and J. Periaux, editors. *Domain decomposition methods for partial differential equations*, Philadelphia, 1988. SIAM. Proc. of the 1st International Symposium, Paris, 1987.

[11] R. Glowinski, Y. A. Kuznetsov, G. A. Meurant, and J. Periaux, editors. *Domain decomposition methods for partial differential equations*, Philadelphia, 1991. SIAM. Proc. of the 4th International Symposium, Moscow, 1990.

[12] M. Goppold, G. Haase, B. Heise, and M. Kuhn. Preprocessing in BE/FE domain decomposition methods. Technical report, Johannes Kepler University Linz, 1995. In preparation.

[13] G. Haase, B. Heise, M. Jung, and M. Kuhn. FEM$\bigotimes$BEM - a parallel solver for linear and nonlinear coupled FE/BE-equations. DFG-Schwerpunkt "Randelementmethoden", Report 94-16, University Stuttgart, 1994.

[14] G. Haase, B. Heise, M. Kuhn, and U. Langer. Adaptive domain decomposition methods for finite and boundary element equations. Technical Report 95-2, Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerische Mathematik und Optimierung, 1995.

[15] G. Haase, B. Heise, M. Kuhn, and U. Langer. FEM$\bigotimes$BEM - the joint power of the domain decomposition FEM/BEM and the Xplorer. Technical report, 1995. In: PowerXplorer User Report, University Düsseldorf and Parsytec Computer, Düsseldorf.

[16] G. Haase, T. Hommel, A. Meyer, and M. Pester. Bibliotheken zur Entwicklung paralleler Algorithmen. Preprint, Faculty for Mathematics SPC 95_20, TU Chemnitz-Zwickau, 1995. 3rd edn., June 1995.

[17] G. Haase and U. Langer. On the use of multigrid preconditioners in the domain decomposition method. In W. Hackbusch, editor, *Parallel Algorithms for PDEs*, pages 101–110, Braunschweig, 1990. Vieweg. Proc. of the 6th GAMM–Seminar, Kiel, 1990.

[18] G. Haase and U. Langer. The non-overlapping domain decomposition multiplicative Schwarz method. *International Journal of Computer Mathematics*, 44:223–242, 1992.

[19] G. Haase and U. Langer. Iterative solution techniques in the FEM on massively parallel computers. Number 93-04 in Informatik Berichte, pages 49 – 61. TU Braunschweig, 1993. Proceedings of the workshop "Parallelisierung im Wissenschaftlichen Rechnen", held at Braunschweig, 2.-4. Juni.

[20] G. Haase and U. Langer. Domain decomposition vs. adaptivity. In M. Křížek, P. Neittaanmäki, and R. Stenberg, editors, *Finite Element Methods – Fifty Years of the Courant Element*, volume 164 of *Lecture Notes in Pure and Applied Mathematics*, pages 243 – 258, New York, Basel, Hong Kong, 1994. Marcel Dekker, Inc.

[21] G. Haase, U. Langer, and A. Meyer. A new approach to the Dirichlet domain decomposition method. In S. Hengst, editor, *Proceedings of the "5-th Multigrid Seminar" held at Eberswalde, May 14-18, 1990*, pages 1–59, Berlin, 1990. Karl-Weierstrass-Institute, Academy of Sciences. Report-Nr. R-MATH-09/90.

[22] G. Haase, U. Langer, and A. Meyer. The approximate Dirichlet domain decomposition method. Part I: An algebraic approach. Part II: Applications to 2nd-order elliptic boundary value problems. *Computing*, 47:137–151 (Part I), 153–167 (Part II), 1991.

[23] G. Haase, U. Langer, and A. Meyer. Domain decomposition preconditioners with inexact subdomain solvers. *J. of Num. Lin. Alg. with Appl.*, 1:27–42, 1992.

[24] G. Haase, U. Langer, and A. Meyer. Parallelisierung und Vorkonditionierung des CG-Verfahrens durch Gebietszerlegung. In *Parallele Algorithmen auf Transputersystemen*, Teubner-Scripten zur Numerik III, Stuttgart, 1992. Teubner. Tagungsbericht der GAMM-Tagung, 31. Mai-1. Juni 1991, Heidelberg.

[25] G. Haase, U. Langer, A. Meyer, and S. V. Nepomnyaschikh. Hierarchical extension operators and local multigrid methods in domain decomposition preconditioners. *East-West J. Numer. Math.*, 2(3):173–193, 1994.

[26] B. Heise. Mehrgitter–Newton–Verfahren zur Berechnung nichtlinearer magnetischer Felder. Wissenschaftliche Schriftenreihe 4/1991, Technische Universität Chemnitz, 1991.

[27] B. Heise. Nonlinear field calculations with multigrid-Newton methods. *IMPACT of Computing in Science and Engineering*, 5:75–110, 1993.

[28] B. Heise. Sensitivity analysis for nonlinear magnetic field simulation. In H. Bandemer, editor, *Modelling Uncertain Data*, volume 68 of *Mathematical Research*, pages 40–45, Berlin, 1993. Akademie Verlag. Proc. of GAMM–Workshop, Bergakademie Freiberg, March 21–24, 1992.

[29] B. Heise. Analysis of a fully discrete finite element method for a nonlinear magnetic field problem. *SIAM J. Numer. Anal.*, 31(3):745–759, 1994.

[30] B. Heise. Nonlinear magnetic field simulation with FE/BE domain decomposition methods on MIMD parallel computers. DFG-Schwerpunkt "Randelementmethoden", Report 94-19, University Stuttgart, 1994.

[31] B. Heise. Nonlinear field simulation with FE domain decomposition methods on massively parallel computers. 1995. Submitted for publication.

[32] B. Heise. Nonlinear simulation of electromagnetic fields with domain decomposition methods on MIMD parallel computers. *Journal of Computational and Applied Mathematics*, 1995. Accepted for publication. To appear.

[33] B. Heise. Parallel solvers for coupled FEM-BEM equations with applications to non-linear magnetic field problems. In W. Hackbusch and G. Wittum, editors, *Numerical Treatment of Coupled Systems*, volume 51 of *Notes on Numerical Fluid Mechanics*, Braunschweig, Wiesbaden, 1995. Vieweg. Proc. of the 11th GAMM–Seminar, Kiel, January 20 to 22, 1995.

[34] B. Heise and M. Kuhn. Parallel solvers for linear and nonlinear exterior magnetic field problems based upon coupled FE/BE formulations. *Computing*, 1995. Accepted for publication. To appear. Also Report No. 486, Inst. of Mathematics, Univ. Linz, 1995.

[35] M. Jung. Parallelization of multigrid methods based on domain decomposition ideas. Preprint SPC 95_27, Technical University Chemnitz–Zwickau, Faculty for Mathematics, 1995.

[36] M. Jung, U. Langer, A. Meyer, W. Queck, and M. Schneider. Multigrid preconditioners and their applications. In G. Telschow, editor, *Third Multigrid Seminar, Biesenthal 1988*, pages 11–52, Berlin, 1989. Karl–Weierstrass–Institut. Report R–MATH–03/89.

[37] D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, editors. *Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1992. SIAM. Proc. of the 5th International Symposium, Norfolk, Va., 1991.

[38] D. E. Keyes and J. Xu. *Domain Decomposition Methods in Scientific and Engineering Computing: Proceedings of the Seventh International Conference on Domain Decomposition*, volume 180 of *Contemporary Mathematics*. American Mathematical Society, Providence, Rhode Island, 1994.

[39] U. Langer. Parallel iterative solution of symmetric coupled FE/BE- equations via domain decomposition. In: [45], pages 335–344 (1994).

[40] U. Langer and W. Queck. Preconditioned Uzawa–type iterative methods for solving mixed finite element equations. Wissenschaftliche Schriftenreihe 3, Technische Universität Karl–Marx–Stadt (Chemnitz), 1987.

[41] P. Leinen. *Ein schneller adaptiver Löser für elliptische Randwertprobleme auf Seriell– und Parallelrechnern*. PhD thesis, Universität Dortmund, 1990. Dissertation.

[42] O. A. McBryan, P. O. Frederickson, J. Linden, A. Schüller, K. Solchenbach, K. Stüben, C.-A. Thole, and U. Trottenberg. Multigrid methods on parallel computers — a survey of recent developments. *IMPACT of Computing in Science and Engineering*, 3:1–75, 1991.

[43] M. Meisel and A. Meyer. Implementierung eines parallelen Schur–Komplement CG–Verfahrens in das Programmpaket FEAP. Preprint SPC 95_2, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[44] M. Meisel and A. Meyer. Kommunikationstechnologien beim parallelen vorkonditionierten Schur–Komplement CG–Verfahren. Preprint SPC 95_19, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[45] A. Quarteroni, J. Périaux, Y. A. Kuznetsov, and O. B. Widlund. *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition*, volume 157 of *Contemporary Mathematics*. American Mathematical Society, Providence, Rhode Island, 1994.

[46] K. Ressel. Gebietszerlegung und Mehrgitterverfahren: Varianten, numerische Untersuchungen und Aspekte der Parallelisierung. Diplomarbeit, Mathematisches Institut, Universität Köln, Köln, 1990.

[47] A. A. Samarskij and E. S. Nikolajev. *Numerical Methods for Grid Equations. Vol. I: Direct Methods*. Birkhäuser, Basel Boston Berlin, 1989.

[48] B. F. Smith. Domain decomposition algorithms for the partial differential equations of linear elasticity. Technical Report 517, Department of Computer Science, Courant Institute, New York, 1990.

[49] B. F. Smith. A parallel implementation of an iterative substructuring algorithm for problems in three dimensions. Preprint MCS–P249–0791, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, 1991.

[50] C. H. Tong, T. F. Chan, and C. J. Kuo. A domain decomposition preconditioner based on a change to a multilevel nodal basis. *SIAM J. Sci. Stat. Comput.*, 12(6):1486–1495, 1991.