Technische Universität Chemnitz-Zwickau

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Bodo Heise* and Michael Jung

Scalability, Efficiency, and Robustness of Parallel Multilevel Solvers for Nonlinear Equations

Preprint SFB393/96_12

Abstract

In this paper we compare the performance, scalability, and robustness of different parallel algorithms for the numerical solution of nonlinear boundary value problems arising in the magnetic field computation and in solid mechanics. These problems are discretized by using the finite element method with triangular meshes and piecewise linear functions. The nonlinearity is handled by a nested Newton solver, and the linear systems of algebraic equations within each Newton step are solved by means of various iterative solvers, namely multigrid methods and conjugate gradient methods with preconditioners based on domain decomposition, multigrid, or BPX techniques, respectively. The basis of the implementation of all solvers is a non-overlapping domain decomposition data structure such that they are well-suited for parallel machines with MIMD architecture.

Key words: nonlinear partial differential equations, parallel computing, multigrid methods, domain decomposition, finite element methods, magnetic field calculations

AMS(MOS) subject classification: 65N55, 65N22, 65N30, 78A30

* The work of this author is supported by the Austrian Science Fund – 'Fonds zur Förderung der wissenschaftlichen Forschung' – under project P 11215-TEC.

Preprint-Reihe des Chemnitzer SFB 393

SFB393/96_12 **September 1996**

Authors' address:

Dr. rer. nat. Bodo Heise
Institute for Mathematics
Johannes Kepler University Linz
Altenberger Straße 69
A - 4040 Linz, Austria

e-mail: `heise@numa.uni-linz.ac.at`

http://www.numa.uni-linz.ac.at/Staff/heise.html

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

e-mail: `michael.jung@mathematik.tu-chemnitz.de`

http://www.tu-chemnitz.de/~jung/jung.html

1 Introduction

Recently, Multiple Instruction Multiple Data (MIMD) parallel computers with message-passing principle have found more and more interest. These parallel machines provide sufficient CPU power and sufficiently large storage capacity as it is necessary for the numerical simulation of complex processes.Therefore, efficient parallel solvers for large systems of algebraic equations resulting from, e.g., the finite element (FE) or boundary element (BE) discretization of partial differential equations, have been developed.

There are different approaches for the construction of efficient parallel solvers. The first one is the application of Domain Decomposition (DD) preconditioners within the preconditioned conjugate gradient (PCG) method. For example, the non-overlapping DD (NODD) methods have been applied successfully. Here, one has to define preconditioning matrices for the stiffness matrices of the corresponding problems in the subdomains, a preconditioner for the Schur complement, and a basis transformation (for more details see, e.g., [2, 6, 7, 10, 14, 15, 20, 27, 28]). The appropriate definition of these preconditioners and of the basis transformation leads to (almost) optimal PCG methods, i.e. the number of iterations for getting an approximate solution with a relative accuracy ε is of the order $\mathcal{O}(\ln h_q^{-1} \ln \varepsilon^{-1})$ or $\mathcal{O}(\ln \varepsilon^{-1})$ (h_q denotes the discretization parameter).

A second possibility for getting parallel solvers is the implementation of well-known global optimal iterative solvers, as e.g. multigrid methods or PCG methods with BPX preconditioners, on parallel machines. Using a NODD data structure, these solvers need, just as the DD-PCG method, a communication cost per iteration step which is one order lower than the FE problem itself (see, e.g., [24]).

A first comparison of two of the algorithms, namely the CG method with DD preconditioner (DD-PCG) and the global multigrid (GMG) method as linear problem solvers in a nested Newton framework (i.e. an inexact Newton method [4, 5], see [17, 19, 20]) can be found in [21]. Therein, we discovered that the Newton-GMG method can be faster than the Newton-DD-PCG method for model problems and a practical problem with low nonlinearity, but we observed the better scalability of the Newton-DD-PCG.

In addition to the solvers examined in [21], we implemented a parallel CG algorithm with global BPX preconditioner (GBPX-PCG) [3] which uses the NODD data structure. Further, the GMG method can serve as a preconditioner (GMG-PCG) in the parallel CG algorithm (cf. [23]), too.

The aim of the present paper is to compare the performance, scalability, and robustness of the different parallel algorithms for difficult nonlinear practical problems. This includes tests for the magnetic field computation in electric machines with complicated interior geometry and strong nonlinearity, i.e., high saturation of the iron parts. The latter causes a local anisotropy of the linear Jacobi operator. Moreover, we present test calculations for another nonlinear test problem originating from a shape design problem (see [8, 25]) that had been already investigated in [17, 20].

Further, we are interested in getting efficiency measures by comparing calculations on different numbers of processors, say p and q, with $p \leq q$. The Relative Scale-up is defined by

$$S(p,q) = \frac{\text{Time per unknown}(p)}{\text{Time per unknown}(q)} = \frac{T(p)}{N(p)} \cdot \frac{N(q)}{T(q)},$$

where $T(p)$ ($T(q)$) denotes the calculation time of the parallel algorithm, and $N(p)$ ($N(q)$) is the number of unknowns (i.e. degrees of freedom) for the problem on p (q) processors

[11]. The Relative Scaled Efficiency is defined by

$$E(p,q) = \frac{p}{q} \cdot S(p,q)$$

For an ideally scalable algorithm, we get $E(p,q) = 1$, otherwise $E(p,q) < 1$.

The aim of discussing these "scaled" efficiency measures is to compare calculations in the case that the number of processors used is adapted to the complexity of the problem, i.e. $N(p)/N(q) \approx p/q$. We have to distinguish these measures from the practically less relevant Relative Speedup $\tilde{S}(p,q) = T(p)/T(q)$ and Relative Efficiency $\tilde{E}(p,q) = p/q \cdot \tilde{S}(p,q)$ in the case $N(p) \approx N(q)$.

The rest of the paper is organized as follows. In Section 2, we formulate the electromagnetic field problem as a nonlinear boundary value problem. We apply the nested Newton-DD method for linearization. We discuss some properties of the discrete Jacobi operators involved in the Newton method. In Section 3, we describe the different methods for solving the arising linear problems. Section 4 is devoted to the numerical results. We present performance results of the magnetostatic field simulation for a direct current motor (DC motor) and an induction machine as well as for the nonlinear shape design problem. Finally, we add some concluding remarks in Section 5.

2 The nonlinear problem and its linearization

A two–dimensional stationary magnetic field problem involving the saturation effects of ferromagnetic materials can be written as a nonlinear boundary value problem in its variational formulation 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

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

Here, $\Omega \subset \mathbf{R}^2$ denotes a bounded domain. The physical model has been developed from Maxwell's equations, see [18] for details. We assume that Ω representing the cross–section of some electromagnetic device lies in the x–y–plane of $\mathbf{R}^3$. Then, the solution u is the z–component of some vector potential $\vec{\mathrm{A}}$. The z–component of the current density is represented by S, and the vector $\vec{\mathrm{H}}_0 = (\mathrm{H}_{0\mathrm{x}}, \mathrm{H}_{0\mathrm{y}}, 0)^\top$ describes the magnetization of permanent magnets. The nonlinearity of the problem is represented 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 assume that the function ν depends on the position $x \in \Omega$, but ν becomes independent of x inside each subdomain $\hat{\Omega}_j$, i.e.

$$\nu(x, B) = \nu^{(j)}(B) \quad \text{if} \quad x \in \hat{\Omega}_j, \quad j = 1, \ldots, N_M. \tag{2}$$

The function $\nu^{(j)}(B)$ is constant, $\nu^{(j)}(B) \equiv \nu_1^{(j)}$, if the material in $\hat{\Omega}_j$ is not ferromagnetic (e.g., copper, air, vacuum). Assuming certain monotonicity and boundedness conditions on the functions $\nu^{(j)}(.)$ we can prove the existence and uniqueness of the solution of the variational problem (1) [17, 18]. Standard FE discretization with linear triangular elements has been discussed in [18]. Therein, the algorithm for monotonicity preserving spline interpolation of a pointwise given material function $\nu^{(j)}(.)$ is described, error estimates are given, too.

The parallel nested Newton (PNN) method for solving the discrete problems combining a Newton-like method with the nested iteration and a suitable parallel solver, is described in detail, e.g., in [19, 20, 21]. Therefore, we recall the main definitions only.

The substructuring into non-overlapping subdomains $\Omega_i, i \in \mathcal{I}^\star := \{1, \ldots, p\}$, which are assigned to p processors of the MIMD computer can be defined as follows

$$\bar{\Omega} = \bigcup_{j=1}^{N_M} \bar{\hat{\Omega}}_j = \bigcup_{i \in \mathcal{I}^\star} \bar{\Omega}_i = \bigcup_{j=1}^{N_M} \bigcup_{i \in \mathcal{I}_j} \bar{\Omega}_i \tag{3}$$

with index sets fulfilling

$$\mathcal{I}_j \subset \mathcal{I}^\star := \{1, \ldots, p\}, \quad \bigcup_{j=1}^{N_M} \mathcal{I}_j = \mathcal{I}^\star, \quad \mathcal{I}_j \cap \mathcal{I}_k = \emptyset \;\; \forall j \neq k,$$

i.e., the subdomains $\hat{\Omega}_j$ determined by the materials may be decomposed further (cf. [10, 14]). 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 $\mathcal{T}_q, q = 1, \ldots, l$, of Ω creating a sequence $V_1, V_2, \ldots, V_l \subset V = H_0^1(\Omega)$ of spaces of linear finite elements. The FE isomorphism is denoted by $\Phi_q : \mathbf{R}^{N_q} \longrightarrow V_q$.

We obtain a sequence of variational problems for $q = 1, \ldots, l$:
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

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

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 $\mathbf{f}_q \in \mathbf{R}^{N_q}$.

In the Newton method, we have to solve linear systems of the form

$$K_q'[\mathbf{v}_q]\mathbf{w}_q = \mathbf{d}_q \tag{6}$$

with the Fréchet derivative of K_q at a vector $\mathbf{v}_q$,

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

which can be represented by the Jacobi matrix.

The analysis shows that this operator can be strongly anisotropic. We recall from [17] that the Jacobian can be rewritten as

$$(K_q'[\mathbf{v}_q]\mathbf{w}_q, \mathbf{z}_q)_q = \sum_{\mathcal{K}\in\mathcal{T}_q} \text{meas}\,(\mathcal{K})\ \nabla^\top\Phi_q\mathbf{w}_q\ \ Q_{j(\mathcal{K})}(\nabla\Phi_q\mathbf{v}_q)\ \ \nabla\Phi_q\mathbf{z}_q \quad \forall\mathbf{w}_q,\mathbf{z}_q\in\mathbf{R}^{N_q}, \quad (7)$$

with

$$Q_j(\mathbf{t}) = \nu^{(j)}(|\mathbf{t}|)I_2 + \frac{\nu^{(j)\prime}(|\mathbf{t}|)}{|\mathbf{t}|}\mathbf{t}\,\mathbf{t}^\top, \quad (8)$$

where $\mathbf{t} = (t_1, t_2)^\top \in \mathbf{R}^2$, the matrix $I_2 \in \mathbf{R}^{2\times 2}$ denotes the identity matrix, and $j(\mathcal{K})$ is the corresponding material index, i.e., $j(\mathcal{K})$ is defined by $\mathcal{K} \subset \bar{\hat{\Omega}}_{j(\mathcal{K})} \quad \forall\mathcal{K}\in\mathcal{T}_q$, cf. (2).

The matrix $Q_j \in \mathbf{R}^{2\times 2}$ can be rewritten as

$$Q_j = \begin{pmatrix} \nu^{(j)}(|\mathbf{t}|) + \frac{\nu^{(j)\prime}(|\mathbf{t}|)}{|\mathbf{t}|}t_1^2 & \frac{\nu^{(j)\prime}(|\mathbf{t}|)}{|\mathbf{t}|}t_1t_2 \\ \frac{\nu^{(j)\prime}(|\mathbf{t}|)}{|\mathbf{t}|}t_1t_2 & \nu^{(j)}(|\mathbf{t}|) + \frac{\nu^{(j)\prime}(|\mathbf{t}|)}{|\mathbf{t}|}t_2^2 \end{pmatrix}. \quad (9)$$

Obviously, we have $Q_j = \text{diag}(\nu_1^{(j)}, \nu_1^{(j)})$ for not ferromagnetic materials. For ferromagnetic materials, there are values of the magnetic induction $B = |\mathbf{t}|$ for which $\nu^{(j)\prime}$ is large, in particular for intermediate values between the linear part

$$\nu^{(j)}(B) \equiv \nu_1^{(j)} \quad \forall B \leq B_1$$

and the asymptotic behaviour

$$\nu^{(j)}(B) \longrightarrow \nu_\infty = 1/\mu_0 \quad \text{for} \quad B\to\infty,$$

since $\nu^{(j)}(B)$ is a monotone increasing function [18]. Then the relation

$$|\mathbf{t}|\nu^{(j)\prime}(|\mathbf{t}|) \gg \nu^{(j)}(|\mathbf{t}|) \quad (10)$$

holds, and the Jacobi operator becomes the discretization of an anisotropic operator.

For the example $\mathbf{t} = (t_1, 0)^\top$, we get from (9) that $Q_j = \text{diag}(q_{11}, q_{22})$ with $q_{22} = \nu^{(j)}(|\mathbf{t}|)$ and $q_{11} = q_{22} + |\mathbf{t}|\nu^{(j)\prime}(|\mathbf{t}|) \gg q_{22}$ holds, and the anisotropy is obvious. The anisotropy is, of course, not restricted to that direction of $\mathbf{t} = \nabla\Phi_q\mathbf{v}_q$.

3 Parallel solvers for linear problems

The standard NODD data structure and the parallelization strategy are presented, e.g., in [10]. The linear equations (6) can be rewritten in the standard block form

$$J\begin{pmatrix}\mathbf{w}_C\\ \mathbf{w}_I\end{pmatrix} = \begin{pmatrix}J_C & J_{CI}\\ J_{IC} & J_I\end{pmatrix}\begin{pmatrix}\mathbf{w}_C\\ \mathbf{w}_I\end{pmatrix} = \begin{pmatrix}\mathbf{d}_C\\ \mathbf{d}_I\end{pmatrix}, \quad (11)$$

where indices "I" and "C" correspond to the nodes belonging to the interior of the subdomains Ω_i and to the coupling boundaries, respectively. Note that $J_I = \text{diag}\,(J_{I,i})_{i=1,2,\ldots,p}$ is a block-diagonal matrix.

DD preconditioned conjugate gradient method:
The parallel CG algorithm with DD preconditioning for solving the systems (11) can be implemented in a standard way, see [10, 22]. It runs completely in parallel with the

exception of the two scalar products, and the preconditioning. The DD preconditioner for J, the matrix C with

$$C = \begin{pmatrix} I_C & J_{CI}B_I^{-T} \\ O & I_I \end{pmatrix} \begin{pmatrix} C_C & O \\ O & C_I \end{pmatrix} \begin{pmatrix} I_C & O \\ B_I^{-1}J_{IC} & I_I \end{pmatrix} , \tag{12}$$

contains three components, i.e., the preconditioners C_C and $C_I = \operatorname{diag}(C_{I,i})_{i=1,2,\ldots,p}$, and the regular matrix $B_I = \operatorname{diag}(B_{I,i})_{i=1,2,\ldots,p}$ defining the basis transformation, which can be adapted to the matrix J in a suitable way [10].

Here, we choose a symmetric multigrid cycle, in our applications a V-cycle (V11) with one pre- and one postsmoothing step of Gauss-Seidel type, in the symmetric Multiplicative Schwarz Method [10] for C_I, and B_I is implicitly defined by hierarchical extension (formally $E_{IC} = -B_I^{-1}J_{IC}$) [15]. We apply a Schur complement preconditioner C_C following Bramble/Pasciak/Schatz [2], which uses the idea of Dryja [6] on the coupling boundaries and a global crosspoint system (BPS-D), or a Bramble/Pasciak/Xu [29, 3] type Schur complement preconditioner together with a global crosspoint system (S-BPX).

Spectral equivalence between J and C has been proved in [10]. Together with the results of [2, 3, 15, 17, 29] we can prove that the numerical effort spent for one Newton step on grid q is at most of order $\mathcal{O}(N_q \ln\ln h_q^{-1} \ln\varepsilon_{\text{lin}}^{-1})$ in the (S-BPX) case, i.e. almost optimal. In the (BPS-D) case we have to add a factor $\ln h_q^{-1}$. Here h_q denotes the discretization parameter, such that $N_q = \mathcal{O}(h_q^{-2})$. We refer to [20, 22] for details.

Additionally, we tested a DD-PCG method with a BPX preconditioner (BPX-C_I) instead of the multigrid cycle for the operator C_I.

Global multigrid method:
Using the NODD data structure, the interpolation and restriction procedures do not need any communication. We developed parallel Gauss-Seidel type smoothers and Jacobi smoothers which require the same communication effort, i.e. a data exchange of the order $\mathcal{O}(N_q^{0.5})$ between the processors, within each smoothing step. The Gauss-Seidel smoother executes the smoothing first for the coupling nodes ("C"), and then for the inner nodes ("I"), i.e., "forward", or in the reverse order, "backward" (see [24]). In our test examples we perform two pre- and two postsmoothing steps on each level. We utilize parallelized preconditioned conjugate gradient methods with a (BPS-D) preconditioner 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. The parallel multigrid algorithm is described in detail in [21, 24].

Obviously, the GMG method can serve as a preconditioner in the CG algorithm, too [23].

Parallel global BPX method:
We realize the parallel CG method with BPX preconditioner [3] including a multilevel diagonal scaling [30] in a similar manner. In our implementation, the data of different levels is exchanged together, i.e., the number of data exchange steps (the startup time) is independent of the number of levels. Additionally, we need a coarse-grid solver which coincides completely with that of the GMG method.

4 Numerical results

4.1 Implementation

The complete algorithms are implemented in the parallel code FEM⊗BEM [12, 13, 24]. Thus, all parts of the parallel nested Newton algorithm, such as the grid generation, the matrix generation and representation, and the defect computation, are identical for all solving methods. We tested the algorithms on the parallel system GC-Power Plus (with maximal 64 processors Power PC 601) with the operating system Parix. In the following we describe our choice of components and parameters in the solvers for linear problems.

Initially, the linear problems arising in the Newton method are solved with a relative accuracy of $\varepsilon_{\text{lin}} = 0.01$. This parameter ε_{lin} can be adapted to the quadratic convergence speed of the Newton method in later Newton iterations [17].

Within the (GMG) solver, we used a V-cycle with 2 pre- and 2 postsmoothing steps of the parallelized Gauss-Seidel type. With respect to the coarse grid solver, a preconditioned Schur complement CG solver turned out to be sufficient where best results have been obtained with a relative accuracy of 0.1 in the multilevel case ($q \geq 2$).

In the GMG-PCG method, we apply one V-cycle as a preconditioner.

To be sure, we demand an error reduction by the relative accuracy $\varepsilon = 10^{-6}$ on the finest grid, therefore, we apply 4 or 5 Newton iterations on the finest grid. From the nested iteration we can expect that for obtaining an approximate solution which differs from the exact solution in the order of the discretization error, two Newton iterations are sufficient, i.e., we can nearly halve the processing time presented in the tables.

4.2 Direct current motor - two examples on different numbers of processors

A technical direct current (DC) motor which is excited by permanent magnets serves as an example of practical interest. The machine has a diameter of 50 mm, it is described in detail in [17]. The motor contains two different ferromagnetic materials; the rotor consists of dynamo sheet, the case is made from rolled steel. From a description of the machine, we created two calculation examples. We decomposed the domain Ω into 32 and 64 subdomains by means of the preprocessing tool ADDPRE (Adaptive DD Preprocessor, see [9, 13]). The decompositions have been created independently of each other. We cannot expect relations between them such as, e.g., that two of the 64 subdomains form one of the 32 subdomains. Figure 1 shows both decompositions.

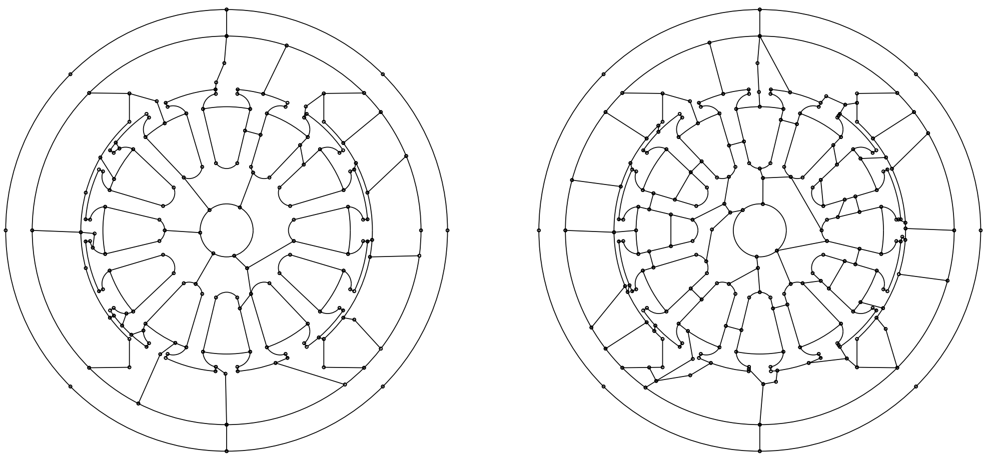

Figure 1: Domain decomposition into 32 (left) and 64 (right) subdomains (DC machine)

In Table 1, we present the iteration count for the Newton algorithm as well as for the different linear solvers for both examples. Figure 2 shows the level lines for the DC motor.

Table 1: Nested Newton algorithm for the DC machine

Solver	DD-PCG (V11, S-BPX)	DD-PCG (BPX-C_I, S-BPX)	GMG	GMG -PCG	GBPX -PCG	Example
Newton iter. 1st grid	6	6	6	6	6	32 sub-domains, 32 pro-cessors, 364 637 unknowns
Newton iter. 2nd grid	2	2	2	2	2	
CG/MG iter. 2nd grid	16,27	17,18	2,2	2,2	14,11	
Newton iter. 3rd grid	2	2	2	2	2	
CG/MG iter. 3rd grid	17,17	17,22	2,2	2,2	13,11	
Newton iter. 4th grid	2	2	2	2	2	
CG/MG iter. 4th grid	16,22	20,25	2,3	2,2	14,15	
Newton iter. 5th grid	4	4	4	4	4	
CG/MG iter. 5th grid	16,23, 17,16	21,30, 23,19	2,4, 2,2	2,3, 2,2	14,17, 17,14	
Newton iter. 1st grid	6	6	6	6	6	64 sub-domains 64 pro-cessors, 688 892 unknowns
Newton iter. 2nd grid	2	2	2	2	2	
CG/MG iter. 2nd grid	18,18	19,22	2,2	2,2	9,7	
Newton iter. 3rd grid	2	2	2	2	2	
CG/MG iter. 3rd grid	18,21	20,28	2,6	2,3	11,13	
Newton iter. 4th grid	2	2	2	2	2	
CG/MG iter. 4th grid	19,31	24,38	3,10	3,4	15,22	
Newton iter. 5th grid	3	4	3	3	3	
CG/MG iter. 5th grid	23, 37,31	28,48, 43,52	3, 10,8	3, 6,5	17, 37,35	

relative accuracy $\varepsilon = 10^{-6}$ on the finest grid

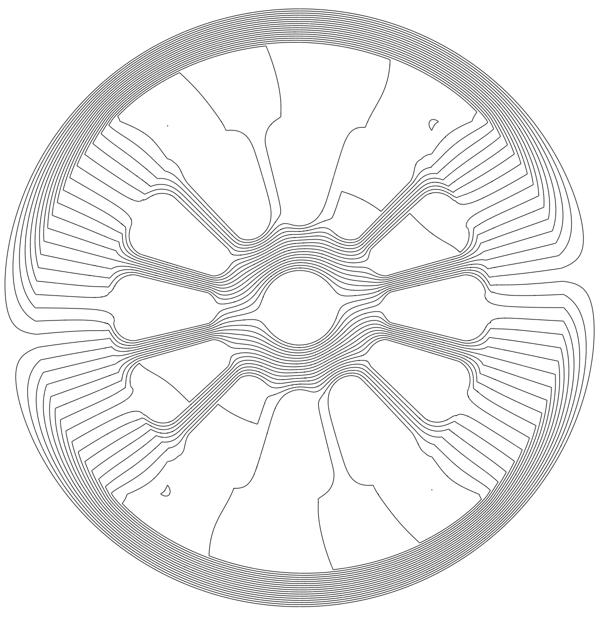

Figure 2: Level lines for the direct current machine

In Table 2 we compare the processing time for five different algorithms for both examples. From the total time, we calculated the Relative Scaled Efficiency defined in Section 1.

Table 2: Performance Comparison for the DC machine

Solver	DD-PCG (V11, S-BPX)	DD-PCG (BPX-C_I, S-BPX)	GMG	GMG -PCG	GBPX -PCG	Example
Time (system generation)	18.3	18.3	17.0	17.0	17.3	32 proc.,
Time (solver)	92.9	141.4	36.1	42.6	100.4	364 637
Total time	111.2	159.7	53.1	59.6	117.7	unknowns
Time (system generation)	17.9	20.0	17.1	17.2	17.2	64 proc.,
Time (solver)	150.0	302.8	110.5	93.3	205.4	688 892
Total time	167.9	322.8	127.6	110.5	222.6	unknowns
Relative Scale-up $S(32,64)$	1.25	0.93	0.79	1.02	1.00	
Relative Scaled Efficiency $E(32,64)$	0.63	0.47	0.39	0.51	0.50	

Time in seconds, GC-Power Plus, relative accuracy $\varepsilon = 10^{-6}$

We find out that the GMG-PCG method and GMG have the best performance. This is due to the excellent convergence properties of the multigrid iteration that keep the number of iterations small, whereas DD-PCG and GBPX-PCG need more iterations to achieve the relative accuracy $\varepsilon_{\text{lin}} = 0.01$.

The Relative Scaled Efficiency ranges between 0.39 and 0.63 for all algorithms. The reason for this relatively "worse" scaled efficiency compared with model problems [21] is that the linear solvers need more iterations for the 64 subdomain example than for the 32 subdomain one. This is due to "worse" spectral equivalence constants. The latter is caused by the geometric properties of the automatically generated decomposition and discretization, and by the $\ln h_q^{-1}$ term in the estimates. We notice that the DD-PCG method with (V11, S-BPX) shows the best scalability, although GMG (resp. GMG-PCG) solves faster. We remark that the system generation is almost ideally parallelizable.

Although the ratio of anisotropy, i.e. the quotient of the eigenvalues of the matrix Q_j (cf. (9)), is about 10 in some finite elements, the anisotropy does not affect the convergence of the GMG method in this example.

4.3 Induction machine - an example with high magnetic saturation

The second example of technical interest, an induction machine (asynchronous motor), is a more challenging one due to its very complicated interior geometry and the stronger influence of saturation (i.e., a stronger nonlinearity). Indeed, we have an air gap of 0.2 mm where the coefficient has a jump by a factor of more than 1000, whereas the machine has a diameter of 45 mm. The electro-magnetic field is simulated for a state with currents in the stator, but no rotor currents. We present the automatically generated decomposition of the cross-section of the machine into 64 subdomains and level lines in Figure 3, performance results in Table 3.

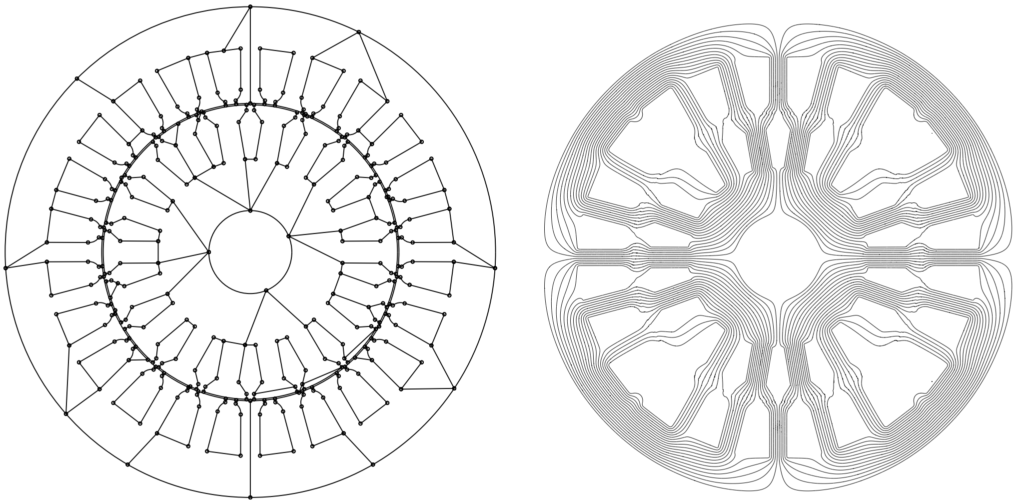

Figure 3: Domain decomposition and level lines for the induction machine

Note that in Table 3 for the (GMG) with 2 Gauss-Seidel "backward" pre-smoothing and 2 "backward" post-smoothing steps, (2b,2b), we achieved the best performance of all solvers. But, for the same number of "forward" Gauss-Seidel steps, (2f,2f), the multigrid solver on the levels $q = 3, 4, 5$ even did not converge.

Table 3: Performance for the induction machine

Solver	DD-PCG (V11,BPS-D)	GMG (2b,2b)	GMG-PCG (2b,2b)	GBPX -PCG
Newton iterations 1st grid	11	11	11	11
Newton iterations 2nd grid	2	2	2	2
CG/MG iterations 2nd grid	43,55	2,2	2,2	13,11
Newton iterations 3rd grid	2	2	2	2
CG/MG iterations 3rd grid	42,30	2,2	3,3	15,18
Newton iterations 4th grid	2	2	2	2
CG/MG iterations 4th grid	13,81	2,2	3,3	16,21
Newton iterations 5th grid	5	5	5	5
CG/MG iterations 5th grid	36,61,60,74,61	2,2,4,8,8	3,4,3,4,5	17,29,35,39,34
Time (system generation)	49.5	48.2	48.3	50.2
Time (solver)	550.2	295.9	347.2	848.8
Total time	599.7	344.1	395.5	899.0

Time in seconds, GC-Power Plus, 64 processors (subdomains); 5 grids, 549 091 unknowns, relative accuracy $\varepsilon = 10^{-6}$, current density S

We discovered that in large areas of the machine the ratio of anisotropy, i.e. the quotient of the eigenvalues of the matrix Q_j (cf. (9)), is between 17 and 18. We conclude from (9) that the dominance direction of the anisotropy is determined by the direction of $\mathbf{t}$. Thus, it depends on the direction of ∇u_q^j for the current approximate solution u_q^j. The direction of ∇u_q^j is perpendicular to the direction of the magnetic induction $\vec{\mathrm{B}}_q^j$. It differs for different parts of a machine, and it cannot be predicted a priori for other than model problems. Because of this anisotropy the Gauss-Seidel iteration may be a poor smoother [16]. The application of ILU smoothers or line smoothers as described in [1, 26] for problems in rectangular domains could lead to better convergence properties of the GMG solver, but it seems to be not clear up to now how to construct and parallelize them for the complicated inner geometry of the electric machines.

For Table 4, the current density is reduced by a factor 0.1. Consequently, the magnetic induction is lower. Various combinations of Gauss-Seidel smoothers perform well. Here, we have 4 Newton iterations on level 1, and 2 Newton iterations with 2 multigrid cycles on the levels 2, 3, and 4 for all 4 columns. The abbreviation "fb" denotes a "forward" followed by a "backward" step.

Table 4: Performance for the induction machine with reduced current density (GMG)

Smoother	(2f,2f)	(2b,2b)	(fb,fb)	(2f,2b)
Newton iterations 5th grid	4	4	5	5
MG iterations 5th grid	2,2,4,8	2,2,4,8	2,2,4,8,8	2,2,4,8,8

GC-Power Plus, 64 processors (subdomains); 5 grids, 549 091 unknowns, relative accuracy $\varepsilon = 10^{-6}$, current density $0.1 * S$

The application of the MG algorithm for defining a preconditioner in the PCG method gives a more robust solver. We observed that (GMG-PCG) *converge* for several variants of Gauss-Seidel smoothers, see Table 5.

Table 5: Performance for the induction machine (GMG-PCG)

Smoother	(2f,2b)	(fb,fb)	(2b,2b)	(2f,2f)
Newton iterations 1st grid	11	11	11	11
Newton iterations 2nd grid	2	2	2	2
CG iterations 2nd grid	3,3	3,3	2,2	2,2
Newton iterations 3rd grid	2	2	2	2
CG iterations 3rd grid	3,3	3,3	3,3	3,3
Newton iterations 4th grid	2	2	2	2
CG iterations 4th grid	3,3	3,4	3,3	3,3
Newton iterations 5th grid	5	5	5	5
CG iterations 5th grid	3,4,4,5,3	3,4,5,5,5	3,4,3,4,5	3,4,3,4,4

GC-Power Plus, 64 processors (subdomains); 5 grids, 549 091 unknowns, relative accuracy $\varepsilon = 10^{-6}$, current density S

4.4 A nonlinear test problem from continuum mechanics

This example demonstrates the application of the algorithms to a quite different nonlinear problem.

Consider the problem in $\Omega = (0,1) \times (0,1) \subset \mathbf{R}^2$:
Find $u \in V = H_0^1(\Omega)$ *such that*

$$\int_\Omega \psi(|\nabla u|)\ \nabla^\top u\ \nabla v\ dx = \int_\Omega v\ dx \quad \forall v \in V \tag{13}$$

where

$$\psi(z) = \begin{cases} 2 & \text{if } z^2 \leq 0.008 \\ 1 & \text{if } z^2 \geq 0.032 \\ \sqrt{0.032}/z & \text{else.} \end{cases} \tag{14}$$

The problem arises in the theory of elasticity (see [25] and [8]). We want to construct an infinitely long elastic bar of given cross–section Ω with maximal torsional rigidity from two different linearly elastic materials of given shear moduli. The proportions of these materials are prescribed. The problem has nearly the same form as (1) but it is not strongly monotone. Further, ψ' is not continuous.

Table 6: Performance for the shape design problem

Solver	DD-PCG (V11, S-BPX)	DD-PCG (BPX-C_I, S-BPX)	GMG -PCG	GBPX -PCG	Example
Time (system)	26.0	25.9	25.9	25.7	16 processors
Time (solver)	178.4	247.5	64.8	154.9	5 grids
Total time	204.4	273.4	90.7	180.6	387 985 nodes
Time (system)	28.2	25.6	28.2	25.5	64 processors
Time (solver)	412.0	289.5	138.4	337.6	5 grids
Total time	440.2	315.1	166.6	363.1	1 554 209 nodes
Relative Scale-up $S(16,64)$	1.86	3.48	2.18	1.99	
Relative Scaled Efficiency $E(16,64)$	0.47	0.87	0.55	0.50	
Time (system)	9.3	8.8	8.1	8.8	64 processors
Time (solver)	109.7	112.7	88.4	216.3	4 grids
Total time	119.0	121.5	96.5	225.1	388 001 nodes
Relative Speedup $\tilde{S}(16,64)$	1.72	2.25	< 1	< 1	
Relative Efficiency $\tilde{E}(16,64)$	0.43	0.56			

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

Our first aim is to demonstrate the performance of the algorithms. Further, we want to check whether in the "homogenized region", i.e., the region where $0.008 \leq |\nabla u_l|^2 \leq 0.032$ (here the two materials have to be "mixed"), the level lines are circular sectors with identical radius, as it has been predicted theoretically for the solution u in [25].

The square domain Ω is divided into 4×4 resp. 8×8 square subdomains, each of them is triangulated automatically by the parallel program FEM∞BEM [12, 13, 24]. We present the performance results in Table 6. The Relative Scale-up and the Relative Efficiency (see Section 1) have been calculated from the total time for the examples with 5 grids. Further, we calculated Relative Speedup and Relative Efficiency on the basis of the examples with circa 388 000 unknowns.

We note that for this example the Newton-GMG method does not converge, nor for Gauss-Seidel smoothers, nor for Jacobi smoothers with up to 8 smoothing steps. On the other hand, the combination of multigrid with a CG method leads to a solver which is both fast, and robust.

The Relative Scaled Efficiency is about one half as we had expected it for this hard test problem. For DD-PCG with BPX-C_I preconditioner, we may calculate a higher Relative Scaled Efficiency, but only a poor performance. The Relative Speedup results show that it makes no sense to distribute the problem onto too much processors.

We show the "homogenized region" on the left-hand side of Figure 4 and the level lines in this region on the right-hand side. We observe that the "homogenized region" has the typical shape, and the level lines can be recognized as circular sectors.

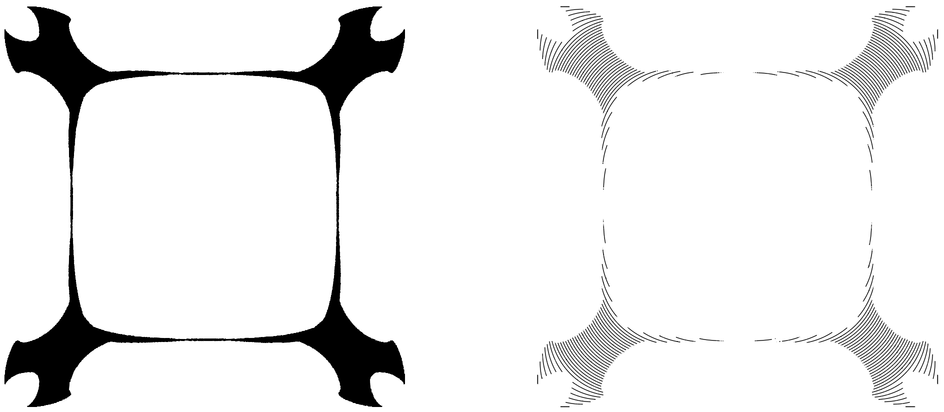

Figure 4: Homogenized region and level lines for the shape design problem

5 Conclusions

We have tested several parallel iterative solvers for the linearized problems arising in a nested Newton method for real-life nonlinear problems. We have found that the DD-PCG method is able to solve the problems, but the convergence can be slow. Replacing the V11 multigrid preconditioner in C_I by a simpler BPX led to more CG iterations, and to higher computing time, in most cases.

The GMG method can converge faster, but its components have to be adapted well to the actual problem. Indeed, for anisotropic linear problems, we have to apply other smoothers than Gauss-Seidel ones, e.g. ILU smoothers [16, 1], but it seems to be not yet clear how to construct them for the complicated interior geometry of an electric machine. For the shape-design problem, a multigrid method had been successfully applied in a quite coarse discretization [17]. For a fine discretization, the difficulties with multigrid algorithms could be overcome in [25] by nonstandard "tricks" only. So we are not surprised that our standard GMG method does not converge.

In [17, 23], we had already found that the use of multigrid as a preconditioner essentially improves the convergence, in particular if the multigrid method converges slowly. Our parallel computations demonstrate that the combination with the PCG algorithm makes multigrid much more robust, even in that cases in which the pure multigrid method does not converge.

In some cases, we get a higher scalability for the DD preconditioner than for the GMG one, but in view of the computing time we may prefer GMG-PCG for solving technical problems on massively parallel computer architectures.

The global BPX preconditioner yields a robust method. It may be a good choice if a black-box solver is required, i.e., the components of the solver cannot be adapted well to the problem.

All the algorithms mentioned above are parallelized using the NODD data structure. The communication overhead may differ between the solver types, but it is in the same order of magnitude for all of them, namely one order lower than the problem itself.

The latter makes the NODD data structure attractive for implementing robust multilevel solvers in three dimensions, too.

References

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

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

[3] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comput.*, 55(191):1–22, 1990.

[4] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[5] P. Deuflhard. Global inexact Newton methods for very large scale nonlinear problems. *IMPACT Comput. Sci. Eng.*, 3:366–393, 1991.

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

[7] M. Dryja and O. B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, editors, *Domain decomposition methods for partial differential equations*, pages 3–21, Philadelphia, 1990. SIAM. Proc. of the 3rd International Symposium, Houston, 1989.

[8] J. Goodman, V. R. Kohn, and L. Reyna. Numerical study of a relaxed variational problem from optimal design. *Comput. Methods Appl. Mech. Eng.*, 57:107–127, 1986.

[9] Mario Goppold, Gundolf Haase, Bodo Heise, and Michael Kuhn. Preprocessing in BE/FE domain decomposition methods. Technical Report 96-2, Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerische Mathematik und Optimierung, 1996.

[10] 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.

[11] Gundolf Haase. *Die nichtüberlappende Gebietszerlegungsmethode zur Parallelisierung und Vorkonditionierung iterativer Verfahren.* PhD thesis, Fakultät Mathematik und Naturwissenschaften, TU Chemnitz–Zwickau, 1993.

[12] Gundolf Haase, Bodo Heise, Michael Jung, and Michael Kuhn. FEM$\otimes$BEM - a parallel solver for linear and nonlinear coupled FE/BE-equations. DFG-Schwerpunkt "Randelementmethoden", Report 94-16, University Stuttgart, 1994.

[13] Gundolf Haase, Bodo Heise, Michael Kuhn, and Ulrich Langer. Adaptive domain decomposition methods for finite and boundary element equations. In Wolfgang L. Wendland, editor, *Boundary Element Topics*, pages 111–137, Berlin, 1996. Springer. Proc. of the Final Conference of the Priority Research Programme *Boundary Element Methods 1989–1995* of the German Research Foundation (DFG), October 2-4, 1995 in Stuttgart. To appear.

[14] Gundolf Haase, Ulrich Langer, and Arnd Meyer. Domain decomposition preconditioners with inexact subdomain solvers. *J. of Num. Lin. Alg. with Appl.*, 1:27–42, 1992.

[15] Gundolf Haase, Ulrich Langer, Arnd 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.

[16] W. Hackbusch. *Multi–Grid methods and applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer–Verlag, Berlin, 1985.

[17] Bodo Heise. Nonlinear field calculations with multigrid-Newton methods. *IMPACT Comput. Sci. Eng.*, 5:75–110, 1993.

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

[19] Bodo Heise. Nonlinear simulation of electromagnetic fields with domain decomposition methods on MIMD parallel computers. *J. Comput. Appl. Math.*, 63:373–381, 1995.

[20] Bodo Heise. Nonlinear field simulation with FE domain decomposition methods on massively parallel computers. *Surv. Math. Ind.*, 1996. To appear.

[21] Bodo Heise and Michael Jung. Comparison of parallel solvers for nonlinear elliptic problems based on domain decomposition ideas. Report 494, Johannes Kepler University Linz, Institute of Mathematics, 1995. Submitted for publication.

[22] Bodo Heise and Michael Kuhn. Parallel solvers for linear and nonlinear exterior magnetic field problems based upon coupled FE/BE formulations. *Computing*, 56(3):237–258, 1996.

[23] 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.

[24] Michael Jung. On the parallelization of multi-grid methods using a non-overlapping domain decomposition data structure. *Appl. Numer. Math.*, 1997. To appear.

[25] B. Kawohl, J. Stara, and G. Wittum. Analysis and numerical studies of a problem of shape design. *Arch. Ration. Mech. Anal.*, 114:349–363, 1991.

[26] Oliver A. McBryan, Paul O. Frederickson, Johannes Linden, Anton Schüller, Karl Solchenbach, Klaus Stüben, Clemens-August Thole, and Ulrich Trottenberg. Multigrid methods on parallel computers — a survey of recent developments. *IMPACT Comput. Sci. Eng.*, 3:1–75, 1991.

[27] S. V. Nepomnyaschikh. Domain decomposition and multilevel techniques for preconditioning operators. Preprint SPC 95_30, Technical University Chemnitz–Zwickau, Faculty for Mathematics, 1995.

[28] B. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, 1996.

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

[30] X. Zhang. Multilevel Schwarz methods. *Numer. Math.*, 63:521–539, 1992.