12th International Conference on Domain Decomposition Methods
Editors: Tony Chan, Takashi Kako, Hideo Kawarada, Olivier Pironneau, ©2001 DDM.org

37. Parallel 3D Maxwell Solvers based on Domain Decomposition Data Distribution

G. Haase[1], M. Kuhn[2], U. Langer[3]

Introduction

The most efficient solvers for finite element (fe) equations are certainly multigrid, or multilevel methods, and domain decomposition methods using local multigrid solvers. Typically, the multigrid convergence rate is independent of the mesh size parameter, and the arithmetical complexity grows linearly with the number of unknowns. However, the standard multigrid algorithms fail for the Maxwell finite element equations in the sense that the convergence rate deteriorates as the mesh-size decreases. To overcome this drawback, R. Hiptmair proposed to modify the smoothing iteration by adding a smoothing step in the discrete potential space [Hip99]. Similarly, D. Arnold, R. Falk and R. Winther suggested a special block smoother that has the same effect [AFW00].

The parallelization of these or, more precisely, of appropriately modified multigrid solvers is certainly the only principle way to enhance the efficiency of these algorithms. Due to the peculiarities of the multigrid methods for the Maxwell equations, the parallelization is not straightforward. In this paper, we propose a unified approach to the parallelization of multigrid methods and domain decomposition methods. In order to develop a basic parallel Maxwell solver that can be used for more advanced problems as basic module, it is sufficient to consider the magnetostatic case. In the magnetostatic case, the Maxwell equation can be reduced to the curl-curl–equation that is not uniquely solvable because of the large kernel of the curl-operator (potential fields). In practice, a gauging condition is imposed in order to pick out a unique solution. The so-called Coulomb gauging aims at a divergence-free solution (vector potential). The weak formulation of the curl-curl–equation and the gauging condition together with a clever regularization leads to a regularized mixed variational formulation of the magnetostatic Maxwell equations in $\mathrm{H}_0(\mathbf{curl}) \times \mathrm{H}_0^1(\Omega)$ that has a unique solution. The discretization by the Nédélec and Lagrange finite elements results in a large, sparse, symmetric, but indefinite system of finite element equations. Eliminating the Lagrange multiplier from the mixed finite element equations, we arrive at a symmetric and positive definite (spd) problem that can be solved by some parallel multigrid preconditioned conjugate gradient (pcg) method. More precisely, this pcg solver contains a standard scaled Laplace multigrid regularizer in the regularization part and a special multigrid preconditioner for the regularized Nédélec finite element equations that we want to solve. (see second section). The parallelization of the

[1]Johannes Kepler University Linz, Institute of Analysis and Computational Mathematics, ghaase@numa.uni-linz.ac.at
[2]Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing", kuhn@sfb013.uni-linz.ac.at
[3]Johannes Kepler University Linz, Institute of Analysis and Computational Mathematics, ulanger@numa.uni-linz.ac.at

pcg algorithm, the Laplace multigrid regularizer and the multigrid preconditioner are based on a unified domain decomposition (dd) data distribution concept that will be briefly described in the following two sections. From the parallelization point of view, we prefer Hiptmair's multigrid method with some modifications for the construction of the special multigrid preconditioner. We also propose a concept for coupling finite elements with boundary elements in 3D. As in 2D, a really efficient parallel solver should be based on a hybrid parallelization concept using some Dirichlet dd preconditioner the components of which are a dd parallelized global multigrid preconditioner for the finite element part and algebraically parallelized components for the boundary element parts. The final part contains some results of our numerical experiments on a parallel machine with distributed memory that show the high efficiency of our approach for a real-life application.

3D Magnetostatic Field Problems

The magnetostatic equations, in which we are interested throughout the paper, can be rewritten as

$$\operatorname{curl}(H) = J, \;\; H = \nu B, \;\; \operatorname{div}(B) = 0, \tag{1}$$

where H and B denote the magnetic field intensity and the magnetic flux density, respectively. The permeability μ ($\nu := 1/\mu \geq \nu_{min} > 0$) and current density J are given. Furthermore, we note that the current density J is physically divergence-free, i.e., $\operatorname{div}(J) = 0$. Theoretically, the computational domain Ω coincides with the space $\mathbb{R}^3$ in any case. The behavior of the magnetic field at infinity is described by radiation conditions. In practice, one may often simplify the problem by considering a bounded, simply connected computational domain $\Omega \subset \mathbb{R}^3$ with Lipschitz boundary $\Gamma = \partial\Omega$ and by replacing the radiation condition by the boundary condition

$$B \cdot n = 0 \quad \text{on } \partial\Omega, \tag{2}$$

where n stands for the unit outward normal with respect to $\partial\Omega$. Introducing some vector potential u for the B-field $B = \operatorname{curl}(u)$ and taking into account the Coulomb gauging condition $\operatorname{div}(u) = 0$ ensuring uniqueness, we arrive at the following mixed variational formulation that is fundamental for our approach to the numerical solution of the magnetostatic Maxwell equations (1):
Find $(u,p) \in X \times M := H_0\,(\operatorname{curl},\Omega) \times H_0^1(\Omega)$ such that

$$\begin{aligned} a(u,v) + b(v,p) &= \langle f, v\rangle \quad \forall\, v \in H_0\,(\operatorname{curl},\Omega), && (3)\\ b(u,q) &= 0 \qquad\quad \forall\, q \in H_0^1(\Omega), && (4)\end{aligned}$$

where $a(u,v) := \int_\Omega \nu \operatorname{curl}(u)\cdot\operatorname{curl}(v)\,dx$, $b(v,p) := \int_\Omega v\cdot\nabla p\,dx$, and $\langle f, v\rangle := \int_\Omega J\cdot v\,dx$. Now it is not difficult to conclude from the Brezzi-Babuška theory that the mixed variational problem (3) - (4) has a unique solution. Moreover, choosing $v = \nabla p \in H_0(\operatorname{curl},\Omega)$ in (3), we immediately observe that $p = 0$. This simple observation is crucial for our approach. Indeed, adding an arbitrary spd bilinear form $c(\cdot,\cdot) : M \times M \to \mathbb{R}^1$ to the second equation of our mixed variational problem (3) - (4) we arrive at the equivalent mixed variational problem: Find $(u, p) \in X \times M$ such that

$$a(u,v) + b(v,p) = \langle f, v\rangle \quad \forall\, v \in X, \tag{5}$$

$$b(u,q) - c(p,q) = 0 \qquad \forall\, q \in M. \tag{6}$$

Let now be $X_h := \mathcal{N}_h^1 \subset X$ and $M_h := \mathcal{S}_h^1 \subset M$ the lowest order edge element space (see [N86́]) and the space of piecewise linear nodal elements on a shape-regular tetrahedral triangularization of Ω with the mesh-width h, respectively [Cia78]. Then the mixed fe approximation to the regularized mixed variational problem (5) - (6) leads us to the following symmetric, but indefinite system

$$\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} \begin{pmatrix} \underline{u}_h \\ \underline{p}_h \end{pmatrix} = \begin{pmatrix} \underline{f}_h \\ 0 \end{pmatrix} \tag{7}$$

of linear finite element equations for defining the edge unknowns $\underline{u}_h$ and the nodal unknowns $\underline{p}_h$, where the matrices A, B, C and the first component $\underline{f}_h$ of the right-hand side are derived from the bilinear forms $a(\cdot,\cdot)$, $b(\cdot,\cdot)$, $c(\cdot,\cdot)$, and the linear form $\langle f, \cdot\rangle$, respectively.

Eliminating $\underline{p}_h = C^{-1}B\underline{u}_h$ from the second equation in (7) and inserting it into the first equation, we obtain the spd Schur complement system

$$G\underline{u}_h := (A + B^T C^{-1} B)\underline{u}_h = \underline{f}_h. \tag{8}$$

Let $\widetilde{C}$ be some spd matrix that is spectrally equivalent to C (briefly, $\widetilde{C} \approx C$). Then the original Schur complement system (8) is equivalent to the modified Schur complement system

$$\widetilde{G}\underline{u}_h := (A + B^T \widetilde{C}^{-1} B)\underline{u}_h = \underline{f}_h. \tag{9}$$

Instead of solving the symmetric, but indefinite system (7), we solve the spd modified Schur complement system (9).

Let us choose the spd bilinear form

$$c(p,q) := \frac{1}{\nu_{\min}} \int\limits_{\Omega} \nabla p \, \nabla q \, dx, \tag{10}$$

corresponding to the Laplace operator scaled by $1/\nu_{\min}$, and let us consider a spd preconditioner C_H for the spd matrix $H := A + \widetilde{M}$, where $\widetilde{M}$ is here the appropriately scaled mass matrix in X_h defined by $(\widetilde{M}\underline{u}_h, \underline{v}_h) := \nu_{\min} \int_\Omega u_h \, v_h \, dx$. The discrete LBB–condition and the spectral equivalence $C_H \approx H$ imply that $C_H \approx G \approx \widetilde{G}$ (see [Kuh98] for the detailed proof). Once a good preconditioner C_H and an appropriate regularizer $\widetilde{C}$ is available, we can solve the modified Schur complement system (9) by the pcg method. In practice, we choose the multigrid preconditioner $C_H := H(I - M_H)^{-1}$ and the multigrid regularizer $\widetilde{C} := C_C := C(I - M_C)^{-1}$, where M_H and M_C are the corresponding multigrid iteration operators with respect to H and C. Choosing appropriate symmetric multigrid cycles, we can now conclude from the results of [AFW00, Hac85, Hip99] that the pcg method is asymptotically optimal with respect to the operation count and to the memory demand [JLM$^+$89]. The numerical results obtained from the serial implementation of this algorithm confirm this statement [KLS00]. In this paper, we are interested in the parallel implementation of this algorithm. The parallelization of this algorithm is far from being straightforward because of the peculiarities connected with the multigrid regularizer C_C and with the special multigrid preconditioner C_H.

A Unified Data Distribution Concept

Vector and matrix types

We decompose $\overline{\Omega}$ in P non-overlapping subdomains $\overline{\Omega}_s$ which are discretized by a mesh $\tau_{h,s}$, such that the whole triangulation $\tau_h = \bigcup_{s=1}^{P} \tau_{h,s}$ of Ω is conform. The index set of the N_s unknowns in Ω_s is denoted by ω_s. The mapping of a vector $\underline{u} \in \mathbb{R}^N$ in global numbering onto a local vector $\underline{u}_s \in \mathbb{R}^{N_s}$ in subdomain $\overline{\Omega}_s$ $(s = \overline{1,P})$ is represented symbolically by subdomain connectivity matrices $\mathcal{A}_s$ of dimension $N_s \times N$ with entries $\mathcal{A}_s^{[i,j]} := 1$ if $j \in \omega$ is the global number of $i \in \omega_s$ and $\mathcal{A}_s^{[i,j]} := 0$ otherwise.
The index set of all those subdomains, an unknown $u^{[j]}$, $j \in \omega$ belongs to, is denoted by $\sigma^{[j]} := \{s \,|\, \exists i \in \omega_s : \mathcal{A}_s^{[i,j]} \neq 0\}$. We store the data of a vector component $u^{[i]}$ in the subdomain Ω_s if $s \in \sigma^{[i]}$.
There are two opportunities to store those components and finally that vector. A vector $\underline{\mathfrak{u}}$ is called an accumulated vector if each vector component $\mathfrak{u}^{[i]}$ is stored in all subdomains Ω_s, $s \in \sigma^{[i]}$ with its full value. The local vectors $\underline{\mathfrak{u}}_s$ can be represented as $\underline{\mathfrak{u}}_s := \mathcal{A}_s \cdot \underline{\mathfrak{u}}$. We name a vector $\underline{\mathsf{r}}$ as distributed vector if it is decomposed into local vectors $\underline{\mathsf{r}}_s$ such that $\underline{\mathsf{r}} = \sum_{s=1}^{P} \mathcal{A}_s^T \cdot \underline{\mathsf{r}}_s$ holds, i.e., all subdomains Ω_s, $s \in \sigma^{[i]}$ store only $\underline{\mathsf{r}}_s$ and possess a portion of the full vector value $\mathsf{r}^{[i]}$ which can be determined only by taking the sum. The conversion of a distributed vector $\underline{\mathsf{v}}$ into an accumulated vector $\underline{\mathfrak{w}}$ can be done by evaluating the sum above and restrict the result afterwards, i.e.,

$$\underline{\mathfrak{w}} \leftarrow \underline{\mathsf{v}} \qquad : \qquad \underline{\mathfrak{w}}_s := \mathcal{A}_s \cdot \underline{\mathfrak{w}} = \mathcal{A}_s \cdot \sum_{s=1}^{P} \mathcal{A}_s^T \cdot \underline{\mathsf{v}}_s \ . \tag{11}$$

With respect to an element-wise domain decomposition, the matrix defined by the bilinear form in (3) can also be stored in two ways. A matrix $\mathfrak{M}$ is called accumulated if its local restrictions $\mathfrak{M}_s$ possess the full entries of it, and we can write $\mathfrak{M}_s := \mathcal{A}_s \cdot \mathfrak{M} \cdot \mathcal{A}_s^T$. We call a matrix K distributed if we have locally stored matrices K_s such that $\mathsf{K} := \sum_{s=1}^{P} \mathcal{A}_s^T \cdot \mathsf{K}_s \cdot \mathcal{A}_s$ holds, i.e., each subdomain $\overline{\Omega}_s$ stores only a part of its full values. We obtain distributed system matrices K_s automatically in our approach.

Basic operations

The inner product of different type vectors requires one global reduce operation of the local inner products, for details see [Haa98, Haa99, HLM91]. The multiplication of a distributed matrix with an accumulated vector results in a distributed vector and its local realization $\underline{\mathsf{v}}_s = \mathsf{K}_s \cdot \underline{\mathfrak{w}}_s$ requires no communication at all:

$$\langle \underline{\mathfrak{w}}, \underline{\mathsf{r}} \rangle = \sum_{s=1}^{P} \langle \underline{\mathfrak{w}}_s, \underline{\mathsf{r}}_s \rangle \qquad \text{and} \qquad \mathsf{K} \cdot \underline{\mathfrak{w}} = \underline{\mathsf{v}} \ . \tag{12}$$

The situation changes if we use an accumulated matrix $\mathfrak{M}$. If the pattern of $\mathfrak{M}$ fulfills the condition

$$\forall i,j \in \omega : \qquad \sigma^{[i]} \not\subseteq \sigma^{[j]} \implies \mathfrak{M}^{[i,j]} = 0 \ , \tag{13}$$

then no communication is needed for the operations $\underline{\mathfrak{w}} = \mathfrak{M} \cdot \underline{\mathfrak{u}}$ and $\underline{\mathfrak{d}} = \mathfrak{M}^T \cdot \underline{\mathsf{r}}$, i.e., we performed locally $\underline{\mathfrak{w}}_s = \mathfrak{M}_s \cdot \underline{\mathfrak{u}}_s$ and $\underline{\mathfrak{d}}_s = \mathfrak{M}_s^T \cdot \underline{\mathsf{r}}_s$, $\forall s = \overline{1, P}$.

Basic algorithms

The operations (12) allow us already to formulate a parallel pcg algorithm for solving the matrix equation $Ku = f$ with a preconditioner C_K. Besides the inner products,

Algorithm 1 Parallel pcg method PCG($\mathsf{K}, \underline{\mathfrak{u}}, \underline{\mathsf{f}}, C_K$)

```
repeat
    $\underline{\mathsf{v}} \leftarrow \mathsf{K} \cdot \underline{\mathfrak{s}}$
    $\alpha \leftarrow \sigma / \langle \underline{\mathfrak{s}}, \underline{\mathsf{v}} \rangle$
    $\underline{\mathfrak{u}} \leftarrow \underline{\mathfrak{u}} + \alpha \cdot \underline{\mathfrak{s}}$
    $\underline{\mathsf{r}} \leftarrow \underline{\mathsf{r}} - \alpha \cdot \underline{\mathsf{v}}$
    $\underline{\mathfrak{w}} \Leftarrow C_K^{-1} \cdot \underline{\mathsf{r}}$
    $\sigma \leftarrow \langle \underline{\mathfrak{w}}, \underline{\mathsf{r}} \rangle$ , $\beta \leftarrow \sigma / \sigma_{\text{old}}$ , $\sigma_{\text{old}} \leftarrow \sigma$
    $\underline{\mathfrak{s}} \leftarrow \underline{\mathfrak{w}} + \beta \cdot \underline{\mathfrak{s}}$
until termination
```

only the preconditioning step $\underline{\mathfrak{w}} \Leftarrow C_K^{-1} \cdot \underline{\mathsf{r}}$ involves communication indicated by using $\Leftarrow$ instead of $\leftarrow$. In the case of $C_K = I$, i.e., no preconditioning, this step reduces to a type conversion (11) involving communication. We require that the communication costs for applying any other preconditioner C_K^{-1} are in the same range.

One possible choice for the preconditioner is $C_K^{-1} = (I - M_K)K^{-1}$, with M_K being the multigrid iteration operator for K. The parallel multigrid iteration is presented in Alg. 2, where ℓ denotes the level such that $\ell = 1$ stands for the coarsest grid. The algorithm needs a smoother SMOOTH with a good parallel performance, e.g., a block Jacobi smoother with Gauss-Seidel smoothing in blocks containing interior unknowns of the subdomains. Furthermore, the interpolation $\mathfrak{P}$ has to fulfill the pattern condition (13) and we take $\mathfrak{P}^T$ as restriction. The coarse grid system can be solved directly or by some iterative method similar to the PCG in Alg. 1. Despite the coarse grid solver, only the smoothing sweep requires communication.

Parallel Multigrid Maxwell Solver

We want to solve (9) by the pcg algorithm (Alg. 1) using a multigrid preconditioner (Alg. 2) for the realization of C_K^{-1}. In this section, we will discuss how these compo-

Algorithm 2 Parallel multigrid $\textsc{pmg}(\mathsf{K}, \underline{\mathfrak{u}}, \underline{\mathsf{f}}, \ell)$

```
if $\ell == 1$ then
    $\underline{\mathfrak{u}} \Leftarrow \textsc{Solve}\,(\sum_{s=1}^{P} \mathcal{A}_s^T \mathsf{K}_s \mathcal{A}_s \cdot \underline{\mathfrak{u}} \;=\; \underline{\mathsf{f}}\,)$
else
    $\widetilde{\underline{\mathfrak{u}}} \Leftarrow \textsc{Smooth}(\mathsf{K}, \underline{\mathfrak{u}}, \underline{\mathsf{f}})$
    $\underline{\mathsf{d}} \leftarrow \underline{\mathsf{f}} - \mathsf{K} \cdot \widetilde{\underline{\mathfrak{u}}}$
    $\underline{\mathsf{d}}^H \leftarrow \mathfrak{P}^T \cdot \underline{\mathsf{d}}$
    $\underline{\mathfrak{w}}^H \Leftarrow \textsc{pmg}(\mathsf{K}^H, \underline{\mathfrak{w}}^H \leftarrow 0, \underline{\mathsf{d}}^H, \ell - 1)$
    $\underline{\mathfrak{w}} \leftarrow \mathfrak{P} \cdot \underline{\mathfrak{w}}^H$
    $\underline{\mathfrak{u}} \Leftarrow \textsc{Smooth}^T(\mathsf{K}, \widetilde{\underline{\mathfrak{u}}} + \underline{\mathfrak{w}}, \underline{\mathsf{f}})$
end if
```

nents have to be adapted to the case of our Maxwell solver presented in the second section.

Our reduced primal formulation (9) has been derived from (7). As discussed in the previous section, the matrices are generated locally, such that the local components A_s, B_s, C_s are available. Denoting the subdomain connectivity matrices with respect to the spaces $\mathbb{X}_h := \mathbb{R}^{n_h}$ and $\mathbb{M}_h := \mathbb{R}^{m_h}$ by $\mathcal{A}_{\mathbb{X},s}$ and $\mathcal{A}_{\mathbb{M},s}$, respectively, we have the following relations:

$$\mathsf{A} = \sum_{s=1}^{P} \mathcal{A}_{\mathbb{X},s} \mathsf{A}_s \mathcal{A}_{\mathbb{X},s}^T, \quad \mathsf{B} = \sum_{s=1}^{P} \mathcal{A}_{\mathbb{M},s} \mathsf{B}_s \mathcal{A}_{\mathbb{X},s}^T, \quad \mathsf{C} = \sum_{s=1}^{P} \mathcal{A}_{\mathbb{M},s} \mathsf{C}_s \mathcal{A}_{\mathbb{M},s}^T.$$

The system matrix in (9) is defined by $\widetilde{\mathsf{G}} := \mathsf{A} + \mathsf{B}^\mathsf{T} \widetilde{C}^{-1} \mathsf{B}$, where $\widetilde{C}$ is a preconditioner for C. In order to apply $\textsc{pcg}(\widetilde{\mathsf{G}}, \underline{\mathfrak{u}}, \underline{\mathsf{f}}, C_{\widetilde{G}})$ we explain in Alg. 3 how the matrix-by-vector operation is defined for the distributed matrix $\widetilde{\mathsf{G}}$. Hereby, the required operation

Algorithm 3 The operation $\underline{\mathsf{v}} \Leftarrow \widetilde{\mathsf{G}} \cdot \underline{\mathfrak{s}}$.

```
$\underline{\mathsf{q}} \leftarrow \mathsf{B} \cdot \underline{\mathfrak{s}}$
$\underline{\mathfrak{p}} \Leftarrow \textsc{pmg}(\mathsf{C}, \underline{\mathfrak{p}}, \underline{\mathsf{q}}, \ell)$
$\underline{\mathsf{v}} \leftarrow \mathsf{A} \cdot \underline{\mathfrak{s}} + \mathsf{B}^\mathsf{T} \cdot \underline{\mathfrak{p}}$
```

$\widetilde{C}^{-1}$ is being realized by one multigrid iteration step $\textsc{pmg}(\mathsf{C}, \underline{\mathfrak{p}}, \underline{\mathsf{q}}, \ell)$ in the space $\mathbb{M}_h$. Although the matrix $\widetilde{\mathsf{G}}$ is distributed, the corresponding matrix-by-vector operation requires as many communications as one multigrid iteration step for C in $\mathbb{M}_h$.

Furthermore, the operation C_K^{-1} in Alg. 1 is now realized by one iteration step of $\textsc{Hybridpmg}(\widetilde{\mathsf{H}}, \mathsf{C}, \underline{\mathfrak{u}}, \underline{\mathsf{f}}, \ell)$ defined in Alg. 4. Comparing Alg. 4 and Alg. 2 we observe that only the smoother has to be adapted to our special application.

In particular we use a hybrid smoother as proposed in [Hip99] which is suitable for parallelization. As in [Hip99], we introduce the lifting operator $\mathfrak{L} : \; \mathbb{X}_h \to \mathbb{M}_h$ where $\mathfrak{L}^{[i,j]} := -1$, $\mathfrak{L}^{[i,k]} := 1$ if the oriented edge with the unknown index i in $\mathbb{X}_h$ connects the two unknowns with the indices j and k in $\mathbb{M}_h$. Otherwise we have $\mathfrak{L}^{[i,j]} := 0$. We observe that $\mathfrak{L}^T$ satisfies the pattern condition (13). Now Alg. 5 is the correct definition of the parallel hybrid smoother $\textsc{hybridsmooth}(\mathsf{H}, \mathsf{C}, \underline{\mathfrak{u}}, \underline{\mathsf{f}})$. Note, using the

Algorithm 4 Parallel hybrid multigrid HYBRIDPMG$(\mathsf{H}, \mathsf{C}, \underline{\mathfrak{u}}, \underline{\mathsf{f}}, \ell)$

```
if $\ell == 1$ then
    $\underline{\mathfrak{u}} \Leftarrow$ SOLVE $( \sum_{s=1}^{P} \mathcal{A}_s^T \mathsf{H}_s \mathcal{A}_s \cdot \underline{\mathfrak{u}} \ = \ \underline{\mathsf{f}} )$
else
    $\widetilde{\underline{\mathfrak{u}}} \Leftarrow$ HYBRIDSMOOTH$(\mathsf{H}, \mathsf{C}, \underline{\mathfrak{u}}, \underline{\mathsf{f}})$
    $\underline{\mathsf{d}} \leftarrow \underline{\mathsf{f}} - \mathsf{H} \cdot \widetilde{\underline{\mathfrak{u}}}$
    $\underline{\mathsf{d}}^H \leftarrow \mathfrak{P}^T \cdot \underline{\mathsf{d}}$
    $\underline{\mathfrak{w}}^H \leftarrow 0$
    $\widetilde{\underline{\mathfrak{w}}}^H \Leftarrow$ HYBRIDPMG$(\mathsf{H}^H, \mathsf{C}^H, \underline{\mathfrak{w}}^H, \underline{\mathsf{d}}^H, \ell - 1)$
    $\underline{\mathfrak{w}} \leftarrow \mathfrak{P} \cdot \widetilde{\underline{\mathfrak{w}}}^H$
    $\widehat{\underline{\mathfrak{u}}} \leftarrow \widetilde{\underline{\mathfrak{u}}} + \underline{\mathfrak{w}}$
    $\underline{\mathfrak{u}} \Leftarrow$ HYBRIDSMOOTH$^T(\mathsf{H}, \mathsf{C}, \widehat{\underline{\mathfrak{u}}}, \underline{\mathsf{f}})$
end if
```

Algorithm 5 Parallel hybrid smoother HYBRIDSMOOTH$(\mathsf{H}, \mathsf{C}, \underline{\mathfrak{u}}, \underline{\mathsf{f}})$

```
$\widetilde{\underline{\mathfrak{u}}} \Leftarrow$ SMOOTH$(\mathsf{H}, \underline{\mathfrak{u}}, \underline{\mathsf{f}})$
$\underline{\mathsf{q}} \leftarrow \mathfrak{L} \cdot (\underline{\mathsf{f}} - \mathsf{H} \cdot \widetilde{\underline{\mathfrak{u}}})$
$\underline{\mathfrak{p}} \leftarrow 0$
$\widetilde{\underline{\mathfrak{p}}} \Leftarrow$ SMOOTH$(\nu_{\min}^2 \cdot \mathsf{C}, \underline{\mathfrak{p}}, \underline{\mathsf{q}})$
$\underline{\mathfrak{u}} \leftarrow \widetilde{\underline{\mathfrak{u}}} + \mathfrak{L}^T \cdot \widetilde{\underline{\mathfrak{p}}}$
```

matrix C derived from (10) for defining the smoother in $\mathbb{M}_h$ we have to use the correct scaling by $\nu_{\min}^2$ corresponding to the definition of scaled mass matrix $\widetilde{M}$. Now, the smoother SMOOTH can be any standard smoother with a good parallel performance, e.g., a block Jacobi smoother with Gauss-Seidel smoothing in blocks containing interior unknowns of the subdomains. Since HYBRIDSMOOTH involves at least two smoothing steps, one in $\mathbb{X}_h$ and one in $\mathbb{M}_h$, at least two subsequent communications are required.

Note, the post–smoothing step HYBRIDSMOOTH$^T(\mathsf{H}, \mathsf{C}, \underline{\mathfrak{u}}, \underline{\mathsf{f}})$. in Alg. 4 is obtained from Alg. 5 by executing step 1 after steps 2-3-4-5 instead of executing the given order 1-2-3-4-5.

Parallel Domain Decomposition Maxwell Solver

If one is interested in the exterior magnetic field, then the coupling of the FEM with the BEM is certainly the natural technique to handle this problem. For simplicity of the presentation, let us consider the case where the magnetic sources and the ferromagnetic materials are located in some bounded and simply connected Lipschitz domain Ω_F where we will use the FEM for approximating the magnetic field. Thus, we suppose that in the exterior BEM subdomain $\Omega_B := (\bar{\Omega}_F)^c$ the electric current density vanishes, i.e., $J = 0$, and $\nu = \nu_B > 0$ (air). We can again introduce the vector potential u for the B–field $B = \mathrm{curl}(u)$ in Ω_F. However, in the exterior domain Ω_B, the H–field can now be represented as a gradient field of some scalar potential φ, i.e. $H = \mathrm{grad}(\varphi)$ in Ω_B.

Therefore, in the exterior subdomain Ω_B, the magnetostatic Maxwell equations (1) are essentially reduced to the scaled Laplace equation for the scalar potential φ. The Cauchy data for the solution of this equation are related by Calderon's integral equations $\varphi = (\frac{1}{2}\mathcal{I} + \mathcal{K})\varphi - \nu_B \mathcal{V}\lambda$ and $\lambda = -\frac{1}{\nu_B}\mathcal{D}\varphi + (\frac{1}{2}\mathcal{I} - \mathcal{K}^*)\lambda$ on the interface $\Gamma := \partial\Omega_F = \partial\Omega_B$, where φ denotes the trace of the scalar potential on Γ, $\lambda = \frac{1}{\nu_B}\frac{\partial\varphi}{\partial n} = B \cdot n$ on Γ, $n :=$ outer unit normal to Ω_F, $\mathcal{V} :=$ single layer potential operator on Γ, $\mathcal{K} :=$ double layer potential operator on Γ, $\mathcal{D} :=$ hypersingular operator on Γ.

Using now Coulomb's gauging condition $\operatorname{div}(u) = 0$ in Ω_F and Cauchy's representation formula of the Cauchy data together with the interface condition predicting the continuity of the tangential part $H \times n$ of the H–field and of the normal component $B \cdot n$ of the B–field, we arrive at the mixed coupled fe-be variational formulation: Find $(u, \varphi, p) \in V := X \times \Phi \times M$ such that:

$$\begin{aligned} a(u,\varphi;v,\psi) + b(v,p) &= \langle f, v\rangle \quad \forall\,(v,\psi) \in X \times \Phi, & (14)\\ b(u,q) - c(p,q) &= 0 \qquad\quad \forall\, q \in M, & (15)\end{aligned}$$

where $X := H(\mathrm{curl}, \Omega_F)$, $\Phi := H^{1/2}_\star(\Gamma)$, and $M := H^1_\star(\Omega_F)$. The bilinear forms are defined by the identities

$$a(u,\varphi;v,\psi) := \int_{\Omega_F} \nu\, \mathrm{curl}(u)\cdot\mathrm{curl}(v)\,dx + \langle \nu_B \mathcal{V}(\mathrm{curl}(u)\cdot n), \mathrm{curl}(v)\cdot n\rangle_\Gamma$$

$$-\langle(\frac{1}{2}\mathcal{I} + \mathcal{K})\varphi, \mathrm{curl}(v)\cdot n\rangle_\Gamma + \langle\frac{1}{\nu_B}\mathcal{D}\varphi, \psi\rangle_\Gamma + \langle(\frac{1}{2}\mathcal{I} + \mathcal{K}^*)(\mathrm{curl}(u)\cdot n), \psi\rangle_\Gamma,$$

$$b(u,q) := \int_{\Omega_F} u\cdot\nabla q\,dx, \quad c(p,q) := \frac{1}{\nu_{min}}\int_{\Omega_F}\nabla p\cdot\nabla q\,dx, \quad \langle f, v\rangle := \int_{\Omega_F} J\cdot v\,dx.$$

The subscribe "$\star$" means that the function of the corresponding space should be L_2–orthogonal to the constant functions. Again one can show existence and uniqueness of the solution. Moreover, $p = 0$ if $\int_{\Omega_F} J\,\nabla q\,dx = 0 \quad \forall\, q \in H^1_\star(\Omega_F)$ (see [Kuh98] for the proof).

Choosing the finite (boundary) element subspaces $X_h := \mathcal{N}^1_h \subset X$, $\Phi_h := \mathcal{S}^1_h \subset \Phi$ and $M_h := \mathcal{S}^1_h \subset M$, we derive from (14) the symmetric coupled fe-be Galerkin scheme: Find $(u_h, \varphi_h, p_h) \in V_h := X_h \times \Phi_h \times M_h$ such that

$$\begin{aligned} a(u_h,\varphi_h;v_h,\psi_h) + b(v_h,p_h) &= \langle f, v_h\rangle \quad \forall\,(v_h,\psi_h) \in X_h \times \Phi_h, & (16)\\ b(u_h,q_h) - c(p_h,q_h) &= 0 \qquad\quad \forall\, q_h \in M_h, & (17)\end{aligned}$$

that is again equivalent to the following symmetric, but indefinite system of coupled fe-be equations

$$\begin{pmatrix} A & K^T & B^T \\ K & -D & 0 \\ B & 0 & -C \end{pmatrix}\begin{pmatrix} \underline{u}_h \\ \underline{\varphi}_h \\ \underline{p}_h \end{pmatrix} = \begin{pmatrix} \underline{f}_h \\ 0 \\ 0 \end{pmatrix}, \tag{18}$$

where $A = A^F + A^B$ consists of the contributions from the first two terms of the bilinear form $a(\cdot,\cdot)$. Eliminating again $\underline{p}_h = C^{-1}B\underline{u}_h$ from the third equation in (18) and inserting it into the first equation, we obtain the Schur complement system

$$\begin{pmatrix} \widetilde{A} & K^T \\ K & -D \end{pmatrix} \begin{pmatrix} \underline{u}_h \\ \underline{\varphi}_h \end{pmatrix} = \begin{pmatrix} \underline{f}_h \\ 0 \end{pmatrix},$$

where $\widetilde{A} = A^F + A^B + B^T C^{-1} B$. In contrast to the finite element case, here the Schur complement system remains symmetric and indefinite. Similar to the 2D case discussed in [Lan94], we can now construct efficient solvers on the basis of the Bramble-Pasciak transformation [BP88]. In [KS00], M. Kuhn and O. Steinbach describe the ingredients of the preconditioner and present numerical results showing the high efficiency of this solver for coupled fe-be equations in 3D.

Numerical Results

In this section, we present an example from magnetostatics and we apply the algorithms presented above. The geometry of our model problem together with the corresponding coarse surface mesh is shown in Fig. 1 on the left. We consider the model of a transformer with a kernel, three coils and the air domain around these parts. The outer boundary is given by an iron casing of high conductivity that motivates the boundary condition (2). The magnetic field which is to be computed is generated by tangential currents within the three coils. The iron core has a permeability of 1000. The resulting magnetic flux density B is shown in Fig. 1 on the right.

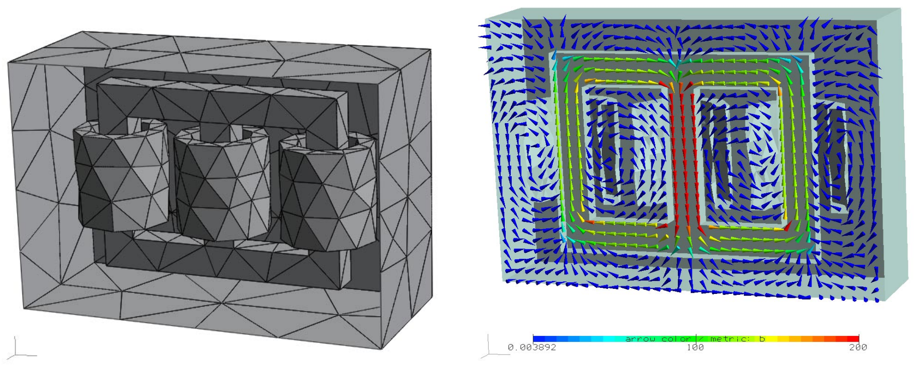

Figure 1: Geometry, initial mesh (left), resulting magnetic flux density B (right).

The basis for our domain decomposition is a tetrahedral mesh generated fully automatically by NETGEN (see [KLS00]). The surface mesh shown in Fig. 1 corresponds to a volume mesh of 2465 tetrahedra. We apply a modified recursive spectral bisection (rsb) algorithm which allows to use any number P of subdomains. The use of the rsb ensures that the elements are distributed almost equally to the P processors being used. Finer meshes corresponding to the levels $\ell = 2, 3, 4$ are obtained from uniform refinement resulting in overall 1262080 tetrahedra. This refinement is purely local and can be realized without communication. However, newly created unknowns at interfaces between different subdomains have to be identified uniquely by all surrounding processors. For this purpose, one root process per interface receives data from all surrounding subdomains, it identifies all unknowns uniquely and it finally distributes this information again to all adjacent processors. This setup phase is required once after each refinement step.

The numerical experiments presented below are carried out on a SGI Origin 2000 machine with 64 CPU R12000, 300 MHz and overall 20 GB main memory. The numerical simulations are carried out using the object oriented C++ code *FEPP* [KLS00]. The message passing is based on MPI from the *SGI Message Passing Toolkit 1.2.* The wall-clock time has been measured by *MPI_WTIME().*

The system (9) has been solved using the PMG algorithm with the relative accuracy 10^{-4}. For the multigrid preconditioner C_H a V–cycle with 1 pre– and 1 post–smoothing step HYBRIDSMOOTH has been used. The multigrid regularisator $\widetilde{C}$ has been realized by a V–cycle with 2 pre– and 2 post–smoothing steps using a standard smoother with good parallel efficiency as described before. Table 1 shows the corresponding results including number of unknowns (dof), number of iterations (It.) and wall-clock time in seconds (T[sec]). We increase the number of unknowns from top to bottom, while the number of processors is increased from left to right. First, we observe that the number of iterations is almost independent of the number of unknowns.

		1		4	16	32	48	60	
ℓ	dof	It.	T	T	T	T	T	It.	T
1	3466	5	0.1	0.1	0.1	0.1	0.1	5	0.1
2	26907	8	7.1	2.9	1.5	1.4	1.4	11	1.7
3	212597	9	84	28	7.3	4.7	3.8	13	4.1
4	1691370	11	1398	496	81	42	28	14	24
T($\ell = 4$)		2593		806	160	86	61	54	
T($\Sigma\ell$)		2852		886.4	188.4	108.5	83.8	79.6	

Table 1: Wall-clock time T for the solver on each level (upper part), overall wall-clock time for $\ell = 4$ and accumulated for all levels ($\Sigma\ell$) in seconds.

This shows the optimality of our algorithm. However, the number of iterations depends slightly on the number of processors. That is because our smoothers depend on the partition of the mesh. In particular, the blocks of unknowns at interfaces where Jacobi steps are performed grow with the number of subdomains. The wall-clock times for the pcg are given in the upper part for each level ℓ separately. Additionally we present the overall time for the finest grid ($\ell = 4$) and together for all grids ($\Sigma\ell$) in the lower part of Table 1. This time includes the grid refinement together with the setup phase for the vector accumulation, the assembling of the matrices and the solution of the system. Table 2 shows the corresponding speedup results. First the overall speedup for the accumulated time over all 4 levels is given. It performs well until $P = 16$ and is no longer optimal for $P = 60$. This loss of efficiency is due to the setup phase for interface unknowns which shows rather bad scalability in the current implementation. So it scales from 32 sec for $P = 16$ to 15 sec for $P = 60$ only. The

P	1	4	16	32	48	60
$\Sigma\ell$	1.0	3.2	15.1	26.2	34.1	35.6
$\ell = 4$	1.0	3.2	16.2	30.0	42.5	48.0
pcg ($\ell = 4$)	1.0	2.8	17.2	33.0	49.6	58.5
1 Iter. ($\ell = 4$)	1.0	3.3	21.9	42.1	63.2	74.4

Table 2: Speedup results for overall time $\Sigma\ell$, time for $\ell = 4$, pcg (solver) for $\ell = 4$ and one iteration of pcg for $\ell = 4$.

speedup computed for $\ell = 4$ shows a slightly better behavior since coarse grid effects are neglected. However, the speedup computed for the solver and $\ell = 4$ only, shows much better results. Here the speedup is almost optimal for $P = 60$. For a more detailed analysis we consider the speedup with respect to one iteration for $\ell = 4$. Now we observe even super-speedups. However this is due to cache effects.

References

[AFW00]Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Multigrid in H(div) and H(curl). *Numer. Math.*, 85(2):197–217, 2000.

[BP88]James H. Bramble and Joseph E. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Mathematics of Computation*, 50(181):1–17, 1988.

[Cia78]Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems.* North-Holland, Amsterdam, 1978.

[Haa98]Gundolf Haase. Parallel incomplete Cholesky preconditioners based on the non-overlapping data distribution. *Parallel Computing*, 24(11):1685–1703, 1998.

[Haa99]Gundolf Haase. *Parallelisierung numerischer Algorithmen für partielle Differentialgleichungen.* Teubner-Verlag, Stuttgart/Leipzig, 1999.

[Hac85]Wolfgang Hackbusch. *Multigrid Methods and Applications.* Springer, Berlin, 1985.

[Hip99]Ralf Hiptmair. Multigrid methods for Maxwell's equations. *SIAM J. Numer. Anal.*, 36:204–225, 1999.

[HLM91]Gundolf Haase, Ulrich Langer, and Arnd Meyer. The approximate Dirichlet decomposition method. part I,II. *Computing*, 47:137–167, 1991.

[JLM$^+$89]Michael Jung, Ulrich Langer, Arnd Meyer, Werner Queck, and Manfred 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.

[KLS00]Michael Kuhn, Ulrich Langer, and Joachim Schöberl. Scientific computing tools for 3D magnetic field problems. In John R. Whiteman, editor, *The Mathematics of Finite Elements and Applications X*, pages 239 – 258. Elsevier, 2000.

[KS00]Michael Kuhn and Olaf Steinbach. FEM–BEM coupling for 3D exterior magnetic field problems. In T. Tiihonen and P. Neittaanmäki, editor, *ENUMATH 99 - Proceedings of the 3rd European Conference on Numerical Mathematics and Advanced Applications, Jyväskylä, Finland, July 26-30, 1999*, pages 180–187, Singapore, 2000. World Scientific.

[Kuh98]Michael Kuhn. *Efficient Parallel Numerical Simulation of Magnetic Field Problems.* PhD thesis, Johannes Kepler University Linz, Institute of Analysis and Computational Mathematics, 1998.

[Lan94]Ulrich Langer. Parallel iterative solution of symmetric coupled BE/FE–equations via domain decomposition. *Contemp. Math.*, 157:335–344, 1994.

[Ń86]J.-C. Nédélec. A new family of mixed finite elements in R^3. *Numer. Math.*, 50:57–81, 1986.