Preprocessing in BE/FE Domain Decomposition Methods *

M. Goppold† G. Haase‡ B. Heise§ M. Kuhn¶

Abstract

The Domain Decomposition Method (DD) is a powerful tool for constructing efficient parallel solvers for Partial Differential Equationd (PDEs) which are well suited to run on Multiple Instruction Multiple Data (MIMD) computers. However, the efficiency of DD–solvers depends heavily on the underlying decomposition of the domain of interest into subdomains. In this paper, we introduce the **A**daptive **D**omain **D**ecomposition **Pre**processor **ADDPre** which realizes an automatic decomposition of the domain under consideration into p subdomains, where p is the number of processors to be used. We discuss both, the codes being involved and the data–formats being used for describing the decomposition of the problem.

Numerical examples, demonstrating the performance of the prepreocessor as well as the efficiency of the parallel solver are presented.

Key words: Partial differential equations, finite element methods, boundary element methods, domain decomposition methods, parallel algorithms, preprocessing, solvers.

AMS subject classifications: 65N55, 65N22, 65F10, 65N30, 65N38, 65Y10, 65Y05, 05C90.

1 Introduction

Nowadays, Domain Decomposition (DD) algorithms are of great interest, since they are the basic tool for constructing algorithms which are well suited to run on Multiple Instruction Multiple Data (MIMD) parallel computers with message-passing. These parallel machines provide sufficient CPU power and sufficiently large storage capacity as it is necessary for the numerical simulation of complex processes. Another important aspect is that DD methods allow us to "marry" the advantages of the Finite Element Method (FEM) to those of the Boundary Element Method (BEM) via a unified coupled variational formulation [6,11,24,27].

Therefore, efficient parallel solvers for large systems of algebraic equations resulting from, e.g., the finite element (FE), the boundary element (BE), or a coupled FE/BE discretization of the partial differential equation, have been developed [11–16,19–23,27,31].

To achieve the inherent high parallel efficiency of the solvers it is necessary to distribute the work to the p processors of the parallel computer with a good load balance. The latter means that the total waiting time, i.e., the sum over all processors of the time that a processor

*This contribution to the *Virtual Proceedings* is a short version of the original paper published as *Technical Report 96–2, Institute of Mathematics, Johannes Kepler University Linz, 1996.*

†Faculty of Mathematics, Technical University Chemnitz, D-90122 Chemnitz, Germany. (E-mail: mgoppold@mathematik.tu-chemnitz.de)

‡Institute of Mathematics, Johannes Kepler University Linz, Altenberger Str. 69, A-4040 Linz, Austria. (E-mail: ghaase@miraculix.numa.uni-linz.ac.at)

§Institute of Mathematics, Johannes Kepler University Linz, Altenberger Str. 69, A-4040 Linz, Austria. (E-mail: heise@miraculix.numa.uni-linz.ac.at)

¶Institute of Mathematics, Johannes Kepler University Linz, Altenberger Str. 69, A-4040 Linz, Austria. (E-mail: kuhn@miraculix.numa.uni-linz.ac.at)

has to wait idle for the others, should be small. As an example, in the FEM-DD for linear elliptic problems, almost equal numbers of nodes, elements, and coupling boundaries should be assigned to each processor. Such a distribution can be set manually for academic test examples only.

A powerful tool for the distribution of an initial mesh represented by a graph is the recursive spectral bisection method (rsb) [4,30]. We will apply an improved version that allows us to decompose a naturally given (by the material coefficients) initial decomposition consisting of P_{mat} subregions of different size into an arbitrary number p of final well-balanced subdomains. First, we determine the number of subdomains in which each subregion is to be decomposed (it needs not to be a power of 2), then we apply the modified rsb to each subregion.

Further, complicated coupling boundaries between the subdomains can reduce the efficiency of the DD based solvers mentioned above. Thus, it is desirable to smooth the coupling boundaries that separate subdomains inside a subregion. For that purpose, a new preprocessing tool has been developed. The coupling boundaries that separate subregions with different coefficients remain unchanged in the standard DD methods.

As a third aspect, adaptive refinement strategies can lead to load-imbalance on the fine grids. Therefore, we provide the possibility to generate an a-priori refined mesh in the preprocessing. If this mesh is used as an input for the spectral bisection method, there will be a quasi-static load balance on the fine grids.

In this paper, we consider a potential problem arising, e.g., from magnetostatics, as a test problem, which can be written formally as follows

$$\begin{aligned} -\mathrm{div}\ (\nu(x)\nabla u(x)) &= \mathrm{S}(x) + \frac{\partial \mathrm{H}_{0\mathrm{y}}(x)}{\partial \mathrm{x}} - \frac{\partial \mathrm{H}_{0\mathrm{x}}(x)}{\partial \mathrm{y}}, \quad x \in \Omega & (1)\\ u(x) &= 0, \quad x \in \Gamma_D \equiv \Gamma := \partial\Omega. & (2) \end{aligned}$$

Here, Ω is a bounded domain, and ν denotes a piecewise constant coefficient function which defines an a-priori decomposition of the domain Ω by

$$\bar{\Omega} = \bigcup_{j=1}^{P_{mat}} \bar{\hat{\Omega}}_j, \quad \text{with} \quad \hat{\Omega}_i \cap \hat{\Omega}_j = \emptyset \quad \forall i \neq j, \quad \text{and} \quad \nu(x) = \nu_j \quad \text{for} \quad x \in \hat{\Omega}_j. \qquad (3)$$

For the physical model, and for $\mathrm{H}_{0\mathrm{x}}$ and $\mathrm{H}_{0\mathrm{y}}$ which stand for sources associated with permanent magnets, we refer to [17,18,20,23].

The rest of the paper is organized as follows. In Section 2, we describe the preprocessing tools in detail. In Section 3, we give a short overview of the parallel algorithm for solving the problem stated above using this decomposition. Numerical result concerning both, the preprocessor and the parallel solver are presented in Section 4. We conclude the paper with some remarks in Section 5.

2 Preprocessing

Having in mind to use a multiprocessor computer to solve DD-equations the most natural idea is to use one processor per subdomain. In our case we assume that the number of processors (p) to be used is (much) larger than the number of a-priori given material-defined domains, i.e., we have $p > P_{mat}$. Thus, we wish to decompose the domains $\hat{\Omega}$ further, which leads finally to

$$\bar{\Omega} = \bigcup_{i \in \mathcal{I}} \bar{\Omega}_i, \quad \text{with} \quad \bar{\hat{\Omega}}_j = \bigcup_{i \in \mathcal{I}_j} \bar{\Omega}_i \quad \forall j = 1, \ldots, P_{mat} \qquad (4)$$

where the sets of indices are given by

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

i.e., the subdomains $\hat{\Omega}_j$ determined by the materials may be decomposed further (see, e.g., [15]). We assume that there exist open balls $B_{\underline{r}_i}$ and $B_{\overline{r}_i}$ $(i \in \mathcal{I})$ with positive radii $\underline{r}_i$ and $\overline{r}_i$, such that

$$B_{\underline{r}_i} \subset \Omega_i \subset B_{\overline{r}_i} \quad \text{and} \quad 0 < \underline{c} \leq \overline{r}_i / \underline{r}_i \leq \overline{c} \quad \forall i \in \mathcal{I} \tag{5}$$

with fixed (i-independent) constants $\underline{c}$ and $\overline{c}$.

The task of decomposing Ω into p subdomains starting with a fixed discretization of the interfaces $\Gamma_M := \cup_{j=1}^{P_{mat}} \partial\hat{\Omega}_j$ aiming at a load-balanced decomposition is not trivial. Actually we seek such a decomposition for an a-priori defined discretization of the boundary which minimizes the imbalance between the processors, i.e.,

$$\max_{i=1,\ldots,p} \{\beta_i\ N_i\} \,/ \min_{i=1,\ldots,p} \{\beta_i\ N_i\} \longrightarrow \text{Minimum} \tag{6}$$

where N_i $(i = 1, \ldots, p)$ is the number of local elements arising after the decomposition and β_i is a piecewise constant (over $\bar{\hat{\Omega}}$) weight function representing the work per element.

In the following we describe the data formats, the file types, the tools and, finally, the preprocessor ADDPre.

2.1 Data Formats

Throughout the process of preprocessing we are concerned with two types of describing data: dd-data and tri-data. Both of them describe the geometry of the problem (shape of the domain, material interfaces) as well as the discretization (density and grading of the mesh). Each of them consists of certain classes of objects which are defined hierarchically.

type	object	definition
dd–data	crosspoints	coordinates
	geometrical edges	two crosspoints, (midpoint), refinement parameter, (boundary condition)
	faces	geometrical edges, material pointer
	subdomains	faces
tri–data	points	coordinates
	edges	two points, (midpoint), (boundary condition)
	triangles	edges, material pointer

The main difference between the two data types is how the discretization is being represented. The dd-data describe the refinement only on the interfaces (as it is sufficient for a BEM discretization), whereas the tri-data describe the full 2d-discretization (as it is required for a FEM discretization) and are thus, in some sense, the 2d–realization of the refinement information contained in the dd-data.

The following parameters are used to describe the refinement on the geometrical edges:

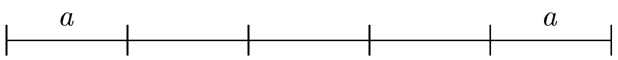
nn	$\longrightarrow$	Number of inner nodes on an edge for the coarsest grid,
kind	$\longrightarrow$	Kind of discretization,
r_1	$\longrightarrow$	Ratio 1 of element sizes for the coarsest grid.

For simplicity we define additionally:

$$ne \longrightarrow \text{Number of elements on an edge for the coarsest grid,}$$

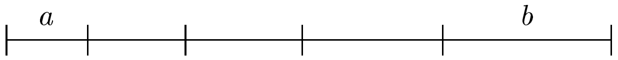
where $ne = nn + 1$ holds. The parameter *kind* has the range $[1, \ldots, 3]$, where the numbers define the coarsest grid as follows.

- kind = 1: Uniform discretization.



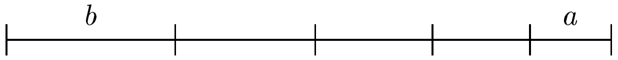
$$r_1 := 1.0$$

- kind = 2: Compression towards the starting point.



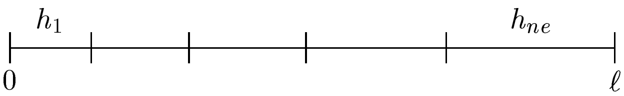
$$r_1 := \frac{b}{a}$$

- kind = 3: Compression towards the end point.



$$r_1 := \frac{b}{a}$$

Internally these three cases are reduced to one (grading into one direction) by introducing the length h_1 of the first interval and for the ratio q_1 of neighboured intervals.



$$r_1 := \frac{h_{ne}}{h_1}$$

The paramters are defined by

$$\begin{aligned} q_1 &= \sqrt[ne-1]{r_1} \\ h_1 &= \begin{cases} \ell/ne & \text{iff} \quad q_1 = r_1 = 1.0 \\ \ell \frac{q_1 - 1}{q_1^{ne} - 1} & \text{otherwise.} \end{cases} \end{aligned}$$

where [.] denotes the integer part and r is assumed not to be one, i.e. the uniform discretization is excluded.

Further grids are obtained by dividing each interval into two subintervals such that one gets a grading with the ratio which is the square root of the previous ratio. That is if two neighboured (old) intervals do have the ratio q_1 the two subintervals will have the ratio $\sqrt{q_1}$.

Remark 2.1 All edges may be straight lines or arcs of a circle. In the latter case an additional point (midpoint) defines this edge. This information would then be inherited throughout all stages of further refinement.

2.2 The files

Alltogether three types of files are being used during the preprocessing. There are $*$*.tri*, $*$*.dd* and $*$*.fb* files. The first two contain the full geometrical information based on triangles and dd-data, respectively, whereas the third type is being used for auxiliary data.

2.3 The Codes

The preprocessor ADDPre uses three main tools which are AdapMesh, Decomp and Tri2DD. Now, we will describe these tools including the main ideas they are based on.

2.3.1 AdapMesh

AdapMesh converts dd-data into tri-data. This code applies an improved version of the mesh generator ParMesh [9] to each of the given subdomains in parallel or sequentially one after another. The resulting global mesh is conforming along the interfaces since the discretization at these interfaces has been predescribed.

Additionally, it is possible to adapt the refinement to singularities, i.e., one has to change the refinement parameter which are given in the input dd-file before starting the mesh generator. For this purpose, a possibly sequential coarse grid calculation could give rise to the right choice of the new refinement parameter.

2.3.2 Decomp

The program Decomp decomposes a domain Ω containing P_{mat} single-material domains $\widehat{\Omega}_k$ into an arbitrary number of subdomains $p \geq P_{mat}$ so that in analogy to (4)

$$\overline{\Omega} = \bigcup_{k=1}^{P_{mat}} \overline{\widehat{\Omega}}_k = \bigcup_{k=1}^{P_{mat}} \bigcup_{i=1}^{p_k} \overline{\Omega}_i = \bigcup_{i=1}^{p} \overline{\Omega}_i, \quad p \equiv \sum_{k=1}^{P_{mat}} p_k$$

holds, where p_k denotes the number of subdomains into which a single-material domain $\widehat{\Omega}_k$ will be decomposed and $\overline{\widehat{\Omega}}_k = \bigcup\limits_{i=1}^{p_k} \overline{\Omega}_i$ is fulfilled. So each subdomain Ω_i belongs to exactly one material. In the context above the term material is not just restricted to different material constants but is also extended to different type of differential equations describing the local problem or different techniques for solving.

The decomposition of the domain Ω cannot be done by pure geometric information of the domain Ω. For the sake of decomposition we need a triangular element grid and decompose this grid. This grid will be read in by the program as a *.tri file. Each element has to be characterized by the number of the material it belongs to.

Let N denote the number of elements in the whole domain Ω, $\widehat{N}_k$ the number of elements in the single-material domains $\widehat{\Omega}_k$ and N_i the number of elements in the subdomains Ω_i. Then we can rewrite the relations between the domains into relations of the numbers of elements

$$N = \sum_{k=1}^{P_{mat}} \widehat{N}_k = \sum_{k=1}^{P_{mat}} \sum_{i=1}^{p_k} N_i = \sum_{i=1}^{p_k} N_i, \quad p \equiv \sum_{k=1}^{P_{mat}} p_k \ .$$

For explaining the technique used for decomposing the grid we first assume that we have just one material in the whole domain, e.g. $\overline{\Omega} = \overline{\widehat{\Omega}}$ and $P_{mat} = 1$. In the case of a decomposition into $p = 2^s$, $s \in \mathbb{N}$ subdomains the usual recursive spectral bisection method (rsb) is available [30]. For distributing the grid into an arbitrary number of subdomains we adapted an idea of Clemens Brand [4]. The aim of all those decompositions is to distribute the elements in such a way that each subdomain possesses the same (or nearly the same) number of elements ($N_i \approxeq N/p$).

In the case of $P_{mat} > 1$ we handle each material domain $\widehat{\Omega}_k$ ($k=\overline{1,P_{mat}}$) separately as described above. Now the difficulty is to determine a–priori the number of subdomains p_k per material domain. The proper calculation for a distribution into p subdomain is done by the following **algorithm:**

1. Initialization: $\overline{p} := P_{mat}$, $p_k := 1$, $N_k := \widehat{N}_k \qquad \forall k = \overline{1, P_{mat}}$
2. WHILE $(\overline{p} < p)$ DO   Find $j : N_j = \max\limits_{i=1,\dots,\overline{p}} N_i$
   Set $\overline{p} := \overline{p} + 1$, $N_{\overline{p}} := N_j/2$, $N_j := N_j/2$

   DONE

A load balanced distribution should also take into account that different material domains may yield to different numerical efforts. So a weighted distribution of the elements is necessary which fulfills the condition

$$\max_{i=\overline{1,p}}\{\beta_i N_i\} \,/\, \min_{i=\overline{1,p}}\{\beta_i N_i\} \quad \longrightarrow \quad \text{Minimum} \quad .$$

The weights β_i denote the average arithmetical work per element for solving the local problem in a subdomain Ω_i. That means, e.g., nonlinear behaviour of the material constants in the subdomain Ω_i will be reflected by a larger β_i. To include this weighting condition in the algorithm above all N_i have to be declared as REAL and the third initialization statement has to be changed into $N_k := \beta_k \widehat{N}_k$.

As a result each triangle obtains the number of the subdomain it belongs to. This information will be written as a *.tri file.

Remark 2.2 The decomposition above is element based. Because of the fact that most of the numerical algorithms solving large systems of equations (resulting from partial differential equations) are node based, the decomposition above does not reflect the numerical load balance correctly but it works sufficiently good. Additionally no load balancing concerning communication is taken into account but one great advantage of the rsb is the minimization of the subdomain boundaries. In some practical problems with a rather strange geometry of the material domains, e.g., long stripes, the communication balance is a -priori very poor.

2.3.3 Tri2DD

Tri2DD converts tri-data into dd-data. Hereby, interfaces between different materials and the refinement information at these interfaces are fully maintained. On the other hand interfaces within one material are to be smoothed to get "simple" polygons as subdomains. The aim is to obtain smooth interfaces between subdomains belonging to the same material and to minimize the global number of crosspoints without destroying the load balance generated by Decomp.

The **algorithm** can be described as follows.

1. Find, for each subdoamin, the local sequence of boundary edges. Define these edges as 'free'.
2. Find primary crosspoints (cp).
   a) cp defined in the fb–file.
   b) points which are the intersection of at least three subdomains, where boundary conditions are interpreted as (virtual) subdomains.
3. DO for all subdomains

   WHILE there are 'free' (local) boundary edges

   a) Find a local cp P_1 which is a vertex of a 'free' boundary edge. If there is no such cp then define some vertex P_1 of a 'free' edge as cp.
   b) Start at P_1, go along the 'free' boundary edges UNTIL for two subsequent points P_2, P_3
      i) P_2 is a cp
      ii) $\overline{P_1P_2}$ would be accepted as new geometrical edge, whereas $\overline{P_1P_3}$ would not be accepted as new geometrical edge. Then define P_2 as new cp.
   c) Define $\overline{P_1P_2}$ as new geometrical edge and set the corresponding boundary edges to 'busy'. Define refinement parameter for the new geometrical edge according to the boundary edges.

   END WHILE

   END DO
4. Renumber new crosspoints.

This algorithm is finite since in each iteration at least one 'free' boundary edge is set to 'busy'. An open question is how to decide whether a virtual edge $\overline{P_1P_2}$ is to be accepted as new geometrical edge or not. For this purpose we distinguish between two cases:

C1: The boundary edges describe an interface between two different materials.

C2: The boundary edges describe an interface between two subdomains which belong to the same material.

The following criteria are applied:

C1: If the new edge describes exactly the interface it is allowed.

C2: The new edge must intersect all polygons defined by the boundary points between $\overline{P_1P_2}$. If this is not the case, or if it intersects a second material it is not allowed.

The refinement parameter are determined such that they reflect the behaviour of the boundary edges (length and grading) as good as possible.

2.4 The Preprocessor ADDPre

The complete **algorithm** can now be formulated as follows.

0. Create (manually) a dd-description of the problem and state a-priori refinement information on the boundaries (see Fig. 1a).
1. Use AdapMesh to convert the dd-data into tri-data. (see Fig. 1b).
2. Use Decomp to decompose the domain, i.e., to assign each triangle to one of the subdomains (see Fig. 1c).
3. Use Tri2DD to convert the tri-data back into dd-data (see Fig. 1d).
4. Stop or, if desired, modify the refinement information in the dd-file and goto (1.).

Starting off with any of the data formats the preprocessor has to maintain the original information throughout all stages ending up with a dd-file which describes finally the decomposition of our domain into subdomains according to the a-priori given refinement information.

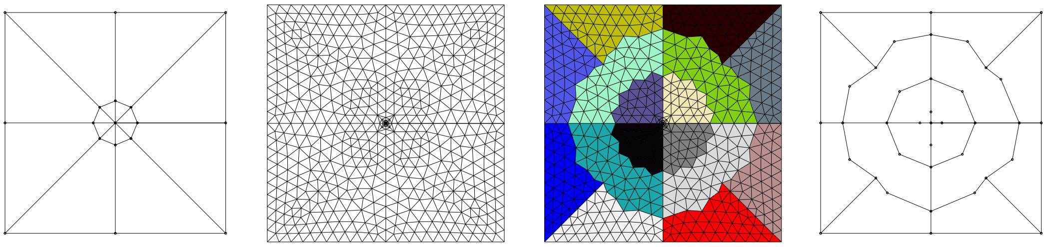


Figure 1: The 4 stages (a–d from left to right) of preprocessing starting with an initial decomposition (D1) on the left and ending up with a load-balanced decomposition (D2) on the right.

The preprocessor performs a quasi-static decomposition. It is possible to restart the preprocessing optionally in step (4). This may be necessary if the refinement information (grading or density of the mesh) have been modified, e.g., according to the bahaviour of the solution obtained from a coarse grid calculation. The a-posteriori change of the refinement information, i.e., of the discretization, destroys in general the load-balance, that is the present decomposition is no longer a solution to (6).

Remark 2.3 The preprocessor allows us to start off with any data format and to decompose Ω into any number $p \geq P_{mat}$ of subdomains. The only data which are fixed throughout the preprocessing are the definition of Γ_M and the a-priori given refinement information on Γ_M !

Remark 2.4 Since the rsb requires a full 2d-mesh it is always, i.e. also for DD–BEM, necessary to generate a full 2d mesh ! At a first glance this may seem disadvantageous, especially if one already starts with a dd-description of the problem. On the other hand a slightly improved decomposition may give major improvements if time-dependent or optimization problems involving repeated calls of the linear solver are to be solved.

3 DD–Formulation and Parallel Solution

Let us consider a linear magnetic field problem of the form (1)-(2) for which a domain decomposition according to (4) is available. In particular, we assume that the index set $\mathcal{I} = \mathcal{I}_F \cup \mathcal{I}_B$

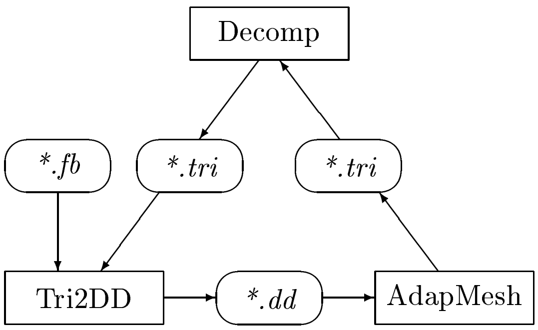


Figure 2: The preprocessor can be started with any file. The output is the dd–file at the bottom.

can be decomposed into two disjoint sets of indices $\mathcal{I}_F$ and $\mathcal{I}_B$ such that

$$(\text{supp S}(.) \cup \text{supp H}_0(.)) \cap \Omega_i = \emptyset \quad \forall i \in \mathcal{I}_B, \tag{7}$$

$$\text{diam}(\Omega_i) < 1 \quad \forall i \in \mathcal{I}_B. \tag{8}$$

For each Ω_i $(i \in \mathcal{I})$ the index i belongs to one of the two index sets $\mathcal{I}_B$ and $\mathcal{I}_F$ according to the discretization method applied to Ω_i, where $\mathcal{I}_B$ and $\mathcal{I}_F$ stand for BEM and FEM, respectively. Note that the condition (8) is only technical and can be fulfilled by scaling the problem appropriately.

Following M.Costabel [6], G.C.Hsiao and W.L.Wendland [24] and others, we can rewrite the weak formulation of the boundary value problem (1)-(2) by means of partial integration in the boundary element subdomains Ω_i, $i \in \mathcal{I}_B$ and by the use of Calderón's representation of the full Cauchy data as a mixed DD coupled domain and boundary integral variational problem: Find $(\lambda, u) \in \mathbf{V} := \mathbf{\Lambda} \times \mathbf{U}_0$ such that

$$a(\lambda, u; \eta, v) = \langle F, v \rangle \quad \forall (\eta, v) \in \mathbf{V}, \tag{9}$$

where

$$\begin{aligned}
a(\lambda, u; \eta, v) &:= a_B(\lambda, u; \eta, v) + a_F(u, v) \\
a_B(\lambda, u; \eta, v) &:= \sum_{i \in \mathcal{I}_B} \nu_i \Big\{ \langle \mathcal{D}_i u_i, v_i \rangle_{\Gamma_i} + \frac{1}{2} \langle \lambda_i, v_i \rangle_{\Gamma_i} + \langle \lambda_i, \mathcal{K}_i v_i \rangle_{\Gamma_i} + \\
&\qquad \langle \eta_i, \mathcal{V}_i \lambda_i \rangle_{\Gamma_i} - \langle \eta_i, \mathcal{K}_i u_i \rangle_{\Gamma_i} - \frac{1}{2} \langle \eta_i, u_i \rangle_{\Gamma_i} \Big\} \\
a_F(u, v) &:= \sum_{i \in \mathcal{I}_F} \int_{\Omega_i} \nu(x) \nabla^T u(x) \nabla v(x) \, dx \\
\langle F, v \rangle &:= \sum_{i \in \mathcal{I}_F} \int_{\Omega_i} \left[\mathrm{S}(x) v(x) - \mathrm{H}_{0\mathrm{y}}(x) \frac{\partial v(x)}{\partial \mathrm{x}} + \mathrm{H}_{0\mathrm{x}}(x) \frac{\partial v(x)}{\partial \mathrm{y}} \right] dx \\
\langle \lambda_i, v_i \rangle_{\Gamma_i} &:= \int_{\Gamma_i} \lambda_i v_i \, ds \text{ and } v_i = v|_{\partial\Omega_i},\ u_i = u|_{\partial\Omega_i},\ \Gamma_i := \partial\Omega_i,
\end{aligned}$$

with the well-known boundary integral operators $\mathcal{V}_i, \mathcal{K}_i, \mathcal{D}_i$ defined by the relations

$$\begin{aligned}
\mathcal{V}_i \lambda_i(x) &:= \int_{\Gamma_i} \mathcal{E}(x, y) \lambda_i(y) \, ds_y \\
\mathcal{K}_i u_i(x) &:= \int_{\Gamma_i} \partial_y \mathcal{E}(x, y) u_i(y) \, ds_y \\
\mathcal{D}_i u_i(x) &:= -\partial_x \int_{\Gamma_i} \partial_y \mathcal{E}(x, y) u_i(y) \, ds_y
\end{aligned} \tag{10}$$

and with the fundamental solution

$$\mathcal{E}(x,y) = -\frac{1}{2\pi}\log|x-y| \tag{11}$$

of the Laplacian (see, e.g., [7]). The spaces $\mathbf{U}_0$ and $\mathbf{\Lambda}$ are defined by the relations

$$\begin{aligned} \mathbf{U}_0 &:= \{u \in \mathrm{H}^1(\Omega_-) : \; u|_{\Gamma_{BE}} \in \mathrm{H}^{1/2}(\Gamma_{BE}), u|_{\partial\Omega_0} = 0\} \\ \mathbf{\Lambda} &:= \left\{\lambda = (\lambda_i)_{i\in\mathcal{I}_B} : \; \lambda_i \in \mathrm{H}^{-1/2}(\Gamma_i), \; i \in \mathcal{I}_B\right\} = \textstyle\prod_{i\in\mathcal{I}_B}\Lambda_i, \end{aligned} \tag{12}$$

with $\Lambda_i = \mathrm{H}^{-1/2}(\Gamma_i)$, $i \in \mathcal{I}_B$. Further we use the notation $\Gamma_{BE} := \bigcup_{i\in\mathcal{I}_B}\partial\Omega_i \setminus \Gamma_D, \Gamma_{FE} := \bigcup_{i\in\mathcal{I}_F}\partial\Omega_i \setminus \Gamma_D, \Gamma_C := \Gamma_{BE} \cup \Gamma_{FE}$ and $\Omega_F := \bigcup_{i\in\mathcal{I}_F}\Omega_i$. Introducing standard norms [11] one can prove that the bilinear form a(.,.) is $\mathbf{V}$–elliptic and $\mathbf{V}$–bounded provided that the domain decomposition satisfies the conditions (5),(8), see also [24]. Therefore, the existence and uniqueness of the solution are a direct consequence of the Lax–Milgram theorem.

The problem may be discretized using a standard nodal FE basis, corresponding basis functions for u on Γ_{BE}, and suitable basis functions for approximating λ on Γ_{BE} (see, e.g., [23,27]). This discretization results in a linear system

$$\begin{pmatrix} K_\Lambda & -K_{\Lambda C} & 0 \\ K_{C\Lambda} & K_C & K_{CI} \\ 0 & K_{IC} & K_I \end{pmatrix} \begin{pmatrix} \mathbf{u}_\Lambda \\ \mathbf{u}_C \\ \mathbf{u}_I \end{pmatrix} = \begin{pmatrix} \mathbf{f}_\Lambda \\ \mathbf{f}_C \\ \mathbf{f}_I \end{pmatrix}. \tag{13}$$

Here the index "Λ" denotes the unknowns associated with λ on Γ_{BE}. An index "C" denotes vector components corresponding to the nodal basis on Γ_C, i.e., the coupling boundaries between the FE/FE, FE/BE, and BE/BE subdomains. The index "I" corresponds to the nodes belonging to the interior of subdomains Ω_i, $i \in \mathcal{I}_F$. The nonsymmetric, positive definite system (13) can be approximately solved by Bramble/Pasciak's CG method [2]. The method requires a preconditioner C_Λ which can be inverted easily and which fulfils the spectral equivalence inequalities $\underline{\gamma}_\Lambda C_\Lambda \leq K_\Lambda \leq \overline{\gamma}_\Lambda C_\Lambda$, with $\underline{\gamma}_\Lambda > 1$. Further, we have to find a preconditioner C_2 for the matrix

$$\tilde{K}_2 = \begin{pmatrix} K_C + K_{C\Lambda}K_\Lambda^{-1}K_{\Lambda C} & K_{CI} \\ K_{IC} & K_I \end{pmatrix}. \tag{14}$$

which we may construct by

$$C_2 = \begin{pmatrix} I_C & K_{CI}B_I^{-T} \\ 0 & I_I \end{pmatrix} \begin{pmatrix} C_C & 0 \\ 0 & C_I \end{pmatrix} \begin{pmatrix} I_C & 0 \\ B_I^{-1}K_{IC} & I_I \end{pmatrix} \tag{15}$$

applying the standard FEM DD principles described in [13,14,16]. The complete method is given in detail in [23,27], see also [20].

The parallel implementation of the PCG algorithm which runs completely in parallel with the exception of the two scalar products and the preconditioning, is described in [27]. An improved version has been presented in [11,23].

The performance of the algorithm depends heavily on the right choice of the preconditioners C_Λ, C_C, C_I and the basis transformation B_I. The matrices C_Λ, C_I and B_I are block-diagonal matrices with the blocks $C_{\Lambda,i}$, $C_{I,i}$ and $B_{I,i}$, respectively. In our experiments, the following components have turned out to be the most efficient ones:

$C_{I,i}$: (**Vmn**) Multigrid V-cycle with m pre- and n post-smoothing steps in the Multiplicative Schwarz Method [12,14].

$C_{\Lambda,i}$: (**mgV**) $\delta_i \cdot K_{\Lambda,i}\,(I_{\Lambda,i} - M_{\Lambda,i})^{-1}$. $M_{\Lambda,i}$ is the multigrid operator satisfying the conditions formulated in [26].

$B_{I,i}$: (**HExt**) Implicitly defined by hierarchical extension (formally $E_{IC,i} = -B_{I,i}^{-1} K_{IC,i}$) [16].

C_C: (**S-BPX**) Bramble/Pasciak/Xu type preconditioner [32].
(**BPS-D**) Bramble/Pasciak/Schatz type preconditioner [3,8].
(**mgD**) $K_{C^*}\,(I_C - M_C)^{-1}$, $(K_{C^*}\,\mathbf{u}_C, \mathbf{v}_C) = \sum_{i \in \mathcal{I}} \nu_i \langle \mathcal{D}_i u_i, v_i \rangle$, as described in [5].

For the components given above, optimal or almost optimal spectral equivalence results have been proved [1,3,5,8,14,12,16,32], see [11] for an overview of the relevant results. Consequently, we can estimate the numerical effort Q to obtain a relative accuracy ε by $Q = \mathcal{O}(h^{-2}\ \ln\ln h^{-1}\ \ln\varepsilon^{-1})$ in the (S-BPX) and (mgD) cases, i.e. almost optimal. In the (BPS-D) case, one has to add a factor $\ln h^{-1}$ [11]. If a BPX-type extension [28] is applied instead of (HExt) in a nested iteration approach, we can prove that $Q = \mathcal{O}(h^{-2})$, i.e., we obtain an optimal method.

4 Numerical Results

4.1 The Slit Problem Using Graded Grids

As a test problem we consider the Dirichlet boundary value problem

$$\Delta u(x) = 0 \quad \text{in}\,\Omega \quad \text{and} \quad u(x) = r^{1/2} \sin \phi/2 \quad \text{on}\,\partial\Omega \tag{16}$$

with Ω being the square $(-1,1) \times (-1,1)$ with a slit $([0,1], 0)$ and (r, ϕ) are the polar coordinates with respect to the origin.

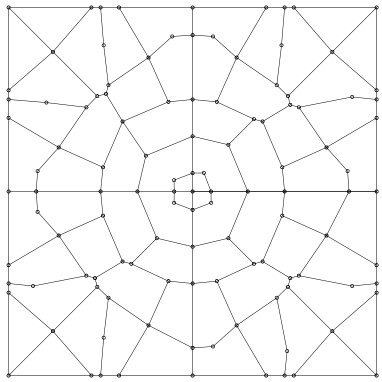


Figure 3: The decomposition (D3) into 64 subdomains.

The components of the preconditioner have been chosen as follows:

$$\begin{array}{rl} C_C: & \text{mgD (multigrid)} \\ C_{I,i}: & \text{V11} \\ B_{I,i}: & \text{HExt.} \end{array}$$

The BE–matrices were computed fully analytically using piecewise linear functions for the displacements and piecewise constant functions for the tractions [31].

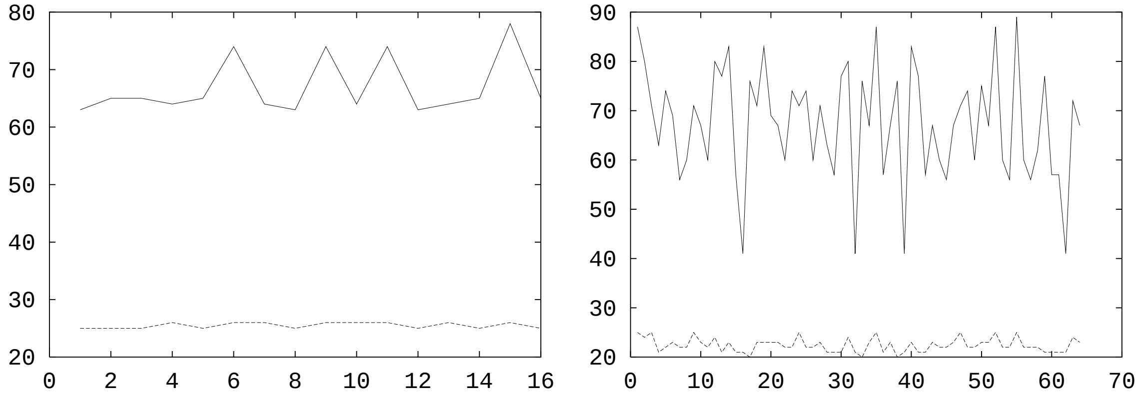


Figure 4: The local number of finite elements (upper curves) and boundary elements (lower curves) for 16 subdomains (D2) (left) and 64 subdomains (D3) (right) after the discretization.

We present the results for two different decompositions: an automatic decomposition into 16 subdomains (D2) (see Fig. 1d) and an automatic decomposition into 64 subdomains (D3) (see Fig. 3), both resulting from a manual decomposition into 16 subdomains (D1) (see Fig. 1a) with a mesh-grading towards the origin with $h_{max}/h_{min} > 100$. The underlying discretization for (D3) is about as double as fine as for (D2), such that the local problem size should be constant, i.e., independent of p.

	BEM		FEM	
	min	max	min	max
(D2)	25	26	68 (63)	68 (78)
(D3)	20	25	65 (41)	66 (89)

Table 1: The direct results of the preprocessor (number of BE/FE–elements, respectively) and the number of FE–elements (in parentheses) after applying the mesh–generator.

The results of the preprocessing for (D2) and (D3) are given in Table 1, Fig. 4 shows the resulting local number of elements after remeshing the subdomains as it is realized in the parallel code.

Looking at the results for (D2) and (D3) in Table 2 we observe constant iteration numbers with respect to both, l and p. Even BEM and FEM results are similar. The values obtained for the scaled efficiency are quite satisfying taking into account that the process of decomposition has been performed fully automatically for a non-standard graded mesh.

4.2 The Induction Machine

As a second application, let us consider a nonlinear magnetic field problem. In many cases, the linear model is not sufficient for modelling electromagnetic phenomena. Therefore we take into account that for ferromagnetic materials the permeability depends on the absolute value of the magnetic induction B, i.e. the material function $\nu(x)$ has to be replaced by a material function $\nu(x, |\mathrm{B}|) = \nu(x, |\nabla u(x)|)$ with

$$\mathrm{H} = \nu(., |\mathrm{B}|)\mathrm{B},$$

where H is the magnetic field strength, i.e., we replace ν_j by $\nu_j(|\nabla u|)$ in (3). We allow ferromagnetic materials with non-constant permeability ν to be in the FEM subdomains

(D2) 16 BEM		(D3) 64 BEM			(D2) 16 FEM		(D3) 64 FEM	
I(ϵ)	CPU	I(ϵ)	CPU	l	I(ϵ)	CPU	I(ϵ)	CPU
18	2.9	18	4.7	1	12	2.0	13	3.7
19	4.2	19	6.5	2	15	2.8	16	5.0
19	5.0	20	7.9	3	16	4.2	17	6.5
20	9.9	20	16.3	4	17	6.6	18	10.2
20	25.1	20	26.9	5	17	15.2	18	21.0
8897		32699		N(5)	136081		550795	
1.00 $\rightarrow$ 0.86				eff.	1.00 $\rightarrow$ 0.79			

Table 2: Iteration count ($I(\epsilon)$, $\epsilon = 10^{-6}$), CPU time (system generation and solution) in seconds for BEM and FEM discretizations and scaled efficiency according to the CPU time and the number of unknowns (N(5)) for the 5th level. The experiments were carried out on a GC–PowerPlus using one processor per subdomain.

$\Omega_i, i \in \mathcal{I}_F$ only. The analysis of nonlinear magnetic field problems can be found in [17,18]. We apply a Newton approach for linearization together with the "nested iteration" method. The latter means that we generate a multilevel sequence of coupled FE/BE discretizations denoted by the grid numbers $q = 1, \ldots, l$. We begin with solving the nonlinear system by Newton's method on the coarsest grid $q = 1$. Then we take the approximate solution on the grid $q-1$ interpolated onto the finer grid q as an initial approximation for Newton's method on the grid q, for $q = 2, \ldots, l$. This allows us to "catch" the nonlinearity on the coarsest grid, cf. [17]. The complete Parallel Nested Newton algorithm is described in [19–21].

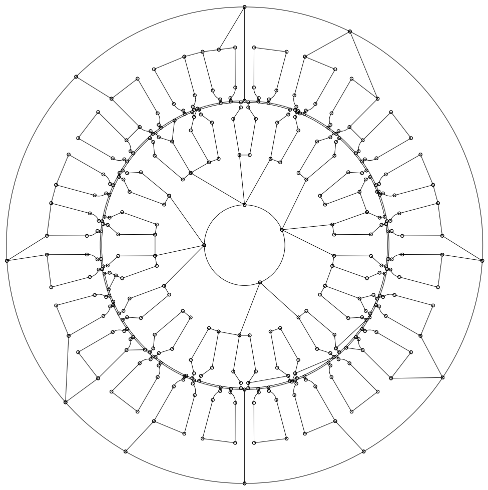


Figure 5: Domain decomposition for the induction machine.

The example under consideration, an induction machine (asynchronous motor), is a real-life application. It is challenging to both the DD preprocessing tool, and the solver, due to its very complicated interior geometry and the strong influence of saturation (i.e., a strong 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.

Starting with a triangulation of the machine consisting of 3520 triangular elements (with the material class assigned to each triangle) given in a tri-data file, we generate a decomposi-

Table 3: Performance for the induction machine

Choice for C_C	BPS-D	S-BPX
Newton iterations 1st grid	10	14
CG iterations 1st grid	$68 \cdots 83$	$42 \cdots 52$
Newton iterations 2nd grid	2	2
CG iterations 2nd grid	43,55	36,45
Newton iterations 3rd grid	2	2
CG iterations 3rd grid	42,30	37,23
Newton iterations 4th grid	2	2
CG iterations 4th grid	13,82	34,59
Newton iterations 5th grid	6	5
CG iterations 5th grid	36,61,64,27,68,64	38,59,22,79,29
Time (generation)	54.5	50.7
Time (linear solver)	506.0	608.5
Total time	560.5	659.2

Time in seconds, GC-Power Plus, 64 processors; 64 FEM subdomains, 549 091 unknowns, choice for C_I: V11, relative accuracy $\varepsilon = 10^{-6}$

tion of the cross-section of the machine into 64 subdomains by applying the routines **Decomp** and **Tri2DD**. We display the resulting decomposition in Figure 5.

Looking at the decomposition, one might imagine that a "manual" decomposition could result in a "more symmetric", "nice" picture. But this would take many hours of engineer's time, and thus contradicts the idea of accelerating field calculations by DD methods.

Performance results for a pure FEM-DD method ($\mathcal{I}_B = \emptyset$) are presented in Table 3. Further numerical results applying the identical decomposition of the machine but other solvers (e.g., global parallel Newton multigrid methods) can be found in [22].

4.3 The Dam

Now, we want to extend the ideas discussed above to problems of plane linear elasticity in which the displacement $u(x) = (u_1(x), u_2(x))^T$ satisfies formally the system of Lamé equations

$$\begin{array}{l} -\mu(x)\Delta u(x) - (\lambda(x) + \mu(x))\text{grad div}\, u(x) = 0 \quad \text{in } \Omega \\ u(x) = 0 \text{ on } \Gamma_D, \quad \sum_{l=1}^{2} \sigma_{kl}(u(x)) n_l = g_k(x) \text{ on } \Gamma_N, \ (k = 1,2) \end{array} \tag{17}$$

where Ω is a bounded Lipschitz domain, $\sigma_{kl}(u)$ are the components of the stress tensor $\sigma(u)$ and $n(x) = (n_1(x), n_2(x))^T$ is the outward normal vector to $\Gamma_D \cup \Gamma_N = \Gamma := \partial\Omega$ ($\Gamma_D \neq \emptyset$) and λ and μ, $\lambda, \mu > 0$, are the Lamé coefficients of the elastic material. In (17), $g = (g_1, g_2)^T$ is the vector of boundary tractions. The extension of the theory presented above for potential problems to linear elasticity is straightforward. The equations and definitions can be found, e.g., in [11].

As a test problem we consider a dam filled with water as sketched in Figure 6. As indicated there, Dirichlet boundary conditions (b.c.) are given on Γ_D (zero displacement) and Neumann b.c. on Γ_N (the tractions are equal to zero, or they are chosen according to the water pressure). The Lamé constants are given for rock by $\mu_r = 7.26\text{e}4\text{MPa}$, $\lambda_r = 3.74\text{e}4\text{MPa}$ and for concrete by $\mu_c = 9.2\text{e}6\text{MPa}$, $\lambda_c = 9.2\text{e}6\text{MPa}$.

We consider two automatic decompositions: one into 16 subdomains (D4) (as drawn in the middle in Fig. 6) and one into 64 subdomains (D5) (as drawn on the right in Fig. 6), where for

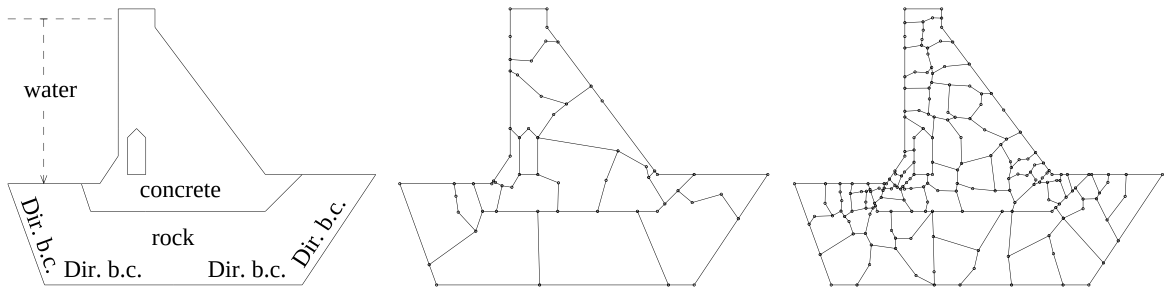


Figure 6: The dam (left) and its decomposition into 16 subdomains (D4) and 64 subdomains (D5).

the latter the discretization on Γ is about as double as fine as for (D4). Both decompositions are the result of the automatic preprocessing started with an a–priori graded grid towards the regions where high stresses are expected.

(D4) 16 BEM		(D5) 64 BEM			(D4) 16 FEM		(D5) 64 FEM	
I(ϵ)	CPU	I(ϵ)	CPU	l	I(ϵ)	CPU	I(ϵ)	CPU
47	3.9	53	24.1	1	32	3.7	30	19.1
51	5.1	56	27.6	2	42	4.8	37	21.5
53	8.8	57	33.8	3	49	6.6	41	24.6
54	24.3	58	45.1	4	53	14.2	44	32.1
54	83.7	60	92.7	5	55	40.8	47	58.2
11647		49146		N(5)	160287		641594	
1.00 $\rightarrow$ 0.95				eff.	1.00 $\rightarrow$ 0.70			

Table 4: Iteration count ($I(\epsilon)$, $\epsilon = 10^{-6}$), CPU time (system generation and solution) in seconds for BEM and FEM discretizations and scaled efficiency according to the CPU time and the number of unknowns (N(5)) for the 5th level. on a GC–PowerPlus using one processor per subdomain.

For the results presented in Table 4, the components of the preconditioner have been chosen as follows:

C_C : S–BPX
$C_{I,i}$: V01 (as symmetric Multiplicative Schwarz method)
$B_{I,i}$: HExt.

The BE–matrices were computed fully analytically using piecewise linear functions for the displacements and piecewise constant functions for the tractions [31]. Looking at the results for (D4) and (D5) we observe almost constant iteration numbers with respect to l and p. The lower efficiency in the FEM–case is also due to the higher local problem size for (D5) compared to (D4).

5 Conclusions

We have shown that the preprocessor ADDPre works very well for a wide range of problems including practical problems from industry. The tool allows us to start off with any description of the problem and to realize the decomposition according to the number of processors automatically. Obviously the preprocessing tool is not restricted to DD solvers. It can be applied for parallel solvers based on DD ideas and data structure such as global multigrid methods and global BPX solvers, see [21,22,25], too. The Domain Decomposition methods can be extended to three dimensions. Further, the recursive spectral bisection method is based on graphs and thus it is independent of the problem dimension. Applying a suitable mesh generator (see, e.g. [29]), the complete preprocessing procedure can be extended to three-dimensional problems as well.

Acknowledgment

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.

References

[1] J. H. Bramble, Z. Leyk, and J. E. Pasciak, *The analysis of multigrid algorithms for pseudodifferential operators of order minus one*, Math. Comp., 63 (1994), pp. 461–478.

[2] J. H. Bramble and J. E. Pasciak, *A preconditioning technique for indefinite systems resulting from mixed approximation of elliptic problems*, Math. Comput., 50 (1988), pp. 1–17.

[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] C. Brand, *A modification of the rsb*. Personal communication at a worshop in Graz, Sept. 1995.

[5] C. Carstensen, M. Kuhn, and U. Langer, *Fast parallel solvers for symmetric boundary element domain decomposition equations*, Report 50x, Institute of Mathematics, Johannes Kepler University Linz, 1995.

[6] M. Costabel, *Symmetric methods for the coupling of finite elements and boundary elements*, in Boundary Elements IX, C. A. Brebbia, W. L. Wendland, and G. Kuhn, eds., Springer–Verlag, 1987, pp. 411–420.

[7] ——, *Boundary integral operators on Lipschitz domains: Elementary results*, SIAM J. Math. Anal., 19 (1988), pp. 613–626.

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

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

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

[11] G. Haase, B. Heise, M. Kuhn, and U. Langer, *Adaptive domain decomposition methods for finite and boundary element equations*, in Reports from the Final Conference of the Priority Research Programme Boundary Element Methods 1989–1995, (W.

Wendland ed.), Stuttgart, October 1995, Berlin, 1996, Springer Verlag, pp. xx–xx. Also Technical Report 95–2, Institute of Mathematics, Johannes Kepler University Linz, 1995.

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

[13] G. Haase, U. Langer, and A. Meyer, *A new approach to the Dirichlet domain decomposition method*, in Fifth Multigrid Seminar, Eberswalde 1990, S. Hengst, ed., Berlin, 1990, Karl–Weierstrass–Institut, pp. 1–59. Report R–MATH–09/90.

[14] ——, *The approximate Dirichlet domain decomposition method. Part I: An algebraic approach. Part II: Applications to 2nd-order elliptic boundary value problems.*, Computing, 47 (1991), pp. 137–151 (Part I), 153–167 (Part II).

[15] G. Haase, U. Langer, and A. Meyer, *Domain decomposition preconditioners with inexact subdomain solvers*, J. of Num. Lin. Alg. with Appl., 1 (1992), pp. 27–42. (Preprint 192, TU Chemnitz, 1991).

[16] G. Haase, U. Langer, A. Meyer, and S. V. Nepomnyaschikh, *Hierarchical extension operators and local multigrid methods in domain decomposition preconditioners*, East-West Journal of Numerical Mathematics, 2 (1994), pp. 173–193.

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

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

[19] ——, *Nonlinear simulation of electromagnetic fields with domain decomposition methods on MIMD parallel computers*, Journal of Computational and Applied Mathematics, 63 (1995), pp. 373–381.

[20] ——, *Nonlinear field simulation with FE domain decomposition methods on massively parallel computers*, Surveys on Mathematics for Industry, (1996). To appear.

[21] B. Heise and M. 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] ——, *Robust parallel newton multilevel methods*, (1996). To appear.

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

[24] G. C. Hsiao and W. Wendland, *Domain decomposition in boundary element methods*, in Proc. of IV Int. Symposium on Domain Decomposition Methods, (R. Glowinski, Y. A. Kuznetsov, G. Meurant, J. Périaux, O. B. Widlund eds.), Moscow, May 1990, Philadelphia, 1991, SIAM Publ., pp. 41–49.

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

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

[27] U. Langer, *Parallel iterative solution of symmetric coupled FE/BE- equations via domain decomposition*, Contemporary Mathematics, 157 (1994), pp. 335–344.

[28] S. V. Nepomnyaschikh, *Optimal multilevel extension operators*, Preprint SPC 95_3, Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 1995.

[29] J. Schöberl, *An automatic mesh generator using geometric rules for two and three space dimensions*, Technical Report 95-3, Institute of Mathematics, Johannes Kepler University Linz, 1995.

[30] H. D. Simon, *Partitioning of unstructured problems for parallel processing*, Comput. System in Eng., 2 (1991), pp. 135–148.

[31] O. Steinbach, *Galerkin– und Kollokationsdiskretisierungen für Randintegralgleichungen in 2D*, Report 96–5, University of Stuttgart, 1996.

[32] 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 (1991), pp. 1486–1495.