ELSEVIER

Comput. Methods Appl. Mech. Engrg. 184 (2000) 303–332

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Parallel adaptive subspace correction schemes with applications to elasticity

Michael Griebel, Gerhard Zumbusch *

Institute for Applied Mathematics, University Bonn, Wegelerstr. 6, D-53115Bonn, Germany

Abstract

In this paper, we give a survey on the three main aspects of the efficient treatment of PDEs, i.e. adaptive discretization, multilevel solution and parallelization. We emphasize the abstract approach of subspace correction schemes and summarize its convergence theory. Then, we give the main features of each of the three distinct topics and treat the historical background and modern developments. Furthermore, we demonstrate how all three ingredients can be put together to give an adaptive and parallel multilevel approach for the solution of elliptic PDEs and especially of linear elasticity problems. We report on numerical experiments for the adaptive parallel multilevel solution of some test problems, namely the Poisson equation and Lamé's equation. Here, we emphasize the parallel efficiency of the adaptive code even for simple test problems with little work to distribute, which is achieved through hash storage techniques and space-filling curves.

Keywords: Subspace correction; Iterative methods; Additive and multiplicative Schwarz methods; Multilevel preconditioning; Multigrid methods; Algebraic multigrid; Domain decomposition; Schur complement; Substructuring; Adaptivity; Adaptive grid refinement; Hash storage; Error estimators; Parallelization; Parallel preconditioning; Space-filling curves; Linear elasticity

1. Introduction

Subspace correction methods are techniques for the numerical solution of equation systems which arise in the discretization of partial differential equations. Here, the function space in which the solution is sought in is decomposed into several smaller spaces. The framework of subspace correction schemes contains both classes of optimal and almost optimal order iterative solvers, multilevel and domain decomposition methods. Furthermore, the efficient solution of the boundary value problem requires adaptive grid refinement in order to reduce the number of unknowns of a discretization for a prescribed tolerance. In addition, it requires the use of parallel computers. However, the efficient parallelization of a code with adaptive grid refinement, where workload is created locally during the computation, is difficult. This is especially true in the presence of optimal and almost optimal subspace correction solvers, because the load-balancing and administration overhead must not exceed the ordinary computation time.

In Section 2 we present the abstract framework of subspace correction algorithms with its theory, and we describe multigrid methods and overlapping and non-overlapping domain decomposition methods. We briefly review adaptive grid refinement techniques in Section 3 and in Section 4, we survey parallel multigrid methods and parallel adaptive methods and discuss the difficulties of their parallel implementations. In Section 5 we present some of our own ongoing research. We combine the different approaches and propose a technique based on space-filling curves and hash storage for the implementation of a parallel adaptive multigrid method. We conclude with some numerical experiments for two model problems, i.e. the Poisson

* Corresponding author.
E-mail addresses: griebel@iam.uni-bonn.de (M. Griebel), zumbusch@iam.uni-bonn.de (G. Zumbusch).

0045-7825/00/$ - see front matter
PII: S0045-7825(99)00233-9

equation and the Lamé equation. Note that the efficient parallelization of such a simple and fast problem code is much harder than for more complicated problems where a lot of work is spent with each degree of freedom and the solution of the equation system and where more work can be distributed to the parallel processors.

2. Subspace correction schemes

Let V be some fixed (already finite-dimensional) Hilbert space. The scalar product in V is denoted by $(\cdot,\cdot)$. We consider a linear, elliptic, self-adjoint partial differential equation, which can be written after discretization wrt. V in operator notation as $Au = f$ with $A : V \to V$ denoting the corresponding symmetric, positive, definite (s.p.d.) operator acting on V. The associated standard weak formulation is: Find $u \in V$ such that

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

with positive definite, symmetric bilinear form $a(u,v) = (Au,v)$, $u,v \in V$ and linear functionals $\Phi(v) \equiv (f,v)$.

2.1. Subspace splitting and abstract algorithms

Now, consider an arbitrary additive representation of V by the sum of a finite number of subspaces $V_j \subset V$, which are not necessarily disjoint:

$$V = \sum_{j=0}^{J} V_j. \tag{2}$$

More precisely, this means that any $u \in V$ possesses at least one representation $u = \sum_{j=0}^{J} u_j$ where $u_j \in V_j$ for all $j = 0,\ldots,J$. Suppose that the V_j are equipped with auxiliary continuous s.p.d. forms $b_j(u_j,v_j) = (B_j u_j, v_j)$ given by the s.p.d. operators $B_j : V_j \to V_j$. These forms might model approximative solvers used on the subspaces, i.e. B_j^{-1} is an approximative inverse for A_j, the restriction of A to V_j. We can rewrite problem (1) as: Find $u \in V$ such that

$$Pu = \phi, \quad \left(P \equiv \sum_{j=0}^{J} T_j \;:\; V \to V \right) \tag{3}$$

with the operators $T_j : V \to V_j$ given by the variational problems

$$b_j(T_j u, v_j) = a(u, v_j) \quad \forall v_j \in V_j \tag{4}$$

and $\phi = \sum_{j=0}^{J} \phi_j$ with $\phi_j \in V_j$ defined via the projection

$$b_j(\phi_j, v_j) = \Phi(v_j) \quad \forall v_j \in V_j, \tag{5}$$

which is analogous to (4).

In operator notation $T_j = B_j^{-1} Q_j A$ where $Q_j : V \to V_j$ denotes the orthoprojection onto V_j with respect to the scalar product $(\cdot,\cdot)$. Since

$$P = \sum_{j=0}^{J} T_j = \left(\sum_{j=0}^{J} B_j^{-1} Q_j \right) A \equiv C A, \tag{6}$$

the switching to formulation (3) can be viewed as preconditioning strategy with the preconditioner C for the original problem (1). Note that the formulation (3) is nothing but the additive Schwarz formulation of (1), which has already been treated by many authors, see [64–66,167,171,173] for further references.

Now we turn to the additive and multiplicative variants of the Schwarz iteration associated with the splitting. The *additive* subspace correction algorithm [167] associated with the splitting (2) and its associated auxiliary forms b_j is defined as the Richardson method applied to (3):

$$u^{(l+1)} = u^{(l)} - \omega\big(Pu^{(l)} - \phi\big) = u^{(l)} - \omega \sum_{j=0}^{J} \big(T_j u^{(l)} - \phi_j\big), \quad l = 0, 1, \ldots \tag{7}$$

Here $u^{(0)} \in V$ is any given initial approximation to the solution u of (1) and (3), respectively, and ω is a relaxation parameter. It is easy to see that the associated error propagation operator is $I - \omega \sum_j T_j$. In practice, the additive method serves as a preconditioner and the Richardson method is often replaced by a conjugate gradient iteration.

In contrast to the parallel incorporation of the subspace corrections $r_j^{(l)} = T_j u^{(l)} - \phi_j$ into the iteration (7), the *multiplicative* algorithm uses them in a sequential way:

$$v^{(l+(j+1)/(J+1))} = v^{(l+j/(J+1))} - \omega(T_j v^{(l+j/(J+1))} - \phi_j), \tag{8}$$

with $j = 0, \ldots, J,\ l = 0, 1, \ldots$ Here the corresponding error propagation operator looks like $\prod_j (I - \omega T_j)$. This explains why the algorithm is called 'multiplicative'.

Following [74,75], it is interesting to rewrite the iterations (7) and (8) as follows: We introduce the Hilbert space

$$\tilde{V} = \left\{ \tilde{u} = \{u_j\} : \sum_j b_j(u_j, u_j) \leqslant \infty \right\}, \quad (\tilde{u}, \tilde{v})_{\tilde{V}} = \sum_j b_j(u_j, v_j),$$

which is just the usual Cartesian product of the Hilbert spaces V_j, i.e.

$$\tilde{V} = V_0 \times V_1 \times \cdots \times V_J.$$

We introduce the operator $\tilde{P}$ (here in matrix representation) with respect to the 'coordinate' spaces V_j and decompose it into lower triangular, diagonal and upper triangular parts

$$\tilde{P} = \{T_{i,j}\}_{i,j=0}^{J} = \tilde{L} + \tilde{D} + \tilde{L}^*, \quad T_{i,j} \equiv T_j|_{V_j} : V_j \to V_i, \tag{9}$$

where $L_{j,i} = T_{j,i}$ for $j < i$ and $D_{j,i} = T_{i,i}$ while all other entries are zero operators between the respective subspaces. We further introduce the summation operator $S : \tilde{u} \in \tilde{V} \to u \equiv S\tilde{u} = \sum_j u_j$ which is a linear operator from $\tilde{V}$ onto V. Finally, to any linear continuous functional Φ on V, we associate the elements $\tilde{\phi} \in \tilde{V}$ by defining $\tilde{\phi} = \{\phi_j\}$ with ϕ_j from (5).

Now, we consider the problem: Find $u = S\tilde{u} \in V$ where $\tilde{u} \in \tilde{V}$ is such that

$$\tilde{P}\tilde{u} = \tilde{\phi}. \tag{10}$$

Here, the solution u is unique, while $\tilde{u}$ is not, since different decompositions are allowed. It can be shown that (10) has the same solution u as (1) and (3), see [79] for further details.

If we now define the Richardson (or Jacobi-type) iteration with respect to (10)

$$\tilde{u}^{(l+1)} = \tilde{u}^{(l)} - \omega \cdot (\tilde{P}\tilde{u}^{(l)} - \tilde{\phi}), \quad l = 0, 1, \ldots$$

and the SOR-like iteration

$$(\tilde{I} + \omega\tilde{L})\tilde{u}^{(l+1)} = (\tilde{I} - \omega(\tilde{D} + \tilde{L}^*))\tilde{u}^{(l)} + \omega\tilde{\phi}, \quad l = 0, 1, \ldots$$

then they recover the iterative procedures (7) and (8) in the sense that they satisfy $u^{(l)} = S\tilde{u}^{(l)}$, whenever this relation was fulfilled for the starting iterate, i.e. $l = 0$. A consequence of this observation, which was made in [74,75,78], is that the analysis of the methods (7), (8) can be carried out in almost the same spirit as in the traditional block-matrix situation if one uses the formulation (10). The sometimes tricky proofs, including the original one in [36,39,40,167], (see also the comments on this point in [171]) for the multiplicative case can be made clearer, or, at least, more classical.

2.2. Theory

Now, we define a norm $|||.|||$ on V by

$$|||u|||^2 = \inf_{u_j \in V_j : u = \sum_{j=0}^{J} u_j} \sum_{j=0}^{J} b_j(u_j, u_j), \tag{11}$$

and introduce the positive and finite values

$$\lambda_{\min} = \inf_{u \in V,\, u \neq 0} \frac{a(u,u)}{|||u|||^2}, \quad \lambda_{\max} = \sup_{u \in V,\, u \neq 0} \frac{a(u,u)}{|||u|||^2}. \tag{12}$$

The quantity

$$\kappa \equiv \frac{\lambda_{\max}}{\lambda_{\min}} \tag{13}$$

will be called condition number of the splitting. It can be seen easily that it is equivalent to the condition number of CA of (6). The spectrum of the operator $P = \sum_{j=0}^{J} T_j$ is given by the constants from (12) compare for example [25,111,164,173] and see also the fictitious space lemma in [119,120]. Note that the condition number does not change if we change the order of the subspaces.

The following results, which also explain the central role of the above splitting concept, are derived from [78,80]; see [91] for statements of this type in the matrix case.

Theorem 1 (Additive and multiplicative Schwarz iteration). *Suppose that V is finite-dimensional, and that the splitting is finite. Let the characteristic numbers $\lambda_{\max}, \lambda_{\min}$, and κ of the splitting be given by* (12) *and* (13). *Furthermore, let $\|\tilde{L}\|_{\tilde{V} \to \tilde{V}}$ be the norm of the lower triangular matrix operator $\tilde{L}$ (cf. (9)) as an operator in $\tilde{V}$, let $\omega_1 \equiv \lambda_{\max}(\tilde{D}) = \max_{j=0,\ldots,J} \{\max_{u_j \in V_j} a(u_j,u_j)/b_j(u_j,u_j)\}$ and let $\tilde{W} := 1/\omega\, \tilde{I} + \tilde{L}$.*

(*a*) *The additive method* (7) *converges for $0 < \omega < 2/\lambda_{\max}$, with the convergence rate*

$$\rho_{as} = \max\{|1 - \omega\lambda_{\min}|, |1 - \omega\lambda_{\max}|\}. \tag{14}$$

The bound in (14) *takes the minimum*

$$\rho_{as}^* = 1 - \frac{2}{1+\kappa} \quad \text{for } \omega^* = \frac{2}{\lambda_{\max} + \lambda_{\min}}. \tag{15}$$

(*b*) *The multiplicative method* (8) *converges for $0 < \omega < 2/\omega_1$, with a bound for the asymptotic convergence rate given by*

$$\rho_{ms} \leqslant \sqrt{1 - \frac{\lambda_{\min}(2/\omega - \omega_1)}{\|\tilde{W}\|^2_{\tilde{V} \to \tilde{V}}}} \leqslant \sqrt{1 - \frac{\lambda_{\min}(2/\omega - \omega_1)}{(1/\omega + \|\tilde{L}\|_{\tilde{V} \to \tilde{V}})^2}}. \tag{16}$$

The bound in (16) *takes its minimum*

$$\rho_{ms}^* \leqslant \sqrt{1 - \frac{\lambda_{\min}}{2\|\tilde{L}\|_{\tilde{V} \to \tilde{V}} + \omega_1}} \quad \text{for } \omega^* = \frac{1}{\|\tilde{L}\|_{\tilde{V} \to \tilde{V}} + \omega_1}. \tag{17}$$

For a proof of this Theorem, see [79]. Note that using the representation (10), the proofs are exactly the same as for the traditional Richardson iteration and the SOR iteration, compare e.g. [91, pp. 82–96]. Note also that, even in the case of divergence, the additive subspace correction serves as an optimal preconditioner as long as κ is independent of J.

Now, we see a difference between the additive and multiplicative method. For the additive method, basically the terms $\lambda_{\min}$ and $\lambda_{\max}$ enter the convergence rate estimates whereas for the multiplicative method the terms $\lambda_{\min}$ and $\|\tilde{L}\|_{\tilde{V} \to \tilde{V}}$ are involved. Note that $\|\tilde{L}\|_{\tilde{V} \to \tilde{V}}$ depends on the ordering of the successive subspace corrections.

Without any additional assumption, there exists an estimate of $\|\tilde{L}\|_{\tilde{V}\to\tilde{V}}$ in terms of $\lambda_{\max}$

$$\|\tilde{L}\|_{\tilde{V}\to\tilde{V}} \leqslant \frac{1}{2}[\log_2(2J)]\lambda_{\max}. \tag{18}$$

This links the estimates of the convergence rate of the multiplicative method to that of the additive method. This estimate is sharp in the general case, for the proof and further details see [79,128].

However, for certain splittings, $\|\tilde{L}\|_{\tilde{V}\to\tilde{V}}$ can be estimated from above by a constant independent of J. In [167,173], for example, additional assumptions, where basically strengthened Cauchy–Schwarz inequalities have to be fulfilled, allow to estimate both $\lambda_{\max}$ and $\|\tilde{L}\|_{\tilde{V}\to\tilde{V}}$ from above by the common upper bound $\| |\tilde{P}| \|_{\tilde{V}\to\tilde{V}}$ where $|.|$ denotes element-wise absolute values, i.e.

$$\left.\begin{matrix}\|\tilde{L}\|_{\tilde{V}\to\tilde{V}} \\ \lambda_{\max}\end{matrix}\right\} \leqslant \| |\tilde{P}| \|_{\tilde{V}\to\tilde{V}} \leqslant \text{const}.$$

Note that $\| |\tilde{P}| \|_{\tilde{V}\to\tilde{V}}$ is an upper bound for *all* possible $\|\tilde{L}\|_{\tilde{V}\to\tilde{V}}$ that arise for all possible traversal orderings of the multiplicative scheme. Now, the convergence rate of the multiplicative method as well as the additive method is independent of J, if, in addition, the splitting has the property that $\lambda_{\min}$ can be estimated from below by a positive constant which is independent of J.

Now the general aim is to find splittings $\sum_{j=0}^{J} V_j$ with associated bilinear forms $b_j(.,.)$ such that

- $\lambda_{\min}, \lambda_{\max}, \|\tilde{L}\|_{\tilde{V}\to\tilde{V}}$ or, alternatively, $\| |\tilde{P}| \|_{\tilde{V}\to\tilde{V}}$ can be estimated independently of J,
- the additive or multiplicative method can be implemented using $O(\dim(V))$ operations per iteration step only,
- and hopefully the terms above are independent of the ellipticity constants of the operator, i.e. robustness with respect to the variations of the coefficients of the PDE is gained.

Note that the first two conditions can be fulfilled by e.g. multigrid or multilevel methods whereas the robustness question is still not satisfactorily settled.

2.3. Multigrid methods

The article [69] by Fedorenko is usually considered as the beginning of the history of multigrid methods. [1] A two-grid method and later a multi-grid method [70] was proposed and analyzed, namely a W-cycle with pre- and post-smoothing using damped Jacobi iterations for the Poisson problem on the unit square. A variable coefficient problem was analyzed by [7], structured triangle grids were considered in [2] and the multigrid method was further developed to general grids in [121]. The interest in multigrid methods was mainly theoretical at that time and was focused on the optimal complexity of the algorithms. Real applications were first considered by Brandt [42], who observed the computational efficiency of multigrid methods. He optimized the components of the algorithms [45] and applied them to adaptively patch-wise refined grids [42] and to non-linear problems (FAS scheme, see [43]). Independently, Hackbusch developed the multigrid method in [86,87] and applied it to variable coefficient, general, second order, nine-point stencil discretizations on arbitrary domains. The analysis of the multigrid method was continued for finite difference stencils in [162] and for more general error norms in [8] leading to what is now called the classical multigrid theory: An abstract convergence proof based on an approximation and a smoothing property was established in [88]. The first V-cycle convergence result for the Poisson equation was proved in [31]. Furthermore, the development was summarized in the book [90], see also the conference proceedings series [53,118] and the monographs [114,163].

As an example, we consider the two-level algorithm. Here, we have two spaces, the fine grid space V_J of piecewise linear functions on the uniform grid Ω_J over $\Omega = [0,1]^d$ with mesh size 2^{-J} and the coarse grid space V_{J-1} of piecewise linear functions on the uniform grid Ω_{J-1} with mesh size $2^{-(J-1)}$. A mapping between these spaces is given by the so-called prolongation operator $P_{J-1}^{J} : V_{J-1} \to V_J$ which resembles linear interpolation, and the so-called restriction operator $R_J^{J-1} : V_J \to V_{J-1}$ (f.e. the adjoint of P_{J-1}^{J}). Discretization on level J results in the system $A_J u_J = f_J$ and discretization on level $J-1$ gives the stiffness matrix A_{J-1}.

[1] For the solution of integral equations Brackhage [30] proposed a two-grid method even earlier.

Now an iteration step of the two-level method consists of the following two parts: First, apply v steps of a so-called smoother, i.e. a classical (convergent) iteration on level J

$$u_J^{it+i/(2v)} = u_J^{it+(i-1)/(2v)} - C_J\left(A_J u_J^{it+(i-1)/(2v)} - f_J\right) \quad \text{for } i = 1, \ldots, v \tag{19}$$

like f.e. a Jacobi iteration, where $C_J = \mathrm{diag}(A_J)^{-1}$. Alternatively, also Gauss–Seidel, SOR or many other iterative schemes can be used here. Second, apply a coarse grid correction step where first the residual is formed and transported to the coarser level by the restriction operator, then the associated coarse level problem is solved exactly and third, using the prolongation operator, this coarse grid solution is used to update the fine grid iterate, i.e.

$$u_J^{it+1} = u_J^{it+1/2} - P_{J-1}^J A_{J-1}^{-1} R_J^{J-1}(A_J u_J^{it+1/2} - f_J). \tag{20}$$

Of course the two-level method can be used recursively instead of A_{J-1}^{-1} in the coarse grid correction step (20). This results in a general multigrid method. Here a variety of cycling strategies have been developed (V-cycle, W-cycle, compare [90]).

Now, subtracting the exact solution u_J of level J from (19) and (20), using the relation $A_J u_J = f_J$ and plugging (19) into (20) results in the error equation

$$e_J^{it+1} = (I_J - P_{J-1}^J A_{J-1}^{-1} R_J^{J-1} A_J)(I_J - C_J A_J)^v e_J^{it}. \tag{21}$$

Note that $(I_J - P_{J-1}^J A_{J-1}^{-1} R_J^{J-1} A_J)(I_J - C_J A_J)^v = (A_J^{-1} - P_{J-1}^J A_{J-1}^{-1} R_J^{J-1}) A_J (I_J - C_J A_J)^v$ holds. Then, using the so-called *smoothing property*

$$\|A_J(I_J - C_J A_J)^v\| \leqslant \eta(v)\|A_J\| \quad \text{for all } v \in \mathbb{N}_0 \tag{22}$$

with $\lim_{v\to\infty} \eta(v) = 0$ and the so-called *approximation property*

$$\|A_J^{-1} - P_{J-1}^J A_{J-1}^{-1} R_J^{J-1}\| \leqslant C/\|A_J\|, \tag{23}$$

we get an upper estimate for the convergence rate of the two-level method (21) by $C/\|A_J\|\,\eta(v)\|A_J\| = C\eta(v)$, where C and $\eta(v)$ are independent of J. Now, for a given $0 < \xi < 1$ there exists a lower bound $\bar{v}$ such that $C\eta(v) < \xi$ for all $v \geqslant \bar{v}$. Of course, the validity of the smoothing property and the approximation property has to be shown. This basic approach can be generalized to various settings using different components in the multigrid procedure, for details we refer to [90]. Note that for the approximation property (23) to hold, often an additional assumption on the regularity of the continuous problem under consideration, i.e. H^2-regularity, is needed.

Regularity free proofs, proofs for non-uniform grids and for the additive variant of the multigrid method were obtained differently. A proof for the hierarchical basis preconditioner and for the hierarchical basis multigrid method was given in [170] and [9]. However, the hierarchical basis does not lead to optimal methods for dimension $d \geqslant 2$. Almost regularity free proofs for the additive and the multiplicative multilevel methods were given in [41,166]. However, there, only sub-optimal convergence rates dependent on J could be shown. Then, for the additive methods, the first proof of an optimal O(1) condition number was given in [124] without any additional regularity assumptions. Proofs for non-uniform grids [57] and for multiplicative multigrid [40,173] followed soon. Further details can be found in [35,75,127,167,171].

With respect to the convergence theory of Section 2.2, these multilevel and multigrid methods are based on a nested sequence of subspaces $V_0 \subset V_1 \subset \cdots \subset V_J$, where the spaces V_j are chosen as piecewise linear functions on the corresponding sequence of nested grids $\Omega_0 \subset \cdots \subset \Omega_J$. In other words, we have a levelwise splitting

$$V = \sum_{j=0}^{J} V_j,$$

where the associated auxiliary forms b_j are given by the smoothers on level j. This splitting can be further decomposed into a splitting of one-dimensional subspaces

$$V = \sum_{j=0}^{J} V_j = \sum_{j=0}^{J} \sum_{i} V_{j,i}, \tag{24}$$

where $V_{j,i} = \mathrm{span}(\phi_{j,i})$ and $\phi_{j,i}$ denote the usual linear basis functions on grid Ω_j. Here the auxiliary forms can even be chosen as $b_{j,i} = a_{j,i}$. Now, the additive subspace correction method (7) results in the BPX-preconditioner [41] whereas the multiplicative subspace correction method (8) is basically equivalent to a multigrid method with Gauss-Seidel pre- or post-smoothing, if a level-wise traversal is chosen. With respect to the convergence theory of Section 2.2 it can be shown that our multilevel splitting (24) results in values for $\lambda_{\min}$, $\lambda_{\max}$, $\|\tilde{L}\|_{\tilde{V}\to\tilde{V}}$ or, alternatively, $\| |\tilde{P}| \|_{\tilde{V}\to\tilde{V}}$, which can be estimated independently of J, see also [28,74,75,127,167,171,173].

Thus, in this respect, optimality is achieved. However the constants do depend on the coefficient functions of the underlying operator, i.e. they depend on the ellipticity constants. Therefore, in practice, the convergence rate is still dependent on the coefficient functions of the underlying operator. Now, the question arises, whether there is a multilevel method with a convergence rate independent of the coefficients, which is *robust* in this respect. To achieve robustness various modifications of the conventional multigrid scheme have been introduced: First of all, modifications of the smoother have been proposed, i.e line- or zebra-smoothers [155] or ILU type smoothing procedures [154,165]. Furthermore, instead of modifying the smoother, also the coarse grid spaces can be modified by using semi-coarsening, matrix dependent prolongations and restrictions [71,137,172] or even specific algebraic coarsening techniques, i.e. so called algebraic multigrid methods, see [3,46,73,143] and [32,158] for the case of elasticity. Further modifications have been proposed to deal with complex geometries, where nested hierarchies of grids are not available, see [12,92,100] in addition to the algebraic multigrid methods. These approaches work fine and apparently robust in practical experiments. However, a proof of robustness for algebraic multigrid and similar variants do not exist. For simple separable self-adjoint operators, however, a robust additive multilevel method based on pre-wavelets has been presented and proved rigorously in [80]. Note however that for the general case in three dimensions it is still a question how a robust multigrid method could be constructed.

Note that multigrid methods have been applied for problems in elasticity using conforming, non-conforming and mixed discretizations [95] of different plate bending models [125,132,174], shell models [130] and for plane elasticity [15,34,48,104,176]. However, in the case of elasticity applications, the robustness of a multigrid method is also subject to actual research. Here the convergence rates are of course independent of the number of unknowns, however, they depend on anisotropies of the material constants, on the aspect ratio of the domain Ω, on the Poisson number, on locking phenomena of a discretization, etc.

Note that non-linear equations can be treated using the multigrid approach as well: Either a Newton iteration can be used as an outer iteration with a linear multigrid method or alternatively non-linear multigrid versions (Brandt's FAS [43] and Hackbusch's non-linear multigrid [90]) with Picard or Newton type smoothers can be used, for a comparison see [93]. Further non-linear subspace corrections methods including theory can be found in [156,168].

2.4. Domain decomposition techniques

The application of the well known 'divide et impera' principle leads in a natural way to domain decomposition techniques. Here the domain is partitioned into sub-domains, on which the arising small problems can be solved easily. Then the local solutions have to be combined and processed furthermore to obtain the solution of the overall problem. The partitioning can be given in a natural way, f.e. by the geometry or the material coefficients of the problem. It can also be heuristically constructed by grid partitioning techniques. In addition a domain decomposition is advantageous for later parallelization. Based on this principle a series of algorithms have been developed, i.e. overlapping Schwarz methods, non-overlapping Schur complement methods, etc. The development of these methods up to now can be found in the conference proceedings [59], see also the surveys [51,148,169].

2.4.1. Overlapping Schwarz methods

The first domain decomposition methods for the solution of PDEs were the overlapping Schwarz methods, named after Schwarz [145]. They were used for the existence proof of a continuous solution of the Poisson equation in a domain, which was composed of the union of two overlapping standard domains, see [4,107,117,150] for further early references. In the general case, the basic idea is to construct the domain Ω as the union of several sub-domains $\Omega = \bigcup_j \Omega_j$ with overlap $\Omega_j \cap \bigcup_{i \neq j} \Omega_i \neq \emptyset$ of positive diameter along inner boundaries $\partial\Omega_j \setminus \partial\Omega$. Discretizations on the overlapping domains usually are based on matching grids: The elements or grid cells of two sub-domains Ω_i and Ω_j coincide within the overlap $\Omega_i \cap \Omega_j$. Thus the overlap has to be at least one element wide. The corresponding spaces $\hat{V}_j$ are constructed by a discretization on the sub-domain Ω_j with Dirichlet boundary conditions on the inner boundary $\partial\Omega_j \setminus \partial\Omega$ along with the bilinear form $a(.,.)$ restricted to Ω_j. The splitting looks like

$$V = \sum_j^J \hat{V}_j,$$

where the associated auxiliary forms b_j are given by exact or inexact sub-domain solvers on Ω_j. The associated iterative methods (7) and (8) lead to the additive and multiplicative Schwarz method, which can be analyzed with the convergence theory of Section 2.2. The convergence rate depends on the size of the overlap and on the number of sub-domains, see Table 1.

Furthermore, the introduction of a coarse grid removes the latter dependence and facilitates immediate information transport between pairs of non-overlapping sub-domains, in analogy to the two-level multigrid method. For example for a Poisson problem, a coarse grid with one degree of freedom per sub-domain is sufficient for this purpose. The coarse space V_{coarse} can be chosen as the span of piecewise constant or piecewise linear functions on the sub-domain scale, where each function is associated to one sub-domain. We denote the mesh size of the coarse grid by H and denote the size of the overlap by βH to indicate that the overlap should be chosen independently of the fine mesh size h. Now, the splitting is

$$V = V_{\text{coarse}} + \sum_{j=0}^J \hat{V}_j,$$

Table 1
Convergence rates of domain decomposition methods for constant coefficient problems with a coarse grid of mesh size H and a fine grid mesh size h (the value βH denotes the grid overlap for the overlapping Schwarz method and the diameter of the vertex spaces for the vertex space method, respectively)

Algorithm		Convergence rate	Subspace splitting
Overlapping Schwarz iteration			
Additive Schwarz		$CH^{-2}(1+1/\beta^2)$	$V = \sum_j \hat{V}_j$
– with coarse grid		$C(1+1/\beta)$	$V = V_{\text{coarse}} + \sum_j \hat{V}_j$
Schur complement iteration			
Nodal basis		$CH^{-1}h^{-1}$	$V = V_\Gamma + \sum_j V_j$
Hierarchical basis	$d=2$	$C(1+\log^2(H/h))$	$V = V_\Gamma^{\text{hb}} + \sum_j V_j$
	$d=3$	$CH/h(1+\log^2(H/h))$	$V = V_\Gamma^{\text{hb}} + \sum_j V_j$
Multilevel basis		C	$V = V_\Gamma^{\text{BPX}} + \sum_j V_j$
Preconditioner for the Schur complement iteration			
BPS	$d=2$	$C(1+\log^2(H/h))$	$V_\Gamma = V_{\text{coarse}} + \sum_k V_{\text{edge}\,k}$
Wirebasket	$d=3$	$C(1+\log^2(H/h))$	$V_\Gamma = V_{\text{coarse}} + \sum_k V_{\text{edge}\,k} + \sum_l V_{\text{face}\,l}$
Vertex space	$d=2$	$C(1+\log^2\beta)$	$V_\Gamma = V_{\text{coarse}} + \sum_k V_{\text{edge}\,k} + \sum_m V_{\text{vertex}\,m}$
	$d=3$	$C(1+\log^2\beta)$	$V_\Gamma = V_{\text{coarse}} + \sum_k V_{\text{edge}\,k} + \sum_l V_{\text{face}\,l} + \sum_m V_{\text{vertex}\,m}$
Numann–Neumann		$CH^{-2}(1+\log^2(H/h))$	$V_\Gamma = \sum_j V_j^{\text{Neumann}}$
– with coarse grid		$C(1+\log^2(H/h))$	$V_\Gamma = V_{\text{coarse}} + \sum_j V_j^{\text{Neumann}}$

where the new associated form is usually chosen as $b_{\text{coarse}}(.,.) = a_{\text{coarse}}$, i.e. the restriction of the bilinear form $a(.,.)$ to V_{coarse}. This approach results in a convergence rate which is independent of h. However, overlapping methods in general are not robust with respect to discontinuous coefficients.

2.4.2. Non-overlapping Schur complement methods

The history of non-overlapping domain decomposition methods begins with direct substructuring methods. The domain Ω is partitioned into two disjoint sub-domains Ω_1 and Ω_2. Then, the separator $\Gamma = \partial\Omega_1 \setminus \partial\Omega$ is removed and the two local sub-problems are solved before we solve for the unknowns on the separator. This scheme can be applied recursively to the sub-problems and is then equivalent to a Gaussian elimination with nested dissection ordering. Early references for this method, which is popular in structural mechanics, can be found in [136]. In case of the simple partitioning into two sub-domains the stiffness matrix is block partitioned correspondingly, i.e.

$$A = \begin{pmatrix} A_{\Gamma\Gamma} & A_{\Gamma 1} & A_{\Gamma 2} \\ A_{1\Gamma} & A_{11} & 0 \\ A_{2\Gamma} & 0 & A_{22} \end{pmatrix}.$$

Now, iterative substructuring methods are based on the iterative solution of the Schur complement system $S = A_{\Gamma\Gamma} - \sum_{i=1}^{2} A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma}$ for the unknowns on the separator Γ. In some algorithms also the sub-domain problems A_{ii}^{-1} are solved iteratively.

The two domain case can easily be generalized to the many domain case. Here Ω is partitioned into a union of disjoint sub-domains $\Omega = \bigcup_j \overline{\Omega_j}$ without overlap $\Omega_j \cap \Omega_i = \emptyset$ for $i \neq j$. We use grids for the Ω_j, which match at the interface

$$\Gamma = \bigcup_j \partial\Omega_j \setminus \partial\Omega.$$

The discretizations in the sub-domains Ω_j also have to match. Otherwise, we have to use more sophisticated methods. In addition to the degrees of freedom in the sub-domains, there exist degrees of freedom on the interface Γ. Now, the space V_j consists of functions which vanish outside the open domain Ω_j. The interface space V_Γ is defined as the span of shape functions of the interface grid Ω_Γ, which are extended to the adjacent sub-domains Ω_j in some way and vanish on the boundary $\partial\Omega$. Then, the subspace splitting according to the convergence theory of Section 2.2 looks like

$$V = V_\Gamma + \sum_{j=0}^{J} V_j.$$

We choose the associated auxiliary forms as $b_j(.,.) = a_j(.,.)$ and use a multiplicative Schwarz method. The choice of the Poincaré–Steklov operator $b_\Gamma(x,y) \equiv x^{\mathrm{T}} S y$ with exact solvers would lead to the direct substructuring method again. However, the Schur complement is expensive to compute and is not directly accessible in an iterative procedure. Hence we choose the auxiliary form on the separator as the L_2 scalar product $b_\Gamma(.,.) = (.,.)$ along with one Richardson iteration step. A conjugate gradient method in V with this preconditioner is equivalent to a conjugate gradient method applied to the smaller Schur complement system on V_Γ. The condition number of this preconditioner is $CH^{-1}h^{-1}$ with a mesh size h of Ω and a sub-domain diameter of H, if as a basis for V_Γ just the nodal basis is chosen, see also Table 1.

In general there are two ways to improve the condition number further: We can extend the functions of V_Γ in a more clever way by the hierarchical basis [83,149] or a multilevel technique [76,85,157]. This results more or less in a harmonic extension operator, which couples the interior of the sub-domains and the separator, see also Table 1. Alternatively we can construct preconditioners for the interface Schur complement, which results in improved forms $b_\Gamma(.,.)$, which we will consider in the next section. Further modifications include the use of approximative solvers $b_j(.,.)$ on the sub-domains, see [84].

2.4.3. Preconditioners for the Schur complement

We are interested in finding a preconditioner for the interface problem in the space V_Γ. Such a preconditioner can be used in an iteration on the Schur complement S or in the construction of a global

preconditioner for the operator A. Historically, the first domain decomposition preconditioners for the Schur complement were designed for the two domain case only. Here, implementations of the analytic preconditioner for $H^{1/2}(\Gamma)$ of Dryja [63] or the improved version [72] were based on the sine transform. The next step was to construct preconditioners for the case of many sub-domains. Then, crosspoints are contained in the separator. Bramble, Pasciak and Schatz were able to treat this case with crosspoints [37]. This algorithm and its three-dimensional generalization, the wirebasket preconditioner [38,147] are based on a splitting of the space

$$\begin{aligned} V_\Gamma &= V_{\text{coarse}} + \sum_k V_{\text{edge}\,k} \quad \text{for } d = 2, \text{ and} \\ V_\Gamma &= V_{\text{coarse}} + \sum_k V_{\text{edge}\,k} + \sum_l V_{\text{face}\,l} \quad \text{for } d = 3, \text{ resp.}, \end{aligned} \tag{25}$$

into one global coarse grid space of crosspoints and into local spaces for each edge or face $\subset \partial\Omega_i \setminus \partial\Omega \subset \Gamma$, which connects crosspoints or crosspoints with the boundary $\partial\Omega$. The associated bilinear form of the coarse grid is $b_{\text{coarse}}(.,.) = a_{\text{coarse}}(.,.)$ and the coarse system is solved directly. However, the bilinear forms of the edges $b_{\text{edge}\ k}$ (and the faces $b_{\text{face}\ l}$) are not available, similar to the Schur complement itself. Hence other Schur complement preconditioners have to be employed for the spaces $V_{\text{edge}\,k}$ and $V_{\text{face}\,l}$, such as the mentioned preconditioner of Dryja. The resulting convergence rates can be found in Table 1.

An extension of this construction principle leads to the vertex space method [146]. It is related to the fictitious domain methods in [111,119,120] and [1]. In addition to the splitting in Eq. (25), for each vertex, a space is constructed with consists of functions on the separator Γ in the vicinity of the vertex x_m, i.e.

$$\Omega_{\text{vertex}\,m} = \mathbb{B}_{\beta H}(x_m) \cap \Gamma,$$

where $\mathbb{B}_{\beta H}(x_m)$ denotes the ball around x_m with radius βH.

A different class of Schur complement preconditioners is based on the solution of auxiliary problems on the sub-domains. The Neumann–Dirichlet preconditioner [26] is based on the solution of a Neumann problem on half of the sub-domains, and in a second step, on Dirichlet problems on the remaining sub-domains. A popular variant of it is the Neumann–Neumann preconditioner [29,103]. There, on each sub-domain a local Neumann problem is solved. Then, the actual right-hand side in V_Γ is transformed into a Neumann boundary condition. Finally, the values on $\partial\Omega_j \cap \Gamma$ are transformed back and summed up. This approach can be interpreted in our additive Schwarz framework. We end up with a splitting of the type

$$V_\Gamma = \sum_j V_j^{\text{Neumann}}$$

with associated bilinear forms $b_j(.,.) = a_j(.,.)$ with appropriate boundary conditions incorporated. There exist variants with a coarse grid to improve the performance in the many sub-domain case [67,102,110]:

$$V_\Gamma = V_{\text{coarse}} + \sum_j V_j^{\text{Neumann}}.$$

More recent developments are concerned with the development of analogous preconditioners for discretizations, where the continuity between sub-domains Ω_j is maintained through Lagrange multipliers, called the FETI [68] and the Mortar [19,109] methods. The basic advantage of non-overlapping domain decomposition is their robustness with respect to discontinuous coefficients aligned to sub-domain boundaries. In many large-scale problems it is possible to partition the domain Ω into sub-domains of constant or slowly varying coefficients such that each sub-domain consists of exactly one material. Based on this decomposition, the resulting preconditioner is often competitive to other methods with better theoretical properties.

3. Adaptivity

Let us assume that the discretization is of order $O(h)$ in the energy norm, see [49]. This means that the discretization error ϵ for $H^2(\Omega)$ regular solutions converges linearly with $h \to 0$. However, in the case of

singularities, the observed convergence can be much slower, i.e. $O(h^\alpha)$ with $0 < \alpha < 1$. Then, instead of $n = \text{tol}^{-d}$ unknowns we have to use at least $n = \text{tol}^{-d/\alpha}$ unknowns on a uniform grid for a given error tolerance tol. A remedy to this explosion of the number of unknowns is to introduce grids with locally varying mesh size $h(x)$, which is adapted to the solution. Sometimes a priori error estimates can be used to construct such a grid. Before a computation is performed, a priori information is fed into the grid generation process. However, often such information is not available or sufficient.

3.1. Error control and adaptive refinement

Alternatively, a feedback approach can be used: The PDE is solved several times on a sequence of grids. Here, the most recent solution is used to guide the construction of the next grid, where an a posteriori error estimator $\eta(x)$ estimates the discretization error $\epsilon(x)$ of the most recent solution. The discrete version of the error estimator η_i is related to either elements, nodes, edges, or faces of the grid. It should be connected to the local errors ϵ_i by

$$c\,\epsilon_i \leqslant \eta_i \leqslant C\,\epsilon_i. \tag{26}$$

An analysis of the local amount of work and the local discretization error reveals that the optimal grid has equi-distributed local errors ϵ_i. Furthermore, the optimal grid consists of $n = C_\alpha \cdot \text{tol}^{-d}$ unknowns, which is equivalent to a convergence rate independent of α, see [6]. Hence, the grid is refined at locations with large estimated error η_i. This can be interpreted as an optimization process to achieve an equi-distribution of the η_i. Eq. (26) guarantees the efficiency of the process. Neither regions with large errors are missed, nor unnecessary refinement is performed. Although Eq. (26) can only be proved under restrictive assumptions (sufficient regularity, saturation, etc.), asymptotically for a grid size $h \to 0$ small enough, such bounds can be showed for many popular error estimators, see [159].

Construction of early error indicators and error estimators was purely heuristic. Here usually simple local gradients were employed. In [5,6,11] local problems were set up and solved. This approach led to residual-based, Dirichlet-based and Neumann-based local error indicators. Alternatively also local subproblems with higher order discretization were used to construct error indicators [54]. A modern, rigorous theory for the construction of local and global error estimators, which is based on the dual problem approach can be found in [16,17]. Note furthermore that the norm equivalence (11) can also be used to derive error estimators, for details see [127], Ch. 5, and [126]. Note finally that for simple elliptic operators it is possible to get rid of the so-called saturation assumption, which is usually a prerequisite in the construction and analysis of error estimators, see e.g. [55,56,60].

The error estimator η_i gives an estimate for the global discretization error ϵ

$$\epsilon \approx \sum_i \eta_i =: \tilde{\epsilon}.$$

This value $\tilde{\epsilon}$ can be used as a termination criterion of the adaptive refinement cycle, see Fig. 1 and [16]. In the absence of error estimators, error indicators can take the role of an error estimator. The termination

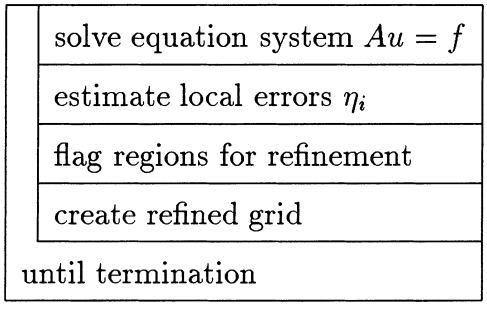

Fig. 1. The adaptive algorithm.

criterion can also be based on other data, e.g. computer resources such as memory or computing time available.

3.2. Grid refinement

The grid can be refined by a complete new *re-meshing* of the domain Ω guided by the error density η_i, by *overlaying patches* of finer mesh size h over regions of refinement, or by local *element-wise* grid refinement, where some elements are substituted by other, smaller elements.

Re-meshing is expensive and details of the most recent solution can be lost due to the refinement. Furthermore, it is difficult to apply multigrid methods. Patch-wise refinement gives a nested sequence of grids. It can be used for multigrids in a natural way, see Fig. 2 left. Specialized multigrid methods have been developed for this type of composite grids, see [42,89,113]. Furthermore, on the patches, which are structured uniform grids, all operations can be implemented very efficiently.

Element-wise refinement can be performed for several element types and in several ways: First we consider quadrilateral and hexahedral elements. A quadrilateral can be bisected and substituted by two quadrilaterals of half the area or it can be substituted by four similar quadrilaterals of one fourth the area, see [138] and Fig. 3. Analogously hexahedra can be bisected or cut into eighth parts. A bisection strategy should take care that the elements decrease in size along all coordinate directions. We need to control the maximal diameter of the elements $h_{\max}$ for convergence by alternating the direction of the bisection. A subdivision of an element into 2^d elements does not affect the aspect ratio of the elements and gives local $h_{\max} \to 0$ anyway. In case of local refinement, the resulting grids contain hanging nodes where elements of different size are joined. Hanging nodes represent a constraint for a conforming discretization, which can be eliminated from the equation system. They are not degrees of freedom.

Next we consider triangular and tetrahedral elements. There are algorithms, which provide element-wise refinement without hanging nodes. Grids based on triangular and tetrahedral elements can be constructed

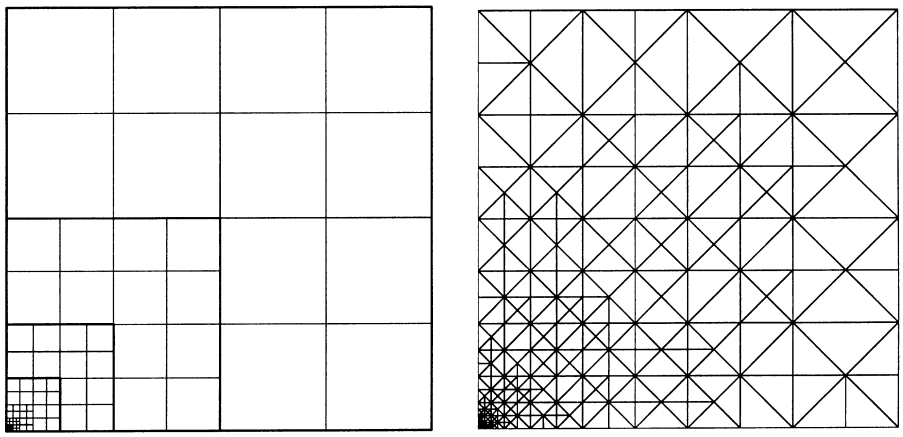

Fig. 2. Adaptively refined grids. Patch-wise refinement (left) and element-wise refinement (right).

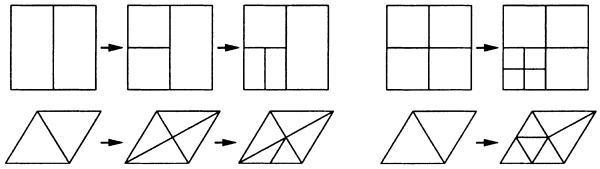

Fig. 3. Element-wise grid refinement with quadrilateral and triangular elements. Bisection (left) and subdivision into four elements (right).

for arbitrary polygonal shaped domains and allow for easier representations of complicated geometries Ω. The idea for grid refinement is to bisect elements or to subdivide them into 2^d elements as above, see Fig. 3. In addition to this refinement which is triggered by an error estimate, further refinement is used to eliminate hanging nodes and to smooth the refined grid. This can be done along some geometric rules. A major concern here is to maintain angle conditions of the grid for reasons of the discretization error. Here, a bounded minimal interior angle ϕ

$$0 < c < \phi < C < \pi \tag{27}$$

or at least a bounded maximal interior angle independent of the mesh size h is desired. For the interpolation properties of the FEM however, the maximum angle condition is sufficient. The minimum angle condition is necessary for the performance of iterative solvers. If one bisects edges opposite to one node too often, the minimum angle condition would be violated. Several bisection strategies such as 'longest edge' and 'newest node' have been developed so far which choose the edge for bisection in a way to guarantee such bounds [115,140]. The subdivision of an element into 2^d elements, 'red refinement', also requires some bisection, 'green refinement', although the subdivision of a triangle itself does not introduce any dangerous angles. This is due to the removal of hanging nodes, which triggers a sequence of element bisections, see [10]. The subdivision of tetrahedral elements into 2^d elements requires a more complicated, stable subdivision scheme, because at least four of the resulting tetrahedra cannot be similar to the original tetrahedron, see [21,22] in the general case.

Finally, there exist many more variations of grid refinement algorithms and rules. There are rules for the refinement of mixed grids consisting of several element types such as triangles and quadrangles in two dimensions and tetrahedra, hexahedra, prisms and pyramids in three dimensions. Furthermore isoparametric distorted quadrangles can be used to eliminate hanging nodes and create a closure of a grid containing refined quadrangles.

3.3. Grid storage and management

The storage and the management of structured grids usually does not require very sophisticated data structures. Ordinary vectors and arrays are sufficient to access an element, a node or its neighbors. Ordinary loops can cycle through all nodes. A multigrid method can be based on a sequence of grids, with a straightforward implementation of inter-grid transfers.

Unstructured grids in contrast require a more flexible representation of the geometric entities and their connectivity. For conforming linear finite elements the degrees of freedom are located at the nodes. The entries of the stiffness matrix are related to edges (and nodes) and the quadrature for the assembly of the stiffness matrix has to be performed on every element. Hence commonly data structures are based on several vectors or linked lists of nodes and edges and elements and their connectivity.

The efficient solution of equation systems by multigrid methods on hierarchies of adaptively refined unstructured grids has been considered e.g. in [13,101,105,108,115,140,142,152]. A multigrid method for the solution of the equation system requires a hierarchy of grids. Data structures of linked lists of elements do not provide a hierarchy but contain the elements of the finest grid in an arbitrary order. One way to include grid hierarchies is to switch to tree data structures. All elements of all grids are stored in one tree as well as edges and nodes [138,142]. A subset of the elements, nodes and edges forms one grid. All nodes together with some of the elements form the finest grid, where a solution is sought. Coarser level grids of level l can be constructed as the union of the elements of tree level l. Alternatively they can be chosen as the grid which was created at refinement step l. The multigrid method and grid refinement algorithms can be formulated to operate completely on trees with optimal complexity [105,140].

However, the storage requirements are quite high compared to structured grids and to unstructured grids. This is due to the additional pointers needed. Furthermore the number of edges and faces in three dimensions is higher than the original number of nodes due to Euler's formula. This means that compared to the storage of the unknowns, which is proportional to the number of nodes for linear elements, the storage of purely geometric entities cannot be neglected. In addition the coarser grid hierarchies contain edges and elements which have to be stored. Hence several authors [13,15,20] avoid the storage of faces

inside the domain Ω in three dimensions. Related to the storage overhead is the question of performance of a code, which has to manipulate a large amount of data.

Instead of linked lists or trees, we propose to use *hash storage* techniques. First we describe a *key based* addressing scheme. An implementation of a key-based scheme with hash tables is described later. Each entity of the grid is assigned to a unique key, which is an integer number. The entity is stored in an abstract vector, where it can be retrieved by its key. Furthermore it is possible to decide, whether a given key is stored in the table or not, and it is possible to loop over all keys stored in the vector. In order to reduce the amount of storage of the grid, we omit any pointers and use keys instead. For a (hyper-) cube shaped domain $\Omega = [0,1]^d$, we can use the coordinates of a node for addressing purposes. The coordinates of hierarchical son nodes and father nodes can be computed from the node's coordinates easily. Hence, the keys of the nodes are available and the nodes can be looked up in the vector, if they exist. Nodes on the finest refinement level can be determined by the fact, that they do not have son nodes. The computation of neighbor nodes requires special care, because it is not immediately clear, where to look for the node. Given a one-irregular grid with hanging nodes, for example, a neighbor node can be located in the distance of h or $2h$ from the node with a local step-size h. In the worst case this results in two vector lookup operations, one in distance h along a coordinate direction and, if it was unsuccessful, one lookup in distance $2h$, see [81]. Similar key-based addressing schemes can be obtained for other grid refinement procedures and for different domains, see [141,152]. For example the element of a general triangulation or tetrahedrization τ_0 of a polygonal domain Ω can be enumerated. Along with a numbering scheme based on local coordinates in each element, a general key addressing scheme can be established. Also quad- or octree-tree techniques can be used, see [161].

An efficient implementation of a key-based addressing scheme has to deal with the problem, that the space of possible keys is huge and only a few of the keys are in use. In comparison to sorted lists and trees, which can also be used here, *hash storage* techniques are more efficient [98]. A given key is mapped to an index of a table by a hash function f, see Fig. 4. An item with key k is stored in location $f(k)$ of the hash table. Several items with different keys $k_1 \neq k_2$ might be mapped to the same location $f(k_1) = f(k_2)$ which is called a collision and which has to be resolved. Collisions may happen because the hash function cannot be injective. Popular techniques to resolve these collisions are chaining (see Fig. 4), linear probing and double hashing. There are many different choices for a good hash function which is responsible to scatter the keys in use to the limited number of entries in the table. The performance of a hash table is usually estimated in a statistical setting, which depends on the quality of the hash function f. Random access is a constant time $\mathcal{O}(1)$ operation in the statistical mean, as long as there are enough empty slots in the table. Nevertheless, f may map several keys in use to the same index. For further details see [98]. In comparison to linked lists and tree, key-based addressing with hash storage is simple, efficient and requires very little memory.

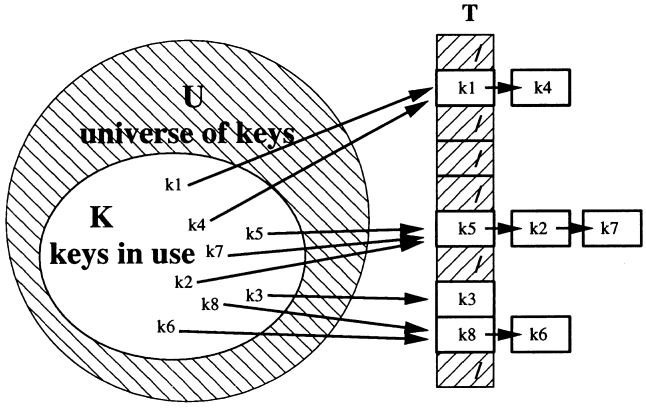

Fig. 4. Hash table, collision resolution with chaining.

4. Parallel computing

For the efficient solution of the equation system arising from a discretization of the PDE we choose the most efficient class of solvers, the multigrid methods. The data are partitioned and mapped to the processors and the multigrid method operating on the data runs in parallel. Here, we assume a distributed memory computer model with message passing for generality. The results also apply to smaller parallel computers with shared memory where some stages of the parallelization are simpler.

In the sequel, we focus on parallel implementations of multigrid methods. The parallelization of the competing domain decomposition methods on the contrary is straightforward, because the methods are constructed for this purpose. Hence we do not have to discuss 'parallel' domain decomposition methods.

Our goal is to solve the equation system as fast as possible. This means, that we have to consider an efficient parallelization and consequently, we have to develop a parallel multigrid code that is almost identical to the sequential implementation. The computational workload has to be distributed into similar sized partitions and, at the same time, the communication between the processors has to be small. The underlying computer model takes into account the local processor execution time and the communication time. The first term is proportional to the number of operations and the second one depends on the amount of data to be transferred between processors. A refined execution model may also take into account the number of messages sent, i.e. the message overhead, which can be compensated for by overlapping computation and communication to some extent.

4.1. Parallel multigrid

A parallel version of multiplicative multigrid usually is based on a partition of all nested grids. The domain Ω is decomposed into several sub-domains Ω_j, which induces partitions of all grids. Each processor holds a fraction of each grid in such a way that these fractions of each grid form a nested sequence. Hence each operation on a specific level is partitioned and mapped to all processors. Furthermore the communication during grid transfer operations is small because of nested sequences on a processor. This means that one has to treat global problems on each level, which are partitioned to all processors. The intra-grid communication has to be small, that is the number of nodes on the boundary of the partition should be small. Furthermore the amount of work on coarse grid levels usually is small and each processor does not compute much. There are several strategies to deal with the coarse grid problem in general, such as to centralize the computation on a master processor, to perform identical computations on all processors or to modify the coarse grid correction step.

4.1.1. Uniform and block-structured grids

A static partition of the domain into strips or squares can be used for uniform grids and has been used for the first parallel multigrid implementations [44,82,151], see also the survey [112]. In contrast to the geometry oriented parallelization of multiplicative multigrid methods, the additive multigrid version or additive multilevel preconditioners can be parallelized in a more flexible way. The overall workload has to be partitioned, but we do not have to consider individual levels. Here, also the communication takes place in a single step for all nodes, which are located on the boundary of at least one grid of the nested sequence. The multilevel BPX preconditioner for a uniform grid has been parallelized in [20,175]. These approaches can easily be generalized to block-structured grids.

There are many suggestions to modify components of the multigrid algorithm for parallel computing. Multiple coarse grids can keep all processors busy during operations on coarse grids, where often some processors are idle. Parallel versions of the smoothers, which take the partition into account, enhance the parallel performance. The usual communication steps during the execution of the smoother on one level can be reduced, if one uses block Jacobi type smoothers. Some other variants such as a parallel point- and domain-block re-formulation can be found in [75–77]. An analogous parallelization method can be found in [47]. For a more popular introduction to parallel multigrid versions see [61]. More recent modifications of multigrid algorithms are concerned with the performance on computers with memory hierarchies, such as RISC processors with memory cashes. The execution order of inter- and intra-grid transfer operations has to be reordered in order to minimize the required memory bandwidth, see [62,153].

4.1.2. Patch-wise adaptive grids

Multigrid methods on adaptive grids are harder to parallelize. The workload is created at runtime during refinement and has to be partitioned and mapped to the processors immediately. This means that the partitioning strategy has to be cheap, or at least computationally as expensive as the multigrid solver itself.

Hence the first implementations of adaptive multigrid methods are based on the simple uniform grid case: The adaptive grid is composed of the union of uniform, rectangular grids of different area and mesh sizes h. During adaptive refinement, new patches are created and overlayed on the existing union of patches where grid refinement is needed. A multigrid method on such a composite grid can be parallelized in two ways: Either each patch is assigned to a single processor or each patch is partitioned and mapped to all processors in the same way as in the uniform grid case. This has been implemented on distributed shared memory architectures by [14]. On distributed memory computers, several packages have been developed so far, such as a multigrid implementation for LPARX [99], AMR [106] based on a parallel array package and the block-structured Navier-Stokes solver LiSS [139].

4.1.3. Element-wise adaptive grids

Parallel multigrid methods on a sequence of unstructured grids require some grid partitioning algorithms. These can be based on graph partitioning heuristics or on geometric heuristics and can be expensive. However, in the framework of adaptive, element-wise refinement, partitions and mappings have to be computed quickly and during runtime. On a shared memory parallel computer, the serial representation of the grid hierarchy in memory and coloring of the elements along with dynamic scheduling of lists of elements can be used. This has been proposed for a code based on triangles and the additive hierarchical bases preconditioner [23,105].

On a distributed memory computer, the grid hierarchy has to be partitioned and maintained, which requires a substantial amount of bookkeeping. In [58] a multiplicative multigrid method or a finite volume discretization on a grid with quadrilaterals and hanging nodes is proposed. The elements are partitioned by a hierarchical recursive coordinate bisection. Test indicated that the repartitioning of coarser levels, when new elements were created, did not pay off. Adaptive conforming grids consisting of triangles, which are refined by a bisection strategy, were employed in the parallel multigrid codes of [13,116,152]. Here, different repartitioning strategies for refined grids were used.

4.2. Grid partitioning methods

The grid partition problem can be equivalently formulated as a graph partitioning problem. However, the general graph partitioning problem is NP-hard. Even the problem to find asymptotically optimal partitions for unstructured grids is NP-hard [50]. Several heuristic algorithms have been developed in the area of parallel computing: There are bisection algorithms based on the coordinates of nodes and elements and there are many algorithms based on the graph of the stiffness matrix. For instance, [135] propose a recursive spectral bisection based on the discrete Laplacian of the graph. In the package Chaco [94] a multilevel heuristic of spectral bisection is used. The package Metis [97] uses a multilevel version of the Kernighan–Lin algorithm, which improves a graph partition by moving nodes from one partition to the other. Its multilevel version does this also on coarser representations of the graph. Other data diffusion heuristics are employed in Jostle [160], in the PDE codes UG [15,152], and in [18,27]. For a survey on grid partitioning methods we refer to [134], see Fig. 5.

The key point of any dynamic data partition method is efficiency. We look for a cheap, linear time heuristic for the solution of the partition problem. Furthermore the heuristic should parallelize well. Here, parallel graph coarsening is popular. It results in a coarser graph on which then a more expensive heuristic on a single processor can be employed. However, graph coarsening is at least a linear time algorithm itself and lowers the quality of the heuristic further. This is why we look for even cheaper partition methods. They are provided by the idea of *space-filling curves*.

4.2.1. An approach by space-filling curves

Space-filling curves originally had been constructed for a continuous mapping of a line segment $[0,1] \subset \mathbb{R}$ to a polygonal area $[0,1]^2 \subset \mathbb{R}^2$, see for an overview [144]. Such curves can also be used for the

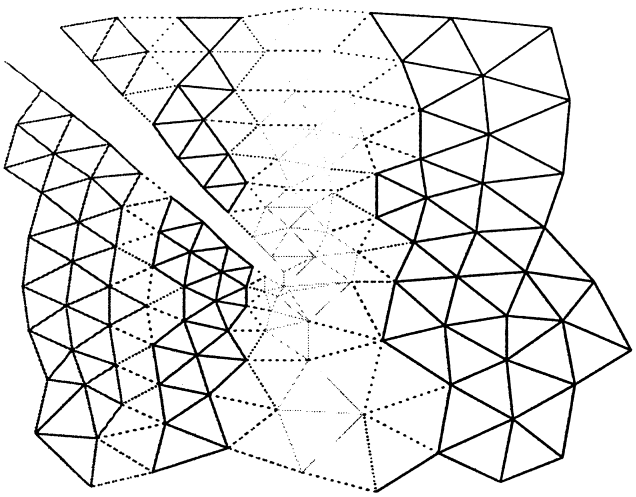

Fig. 5. Partition of an FEM graph to four parallel processors.

inverse mapping f from a domain $\Omega \subset \mathbb{R}^d$ to an interval $[0,1] \subset \mathbb{R}$. This means that we can map geometric entities in $\mathbb{R}^d$ to the one-dimensional interval. Entities, which are neighbors on the interval, are also neighbors in the volume $\mathbb{R}^d$. Unfortunately the reverse cannot be true and neighbors in the volume may be separated through the mapping. However, we know how to solve a one-dimensional partition problem: We cut the interval into equal sized pieces, which gives perfect load-balance and small separators of size one, see Fig. 6. The partition of the volume $\mathbb{R}^d$ induced by the space-filling curve, see Figs. 7–9, still gives perfect load-balance. However, the separators usually are larger than the optimal separators. As a technical detail, we consider the partition and mapping of nodes of the grid and we choose a space-filling curve which is aligned to the grids. Hence a sufficiently fine, finite representation of the space-filling curve contains all nodes and covers a larger domain than the domain of interest Ω.

Partitioning by space-filling curves has been employed for finite element computations in [123,129] and has been compared to other heuristics in [133]. The main advantage of space-filling curves in this context is their simplicity: If we want to bisect a set S of points $x_i \in \Omega$ on a space-filling curve, we can do this by a single number s and the inverse space-filling curve mapping f

$$S = \{x_i \mid f(x_i) \leqslant s\} \cup \{x_i \mid f(x_i) > s\}.$$

Each point is either left or right of the reference s on the space-filling curve. We take s as the median of $f(S)$, both subsets are of the same size. In the same way we can partition the set of points S into p different subsets, if we partition $f(S) \subset [0,1]$ by $p-1$ separators s_i like $\{x_i \mid s_j < f(x_i) \leqslant s_{j+1}\}$. There is almost no bookkeeping necessary, because the partition is deterministic and can be computed on the fly from the

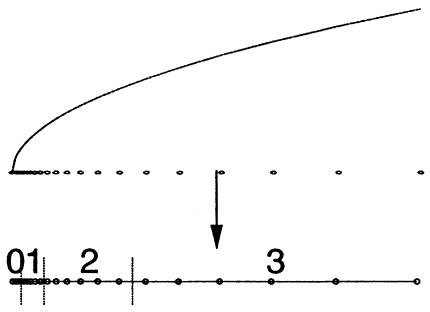

Fig. 6. Mapping a 1D adapted grid to a parallel processor.

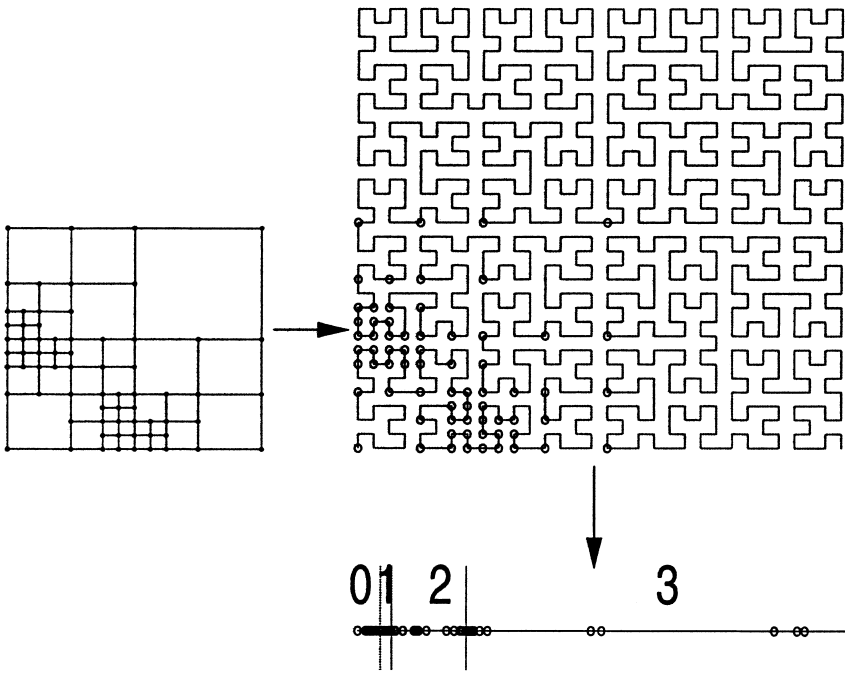

Fig. 7. Mapping a 2D adapted grid to a space-filling curve (left) and mapping points on a space filling curve to a parallel processor (right).

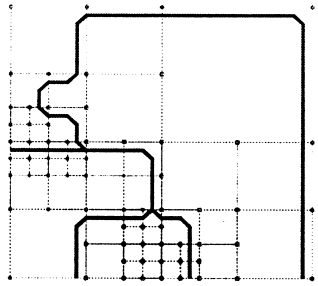

Fig. 8. A decomposition of the domain induced by the space filling curve mapping.

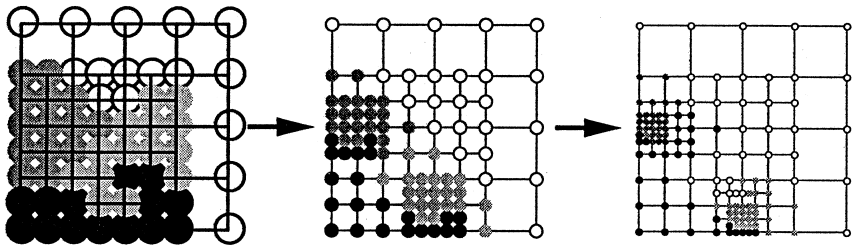

Fig. 9. A sequence of adaptively refined grids mapped to four processors.

separators. Each sub-set is assigned to one processor. Only the $p-1$ separators have to be stored on each processor, see Fig. 7.

We have to compute the separators such that the partitions contain the same number of nodes. This can be accomplished on a list of nodes sorted by their position on the space-filling curve $f(x_i)$. The list is cut into equal-sized pieces and each piece is mapped to one processor. Hence the partition of nodes can be done

with an ordinary sorting algorithm. Given a set of nodes x_i with their keys $f(x_i)$, we first create a sorted list of keys. However, we need to perform the partition algorithm on a parallel computer. Given a set of nodes with their keys, which are distributed over the processors, we look for a parallel sort algorithm, which results in a partition where each partition resides on the appropriate processor. There is no need to gather all keys on a master processor, but a pure scalable parallel algorithm performs better with respect to communication volume, memory usage and scalability. Currently we employ single step radix sort, where the previous separators serve as an initial guess for the sorting. For uniform grid refinement f.e., two steps of local neighbor communication are only required.

The remaining question is about the quality of this simple heuristic because it is cheap and scalable. The workload is partitioned evenly. Hence the communication volume cannot be optimal for $d > 1$. However, we can formulate

Theorem 2 (Separator sizes for Hilbert curve partitionings). *Suppose that we partition a uniform grid which covers the cube* $[0,1]^d$ *by a Hilbert curve, then for any sub-domain* Ω_i *with n nodes obtained this way*

(a) *the number of boundary edges s is* $s \approx n^{d-1/d}$ *with constants dependent only on d and the discretization stencil,*

(b) *and for the two-dimensional case, the number s is bounded tightly by* $s \leqslant 6\sqrt{n+1}$ *for 5-point stencils and* $s \leqslant 18\sqrt{n+1}$ *for 9-point stencils.*

Proof. The bounds of part (b) can be easily proved by induction over groups ($n < 16, 16^2, 16^3 \cdots$) of increasing n using the recursive definition of the Hilbert space-filling curve, see also the 'worst' case scenario in Fig. 10, where for a given number of nodes a maximum diameter is reached. The upper bound of the general estimate (a) can be proved analogously by induction. The lower bound is a general geometric property of any partition. Note that this result is related to estimates for distances of Hilbert indexings in [122].

Hence the separator sizes for uniform grids obtained by space-filling curve partitionings are optimal up to a constant, if we neglect effects of the boundary $\partial\Omega$. This indicates, that the partitioning algorithm performs well in this case. In more general cases of adaptive refined grids, numerical experiments indicate that the graph separators of space-filling curve partitions are of sufficient quality, see [133].

4.2.2. Hash storage and space-filling curves

The parallelization of an adaptive code usually is non-trivial and requires a substantial amount of code for the parallelization only. Hierarchies of refined grids, where neighbor elements may reside on different processors, have to be managed [24]. That is appropriate ghost nodes and elements have to be created and updated, when the parallel algorithm performs a communication operation. This happens both in the numerical part, where an equation system is set up and solved, and in the non-numerical part, where grids are refined and partitioned, see also [13,96].

In Section 3.3 we have considered hash storage techniques to simplify the implementation of a sequential, adaptive code. Now, we generalize the concept of key addressing and hash tables to the parallel case. The idea is to store the data in a hash table located on the local processor. However, we use global keys, so a ghost copy of the node may also reside in the hash table of a neighbor processor. Furthermore we base the code on space-filling curve partitions of the previous section. The position of a node on the space-

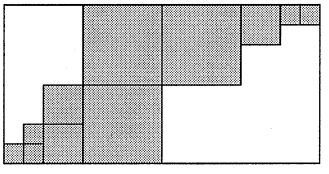

Fig. 10. Worst case for a Hilbert curve partitioning.

filling curve, along with the known partition, defines the home processor of a node. Given a node on a processor, it is easy to determine to which processor the node belongs to. If the node does not belong to the processor, it must be a ghost copy, and it is known where to find its original.

The next idea is to combine the position on the space-filling curve with the hash key [131,161]. The efficiency of the hash storage scheme is based on the choice of an appropriate hash function, which maps a node to an index in the hash table. This can be any mapping $[0,1] \to [0,m]$ with a large integer number m, preferably a prime. Many cheap functions related to pseudo-random numbers will do here. A function that scatters the nodes very broad across the entries of the table in general avoids collisions in the hash table, while a narrow distribution of the hash functions can improve memory-cache performance. The space-filling curve mapping can be considered as a hash function with a narrow distribution, because they introduce locality in the key addressing scheme, which is also exploited for the parallelization of the code. Using the data locality once again on the local processor, one can optimize the usage of secondary disk storage and of the memory hierarchy of cashes, which is difficult otherwise [62]. In order to increase the statistical variation of the hash function, we are using some simple scrambling of the binary representation of the hash key. However, some numerical experiments indicate that the overall performance of the hash table in fact does not really depend on the precise scrambling procedure. The space-filling curve mapping, i.e. the position on the curve can be computed for any given coordinate tuple. It is a unique mapping $[0,1]^d \to [0,1]$ similar to mapping required for hash keys. A scrambled representation of the position can be also used as a hash key.

This framework for the parallelization of adaptive codes originally has been invented for particle methods [161] and has been generalized to programming environments for some grand challenge PDE projects [131]. Multigrid methods have been considered in [81].

5. Numerical experiments

We will present some numerical experiments to demonstrate the suitability and efficiency of the proposed parallel multigrid method based on space-filling curve decomposition and hash storage. The applications, a Poisson problem and the Lamé equation of linear elasticity serve as simple test cases only. However, due to the few operations per node, it is rather hard to achieve good parallel efficiencies and with this respect, these are even hard test problems. We have to mention that in the area of elasticity, domain decomposition methods are also very popular and a number of industrial parallel application are based on these algorithms. We will not cover this aspect and refer instead to the literature [68,102,146] and other articles of this issue.

The following numerical experiments are based on an adaptive parallel multigrid solver which uses a Hilbert space-filling curve for decomposition and a hash table for addressing, see [81]. A finite difference discretization is used, where the degrees of freedom are located at the nodes of the grids. The hanging nodes of the 1-irregular grid of the unit square or unit cube are defined by interpolation. The linear equation systems are solved by a Krylov method, preconditioned by an additive multigrid or BPX-type cycle. All numbers reported are CPU times measured on our parallel computing cluster 'Parnass2'. It consists of dual processor Pentium II 400 MHz boards with at least 256 Mbytes of main memory, interconnected by a Myrinet network in a fat-tree configuration. The MPI message passing protocol implementation Mpich-PM showed a bandwidth between each two boards of 850 Mbit/s, see also [177]. We list execution times of the linear equation solver for different problem sizes and numbers of processors. Furthermore we compare the execution time of the load-balancing step with the execution time of the linear solver and compute their ratio $\alpha := t_{\text{balancing}}/t_{\text{solving}}$.

5.1. Poisson problem

We consider the Poisson equation

$$-\Delta u = f \quad \text{on } \Omega = [-1,1]^d, \quad d = 2,3$$

with the Dirichlet boundary conditions $u = 0$ on part of the boundary $\Gamma_D \subset \partial\Omega$ and homogeneous Neumann boundary conditions $\partial/\partial n\, u = 0$ on the remaining boundary $\Gamma_N = \partial\Omega \setminus \Gamma_D$. In the following, we consider two types of boundary conditions Γ_D. First, we consider a pure Dirichlet problem $\Gamma_N = \emptyset$, which result in a smooth solution. Here, a uniformly refined grid is sufficient for discretization and adaptivity is not needed. Second, we consider the case where the solution possesses almost a corner singularity due to the source term f. We run our adaptive multilevel finite difference code to solve it.

5.1.1. Uniform example

In the first test we consider regular grids (uniform refinement). Tables 2 and 3 show wall clock times for the solution of the equation system on a regular grid of different levels using different numbers of processors.

For a fixed number of processors, we observe a scaling of a factor of 2^d from one level to the next finer level which corresponds to the factor of 2^d increase in the amount of unknowns on that level. Furthermore, for a fixed level the measured times scale roughly with $1/p$ of the number of processors. However, the 32 and 64 processors perform efficiently only for sufficiently large problems, i.e. for problems with more than some thousand degrees of freedom. For larger problems we even obtain some super-linear speed-ups, probably due to caching effects. If we fix the amount of work, that is the number of nodes per processor, we obtain the scale-up. Comparing a time at one level l and a number of processors p with the time of one level finer $l + 1$ and $2^d p$ processors, we obtain nearly a perfect scaling of the method. The 2D example shows slightly higher parallel efficiencies than the 3D example, because the data to be exchanged for n degrees of freedom increase from $O(n^{1/2})$ (2D) to $O(n^{2/3})$ (3D), which makes three-dimensional problems harder to parallelize.

Note that in this case of uniform grid refinement, an a priori partition of the uniform grids into stripes would be superior to any dynamic partitioning scheme. However, our dynamic load balancing scheme performs well and introduces only little over-head, which can also be seen in Section 5.1.3.

Table 2
Uniform refinement example 2D, timing, levels 5–11, 1–64 processors

Time		Processors						
Level	Nodes	1	2	4	8	16	32	64
5	1089	3.30	2.18	1.32	0.94	0.69	0.52	0.43
6	4225	16.0	8.52	4.70	2.88	1.74	1.16	0.79
7	16,641	66.9	33.6	17.6	9.87	5.40	3.17	1.88
8	66,049	283	141	70.4	36.2	19.0	10.4	5.64
9	2,63,169	1160	583	291	147	73.8	37.5	19.4
10	1,050,625			1162	584	294	147	74.0
11	4,198,401					1162	585	292

Table 3
Uniform refinement example 3D, timing, levels 3–7, 1–64 processors

Time		Processors						
Level	Nodes	1	2	4	8	16	32	64
3	729	3.05	2.07	1.51	1.15	1.14	1.01	
4	4913	30.2	17.6	10.6	6.87	5.11	3.64	2.73
5	35,937	277	150	81.4	46.4	29.0	17.9	11.4
6	274,625	2455	1297	674	356	198	109	61.8
7	2,146,689				2782	1482	774	410

5.1.2. Adaptive example

In the next test we consider adaptive refined grids for a problem with singularities, where the grids are refined towards a singularity located in the lower left corner, see also Fig. 11. Tables 4 and 5 depict times in the adaptive case. These numbers give the wall clock times for the solution of the equation system again, now on different levels of adaptive grids and on different numbers of processors.

We obtain a scaling of about a factor two to three from one level to the next finer level, i.e. the times are proportional to the number of unknowns for a fixed number of processors. This is due to the adaptive grid refinement heuristic. Increasing the number of processors speeds up the computation accordingly.

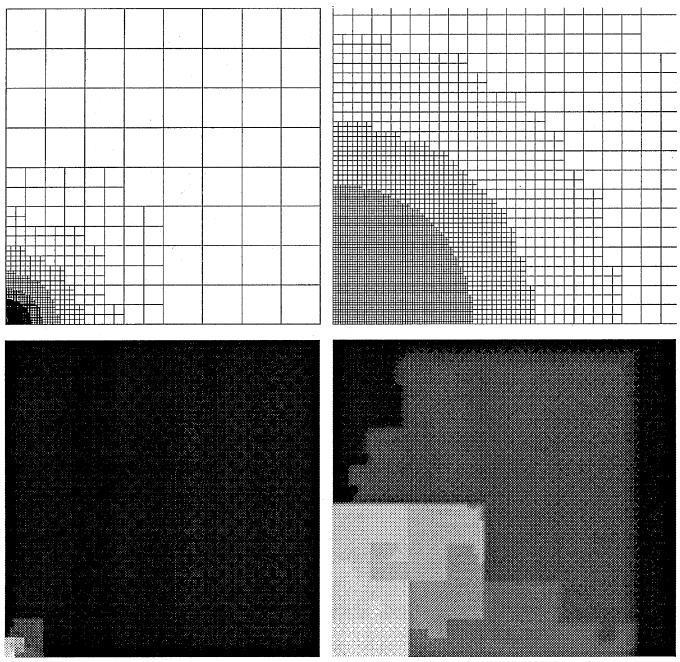

Fig. 11. 2D refined grid, mapped to 8 processors, full domain left, zoomed image on the right, grid and color coded partition.

Table 4
Adaptive refinement example 2D, timing, 1–64 processors

Time	Processors						
Nodes	1	2	4	8	16	32	64
134	0.37	0.26	0.24	0.24	0.24	0.27	0.27
224	0.69	0.49	0.36	0.34	0.31	0.30	0.32
384	1.27	0.85	0.69	0.51	0.42	0.37	0.35
682	2.38	1.48	1.04	0.75	0.57	0.45	0.41
1243	4.54	2.81	1.81	1.21	0.83	0.60	0.51
2320	8.75	4.92	3.13	1.95	1.25	0.86	0.62
4391	17.0	9.30	5.19	3.26	1.89	1.25	0.85
8460	33.5	17.8	10.1	5.57	3.27	1.92	1.26
16,469	66.9	34.4	18.1	10.1	5.50	3.21	1.99
32,291	133	67.7	35.4	19.3	10.3	5.50	3.27
63,736	263	134	68.5	36.6	19.2	10.5	5.66
126,271	529	272	139	70.4	36.7	19.1	10.3
250,911		560	278	143	71.8	36.8	19.1

Table 5
Adaptive refinement example 3D, timing, 1–64 processors

Time	Processors						
Nodes	1	2	4	8	16	32	64
1191	5.82	3.64	2.53	1.75	1.35	1.21	1.22
2178	12.3	7.94	5.07	3.82	3.02	2.20	1.97
4454	28.5	17.0	11.0	7.00	4.74	3.57	3.00
10,061	71.9	43.9	26.5	16.2	10.2	7.04	5.16
24,215	190	108	60.8	36.3	21.0	14.0	9.07
61,361	510	280	157	87.7	49.0	29.5	17.7
160,384	1418	772	404	217	125	70.5	40.8
429,613				602	318	175	95.8

5.1.3. Load balancing

Now we compare the time for solving the equation system with the time required for sorting the nodes and mapping them to processors before the computation starts.

The ratio α indicates how expensive the load balancing and mapping task is in comparison to the linear algebra part of the code. We give the values in Table 6 for the previous uniform and adaptive refinement examples using different numbers of processors.

In the single processor case, no load balancing is needed, so the partitioning and mapping time and the ratio α is zero. Otherwise, the nodes have to be partitioned and mapped by a parallel (partial) sort algorithm. In the uniform grid case the relative cost of partitioning nodes α is of the order $1e-3$. In the case of uniform refinement, for a refined grid, there are only few nodes located at processor boundaries which may have to be moved during the mapping. Hence our load balancing is also very cheap in this case. In the adaptive grid case, dynamic load balancing generally is required. Note that in all cases load balancing is much cheaper than solving the equation system. However, higher number of processors make the mapping and partitioning slightly slower. Mapping data for adaptive refinement requires the movement of a large amount of data, even if most of the nodes stay on the processor. Other load balancing strategies can be quite expensive for adaptive refinement procedures, see [13,152].

5.2. Linear elasticity

As a second test case for our approach, we consider the linear elasticity problem. We consider the Lamé equation [33,52] in the displacement formulation, either as a linear three-dimensional problem or as a two-dimensional plain-strain problem

$$\begin{aligned} \mu\vec{\Delta}\vec{u} + (\lambda+\mu)\nabla(\nabla\vec{u}) &= \vec{f} \quad \text{in } \Omega \subset \mathbb{R}^d, \quad d = 2,3 \\ \vec{u} &= \vec{0} \quad \text{on } \Gamma_{\mathrm{D}} \subset \partial\Omega \\ \sigma(\vec{u})\vec{n} &= \vec{0} \quad \text{on } \Gamma_{\mathrm{N}} = \partial\Omega \setminus \Gamma_0 \end{aligned}$$

Table 6
Ratio α of execution times of partitioning and mapping nodes to solving the equation system, 1–64 processors

Ratio α		Processors						
	Nodes	1	2	4	8	16	32	64
Uniform 2D	66,049	0	9.7e−4	1.1e−3	1.3e−3	1.3e−3	3.2e−3	4.9e−3
Adaptive 2D	63,736	0	6.8e−4	7.9e−4	9.1e−4	1.1e−3	2.9e−3	3.5e−3
Uniform 3D	35,937	0	3.2e−4	4.1e−4	1.9e−3	1.4e−3	1.6e−3	3.7e−3
Adaptive 3D	61,361	0	3.4e−4	7.2e−4	8.8e−4	9.8e−4	1.1e−3	1.5e−3

with Lamé's constants

$$\lambda = \frac{Ev}{(1+v)(1-2v)} \quad \mu = \frac{E}{2(1+v)}$$

and linearized strain $\epsilon(\vec{u}) = \frac{1}{2}(\partial u_i/\partial x_j + \partial u_j/\partial x_i)_{ij}$, $\lambda > 0$, $\mu > 0$ and $E > 0$. The problem is elliptic due to Korn's inequality for $0 < v < \frac{1}{2}$. We consider two test cases: First, we take a homogeneous square or cube under internal forces parallel to one coordinate axis, fixed at two adjacent edges ($d = 2$) or three faces ($d = 3$). The remaining edges or faces are free (homogeneous Neumann conditions). The solution is smooth and uniform refined grids are sufficient, see Table 7. Second, the body under a different load $\vec{f}$ is considered such that singularities occur. Here adaptive refinement is used to resolve the singularities next to the edges or faces (left, bottom and front), which separate homogeneous Dirichlet (fixed faces) and homogeneous Neumann conditions (free faces), see Table 9 and Fig. 12. All numbers reported are again CPU times measured on the cluster 'Parnass2'.

Also for Lamé's equation, principally the same behavior of our approach as for the Poisson problem can be seen. Tables 7–10 show that our method scales well in the uniform and adaptive cases. Note that the parallel efficiency is even higher than for the Poisson problem. This is due to the higher amount of work associated with each node, while the expenses associated with the grid stay the same. In the elasticity case, there are d degrees of freedom located at each node.

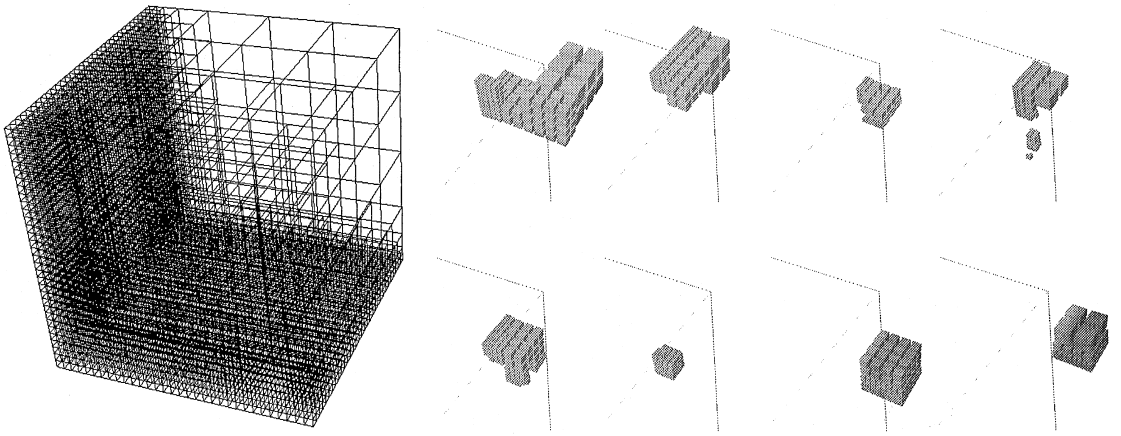

Fig. 12. 3D refined grid, mapped to 8 processors (grid on the left, gray shaded partitions on the right).

Table 7
Uniform refinement example, elasticity 2D, timing, levels 4–10, 1–64 processors

Time			Processors						
Level	Nodes	Dof	1	2	4	8	16	32	64
4	289	578	1.27	0.74	0.48	0.34	0.26	0.22	0.22
5	1089	2178	5.31	2.83	1.60	0.96	0.61	0.41	0.34
6	4225	8450	21.9	11.2	5.93	3.24	1.77	1.05	0.69
7	16,641	33,282	91.6	45.3	23.1	12.0	6.19	3.35	1.89
8	66,049	132,098	386	192	94.2	46.7	23.6	12.2	6.34
9	263,169	526,338	1578	789	392	195	95.8	47.3	23.9
10	1,050,625	2,101,250			1559	777	388	192	95.6

Table 8
Uniform refinement example, elasticity 3D, timing, levels 3–7, 1–64 processors

Time			Processors						
Level	Nodes	Dof	1	2	4	8	16	32	64
3	729	2187	3.05	1.99	1.32	0.91	0.73	0.53	
4	4913	14,739	27.6	15.3	8.81	5.49	3.51	2.07	1.30
5	35,937	107,811	249	131	68.2	38.1	21.4	11.7	6.57
6	274,625	823,875	2164	1124	573	300	158	80.5	41.7
7	2,146,689	6,440,067					1211	618	311

Table 9
Adaptive refinement example, elasticity 3D, timing, 1–64 processors

Time		Processors						
Nodes	Dof	1	2	4	8	16	32	64
125	375	0.10	0.12	0.11	1.50	2.91		
450	1350	1.44	0.99	0.80	1.35	0.50	0.39	1.05
1155	3465	4.14	2.48	1.71	1.32	1.00	0.70	2.74
4412	13,236	19.0	10.3	6.09	5.23	3.07	1.89	1.21
18,890	56,670	98.6	50.3	28.1	20.6	11.6	6.35	3.70
93,021	279,063	582	294	157	102	54.8	28.2	15.1
506,620	1,519,860				556	306	155	78.1
3,178,218	9,534,654							494

Table 10
Ratio α of execution times of partitioning and mapping nodes to solving the equation system, elasticity 3D, 1–64 processors

Ratio α		Processors						
	Nodes	1	2	4	8	16	32	64
Uniform 3D	274,625	0	3.4e−4	3.2e−4	3.3e−4	1.7e−3	6.7e−4	1.4e−3
Adaptive 3D	93,021	0	3.6e−4	3.5e−4	4.0e−4	4.9e−4	6.9e−4	1.4e−3

6. Concluding remarks

In this paper we gave a survey of the basic ingredients of an efficient solver for self-adjoint elliptic PDEs, i.e. multilevel and domain decomposition solvers, adaptive grid refinement and parallelization. We focused on the interplay between these ingredients and tried to illustrate the they can be glued together into an adaptive parallel multilevel method. Here we proposed the application of hash storage techniques for data management and the use of space-filling curves for load balancing in the parallel version of the algorithms and presented a first version of a parallel adaptive multilevel method based on these approaches.

Let us stress in the end, that there are still many open questions and problems with adaptive parallel multilevel solvers. First of all, when the domain of interest is quite complicated and cannot be resolved by the coarser levels, the multigird must be modified accordingly. Then, remember that especially for three-dimensional problems the robustness is still an open question. Furthermore, the interplay between adaptivity and parallel efficiency is not fully understood and needs further elaboration. Finally, for the application of non-linear elasto-plastic materials and contact problems there remains a lot of work to do.

References

[1] V.I. Agoshkov, V.I. Lebedev, Poincaré-Steklov operators and domain decomposition methods in variational problems, Computational Processes and Systems 2 (1985) 173–227.
[2] G.P. Astrakhantsev, An iterative method of solving elliptic net problems, Sov. J. Comput. Math. Math. Phys. 11 (1971) 439–448.
[3] O. Axelsson, P.S. Vassilevski, Algebraic multilevel preconditioning methods, I, Numer. Math. 56 (1989) 157–177.
[4] I. Babuška, Über Schwarzsche Algorithmen in partiellen Differentialgleichungen der mathematischen Physik, ZAMM 37 (1957) 243–245.
[5] I. Babuška, A. Miller, A feedback finite element method with a posteriori error estimation: Part I. The finite element method and some basic properties of the a posteriori error estimator, Comput. Meth. Appl. Mech. Engrg. 61 (1987) 1–40.
[6] I. Babuška, W.C. Rheinboldt, Error estimates for adaptive finite element computations, SIAM J. Numer. Anal. 15 (1978) 736–754.
[7] N.S. Bachvalov, On the convergence of a relaxation method with natural constraints on the elliptic operator, Sov. J. Comput. Math. Math. Phys. 6 (1966) 861–883.
[8] R.E. Bank, T. Dupont, An optimal order process for solving elliptic finite element equations, Math. Comp. 36 (1981) 35–51.
[9] R.E. Bank, T. Dupont, H. Yserentant, The hierarchical basis multigrid method, Numer. Math. 52 (1988) 427–458.
[10] R.E. Bank, A.H. Sherman, H. Weiser, Refinement algorithms and data structures for regular local mesh refinement, in: R. Stepleman (Ed.), Scientific Computing, IMACS, North-Holland, Amsterdam, 1983, pp. 3–17.
[11] R.E. Bank, A. Weiser, Some a-posteriori error estimators for elliptic partial differential equations, Math. Comp. 44 (1985).
[12] R.E. Bank, J. Xu, The hierarchical basis multigrid method and incomplete LU decompostion, in: Proceedings of the Domain Decomposition Methods 7, Contemporary Mathematics, vol. 180, AMS, Providence, Rhode Island, 1994, pp. 163–173.
[13] P. Bastian, Parallele Adaptive Mehrgitterverfahren, Teubner, Stuttgart, 1996.
[14] P. Bastian, J.H. Ferziger, G. Horton, J. Volkert, Adaptive multigrid solution of the convection-diffusion equation on the DIRMU processor, in: W. Hackbusch, (Ed.), Robust Multi-grid methods, Notes on Numerical Fluid Mechanics, vol. 23, Vieweg, Braunschweig, 1989, pp. 27–36.
[15] P. Bastian, S. Lang, K. Eckstein, Parallel adaptive multigrid methods in linear structural mechanics applications, Linear Algebra Appl., to appear.
[16] R. Becker, C. Johnson, R. Rannacher, Adaptive error control for multigrid finite element methods, Computing 55 (1995) 271–288.
[17] R. Becker, R. Rannacher, Weighted a posteriori error control in FE methods, in: Proceedings of the ENUMATH 95, 1995.
[18] M.J. Berger, S. Bokhari, A partitioning strategy for nonuniform problems on multiprocessors, IEEE Trans. Comput. C–36 (1987) 570–580.
[19] C. Bernadi, Y. Maday, A. Patera, A new nonconforming approach to domain decomposition: The mortar method, in: H. Brezis, J.L. Lions (Eds.), Nonlinear Partial Differential Equations and their Applications, Pitman, London, 1989.
[20] J. Bey, Analyse und Simulation eines Konjugierte-Gradienten-Verfahrens mit einem Multilevel Präkonditionierer zur Lösung dreidimensionaler elliptischer Randwertprobleme für massiv parallele Rechner, Master's Thesis, RWTH Aachen, 1991.
[21] J. Bey, Tetrahedral grid refinement, Computing 55 (1995) 355–378.
[22] J. Bey, Simplicial grid refinement: On Freudenthal's algorithm and the optimal number of congruence classes, Tech. Rep. 151, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 1998.
[23] K. Birken, Ein Parallelisierungskonzept für adaptive, numerische Berechnungen, Master's Thesis, Universität Erlangen-Nürnberg, 1993.
[24] K. Birken, C. Helf, A dynamic data model for parallel adaptive PDE solvers, in: B. Hertzberger, G. Serazzi (Eds.), Proceedings of HPCN Europe 1995, Milan, Italy. Lecture Notes in Computer Science, vol. 919, Springer, New York, 1995.
[25] P.E. Bjørstad, J. Mandel, On the spectra of orthogonal projections with applications to parallel computing, BIT 31 (1991) 76–88.
[26] P.E. Bjørstad, O.B. Widlund, Iterative methods for the solution of elliptic problems on regions partitioned into substructures, SIAM J. Numer. Anal. 23 (1986) 1097–1120.
[27] S.H. Bokhari, T.W. Crockett, D.N. Nicol, Parametric binary dissection, Tech. Rep. 93–39, ICASE, 1993.
[28] F.A. Bornemann, Interpolation spaces and optimal multilevel preconditioners, in: Domain Decomposition Methods 7, Contemporary Mathematics, vol. 180, AMS, Providence, Rhode Island,1994, pp. 3–8.
[29] Bourgat, R. Glowinski, P. Le Tallec, M. Vidrascu, Variational formulation and algorithm for trace operator in domain decomposition calculations, in: T.F. Chan, R. Glowinski, J. Périaux, O.B. Widlund, (Eds.), Proceedings of the Domain Decomposition Methods 2, SIAM, Philadelphia, 1989, pp. 3–16.
[30] H. Brackhage, Über die numerische Behandlung von Intregralgleichungen nach der Quadraturformelmethode, Numer. Math. 2 (1960) 183–196.
[31] D. Braess, The contraction number of a multigrid method for solving the Poisson equation, Numer. Math. 37 (1981) 387–404.
[32] D. Braess, Towards algebraic multigrid for elliptic problems of second order, Computing 55 (1995) 379–393.
[33] D. Braess, Finite Elemente, 2nd ed., Springer, Berlin, 1996.
[34] D. Braess, C. Blömer, A multigrid method for a parameter dependent problem in solid mechanics, Numer. Math. 57 (1990) 747–761.
[35] J.H. Bramble, Multigrid Methods, Pitman Research Notes in Mathematical Sciences, vol. 294, Longman Scientific & Technical, Essex, England, 1993.
[36] J.H. Bramble, J.E. Pasciak, New estimates for multilevel algorithms including the V-cycle, Math. Comp. 60 (1993) 447–471.
[37] J.H. Bramble, J.E. Pasciak, A.H. Schatz, An iterative method for elliptic problems on regions partitioned into substructures, Math. Comp. 46 (1986) 361–369.
[38] J.H. Bramble, The construction of preconditioners for elliptic problems by substructuring IV, Math. Comp. 53 (1989) 1–24.

[39] J.H. Bramble, J.E. Pasciak, J. Wang, J. Xu, Convergence estimates for multigrid algorithms without regularity assumptions, Math. Comp. 57 (1991) 23–45.
[40] J.H. Bramble, J.E. Pasciak, J. Wang, J. Xu, Convergence estimates for product iterative methods with applications to domain decomposition, Math. Comp. 57 (1991) 1–21.
[41] J.H. Bramble, J.E. Pasciak, J. Xu, Parallel multilevel preconditioners, Math. Comp. 55 (1990) 1–22.
[42] A. Brandt, Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems, in: H. Cabannes, R. Teman (Eds.), Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, Lecture Notes in Physics, vol. 18, Springer, Berlin, 1973, pp. 82–89.
[43] A. Brandt, Multi-level adaptive solutions to boundary value problems, Math. Comp. 31 (1977) 333–390.
[44] A. Brandt, Multigrid solvers on parallel computers, in: M.H. Schultz (Ed.), Elliptic Problem Solvers, Academic Press, New York, 1981, pp. 39–83.
[45] A. Brandt, Guide to multigrid development, in: W. Hackbusch, U. Trottenberg (Eds.), Multigrid Methods, Lecture Notes in Mathematics, vol. 960, Springer, Berlin, 1982, pp. 220–312.
[46] A. Brandt, Algebraic multigrid theory: The symmetric case, Appl. Math. Comput. 19 (1986) 23–56.
[47] A. Brandt, B. Diskin, Multigrid solvers on decomposed domains, in: Proceedings of the Domain Decomposition Methods 6, Contemporary Mathematics, vol. 157, AMS, Providence, Rhode Island, 1994, pp. 135–155.
[48] S.C. Brenner, A nonconforming mixed multigrid method for the pure displacement problem in planar linear elasticity, SIAM J. Numer. Anal. 30 (1993) 116–135.
[49] S.C. Brenner, L.R. Scott, The Mathematical Theory of Finite Element Methods, Texts in Applied Mathematics, Springer, New York, 1994.
[50] T.N. Bui, C. Jones, Finding good approximate vertex and edge partitions is NP-hard, Inf. Process. Lett. 42 (1992) 153–159.
[51] T.F. Chan, T.P. Mathew, Domain Decomposition Algorithms, Acta Numerica, vol. 3, Cambridge University Press, Cambridge, 1994, pp. 61–143.
[52] P. Ciarlet, Three-Dimensional Elasticity, Mathematical Elasticity, vol. 1, North-Holland, Amsterdam, 1988.
[53] Proceedings of the Copper Mountain Multigrid Conference, Appl. Math. Comput./ Marcel Dekker/ SIAM/ NASA (Langley), 1981/ 1986/ 1987/ 1989/ 1991/ 1993/ 1995.
[54] A.W. Craig, J.Z. Zhu, O.C. Zienkiewicz, A posteriori error estimation, adaptive mesh refinement and multigrid methods using hierarchical finite element bases, in: MAFELAP84, The Mathematics of Finite Elements and Applications, 1984.
[55] S. Dahlke, W. Dahmen, R. Hochmuth, R. Schneider, Stable multiscale bases and local error estimation for elliptic problems, Appl. Numer. Math. 1 (1997) 21–48.
[56] W. Dahmen, Wavelet and multiscale methods for operator equations, in: Acta Numerica, vol. 6, Cambridge University Press, New York, 1997, pp. 55–228.
[57] W. Dahmen, A. Kunoth, Multilevel preconditioning, Numer. Math. 63 (1992) 315–344.
[58] J. De Keyser, D. Roose, Partitioning and mapping adaptive multigrid hierarchies on distributed memory computers, Tech. Rep. TW 166, Univ. Leuven, Department of Computer Science, 1992.
[59] Proceedings of Domain Decomposition Methods in Science and Engineering, SIAM/AMS/Wiley, 1988–1997.
[60] W. Dörfler, A convergent adaptive algorithm for Poisson's equation, SIAM J. Numer. Anal. (1996) 1106–1124.
[61] C.C. Douglas, Parallel multilevel and multigrid methods, SIAM News 25 (1992) 14–15.
[62] C.C. Douglas, Caching in with multigrid algorithms: problems in two dimensions, Paral. Alg. Appl. 9 (1996) 195–204.
[63] M. Dryja, An algorithm with a capacitance matrix for a variational-difference scheme, in: G.I. Marchuk (Ed.), Variational-Difference Methods in Mathematical Physics, USSR Academy of Sciences, Novosibirsk, 1981, pp. 63–73.
[64] M. Dryja, B.F. Smith, O.B. Widlund, Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions, SIAM J. Numer. Anal. 31 (1994) 1662–1694.
[65] M. Dryja, O.B. Widlund, Towards a unified theory of domain decomposition algorithms for elliptic problems, in: T.F. Chan, R.G. ndJ.Périaux, O.B. Widlund (Eds.), Proceedings of the Domain Decomposition Methods 3, SIAM, 1990.
[66] M. Dryja, O.B. Widlund, Additive Schwarz methods for elliptic finite element problems in three dimensions, in: T.F. Chan, D.E. Keyes, G. Meurant, J.S. Scroggs, and R.G. Voigt, (Eds.), Proceedings of the Domain Decomposition Methods 5, SIAM, 1992.
[67] M. Dryja, O.B. Widlund, Schwarz methods of Neumann–Neumann type for three-dimensional elliptic finite element problems, Comm. Pure Appl. Math. 48 (1995) 121–155.
[68] C. Farhat, F.-X. Roux, Implicit parallel processing in structural mechanics, Comput. Mech. Adv. 2 (1994).
[69] R.P. Fedorenko, A relaxation method for solving elliptic difference equations, Sov. J. Comput. Math. Math. Phys. 1 (1961) 922–927.
[70] The speed of convergence of one iterative process, Sov. J. Comput. Math. Math. Phys. 4 (1964) 559–563.
[71] J. Fuhrmann, On the convergence of algebraically defined multigrid methods, Tech. Rep., Institut für Angwandte Analysis und Stochastik, Berlin, 1992.
[72] G.H. Golub, D. Mayers, The use of preconditioning over irregular regions, in: R. Glowinski, J.-L. Lions (Eds.), Computing Methods in Applied Sciences and Engineering VI, North-Holland, Amsterdam, 1984, pp. 3–14.
[73] T. Grauschopf, M. Griebel, H. Regler, Additive multilevel-preconditioners based on bilinear interpolation, matrix dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic PDEs, Appl. Numer. Math. 23 (1997) 63–96.
[74] M. Griebel, Multilevel algorithms considered as iterative methods on semidefinite systems, SIAM Int. J. Sci. Stat. 15 (1994) 547–565.
[75] M. Griebel, Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen, Skripten zur Numerik, Teubner, Stuttgart, 1994.

[76] M. Griebel, Parallel point-oriented multilevel methods, in Multigrid Methods IV, EMG93, P. Hemker, P. Wesseling (Eds.), International Series of Numerical Mathematics, 1994, pp. 215–232.
[77] M. Griebel, T. Neunhoeffer, Parallel point- and domain-oriented multilevel methods for elliptic PDE's on workstation networks, J. Comp. Appl. Math. 66 (1996) 267–268.
[78] M. Griebel, P. Oswald, Remarks on the abstract theory of additive and multiplicative Schwarz methods, Tech. Rep. TUM-I9314, TU München, 1993.
[79] M. Griebel, P. Oswald, On the abstract theory of additive and multiplicative Schwarz algorithms, Numer. Math. 70 (1995) 163–180.
[80] M. Griebel, P. Oswald, Tensor product type subspace splittings and multilevel iterative methods for anisotropic problems, Adv. in Comput. Math. 4 (1995) 171–206.
[81] M. Griebel, G. Zumbusch, Hash-storage techniques for adaptive multilevel solvers and their domain decomposition parallelization, in: J. Mandel, C. Farhat, X.-C. Cai (Eds.), Proceedings of the Domain Decomposition Methods 10, Contemporary Mathematics, vol. 218, AMS, Providence, Rhode Island, 1998, pp. 279–286.
[82] C.E. Grosch, Poisson solvers on large array computers, in: B.L. Buzbee, J.F. Morrison (Eds.), Proceedings of the 1978 LANL Workshop on Vector and Parallel Processors, 1978.
[83] G. Haase, U. Langer, A. Meyer, A new approach to the Dirichlet domain decomposition method, in: S. Hengst (Ed.), Fifth Multigrid Seminar, Report R-MATH-09/90, Karl-Weierstrass-Institut, Eberswalde, Berlin, 1990, pp. 1–59.
[84] G. Haase, U. Langer, A. Meyer, Domain decomposition methods with inexact subdomain solvers, J. Numer. Linear Algebra Appl. 1 (1991) 27–41.
[85] G. Haase, U. Langer, A. Meyer, S.V. Nepomnyaschikh, Hierarchical extension operators and local multigrid methods in domain decomposition preconditioners, E. W. J. Numer. Math. 2 (1994) 173–193.
[86] W. Hackbusch, Ein iteratives Verfahren zur schnellen Auflösung elliptischer Randwertprobleme, Tech. Rep. 76–12, Mathematisches Institut, Universität zu Köln, 1976.
[87] W. Hackbusch, A fast numerical method for elliptic boundary value problems with variable coefficients, in: E.H. Hirschel, W. Geller, (Eds.), Proceedings of the Second GAMM-Conference on Numerical Methods in Fluid Mechanics, DFVLR, Köln, 1977, pp. 50–57.
[88] W. Hackbusch, Multi-grid convergence theory, in Multigrid Methods, W. Hackbusch, U. Trottenberg (Eds.), Lecture Notes in Mathematics, vol. 960, Springer, Berlin, 1982, pp. 177–219.
[89] W. Hackbusch, Local defect correction methods and domain decompositon techniques, in: K. Böhmer, H.J. Stetter, (Eds.), Defect Correction Methods: Theory and Applications, Computing Suppl. 5, Springer, Vienna, 1984, pp. 89–113.
[90] W. Hackbusch, Multi-Grid Methods and Applications, Springer, Berlin, 1985.
[91] W. Hackbusch, Iterative solution of large sparse systems of equations, Springer, Berlin, 1994.
[92] W. Hackbusch, S.A. Sauter, Composite finite elements for the approximation of PDEs on domains with complicated micro-structures, Numer. Math. 75 (1997) 447–472.
[93] P.W. Hemker, B. Koren, A non-linear multigrid method for the steady Euler equations, in: A. Dervieux, B. Leer, J. Périaux, A. Rizzi (Eds.), Numerical Simulation of Compressible Euler Flows, Notes on Numerical Fluid Mechanics, vol. 26, Vieweg, Braunschweig, 1989, pp. 175–196.
[94] B. Hendrickson, R. Leland, An improved spectral graph partitioning algorithm for mapping parallel computations, SIAM J. Sci. Stat. Comput. 16 (1995) 452–469.
[95] R. Hiptmair, Multilevel Preconditioning for Mixed Problems in Three Dimensions, Ph.D. Thesis, Mathematisches Institut, Universität Augsburg, 1996.
[96] M.T. Jones, P.E. Plassmann, Parallel algorithms for adaptive mesh refinement, SIAM J. Scientific Computing 18 (1997) 686–708.
[97] G. Karypis, V. Kumar, Multilevel *k*-way graph partitioning for irregular graphs, J. Parallel and Distributed Comp. 48 (1998) 96–129.
[98] D.E. Knuth, The Art of Computer Programming, vol. 3, Addison-Wesley, Reading, MA, 1975.
[99] S.R. Kohn, S.B. Baden, A robust parallel programming model for dynamic non-uniform scientific computations, Tech. Rep. CS94–354, UCSD, Department of Computer Science, 1994.
[100] R. Kornhuber, H. Yserentant, Multilevel methods for elliptic problems on domains not resolved by the coarse grid, in: Proceedings of the Domain Decomposition Methods 7, Contemporary Mathematics, vol. 180, AMS, Providence, Rhode Island, 1994, pp. 49–60.
[101] M.H. Lallemand, H. Steve, A. Dervieux, Unstructured multigridding by volume agglomeration: Current status, Comput. Fluids 21 (1992) 397–433.
[102] P. Le Tallec, Domain decomposition methods in computational mechanics, in: J.T. Oden (Ed.), Computational Mechanics Advances, North-Holland, Amsterdam, 1994, pp. 121–220.
[103] P. Le Tallec, Y.-H. de Roeck, M. Vidrascu, Domain–decomposition methods for large linearly elliptic three-dimensional problems, J. Comput. Appl. Math. 34 (1991) 93–117.
[104] C.-O. Lee, Multigrid methods for the pure traction problem of linear elasticity: Mixed formulation, SIAM J. Numer. Anal. 35 (1998) 121–145.
[105] P. Leinen, Ein schneller adaptiver Löser für elliptische Randwertprobleme auf Seriell- und Parallelrechnern, Ph.D. Thesis, Universität Dortmund, 1990.
[106] M. Lemke, D. Quinlan, Fast adaptive composite grid methods on distributed parallel architectures, Comm. Appl. Numer. Meth. 8 (1992) 609–619.
[107] P.-L. Lions, Interprétation stochastique de la méthode alternée de Schwarz, C.R. Acad. Sci. Paris 268 (1978) 325–328.

[108] R. Löhner, K. Morgan, An unstructured multigrid method for elliptic problems, Int. J. Numer. Meth. Engng. 24 (1987) 101–115.
[109] Y. Maday, C. Mavriplis, A.T. Patera, Nonconforming mortar element methods: Application to spectral discretizations, in: T.F. Chan, R. Glowinski, J. Périaux, O.B. Widlund (Eds.), Proceedings of the Domain Decomposition Methods 2, SIAM, Philadelphia, 1989, pp. 392–418.
[110] J. Mandel, Balancing domain decomposition, Comm. Numer. Meth. Engrg. 9 (1993) 233–241.
[111] A.M. Matsokin, S.V. Nepomnyaschikh, Schwarz alternating method in a subspace, Sov. Math. (Izv. vuz.) 29 (1985) 78–84.
[112] O.A. Mc Bryan, P.O. Frederickson, J. Linden, A. Schüller, K. Solchenbach, K. Stüben, C.A. Thole, U. Trottenberg, Multigrid methods on parallel computers – a survey of recent developments, Impact of Computing in Science and Engineering 3 (1991) 1–75.
[113] S.F. McCormick, Fast adaptive composite grid (FAC) methods: Theory for the variational case, in: K. Böhmer, H.J. Stetter (Eds.), Defect Correction Methods: Theory and Applications, Computing Suppl. 5, Springer, Vienna, 1984, pp. 115–121.
[114] S.F. McCormick, Multilevel Adaptive Methods for Partial Differential Equations, Frontiers in Applied Mathematics, vol. 6, SIAM Books, Philadelphia, 1989.
[115] W.F. Mitchell, A comparison of adaptive refinement techniques for elliptic problems, ACM Trans. Math. Software 15 (1989) 326–347.
[116] W.F. Mitchell, A parallel multigrid method using the full domain partition, Electronic Transactions on Numerical Analysis, (97). Special issue for Proceedings of the Eighth Copper Mountain Conference on Multigrid Methods.
[117] D. Morgenstern, Begründung des alternierenden Verfahrens durch Orthogonalprojektion, ZAMM 36 (1956) 7–8.
[118] Multi-grid methods I-V, Springer Berlin/ Birkhäuser, Basel, 1982/ 1986/ 1991/1993/ 1997.
[119] S.V. Nepomnyaschikh, Mesh theorems on traces, normalization of function traces and their inversion, Sov. J. Numer. Anal. Math. Modelling 6 (1991) 223–242.
[120] S.V. Nepomnyaschikh, Decomposition and fictitious domain methods for elliptic boundary value problems, in: T.F. Chan, D.E. Keyes, G. Meurant, J.S. Scroggs, R.G. Voigt (Eds.), Proceedings of the Domain Decomposition Methods 5, SIAM, 1992.
[121] R.A. Nicolaides, On the l^2 convergence of an algorithm for solving finite element equations, Math. Comp. 31 (1977) 892–906.
[122] R. Niedermeier, K. Reinhardt, P. Sanders, Towards optimal locality in mesh-indexings, in: Proceedings of the 11th International Symposium on Fundamentals of Computation Theory, No. 1279 in Lecture Notes in Computer Science, Springer, Berlin, 1997, pp. 364–375.
[123] J.T. Oden, A. Patra, Y. Feng, Domain decomposition for adaptive hp finite element methods, in: Proceedings of the Domain Decomposition 7, vol. 180 of Contemporary Mathematics, AMS, 1994, pp. 295–301.
[124] P. Oswald, On discrete norm estimates realted to multilevel preconditioners in the finite element method, in: K.G. Ivanov, P. Petrushev, B. Sendov (Eds.), Constructive Theory of Functions, Proceedings of the International Conference in Varna 1991; Sofia, 1992, Bulg. Acad. Sci., pp. 203–214.
[125] P. Oswald, On a BPX-preconditioner for P1 elements, Computing 51 (1993) 125–133.
[126] P. Oswald, Stable splittings of Sobolev spaces and applications, Tech. Rep. Math/93/5, FSU Jena, 1993.
[127] P. Oswald, Multilevel Finite Element Approximation, Teubner, 1994.
[128] P. Oswald, On the convergence rate of SOR: A worst case estimate, Computing 52 (1994) 245–255.
[129] C.-W. Ou, S. Ranka, G. Fox, Fast and parallel mapping algorithms for irregular and adaptive problems, in: Proceedings of International Conference on Parallel and Distributed Systems, 1993.
[130] E.E. Ovtchinnikov, L.S. Xanthis, Iterative subspace correction methods for thin elastic structures and Korn's type inequality in subspaces, Tech. Rep., University of Westminster, London, 1997.
[131] M. Parashar, J.C. Browne, On partitioning dynamic adaptive grid hierarchies, in: Proceedings of the 29th Annual Hawai International Conference on System Sciences, 1996.
[132] P. Peisker, D. Braess, A conjugate gradient method and a multigrid method for Morley's finite element approximation of the biharmonic equation, Numer. Math. 50 (1987) 567–586.
[133] J.R. Pilkington, S.B. Baden, Partitioning with spacefilling curves, Tech. Rep. CS94–349, UCSD, Department of Computer Science, 1994.
[134] A. Pothen, Graph partitioning algorithms with applications to scientific computing, in: D.E. Keyes, A. Sameh, V. Venkatakrishnan (Eds.), Parallel Numerical Algorithms, Kluwer, Dordrecht, 1997, pp. 323–368.
[135] A. Pothen, H. Simon, K.-P. Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM J. Matrix Anal. Appl. 11 (1990) 430–452.
[136] J.S. Przemieniecki, Theory of Matrix Structural Analysis, ch. 9, McGraw-Hill, New York, 1968.
[137] A. Reusken, Multigrid with matrix dependent transfer operators for a singular perturbation problem, Computing 50 (1993) 199–211.
[138] W.C. Rheinboldt, C.K. Mesztenyi, On a data structure for adaptive finite element mesh refinements, ACM Trans. Math. Softw. 6 (1980) 166–187.
[139] H. Ritzdorf, K. Stüben, Adaptive multigrid on distributed memory computers, in: Multigrid Methods IV, Proceedings of the Fourth European Multigrid Conference, Amsterdam, July 6–9, 1993; vol. 116 of ISNM, Basel, 1994, Birkhäuser, pp. 77–95.
[140] M.C. Rivara, Algorithms for refining triangular grids suitable for adaptive and multigrid techniques, Int. J. Numer. Meth. Engrg. 20 (1984) 745–756.
[141] S. Roberts, S. Kalyanasundaram, M. Cardew-Hall, W. Clarke, A key based parallel adaptive refinement technique for finite element methods, in: Proceedings of the Computational Techniques and Applications: CTAC '97, World Scientific, to appear.
[142] U. Rüde, Mathematical and Computational Techniques for Multilevel Adaptive Methods, Frontiers in Applied Mathematics, vol. 13, SIAM, Philadelphia, 1993.

[143] J. W. Ruge, K. Stüben, Algebraic multigrid (AMG), in: S.F. McCormick, (Ed.), Multigrid Methods, Frontiers in Applied Mathematics, vol. 3, SIAM, Philadelphia, PA, 1987, pp. 73–130.
[144] H. Sagan, Space-Filling Curves, Springer, New York, 1994.
[145] H.A. Schwarz, Über einige Abbildungsaufgaben, Vierteljahresschrift Naturforsch. Ges. Zürich 15 (1870) 272–286.
[146] B.F. Smith, Domain decomposition algorithms for the partial differential equations of linear elasticity, Ph.D. Thesis, Department of Computer Science, Courant Institute, New York, 1990.
[147] B.F. Smith, A domain decomposition algorithm for elliptic problems in three dimensions, Numer. Math. 60 (1991) 219–234.
[148] B.F. Smith, P.E. Bjørstad, W.D. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, New York, 1996.
[149] B.F. Smith, O.B. Widlund, A domain decomposition algorithm using a hierarchical basis, SIAM J. Sci. Stat. Comput. 11 (1990) 1212–1220.
[150] S.L. Sobolev, The Schwarz algorithm in the theory of elasticity, Sokl. Acad. N. USSR, IV (XIII) (1936) 243–246.
[151] K. Solchenbach, C.A. Thole, U. Trottenberg, Parallel multigrid methods: Implementation on SUPRENUM-like architectures and applications, in: Supercomputing, Lecture Notes in Computer Science, vol. 297, Springer, New York, 1987, pp. 28–42.
[152] L. Stals, Parallel Implementation of Multigrid Methods, Ph.D. Thesis, Department of Mathematics, Australian National University, 1995.
[153] L. Stals, U. Rüde, Techniques for improving the data locality of iterative methods, Tech. Rep. MRR97–038, School of Mathematical Sciences, Australian National University, 1997.
[154] R.P. Stevenson, Robustness of the additive and multiplicative frequency decomposition multilevel method, Computing 54 (1995) 331–346.
[155] K. Stüben, U. Trottenberg, Multigrid methods: Fundamental algorithms, model problem analysis and applications, in: Multigrid Methods, W. Hackbusch, U. Trottenberg (Eds.), Lecture Notes in Mathematics, vol. 960, Springer, Berlin, 1982, pp. 1–176.
[156] X.-C. Tai, J. Xu, Global convergence of subspace correction methods for convex optimization problems, Math. Comp., submitted.
[157] C.H. Tong, T.F. Chan, C.C.J. Kuo, A domain decomposition preconditioner based on a change to a multilevel nodal basis, SIAM J. Sci. Stat. Comput. 12 (1991) 1486–1495.
[158] P. Vaněk, J. Mandel, M. Brezina, Algebraic multigrid based on smoothed aggregation for second and fourth order problems, Computing 56 (1996) 179–196.
[159] R. Verfürth, A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques, Wiley & Teubner, 1996.
[160] C.H. Walshaw, M. Berzins, Dynamic load-balancing for PDE solvers on adaptive unstructured meshes, Tech. Rep., Computer Studies 92.32, University of Leeds, 1992.
[161] M. Warren, J. Salmon, A portable parallel particle program, Comput. Phys. Comm. 87 (1995) 266–290.
[162] P. Wesseling, The rate of convergence of a multiple grid method, in: G.A. Watson (Ed.), Numerical Analysis, Lecture Notes in Mathematics, vol. 773, Berlin, Springer, 1980, pp. 164–180.
[163] P. Wesseling, An Introduction to Multigrid Methods, Wiley, Chichester, 1992.
[164] O.B. Widlund, Some Schwarz methods for symmetric and nonsymmetric elliptic problems, in: T.F. Chan, D.E. Keyes, G. Meurant, J.S. Scroggs, R.G. Voigt (Eds.), Proceedings of the Domain Decomposition Methods 5, SIAM, 1992.
[165] G. Wittum, On the robustness of ILU-smoothing, SIAM J. Sci. Stat. Comput. 10 (1989) 699–717.
[166] J. Xu, Theory of Multilevel Methods, Ph.D. Thesis, Cornell University, 1989.
[167] J. Xu, Iterative methods by space decomposition and subspace correction, SIAM Rev. 34 (1992) 581–613.
[168] J. Xu, Two-grid discretization techniques for linear and nonlinear PDEs, SIAM J. Num. Anal. 33 (1996) 1759–1777.
[169] J. Xu, J. Zou, Non-overlapping domain decomposition methods, SIAM Rev., submitted.
[170] H. Yserentant, On the multi-level splitting of finite element spaces, Numer. Math. 49 (1986) 379–412.
[171] H. Yserentant, Old and new convergence proofs for multigrid methods, in: Acta Numerica, vol. 2, Cambridge University Press, New York, 1993, pp. 285–326.
[172] P.M.d. Zeeuw, Matrix-dependent prolongations and restrictions in a black-box multigrid solver, J. Comput. Appl. Math. 33 (1990) 1–27.
[173] X. Zhang, Multilevel Schwarz methods, Numer. Math. (1992) 521–539.
[174] X. Zhang, Multilevel Schwarz methods for the biharmonic Dirichlet problem, SIAM J. Sci. Comput. 15 (1994) 612–644.
[175] G. Zumbusch, Adaptive parallele Multilevel-Methoden zur Lösung elliptischer Randwertprobleme, SFB-Report 342/19/91A, TUM-I9127, TU München, 1991.
[176] G. Zumbusch, Simultanous $h-p$ adaptation in multilevel finite elements, Ph.D. Thesis, FU Berlin, 1995.
[177] G. Zumbusch, Parnass2: A cluster of PCs. http://wwwwissrech.iam.uni-bonn.de/research/projects/parnass2/, 1998.