Multigrid Methods on Adaptively Refined Grids

Peter Bastian* Christian Wieners‡

July 31, 2006

*Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany

‡Institut für Praktische Mathematik, Universität Karlsruhe, Englerstraße 2, 76128 Karlsruhe, Germany

The use of multigrid solvers in the adaptive finite element method yields a powerful tool for solving large-scale partial differential equations that exhibit localized features such as singularities or shocks. The authors first give some historical background, then describe the basic method and related theory, and finish up with numerical demonstrations of the performance and utility of the method on interesting 3d problems.

Multigrid methods are solution methods of optimal complexity for a broad class of large and sparse linear systems, see [28, this issue]. Adaptive methods provide a sequence of discretizations of partial differential equations (PDEs) with optimal approximation quality. Here, we discuss the combination of both techniques: we consider multigrid solution methods of optimal complexity on adaptively refined grids.

In many challenging applications, the solution of the overall problem requires the full combination of effective linear and nonlinear solvers, accurate discretizations, adaptivity in space and time, and, last but not least, parallelism. Thus, multigrid and adaptivity are now key technologies for numerical simulations in PDEs.

Here, we give a short overview on the background and on the special interaction of both concepts. In the first part, we summarize the main ideas and milestones in the development of adaptive multigrid methods. Then, studying simple problems and geometries, we explain basic features about the interplay of multigrid and adaptivity. We present local multigrid methods in the framework of subspace correction methods. Finally, we compare different algorithmic concepts by numerical tests for the model problem.

1 A short historical review

Already one of the first publications on multigrid methods by A. Brandt [13] outlined the combination of multigrid and adaptivity (see also [2]). In the beginning of the 1980s S. McCormick and coworkers developed the fast adaptive composite (FAC) grid method [17, 19]. This was one of the first methods that looked at ways of improving the computational efficiency of the

multigrid method by restricting the fine grid to local subdomains where the error is large. They also introduced a variant that removes the inter-grid data dependencies inherent in standard multigrid methods thus improving the parallel efficiency [18]. While the FAC method has been invented for locally refined structured grids at about the same time, M. C. Rivara [22] developed an optimal complexity multigrid method in the context of an adaptive finite element procedure on triangular grids in 2d with bisection refinement. Mitchell [20] developed a similar method using higher-order finite element methods. Another type of grid refinement algorithm, the so-called red/green refinement scheme, was pioneered by R. Bank and coworkers in [4]. It was the basis of PLTMG, one of the most successful implementations of the adaptive finite element method. As a solver, this code used the hierarchical basis multigrid method [3], which was a variant of the hierarchical basis method [29] developed earlier by H. Yserentant. The hierarchical basis methods are less efficient in three space dimensions, so new ideas were required. Bramble, Pasciak and Xu [12] were the first to prove optimal convergence properties for a multigrid algorithm that works on unstructured, locally refined grid independent of the space dimension. A more robust and efficient variant, the local multigrid method, has been shown to work efficiently also for more complicated problems in [5]. A breakthrough in a unified presentation and analysis of the various methods has been achieved by J. Xu with the notion of subspace correction methods in [26]. Unifying presentations that elegantly integrate adaptive refinement, local sharp error measures and fast solvers have been given by U. Rüde [23] and M. Griebel [15]. While a parallel implementation of adaptive multigrid methods in the context of locally refined structured grids was developed in the late 1980s by S. McCormick and D. Quinlan [18], the first optimal complexity implementation of multigrid methods in the context of fully unstructured grids in two space dimensions were given by one of the authors (P. B.) in [6] and by Mitchel [21]. The first implementation in three space dimensions including time-dependent problems was given by S. Lang in [16].

2 Adaptive Finite Element Framework

As a prototype application, we consider the linear elliptic PDE

$$\begin{aligned} -\nabla \cdot \big(K(x)\nabla u\big) &= f \quad \text{in } \Omega \subset \mathbb{R}^d, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

which models, e. g., stationary heat conduction or fully saturated groundwater flow in d space dimensions. Here, $K(x) : \Omega \to \mathbb{R}^{d\times d}$ is a (in general position dependent) diffusion tensor, $u : \Omega \to \mathbb{R}$ is the unknown solution (temperature, pressure) and $f : \Omega \to \mathbb{R}$ is the source/sink term. To keep the technical details at a minimum, we restrict ourselves to homogeneous Dirichlet boundary conditions, but the methods presented in the sequel can handle all kinds of boundary conditions.

The finite element method (FEM) is one of the most popular methods for the numerical solution of PDEs. This is due to its applicability to a wide range of problems and the existing large body of mathematical theory (see, e. g., [14] for an introduction). The FEM is based on the weak formulation of (1), which reads as follows: Find $u \in V = H_0^1(\Omega)$ such that

$$\underbrace{\int_\Omega (K(x)\nabla u)\cdot\nabla v\,dx}_{a(u,v)} = \underbrace{\int_\Omega f v\,dx}_{\ell(v)} \quad \text{for all } v \in V\,. \tag{2}$$

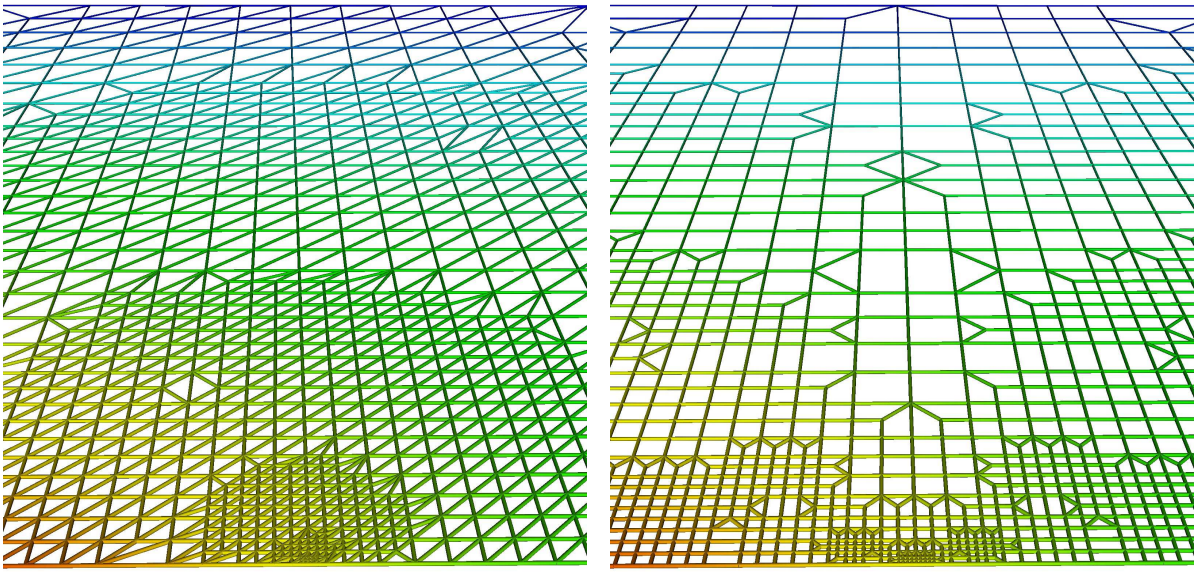

Figure 1: Grids generated by hierarchical adaptive grid refinement in two space dimensions. The grid is refined in order to resolve a point singularity. The left picture shows refinement obtained with the red/green refinement algorithm using triangles. The right picture shows red/green type refinement using quadrilaterals and triangles. Both grids were generated with the PDE software Dune/UG [8, 7]. Visualization is done with ParaView/VTK.

This formulation follows from (1) by multiplication with a test function v and integration by parts. Here, $H_0^1(\Omega)$ is the Sobolev space of functions that, together with all first-order derivatives, are square integrable and that are zero on the boundary $\partial\Omega$, $a(u,v)\colon V \times V \to \mathbb{R}$ is a symmetric and continuous bilinear form, and $\ell(v)\colon V \to \mathbb{R}$ is a bounded linear functional. Under certain assumptions, the solution of the weak formulation (2) and the PDE (1) are equivalent.

To solve (2) on a computer, the infinite-dimensional function space V is replaced by a finite dimensional approximation V_h consisting of continuous and piecewise polynomial (in the simplest case, piecewise linear) functions. This requires the partitioning of Ω into elements of simple shape (e. g., triangles, quadrilaterals or tetrahedra), called a computational grid. The subscript h in V_h denotes the dependence of the quality of approximation on the diameter h of the elements in the grid.

There are many different types of grids, and grid generation is a very important subject in the practical application of the FEM. In adaptive methods, grids are typically generated through hierarchical grid refinement, a process that is explained in more detail below. Figure 1 shows two grids generated in this way. The adaptive finite element method then consists of the following steps:

1. Generate an initial grid. This grid can be quite coarse, but it should resolve important geometric details of the domain Ω.

2. Compute a numerical approximation $u_h \in V_h$ to the solution u of the weak formulation (2) using the FEM. This involves the solution of a large and sparse linear system of equations.

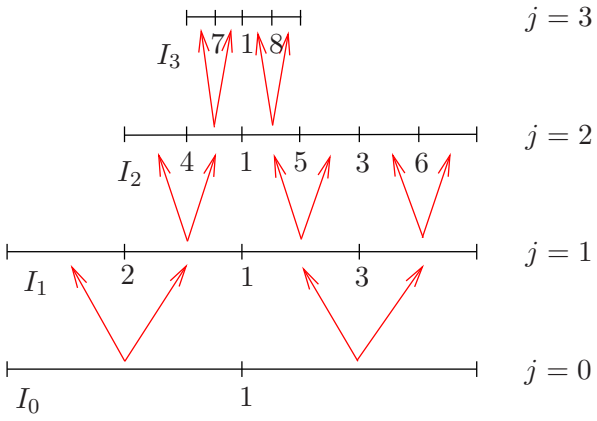

Figure 2: Hierarchical grid refinement for a domain $\Omega = (0, 1)$. The grid is constructed as follows: start with an intentionally coarse grid which is indicated as level $j = 0$. Then each element is subdivided into two elements of half the size each resulting in the grid on level $j = 1$. On this level only three out of the four elements are refined, which yields grid level $j = 2$, and then another two out of the six elements are refined to obtain grid level $j = 3$.

3. Compute an estimate E of the error $\|u - u_h\|$ in some norm using a posteriori error estimators (see, e. g., [1]). If $E < \text{TOL}$, then stop.

4. Tag elements of the grid for refinement where the local error is large. An optimal refinement strategy tries to equilibrate the error in each element. In time-dependent problems, coarsening (removal of refinements from previous time steps) is also possible.

5. Refine the grid according to the refinement tags.

6. Interpolate the approximate solution u_h from the previous grid to the new grid as an initial guess. Go to step 2.

3 Hierarchical Grid Refinement and Multilevel Basis

The adaptive FEM produces a sequence of finite-dimensional function spaces V_h that are adapted to the particular PDE problem to be solved. The construction of V_h is closely related to hierarchical grid generation and to the multigrid solution of the arising large and sparse systems of linear equations.

The process of hierarchical grid generation is illustrated in Figure 2. The initial grid, also called coarse or macro grid, is intentionally coarse. Here it consists of two line elements and three nodes. Then individual elements are recursively subdivided into smaller elements, which leads to a tree structure. This enables an efficient implementation of the refinement algorithm. The elements on a given grid level j form a subdomain $\Omega_j \subseteq \Omega$. The local refinement gives rise to a nested sequence of subdomains

$$\Omega_J \subset \Omega_{J-1} \subset \cdots \subset \Omega_1 \subset \Omega_0 = \Omega$$

Here and throughout the rest of the paper J denotes the highest level in the grid hierarchy.

The nodes of the grid are each assigned a number as follows (see Figure 2):

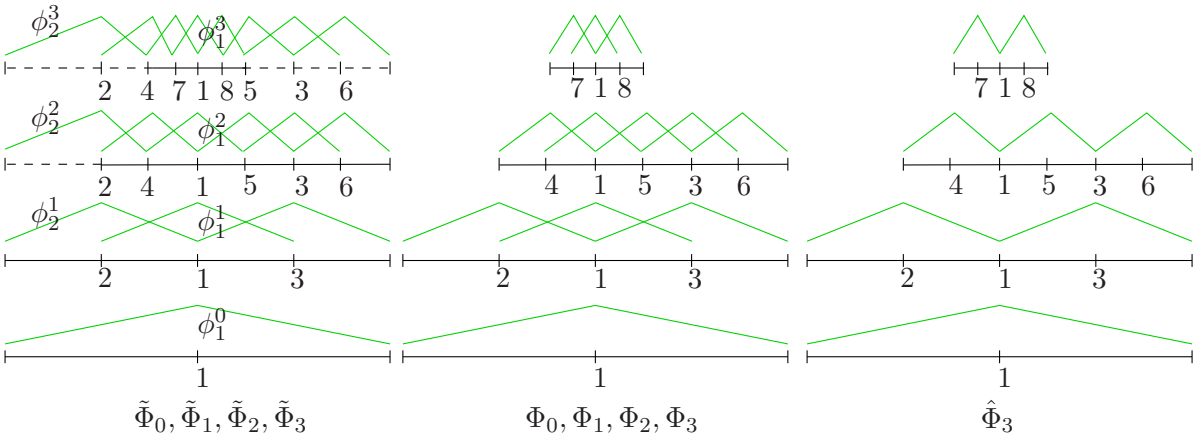

Figure 3: Nodal basis functions corresponding to the grid hierarchy from figure 2. On the left the global nodal bases $\tilde{\Phi}_0, \tilde{\Phi}_1, \tilde{\Phi}_2, \tilde{\Phi}_3$ are shown. The middle and right drawing illustrate the local nodal basis $\Phi_0, \Phi_1, \Phi_2, \Phi_3$ and the hierarchical basis $\hat{\Phi}_3$ for the finest level $J = 3$.

- The *interior* nodes of grid level $j = 0$ are assigned unique numbers that form the index set $I_0 \subset \mathbb{N}$.
- On grid level $j > 0$ the new nodes created by the refinement process are assigned unique numbers that have not been used on coarser levels. Those nodes that were already present on level $j-1$ are assigned the same number as on the coarser level. The indices for nodes on level j form the index set I_j.

An important step in the finite element method is the construction of a basis for the finite element space V_h. One possibility is the so-called nodal basis which consists of nodal basis functions. A nodal basis function $\phi\colon \bar{\Omega} \to \mathbb{R}$ has the value 1 at one nodal point of a grid, the value 0 at all other nodal points and is linear on each element.

Starting with the hierarchical grid construction from Figure 2 we can complete each grid level by all elements from coarser levels that have not been refined. This is illustrated on the left in Figure 3. In this extended construction each grid level forms a partitioning of the domain Ω. The finest grid on level J is called the *leaf grid*. The leaf grid is formed by all the elements from Figure 2 that are leaves of their refinement tree, and this is the grid used in step 2 of the adaptive finite element algorithm. Note that figures 1 and 4 only show the leaf grid.

Now, nodal basis functions can be defined on each level of the extended hierarchical grid as shown in Figure 3 on the left. Due to the Dirichlet boundary conditions of our model problem (1), only basis functions corresponding to interior grid nodes are required. By ϕ_i^j, $i \in \bigcup_{k=0}^{j} I_k$, we denote the nodal basis function corresponding to the i'th grid node on level j.

We are now in a position to define several sets of basis functions that are relevant in the context of adaptive multigrid methods. We begin with the global nodal basis

$$\tilde{\Phi}_j = \left\{ \phi_i^j \;\middle|\; i \in \bigcup_{k=0}^{j} I_k \right\}, \qquad 0 \le j \le J\,,$$

consisting of all the basis functions defined on one grid level of the extended hierarchical grid

as shown on the left in Figure 3. On the finest level J,

$$\Phi_h = \tilde{\Phi}_J$$

is the set of nodal basis functions on the leaf grid. A basis function $\phi_i \in \Phi_h$ is uniquely determined by its index $i \in I_h = \bigcup_{j=0}^{J} I_j$, i. e., we can safely omit the level superscript.

Next, the local nodal basis

$$\Phi_j = \{\phi_i^j \,|\, i \in I_j\}, \qquad 0 \le j \le J.$$

is made up by the basis functions on level j corresponding to the subdomain Ω_j. These are shown in the middle in Figure 3.

Finally, the hierarchical basis on level j is given by

$$\hat{\Phi}_j = \Phi_0 \cup \bigcup_{k=1}^{j} \{\phi_i^k \,|\, i \in I_k \setminus I_{k-1}\}, \qquad 0 \le j \le J.$$

It picks ϕ_i^k from the coarsest level $k \le j$ where node i is present. It is illustrated for $j = 3$ on the right in Figure 3.

Each of the bases defined above generates a corresponding finite element space:

$$V_j = \operatorname{span} \Phi_j, \qquad \tilde{V}_j = \operatorname{span} \tilde{\Phi}_j, \qquad \hat{V}_j = \operatorname{span} \hat{\Phi}_j, \qquad 0 \le j \le J.$$

In particular, $V_h = \tilde{V}_J = \hat{V}_J$ is the leaf grid finite element space, generated by both the global nodal basis and the hierarchical basis.

The hierarchical grid construction can be extended to multiple space dimensions. The FEM requires that the maximum interior angle in any element is bounded away from 180 degrees, a requirement that has led to a number of different grid refinement strategies. Most of these refinement strategies subdivide an element in $n > 2$ smaller elements. The binary tree in Figure 2 is then replaced by an n-ary tree.

4 Subspace Correction Methods

The finite element solution $u_h = \sum_{i \in I_h} \underline{u}_i \phi_i$, ϕ_i the nodal basis functions on the leaf grid, is determined by the variational equation

$$a(u_h, \phi_i) \;\; = \;\; \ell(\phi_i), \quad \phi_i \in \Phi_h,$$

which is equivalent to the linear system

$$\underline{A}\,\underline{u} = \underline{f}, \qquad \underline{A} = \big(a(\phi_j, \phi_i)\big)_{\phi_i, \phi_j \in \Phi_h}, \; \underline{f} = \big(\ell(\phi_i)\big)_{\phi_i \in \Phi_h} .$$

Note that the finite element stiffness matrix $\underline{A}$ with respect to the nodal basis Φ_h is sparse, i. e., the matrix-vector product $\underline{A}\,\underline{u}$ requires $O(N_h)$ operations, where $N_h = \dim V_h$, the number of interior grid points in the leaf grid. If N_h is large, direct solving is quite expensive or even practically impossible. On the other hand, the iteration count for simple linear iterative methods, such as Gauß-Seidel, increases by a factor of four with each refinement of the grid which makes these methods also impractical.

4.1 Multigrid methods with global smoothing

The standard multigrid method (as presented, e. g., by [28] in this volume) can be applied in a straightforward way to locally refined grids by considering the extended hierarchical grid construction shown on the left in Figure 3. Therefore, we consider the hierarchy of linear systems

$$\underline{A}_j \underline{u}_j = \underline{f}_j, \qquad \underline{A}_j = \big(a(\phi_k, \phi_i)\big)_{\phi_k, \phi_i \in \tilde{\Phi}_j},\ \underline{f}_j = \big(\ell(\phi_i)\big)_{\phi_i \in \tilde{\Phi}_j},\ 0 \le j \le J \tag{3}$$

obtained from the global nodal basis $\tilde{\Phi}_j$ on each grid level. Then, a multigrid preconditioner can be constructed as follows:

(S0) Set the current level $j = J$ to the finest level, and start with the correction $\underline{c} = 0$. Compute the residual $\underline{r}_j = \underline{f}_j - \underline{A}_j\, \underline{u}_j^*$ from the current iterate $\underline{u}_j^*$.

(S1) If $j > 0$, apply a simple preconditioner (the so-called smoother) and compute the correction on level j. Update the residual and then restrict the residual on level j to the next coarser level $j - 1$.
Set $j := j - 1$ and repeat (S1) until $j = 0$.

(S2) Now, for $j = 0$, compute the coarse grid correction by (approximately) solving $\underline{A}_0 \underline{c}_0 = \underline{r}_0$ (in general, the coarse grid is assumed to be small enough for direct solving). Set $j = 1$.

(S3) Interpolate the correction from level $j - 1$ to level j, update the residual and add the interpolated correction to the current correction on level j. Again, perform a smoothing step and update the correction.
If $j < J$, set $j := j + 1$ and repeat (S2) until $j = J$.
The final correction on level J is the result of the multigrid preconditioner.

Since each level of the grid used in the standard multigrid algorithm covers the full domain Ω we speak of multigrid with *global smoothing*. The implementation of the multigrid cycle with global smoothing requires $O(M_h)$ operations in total, where $M_h = \dim \tilde{V}_1 + \cdots + \dim \tilde{V}_J$, assuming that the cost for the coarsest grid is negligible. Under the assumption of geometric growth, i. e., $\dim \tilde{V}_j \ge q \dim \tilde{V}_{j-1}$, $q > 1$, one can show that $M_h \simeq N_h$. In the case of strong local refinement, e. g., towards a point singularity, the growth is not geometric, but one can show that $M_h \simeq N_h \log N_h$ is possible. In that case the multigrid preconditioner has non-optimal computational complexity.

The major challenge for adaptive multigrid methods is the selection of appropriate basis functions such that: (1) the computational cost per cycle is $O(N_h)$ and (2) the convergence rate of the method is independent of N_h. Historically, the hierarchical basis method [29, 3] was important step in the development of adaptive solvers because it provided a multigrid preconditioner with optimal computational complexity. It is based on the linear system

$$\underline{\hat{A}}\, \underline{\hat{u}} = \underline{\hat{f}}, \qquad \underline{\hat{A}} = \big(a(\phi_k, \phi_i)\big)_{\phi_k, \phi_i \in \hat{\Phi}_J},\ \underline{\hat{f}} = \big(\ell(\phi_i)\big)_{\phi_i \in \hat{\Phi}_J}\ ,$$

assembled with respect to the hierarchical basis $\hat{\Phi}_J$. Although the matrix $\underline{\hat{A}}$ is not sparse, one can show that a Jacobi or Gauss Seidel iteration applied to the system $\underline{\hat{A}}\, \underline{\hat{u}} = \underline{\hat{f}}$ can be implemented with $O(N_h)$ computational cost independent of the locality of refinement. The number of iterations needed to solve the system sufficiently accurate can by bounded by $O(\log N_h)$ in two space dimensions. Thus, the method has optimal computational cost for the single iteration, but suboptimal iteration count. Unfortunately, it turned out that in three space dimensions the number of iterations increases even to $O(N_h^{1/3})$.

4.2 Local smoothing on subspaces

Optimal methods, both with respect to computational complexity and iteration count can be obtained by employing the local nodal basis functions Φ_j related to the subdomains Ω_j. Therefore, these methods are termed multigrid methods with *local smoothing*.

The standard matrix based notation is quite technical for local multigrid methods (see, e. g., [6, 7] for a detailed description). Thus, to present the main ideas we define in the following the algorithms in equivalent form for the corresponding operators. Therefore, let $A_h \colon V_h \to V_h$ be the operator defined by $(A_h v, w)_{L^2(\Omega)} = a(v, w)$ (note that $\underline{A}$ is the corresponding matrix representation with respect to the standard nodal basis Φ_h). Then, a linear solver is defined by a preconditioner $B_h \colon V_h \to V_h$ and the linear iteration

$$u^{k+1} = u^k + B_h(f_h - A_h u^k), \qquad k = 0, 1, 2, ...$$

(here, f_h is the L_2-projection of f onto V_h). The linear iteration is convergent for all initial functions u^0 and all right-hand sides f_h, if and only if the spectral radius of $I - B_h A_h$ is smaller than 1 (where I denotes the identity operator). In general, the linear iteration will be accelerated by a Krylov method, e. g., the conjugate gradient (CG) method for symmetric positive definite problems. A preconditioner B_h is optimal if the condition number of $B_h A_h$ is bounded independently of the number of levels and the number of unknowns.

On every level j, we consider a decomposition $V_j = \sum_{i \in I_j} V_i^j$ into one-dimensional subspaces $V_i^j = \operatorname{span}\{\phi_i^j\}$ spanned by the nodal basis functions $\phi_i^j \in \Phi_j$. Based on this decomposition two types of smoothing can be defined. An additive (or parallel) subspace correction on level j corresponds to local Jacobi smoothing:

(J0) For the actual residual $r = f_h - A_h u^k$ compute independently for every nodal point $i \in I_j$ the one-dimensional correction $c_i^j \in V_i^j$ satisfying

$$a(c_i^j, \phi_i^j) = (r, \phi_i^j)_{L^2(\Omega)} .$$

(J1) Then, sum all corrections $c_j = \sum_{i \in I_j} c_i^j$.

The corresponding multiplicative (or successive) subspace correction is equivalent to local Gauss Seidel smoothing:

(GS) Starting with the residual $r = f_h - A_h u^k$ and $c_j = 0$, compute successively for every nodal point $i \in I_j$ the one-dimensional correction $c_i^j \in V_i^j$ satisfying

$$a(c_i^j, \phi_i^j) = (r, \phi_i^j)_{L^2(\Omega)} - a(c_j, \phi_i^j)$$

and update the correction $c_j := c_j + c_i^j$.

Note that in operator notation no restriction and prolongation is needed. They are introduced naturally, however, when a basis representation of the functions is inserted.

Now, combining the local smoothers with a coarse grid correction results in different preconditioners: additive local multigrid (a weighted Jacobi scheme), multiplicative local multigrid (a Gauß-Seidel scheme), and a hybrid iteration in the parallel case that is multiplicative within a processor and additive between processors.

The additive local multigrid (or parallel subspace correction) preconditioner B_{add} is defined by the following algorithm: for the actual residual $r = f_h - A_h u^k$, compute the correction $c = B_{\text{add}} r$ by

(A0) Compute on level $j = 0$ the coarse grid correction $c_0 \in V_0$ solving

$$a(c_0, v) = (r, v)_{L^2(\Omega)} \quad \text{for all} \quad v \in V_0 \,.$$

(A1) Independently, compute on all levels $j = 1, ..., J$ and for all $i \in I_j$ the one-dimensional correction $c_i^j \in V_i^j$ satisfying

$$a(c_i^j, \phi_i^j) = (r, \phi_i^j)_{L^2(\Omega)} \,.$$

(A2) Finally, collect the additive multigrid correction

$$c = c_0 + \sum_{j=1}^{J} \sum_{i \in I_j} c_i^j \,.$$

The corresponding damped linear iteration is convergent for the model problem, and B_{add} is an optimal preconditioner for Krylov methods. The additive variant is in particular well suited for the parallel realization, since, in the algorithm, the corrections on all levels and all spaces V_i^j can be computed independently.

A faster preconditioner is obtained by the corresponding successive subspace correction method, which is equivalent to the multigrid V-cycle with local Gauss Seidel post-smoothing: the correction $c = B_{\text{mult}} r$ is defined by

(M0) Compute on level $j = 0$ the coarse grid correction $c_0 \in V_0$ solving

$$a(c_0, v) = (r, v)_{L^2(\Omega)} \quad \text{for all} \quad v \in V_0 \,.$$

Set $c = c_0$.

(M1) Successively, compute on all levels $j = 1, ..., J$ and for all $i \in I_j$ the one-dimensional correction $c_i^j \in V_i^j$ satisfying

$$a(c_i^j, \phi_i^j) = (r, \phi_i^j)_{L^2(\Omega)} - a(c, \phi_i^j)$$

and update the correction $c := c + c_i^j$.

The spectral radius of the corresponding linear iteration is bounded independently of the number of levels and the number of unknowns, and the preconditioner is optimal.

Although in general the performance of the multiplicative method is better, it is difficult to realize a Gauss Seidel smoother in parallel, whereas the additive method completely decouples in parallel (if a parallel coarse grid solver is available). A fairly efficient parallel multigrid method is obtained by decomposing the indices $I_j = I_j^1 \cup \cdots \cup I_j^p$ on every level (requiring a separate load balancing on every level j) and defining the parallel correction $c = B_{\text{par}} r$ by

(P0) Compute on level $j = 0$ on one processor the coarse grid correction $c_0 \in V_0$ solving

$$a(c_0, v) = (r, v)_{L^2(\Omega)} \quad \text{for all} \quad v \in V_0 \,.$$

Set $c = c_0$.

(P1) Successively, on all levels $j = 1, ..., J$, set $c^q = 0$ for all processors $q = 1, ..., p$.
Then, compute in parallel for all processors $q = 1, ..., p$ and on every processor successively for all $i \in I_j^q$ the one-dimensional correction $c_i^j \in V_i^j$ satisfying

$$a(c_i^j, \phi_i^j) = (r, \phi_i^j)_{L^2(\Omega)} - a(c_j^q, \phi_i^j) - a(c, \phi_i^j)$$

and update the correction $c^q := c^q + c_i^j$. Then, collect the results on level j from all processors and update

$$c := c + \sum_{q=1}^{p} c^q \ .$$

Several concepts may improve the robustness of the multigrid algorithm, e. g., an extension of the local smoother on I_j to some overlapping region $\tilde{I}_j \subset I_j \cup I_{j-1}$ in combination with multiple smoothing. In the case of hanging nodes, such an approach is analyzed in [9].

Moreover, in the case of systems it is recommended to use a Block Gauss Seidel method where V_i^j combines all unknowns at a single nodal point. For saddle point systems or discontinuous Galerkin approximations, even more involved smoothers such as overlapping Block Gauss Seidel (corresponding to an overlapping subspace correction method) are required.

4.3 A remark on the analysis of local multigrid methods

Analytically, the optimality of local multigrid methods relies on the norm equivalence

$$|\!|\!| v |\!|\!|^2 \simeq \inf_{v=v_0+\cdots+v_J \in V_0+V_1+\cdots+V_J} \left(|\!|\!| v_0 |\!|\!|^2 + \sum_{j=1}^{J} h_j^{-2} \|v_j - v_{j-1}\|_{L^2(\Omega)}^2 \right)$$

(h_j denoting grid size on level j) for the energy norm $|\!|\!| v |\!|\!| = \sqrt{a(v,v)}$. This is proved, e. g., in the case of triangles and regular refinement with hanging nodes in [10]. The norm equivalence implies that the additive preconditioner B_{add} is optimal.

Let P_0 and P_i^j be the Galerkin projections onto the coarse space V_0 and onto the one-dimensional subspaces V_i^j, respectively. Then, the multiplicative local multigrid preconditioner B_{mult} satisfies the norm identity derived by Xu-Zikatanov [27]

$$|\!|\!| \mathrm{id} - B_{\text{mult}} A_h |\!|\!|^2 = 1 - \frac{1}{1+c_0} \,,$$

where

$$c_0 = \sup_{|\!|\!| v |\!|\!|=1} \inf_{v=v_0+\sum_{j=1}^{J}\sum_{i\in I_j} v_i^j} \left(|\!|\!| P_0(v-v_0) |\!|\!|^2 + \sum_{j=1}^{J} \sum_{i\in I_j} \left\| P_i^j \sum_{(k,l)>(i,j)} v_k^l \right\|^2 \right) \,,$$

which is proved, e. g., in the case of triangles and bisection refinement, to be bounded independently of the number of levels and the number of unknowns. Although the analytical results up to now are quite restrictive, this theory provides the optimal framework for a profound theoretical investigation of many relevant adaptive algorithms.

5 Numerical results

We demonstrate the performance of multigrid methods in the context of adaptive local grid refinement for the model problem (1). In both, the two- and three-dimensional test cases, the diffusion coefficient was set to $K(x) = I$. Adaptive refinement was controlled by a residual based error estimator. In each adaptive step, a certain percentage of the elements with the largest contribution to the error were refined.

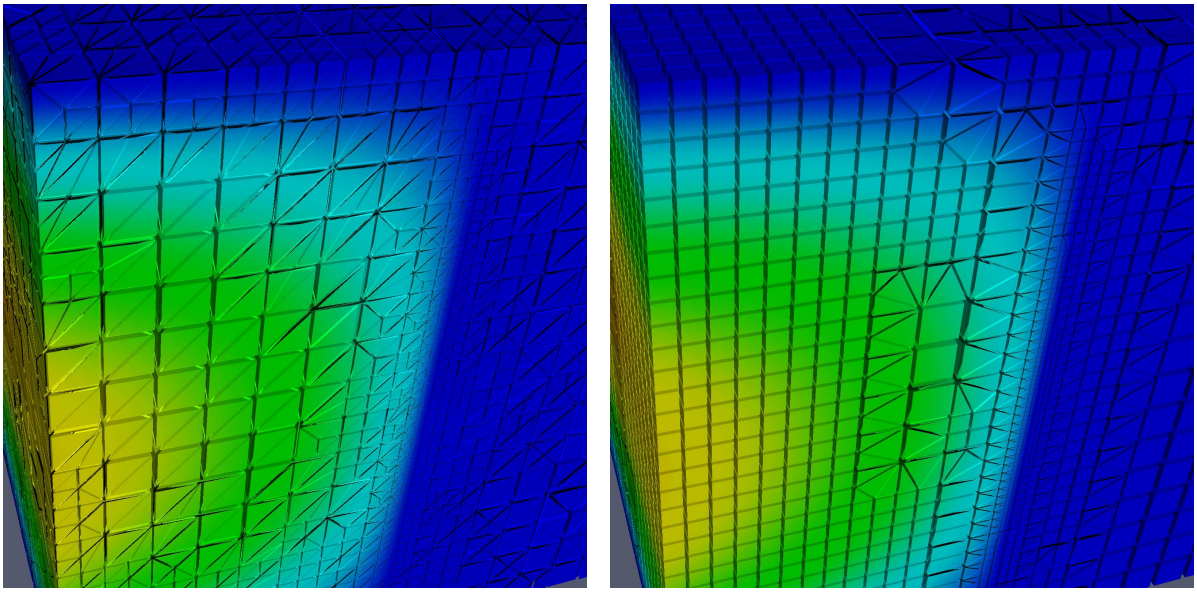

Figure 4: Adaptive solution of the elliptic model problem in three space dimensions with P_1 conforming finite elements and residual based error estimator for an example with singularity on an edge. Left: tetrahedral grid with red/green refinement. Right: hexahedral grid with red/green refinement using pyramids and tetrahedra for closure. The computations were done with the PDE software Dune/UG, visualization with ParaView/VTK.

In the 2d example, the domain was the unit square $\Omega = (0,1)^2$ and the solution exhibited a point singularity at $(0, 1/2)$, see Figure 1. In the 3d test case the domain was the unit cube $\Omega = (0,1)^3$ and the solution exhibited a line singularity along $(1/2, 0, z)$. Two example grids are shown in Figure 4.

Table 1 compares several solvers for the elliptic model problem in two and three space dimensions on simplicial and cube grids. The following solvers were used:

MGC Multiplicative local multigrid with 2 symmetric Gauss Seidel post-smoothing steps. This is the successive subspace correction method labeled M0-M1 on page 9.

BICGMGC The same local multigrid method used as a preconditioner in the BiCGSTAB [25] method.

CGBPX Additive multigrid with one Jacobi smoothing step used as preconditioner in the conjugate gradient (CG) method. This is the additive local multigrid preconditioner labelled A0-A2 on page 8.

	MGC	BICGMGC	CGBPX	CGAMG	CGILU0
2d, triangular elements, $N = 1698410$, $E = 3393283$					
IT	9	7	29	23	2178
T_{solve}	8.9	7.4	16.0	33.5	637.7
T_{setup}	76.0	76.0	76.0	17.0	0.6
2d, quadrilateral elements, $N = 1412468$, $E = 1412854$					
IT	7	6	18	18	1183
T_{solve}	7.1	6.4	9.6	27.3	334.0
T_{setup}	50.0	50.0	50.0	17.1	0.7
3d, tetrahedral elements, $N = 287092$, $E = 1693032$					
IT	22	13	49	15	156
T_{solve}	7.3	4.5	6.6	8.5	14.6
T_{setup}	68.3	68.3	68.3	7.0	0.5
3d, hexahedral elements, $N = 622370$, $E = 941302$					
IT	11	8	29	9	85
T_{solve}	10.3	7.7	9.8	14.4	21.2
T_{setup}	77.5	77.5	77.5	25.2	2.0

Table 1: Comparison of different linear solvers for 2d and 3d adaptively refined solution of the model problem (1).

CGAMG Agglomeration type algebraic multigrid with 2 symmetric Gauss Seidel pre- and post-smoothing steps used as preconditioner in the CG method. This method is similar to the method introduced in [11]. It has been included here to show that algebraic multigrid methods are also very competitive for linear systems arising from adaptive local refinement. For more details on algebraic multigrid see also the article in this issue.

CGILU0 Incomplete LU decomposition with no additional fill-in as a preconditioner in the CG method. This method has been included to give a comparison with a standard single grid preconditioner.

The table gives iteration numbers (IT) for a reduction of the Euclidean norm of the residual by the factor 10^{-8} in the last solution step of the adaptive procedure. The size of the finest leaf grid obtained in each test case is given by the number of nodes N and the number of elements E. Since the cost per iteration is different for the methods given we also show the time spent for all iterations as T_{solve}. All times are given in seconds and have been measured on a Laptop-PC with an Intel T2500 Core Duo processor with 2.0 GHz, 667 MHz FSB and 2 MB L2 cache using the GNU C++ compiler in version 4.0 and -O3 optimization.

The table also includes the setup time T_{setup} necessary for each preconditioner. In the implementation used for the tests the matrix $\underline{A}$ as well as the level-wise matrices $\underline{A}_j$ and the grid transfer operators used in the multigrid preconditioners are assembled as sparse matrices in compressed row storage format. The multigrid preconditioner can then be written completely in terms of (fast) matrix-vector and vector operations. The setup time for the local multigrid preconditioners is the time needed for assembling the extra matrices $\underline{A}_j$ and the grid transfer operators. For the algebraic multigrid preconditioner it is the construction of the coarse grid operators and for the ILU method it is the computation of the incomplete decomposition. The

setup time for the local multigrid methods seems to be rather high compared to the algebraic multigrid method. This is due to the generality of the assembly procedure which works in any dimension, for any kind of grid (including e. g. refinement with hanging nodes) and which assumes a general, position dependent diffusion tensor. On the other hand this is a quite realistic situation with respect to real world applications. There often the assembly of the finite element stiffness matrix is the most costly part of the simulation.

From Table 1, we conclude that local multigrid used as a preconditioner in BiCGSTAB has always the lowest iteration count and the minimum solution time T_{solve}. The additive multigrid method is between a factor 1.3 and 2.2 slower than the multiplicative method. The iteration numbers for the incomplete decomposition preconditioner increase with $O(h^{-1})$ and thus the single grid method is always slower than the multigrid methods provided the grid is fine enough. This point has been clearly reached in the two-dimensional examples while, in three space dimensions, the single grid method is the winner in total time to solution $T_{\text{solve}} + T_{\text{setup}}$. The algebraic multigrid method is very competitive. It is the fastest method in time to solution in 2d and it is very close to the single grid method even in 3d.

2d, quadrilateral elements						
N	J	MGC	BICGMGC	CGBPX	CGAMG	CGILU0
2092	9	8	6	18	9	81
6094	11	8	6	18	11	140
17751	13	8	6	19	12	228
55297	14	8	6	19	14	303
162963	16	7	6	18	16	585
500045	17	7	6	18	18	747
1412468	19	7	6	18	18	1183
3d, hexahedral elements						
N	J	MGC	BICGMGC	CGBPX	CGAMG	CGILU0
689	3	8	6	16	4	10
4540	5	9	7	24	6	20
26903	7	10	7	28	7	35
129738	9	11	8	30	8	55
622370	11	11	8	29	9	85

Table 2: Demonstration of grid independence of the convergence rate for the adaptively refined solution of the model problem (1) in two space dimensions.

In Table 2, we demonstrate that the iteration numbers are independent of the grid size for the various multigrid algorithms discussed in this paper. The iteration numbers for the AMG preconditioner and the ILU preconditioned conjugate gradient method are also given for comparison.

Although we restricted the investigation of adaptive multigrid methods to the model problem, this is a basic technique which can be applied to solve the linearized problem within many nonlinear and time dependent applications on locally adapted grids (e. g., the authors have experiences with two-phase flow in porous media and plasticity). Since nonlinear applications are not the main topic of this contribution, we cannot give a representative overview of adaptive multigrid applications.

6 Future challenges

Multigrid methods on locally refined hierarchical grids are now a standard tool for the efficient numerical solution of a wide range of partial differential equations. The construction process of suitably adapted grids and the solution method on these grids are completely independent. The decoupling of error control and adaptive multigrid solver relies on the paradigm that the problem can be solved with simple discretizations on sufficiently fine locally adapted grids, where the grid resolves the specific features of the application. Thus, the design of these multigrid methods solely relies on geometric quantities.

In modern applications with highly nonlinear problems in 3d it becomes clear that simple discretization concepts are not sufficient for an overall effective solution process. It is necessary to employ optimal discretizations such as higher order finite element methods, discontinuous Galerkin methods, anisotropic grids or higher order upwinding together with adaptivity with respect to grid size and polynomial degree (hp-adaptivity). Then, the design of the solver cannot rely only on geometric quantities, but algebraic properties of the resulting system matrices must be taken into account. The grand challenge in the construction and the analysis of adaptive multigrid methods is the development of robust subspace correction techniques for a broad class of optimal hp-adaptive discretizations and its application to demanding problem classes.

Acknowledgments

The numerical results presented in this paper have been obtained with the PDE framework Dune [8], which uses generic programming techniques to provide an efficient, uniform and dimension-independent access to various finite element software packages such as UG [7] and Alberta [24]. The use of all this software is greatly appreciated and we thank all contributors, in particular, M. Blatt for providing his implementation of an agglomeration based algebraic multigrid method. We also thank the anonymous referees for many helpful comments.

References

[1] M. Ainsworth and J. T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. Wiley, 2000.

[2] D. Bai and A. Brandt. Local mesh refinement multilevel techniques. *SIAM J. Sci. Stat. Comp.*, 8:109–134, 1987.

[3] R. Bank, T. F. Dupont, and H. Yserentant. The hierarchical basis multigrid method. *Numer. Math.*, 52:427–458, 1988.

[4] R. E. Bank, A. H. Sherman, and A. Weiser. Refinement algorithms and data structures for regular local mesh refinement. In *Scientific Computing*, IMACS. North-Holland, Amsterdam, 1983.

[5] P. Bastian. Locally refined solution of unsymmetric and nonlinear problems. In *Proceedings of the 8^{th} GAMM Seminar*, volume 46 of *Notes on Numerical Fluid Mechanics*, pages 12–21. Vieweg, 1993.

[6] P. Bastian. *Parallele adaptive Mehrgitterverfahren*. Teubner Skripten zur Numerik. Teubner-Verlag, 1996.

[7] P. Bastian, K. Birken, S. Lang, K. Johannsen, N. Neuß, H. Rentz-Reichert, and C. Wieners. UG: A flexible software toolbox for solving partial differential equations. *Computing and Visualization in Science*, 1:27–40, 1997.

[8] P. Bastian, M. Droske, C. Engwer, R. Klöfkorn, T. Neubauer, M. Ohlberger, and M. Rumpf. Towards a unified framework for scientific computing. In R. Kornhuber, R.H.W. Hoppe, D.E. Keyes, J. Périaux, O. Pironneau, and J. Xu, editors, *Proceedings of the 15th Conference on Domain Decomposition Methods*, number 40 in LNCSE, pages 167–174. Springer-Verlag, 2004.

[9] R. Becker and M. Braack. Multigrid techniques for finite elements on locally refined meshes. *Numerical Linear Algebra with Applications*, 7(6):363–379, 2000.

[10] F. Bornemann and H. Yserentant. A basic norm equivalence for the theory of multilevel methods. *Numer. Math.*, 64:455–476, 1993.

[11] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55:379–393, 1995.

[12] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comput.*, pages 1–22, 1990.

[13] A. Brandt. Multi-level adaptive solutions to boundary-value problem. *Math. Comput.*, 31:333–390, 1977.

[14] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. *Computational Differential Equations*. Cambridge University Press, 1996.

[15] M. Griebel. Multilevel algorithms considered as iterative methods on semidefinite systems. *SIAM Sci. Comp.*, 15(3):547–576, 1994.

[16] S. Lang. *Parallele Numerische Simulation instationärer Probleme mit adaptiven Methoden auf unstrukturierten Gittern*. PhD thesis, Universität Stuttgart, 2001.

[17] S. McCormick. The fast adaptive composite grid (FAC) methods: theory for the variational case. In K. Böhmer and H. J. Stetter, editors, *Defect Correction Methods: Theory and Applications*, volume 5 of *ComputationsSupplementation*, pages 115–122, 1984.

[18] S. McCormick and D. Quinlan. Asynchronous multilevel adaptive methods for solving partial differential equations. *Parallel Comput.*, 12:145–156, 1989.

[19] S. McCormick and J. Thomas. The fast adaptive composite grid (FAC) method for elliptic equations. *Math. Comput.*, 1986.

[20] W. F. Mitchell. *Unified multilevel adaptive finite element methods for elliptic problems*. PhD thesis, University of Illinois at Urbana-Champaign, 1988.

[21] W. F. Mitchell. The full domain partition approach for parallel multigrid on adaptive grids. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing*, 1997.

[22] M. C. Rivara. Design and data structure of a fully adaptive multigrid finite element software. *ACM Trans. on Math. Software*, 10:242–264, 1984.

[23] U. Rüde. *Mathematical and computational techniques for multilevel adaptive methods*, volume 13 of *Frontiers in Applied Mathematics*. SIAM, 1993.

[24] K. Siebert and A. Schmidt. *Design of adaptive finite element software: The finite element toolbox ALBERTA*. Springer, 2005.

[25] H. A. Van der Vorst. BiCGSTAB: A fast and smoothly converging variant of Bi–CG for the solution of non–symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.

[26] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34:581–613, 1992.

[27] J. Xu and L Zikatanov. The method of alternating projections and the method of subspace corrections in Hilbert space. *J. Amer. Math Soc.*, 15:573–597, 2002.

[28] I. Yavneh. Why multigrid methods are so efficient. *CiSE*, 2006.

[29] H. Yserentant. On the multi-level splitting of finite element spaces. *Numer. Math.*, 1986.