SIAM J. SCI. COMPUT.
Vol. 33, No. 4, pp. 2095–2114

ADAPTIVE MULTILEVEL METHODS WITH LOCAL SMOOTHING FOR H^1- AND H^{curl}-CONFORMING HIGH ORDER FINITE ELEMENT METHODS*

BÄRBEL JANSSEN† AND GUIDO KANSCHAT‡

Abstract. A multilevel method on adaptive meshes with hanging nodes is presented, and the additional matrices appearing in the implementation are derived. Smoothers of overlapping Schwarz type are discussed; smoothing is restricted to the interior of the subdomains refined to the current level; thus it has optimal computational complexity. When applied to conforming finite element discretizations of elliptic problems and Maxwell equations, the method's convergence rates are very close to those for the nonadaptive version. Furthermore, the smoothers remain efficient for high order finite elements. We discuss the implementation in a general finite element code using the example of the deal.II library.

Key words. multigrid methods, adaptive mesh refinement, finite element method, hanging nodes, Maxwell equations

AMS subject classifications. 65N55, 65N22, 65N30, 65F08

DOI. 10.1137/090778523

1. Introduction.

In this article, we derive a consistent multilevel method for discretization on meshes with hanging nodes. The implementation for H^1- and H^{curl}-conforming finite elements of arbitrary order is discussed. Our design goals for such a method are the following:

1. Convergence rates may not be significantly worse than on regular meshes without local refinement.
2. Each step should be performed with optimal computational complexity.
3. The matrix structures involved must be easy to obtain in a finite element code and may not severely increase memory requirements.
4. Smoothing should happen only on subgrids without hanging nodes to simplify cell- or patch-based and hybrid smoothers.
5. The scheme should be able to use the continuity conditions across faces for any finite element, for instance, H^1- or H^{curl}-conformity.
6. The smoother should be efficient for high order finite elements.

With respect to item 1, we note that, qualitatively, it is clear from work by Bramble [13], Griebel and Oswald [22], and Xu [44] that local smoothing, if correctly implemented, yields convergence rates independent of the mesh size. Nevertheless, it is a priori not clear whether the constants in these estimates deteriorate in the presence of refinement edges. Furthermore, the estimates are usually not uniform in

*Submitted to the journal's Methods and Algorithms for Scientific Computing section November 30, 2009; accepted for publication (in revised form) June 2, 2011; published electronically August 25, 2011.

http://www.siam.org/journals/sisc/33-4/77852.html

†Institut für Angewandte Mathematik, Universität Heidelberg, Im Neuenheimer Feld 293/294, 69120 Heidelberg, Germany (baerbel.janssen@iwr.uni-heidelberg.de). This author's work was supported by the German Research Association (DFG) and the International Graduate College IGK 710.

‡Department of Mathematics, Texas A&M University, 3369 TAMU, College Station, TX 77843 (kanschat@tamu.edu). This author's work was supported by the National Science Foundation under grants DMS–0713829 and DMS–0810387 and by the King Abdullah University of Science and Technology (KAUST) through award KUS-C1-016-04. Some of the experiments were performed during a visit to the Institute of Mathematics and its Applications in Minneapolis.

the polynomial degree of the finite element shape functions. In section 3, we provide numerical evidence that these rates are quantitatively not worse than those obtained on uniform meshes and depend only weakly on the polynomial degree.

The main focus of this article lies in demonstrating that such a method can be conveniently implemented within the framework of a general purpose finite element code. Item 3 ensures that neither coding effort nor memory requirements at runtime increase significantly due to local smoothing. By item 4, we guarantee that smoothing schemes which have been developed without local refinement in mind can be easily applied within our framework. Finally, if the method is to work with higher order elements or more complicated vector valued elements, then it becomes essential, from the point of view of code development, that only the continuity information determined by the node functionals of the finite element is used as mandated in item 5. In order to achieve these goals, we follow the route laid out in [30] for discontinuous Galerkin methods and describe the necessary modifications for conforming methods in this article.

When we consider adaptively refined meshes for multigrid methods, we have to distinguish between meshes with hanging nodes and conforming meshes. The latter can be generated by either red-green refinement [34] or bisection [32, 38], and often appear with simplicial meshes. In that case, the question of dealing with hanging nodes does not arise. Local adaptive multigrid methods for these meshes can be found, for instance, in [6, 7, 8, 35].

On meshes with hanging nodes, several types of optimal multilevel preconditioners have been devised:

1. Local smoothing *inside* the region of local refinement, but not at the interface between refined and coarser regions, was introduced by Brandt [15] for finite difference methods. In McCormick's monograph [31], this method is discussed for the finite volume element method under the name of "multilevel composite grid scheme." These methods are used in applications that allow for dynamically changing meshes in the multigrid procedure [41]. A local multigrid method with smoothing on the same subdomain for curl-elliptic problems with lowest order Nédélec elements was developed in [25]. In this article, we extend these methods to higher order finite element methods in the H^1- and H^{curl}-conforming case.
2. Local smoothing in the region of local refinement, including the interface between fine and coarse cells, and the support of basis functions associated with node values on the interface were introduced in [10]. In [31], this scheme is applied to finite volume methods as the "bordered multilevel scheme." While this scheme performs very well with standard smoothers, it has drawbacks from the computational point of view: The "level matrices" used for local smoothing extend over two levels. Therefore, they cannot easily be decomposed into a fine and a coarse part, and avoiding memory overhead comes at the price of complicated data structures. In addition, the presence of hanging nodes makes the implementation and analysis of hybrid smoothers for Maxwell equations like those of Hiptmair [24] and Arnold, Falk, and Winther [1] more complicated.
3. The method of global coarsening avoids the problem of setting a border to the subdomain for smoothing by introducing a hierarchy of level spaces where each space covers the whole computational domain. By choosing the levels carefully, these methods still maintain optimal complexity. They share the drawbacks of the bordered scheme but offer advantages in problems with

global constraints like the zero mean value property of the pressure in incompressible flow problems. This method has been applied, for instance, in [11, 12, 27, 28, 37]. In [10], it is compared to the bordered scheme. Global smoothing does not yield optimal complexity [8] of the multilevel algorithm, but in this case no artificial boundary has to be created.

In a finite element context, we use purely algebraic elimination for hanging nodes, thus simplifying the setup of the system. As a consequence, our approach applies to any conforming finite element method and to any polynomial order of the shape functions. Furthermore, there are no fine grid nodes on the "refinement edge," the border between a refined and a coarser part of the mesh.

Multilevel methods on adaptively refined meshes using finite elements and their analysis go back to [15]. Numerical tests and analysis of multilevel methods applied to higher order discretizations on uniform meshes are presented in [26]. The convergence theory for the method we consider is laid out in [13, 14] in a classical multigrid context. In [22, 44] an abstract convergence theory of the additive and multiplicative multilevel Schwarz methods is proposed, which applies to the analysis of the block smoother applied in this article. The theory of the smoother for Maxwell problems is laid out in [1], at least for regular refinement. Nevertheless, no quantitative considerations on the effect of local smoothing on the convergence rate are discussed in any of these works.

This article is organized as follows: In the following section, we review the V-cycle algorithm and derive its description when applied to local smoothing on adaptive meshes, leading to Algorithm 2. The splitting of the spaces and operators in subsection 2.3 implies that our goals 3–5 have been achieved. Furthermore, in subsections 2.7 and 2.8 we discuss that runtime and memory requirements of one step of this algorithm are not increased substantially, verifying goal 2. In section 3, we confirm the efficiency of the resulting method with numerical experiments. The key result of this paper is that computations in two and three dimensions for finite elements up to order nine indicate that the convergence speed is independent of the introduction of refinement edges, and that dependence on the polynomial degree is weak.

2. The algorithm. First let us introduce some notation: We are concerned with approximating solutions to standard second order elliptic and eddy current boundary value problems on the bounded open domain $\Omega \subset \mathbb{R}^d$ with $d = 2, 3$ subject to suitable boundary conditions by a conforming finite element method.

In weak formulation, such a problem reads as follows: Find $u \in V$ such that for all $v \in V$ it holds that

$$a(u, v) = (f, v). \tag{2.1}$$

Here, in the elliptic case, V denotes the subspace of $H^1(\Omega)$ with suitable boundary conditions. In the Maxwell case V is the subspace of $H^{\mathrm{curl}}(\Omega)$ with suitable conditions on the tangential traces at the boundary. The bilinear forms we consider for these two cases are

$$\begin{aligned} a_\Delta(u, v) &= (\nabla u, \nabla v), \\ a_{\mathrm{curl}}(u, v) &= (\nabla \times u, \nabla \times v) - \sigma(u, v), \end{aligned} \tag{2.2}$$

respectively. By $(\cdot, \cdot)$ we denote the standard inner product of $L^2(\Omega; \mathbb{R}^{d'})$ with $d' = 1$ and $d' = d$, respectively. If $d = 2$, the operator $\nabla \times u$ denotes the standard scalar curl of the two-dimensional vector field u, namely, the third component of the three-dimensional curl applied to the extended field $(u_1, u_2, 0)$. For the eddy current prob-

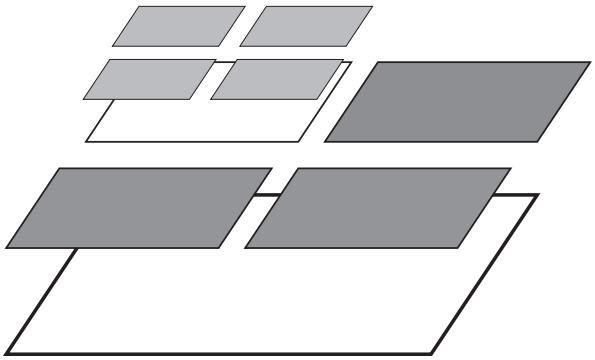

FIG. 2.1. *A hierarchy of three meshes with local refinement (active cells shaded).*

lem, the constant σ is chosen less than the smallest nonzero eigenvalue of the Maxwell operator.

We approximate solutions to this problem by a conforming finite element method, for instance, conforming finite elements with tensor product shape functions of degree k (referred to as $\mathbb{Q}_k$) or Nédélec elements $\mathbb{N}_k$ of order k. To this end, we introduce a hierarchy of quadrilateral/hexahedral meshes $\{\mathbb{T}_\ell\}$, $0 \le \ell \le L$, obtained from a coarse mesh $\mathbb{T}_0$ by consecutive, possibly local, refinement. Let us point out here that, while the process of generating such a hierarchy may involve refinement and coarsening of cells, the final hierarchy can be understood as a quadtree/octree graph oriented only from coarse to fine cells. See Figure 2.1 for an example of such a hierarchy.

In order to compute the approximation of u_ℓ on the mesh $\mathbb{T}_\ell$, we introduce a basis of finite element functions on $\mathbb{T}_\ell$ to generate a "discrete" linear system (see, e.g., [16, 18]). Restricting the bilinear form $a(\cdot,\cdot)$ and the linear form in (2.1) to the finite element spaces V_ℓ of dimension n_ℓ, we obtain linear systems denoted as

$$A_\ell u_\ell = f_\ell. \tag{2.3}$$

Here, u_ℓ and f_ℓ are vectors in $\mathbb{R}^{n_\ell}$ and A_ℓ is a quadratic matrix of dimension n_ℓ. We will refer to the system on the finest level L as the "global" system in order to distinguish it from the level problems introduced later.

In order to solve this system, typically a Krylov space solver is employed. In a Krylov space method, new iterates are formed from residual r^k of the current iteration step k. While these methods use orthogonalization techniques to obtain minimization properties of the next iterate and speed up performance (see, e.g., [36]), they slow down on fine meshes. To understand this, it is sufficient to consider the Richardson iteration

$$u_\ell^{k+1} = u_\ell^k + \omega r_\ell^k \tag{2.4}$$

as a prototype, denoting that the operation (2.4) is also a part of the conjugate gradient (cg) method. Here, $r_\ell^k = A_\ell e_\ell^k = f_\ell - A_\ell u_\ell^k$ is the residual of step k and $e_\ell^k = u_\ell - u_\ell^k$ is the error between the true and the iterative solution in step k. While the addition in (2.4) looks straightforward from the point of view of linear algebra, it is all but this in the function space setting of elliptic problems. There, the function u^k is in the space $H^1(\Omega)$, while the residual is in the space $H^{-1}(\Omega)$ of bounded linear forms on $H^1(\Omega)$ (subject to boundary conditions). The norms of both spaces have

very different scaling behavior if applied to oscillating functions, for instance, standard finite element basis functions. In a nutshell, this is the reason why iterative solvers do not have uniform convergence properties with respect to the refinement level ℓ and slow down considerably on finer meshes. This problem can be overcome by choosing a basis with better conditioning, for instance, a hierarchical basis [45] or a wavelet basis [19, 20], or by multigrid methods.

Remark 1. For convenience, we remark that a discrete element of the dual space is understood as a vector in the sense $l_i = (l, v_i)$ for all $v_i \in V_h$.

In order to speed up convergence, a preconditioner is introduced, which maps the residual back into a function, transforming, for instance, the Richardson iteration (2.4) into its preconditioned form

$$u_\ell^{k+1} = u_\ell^k + \omega P_\ell^{-1} r_\ell^k. \tag{2.5}$$

For the purpose of this work, P is the multigrid preconditioner studied extensively in the literature (see, e.g., [13, 23, 44]) and discussed here on locally refined meshes. Given smoothers $S_\ell^{(i)}$ and embedding operators $R_\ell^T : V_\ell \to V_{\ell+1}$, the action of P_ℓ^{-1} on a residual vector d_ℓ can be recursively denoted by the following algorithm.

ALGORITHM 1 (V-Cycle). Let $P_0 = A_0$. Set $x^{(0)} = 0$ and compute $P_\ell^{-1} d_\ell$ by the following steps:

1. (Presmoothing) Compute $x^{(m_\ell)}$ iteratively by
$$x^{(i)} = x^{(i-1)} + S_\ell^{(i)}(d_\ell - A_\ell x^{(i-1)}), \quad i = 1, \dots, m_\ell.$$
2. (Coarse grid correction) Let
$$y^{(0)} = x^{(m_\ell)} + R_{\ell-1}^T P_{\ell-1}^{-1} R_{\ell-1}(d_\ell - A_\ell x^{(m_\ell)}).$$
3. (Postsmoothing) Compute $y^{(m_\ell)}$ iteratively by
$$y^{(i)} = y^{(i-1)} + S_\ell^{(m_\ell+i)}(d_\ell - A_\ell y^{(i-1)}), \quad i = 1, \dots, m_\ell.$$
4. Set $P_\ell^{-1} d_\ell = y^{(m_\ell)}$.

We consider two variants of the V-cycle:

classical V-cycle. m_ℓ is fixed to the same number independent of the level ℓ; for isotropic, elliptic problems, typically one or two steps are enough.

variable V-cycle. The numbers m_ℓ grow geometrically when the level ℓ decreases. Namely, for $\ell = 1, \dots, L$ there exist $1 < \beta_0 \le \beta_1$ such that

$$\beta_0 m_\ell \le m_{\ell-1} \le \beta_1 m_\ell. \tag{2.6}$$

A typical choice is $\beta_0 = \beta_1 = 2$, which results in doubling the number of smoothing steps when reducing the level and leads to a complexity comparable to the W-cycle. This method is known to yield a uniform preconditioner for any number of smoothing steps on the finest level [13].

In the following subsections, we will first recast Algorithm 1 in the context of adaptively refined meshes and local smoothing and then fix a smoother for actual computations.

2.1. Hanging nodes. We treat refinement edges by introducing "hanging nodes" on the refined side, not by introducing additional refinement on the coarse side. These hanging nodes correspond to nominal degrees of freedom in the discretization but are

constrained through the requirement that finite element functions must be conforming across the refinement edge. One option for dealing with these degrees of freedom would be eliminating them completely from the linear system. We do not use this technique, since it involves a numbering of degrees of freedom, which is not easily represented in the mesh anymore. Instead, we follow [2, 3, 29] in keeping the degrees of freedom in the linear system.

If hanging nodes are used, additional equations are needed to deal with them. These equations are obtained from the continuity condition of the finite element space. This condition essentially states that along the refinement edge, the trace of a function on the refined part of the mesh must be equal to the one on the coarse part. For H^{curl}, this must be the trace of the tangential components, while for H^1, it is just the trace of the function. In general, it is the trace operator that ensures conformity of the finite element space.

Thus, we obtain a small linear system of equations, which allows us to eliminate some of the degrees of freedom on the refinement edge [2]. After doing so, a convention has to be found for representing a function by a coefficient vector. In the implementation in deal.II, two forms are used:

"*condensed*." All degrees of freedom corresponding to hanging nodes are always zero. This is the natural representation for linear forms, since there is no basis function in the finite element space.

"*distributed*." The coefficients on the refined side are set such that the functions on both sides coincide. This is the natural representation for finite element functions, since they are conforming.

As an example, for shape functions linear on edges in two dimensions, the distributed form assigns the mean value of the two neighbors to the hanging node. In order to convert from the distributed form to the condensed form, half of the value in the hanging node is added to its neighbors and its value itself is set to zero. The condensed form is used for the vectors in a linear solver, so as not to spoil the residual by artificial values. Then, whenever multiplication with the system matrix, which was built cellwise and does not know about the hanging node, is needed, a conversion to distributed and back is needed.

We point out that the concept of hanging nodes is not restricted to linear and bilinear finite elements. Details of its application to higher order elements can be found in [2], and its application to local *hp*-refinement in [4]. It has been implemented in the deal.II library [3].

Accordingly, the residual is in condensed form and the vector must be returned in the same representation. With this information, we can start rewriting Algorithm 1 for locally refined meshes. We will first have to determine, what we consider a "level" in such a case and then determine the subspaces for smoothing.

2.2. Splitting of adaptive meshes into levels. The meshes considered in this article are obtained from a quasi-uniform, conforming coarse mesh $\mathbb{T}_0$ by consecutively refining mesh cells. In order to achieve high resolution in regions where required, refinement may be restricted to these areas of the domain; examples for controlling this refinement can be found in the rich literature on adaptive mesh refinement (see, e.g., [5, 17, 43]).

We introduce the notion of an *active cell* of the hierarchy $\{\mathbb{T}_\ell\}$ (see, e.g., [2]). These are the cells which are not refined further in any of the triangulations $\mathbb{T}_\ell$. The shape functions of all active cells constitute the space V_L on the finest level. We can also consider the hierarchy of meshes as a tree graph where the grid cells correspond

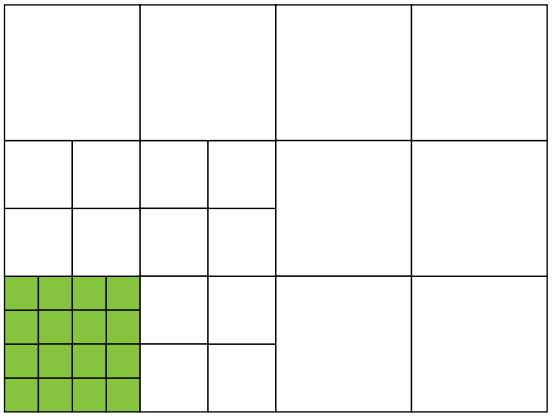

FIG. 2.2. *Splitting of* $\mathbb{T}_\ell$ *into* $\mathbb{T}_\ell^S$ *(shaded cells) and* $\mathbb{T}_\ell^L$ *(white cells).*

to nodes of the graph and "refinement" corresponds to edges. Then, the cells of $\mathbb{T}_0$ correspond to the roots of the tree (or rather forest), and active cells are the leaves (shaded in Figure 2.1).

The *level* ℓ_T of the cell T in the triangulation hierarchy $\{\mathbb{T}_\ell\}$ is defined recursively as follows: If a cell belongs to the coarse mesh $\mathbb{T}_0$, its level is zero. Otherwise, it is obtained by refinement of another grid cell T_p, and we set $\ell_T = \ell_{T_p} + 1$. Since there is no notion of coarsening in a single hierarchy $\{\mathbb{T}_\ell\}$, this level is uniquely defined. The level ℓ_F of a face is defined to be the highest level of the adjacent mesh cells.

We remark that a triangulation $\mathbb{T}_\ell$ does not consist of cells on level ℓ only, but covers the whole domain Ω. Therefore, a single grid cell T with level ℓ_T can belong to several meshes $\mathbb{T}_\ell, \mathbb{T}_{\ell+1}, \ldots$, as shown in Figure 2.1. For instance, the shaded cells on the intermediate level (cell level 1) belong to the meshes $\mathbb{T}_1$ and $\mathbb{T}_2$, and the white cell on that level belongs to $\mathbb{T}_1$ only.

Each triangulation $\mathbb{T}_\ell$ will be partitioned into the set of cells strictly on level ℓ,

$$\mathbb{T}_\ell^S = \big\{T \in \mathbb{T}_\ell \big| \ell_T = \ell\big\},$$

and the set of cells on lower levels,

$$\mathbb{T}_\ell^L = \big\{T \in \mathbb{T}_\ell \big| \ell_T < \ell\big\}.$$

This partitioning is explained in Figure 2.2, where $\mathbb{T}_2^S$ is shaded and cells in $\mathbb{T}_2^L$ are white. We remark that cells in $\mathbb{T}_\ell^L$ may contain grid cells of several lower levels. Obviously, there holds

$$\mathbb{T}_\ell^S \cup \mathbb{T}_\ell^L = \mathbb{T}_\ell, \qquad \mathbb{T}_\ell^S \cap \mathbb{T}_\ell^L = \emptyset.$$

By $\mathbb{F}_\ell$, we denote the set of all faces of cells in $\mathbb{T}_\ell$. In particular, the set of interior faces $\mathbb{F}_\ell^i$ is the set of faces $F_{ij} = \overline{T}_i \cap \overline{T}_j$, where T_i and T_j are two cells of $\mathbb{T}_\ell$. We call the faces between the sets $\mathbb{T}_\ell^S$ and $\mathbb{T}_\ell^L$ the refinement edge $\mathbb{F}_l^E$ between levels ℓ and $\ell - 1$, that is,

$$\mathbb{F}_\ell^E = \left\{ F \in \mathbb{F}_\ell^i \Big| F \cap \bigcup_{\mathbb{T}_\ell^S} \overline{T} = F \ \wedge \ F \cap \bigcup_{\mathbb{T}_\ell^L} \overline{T} = F \right\}.$$

Mostly for technical reasons, we limit the jump of cell levels ℓ_T across the refinement edge. To this end, we introduce the following two notions.

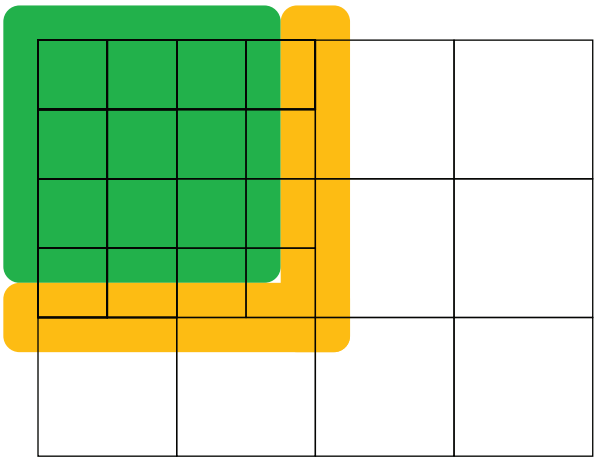

FIG. 2.3. *The subspaces V_ℓ^S (dark) and V_ℓ^L (white). The space V_ℓ^E is associated with the coarse level degrees of freedom in the medium colored region. The space $\tilde{V}_\ell$, consists of V_ℓ^S and basis functions associated with fine level degrees of freedom in the medium colored region.*

DEFINITION 2.1. *A mesh is* one-irregular *if the levels of all active cells sharing a face differ by a maximum of one.*

DEFINITION 2.2. *A mesh is* v-one-irregular *if the levels of all active cells sharing a vertex or a face differ by a maximum of one.*

The notion of one-irregular meshes is ubiquitous in the literature on adaptive refinement with hanging nodes. The additional requirement of being v-one-irregular acts only at corners of the refinement edge. While one-irregularity was the only condition on the mesh for discontinuous Galerkin methods [30], it is not sufficient if there are degrees of freedom located on vertices. In this case, we require the mesh to be v-one-irregular. We point out that, while these conditions are not really necessary for the analysis or for the implementation (see, e.g., [40]), they are convenient, because they ensure that refinement edges on different levels are separated. In our experience, they are not harmful, since (a) they will not cause global spread of refinement and (b) they are consistent with a uniform approximation quality in most cases.

Following [13, 31], we perform the multilevel method on the complete level spaces V_ℓ, but we restrict the smoother to the part that is really refined to level ℓ, namely, the subspace of functions with support covered by $\mathbb{T}_\ell^S$.

2.3. Splitting of level spaces. Partitioning of the spaces V_ℓ into subspaces follows the splitting of $\mathbb{T}_\ell$:

$$V_\ell = V_\ell^S \oplus V_\ell^E \oplus V_\ell^L, \tag{2.7}$$

where V_ℓ^S and V_ℓ^L are the functions in V_ℓ with support in $\mathbb{T}_\ell^S$ and $\mathbb{T}_\ell^L$, respectively. V_ℓ^E is the remainder of V_ℓ. It consists of the functions with support in $\mathbb{T}_\ell^S$ **and** $\mathbb{T}_\ell^L$ and is spanned by the coarse level basis functions corresponding to node functionals on the refinement edge (see Figure 2.3). Note that in the case of discontinuous Galerkin methods, V_ℓ^E is empty. While these spaces are appropriate for the mathematical formulation of the method, neither of them appears in the implementation. Instead, in order to keep data structures easily manageable, we deal with the level spaces $\tilde{V}_\ell$

spanned by the functions in V_ℓ^S and the fine level basis functions on the refinement edge $\mathring{V}_\ell$. Since $\mathring{V}_\ell$ is not part of the algorithm, components of vectors in this subspace of $\tilde{V}_\ell$ either have to be forced to zero or are ignored altogether. This is the main complication compared to the algorithm for discontinuous Galerkin methods in [30].

A basis for the other subspaces is obtained by restricting the definition of the basis of V_ℓ to the subsets of the triangulation. We will assume that the basis is ordered in such a way that functions in V_ℓ^S are before those in V_ℓ^E and the functions in V_ℓ^E are before all in V_ℓ^L. Then, a function $u \in V_\ell$ is represented by a coefficient vector u_ℓ of the form $(u_\ell^S, u_\ell^E, u_\ell^L)^T$. Using the splitting of spaces, equation (2.3) can be partitioned into the system

$$\begin{pmatrix} A_\ell^{SS} & A_\ell^{SE} & \\ A_\ell^{ES} & A_\ell^{EE} & A_\ell^{EL} \\ & A_\ell^{LE} & A_\ell^{LL} \end{pmatrix} \begin{pmatrix} u_\ell^S \\ u_\ell^E \\ u_\ell^L \end{pmatrix} = \begin{pmatrix} f_\ell^S \\ f_\ell^E \\ f_\ell^L \end{pmatrix}, \tag{2.8}$$

where $u_\ell^S \in V_\ell^S$, $u_\ell^L \in V_\ell^L$, and $u_\ell^E \in V_\ell^E$. The parts of the right-hand side belong to the corresponding dual spaces. The matrices A_ℓ^{SS}, A_ℓ^{LL}, and A_ℓ^{EE} are the result of restricting the bilinear form $a(\cdot,\cdot)$ to the spaces V_ℓ^S, V_ℓ^L, and V_ℓ^E, respectively. In particular, A_ℓ^{SS} corresponds to a matrix assembled for the interior degrees of freedom of the fine cells with homogeneous Dirichlet boundary conditions on the refinement edge. The matrices A_ℓ^{SE} and A_ℓ^{ES} consist of coupling terms from V_ℓ^S to V_ℓ^E and vice versa. Since we expect the degrees of freedom of V_ℓ^E to be a small number compared to the whole mesh, these matrices have only a few nonzero entries and can be stored efficiently. Due to the elimination of hanging nodes on the refinement edge, all actual degrees of freedom in V_ℓ^E are coarse level degrees of freedom. Therefore, in the actual implementation, the splitting (2.7) reduces to the two spaces V_ℓ^S and V_ℓ^L only, where V_ℓ^E has been added to the latter. Nevertheless, for the sake of presentation, we will continue using a splitting in three spaces.

Differing from discontinuous Galerkin methods, the matrix A_ℓ^{SS} does not originally exist in our data structures, since the actual level space $\tilde{V}_\ell$ includes V_ℓ^S and $\mathring{V}_\ell$. Instead of A_ℓ^{SS}, we generate the matrix

$$\tilde{A}_\ell = \begin{pmatrix} A_\ell^{SS} & 0 \\ 0 & I \end{pmatrix}, \tag{2.9}$$

which corresponds to the fine level matrix after eliminating "boundary values" on the refinement edge. Here, I is the identity on $\mathring{V}_\ell$, the functions on the refinement edge including the hanging nodes.

With this setup, we are ready to compute level residuals. Let $d_\ell, x_\ell \in V_\ell$ split into the S, E, and L components, respectively. Since the matrix A_ℓ^{SS} is not available to us, we first extend d_ℓ^S and x_ℓ^S by zero to vectors in $\tilde{V}_\ell$ and compute the auxiliary vector $\tilde{r}_\ell = \tilde{d}_\ell - \tilde{A}_\ell \tilde{x}_\ell$, composed of $r_\ell^S \in V_\ell^S$ and a zero component in $\mathring{V}_\ell$. This yields the multilevel residual

$$r_\ell = \begin{pmatrix} r_\ell^S - A_\ell^{SE} x_\ell^E \\ d_\ell^E - A_\ell^{ES} x_\ell^S - A_\ell^{EE} x_\ell^E - A_\ell^{EL} x_\ell^L \\ d_\ell^L - A_\ell^{LE} x_\ell^E - A_\ell^{LL} x_\ell^L \end{pmatrix}.$$

We note that in this formula, only the matrices A_ℓ^{SE} and A_ℓ^{ES} actually exist in the implementation. All other parts of the residual are computed recursively on lower levels.

2.4. Prolongation and restriction. The prolongation of a coarse grid vector to the next finer mesh in a finite element context is usually the embedding operator. It will be used here as well, but we have to study its action on the different subspaces. It amounts to representing a coarse grid function by fine grid basis functions. This operation is usually performed with a stencil computed by interpolation on each coarse mesh cell. Since the result of this operation is a coarse grid function, values in hanging nodes will automatically conform to the coarse side of the face. Nevertheless, our goal is representing the vector in condensed form. Therefore, an additional condense operation is necessary. Thus, the structure of the prolongation operator $R_{\ell-1}^T : V_{\ell-1} \to V_\ell$ is as follows:

- The identity for functions in $V_{\ell-1}$, which are in V_ℓ^L as well.
- Standard embedding from $V_{\ell-1}$ into V_ℓ^S, taking boundary conditions into account.
- For those functions in $V_{\ell-1}$ which require functions in V_ℓ^S and V_ℓ^E for their representation in V_ℓ, we use the standard embedding as well, taking into account that the basis functions in V_ℓ^E after condensation of hanging nodes have nonstandard shapes and that there are no basis functions for the node functionals on the refinement edge which do not belong to the coarse grid. To make sure that the function is still in condensed form we incorporate boundary conditions into the restriction matrix.

The restriction operator $R_{\ell-1} : V_\ell \to V_{\ell-1}$ is chosen as the transpose of the prolongation operator in order to preserve symmetry of the method.

2.5. Local smoothing. In order to bound the overall complexity of the algorithm linearly by the number of degrees of freedom, we restrict the smoothing method to the subspace V_ℓ^S, that is, to functions with support inside the region $\mathbb{T}_\ell^S$. Since the fine grid degrees of freedom on the refinement edge are eliminated from the global system as "hanging nodes" (see, e.g., [29]), they do not contribute to the fine level and can be smoothened on the coarser level. Smoothing on V_ℓ^L will be performed on a coarser level as well. Then, the smoother in the local version of the algorithm is given as

$$\tilde{S}_\ell^{(i)} = \begin{pmatrix} S_{\ell;S}^{(i)} & 0 \\ 0 & I \end{pmatrix}, \qquad S_\ell^{(i)} = \begin{pmatrix} S_{\ell;S}^{(i)} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

where $S_{\ell;S}^{(i)}$ is the restriction of the smoother, for instance, the Gauss–Seidel method, to the space V_ℓ^S and $S_{\ell;S}^{(i)}$ is its representation on the whole space V_ℓ. The operator $\tilde{S}_\ell^{(i)}$ is the actually implemented smoothing operator for the matrix $\tilde{A}_\ell$. Instead of Algorithm 3 below, any smoothing method suitable for this matrix could be used here.

Entering the definitions of grid transfer and smoothing operators for locally refined grids into Algorithm 1 and simplifying yields the following algorithm.

ALGORITHM 2. Let $P_0 = A_0$ and $x^{(0)} = 0$. Then, the action of the operator P_ℓ^{-1} on a vector d_ℓ is defined as follows:

1. (Presmoothing) On the subspace V_ℓ^S only, compute $\tilde{x}^{(m_\ell)}$ iteratively by

$$\tilde{x}^{(i)} = \tilde{x}^{(i-1)} + \tilde{S}_\ell^{(i)}(\tilde{d}_\ell - \tilde{A}_\ell \tilde{x}^{(i-1)}), \quad i = 1, \dots, m_\ell,$$

with $\tilde{d}_\ell = (d_\ell^S, 0)^T$. Let $x^{(m_\ell)} = (x_S^{(m_\ell)}, 0, 0)^T$ with $x_S^{(m_\ell)}$ the restriction of $\tilde{x}^{(m_\ell)}$ to V_ℓ^S. Since, due to the form of $\tilde{S}_\ell^{(i)}$ and $\tilde{A}_\ell$, the boundary values of $\tilde{x}^{(m_\ell)}$ are equal to zero, we have $x^{(m_\ell)} = (\tilde{x}^{(m_\ell)}, 0)^T$.

2. (Coarse grid correction) Let

$$y^{(0)} = x^{(m_\ell)} + R_{\ell-1}^T P_{\ell-1}^{-1}\big(R_{\ell-1}^S(d_\ell^S - A_\ell^{SS} x_S^{(m_\ell)}) + d_\ell^E - A_\ell^{ES} x_S^{(m_\ell)}\big).$$

3. (Postsmoothing) Compute $y^{(m_\ell)}$ iteratively by

$$\tilde{y}^{(i)} = \tilde{y}^{(i-1)} + \tilde{S}_\ell^{(m_\ell+i)}(\tilde{g}_\ell - \tilde{A}_\ell \tilde{y}^{(i-1)}), \quad i = 1, \dots, m_\ell.$$

where $\tilde{g}_\ell = (d_\ell^S, 0)^T - (A_\ell^{SE} y_E^{(0)}, 0)^T$.

4. Set $P_\ell^{-1} d_\ell = (y_S^{(m_\ell)}, y_E^{(0)}, y_L^{(0)})$.

2.6. Overlapping Schwarz smoother. It is well known that standard Jacobi and Gauss–Seidel smoothers deteriorate dramatically when the polynomial degree of the finite element discretization is increased. Therefore, we are looking for a smoother which is nearly as simple but overcomes this problem. This smoother can be found by using a multiplicative Schwarz method with subspaces related to cells or patches of cells. Let $\{V_{\ell,k}\}$ with $k = 1, \dots, N_\ell$ be such a set of subspaces in V_ℓ^S which will be specified in detail below. Then, instead of defining the action of the operator S_ℓ, the following algorithm directly describes how to obtain $x^{(i)}$ from $x^{(i-1)}$ in the presmoothing step of Algorithm 2 (for details on the relation to S_ℓ, see, for instance, [44]).

ALGORITHM 3. One step of the symmetric, multiplicative Schwarz smoother with right-hand side d_ℓ is defined by the following:

1. Let $y^{(0)} = \tilde{x}^{(i-1)}$.
2. For each $k = 1, \dots, N_\ell$, compute $\tilde{y}^{(k)} \in V_{\ell,k}$ as the solution of

$$A_{\ell,k}\tilde{y}^{(k)} = P_{\ell,k}\left(\tilde{d}_\ell - \tilde{A}_\ell \tilde{y}^{(k-1)}\right),$$

where $P_{\ell,k}$ is the ℓ^2-projection from V_ℓ to $V_{\ell,k}$, and let

$$\tilde{y}^{(k)} = \tilde{y}^{(k-1)} + \tilde{y}^{(k)}.$$

3. For each $k = N_\ell, \dots, 1$, compute $\tilde{y}^{(2N_\ell+1-k)} \in V_{\ell,k}$ as the solution of

$$A_{\ell,k}\tilde{y}^{(2N_\ell+1-k)} = P_{\ell,k}\left(\tilde{d}_\ell - \tilde{A}_\ell \tilde{y}^{(2N_\ell-k)}\right),$$

and let

$$\tilde{y}^{(2N_\ell+1-k)} = \tilde{y}^{(2N_\ell-k)} + \tilde{y}^{(2N_\ell+1-k)}.$$

4. Let $\tilde{x}^{(i)} = \tilde{y}^{(2N_\ell)}$.

Here, $A_{\ell,k}$ is the projection of the matrix $\tilde{A}_\ell$ onto the subspace $V_{\ell,k}$. Postsmoothing is done accordingly.

It remains to specify the subspaces $V_{\ell,k}$. In the elliptic case, the inversion of cell matrices has proved very successful for discontinuous Galerkin methods [30]. There, a block Gauss–Seidel smoother based on inverting cell matrices yields preconditioners with very weak dependence on the polynomial degree. With continuous elements, an overlapping smoother is more natural. Thus, for a cell T_k on level ℓ let $V_{\ell,k}$ be the subspace of functions in V_ℓ^S which are not identically zero on T_k. Computational results for this smoother are reported in subsection 3.1.

For Maxwell problems, we apply the same type of smoother but follow [1] in associating the subspaces $V_{\ell,k}$ with all the interior degrees of freedom of a patch $\Omega_{\ell,k}$ of cells around a common vertex on level ℓ. It has been proved in [1] that this smoother yields a uniformly convergent multigrid method on regular meshes. A crucial ingredient in the proof is an exact sequence of discrete spaces on the chosen

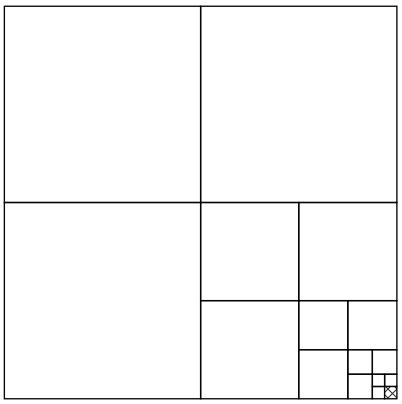

FIG. 2.4. *A mesh hierarchy with the same number of degrees of freedom on each level. Half of the interior edges bear hanging nodes.*

patches. Since smoothing *inside* the refined region is equivalent to smoothing on a domain with strong boundary conditions, this sequence retains the same structure as in the globally refined case, for instance, in three dimensions with Ω_ℓ the part of the domain covered by $\mathbb{T}_\ell^S$,

$$\begin{array}{ccccccc} H_0^1(\Omega_{\ell,k}) & \xrightarrow{\nabla} & H_0^{\mathrm{curl}}(\Omega_{\ell,k}) & \xrightarrow{\nabla\times} & H_0^{\mathrm{div}}(\Omega_{\ell,k}) & \xrightarrow{\nabla\cdot} & L^2(\Omega_{\ell,k})/\mathbb{R} \\ \cup & & \cup & & \cup & & \cup \\ Z_h & & V_h & & R_h & & Q_h \end{array},$$

where the discrete spaces Z_h, V_h, R_h, and Q_h are the conforming finite element discretizations by Lagrange, Nédélec, Raviart–Thomas, and discontinuous tensor product polynomial elements, respectively. Smoothing on the refinement edge would require a sequence of the spaces modified to account for patches with hanging nodes. We report results for this method in subsection 3.2. A smoother with similar choice of subspaces has recently been applied to the Poisson problem in [39], where optimality was proved under certain conditions; we point out, though, that the overlap is much higher than that resulting from using cell matrices.

2.7. Complexity of the algorithm. We show that the number of operations of Algorithm 2 grows linearly with the number of degrees of freedom on the finest level if the classical V-cycle is chosen. To this end, we have to assume that the complexity of the smoother $S_{\ell;S}^{(i)}$ is linear with respect to the dimension of the space it acts on, which holds for standard relaxation methods as well as incomplete LU and Cholesky factorization with limited fill-in. It holds for the smoother outlined in subsection 2.6.

First, we note that the pre- and postsmoothing steps operate on V_ℓ^S only; thus in the whole recursive cycle, every degree of freedom will be touched only once by these operations. The exception from this are the degrees of freedom in V_ℓ^E, which are part of the space $\tilde{V}_\ell$, and are thus involved in smoothing on level ℓ, although not actually smoothened themselves. Additionally, they will be smoothened on level $\ell-1$. Due to the restriction to one-irregular meshes, they will be in $V_{\ell-1}^S$ and will not be smoothened on any coarser mesh. Thus, they might be operated on at most twice per smoothing. Figure 2.4 shows that the number of faces with hanging nodes is not necessarily of lower order than the degrees of freedom but is bounded by them. In actual adaptive refinement cycles, it is much more likely that the refinement edges

form a small subset of the total set of edges. In that case, the additional work due to multiplication with the matrices A^{SE} and A^{ES} becomes negligible.

We conclude that the contribution of the two smoothing steps is linear with respect to the total number of degrees of freedom in the hierarchy $\{T_\ell\}$, which again is bounded by the number of degrees of freedom on the finest level by a geometric sum.

It remains to study the complexity of the grid transfer. According to step 2 of Algorithm 2, this transfer consists of three parts: First, we have the restriction of the initial residual d_L onto all lower level meshes, which is of the order of the total degrees of freedom in the hierarchy. Second, we need to restrict the local residuals after local smoothing. This involves only the result of $A_\ell^{SS} x_S$, which, summed up over all levels, gives again the number of degrees of freedom in the hierarchy. The same holds for the prolongation operator R_ℓ^T, such that we can conclude that the whole intergrid transfer is of the order of the number of degrees of freedom in the mesh hierarchy.

The complexity analysis of standard multigrid methods without local refinement continues by noting that the number of degrees of freedom increases geometrically by factors of four and eight from one level to the next in two and three dimensions, respectively. Thus, it is concluded that the complexity of the V-cycle, variable V-cycle, and W-cycle is bounded linearly by the number of degrees of freedom on the finest level. Such an argument is invalid here. In particular, in hierarchies obtained through adaptive iterations before saturation is achieved, this condition is usually violated. An extreme example is the hierarchy in Figure 2.4, which exhibits the same number of degrees of freedom on each level. Thus, we conclude that on general hierarchies we can show optimal complexity for the V-cycle only with respect to the total number of degrees of freedom in the hierarchy.

2.8. Memory requirements. The situation for memory requirements is similar to that of the computational complexity. A standard multigrid method without hanging nodes involves only the matrices A_ℓ^{SS}, since the triangulation subsets $\mathbb{T}_\ell^L$ and $\mathbb{F}_\ell^E$ are empty. The only additional matrices stored in the local method are A^{SE} and A^{ES}. In the worst-case example of Figure 2.4, these matrices are of about the same size as A_ℓ^{SS}. Under the more reasonable assumption that $\mathbb{F}_\ell^E$ is a small subset of the set of faces, these matrices can be stored in a compact way involving only degrees of freedom in V_ℓ^E, thus with negligible memory overhead.

3. Numerical experiments.

3.1. The Poisson problem. We test our algorithm with the following model problem: Let $\Omega = (-1,1)^d$, choose the bilinear form $a_\Delta(\cdot,\cdot)$ in (2.2), and let $f \equiv 1$. Starting with a single grid cell $\mathbb{T}_0 = \{\bar{\Omega}\}$, we apply four refinement strategies:

1. Refinement of each grid cell (global refinement) in each step for comparison.
2. Refinement of all cells in the positive quadrant/octant. Figure 3.1 shows the resulting grids with finest cells on levels 2, 3, and 6. The grids are made v-one-irregular (see Definition 2.2) to accommodate the multilevel method.
3. Refinement of all cells intersecting the circle of radius $1/(4\pi)$. The resulting grids on levels 3, 4, and 9 are shown in Figure 3.2. Again, the refinement is smoothened to assure that the grid is v-one-irregular.
4. Applied to the singular solution on a square with a slit, we refine according to the a posteriori error estimator presented in [43] using the bulk criterion in [21].

The linear systems resulting from the weak formulation (2.1) on these meshes are solved by the cg method with the multilevel preconditioner developed above. First,

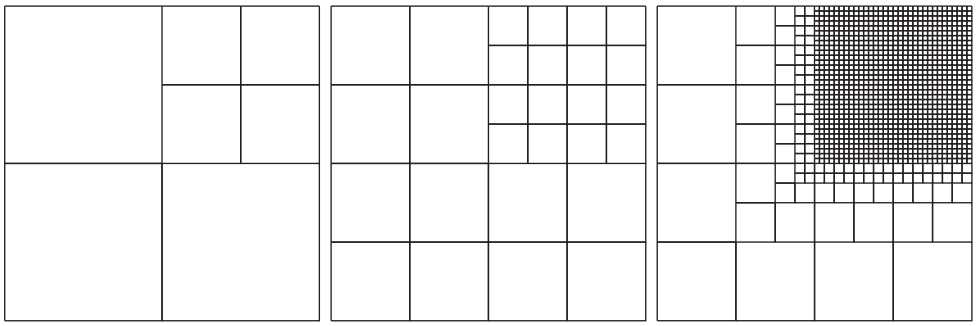

FIG. 3.1. *Refinement of the positive quadrant, levels* 2, 3, *and* 6, *with v-one-irregular closure.*

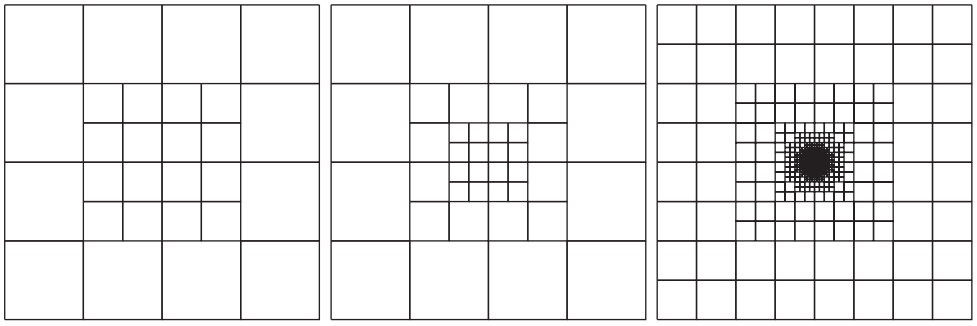

FIG. 3.2. *Refinement of a circle, levels* 3, 4, *and* 9, *with v-one-irregular closure.*

we use tensor product polynomials of degree 1–3 on each cell, denoted by $\mathbb{Q}_1$ to $\mathbb{Q}_3$. The start vector is $u^{(0)} = 0$ on each mesh. The values displayed in the tables are the number of steps n_{10} needed to reduce the norm of the residual r by a factor of 10^{10} and the average logarithmic convergence rate according to Varga [42]

$$\bar{r} := \frac{1}{n} \log_{10} \frac{|r_0|}{|r_n|},$$

where $|r_n|$ is the Euclidean norm of the residual vector r_n after the nth cg step. Note that while $\bar{r}$ is approximately $10/n_{10}$, it is not rounded to a single digit and thus is a finer measure of convergence speed.

In Table 3.1, we report results for the classical V-cycle with one symmetric pre- and one symmetric postsmoothing step on each level for the different refinement cases; global refinement is included as a benchmark. As can be seen, the number of steps is independent of the refinement level in all three cases. Moreover, on locally refined meshes the method performs only slightly worse. When we turn to the variable V-cycle, Table 3.2 shows that this is actually reverted if we use only one smoothing step on the finest level and choose $\beta_0 = \beta_1 = 2$ in (2.6). We compared the same methods to higher order polynomials and obtained the same results. In Table 3.3 we report convergence rates for refinement of a circle and polynomial spaces up to $\mathbb{Q}_9$. They are independent of the refinement level or of the existence of hanging nodes for increasing polynomial degree. Tables 3.4 and 3.5 show convergence rates obtained on a slit domain with uniform and adaptive refinement, respectively. The adaptive refinement was performed using the estimator proposed in [43].

TABLE 3.1

Iteration steps and convergence rates for the preconditioned cg method. One symmetric pre- and one symmetric postsmoothing step on each level with $\mathbb{Q}_1$-elements in two dimensions.

	Global		Quadrant		Circle	
L	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$
2	1	16.00	1	16.00	1	16.00
3	3	4.83	1	15.99	3	4.83
4	4	2.84	4	2.56	5	2.29
5	5	2.25	6	1.69	6	1.83
6	5	2.15	7	1.61	6	1.76
7	5	2.12	7	1.60	7	1.44
8	5	2.08	7	1.60	7	1.61
9	6	2.06	7	1.60	7	1.57
10	6	2.02	7	1.60	7	1.59
11	6	1.96	7	1.60	7	1.59
12	6	1.92	7	1.60	7	1.59

TABLE 3.2

Iteration steps and convergence rates for the preconditioned cg method for $\mathbb{Q}_1$-elements. Variable smoothing with one pre- and one postsmoothing step on the finest level.

	Global		Quadrant		Circle	
L	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$
2	1	16.00	1	16.00	1	16.00
3	4	2.53	1	15.99	4	2.53
4	9	1.15	6	1.96	6	1.68
5	9	1.16	8	1.39	6	1.78
6	8	1.28	7	1.46	6	1.82
7	8	1.33	7	1.51	7	1.48
8	8	1.35	7	1.56	7	1.58
9	7	1.43	7	1.59	7	1.57
10	7	1.44	7	1.62	6	1.69
11	7	1.44	7	1.64	6	1.73

TABLE 3.3

Convergence rates for the preconditioned cg method. Refinement into a circle in $2d$. Variable smoothing with one pre- and one postsmoothing step on the finest level.

L	$\mathbb{Q}_1$	$\mathbb{Q}_2$	$\mathbb{Q}_3$	$\mathbb{Q}_4$	$\mathbb{Q}_5$	$\mathbb{Q}_6$	$\mathbb{Q}_7$	$\mathbb{Q}_8$	$\mathbb{Q}_9$
2	16.0	1.9	1.9	1.6	1.7	1.5	1.5	1.5	1.5
3	2.5	1.3	1.3	1.5	1.5	1.5	1.4	1.5	1.4
4	1.7	1.4	1.5	1.5	1.5	1.5	1.4	1.5	1.5
5	1.8	1.5	1.6	1.6	1.6	1.6	1.5	1.5	1.5
6	1.8	1.7	1.7	1.7	1.6	1.6	1.6	1.6	1.5
7	1.5	1.5	1.6	1.5	1.5	1.5	1.5	1.5	1.4
8	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.5
9	1.6	1.6	1.6	1.7	1.6	1.6	1.6	1.6	1.5
10	1.7	1.7	1.7	1.7	1.7	1.7	1.6	1.6	1.6
11	1.7	1.7	1.7	1.8	1.7	1.7	1.7	1.7	1.6
12	1.7	1.7	1.7	1.7	1.7	1.8	1.7	1.7	1.6

In three dimensions, we compute the same test cases on the cube $(-1,1)^3$ for examples of the resulting locally refined meshes we refer to in Figure 3.3. Convergence rates of the preconditioned cg method for this case are shown in Tables 3.6 and 3.7. Again, we see that the convergence rates are independent of the refinement level and nearly independent of the presence of hanging nodes on all meshes. Table 3.8 shows that the convergence rates in three dimensions as well are independent of the refinement level or of the existence of hanging nodes for increasing polynomial degree.

TABLE 3.4
Convergence rates for the preconditioned cg method. Slit domain with uniform refinement. Smoothing with one pre- and one postsmoothing step on each level.

L	$\mathbb{Q}_1$	$\mathbb{Q}_2$	$\mathbb{Q}_3$	$\mathbb{Q}_4$	$\mathbb{Q}_5$	$\mathbb{Q}_6$	$\mathbb{Q}_7$	$\mathbb{Q}_8$	$\mathbb{Q}_9$
2	6.60	15.49	14.86	14.55	14.22	13.78	13.29	13.03	11.49
3	3.13	3.02	2.47	2.35	2.27	2.24	2.16	2.15	2.07
4	2.31	2.51	2.23	2.15	2.19	2.16	2.06	2.08	2.06
5	2.12	2.35	2.14	2.07	2.15	2.16	2.06	2.08	2.06
6	2.03	2.29	2.11	2.04	2.13	2.16	2.06	2.08	2.06
7	2.00	2.26	2.09	2.02	2.11	2.16	2.06	2.08	2.06
8	1.95	2.24	2.08	2.00	2.10	2.16	2.06	2.08	
9	1.91	2.20	2.06	2.03	2.09	2.16			
10	1.88	2.17	2.06	2.02					

TABLE 3.5
Convergence rates for the preconditioned cg method. Slit domain with adaptive refinement. Smoothing with one pre- and one postsmoothing step on each level.

L	$\mathbb{Q}_1$	$\mathbb{Q}_2$	$\mathbb{Q}_3$	$\mathbb{Q}_4$	$\mathbb{Q}_5$	$\mathbb{Q}_6$	$\mathbb{Q}_7$	$\mathbb{Q}_8$	$\mathbb{Q}_9$
2	6.60	3.02	2.47	2.35	2.27	2.24	2.16	2.15	2.07
3	3.89	1.91	1.90	1.85	1.84	1.83	1.80	1.81	1.77
4	1.59	1.68	1.72	1.72	1.72	1.72	1.73	1.78	1.76
5	1.42	1.67	1.72	1.70	1.71	1.72	1.72	1.79	1.76
6	1.34	1.53	1.71	1.69	1.70	1.72	1.73	1.79	1.76
7	1.38	1.53	1.69	1.69	1.70	1.73	1.74	1.80	1.77
8	1.42	1.51	1.67	1.70	1.71	1.73	1.75	1.80	1.78
9	1.40	1.51	1.67	1.70	1.72	1.74	1.76	1.81	1.78
10	1.44	1.53	1.56	1.70	1.73	1.75	1.77	1.81	1.79
11	1.45	1.52	1.53	1.70	1.73	1.75	1.78	1.81	1.80

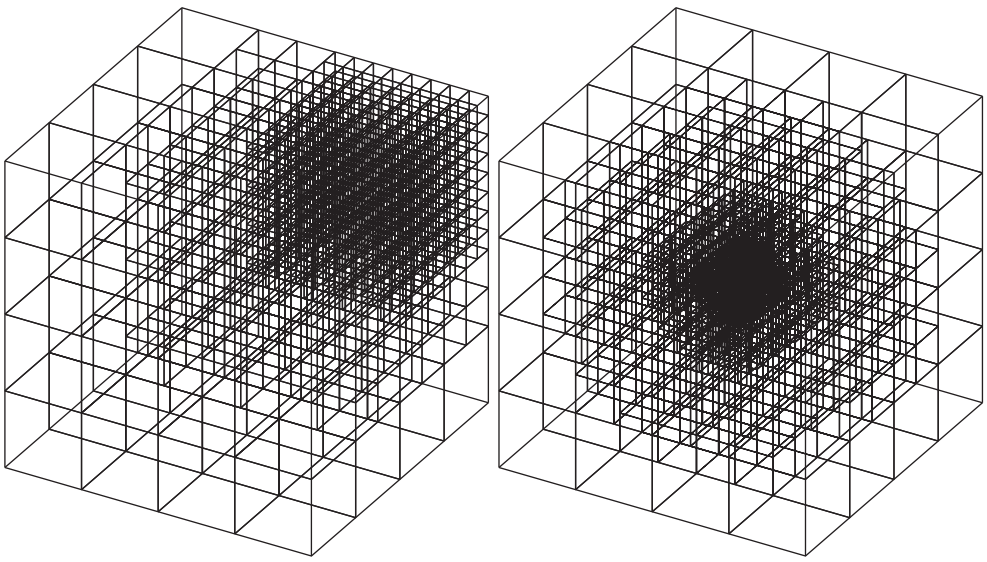

FIG. 3.3. *Locally refined meshes in three dimensions. First octant refined to level* 4 *(left), and ball refined to level* 6 *(right).*

TABLE 3.6

Performance of the preconditioned cg method in three dimensions. One symmetric pre- and one postsmoothing step on each level.

	$\mathbb{Q}_1$-elements						$\mathbb{Q}_2$-elements					
	Global		Octant		Ball		Global		Octant		Ball	
L	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$
2	1	16.00	1	16.00	1	16.00	3	4.76	3	4.76	3	4.76
3	2	5.38	1	16.85	2	5.38	4	2.99	5	2.70	4	2.99
4	4	3.13	5	2.28	5	2.08	5	2.51	6	1.72	6	1.73
5	5	2.28	7	1.60	6	1.94	5	2.35	7	1.58	6	1.70
6	5	2.15	7	1.58	6	1.89	5	2.28	7	1.57	6	1.70
7	5	2.08	7	1.58	8	1.37	5	2.27	7	1.57	8	1.35
8	5	2.03	7	1.58	7	1.52			7	1.57	7	1.50

TABLE 3.7

Performance of the preconditioned cg method in three dimensions. Variable block smoothing with one pre- and one postsmoothing step on the finest level.

	$\mathbb{Q}_1$-elements						$\mathbb{Q}_2$-elements					
	Global		Octant		Ball		Global		Octant		Ball	
L	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$	n_{10}	$\bar{r}$
2	1	16.00	1	16.00	1	16.00	5	2.39	5	2.39	5	2.39
3	4	2.99	1	16.85	4	2.99	7	1.57	5	2.17	7	1.57
4	8	1.32	6	1.94	6	1.84	7	1.46	7	1.58	7	1.55
5	8	1.30	7	1.50	6	1.89	7	1.44	7	1.52	6	1.70
6	8	1.32	7	1.51	6	1.94	7	1.51	7	1.56	6	1.76
7	7	1.44	7	1.57	7	1.46	7	1.55	7	1.60	8	1.41
8	7	1.47	7	1.62	7	1.57			6	1.69	7	1.55

TABLE 3.8

Convergence rates for the preconditioned cg method. Refinement into a ball in three dimensions. Variable block smoothing with one pre- and one postsmoothing step on the finest level.

L	$\mathbb{Q}_1$	$\mathbb{Q}_2$	$\mathbb{Q}_3$	$\mathbb{Q}_4$	$\mathbb{Q}_5$	$\mathbb{Q}_6$	$\mathbb{Q}_7$	$\mathbb{Q}_8$
2	16.0	2.4	2.0	1.8	1.7	1.5	1.5	1.43
3	3.0	1.6	1.5	1.4	1.6	1.5	1.4	1.44
4	1.8	1.6	1.5	1.5	1.5	1.5	1.4	1.44
5	1.9	1.7	1.6	1.6	1.6	1.6	1.5	1.52
6	1.9	1.8	1.7	1.7	1.7	1.7	1.5	1.58
7	1.5	1.4	1.4	1.4	1.4	1.4	1.4	
8	1.6	1.6	1.6	1.6	1.6	1.7	1.6	
9	1.4	1.4	1.5	1.5	1.4	1.5		
10	1.5	1.4	1.5	1.5				
11	1.4	1.4						

3.2. The eddy current problem. In this section we study the application to discretization of the bilinear form $a_{\mathrm{curl}}(\cdot,\cdot)$ with $\sigma = 1$ and the curl-conforming elements introduced by Nédélec in [33] (with their natural restriction to two dimensions), denoted as $\mathbb{N}_k$, where the element for $k = 0$ has constant tangential traces. We use the L-shaped domain $\Omega = (-1,1)^2 \setminus (0,1)^2$ in two dimensions and the right-hand side $f = (1,1)^T$. The AWF smoother is implemented using the QR-method for inversion of the local problems on each patch. The global discrete problems are solved by a GMRES method with reorthogonalization, since the operator is indefinite. For mesh refinement, we use the error estimator from [9].

TABLE 3.9
GMRES convergence rates for the Arnold–Falk–Winther smoother depending on mesh level L and order of Nédélec space $\mathbb{N}_k$. Uniform meshes and adaptive meshes. Last three lines show adaptive iteration steps 16–18 *instead of finest level L.*

	Uniform mesh					Adaptive refinement				
L	$\mathbb{N}_0$	$\mathbb{N}_1$	$\mathbb{N}_2$	$\mathbb{N}_3$	$\mathbb{N}_4$	$\mathbb{N}_0$	$\mathbb{N}_1$	$\mathbb{N}_2$	$\mathbb{N}_3$	$\mathbb{N}_4$
2	2.17	1.72	1.68	1.60	1.60	2.17	1.72	1.68	1.60	1.60
3	1.34	1.54	1.76	1.75	1.74	1.22	1.35	1.27	1.30	1.28
4	1.36	1.57	1.74	1.77	1.82	1.29	1.34	1.22	1.23	1.26
5	1.39	1.62	1.68	1.79	1.83	1.26	1.26	1.22	1.27	1.27
6	1.39	1.65	1.71	1.81	1.86	1.25	1.04	1.34	1.27	1.27
7	1.39	1.67	1.72	1.83	1.87	1.27	1.13	1.26	1.27	1.27
8	1.40	1.65	1.72	1.84	1.88	1.28	1.11	1.18	1.19	1.15
9	1.40	1.63	1.71	1.85	—	1.29	1.13	1.24	1.27	1.21
S16						1.27	1.13	1.32	1.33	1.32
S17						1.33	1.15	1.32	1.29	1.33
S18						1.25	1.13	1.35	1.32	1.34

TABLE 3.10
GMRES convergence rates for the Arnold–Falk–Winther smoother in three dimensions, adaptive meshes on the "L-shaped" cube $\Omega = (-1,1)^3 \setminus (0,1)^3$.

L	$\mathbb{N}_0$	$\mathbb{N}_1$	$\mathbb{N}_2$	$\mathbb{N}_3$
2	1.65	1.06	0.85	0.75
3	1.07	0.93	0.82	0.68
4	1.07	0.96	0.90	0.75
5	1.04	0.94	0.95	0.87
6	0.95	0.93	0.96	0.86
7	0.91	0.95	0.96	0.86
8	0.93	0.95	0.96	0.86
9	0.93	0.86	0.96	0.86

In Table 3.9, we report convergence rates for one pre- and one postsmoothing step of the smoother for curl-conforming methods. First, in all columns, we observe convergence rates independent of the mesh size. For $\mathbb{N}_0$, these results coincide with those reported in [1]. Additionally, the convergence rates appear independent of the polynomial degree. The numbers for uniform and adaptive refinement, where we used the case in Figure 3.2, are the same; thus the method is also robust with respect to local refinement.

In three dimensions, we choose as our domain the cube with one corner removed, $\Omega = (-1,1)^3 \setminus (0,1)^3$, and the right-hand side $f = (1,1,1)^T$. In Table 3.10, we report results for adaptive meshes and varying polynomial degrees, confirming the robustness observed in the previous examples.

Conclusions. A multigrid method with local smoothing on locally refined meshes with conforming finite elements of arbitrary order was derived. In particular, we discussed a consistent treatment of hanging nodes on the refinement edge. Since smoothing is applied only locally, this method performs with optimal computational complexity. The resulting convergence rates are mesh independent, and the presence of hanging nodes has a small effect.

Acknowledgments. The authors thank W. Bangerth and M. Besier for their valuable comments on the manuscript. J. Gopalakrishnan and R. Hoppe contributed helpful advice on curl-elliptic problems. The method has been implemented using the

deal.II library [2, 3]; it will be part of future versions of this library, and an example of its use will be incorporated into the deal.II tutorials.

REFERENCES

[1] D. N. Arnold, R. S. Falk, and R. Winther, *Multigrid in H(*div*) and H(*curl*)*, Numer. Math., 85 (2000), pp. 197–217.

[2] W. Bangerth, R. Hartmann, and G. Kanschat, *deal.*II*—A general-purpose object-oriented finite element library*, ACM Trans. Math. Software, 33 (2007), 24.

[3] W. Bangerth, R. Hartmann, and G. Kanschat, `deal.II` *Differential Equations Analysis Library, Technical Reference*, release 6.3, 2010.

[4] W. Bangerth and O. Kayser-Herold, *Data structures and requirements for hp finite element software*, ACM Trans. Math. Software, 36 (2009), 4.

[5] W. Bangerth and R. Rannacher, *Adaptive Finite Element Methods for Differential Equations*, Birkhäuser, Verlag, Basel, 2003.

[6] R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations: Users' Guide* 8.0, SIAM, Philadelphia, 1998.

[7] P. Bastian, *Load balancing for adaptive multigrid methods*, SIAM J. Sci. Comput., 19 (1998), pp. 1303–1321.

[8] P. Bastian and C. Wieners, *Multigrid methods on adaptively refined grids*, Comput. Sci. Eng., 8 (2006), pp. 44–54.

[9] R. Beck, R. Hiptmair, R. H. W. Hoppe, and B. Wohlmuth, *Residual based a posteriori error estimators for eddy current computation*, M2AN Math. Model. Numer. Anal., 34 (2000), pp. 159–182.

[10] R. Becker and M. Braack, *Multigrid techniques for finite elements on locally refined meshes*, Numer. Linear Algebra Appl., 7 (2000), pp. 363–379.

[11] M. Berger, M. Aftosmis, and G. Adomavicius, *Parallel multigrid on Cartesian meshes with complex geometry*, in Parallel Computational Fluid Dynamics (Trondheim, 2000), North-Holland, Amsterdam, 2001, pp. 283–290.

[12] F. Boyer, C. Lapuerta, S. Minjeaud, and B. Piar, *A Local Adaptive Refinement Method with Multigrid Preconditioning Illustrated by Multiphase Flows Simulations*, Technical report, 2008; available online at http://hal.archives-ouvertes.fr/hal-00307186.

[13] J. H. Bramble, *Multigrid Methods*, Pitman Res. Notes Math. Ser. 294, Longman Scientific & Technical, Harlow, UK, 1993.

[14] J. H. Bramble, J. E. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[15] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[16] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, 2nd ed., Springer-Verlag, New York, 2002.

[17] J. M. Cascon, C. Kreuzer, R. H. Nochetto, and K. G. Siebert, *Quasi-optimal convergence rate for an adaptive finite element method*, SIAM J. Numer. Anal., 46 (2008), pp. 2524–2550.

[18] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, New York, Oxford, 1978.

[19] A. Cohen, W. Dahmen, and R. DeVore, *Multiscale decompositions on bounded domains*, Trans. Amer. Math. Soc., 352 (2000), pp. 3651–3685.

[20] A. Cohen, W. Dahmen, and R. DeVore, *Adaptive wavelet methods for elliptic operator equations: Convergence rates*, Math. Comp., 70 (2001), pp. 27–75.

[21] W. Dörfler, *A convergent adaptive algorithm for Poisson's equation*, SIAM J. Numer. Anal., 33 (1996), pp. 1106–1124.

[22] M. Griebel and P. Oswald, *On the abstract theory of additive and multiplicative Schwarz algorithms*, Numer. Math., 70 (1995), pp. 163–180.

[23] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[24] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.

[25] R. Hiptmair and W. Zheng, *Local multigrid in* **H**(**curl**), J. Comput. Math., 27 (2009), pp. 573–603.

[26] V. John, P. Knobloch, G. Matthies, and L. Tobiska, *Non-nested multi-level solvers for finite element discretisations of mixed problems*, Computing, 68 (2002), pp. 313–341.

[27] A. C. Jones, *A Projected Multigrid Method for the Solution of Nonlinear Finite Element Problems on Adaptively Refined Grids*, Dissertation, The University of Leeds, Leeds, UK, 2005.
[28] A. C. Jones and P. K. Jimack, *An adaptive multigrid tool for elliptic and parabolic systems*, Internat. J. Numer. Methods Fluids, 47 (2005), pp. 1123–1128.
[29] G. Kanschat, *Parallel and Adaptive Galerkin Methods for Radiative Transfer Problems*, Dissertation (preprint SFB 359, 1996-29), Universität Heidelberg, Heidelberg, Germany, 1996.
[30] G. Kanschat, *Multi-level methods for discontinuous Galerkin FEM on locally refined meshes*, Comput. & Structures, 82 (2004), pp. 2437–2445.
[31] S. F. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, Frontiers Appl. Math. 6, SIAM, Philadelphia, 1989.
[32] W. F. Mitchell, *Parallel adaptive multilevel methods with full domain partitions*, Appl. Numer. Anal. Comput. Math., 1 (2004), pp. 36–48.
[33] J.-C. Nédélec, *Mixed finite elements in* $\mathbf{R}^3$, Numer. Math., 35 (1980), pp. 315–341.
[34] M.-C. Rivara, *Design and data structure of fully adaptive, multigrid, finite-element software*, ACM Trans. Math. Software, 10 (1984), pp. 242–264.
[35] U. Rüde, *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, Frontiers Appl. Math. 13, SIAM, Philadelphia, 1993.
[36] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[37] R. S. Sampath and G. Biros, *A parallel geometric multigrid method for finite elements on octree meshes*, SIAM J. Sci. Comput., 32 (2010), pp. 1361–1392.
[38] A. Schmidt and K. Siebert, *Design of Adaptive Finite Element Software. The Finite Element Toolbox ALBERTA*, Lect. Notes Comput. Sci. Eng. 42, Springer-Verlag, Berlin, 2005.
[39] J. Schöberl, J. M. Melenk, C. Pechstein, and S. Zaglmayr, *Additive Schwarz preconditioning for p-version triangular and tetrahedral finite elements*, IMA J. Numer. Anal., 28 (2008), pp. 1–24.
[40] P. Šolín, J. Červený, and I. Doležel, *Arbitrary-level hanging nodes and automatic adaptivity in the hp-FEM*, Math. Comput. Simulation, 77 (2008), pp. 117–132.
[41] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.
[42] R. S. Varga, *Matrix Iterative Analysis*, 2nd ed., Springer Ser. Comput. Math. 27, Springer-Verlag, Berlin, 2000.
[43] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*, Wiley/Teubner, Chichester, New York, 1996.
[44] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.
[45] H. Yserentant, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.