SIAM J. SCI. COMPUT.
Vol. 33, No. 5, pp. 2888–2902

ALGEBRAIC MULTIGRID FOR HIGH-ORDER HIERARCHICAL $H(curl)$ FINITE ELEMENTS*

JAMES H. LAI† AND LUKE N. OLSON†

Abstract. Classic multigrid methods are often not directly applicable to nonelliptic problems such as curl-type partial differential equations (PDEs). Curl-curl PDEs require specialized smoothers that are compatible with the gradient-like (near) null space. Moreover, recent developments have focused on replicating the grad-curl-div de Rham complex in a multilevel hierarchy through smoothed aggregation based algebraic multigrid. These approaches have been successful for Nédélec finite elements (i.e., $H(curl)$ edge elements), but do not extend naturally to high-order representations. In this paper we consider hierarchical high-order Whitney elements for the curl-curl eddy current problem and devise a scalable multilevel approach. Our method generates a hierarchy similar to p-type multigrid, which results in a coarse level that is amenable to further coarsening through the established process of a multilevel complex. The natural hierarchy of the elements induces an effective interpolation operator and motivates the construction of a compatible gradient smoothing process. We detail the multilevel solver for a hierarchical $H(curl)$ basis and present numerical results in support of the method.

Key words. algebraic multigrid, high-order, curl, edge elements

AMS subject classification. 65N55

DOI. 10.1137/100799095

1. Introduction.

High-order finite element methods are popular due to their spectral convergence properties and blocked data pattern. However, the high-order construction in the approximation leads to reduced sparsity in the system and increased valence in the matrix graph. As a result, traditional algebraic multigrid methods are not directly applicable. Moreover, since the condition number of the system increases disproportionately with the polynomial order of the approximation in comparison to h-type enrichment of the basis, the adverse effects on multigrid convergence are even more pronounced.

Multigrid methods have been used to effectively solve high-order discretizations, but these approaches target scalar H^1 finite element spaces [6, 9, 13]. Considering instead function spaces $H(curl)$ and $H(div)$, defined as

$$H(curl) = \{\mathbf{u} \in \mathbf{L_2}(\Omega) \,|\nabla \times \mathbf{u} \in L_2(\Omega)\},$$
$$H(div) = \{\mathbf{u} \in \mathbf{L_2}(\Omega) \,|\nabla \cdot \mathbf{u} \in L_2(\Omega)\},$$

the extension of these multigrid methods to high-order vector elements is not clear and standard methods do not perform well for high-order vector discretizations (cf. Figure 5.1).

One difficulty in solving $H(curl)$ problems is attributed to the large kernel of the curl operator. Since $\nabla \times \nabla u = 0$ for any differentiable u, the kernel of the curl operator contains gradients of all differentiable functions. The relationship between

*Received by the editors June 16, 2010; accepted for publication (in revised form) May 19, 2011; published electronically October 27, 2011. This work was partially supported by the NSF under award DMS 07-46676.
http://www.siam.org/journals/sisc/33-5/79909.html
†Siebel Center for Computer Science, University of Illinois at Urbana-Champaign, 201 N. Goodwin Avenue, Urbana, IL 61801 (jhlai2@illinois.edu, lukeo@illinois.edu).

H^1, $H(curl)$, and $H(div)$ is given by the de Rham differential complex

$$H^1 \xrightarrow{\nabla} H(curl) \xrightarrow{\nabla\times} H(div) \xrightarrow{\nabla\cdot} L_2, \tag{1.1}$$

which forms an exact sequence. That is, the image of one mapping is equal to the kernel of the next. For example,

$$\begin{aligned} \text{Image}(\nabla) &= \text{Kernel}(\nabla\times), \\ \text{Image}(\nabla\times) &= \text{Kernel}(\nabla\cdot). \end{aligned}$$

An effective multilevel strategy [10] is to preserve the complex (1.1) when projecting to different levels in the hierarchy.

In this paper we develop an algorithm based on algebraic multigrid for solving systems discretized by high-order $H(curl)$ finite elements. In particular, we focus on the solution to the eddy current problem from electromagnetics, which results in an $H(curl)$-elliptic weak form. Let $\Omega \subset \mathbb{R}^2$ be a convex domain and consider functions μ, σ, and $\mathbf{f}$. The eddy current problem is of the form

$$\nabla \times (\mu \nabla \times \mathbf{u}) + \sigma \mathbf{u} = \mathbf{f} \quad \text{in } \Omega, \tag{1.2}$$

$$\mathbf{n} \times \mathbf{u} = 0 \quad \text{on } \partial\Omega. \tag{1.3}$$

The corresponding weak form of this problem is to find $\mathbf{u} \in H_0(curl)$ such that

$$\int_\Omega \mu \, (\nabla \times \mathbf{u}) \cdot (\nabla \times \mathbf{v}) \, d\Omega + \int_\Omega \sigma \mathbf{u} \cdot \mathbf{v} d\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v} d\Omega \tag{1.4}$$

for all $v \in H_0(curl)$. We assume (1.4) is discretized by high-order $H(curl)$ conforming elements on a triangular mesh, which results in the linear system

$$K^{(e)} u = f. \tag{1.5}$$

We assume that $\sigma > 0$ so that the resulting system is positive definite.

Because $K^{(e)}$ is not an M-matrix, standard multigrid techniques do not typically perform well. Strategies for geometric multigrid methods for lowest order $H(curl)$ discretizations require hybrid smoothers [1, 7]. An algebraic multigrid method for these problems was first presented in [10], which observed that edge aggregates can be induced via nodal aggregates. The combination of this idea along with hybrid smoothers presents an amenable solver for the curl-curl problem; however, the particular choice of prolongation operator used in [10] results in a method that is dependent on the mesh size h. As a result, smoothed aggregation techniques have been used to improve the algebraic multigrid method for $H(curl)$ systems with much success [4]. We detail our use of this framework in section 3.

$H(curl)$ conforming elements enforce tangential continuity between element boundaries while leaving the normal component unconstrained. The lowest order $H(curl)$ elements are also known as edge elements, Nédélec elements, or Whitney forms. As with high-order bases for H^1 finite elements, there also are interpolatory [5] and hierarchical bases [11, 15] for $H(curl)$ finite elements. In this paper we assume (1.4) is discretized using a particular hierarchical form [11]. We give an overview of this basis in section 2.

There has been some success in designing multigrid methods for high-order $H(curl)$ discretizations, but few attempts based on algebraic multigrid (AMG) have been devised. A two-level additive Schwarz method [16] has shown to be successful for the

hierarchical basis [11], where the coarse level is the lowest order edge elements. Here, the coarsest level is solved either directly or by geometric $H(curl)$ multigrid methods [1, 7]. A two-level p-multigrid algorithm [12] for (1.4) using a different hierarchical basis [15] has also been investigated. The coarse grid is given by the lowest order edge elements, but the smoother used on each level is a Gauss–Seidel smoother, which limits the applicability to $p = 3$.

We assume that (1.4) is discretized by the hierarchical basis of [11] and present a fully multilevel solver. Our algorithm combines ideas from p multigrid and algebraic multigrid for lowest order edge elements. In particular, we define a new high-order discrete gradient operator, which allows hybrid smoothing and preserves the kernel of the curl operator on all levels via p-type coarsening of the hierarchical basis. Finally, we show promising numerical results for the method, and we highlight the main ingredients necessary for a successful method.

Our proposed method is an important step in the development of more robust multigrid methods for high-order curl problems. With a purely algebraic approach on the finest level, a coarse grid is not exposed to take advantage of the edge-based AMG solvers. With our high-order construction in particular and with the p-multigrid approach that we employ, the coarsest p-level in the hierarchy is open to edge-based AMG. As a result, another advantage of our approach is that we can scale the discretization with p and maintain a scalable multigrid method without relying on nonstandard coding components. That is, the proposed method is straightforward to integrate into existing multigrid solver packages. In summary, this approach represents a valuable step in multilevel development as we move toward an efficient high-order basis. And our approach to approximating the discrete gradient will be necessary for more complicated bases such as an interpolatory $H(curl)$ basis and a general $H(div)$ basis.

2. High-order basis. High-order finite element methods are popular for their improved convergence properties and increased data locality in comparison to low-order approximation methods. For curl-type problems, high-order $H(curl)$ bases have been developed in both interpolatory and hierarchical settings. Similar to interpolatory H^1 bases, interpolatory $H(curl)$ bases [5] specify the value of the tangential component at a set of points in the domain. Since the tangential component at the node of an element edge is not well defined, the points of interpolation are located on the interior of edges and on the interior of the element. One approach in defining a high-order $H(curl)$ basis is through the product of an interpolation polynomial (through the specified interpolation points) with the associated lowest order edge basis function. As with high-order H^1 bases, the matrix resulting from a basis with equispaced interpolation points suffers from poor conditioning, but the poor conditioning is remedied by basing the function on different interpolation points [8]. Another approach is to use a hierarchical basis. In a hierarchical basis, given a set of basis functions V^p for an order p approximation and a set V^q for order q with $p < q$, we have that $V^p \subset V^q$.

A hierarchical basis for $H(curl)$ was first defined in [15] and later in [11, 16] to satisfy the de Rham complex. The basis is composed of three types of basis functions: the lowest order edge functions, high-order edge functions, and high-order interior functions. The basis satisfies the de Rham complex by building from a hierarchical basis for H^1. To this end, the gradients of the H^1 basis are used to construct the $H(curl)$ basis, and thus naturally satisfy the complex. Since $\{u \,\|\, u = \nabla\phi,\ \phi \in H^1\} \subsetneq H(curl)$, it is necessary to enrich the $H(curl)$ basis with additional, linearly independent func-

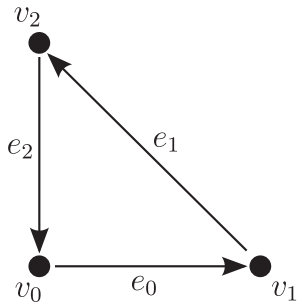


FIG. 2.1. *Reference element numbering of nodes and edges.*

tions. A benefit of this particular $H(curl)$ basis is that the gradients of the H^1 basis are directly represented, thus leading to a straightforward description of the (weak) kernel of $\nabla\times$.

In the following, we define basis functions according to the reference element shown in Figure 2.1.

In contrast to H^1 elements where the lowest order is $p = 1$—i.e., the linear elements—the lowest order $H(curl)$ conforming elements are with $p = 0$. For example, given a triangle (v_0, v_1, v_2), the lowest order $H(curl)$ basis function associated with edge $e_0 = (v_0, v_1)$ is given by the Whitney form

$$\phi_0(x, y) = \lambda_{v_0}\nabla\lambda_{v_1} - \lambda_{v_1}\nabla\lambda_{v_0}, \tag{2.1}$$

where λ_{v_i} denotes the barycentric coordinates or, in the case of a triangle whose endpoints are $(0, 0), (1, 0), (0, 1)$, the linear H^1 Lagrange basis function for node v_i. We classify the lowest order $H(curl)$ basis functions as $p = 0$ since the Whitney forms have constant tangential component along one edge and vanishing tangential components along all other edges. To define high-order basis functions, it is necessary to introduce Legendre, integrated Legendre, and scaled integrated Legendre polynomials [11, 14].

The Legendre polynomials are a class of orthogonal polynomials, defined on $[-1, 1]$, and are obtained efficiently by the following three term recurrence:

$$l_k(x) = \frac{2k-1}{k}xl_{k-1}(x) - \frac{k-1}{k}l_{k-2}(x), \qquad k \geq 2,$$

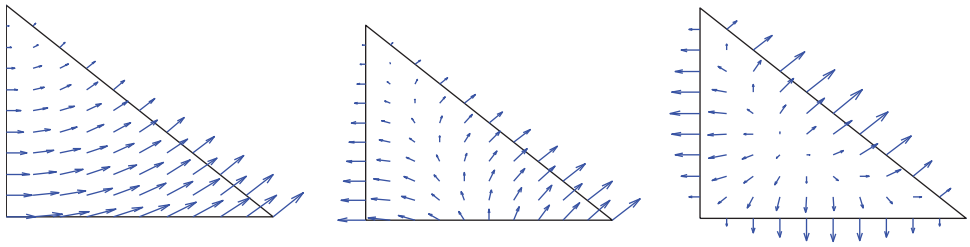
with $l_0(x) = 1$, and $l_1(x) = x$. Further, the *integrated* Legendre polynomials are obtained by integrating the Legendre polynomials over $[-1, x]$ so that

$$L_0(x) = \frac{1-x}{2}, \quad L_1(x) = \frac{x+1}{2}, \quad \text{and} \quad L_k(x) = \int_{-1}^{x} l_{k-1}(\xi)d\xi, \quad k \geq 2.$$

Aside from the normalization constant, which plays an important role in conditioning, the integrated Legendre polynomials are simply the Lobatto shape functions. In this paper we utilize the scaled integrated Legendre polynomials [11], defined as

$$L_k^S(s, t) = t^k L_k\left(\frac{s}{t}\right).$$

With these basis functions we complete the definition of the hierarchical basis. We begin with high-order edge basis functions. These functions are designed with a



(a) Whitney (b) Edge (c) Interior

FIG. 2.2. *Hierarchical $H(curl)$ basis functions.* (a) $p = 0$ *basis function (Whitney form) associated with edge* e_0. (b) $p = 1$ *basis function associated with edge* e_0. (c) $p = 2$ *interior basis function.*

tangential component that is polynomial along one edge and vanishing on the other two edges. For an order p basis, we have for each edge e_i,

$$E_{e_i,j}(x,y) = \nabla L^S_{j+2}(\lambda_{v_0} - \lambda_{v_1}, \lambda_{v_0} + \lambda_{v_1}), \qquad 0 \le j \le p-1. \tag{2.2}$$

A notable attribute of this subset of the hierarchical basis is that they are directly the gradients of high-order H^1 edge functions. Let $\mathcal{P}^p(\Omega)$ denote the set of polynomials of degree $\le p$ over a domain Ω. For a set of $p+1$ edge basis functions, the tangential components span $\mathcal{P}^p(e_i)$. An example of (2.2) is depicted in Figure 2.2(b).

Next, we define functions that span $\mathcal{P}^p(\tau)$, where τ is the interior of the element. To do this, we use interior bubble functions with vanishing tangential components on the edges. Notice that the functions

$$\begin{aligned} u_j(x,y) &= L^S_{j+2}(\lambda_{v_1} - \lambda_{v_0}, \lambda_{v_0} + \lambda_{v_1}), \\ v_k(x,y) &= \lambda_{v_2} l_k(2\lambda_{v_2} - 1) \end{aligned}$$

vanish on edges e_0 and e_2 for u_j, and on edge e_1 for v_k. The result is that the product $u_j v_k$ vanishes on all edges, yielding an interior bubble function. The following three interior basis functions are defined for $0 \le i + j \le p - 2$:

$$\begin{aligned} F^1_{i,j}(x,y) &= \nabla u_i v_j, \\ F^2_{i,j}(x,y) &= \nabla u_i v_j - \nabla v_j u_i, \\ F^3_j(x,y) &= (\lambda_1 \nabla \lambda_2 - \lambda_2 \nabla \lambda_1) v_j. \end{aligned}$$

Here $F^1_{i,j}$ represents the gradient of a corresponding H^1 basis function and $F^2_{i,j}, F^3_j$ are functions that are linearly independent from the rest of the basis. This set of functions forms a basis for a pth order finite element subspace of $H(curl)$ since there are $(p+1)(p+2)$ linearly independent functions, which is equal to the size of the space [16]. An example of an interior basis function is depicted in Figure 2.2(c).

3. Low-order edge AMG. Algebraic multigrid is often successful in solving scalar H^1 elliptic problems. Standard relaxation methods, such as Gauss–Seidel, attenuate high energy error in a few iterations; however, coarse grid correction is necessary to handle the remaining low energy error. For $H(curl)$-type problems, however, classic AMG is not competitive without modification. As seen in Figure 5.1 conjugate gradient alone and preconditioned conjugate gradient (smoothed aggregation AMG)

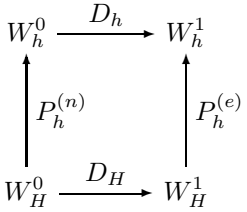


FIG. 3.1. *De Rham diagram for lowest order.*

do not perform well for high-order $H(curl)$ problems. Convergence is irregular and eventually stagnates even with preconditioning.

Since our algorithm builds upon the low-order edge algorithms [2, 4, 10] we give a brief overview. In this section we assume that (1.4) is discretized by the lowest order edge element or Whitney form. The main ingredients in defining a multigrid algorithm are the smoother and the interpolation between levels.

To define interpolation based on aggregation, we need an aggregation approach for edge-type degrees of freedom. There are several approaches to doing this, including a direct aggregation scheme; however, a successful process for this problem is to *induce* an edge aggregation based on an associated nodal aggregation [10]. From a given nodal aggregation, and hence a nodal interpolation operator $P_h^{(n)}$, an edge interpolation $P_h^{(e)}$ is defined so that the following diagram in Figure 3.1 commutes. Here, W^0 and W^1 are the finite element subspaces of H^1 (nodal) and $H(curl)$ (edge), respectively, and subscripts h and H are used to identify fine and coarse representations. We introduce the notation W^0 and W^1 for discrete nodal and edge spaces, respectively, since in the language of differential forms, they are 0-forms and 1-forms, respectively.

The discrete gradients D_h and D_H are a mapping between the W^0 and W^1 finite element spaces and are defined by a matrix in which each row represents an edge and each column a node. For example, if the nth edge is given by (v_i, v_j), then

$$D(n,i) = -1, \tag{3.1a}$$

$$D(n,j) = 1. \tag{3.1b}$$

From commutativity, $P_h^{(e)}$ is defined based on D_h, D_H, and $P_h^{(n)}$ so that

$$D_h P_h^{(n)} = P_h^{(e)} D_H. \tag{3.2}$$

The result is that the range of interpolation of a coarse grid gradient function through $P_h^{(e)}$ is in the space of fine grid gradient functions. That is, the kernel of the curl operator is preserved on both fine and coarse levels.

The edge aggregation operator is defined with respect to nodal aggregates. Therefore, in order to obtain the nodal aggregation, we perform smoothed aggregation on the associated H^1 problem,

$$\hat{a}(u,v) = \int_\Omega (\mu \nabla u \cdot \nabla v + \sigma uv)\, d\Omega, \tag{3.3}$$

to obtain a hierarchy of nodal interpolation operators.

Edge interpolation $P_h^{(e)}$ follows from $P_h^{(n)}$ by noticing that nodal aggregates induce edge aggregates. Denote by $\mathcal{A}_j$ the set of nodes in aggregate j. Then a coarse edge is

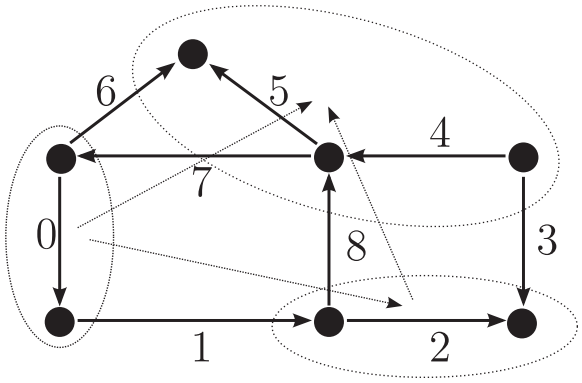


FIG. 3.2. *Example of $P_h^{(e)}$ operator. Nodal aggregates are contained within the dotted ellipses. The induced edges are represented by dotted edges.*

defined as an edge between two nodal aggregates (see Figure 3.2), and a coarse edge exists only if a fine edge exists that connects two nodes of separate aggregates. From this description of edge aggregation, we define interpolation $P_h^{(e)}$ from coarse edges to fine edges.

Formally, assuming $i = (i_1, i_2)$ is a fine edge and $j = (j_1, j_2)$ is a coarse edge, then the edge interpolation operator is defined as

$$P_h^{(e)}(i,j) = \begin{cases} 1, & \text{if } i_1 \in \mathcal{A}_{j_1} \text{ and } i_2 \in \mathcal{A}_{j_2}, \\ -1, & \text{if } i_2 \in \mathcal{A}_{j_1} \text{ and } i_1 \in \mathcal{A}_{j_2}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

For example, in Figure 3.2, the nodal aggregates are represented by dotted ellipses and the induced coarse grid edges are dotted lines.

The $P_h^{(e)}$ defined in (3.4) are piecewise constant. Although defining $P_h^{(e)}$ in this way ensures the commutativity of the diagram (see (3.2)), it does not yield an accurate interpolation operator. As a result, it has been demonstrated [4] that the convergence of the resulting method decreases as the mesh is refined and further shown that smoothing $P_h^{(e)}$ by one iteration of weighted Jacobi is effective. By doing so, the resulting method becomes less dependent on h. Moreover, it has also been shown that applying two smoothing iterations to the tentative prolongation operator only moderately increases the cycle complexity for this problem [2]. We therefore use two iterations of weighted Jacobi to smooth the tentative prolongator for the lowest order grids.

To complement interpolation, the relaxation scheme should attenuate high-energy error for $H(curl)$ problems, which is not accomplished by standard pointwise Gauss–Seidel due to the large kernel of the curl operator. The gradients of high frequency modes are also of high frequency resulting in gradient fields that are not reduced by pointwise Gauss–Seidel. As a result, a hybrid smoother [7] is used to specifically target these gradient modes.

4. High-order edge AMG. Because of the hierarchical nature of the basis defined in section 2, it is convenient to construct a multigrid hierarchy where coarse grids are low-order approximations of the fine grid, which is reminiscent of p-type multigrid. In order to fully retain a successful AMG method at a coarse level, the

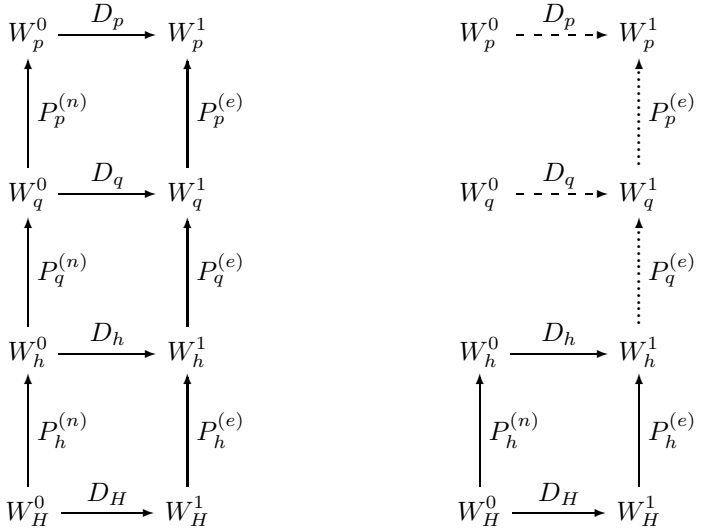


FIG. 4.1. *High-order complexes.* (a) *represents full complex and* (b) *represents partial complex.*

kernel of the curl operator should be preserved on the coarse grid. Thus, we extend this idea to high-order levels. For example, consider the diagrams in Figure 4.1.

In (4.1a), the hierarchy is simply extended from Figure 3.1 to high-order polynomial degrees q and p (with $q < p$). In order to satisfy (4.1a), high-order nodal and edge interpolation operators, along with discrete gradients D_q and D_p are required in the multilevel construction. A benefit of the hierarchical construction of the elements is that direct construction of $P_q^{(e)}$ and $P_p^{(e)}$ is straightforward. It is not necessary to aggregate on the auxiliary H^1 problem in order to aggregate in the $H(curl)$ space. Thus, in Figure 4.1b, nodal interpolation operators are not necessary to induce edge interpolation operators. Moreover, since gradient functions are also inherent in the description, a definition of D_q and D_p is possible *without* relating the high-order nodal spaces W_q^0 and W_p^0 to the low-order nodal elements in W_h^0. The high-order discrete gradient operators are necessary to the hybrid smoother, which we define later in this section.

Given an order p discretization of (1.4), we seek a coarse grid corresponding to an order q discretization where $q < p$. The interpolation operator $P_p^{(e)}$ between these levels is straightforward to define. Assuming a convenient ordering of the unknowns, the high-order $P_q^{(e)}$ is defined as

$$P_q^{(e)} = \begin{bmatrix} I_q \\ 0 \end{bmatrix}, \tag{4.1}$$

where I_q is the identity corresponding to the q degrees of freedom.

Because the coarse grid contains all degrees of freedom of a low-order discretization, the kernel of the curl operator is preserved on all levels. Correspondingly, we coarsen the problem until $p = 0$. A central part of our approach is that the low-order AMG algorithm detailed in section 3 is applicable to the lowest order level.

To account for the gradient functions, a hybrid Gauss–Seidel[1] smoother on the high-order levels is used. Similar to the low-order case, it is composed of one iteration

[1]It is possible to use other smoothers such as weighted Jacobi or polynomial smoothers for better parallelizability of the algorithm. For simplicity and consistency with [2, 7, 10] we use Gauss–Seidel.

of Gauss–Seidel followed by one iteration of Gauss–Seidel on the gradient space. But in order to perform relaxation on the gradient space, a high-order discrete gradient operator needs to be defined. Since the hierarchical basis explicitly includes gradients of H^1 functions, it is possible to define a high-order discrete gradient operator. We define the high-order discrete gradient

$$D_p = \begin{bmatrix} D_0 & 0 \\ 0 & \hat{D}_p \end{bmatrix}, \tag{4.2}$$

where D_0 is the lowest order discrete gradient operator for the finest mesh, as defined by (3.1), and $\hat{D}_p$ is a matrix mapping from the high-order H^1 basis to the high-order gradient basis functions in $H(curl)$ defined in the following.

Consider a basis W_p^0 with $W_h^0 \subset W_p^0$ and identify $\mathcal{N}^0$ as the set of indices associated with basis functions $\phi \in W_p^0 \backslash W_h^0$. Similarly, denote by $\mathcal{N}^1$ the set associated with basis functions $\psi \in W_p^1 \backslash W_h^1$. Then we define $\hat{D}_p$, a discrete gradient operator corresponding to high-order degrees of freedom, by identifying basis functions ψ_i for $i \in \mathcal{N}^1$ with $\nabla\phi_j$ for $j \in \mathcal{N}^0$. That is,

$$\hat{D}_p(i,j) = \begin{cases} 1, & \text{if } i \in \mathcal{N}^1, j \in \mathcal{N}^0, \text{ and } \psi_i = \nabla\phi_j, \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

To further motivate this representation of D_p in (4.2), we consider the first block, which is the lowest order discrete gradient operator D_0. Thus, we preserve the action of the discrete gradient operator for the lowest order degrees of freedom. For the degrees of freedom associated with higher order basis functions, recall from section 2 that the gradients of H^1 basis functions are explicitly used as basis functions for $H(curl)$. Thus, $\hat{D}_p$ is a mapping between the H^1 basis and their corresponding gradients in the $H(curl)$ space.

As a result of defining D_p as (4.2), we obtain a high-order discrete gradient operator and a mapping to project $K^{(e)}$ onto the gradient space. The construction of the high-order discrete gradient and high-order interpolation operators are detailed in Algorithm 1.

In this algorithm, let $\mathcal{G}$ denote the set of degrees of freedom that correspond to gradients. Our method utilizing Figure 4.1b is detailed in Algorithm 2. It is assumed that the high-order discrete gradients D and interpolation operators P are obtained from Algorithm 1.

The hybrid smoother used in Algorithm 2 is fully defined with our discrete gradient operator (4.3). Once the high-order discrete gradient operator is obtained, the hybrid smoother is invoked similarly at any order. In Algorithm 3, all Gauss–Seidel iterations are assumed to be symmetric unless specified otherwise. An important aspect of multigrid algorithms is $O(n)$ complexity. The amount of work per multigrid cycle is approximated by the cycle complexity, defined as

$$\mathbb{C}_{cycle} = \frac{\sum_{l=0}^{l_{max}} NNZ(A_l)\nu_l}{NNZ(A_0)}, \tag{4.4}$$

where ν_l is the number of smoothing iterations on level l. For the case of $V(1,1)$ cycling, we perform one presmoothing sweep and one postsmoothing sweep—i.e., $\nu_l = 2$ for all l. In order to keep the cycle complexity independent of the order of approximation, each p should not be visited in the hierarchy. Instead, we coarsen an order p discretization by choosing the coarse grid as an order $p/2$ discretization as detailed in the next section.

Algorithm 1. HO_Hierarchy($\mathcal{G}, levels, p, D_0$)

foreach $q \in levels$ **do**
  $D_q = \begin{bmatrix} D_0 & 0 \\ 0 & 0 \end{bmatrix}$
  $j = |W_0^0|$
  $k = 0$
  for $i = 0$ **to** $|W_p^1|$ **do**
    if $i \in \mathcal{G}$ **then**
      $(D_q)_{ij} = 1$
      $j = j + 1$
    if $i \in W_q^1$ **then**
      $(P_q^{(e)})_{ik} = 1$
      $k = k + 1$
  $\mathcal{G} = \mathcal{G} \setminus \left(\bigcup_{\psi_m \in W_p^1} m \setminus \bigcup_{\psi_n \in W_q^1} n \right)$ {remove high-order d.o.f}
  $p = q$
return D_q, $P_q^{(e)}$ for each $q \in levels$

Algorithm 2. HO_AMG($A, x, b, level$)

if $level = maxlevel$ **then**
  solve $Ax = b$ using direct method
if $level == HO$ **then**
  $x =$ hybrid_smooth(A, x, b, D_{level})
  $r_{fine} = b - Ax$
  $P = P_{level}$ {take HO P as tentative prolongator }
  $r_{coarse} = P^T r_{fine}$
  $A_{coarse} = P^T AP$
  $e_{coarse} = 0$
  HO_AMG($A_{coarse}, e_{coarse}, r_{coarse}$)
  $x = x + Pe_{coarse}$
  $x =$ hybrid_smooth($A, x, b, D[level]$)
else
  SA_EDGE_AMG(A, x, b)

Algorithm 3. hybrid_smooth(A, x, b, D)

$x =$ gauss_seidel(A, x, b)
$r = b - Ax$
$\hat{x} =$ gauss_seidel($D^T AD, 0, D^T r$)
$x = x + D\hat{x}$
$x =$ gauss_seidel(A, x, b)

5. Numerical results. In this section we provide numerical evidence in support of the multilevel approach developed in the previous section. One attractive practical property of the algorithm is the ease of integration into existing smoothed aggregation based codes; we implement our multilevel approach using the PyAMG package [3].

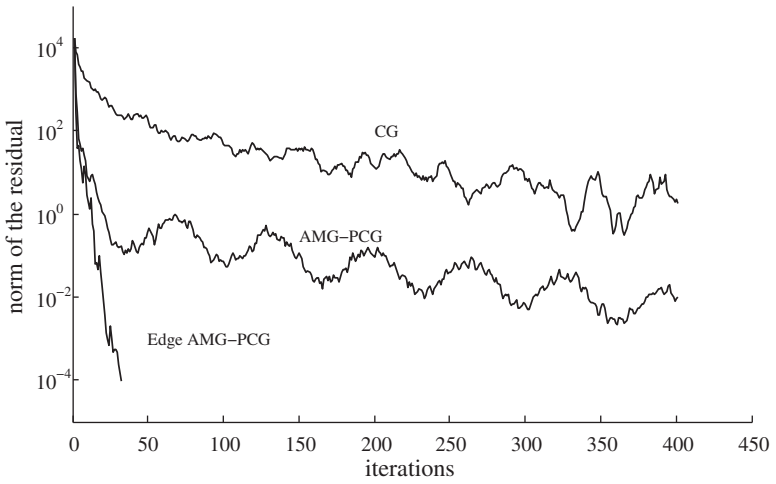


FIG. 5.1. *High-order edge AMG PCG compared with conjugate gradient and smoothed aggregation AMG PCG on a* $p = 8$ *discretization of* (1.4) *with* $\sigma = 10^{-2}$.

TABLE 5.1
Iteration counts for different smoothers on a mesh with 512 *elements and* $\sigma = 10^{-2}$.

$p =$	1	2	3	4	5	6	7	8	9
Hybrid	12	11	11	10	12	15	21	30	46
Pointwise GS	15	14	15	15	20	25	37	53	85

We consider (1.4) on a unit square $\Omega = [0,1] \times [0,1]$ discretized with high-order hierarchical finite elements. In all experiments we use our method to precondition conjugate gradient iterations and, unless otherwise noted, we iterate until the residual has been reduced by 10^8. To obtain the right-hand side we take a random vector x and multiply

$$K^{(e)}x = b.$$

First, we compare the residual history of the proposed preconditioner with standard methods in Figure 5.1. We notice that the residual history of our high-order edge-based AMG (labeled "Edge AMG-PCG") indicates an effective preconditioner. In this case, the elements are of very high order ($p = 8$) and the mesh is well refined (500+ elements), yet the residual is more consistently reduced and does not stagnate as in the case of standard AMG, wherein the gradient space is not specifically addressed.

An important component in achieving an efficient reduction of the residual as indicated in Figure 5.1 is the gradient specific smoother. The hybrid smoother specifically targets the gradient space on high-order and low-order levels. The effectiveness of the hybrid smoother is compared with pointwise Gauss–Seidel in Table 5.1. Pointwise Gauss–Seidel drops in performance as the order p increases, due to the enrichment of the near-null space with gradient functions. Hence, as p increases, there are more oscillatory components that pointwise Gauss–Seidel does not attenuate well, leading to a decrease in performance.

In addition to iteration count, we compare the cost of the hybrid smoother with that of pointwise Gauss–Seidel. Since the hybrid smoother performs smoothing on the gradient space, we compute the additional work performed during hybrid smoothing.

TABLE 5.2
Percent additional work ($\mathbb{W}_D$) in smoothing gradient space.

$p=$	1	2	3	4	5	6	7	8	9
	78.5	62.0	47.0	42.9	38.1	35.7	34.5	34.2	33.4

TABLE 5.3
PCG iteration count for different smoothers.

$p=$	1	2	3	4	5	6	7	8	9
Gradient space	13	12	12	11	15	19	25	38	60
Pointwise GS	15	14	15	15	20	25	37	53	85
Pointwise GS (2 iterations)	14	13	14	14	16	20	28	40	55

The additional work is proportional to the number of nonzeros in the gradient space. Thus, the additional work per multigrid cycle is obtained by summing the number of nonzeros in the gradient space over each level in the hierarchy, i.e.,

$$\mathbb{W}_D = \sum_i \frac{NNZ(D_i^T A_i D_i^T)}{NNZ(A_i)} \times 100. \tag{5.1}$$

The additional work is summarized in Table 5.2.

To compare the work of the hybrid smoother with that of pointwise Gauss–Seidel we use one iteration of *forward* Gauss–Seidel on A followed by one iteration of *symmetric* Gauss–Seidel on the gradient space $D^T AD$ completed by one iteration of *backward* Gauss–Seidel on A. Thus, the additional work of the hybrid smoother will only be a single iteration of symmetric Gauss–Seidel on the gradient space. We compare the effectiveness of smoothing on the gradient space with an additional iteration of smoothing on the entire matrix in Table 5.3. It is evident that the hybrid smoother is more effective than two iterations of pointwise Gauss–Seidel while performing less work.

Remark 1. Although pointwise Gauss–Seidel is not as effective as the hybrid smoother, it still attenuates gradient-like error. This is a result of the chosen basis since the gradient fields are explicitly defined as degrees of freedom. In contrast, when the gradient fields are not explicitly defined, such as in interpolatory bases, pointwise Gauss–Seidel is not as effective.

The hybrid smoother plays an important role in maintaining an efficient solver; moreover, it relies on our choice of the discrete gradient operator to mimic the process described in diagram (4.1b). For example, since gradient functions are only introduced for high-order elements, it is natural to consider a simpler view of the discrete gradient operator that accounts for only these functions. That is,

$$D_p' = \begin{bmatrix} 0 & 0 \\ 0 & \hat{D}_p \end{bmatrix}.$$

In Table 5.4 we compare this choice with the proposed high-order discrete gradient operator, D_p, in (4.2). Although the difference is not severe, D_p consistently outperforms D_p'. By including the D_0 in the $(1,1)$ block of D_p, D_p is a more accurate representation of a high-order discrete gradient operator. The additional cost in using D_p over D_p' is summarized in Table 5.5 by summing the additional number of nonzeros produced in the gradient space over all levels of the hierarchy. The additional cost is

TABLE 5.4
Iteration count for discrete gradient operators on a mesh with 512 *elements and* $\sigma = 10^{-2}$.

$p =$	1	2	3	4	5	6	7	8	9
D_p	12	11	11	10	12	15	21	30	46
D'_p	14	13	14	14	15	18	24	35	50

TABLE 5.5
Percent additional cost in using D_p *over* D'_p *on a mesh with* 512 *elements and* $\sigma = 10^{-2}$.

$p =$	1	2	3	4	5	6	7	8	9
Additional cost	47.4	24.7	10.7	7.5	4.5	2.6	1.8	1.6	1.2

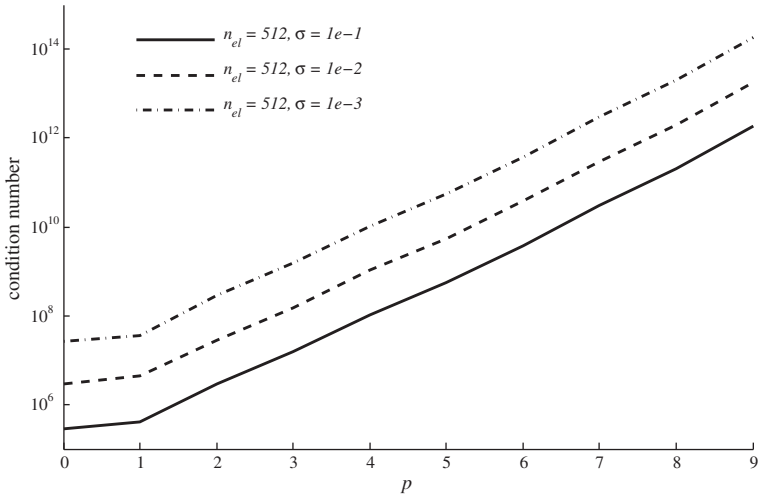


FIG. 5.2. *Growth in condition number of* $K^{(e)}$ *for different* σ *and number of elements,* n_{el}.

TABLE 5.6
Dependence on σ *using* 512 *elements.*

$p =$	1	2	3	4	5	6	7	8	9
$\sigma = 10^{-1}$	13	11	11	10	13	17	24	32	58
10^{-2}	12	11	11	10	12	15	21	30	46
10^{-3}	12	11	11	10	12	13	18	25	40

computed by comparing the total number of nonzeros present in the gradient space when using the two different discrete gradient operators. For $p < 3$, D_p performs more overall work. However, since the additional work in using D_p diminishes with increasing p, we see that for $p \geq 3$, choosing D_p indeed is more efficient.

As $\sigma \to 0$, the problem moves closer to a pure curl-curl problem with a rich gradient near-null space. As a result, conditioning degrades, as depicted in Figure 5.2. The condition number grows as $\sigma \to 0$ and increases strongly with p (exponentially).

Yet, our method exhibits only modest dependence on p and invariance with respect to typical choices of σ. Indeed, as detailed in Table 5.6, as $\sigma \to 0$, the iterations stay constant (or decrease). Since the contribution of the curl-curl stiffness term be-

TABLE 5.7
Dependence on h with $\sigma = 10^{-2}$ using smoothed / unsmoothed $P_h^{(e)}$ on low-order grids.

n_{el} =	32	128	512	2048	8192
$p = 1$	6 / 6	8 / 9	12 / 16	18 / 26	24 / 44
2	7 / 7	8 / 9	11 / 16	18 / 26	25 / 44
3	8 / 8	8 / 10	11 / 15	18 / 25	25 / 44
4	10 / 10	10 / 10	10 / 16	18 / 26	25 / 44
5	14 / 13	14 / 14	12 / 16	18 / 25	25 / 44

TABLE 5.8
Dependence on h with $\sigma = 10^{-2}$ on unstructured meshes.

n_{el}	110	440	1760
$p = 1$	7	13	20
2	8	13	20
3	8	13	20
4	10	12	20
5	14	14	20

TABLE 5.9
Iterations and cycle complexity for different coarsening schemes, 512 elements, $\sigma = 10^{-2}$.

	$p \to p-1$	$p \to \frac{p}{2}$	$p \to 0$
$p = 1$	12 / 2.60	12 / 2.60	12 / 2.60
2	11 / 2.66	11 / 2.66	12 / 2.15
3	10 / 3.03	11 / 2.26	12 / 2.06
4	9 / 3.46	10 / 2.50	12 / 2.03
5	9 / 3.90	12 / 2.27	14 / 2.16
6	12 / 4.31	15 / 2.35	17 / 2.01
7	15 / 4.79	21 / 2.23	24 / 2.01
8	21 / 5.30	30 / 2.36	32 / 2.00
9	29 / 5.71	46 / 2.25	60 / 2.00

comes more dominant, this indicates that high-order gradient smoothing is effective at targeting the near-null space.

Our method also exhibits robustness in p as the finite element mesh is refined. Table 5.7 shows a growth in h-refinement consistent with previous low-order results [10, 4]. Even at higher orders, the dependence on the mesh remains the same. In Table 5.7, we also show the benefit of a *smoothed* prolongation operator $P_h^{(e)}$. As in [4] we observe both accuracy and efficiency gains by using a smoothed prolongation operator on the lowest order grids. The dependence on h is compared for unstructured meshes in Table 5.8, and the dependence on h is similar to the structured case.

In the previous section, we advocate coarsening in p that reduces the coarse grid to $p/2$, for complexity reasons. In Table 5.9 we compare the convergence and cycle complexity of different coarsening schemes. If we coarsen to just $p-1$, then we obtain the best convergence rate out of the three methods; however, the cycle complexity increases with p. In contrast, by coarsening directly to $p = 0$, we obtain constant cycle complexity with respect to p; however, the convergence deteriorates in this case, particularly at the highest orders. Instead, if we coarsen to $p/2$, we find a balance between cycle complexity and convergence.

6. Conclusion. In this paper we have presented an algebraic multigrid algorithm that incorporates ideas from p-multigrid and edge-based smoothed aggregation algebraic multigrid to target high-order $H(curl)$ problems. An efficient solver for high-order $H(curl)$ problems is critical due to the high complexity of the problem and due to the poor performance of existing preconditioners. Standard edge-based smoothed aggregation AMG cannot be applied directly, but with the use of p-multigrid, we expose a coarse grid that is amenable to this solution approach. Through several key ingredients, we have designed a method that yields scalability in p similar to that of h refinement. Namely, we have devised discrete gradient operators for the high-order problem and a hybrid smoother that targets the high-order gradient space. The effectiveness of the proposed method was demonstrated through several numerical tests that highlight robustness for the eddy current problem.

REFERENCES

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

[2] N. Bell and L. Olson, *Algebraic multigrid for k-form Laplacians*, Numer. Linear Algebra Appl., 15 (2008), pp. 165–185.

[3] W. N. Bell, L. N. Olson, and J. Schroder, *PyAMG: Algebraic multigrid solvers in Python*, 2008. Version 1.0.

[4] P. Bochev, C. Garasi, J. Hu, A. Robinson, and R. Tuminaro, *An improved algebraic multigrid method for solving Maxwell's equations*, SIAM J. Sci. Comput., 25 (2003), pp. 623–642.

[5] R. Graglia, D. Wilton, and A. Peterson, *Higher order interpolatory vector bases for computational electromagnetics*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 329–342.

[6] J. Heys, T. Manteuffel, S. McCormick, and L. Olson, *Algebraic multigrid for higher-order finite elements*, J. Comput. Phys., 204 (2005), pp. 186–221.

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

[8] X. Lin, L. Olson, and J. Jin, *An interpolatory spectral element method using curl-conforming vector basis functions on tetrahedra*, Proc. IEEE Antennas Propagation Int. Symp., 2007.

[9] L. Olson, *Algebraic multigrid preconditioning of high-order spectral elements for elliptic problems on a simplicial mesh*, SIAM J. Sci. Comput., 29 (2007), pp. 2189–2209.

[10] S. Reitzinger and J. Schöberl, *An algebraic multigrid method for finite element discretizations with edge elements*, Numer. Linear Algebra Appl., 9 (2002), pp. 223–238.

[11] J. Schöberl and S. Zaglmayr, *High order Nédélec elements with local complete sequence property*, Int. J. Comput. Math. Electr. Electron. Engrg., 24 (2005), pp. 374–384.

[12] Y. Sheng, R. Chen, and X. Ping, *An efficient p-version multigrid solver for fast hierarchical vector finite element analysis*, Finite Elem. Anal. Des., 44 (2008), pp. 732–737.

[13] S. Shu, D. Sun, and J. Xu, *An algebraic multigrid method for higher-order finite element discretizations*, Computing, 77 (2006), pp. 347–377.

[14] P. Solin, K. Segeth, and I. Dolezel, *Higher-Order Finite Element Methods*, Chapman & Hall/CRC, London, 2004.

[15] J. Webb, *Hierarchal vector basis functions of arbitrary order for triangular and tetrahedral finite elements*, IEEE Trans. Antennas and Propagation, 47 (1999), pp. 1244–1253.

[16] S. Zaglmayr, *High Order Finite Element Methods for Electromagnetic Field Computation*, Ph.D. thesis, Johannes Kepler Universität Linz, Linz, Austria, 2006.