On Long Range Interpolation Operators for Aggressive Coarsening

Ulrike Meier Yang

September 30, 2009

Numerical Linear Algebra with applications

On Long Range Interpolation Operators for Aggressive Coarsening

Ulrike Meier Yang

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Box 808, L-560, Livermore, CA 94551

SUMMARY

Algebraic multigrid (AMG) is a very efficient scalable preconditioner for solving sparse linear systems on unstructured grids. Currently AMG solvers with good numerical scalability can still have larger than desired complexities, while variants with very low complexities exhibit decreased numerical scalability, which presents a problem for future high performance computers with millions of cores and decreased memory per core. It is therefore necessary to design more sophisticated interpolation operators to improve numerical scalability while preserving low memory usage. Two new long range interpolation operators to be used in combination with aggressive coarsening are presented. Their convergence and performance is examined and compared to multipass interpolation, the interpolation currently most commonly used with aggressive coarsening, and a higher complexity AMG variant. While the new interpolation operators require a more complex setup, leading to larger setup times, they exhibit better convergence than multipass interpolation, often resulting in better solve times. Copyright © 2000 John Wiley & Sons, Ltd.

1. Introduction

Sparse iterative linear solvers are an important part of many application codes, and can consume a significant portion of the total run time. Therefore efficient linear solvers are critical to enabling large-scale simulations on high performance computers. Algebraic multigrid (AMG) preconditioners have the potential to be weakly scalable on large number of processors, i.e. run times remain constant or close to constant if both the problem size and the number of processors are increased proportionally. Currently AMG solvers with good numerical scalability, i.e. the number of iterations stays constant while scaling up the problem, can still have larger than desired complexities, while convergence decreases for variants with very low complexities. However, with the prospect of large scale systems with millions of cores and decreasing memory per core, it is important to further decrease memory usage [4], while at the same time preserve numerical scalability.

There are several approaches on generating coarse level variables for AMG. Classical AMG [2, 3, 5, 12] coarsens by choosing a subset of the fine level variables, whereas aggregation-based AMG methods [13, 1] determine the coarse level variables by agglomerating several fine level variables, a method that often leads to very low complexities. A very effective tool to reduce computational complexities in classical AMG has been aggressive coarsening [12], however it has been difficult to generate an interpolation operator to deal effectively with the long distances between coarse grid points. Usually aggressive coarsening is combined with multipass interpolation, an interpolation based on direct interpolation, which is fairly simple, and often not very accurate, leading to decreased numerical scalability. While this interpolation can be improved when followed with one or two iterations of Jacobi iterations, this process can be expensive.

In this paper, we investigate ways on incorporating distance-two interpolation operators, which are of higher quality than direct interpolation, into multipass interpolation, and so to obtain long distance interpolation schemes with better convergence properties and numerical scalability than multipass interpolation. Two approaches are considered. The first one is an operator in the spirit of the original multipass interpolation, i.e. using various passes, but starting with a distance-two interpolation, and is denoted here as *distance-two multipass interpolation*. The second approach takes advantage of the stages of aggressive coarsening and consists effectively of the use of two different sets of coarse points, the evaluation of interpolation operators for different points, followed by a matrix multiplication to obtain the final interpolation operator. We will denote this interpolation *two stage interpolation*. We give a brief overview of AMG in Section 2, followed by the description of the coarsening algorithms (Section 3) and interpolation operators (Section 4) considered here. Section 5 contains comments on the complexities of the new algorithms Finally a variety of numerical experiments, on both sequential (Section 6) and parallel (Section 7) computers, are presented.

2. Algebraic Multigrid

In this section we give a brief outline of the basic principles and techniques that comprise AMG, and we define terminology and notation. Detailed explanations may be found in [11, 12, 3]. Consider a problem of the form

$$Au = f, \tag{1}$$

where A is an $n \times n$ matrix with entries a_{ij}. Furthermore, we assume that $a_{ii} > 0$. For convenience, the indices are identified with grid points, so that u_i denotes the value of u at point i, and the grid is denoted by $\Omega = \{1, 2, \ldots, n\}$. In any multigrid method, the central idea is that "smooth error," e, that is not eliminated by relaxation must be removed by coarse-grid correction. This is done by solving the residual equation $Ae = r$ on a coarser grid, then interpolating the error back to the fine grid and using it to correct the fine-grid approximation.

Using superscripts to indicate level number, where 1 denotes the finest level so that $A_1 = A$ and $\Omega_1 = \Omega$, AMG needs the following components: "grids" $\Omega_1 \supset \Omega_2 \supset \ldots \supset \Omega_M$, grid operators $A_1, \ldots, A_M$, interpolation operators P_k, restriction operators R_k (often $R_k = (P_k)^T$), and smoothers S_k, where $k = 1, 2, \ldots M-1$. These components of AMG are determined in a first step, known as the *setup phase*. During the setup phase, on each level k, $k = 1, \ldots, M-1$, Ω^{k+1} is determined using a coarsening algorithm, P_k and R_k are defined, and $A_{k+1} = R_k A_k P_k$.

Once the setup phase is completed, the *solve phase*, a recursively defined cycle, can be performed as follows, where $f^{(0)} = f$ and $u^{(0)}$ is an initial guess for u:

```
Algorithm: $MGV(A_k, R_k, P_k, S_k, u^{(k)}, f^{(k)})$.
    If $k = M$, solve $A_M u^{(M)} = f^{(M)}$.
    Otherwise:
        Apply smoother $S_k$ $\mu_1$ times to $A_k u^{(k)} = f^{(k)}$.
        Perform coarse grid correction:
            Set $r^{(k)} = f^{(k)} - A_k u^{(k)}$.
            Set $r^{(k+1)} = R_k r^{(k)}$.
            Set $e^{(k+1)} = 0$.
            Apply $MGV(A_{k+1}, R_{k+1}, P_{k+1}, S_{k+1}, e^{(k+1)}, r^{(k+1)})$.
            Interpolate $e^{(k)} = P_k e^{(k+1)}$.
            Correct the solution by $u^{(k)} \leftarrow u^{(k)} + e^{(k)}$.
        Apply smoother $S_k$ $\mu_2$ times to $A_k u^{(k)} = f^{(k)}$.
```

The algorithm above describes a V(μ_1, μ_2)-cycle; other more complex cycles such as W-cycles are described in [3]. In every V-cycle, the error is reduced by a certain factor, which is called the convergence factor. A sequence of V-cycles is executed until the error is reduced below a specified tolerance. For a scalable AMG method, the convergence factor is bounded away from one and independent of the problem size n, and the computational work in both the setup and solve phases is proportional to the problem size n.

3. Coarsening Algorithms

We briefly describe the two coarsening algorithms HMIS and PMIS [6], considered here, and the aggressive coarsening strategy[12], which can be applied to any coarsening algorithm, but here will be only applied to HMIS and PMIS.

One of the concepts used here and in the following sections is *strength of connection.* A point j strongly influences a point i or i strongly depends on j if

$$-a_{ij} > \alpha \max_{k \neq i}(-a_{ik}), \tag{2}$$

where $0 < \alpha < 1$. This definition was originally motivated by the assumption that the matrix A is a symmetric M-matrix, i.e. positive definite and off-diagonally nonpositive, it can however formally be applied to more general matrices. Note that with this definition positive F-to-F couplings are always treated as weak connections. Several approaches for special treatment of positive connections are presented in [12], however, these are not pursued here. In the remainder of the paper, $\alpha = 0.25$. All coarsening algorithms considered here proceed by first generating the directed graph of the strength matrix S with coefficients $s_{ij} = 1$ if point i strongly depends on point j and $s_{ij} = 0$ otherwise. The *measure* of a point i, λ_i, is defined as the number of points that strongly depend on i or, equivalently, the number of nonzero elements of the i-th column of the strength matrix S.

PMIS (short for Parallel Maximal Indepent Set algorithm) is based on a parallel algorithm proposed by Luby [10], which uses random numbers to find maximal independent sets. PMIS proceeds by first determining the measures λ_i. Then a random number between 0 and 1

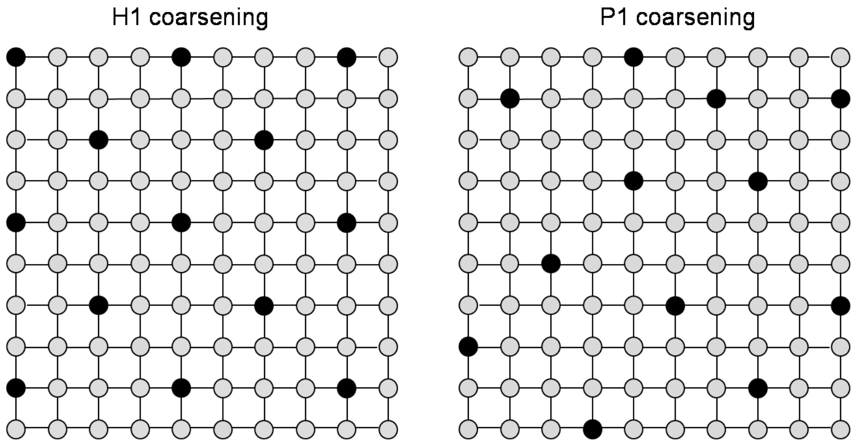

Figure 1. Examples of H1 (left) and P1 (right) coarsening applied to a 10 × 10 5-point 2D Laplace operator, black points are coarse points, grey points are fine points.

is added to each measure. This helps to distinguish neighbor points with equal numbers of strong connections and allows us to determine local maxima, a completely parallel process. All points with a local maximal measure now become coarse points (C-points) and make up the first independent set. Then those points that strongly depend on the new C-points are determined to be fine points (F-points). Both C- and F-points are eliminated from the graph, and the process proceeds for the remaining undetermined points until all points are either F- or C-points. Points with a measure smaller than 1 are determined to be F-points. For a more formal description of the PMIS algorithm and further details see [6]. Note that this algorithm is independent of the number of processors if the random numbers are chosen independently of the number of processors.

The second coarsening algorithm, HMIS (Hybrid Maximal Independent Set algorithm), which is a hybrid between PMIS and the classical coarsening algorithm, is not independent of p. Here the first pass of the classical coarsening algorithm, described in [11], is performed independently on each processor. Each point i is assigned a measure λ_i. Then a point with a maximal λ_i (there are usual several) is selected to be the first C-point. Now all points that strongly depend on i become F-points. For each point j that strongly influences one of these new F-points, λ_j is increased to improve j's chances to become a new C-point. This highly sequential process is repeated until all points are either F- or C-points. Now we use all interior C-points, i.e. those C-points that are not located on the processor boundaries, as the first independent set to be fed into the PMIS algorithm, which proceeds as described above.

In order to generate sparser coarse grids than those obtained by HMIS or PMIS, it is necessary to extend the definition of strong connectivity, as can be seen in [12] in the following way: A point i *strongly depends on* j *along a path of length* l if there exists a sequence of variables $i_0, i_1, \ldots, i_l$ with $i = i_0$ and $j = i_l$ such that i_k strongly depends on i_{k+1} for $k = 0, 1, \ldots l-1$. A point i *strongly depends on* j *w.r.t.* (p, l) if at least p paths of length $\leq l$ exist such that i strongly depends on j along each of these paths.

Aggressive coarsening proceeds by applying any coarsening algorithm to the graph of a strength matrix based on the new concept of strength. This matrix is just S^l. Note that the coefficients of S^l indicate the number of paths between i and j. In practice usually only the cases $l = 2$ and $p = 1$ or $p = 2$ are considered, and we will focus here on $p = 1$, and $l = 2$, which leads to sparser coarse grids than choosing $p = 2$. Since applying the coarsening algorithm to S^2 can be expensive, it is more efficient to implement aggressive coarsening in two stages. In the first stage one applies a coarsening algorithm (here PMIS or HMIS) to S and receives a set of coarse points C, which is larger than desired. In the second stage the coarsening algorithm is applied to points in C only, the measures now being defined as the number of all points i in C that strongly depend on j w.r.t. (1,2), or, equivalently, to the matrix S^2, restricted to all variables belonging to C.

For $p = 1$, aggressive coarsening with HMIS coarsening is denoted as H1 coarsening and with PMIS coarsening as P1 coarsening. H1 coarsening performed on one processor is equivalent to A1 coarsening as described in [12]. Figure 1 shows both H1 and P1 coarsening applied to a 10×10 5-point 2D Laplace problem. In general, P1 coarsening leads to slightly coarser grids than H1 coarsening. It is important to mention that when aggressive coarsening is implemented in two stages, certain special cases need to be treated differently in the second stage. For example, while it makes sense that for PMIS a point that has no connections to any other points is turned into a fine point, this is the wrong strategy now that we are actually dealing with C-points, since eliminating such a point might lead to the elimination of a whole area of points that depend on the eliminated C-point. Therefore such a point stays a C-point in the second stage of aggressive coarsening.

4. Interpolation Operators

Before we describe various interpolation operators that will be used in the remainder of the paper, we will define the following sets:

$$\begin{aligned} N_i &= \{j \neq i | a_{ij} \neq 0\}, \\ S_i &= \{j \in N_i | j \texttt{ strongly influences i}\}, \\ F_i^s &= F \cap S_i, \\ C_i^s &= C \cap S_i, \\ N_i^w &= N_i \setminus (F_i^s \cup C_i^s). \end{aligned}$$

Based on assumptions on small residuals for smooth error [2, 3, 11, 12], an interpolation formula can be derived as follows. The assumption that algebraically smooth error has small residuals after relaxation

$$A\,e \approx 0,$$

can be rewritten as

$$a_{ii}e_i \approx -\textstyle\sum_{j \in N_i} a_{ij}e_j,$$

or

$$a_{ii}e_i \approx -\textstyle\sum_{j \in C_i^s} a_{ij}e_j - \sum_{j \in F_i^s} a_{ij}e_j - \sum_{j \in N_i^w} a_{ij}e_j.$$

From this expression, various interpolation formulae can be derived. We use the terminology of [12] for the various interpolation strategies.

We now introduce various interpolation operators that will be used in the remainder of the paper.

4.1. Direct Interpolation

The so-called ‘direct interpolation’ strategy [12] has one of the most simple interpolation formulae.

$$w_{ij} = -\frac{a_{ij}}{a_{ii}} \frac{\sum_{k \in N_i} a_{ik}}{\sum_{k \in C_i^s} a_{ik}}, \quad j \in C_i^s, \tag{3}$$

where C_i^s is the coarse interpolatory set.

4.2. Distance-Two Interpolation Operators

We describe two distance-two interpolation operators, standard interpolation and extended+i interpolation.

For standard interpolation (std) [12], the stencil obtained through (3) is extended via substitution of every e_j with $j \in F_i^s$ by $1/a_{jj} \sum_{k \in N_j} a_{jk} e_k$. This leads to the following formula

$$\hat{a}_{ii} e_i + \sum_{j \in \hat{N}_i} \hat{a}_{ij} e_j \approx 0 \tag{4}$$

with the new neighborhood $\hat{N}_i = N_i \cup \bigcup_{j \in F_i^s} N_j$ and the new coarse point set $\hat{C}_i = C_i^s \cup \bigcup_{j \in F_i^s} C_j^s$. This can greatly increase the size of the interpolatory set.

Standard interpolation is now defined by applying direct interpolation to the new stencil, leading to

$$w_{ij} = -\frac{\hat{a}_{ij}}{\hat{a}_{ii}} \frac{\sum_{k \in \hat{N}_i} \hat{a}_{ik}}{\sum_{k \in \hat{C}_i} \hat{a}_{ik}}. \tag{5}$$

Another distance-two interpolation, which is an extension of the classical interpolation described in [11] with a slight modification, was introduced in [7]. It differs from a straightforward extension by using not only connections a_{jk} from strong fine neighbors j of i to points k of the interpolatory set but also connections a_{ji} from j to i itself and is therefore named extended+i interpolation (ei). It is given by the following formula:

$$w_{ij} = -\frac{1}{\tilde{a}_{ii}} \left(a_{ij} + \sum_{k \in F_i^s} a_{ik} \frac{\bar{a}_{kj}}{\sum_{l \in \hat{C}_i \cup \{i\}} \bar{a}_{kl}} \right), \quad j \in \hat{C}_i, \tag{6}$$

with

$$\tilde{a}_{ii} = a_{ii} + \sum_{n \in N_i^w \setminus \hat{C}_i} a_{in} + \sum_{k \in F_i^s} a_{ik} \frac{\bar{a}_{ki}}{\sum_{l \in \hat{C}_i \cup \{i\}} \bar{a}_{kl}}, \tag{7}$$

and $\bar{a}_{ij} = a_{ij}$ if $a_{ij} > 0$ and $\bar{a}_{ij} = 0$ otherwise, i.e. positive coefficients are ignored, a modification suggested in [8] to avoid very small numbers or cancellations in the denominator.

4.3. Multipass Interpolation

Since any of the previously mentioned interpolation schemes require coarse grids with coarse points that are not too far apart (and therefore are not adequate for use with aggressive coarsening), in [12], a further reaching interpolation, denoted as multipass interpolation (mp), was developed. It proceeds as follows:

1. Use direct interpolation for all F-points i, for which $C_i^s \neq \emptyset$. Place these points in set F^*.
2. For all $i \in F \setminus F^*$ with $F^* \cap F_i^s \neq 0$, replace, in Equation (3), for all $j \in F_i^s \cap F^*$, e_j by $\sum_{k \in C_j} w_{jk} e_k$, where C_j is the interpolatory set for e_j. Apply direct interpolation to the new equation. Add i to F^*. Repeat step 2 until $F^* = F$.

Figure 2a. shows the various passes for multipass interpolation as illustrated for a 8 × 8 5-point 2D Laplace problem for H1 (or A1) coarsening, and Figure 3a. shows them for P1 coarsening. Points marked i are processed in the i-th pass. For both examples, three passes are required to generate the final interpolation operator. It is certainly possible to have more than 3 passes. Larger numbers can for example occur when increasing the strength threshold α, leading to larger numbers of weak connections. For the example chosen here the set of variables belonging to all fine points can be divided into three sets $F = F^1 \cup F^2 \cup F^3$, where F^i consists of the variables belonging to the points marked i. We define operators P_i by applying direct interpolation to F^i and treating their range as a set of coarse points, (I is the identity):

$$\begin{array}{rcl} I \;:\; C & \rightarrow & C \\ P_1 : \; C & \rightarrow & F^1 \\ P_2 : F^1 & \rightarrow & F^2 \\ P_3 : F^2 & \rightarrow & F^3 \end{array}$$

If we assume that the variables are ordered in the following way, first the coarse points C followed by F^1, F^2, and then F^3, then multipass interpolation can be presented for this example as the following operator:

$$P_{mp} = \begin{pmatrix} I \\ P_1 \\ P_2P_1 \\ P_3P_2P_1 \end{pmatrix}. \tag{8}$$

Multipass interpolation is fairly cheap and can be efficiently implemented. However, it is not very powerful, since it is based on direct interpolation. This leads very often to only constant interpolation, and to overall deteriorated convergence and scalability, For an example see Figure 4, where many fine points are interpolated by one C-point only.

4.4. Distance-Two Multipass Interpolation

Our goal is to incorporate an interpolation of higher quality, such as a distance-two interpolation operator, into multipass interpolation. If we look at the examples in Figure 1, we see that many coarse points are distance 4 away from each other. Therefore the set of points that are connected to an element of F^1, i.e. an immediate neighbor of a C-point, by a path of

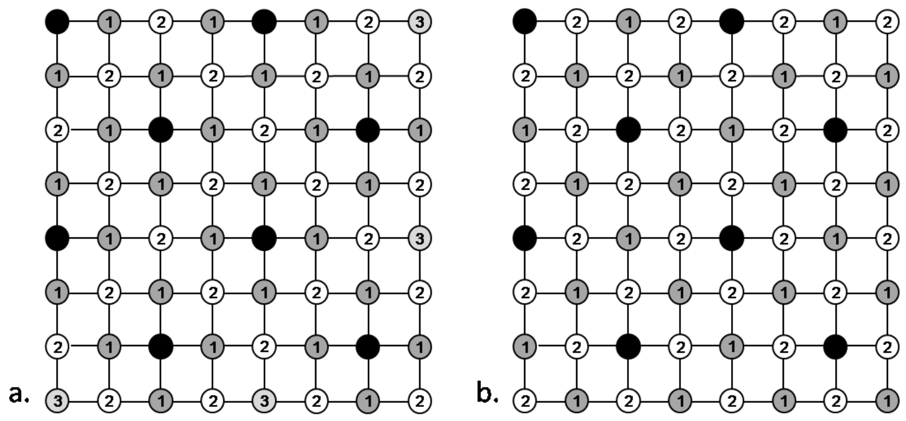

Figure 2. Processing order of F-points for various long range interpolation operators (a. multipass interpolation, b. distance-two multipass interpolation, two-stage interpolation) for H1 coarsening applied to an 8×8 5-point 2D Laplace operator, black points are coarse points, points marked with i are processed in i-th pass.

length <=2 contains generally only one C-point. Applying a distance-two interpolation to such a point will therefore not improve the situation. It is better to use distance-two interpolation for points that are distance two away from the C-points, e.g. points in F^2, since they often have several C-points in their distance-two neighborhood.

There are several ways to choose the points for the first pass. A first option is to use the passes generated in the original multipass interpolation algorithm, see Figures 2a. and 3a., and start with all points that are processed in the second pass of multipass interpolation, i.e. points in F^2. Since they are processed first in the new algorithm, they are now marked '1', see Figures 2b. and 3b. A second option is to start with all points that are strongly influenced by a C-point along a path of length 2. For the example in Figure 2b., both approaches generate the same set of points. However in the example in Figure 3c., the second strategy generates more points for the first pass and leads to less points interpolated by one C-point only, see Figure 4c., than the first strategy, see Figure 4b. Since the first strategy can be implemented significantly more efficiently, and our experiments showed no significant difference in convergence for both versions, we consider only the first option in the remainder of the paper.

After applying distance-two interpolation to the first set of points and so completing the first pass, the algorithm proceeds by (possibly several times) performing step 2 of multipass interpolation, using now direct interpolation, until all remaining points are processed. We will denote this interpolation by distance-two multipass interpolation (d2-mp).

To facilitate comparison between d2-mp and mp, we use the order of variables and definition of sets of Section 4.3. $Q : C \rightarrow F^2$ is a distance-two interpolation operator, and $P_0 : F^2 \rightarrow F^1$ is the direct interpolation operator that interpolates variables in F^1 from variables in F^2. Then

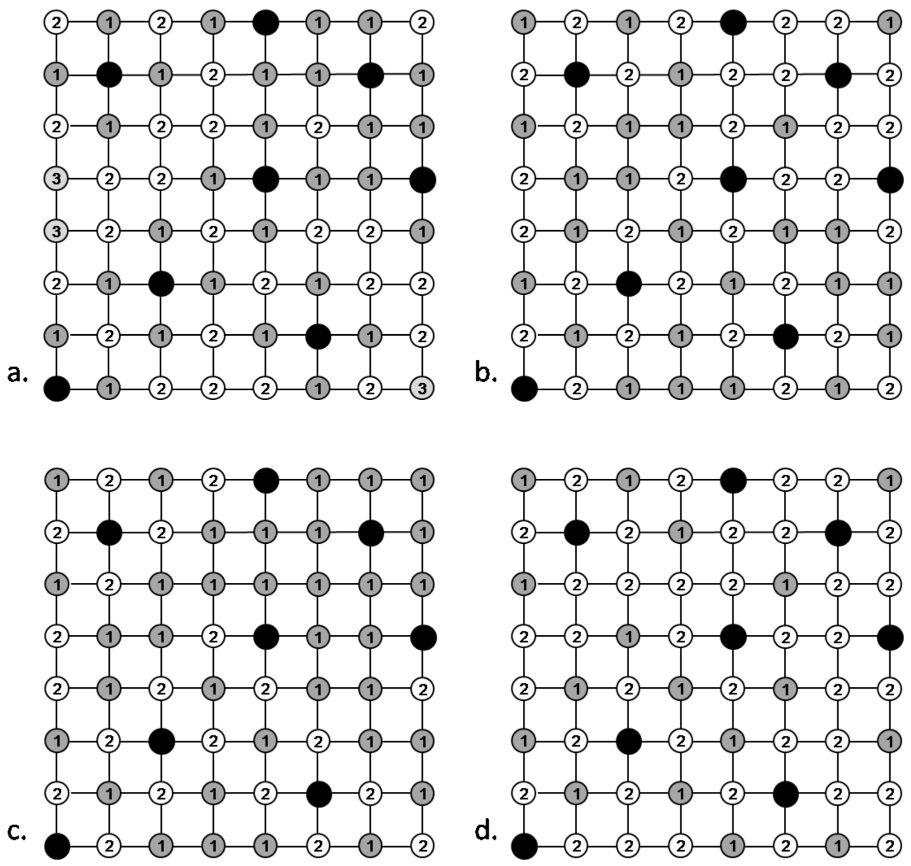

Figure 3. Processing order of F-points for various long range interpolation operators (a. multipass interpolation, b. distance-two multipass interpolation, option 1, c. distance-two multipass interpolation, option 2, d. two-stage interpolation) for P1 coarsening applied to an 8×8 5-point 2D Laplace operator, black points are coarse points, points marked with i are processed in i-th pass.

the new interpolation for the examples in Figure 1 is given by

$$P_{d2-mp} = \begin{pmatrix} I \\ P_1 + P_0 Q \\ Q \\ P_3 Q \end{pmatrix}, \tag{9}$$

Table I presents two-level convergence factors for a few Laplace problems (ei-mp stands for distance-two multipass interpolation using extended+i interpolation and std-mp uses standard interpolation for the distance-two interpolation operator). There is clearly an improvement for this new interpolation operator over multipass interpolation when using H1 coarsening and a 5-point or a 7-point stencil, however the difference becomes very small when using P1 coarsening or larger stencils. When investigating the quality of the new interpolation on the P1 grid, we find still a fairly large amount of points interpolated by one C-point only, see Figure 4b, albeit

Coarsening	Problem	mp	ei-mp	std-mp	2s-ei	2s-std
H1	5pt 2D, 32×32	0.45	0.35	0.35	0.35	0.35
	9pt 2D, 32×32	0.58	0.54	0.49	0.49	0.43
	7pt 3D, 10×10×10	0.37	0.22	0.23	0.22	0.23
	27pt 3D, 10×10×10	0.47	0.45	0.42	0.37	0.36
P1	5pt 2D, 32×32	0.69	0.68	0.68	0.56	0.55
	9pt 2D, 32×32	0.65	0.61	0.61	0.49	0.53
	7pt 3D, 10×10×10	0.47	0.41	0.41	0.35	0.35
	27pt 3D, 10×10×10	0.39	0.37	0.34	0.31	0.29

Table I. Two grid convergence factors for various problems using Gauss-Seidel as a smoother

significantly less than for multipass interpolation.

4.5. Two Stage Interpolation

To further improve convergence we decided to completely eliminate the use of direct interpolation and develop a long range interpolation operator suited for aggressive coarsening that uses distance-two interpolation only. Since distance-two interpolation was designed to work well with both HMIS and PMIS, it appears that just as aggressive H1 and P1 coarsening consists of two stages of HMIS and PMIS coarsening, we should be able to take advantage of the results of the two coarsening stages. A good set to use in a first pass appear to be the C-points that are discarded after the first coarsening stage. Let us denote the set of variables associated with these points by F^C. This leads us to three sets of variables, F^C, F^F, the set of variables associated with the remaining F-points, and C, where the complete set of fine points is $F = F^F \cup F^C$, and $C \cup F^C$ is the set of coarse points generated during the first stage.

Since the points in F^C are distance two away from points in C by construction, we can now apply a distance-two formula for these points interpolating from points in C obtaining interpolation operator $Q_1 : C \rightarrow F^C$. Next we apply a distance-two formula for all points in F^F, but now interpolating from $C \cup F^C$, leading to the interpolation operator $Q_2 : C \cup F^C \rightarrow F^F$. Q_2 is essentially the distance-two interpolation operator that interpolates from a coarse grid obtained by just applying HMIS or PMIS to the original grid. Q_2 can be decomposed into the two operators $Q_2^C : C \rightarrow F^F$ and $Q_2^F : F^C \rightarrow F^F$, $Q_2 = (Q_2^C \; Q_2^F)$, leading to the final interpolation operator

$$P_{2s} = \begin{pmatrix} I \\ Q_1 \\ Q_2^C + Q_2^F Q_1 \end{pmatrix}, \tag{10}$$

assuming the variables are ordered with C first, followed by F^C, and F^F last.

Figure 4d shows that for this interpolation significantly less points are interpolated by one C-point only, and convergence also improves for all test problems in Table I. (2s-ei denotes two stage interpolation using extended+i interpolation, 2s-std uses standard interpolation).

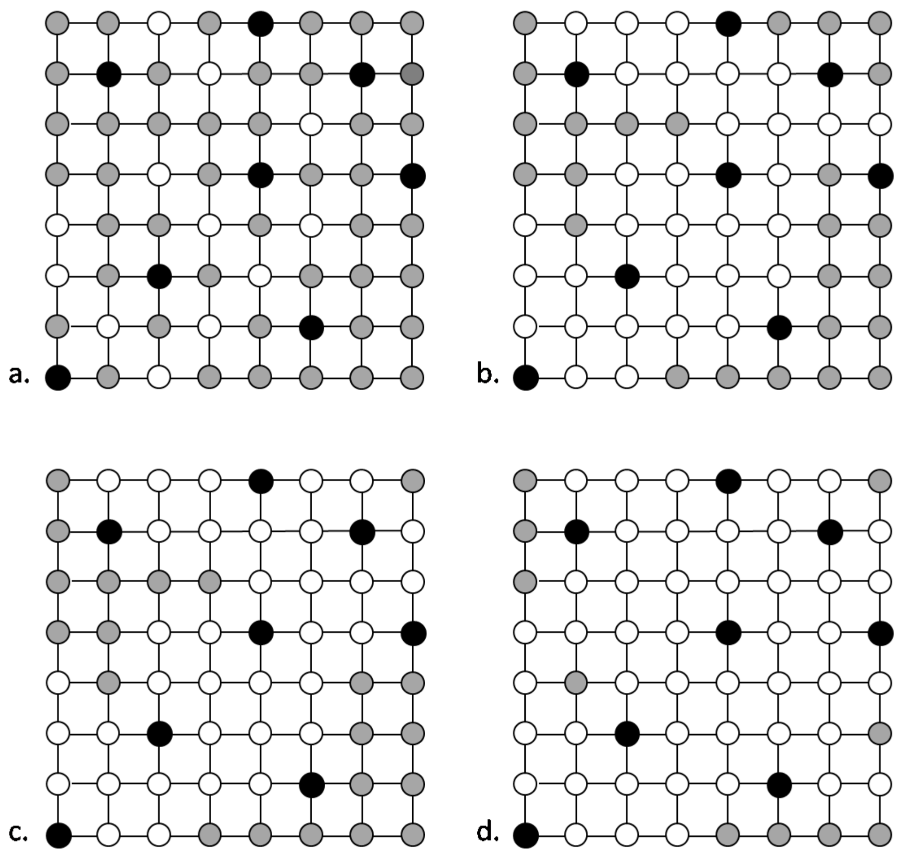

Figure 4. Gray points are interpolated by one C-point only for a. multipass interpolation, b. distance-two multipass interpolation, option 1, c. distance-two multipass interpolation, option 2, d. two-stage interpolation.

5. Some Comments on Complexity

While our preliminary two grid convergence tests show that the new interpolation operators lead to improved convergence, we have not addressed their actual computational complexities yet. Clearly, distance-two interpolation require more operations than direct interpolation, see [7] for the detailed analysis of these methods. Consequently generating distance-two multipass interpolation or stage two interpolation is more expensive than multipass interpolation. Comparing the operators P_{mp} in (8) and P_{d2-mp} in (9) it is obvious that the first pass in multipass interpolation (P_1) is cheaper than the second pass in distance-two multipass interpolation ($P_1 + P_0Q$), since the number of interpolatory points increases when applying Q to the first pass, leading to larger complexities. Two stage interpolation requires the use of distance-two interpolation for all fine points and requires an additional matrix-matrix multiplication to obtain the final operator. Depending on the stencil sizes of the matrices

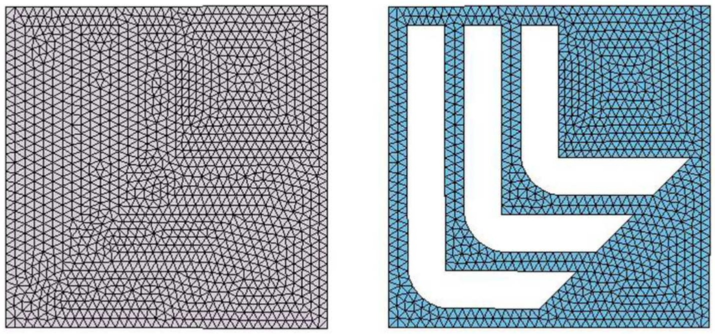

Figure 5. Grid used in Problem 4

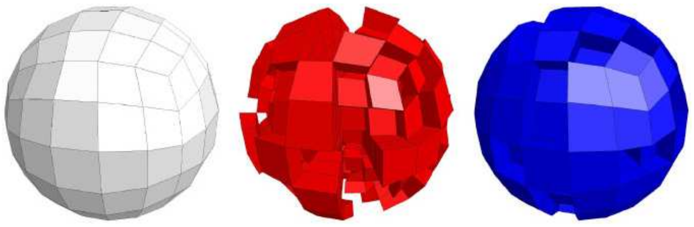

Figure 6. Grid used in Problem 7

involved this can be very expensive. In [7] large complexities in distance-two interpolation were controlled by truncation. It makes sense to use truncation here also.

There are various ways to use truncation here. One can compute the complete interpolation operator and truncate at the end. Another option is to use a truncated distance-two interpolation within the new interpolation operators. We only examined this for stage two interpolation, where this can lead to much smaller setup times and a significantly cheaper matrix-matrix multiplication. Of course for this to be efficient it is necessary that the truncated operators are of sufficiently high quality to not deteriorate convergence too much.

In order to get some indication of computational complexity as well as memory usage when using these interpolation operators within AMG, we will present for every problem considered here the resulting *operator complexity*, which is defined as $C_{op} = \sum_{i=1}^{m} nnz(A_i)/nnz(A_1)$ ($nnz(A)$ denotes the number of nonzeroes of the matrix A).

6. Numerical Experiments on one Processor

In this section, we present results obtained when applying the various interpolation operators to a few test problems. The goal was to test both two-dimensional as well as three-dimensional problems, and structured problems generated using finite differences as well as unstructured problems using finite elements. We considered standard Laplace problems, problems with rotated anisotropies as well as diffusion problems with jumps. The unstructured problems were generated with the unstructured finite element package aFEM. For all problems we used AMG as a preconditioner for conjugate gradient with symmetric Gauss-Seidel as a smoother. Convergence was halted when a convergence tolerance of 10^{-8} was reached. We used a modification of the code BoomerAMG in the hypre software library [9].

We consider the following problems:

Problem 1 is the 2 dimensional 5-point Laplace problem with Dirichlet boundary conditions on a 1000×1000 uniform grid.

Problem 2 is a 2-dimensional rotated anisotropic problem on a uniform 1000×1000 grid: $-(c^2 + \epsilon s^2)u_{xx} + 2(1-\epsilon)scu_{xy} - (s^2 + \epsilon c^2)u_{yy} = 1$ where $c = \cos\gamma$, $s = \sin\gamma$, and $\epsilon = 0.01$. Here $\gamma = 45^0$.

Problem 3 is the partial differential equation given in Problem 2, but now $\gamma = 60^0$.

Problem 4 is the diffusion problem $-(a(x,y)u_x)_x - (a(x,y)u_y)_y = f$ with Dirichlet boundary conditions on an unstructured square using triangular finite elements shown in Figure 5 with 1,383,617 degrees of freedom. There are four regions as shown in the right figure with $a(x,y) = 1$ on the three inner domains and $a(x,y) = 0.001$ on the outer domain.

Problem 5 is a 3-dimensional 7-point Laplace problem with Dirichlet boundary conditions on a 100×100×100 uniform grid.

Problem 6 is a diffusion problem with jumps on a 100×100×100 uniform grid. The domain is the unit cube. The partial differential equation is $-(a(x,y,z)u_x)_x - (a(x,y,z)u_y)_y - (a(x,y,z)u_z)_z = f$ with Dirichlet boundary conditions, where $a(x,y,z) = 1000$ for $0.1 < x,y,z < 0.9$, $a(x,y,z) = 0.1$ for $0 < x,y,z < 0.1$ and the cubes of size $0.1 \times 0.1 \times 0.1$ that are located at the corners of the domain, and $a(x,y,z) = 1$ elsewhere.

Problem 7 is a diffusion problem with jumps on an unstructured sphere using tetrahedral final elements divided in two domains as illustrated in Figure 6 with 453,001 degrees of freedom. Its PDE is given in Problem 6 with $a(x,y,z) = 1$ on one domain and $a(x,y,z) = 1000$ on the other domain.

Table II lists the number of iterations and the operator complexities for these problems. We include results for an AMG variant without aggressive coarsening. For interpolation we used extended+i interpolation truncated to 4 elements per row, denoted with 'ei(4)', because it in general achieves the best performance with HMIS and PMIS coarsening (see [7]).

Although in practice aggressive coarsening is generally used on only one or two levels, we will also present results using aggressive coarsening on all levels. While it is known that convergence decreases significantly for aggressive coarsening with multipass interpolation when used on more than one or two levels, we expect the new interpolation operators to perform better, and a comparison of these methods on all levels is at least of theoretical interest. Also, on future architectures with millions of processors, which will allow to solve even larger systems, leading to an increasing number of levels, it might be desirable to use aggressive coarsening on more than one or two levels for better performance. Therefore Table II contains also results for aggressive coarsening on all levels using multipass, ei-mp, and 2s-ei interpolation. as well as results with aggressive coarsening on one level only, denoted with 'mp-1' for multipass interpolation, and '2s-ei(4)-1' for 2s-ei interpolation truncated to at most 4 elements per row, to control complexities and setup times. On the levels without aggressive coarsening, ei(4) is used as the interpolation operator. While we also tested std-mp and 2s-std interpolation, we did not include the results here, since (except for Problem 2 with H1 coarsening) ei-mp and

Coarsening	Problem	ei(4)	mp	ei-mp	2s-ei	mp-1	2sei(4)-1
H1, HMIS	1	8(2.72)	46(1.26)	17(1.24)	9(1.26)	15(1.61)	8(1.41)
	2	7(2.59)	47(1.36)	14(1.42)	11(1.42)	15(1.99)	11(1.95)
	3	9(3.20)	72(1.21)	15(1.31)	14(1.31)	30(1.31)	13(1.55)
	4	10(2.58)	96(1.13)	41(1.31)	19(1.49)	23(1.28)	17(1.44)
	5	7(3.26)	24(1.22)	9(1.42)	8(1.45)	14(1.37)	8(1.58)
	6	8(3.27)	33(1.23)	15(1.42)	10(1.44)	15(1.38)	11(1.55)
	7	10(2.22)	26(1.07)	20(1.09)	13(1.11)	17(1.09)	12(1.08)
P1, PMIS	1	9(2.40)	88(1.16)	65(1.19)	27(1.27)	23(1.29)	15(1.37)
	2	9(2.02)	80(1.28)	59(1.33)	28(1.33)	24(1.47)	16(1.51)
	3	16(2.61)	124(1.13)	97(1.15)	44(1.23)	50(1.22)	28(1.33)
	4	10(1.87)	115(1.09)	78(1.13)	36(1.18)	26(1.15)	19(1.23)
	5	10(2.75)	31(1.18)	27(1.23)	14(1.46)	19(1.32)	13(1.45)
	6	11(2.85)	40(1.21)	36(1.29)	19(1.57)	20(1.37)	14(1.51)
	7	10(1.95)	29(1.04)	25(1.06)	16(1.11)	18(1.07)	13(1.09)

Table II. No. of iterations (C_{op} for AMG-CG using various interpolation operators on various test problems using H1 or P1 coarsening on levels with aggressive coarsening and HMIS or PMIS on levels without

2s-ei interpolation performed at least as well.

The results show that generally H1 (or HMIS) coarsening leads to better convergence than P1 (or PMIS), but also has higher complexities. Without aggressive coarsening one achieves the lowest number of iterations but also the highest complexities, which are often twice as large. When using aggressive coarsening on all levels, as expected, multipass interpolation converges the slowest, ei-mp shows better convergence, and 2s-ei exhibits the best convergence. When using H1 coarsening, the convergence for 2s-ei is often only about 50% slower than that of ei(4) with significantly lower complexities and is always better than mp-1. However its convergence deteriorates when using P1 coarsening. Using aggressive coarsening on one level only improves convergence for both mp and 2s-ei, with 2sei(4)-1 converging faster than mp-1.

Since numbers of iterations and operator complexities do not necessarily reflect the actual run times, those are listed in Table III. While 2sei(4)-1 exhibits better total times than mp-1 for the structured two dimensional problems, somewhat lower complexities and much more efficient setup lead to total times that are only slightly larger and sometimes even better for the remaining problems.

7. Parallel Experiments

In this section, we examine parallel performance of the previous algorithms. We are both interested in run times as well as weak scalability. While in the previous section we saw that most often H1 coarsening led to better results than P1 coarsening, we need to remember that on one processor H1 coarsening is just A1 coarsening and completely sequential. If running truly in parallel, H1 coarsening is a hybrid of A1 coarsening and P1 coarsening, and one might expect its performance to be worse than that of A1 coarsening.

Coarsening	Problem	ei(4)	mp	ei-mp	2s-ei	mp-1	2sei(4)-1
H1, HMIS	1	9.7	20.8	10.2	7.0	9.8	6.8
	2	9.3	25.9	10.5	8.8	12.8	10.6
	3	14.0	35.5	10.9	10.5	17.0	10.7
	4	31.2	97.5	53.7	35.2	30.4	30.4
	5	16.8	16.1	13.8	15.8	12.6	11.8
	6	17.6	20.4	16.6	14.4	12.8	13.2
	7	24.7	26.5	25.6	21.0	19.4	17.4
P1, PMIS	1	10.2	36.4	28.8	14.6	12.0	9.7
	2	9.9	42.3	33.2	17.7	15.9	12.0
	3	18.7	58.5	47.8	25.1	26.1	17.8
	4	25.6	113.7	84.2	45.8	31.3	29.2
	5	18.0	18.1	17.7	14.9	13.2	12.8
	6	19.1	22.8	22.9	18.8	14.0	13.6
	7	22.3	26.9	26.9	22.9	18.6	18.0

Table III. Run times for AMG-CG using various interpolation operators on various test problems using H1 or P1 coarsening on levels with aggressive coarsening and HMIS or PMIS on levels without

Since our previous experiments showed that two stage interpolation leads generally to better results than distance-two multipass interpolation, we only coded a parallel version of two stage interpolation and distance-two multipass interpolation is not considered here. Just as in the previous section we also omit results for standard two stage interpolation since for all problems considered here extended+i two stage interpolation gave better or equivalent results.

We consider Problems 3, 4, 5 and 7 from the previous section and examine their weak scalability. Problem 3 (now using a grid size of 500×500 per processor) and Problem 5 (with $50 \times 50 \times 50$ grid points per processor) were run on BG/L, which allowed us to use larger numbers of processors. Problem 4 (with approximately 86,000 degrees of freedom per processor) and Problem 7 (with approximately 58,000 degrees of freedom per processor) were run on Hera, a quadcore quadsocket Opteron Linux system, since aFEM was not available to us on BG/L.

Since the use of distance-two interpolation operators generally leads to large complexities for three dimensional problems, we employ truncation for 2s-ei to at most 4 elements per row for Problems 5 and 7, denoted with 2s-ei(4). For Problems 4, 5 and 7, we truncate the operators Q_1 and Q_2 to at most 4 elements per row in addition to the final interpolation operator, and denote this approach with '2s-ei(444)', to further control complexities and so receive smaller setup times. Using this approach on all levels generally leads to worse convergence than using 2sei(4) on all levels, however improves timings when used on one or two levels. Particularly for three-dimensional problems the use of truncation is crucial to achieve lower setup times.

For all problems, two stage interpolation shows better numerical scalability than multipass interpolation when used on all levels. However when aggressive coarsening is used on one or two levels only, as is usually the case in practice, numerical scalability improves for multipass interpolation significantly, approaching that of two stage interpolation, with two stage interpolation generally exhibiting a lower number of iterations.

Using 2s-ei on all levels for the rotated anisotropic Problem 3 (see Table IV and Figure 7)), led to a larger decrease in numerical scalability than desired, requiring the number of levels

No. of procs	ei(4)	mp	mp-1	2s-ei	2s-ei-1
64	10	145	28	24	19
256	11	205	30	33	19
1024	11	312	33	42	21
4096	12	> 400	35	51	21
16384	13	> 400	37	78	22
32400	13	> 400	38	91	22
65536	14	> 400	39	106	23
C_{op}	3.21	1.20	1.31	1.31	1.55

Table IV. No. of iterations for AMG-CG using various interpolation operators to Problem 3 with 500×500 grid points per processor using H1 coarsening, operator complexities were obtained for largest run.

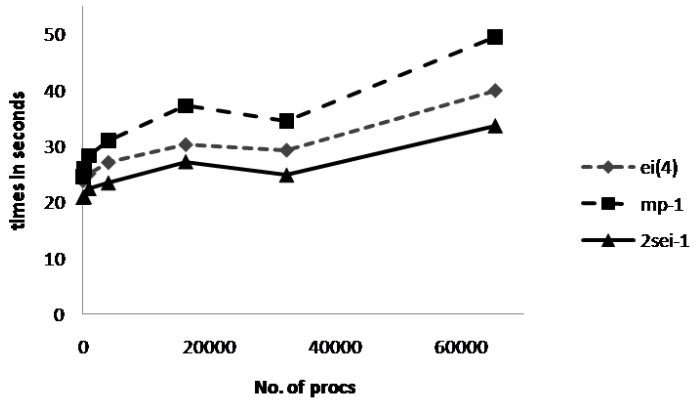

Figure 7. Run times for Problem 3 (2D 5pt) with 500×500 grid points per processor using H1 coarsening

No. of procs	ei(4)	mp	mp-1	2s-ei	2s-ei(4)	2s-ei(444)-1
1	1.5(9)	2.8(54)	1.3(20)	1.6(16)	1.4(16)	1.4(15)
64	3.5(12)	17.3(161)	3.4(24)	5.0(29)	4.8(32)	3.7(18)
256	4.4(13)	29.2(268)	4.6(27)	6.0(34)	5.5(35)	4.6(20)
1024	6.6(15)	32.8(280)	5.8(28)	9.4(55)	6.3(38)	5.3(21)
4096	11.2(16)	70.9(449)	7.8(29)	12.0(47)	10.9(47)	7.3(21)
C_{op}	2.65	1.13	1.30	1.49	1.32	1.45

Table V. Total times (No. of iterations) for AMG-CG applying various interpolation operators to Problem 4 (LLNL) with 500×500 grid points per processor using H1 coarsening, operator complexities were obtained for largest run

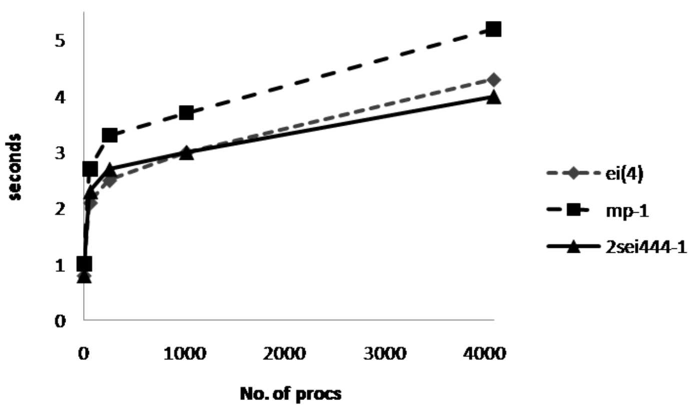

Figure 8. Solve times for Problem 4 (LLNL) with approximately 86,000 grid points per processor using H1 coarsening

No. of procs	ei(4)	mp	mp-1	mp-2	2s-ei	2s-ei(4)	2s-ei(444)-2
512	13	49	16	23	15	17	14
4096	16	75	20	27	17	21	15
8000	18	85	21	28	18	21	16
15625	26	95	21	29	18	22	17
32768	25	107	23	30	19	23	18
C_{op}	3.17	1.22	1.35	1.22	1.40	1.36	1.35

Table VI. No. of iterations for AMG-CG applying various interpolation operators to Problem 5 (7pt 3D) with 50×50×50 grid points per processor using H1 coarsening, operator complexities were obtained for largest run

No. of procs	ei(4)	mp	mp-1	2s-ei	2s-ei(4)	2s-ei(444)-1
1	2.9(10)	2.9(23)	2.5(19)	2.6(14)	2.4(15)	2.3(14)
64	9.0(12)	10.0(40)	6.2(19)	9.7(17)	8.4(21)	7.0(14)
512	20.5(15)	18.8(55)	13.1(19)	18.4(19)	14.8(22)	13.1(14)
4096	36.4(17)	31.7(77)	23.6(27)	40.2(24)	24.3(27)	23.1(19)
C_{op}	2.31	1.08	1.11	1.13	1.08	1.11

Table VII. Total times (No. of iterations) for AMG-CG applying various interpolation operators to Problem 7 (sphere) with approximately 58,000 grid points per processor using H1 coarsening, operator complexities were obtained for largest run

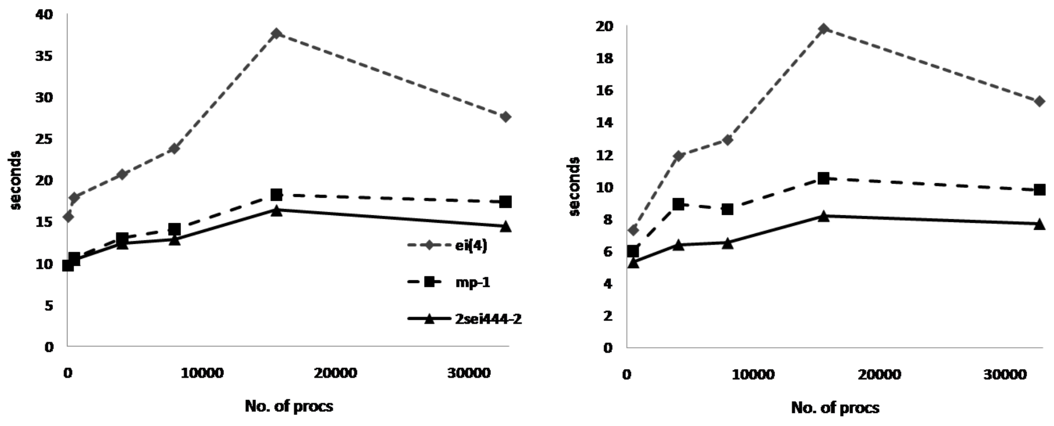

Figure 9. Total times (left) and solve times (right) for Problem 5 (7pt 3D) with 50×50×50 grid points per processor using H1 coarsening

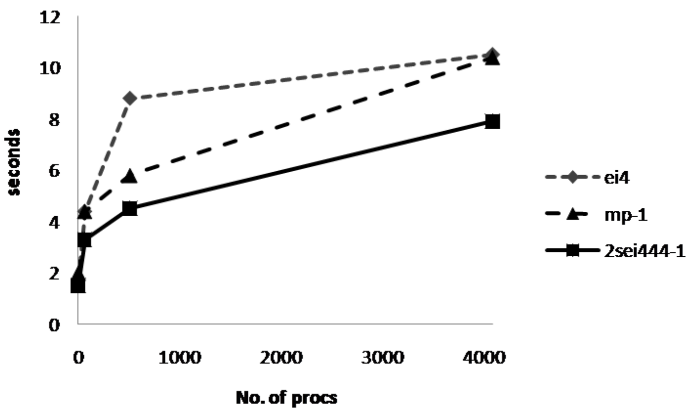

Figure 10. Solve times for Problem 7 (phere) with approximately 58,000 grid points per processor using H1 coarsening

of aggressive coarsening to be restricted. 2s-ei-1 converges faster than mp-1 here and exhibits lower total run times.

For Problem 4, 2s-ei and 2s-ei(4), exhibit also a larger than desired increase in number of iterations with increasing number of processors. Total times are almost identical for 2s-ei(444)-1 and mp-1, see Table V, but solve times are faster for 2s-ei(444)-1, see Figure 8.

For Problem 5, both 2s-ei and 2s-ei(4), show good convergence and numerical scalability when used on all levels. As a matter of fact, timings for 2s-ei(4) were almost identical to those of mp-1. However they were further improved, when using two levels of aggressive coarsening with 2s-ei(444), denoted with '2s-ei(444)-2'. 2s-ei(444)-2 was also about 20-30% faster than 2s-ei(444)-1, due to better complexities,and number of iterations. Using two levels for multipass interpolation led to a deterioration in number of iterations (see Table VI), about 20 % slower solve times, but faster setup times, leading to overall similar total times compared to one level of multipass interpolation. Total times and solve times for ei(4), mp-1 and 2s-ei(444)-1 are presented in Figure 9.

For Problem 7, the unstructured diffusion problem with jumps on a sphere, aggressive coarsening with 2s-ei(4) on all levels, shows reasonable performance, see Table VII. Total times are almost identical for 2s-ei(444)-1 and mp-1, with lower solve times for 2s-ei(444)-1, see Figure 10.

Using two stage interpolation on all levels led to decreasing numerical scalability for the two-dimensional problems, however gave reasonable results for the three-dimensional problems.

For all problems considered here, number of iterations and solve times for two stage interpolation are smaller than those of multipass interpolation. This makes this method attractive for situations where the preconditioner is set up only occasionally and reused many times, and where possibly a reevaluation of the preconditioner would not be required as often as for preconditioners using multipass interpolation.

In Figures 7, 8 and 10, the gap between mp-1 and 2sei-1 or 2sei(444)-1 appears to widen when increasing the number of processors, which, if this trend continues, would make two stage interpolation the better choice for millions of processors, our target architecture, for these problems. Whether this actually will be the case will have to be seen, since the final performance of the considered algorithms will depend on many factors, such as convergence, implementation, etc.

Setting up two stage interpolation is more expensive than multipass interpolation. We should note here that while we have a fairly efficient multipass interpolation routine, our implementation of two stage interpolation is straightforward, i.e. evaluates Q_1 and Q_2, followed by a sparse matrix-matrix multiplication to generate the operators. No effort has been made to increase its efficiency, by e.g. taking advantage of the cache. A better implementation will decrease the setup time, leading to better overall times. This however is the focus of future research. Nevertheless due to the low number of operations required to set up multipass interpolation, we expect its setup to always be faster.

8. Conclusion

Two new interpolation operators, distance-two multipass interpolation and two stage interpolation, designed to be used with aggressive coarsening have been introduced. Both of them are generated using distance-two interpolation, which are of higher quality than direct interpolation, the method used in multipass interpolation.

Our experiments show that both exhibit overall better convergence behavior than multipass interpolation, with two stage interpolation converging faster than distance-two multipass interpolation. While performing aggressive coarsening with multipass interpolation on all levels generally leads to decreasing convergence and numerical scalability (and has therefore not been used in practice), combining aggressive coarsening with two stage interpolation on all levels leads to fairly good numerical scalability for the three-dimensional diffusion problems considered here.

When aggressive coarsening is used on only one or two levels, total runtimes for two stage interpolation are only slightly lower or comparable to those for multipass interpolation, since it requires a more complex, and hence expensive, setup. Solve times are generally lower due to its better convergence and lower number of iterations. This makes this method attractive for situations where the preconditioner is set up only occasionally and reused many times, and where possibly a reevaluation of the preconditioner would not be required as often as for preconditioners using multipass interpolation.

Acknowledgments

The author thanks Joshua Nolting for implementing the version of the distance-two multipass interpolation routine that uses option 2 to choose the points for the first pass and Tzanio Kolev for providing the aFEM package and Figure 6. The author also thanks Rob Falgout and the referees for many helpful comments.

REFERENCES

1. D. Braess, Towards Algebraic Multigrid for Elliptic Problems of Second Order, *Computing* 55 (1995) 379–393.
2. A. Brandt, S. F. McCormick, and J. W. Ruge, Algebraic multigrid (AMG) for sparse matrix equations, in *Sparsity and Its Applications*, D. J. Evans, ed., Cambridge University Press, Cambridge, 1984.
3. W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial* (SIAM, Philadelphia, PA, second ed., 2000).
4. A. Buttari, J. Dongarra, J. Kurzak, P. Luszczek, The Impact of Multicore on Math Software, in *Applied Parallel Computing. State of the Art in Scientific Computing*, LNCS (2007) 4699.
5. A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, G. Miranda, J. Ruge, Robustness and Scalability of Algebraic Multigrid, *SIAM Journal on Scientific Computing* 21 (2000) 1886–1908.
6. H. De Sterck, U. M. Yang, and J. J. Heys, Reducing Complexity in Parallel Algebraic Multigrid Preconditioners, *SIAM Journal on Matrix Analysis and Applications* 27 (2006) 1019–1039.
7. H. De Sterck, R. D. Falgout, J. Nolting, and U. M. Yang, Distance-Two Interpolation for Parallel Algebraic Multigrid, *Numerical Linear Algebra with Applications* 15 (2008) 115–139.
8. V. Henson, U. M. Yang, BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner, *Applied Numerical Mathematics* 41 (2002) 155–177.
9. hypre: High Performance Preconditioners. http://www.llnl.gov/CASC/linear_solvers/.
10. M. Luby, A Simple Parallel Algorithm for the Maximal Independent Set Problem, *SIAM Journal on Computing* 15 (1986) 1036–1053.
11. J. W. Ruge and K. Stüben, Algebraic multigrid (AMG), in : S. F. McCormick, ed., *Multigrid Methods, vol. 3 of Frontiers in Applied Mathematics* (SIAM, Philadelphia, 1987) 73–130.
12. K. Stüben, Algebraic multigrid (AMG): an introduction with applications, in : U. Trottenberg, C. Oosterlee and A. Schüller, eds., *Multigrid* (Academic Press, 2000).
13. P. Vanek, J. Mandel, M. Brezina, Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order, *Computing* 56 (1996) 179–196.