Computing and Visualization in Science manuscript No.
(will be inserted by the editor)

Michael Griebel · Bram Metsch · Marc Alexander Schweitzer

Coarse grid classification–Part II: Automatic coarse grid agglomeration for parallel AMG

the date of receipt and acceptance should be inserted later

Abstract Multigrid methods (MG) are known to be optimal solvers for large sparse linear systems arising from the finite element, finite difference or finite volume discretization of a partial differential equation (PDE). Algebraic multigrid methods (AMG) extend this approach to wide a class of problems, e.g. anisotropic operators or unstructured grids. However, the parallelization of AMG, especially the construction of the coarse grids, is a challenging task. In this paper, we present an extension of the coarse grid classification scheme (CGC) for parallel AMG coarsening. Our new approach allows coarsening rates that are (essentially) independent of the number of processors. This consequently means that the presented scheme can coarsen a grid down to a single point independent of the number of processors, i.e. our scheme can be interpreted as an automatic coarse grid agglomeration scheme. The results of our numerical experiments in two and three space dimensions indicate that the presented scheme gives robust coarsening rates independent of the number of processors and provides small operator and grid complexities.

Supported by Sonderforschungsbereich 611, *Singuläre Phänomene und Skalierung in mathematischen Modellen*, supported by the *Deutsche Forschungsgemeinschaft*.

Michael Griebel
Institut für Numerische Simulation
Wegelerstraße 6
D-53115 Bonn
Germany
E-mail: griebel@ins.uni-bonn.de

Bram Metsch
Institut für Numerische Simulation
Wegelerstraße 6
D-53115 Bonn
Germany
E-mail: metsch@ins.uni-bonn.de

Marc Alexander Schweitzer
Institut für Numerische Simulation
Wegelerstraße 6
D-53115 Bonn
Germany
E-mail: schweitzer@ins.uni-bonn.de

Keywords algebraic multigrid · parallel computing
AMS Subject Classification 65N55; 65Y05; 65F10

1 Introduction

Multigrid methods (MG) are known to be optimal solvers for large sparse linear systems arising from the finite element, finite difference or finite volume discretization of a partial differential equation (PDE). However, the convergence behavior of a (geometric) multigrid method is (in most cases) strongly dependent on the coefficient functions of the differential operator; i.e., the solver is not robust with respect to a deterioration of the coefficient functions.

The development of so-called operator-dependent or matrix-dependent prolongations [1,11] was a first step toward robust (geometric) multigrid methods. However, in these methods, coarsening was still done in the classical way, for example by doubling the mesh size $h \mapsto 2h$. Especially for anisotropically structured problems this requires the usage of a complex smoother like alternating line relaxation or incomplete LU-factorization. In three spatial dimensions, these smoothers become rather complicated or even may loose their smoothing property.

With the introduction of algebraic multigrid (AMG) [2–4] in the 1980s many of these challenges could be dealt with. But the parallelization of AMG proved to be rather cumbersome. The direct parallelization of AMG is not feasible since the original Ruge–Stüben coarsening (RSC) algorithm [8] is inherently sequential.

More precisely, in most parallelization approaches a static disjoint domain decomposition of the computational domain is assumed; i.e., each processor p is assigned to a certain part of the domain, see Figure 1). Each processor then runs the AMG coarsening algorithm, e.g. RSC, locally on its data. However, these independently constructed local coarse grids may not align or match at processor boundaries. Even worse, in the case of anisotropies the resulting global coarse grids may not represent the physics of the diffusion problem especially

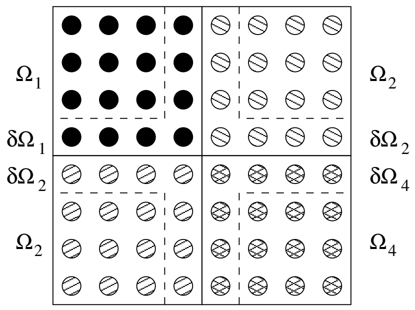

Fig. 1 Disjoint partitioning of a discretized domain Ω. We denote by Ω_p the part of Ω that is being handled by processor $p \in \{1, \ldots, P\}$. The boundary $\partial\Omega_p$ is defined as the points $i \in \Omega_p$ which are coupled to points $j \notin \Omega_p$ on other processors.

if the domain decomposition is not aligned with the isolines of the coefficient functions. Meanwhile, many different approaches to this matching problem have been proposed, see [5–7, 10]. But these algorithmic changes due to the parallelization in some cases have an adverse effect on the convergence behavior or the robustness of the overall solver.

The coarse grid classification (CGC) algorithm [5] is an efficient technique for the coarse grid selection in parallel AMG for large grids when the number of grid points N is much larger than the number of processors P. In contrary, if the number of grid points is of the same order as the number of processors, i.e. $N = O(P)$, then the CGC coarsening essentially stops. Hence, the size of the coarsest grid in CGC is $O(P)$ and the quality of the coarse grid correction with only simple relaxation will deteriorate. Also a direct solver is not practicable due to the involved costs for large P.

In this paper, we present an extension of the CGC which provides optimal coarsening rates even in the case where $N = O(P)$. The presented CGC-E scheme allows for an automatic coarsening down to a single grid point independent of the number of processors P. Hence, the quality of the coarse grid correction with simple smoothing is also robust with respect to P and we additionally obtain small operator complexities and grid complexities.

The remainder of this paper is organized as follows: In Section 2 we give a short summary of the sequential AMG algorithms and heuristics. Then, we review the CGC algorithm in Section 3 and discuss its drawbacks in the case when the number of nodes N is of the same order as the number of processors P. In Section 4 we present the so-called CGC-E scheme, which allows for fast coarsening even when $N = O(P)$. The key idea of the CGC-E scheme is that we allow a processor to select an "empty coarse grid" when local coarsening becomes inefficient. Then a number of processors become idle in CGC-E and we thus can interpret the method as an automatic coarse grid agglomeration scheme. From the numerical results in Section 5, we see that this new approach allows both a fast parallel AMG setup and an optimal multigrid convergence. Finally, we conclude with some remarks in Section 6.

2 Algebraic multigrid

In this section we give a short review of the algebraic multigrid (AMG) method. We consider a linear system $Au = f$, which comes from a discretization of a PDE on a grid $\mathbf{\Omega} := \{x_i\}_{i\in\Omega}$ with $\Omega := \{1, \ldots, N\}$ being the index set to enumerate the grid points, $A = (a_{ij})_{i,j=1}^N$ being a large sparse real matrix, $u = (u_i)_{i=1}^N$ and $f = (f_i)_{i=1}^N$ being vectors of length N. We assume that A is a symmetric, positive definite M-matrix or an essentially positive matrix. To be able to solve this equation using the

Program 1 Multigrid cycle $MG(A^l, f^l, u^l)$

```
begin
  for $\nu \leftarrow 1$ to $\nu_1$ do $u^l \leftarrow S^l u^l$; od;
  $r^l \leftarrow f^l - A^l u^l$;
  $f^{l+1} \leftarrow R^l r^l$;
  if $l + 1 = L$
    then $u^{l+1} \leftarrow \left(A^{l+1}\right)^{-1} f^{l+1}$;
    else for $\mu \leftarrow 1$ to $\mu_1$ do
           $MG(A^{l+1}, f^{l+1}, u^{l+1})$);
         od;
  fi;
  $u^l \leftarrow u^l + P^l u^{l+1}$;
  for $\nu \leftarrow 1$ to $\nu_2$ do $u^l \leftarrow S^l u^l$; od
end
```

multigrid scheme[1] (Program 1), we need to specify the sequence of coarse grid operators A^l, transfer operators P^l and $R^l = (P^l)^T$, the smoothing operators S^l and the sequence of coarse grids Ω^l.

In an algebraic multigrid method [2–4] we choose a simple smoothing scheme S^l, e.g. Gauss–Seidel or Jacobi relaxation. Then, we construct the grids $\{\Omega^l\}_{l=1}^L$, the transfer operators $\{P^l\}_{l=1}^{L-1}$ and the coarse grid operators $\{A^l\}_{l=1}^L$ in a recursive fashion depending on the fine grid operator $A = A^1$ only. This construction is carried out in the so-called setup phase which consists of three steps: the selection of an appropriate coarse grid

$$\Omega^{l+1} := C^l =: \Omega^l \setminus F^l,$$

the construction of a stable prolongation operator P^l, and the computation of the Galerkin coarse grid operator $A^{l+1} := (P^l)^T A^l P^l$, see Program 2. In the following, we omit the level index l where possible to simplify the notation.

[1] In classical geometric MG, Ω^L denotes the finest grid and Ω^1 the coarsest grid. Here, as usual in AMG, the levels are numbered the opposite way, i.e. starting with Ω^1 as the finest grid.

Program 2 Setup phase of AMG *AmgSetup* $(\Omega, A, N_{min}, L_{max}, L, \{A^l\}_{l=1}^{L}, \{P^l\}_{l=1}^{L-1}\{R^l\}_{l=1}^{L-1})$

```
begin
  $\Omega^1 \leftarrow \Omega$;
  $A^1 \leftarrow A$;
  for $l \leftarrow 1$ to $L_{max} - 1$
      do
      partition $\Omega^l$ into $C^l$ and $F^l$;
      $\Omega^{l+1} \leftarrow C^l$;
      compute interpolation $P^l$;
      $R^l \leftarrow (P^l)^T$;
      $A^{l+1} \leftarrow R^l A^l P^l$;
      if $|\Omega^{l+1}| \leq N_{min}$ then break;
      fi;
  od;
  $L \leftarrow l + 1$;
end.
```

Essential to many AMG schemes is the notion of a strong coupling between the grid point i and the grid point j. A grid point i is called *strongly coupled* to a grid point j if the corresponding matrix entry a_{ij} is relatively large. In the RSC algorithm [8,9] for instance the coarsening (see Program 3) is based on the sets

$$
\begin{aligned}
S_i &:= \{j \neq i : \ -a_{ij} \geq \alpha \max_{k \neq i} |a_{ik}|\}, \\
S_i^T &:= \{j \neq i : \ i \in S_j\},
\end{aligned}
$$

(typically $\alpha = 0.25$) which describe the strong connectivity graph of a matrix A. Along these strong couplings, the smooth parts of the error vary slowly. For an efficient coarse-grid correction of these parts, interpolation must also follow these couplings; i.e., each fine grid point i must be strongly coupled to at least one coarse grid point j.[2] Furthermore it is necessary to enforce

$$
\frac{\sum_{k \in S_i \cap C} -a_{jk}}{\max_k |a_{jk}|} > \beta \cdot \frac{-a_{ij}}{\max_k |a_{ik}|}, \tag{1}
$$

with a typical value of $\beta = 0.35$, for all pairs of fine grid points i, j to ensure that each fine grid point i is sufficiently "surrounded" by coarse grid points.

The coarsening process is carried out in two phases. In the first phase, see Program 3, an independent set C of coarse grid points is determined. A second phase then checks condition (1) for all pairs $i, j \in F$ of fine grid points. If this equation is not fulfilled for a certain pair, then one of these points is added to the set of coarse grid points C.

Note that in *AmgPhaseI* each point i chosen for the coarse grid C results in a change of the weights λ_j of all points j within two layers around the grid point i. This shows the sequential character of this coarsening algorithm, as the weight updates propagate throughout the whole domain during the coarsening loop. Hence,

[2] If a point i has no strong couplings at all, the error at this points can be reduced efficiently by smoothing only. Hence, we can add i to the set of fine grid points F.

Program 3 *AmgPhaseI*(Ω, S, S^T, C, F)

```
begin $U \leftarrow \Omega$;
  $C \leftarrow \emptyset$;
  $F \leftarrow \emptyset$;
  for $i \in \Omega$ do $\lambda_i \leftarrow |S_i^T|$; od;
  while $\max_{i \in U} \lambda_i \neq 0$ do
        $i \leftarrow \arg\max_{j \in U} \lambda_j$;
        $C \leftarrow C \cup \{i\}$;
        for $j \in S_i^T \cap U$ do $F \leftarrow F \cup \{j\}$;
            for $k \in S_j \cap U$ do $\lambda_k \leftarrow \lambda_k + 1$; od;
        od;
        for $j \in S_i \cap U$ do $\lambda_j \leftarrow \lambda_j - 1$; od;
  od;
  $F \leftarrow F \cup U$;
end
```

the parallelization of RSC-based AMG schemes is not straightforward. To this end, many different parallelization techniques have been suggested over the years [6,7,10]. Most of them first employ the Ruge-Stüben coarsening scheme on each processor subdomain locally and then employ a special treatment of the processor subdomain boundaries to fix inconsistencies in the composed global grid. However, these boundary treatment algorithms often do not respect the structure of the underlying operator and produce unphysical coarse grids. Recently, we proposed a new approach to the the parallel coarse grid selection, the so-called coarse grid classification (CGC) scheme [5], which overcomes this problem to some extent.

3 Coarse Grid Classification

The key observation which led to the development of the CGC scheme is that the sequential character of RSC can be used to introduce an additional degree of freedom in the coarse grid selection, namely, the initial choice of the first coarse grid point. Since the coarse grid points in RSC are chosen sequentially, it is clear that we obtain a different coarse grid by altering the initial choice for the first coarse grid point, see Figure 2. Note that the quality of the resulting coarse grids with respect to multigrid convergence and memory requirements is nevertheless very similar. Hence, there is no special advantage of using one of these coarse grids over another in a sequential computation. In parallel computations, however, this gives us a degree of freedom to consistently match the coarse grids obtained on each processor individually by RSC at processor subdomain boundaries.

In summary, the CGC approach is as follows (see [5] for details). First, we construct multiple (disjoint) coarse grids on each processor p independently by running the RSC algorithm multiple times with different initial coarse grid points on its subdomain Ω_p, compare Figure 1. Then, we need to select exactly one grid for each processor subdomain such that the union of these local coarse grids forms a suitable global coarse grid for

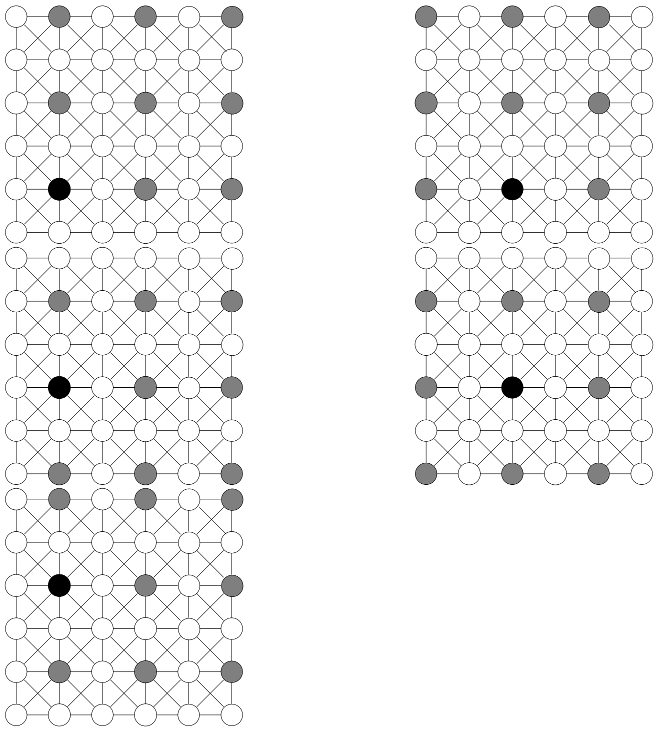
Fig. 2 Resulting coarse grids for a 9-point discretization of the Laplace operator constructed by five different initial choices. The gray points indicate the respective coarse grid points, the black point indicates the first coarse grid point chosen.

Program 4 $CGC(S, S^T, ng, \{C_i\}_{i=1}^{ng}, \{F_i\}_{i=1}^{ng})$

```
for $j \leftarrow 1$ to $|\Omega|$ do $\lambda_j \leftarrow |S_j^T|$; od;
$C_0 \leftarrow \emptyset$;
$\lambda_{\max} \leftarrow \max_{k \in \Omega} \lambda_k$;
do
    $U \leftarrow \Omega \setminus \bigcup_{i \leq it} C_i$;
    for $j \in U$ do $\lambda_j \leftarrow |S_j^T|$; od;
    if $\max_{k \in U} \lambda_k < \lambda_{\max}$ then break; fi;
    $it \leftarrow it + 1$;
    $F_{it} \leftarrow \emptyset$;
    $C_{it} \leftarrow \emptyset$;
    do
        $j \leftarrow arg \max_{k \in U} \lambda_k$;
        if $\lambda_j = 0$ then break; fi;
        $C_{it} \leftarrow C_{it} \cup \{j\}$;
        for $k \in S_j^T \cap U$ do
            $F_{it} \leftarrow F_{it} \cup \{k\}$;
            for $l \in S_k \cap U$ do $\lambda_l \leftarrow \lambda_l + 1$; od;
        od;
        for $k \in S_j \cap U$ do $\lambda_k \leftarrow \lambda_k - 1$; od;
    od;
    $F_{it} \leftarrow F_{it} \cup U$;
od;
$ng \leftarrow it$;
```

the whole domain. We achieve this by defining a weighted graph whose *vertices* represent the *grids* constructed by the multiple coarsening runs. The *edges* of this weighted graph are defined between vertices which represent grids on neighboring processor subdomains. Each edge weight measures the quality of the boundary constellation if these two grids would be chosen to be part of the composed global grid. Finally, we use a graph clustering technique on the resulting weighted graph to choose one coarse grid for each processor subdomain which automatically matches with most of its neighbors.

Note that this procedure is computationally efficient since one iteration of the RSC requires only a very small amount of computational time compared to the construction of the prolongation and coarse grid operators, while a well-constructed grid can save a large amount of time during the operator construction and in the multigrid cycle. To control the computational complexity of the CGC algorithm, we limit ourselves to the construction of disjoint coarse grids only, see Program 4.

In detail, in our CGC we proceed as follows: Each processor p first determines the maximal weight $\lambda_{\max}$ of all points $i \in \Omega_p$. Any point with this weight can be chosen as an initial point for the RSC algorithm, compare Program 3. We choose one particular point $\tilde{i}$ and construct a coarse grid $C_{(p),1}$. We now re-initialize the weights $\lambda_i := |S_i^T|$ of all remaining points $i \in \Omega \setminus C_{(p),1}$ to their original values. From these points, we select another point $\tilde{j}$ with weight $\lambda_{\max}$ and construct a second coarse grid $C_{(p),2}$ starting with this point. However, only points not contained in $C_{(p),1}$ may be inserted into $C_{(p),2}$, i.e. we construct disjoint coarse grids. We repeat these steps as long as there is a point with weight $\lambda_{\max}$ that is not already a member of a coarse grid $C_{(p),it}$.

We now have obtained ng_p valid coarse grids $\{C_{(p),i}\}_{i=1}^{ng_p}$ on each processor p. To determine which grid to choose on each processor, we construct a directed, weighted graph $G = (V, E)$ whose vertices represent the created coarse grids, i.e.

$$V_p := \{C_{(p),i}\}_{i=1,\ldots,ng_p}, \quad V := \cup_{p=1}^{P} V_p.$$

The set of edges E consists of all pairs (v, u), $v \in V_p$, $u \in V_q$ such that $q \in \mathcal{S}_p$ is a neighboring processor of p,

$$E_p := \{\cup_{q \in \mathcal{S}_p} \cup_{v \in V_p,\ u \in V_q} (v, u)\}, \quad E := \cup_{p=1}^{P} E_p,$$

where $\mathcal{S}_p$ is defined as the set of processors q with points j which strongly influence points i on processor p, i.e.

$$\mathcal{S}_p := \{q \neq p :\ \exists i \in \Omega_p,\ j \in \Omega_q :\ j \in S_i\}.$$

We determine the weight $\gamma(e)$ of an edge $e = (v, u)$ with $v \in V_p$, $u \in V_q$ by assessing the quality of the resulting joint coarse grid $u \cup v$ for the subdomain $\Omega_p \cup \Omega_q \subset \Omega$ at their common interface. To this end, we identify three classes of grid configurations across the interface, namely $C{-}C$, $C{-}F$ or $F{-}C$, and $F{-}F$. Let us further denote by $c_{C,C}$ the number of strong $C-C$-couplings, by $c_{C,F}$ the number of strong $C-F$-couplings, by $c_{F,C}$ the number of strong $F-C$-couplings and by $c_{F,F}$ the number of strong $F-F$-couplings. With their help we can define the edge weight

$$\gamma(e) := c_{C,C}\gamma_{C,C} + (c_{C,F} + c_{F,C})\gamma_{C,F} + c_{F,F}\gamma_{F,F}$$

with $\gamma_{C,C} = -1$, $\gamma_{C,F} = 0$, and $\gamma_{F,F} = -8$, see [5] for details.

Now that we have constructed the graph G of admissible local grids, we can use it to choose a particular coarse grid for each processor such that the union of these local grids automatically matches at subdomain boundaries. Note here that the number of vertices is related to the number of processors P only, i.e., it is much smaller than the number of unknowns N. Furthermore, the cardinality of E is small compared to N since edges are only constructed between neighboring processors. Thus, we can transfer the whole graph onto a single processor without large communication costs.[3] On this processor we use a heavy edge matching approach to select exactly one coarse grid per processor, see [5] for details. This algorithm preforms efficiently for $N \ll P$, as can be seen from the numerical results in [5].

We now point out a drawback of this CGC algorithm. As long as at least one coarse grid is constructed on a particular subdomain, exactly one coarse grid is chosen by the CGC selection mechanism. Therefore, CGC will in many cases not be able to coarsen down to a single grid point, and, consequently, the coarsest grid in CGC will often be of substantial size, i.e. at least $O(P)$. However, in most cases strong couplings are still present on this level, which means that a simple relaxation scheme will not be able to reduce the associated error efficiently. Thus, the quality of the resulting coarse grid correction will be poor and the convergence rate will deteriorate with increasing number of processors P. With a (redundant) direct solver this can be cured, but only at the prize of substantially increased computational and memory costs.

In the following section, we present an extension of the CGC scheme which will resolve this problem.

4 Coarse Grid Classification with empty candidate coarse grids

The original CGC algorithm, like the classical RSC algorithm, is not able to coarsen the whole grid down to a single point in most cases. While all strong couplings *inside* a processor subdomain can be eliminated during the construction of the grid hierarchy, this is not true for the couplings *across* subdomain boundaries. An example is depicted in Figure 3. On each processor subdomain, the RSC algorithm coarsened each vertical grid line down to a single point, but the classical RSC algorithm is not able to coarsen this grid any further. We see that each point i has a strong coupling, hence its weight λ_i (see Program 3) is different from zero.[4] In consequence, the RSC algorithm picks such a point and adds it to the set of coarse grid points C. However, the strongly coupled neighbors are not assigned to the set of fine grid points as they reside on another processor subdomain. On the neighbor processor subdomain, the same happens. All points are thus assigned to the set of coarse grid points C. In consequence, the setup phase stops at this point even if the vertical grid lines could be coarsened further. The coarsest grid which we then obtain is of the same size as the subdomain boundary. However, such a coarsest grid is not desirable for an efficient algebraic multigrid solver. As long as points are coupled, the error at such points cannot be reduced efficiently by smoothing only. Hence, a multigrid cycle that employs smoothing on the coarsest level only will not allow a convergence behavior independent on the number of unknowns. On the other hand, a direct coarse grid correction, e.g. simple Gaussian elimination, not only requires $O(n^3)$ time, but also $O(n^2)$ storage to solve the coarse grid system, where n is the number of unknowns on the coarsest grid. Due to memory limitations, it may even be impossible to carry out a direct solver at all.

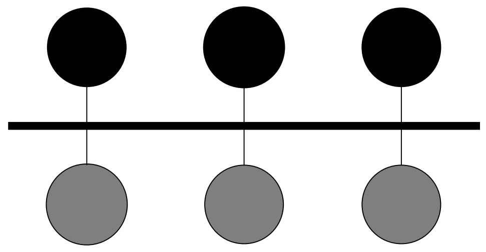
Fig. 3 Coarsening of an anisotropic operator with strong couplings in vertical direction only. Depicted is the coarse grid structure near the processor subdomain boundary (shown as bold horizontal line) , where each processor has coarsened each vertical grid line down to a single point.

Note that the classical CGC algorithm is also not able to overcome this problem. In the example described above, all points are already included in the first candidate coarse grid. There is just no degree of freedom to construct a different grid. Hence, the CGC algorithm constructs the same coarse grid as the classical RSC algorithm.

In the example described above, the coarsening process terminates too soon. In addition, the coarsening ratios often decrease during the setup phase. An example is given in Figure 4, where we see the number of unknowns on each level for a three-dimensional anisotropic problem. While a nearly perfect coarsening ratio is achieved on the first levels, on coarser levels nearly all points are

[3] For very large numbers of processors P, one can extend this algorithm by splitting the graph among a subset of the processors instead of transferring it onto a single processor. A local matching and a recursive application of our classification then gives the globally consistent coarse grid.

[4] We assume that the weights are updated before the coarsening process starts on each subdomain.

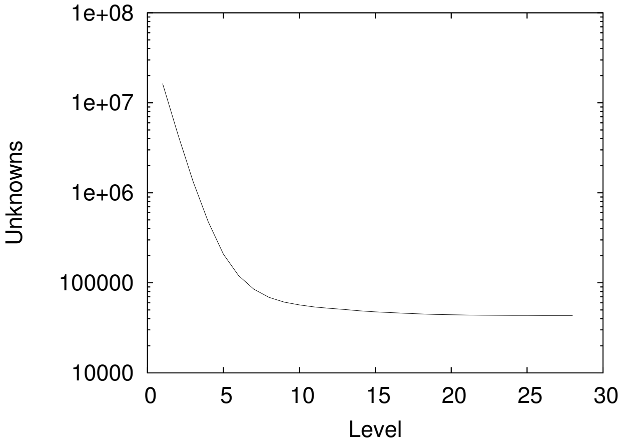

Fig. 4 Coarsening history for a three-dimensional anisotropic problem (cp. Example 4). Depicted is the number of unknowns on each level as produced by the CGC algorithm.

carried over in the next coarser grid. The reason for this lies in the fact that, as the number of unknowns decreases, a larger part of the points has strong connections across a subdomain boundary.

One solution to this problem is to employ a fully parallel coarsening algorithm. Such a method is given by the CLJP method [6]. This algorithm iteratively constructs multiple independent sets in the strength connectivity graph S and then takes the union of these sets as the coarse grid. However, a communication step is required after each independent set is determined and the resulting coarse grid contains more points than the classical Ruge-Stüben algorithm. A hybrid RSC/CLJP coarsening scheme, like the Falgout algorithm [6], reduces the number of CLJP iterations, but still creates too many coarse grid points especially at the subdomain boundaries.

Let us return to the example depicted in Figure 3. It becomes clear that in this case it is sufficient for stable interpolation if a single coarse grid is selected on one of the processor subdomains while all points on the other subdomain are designated as fine grid points.

This motivates us to add the *empty* grid (i.e. no coarse grid points on this processor subdomain) to the set of candidate coarse grid points. As in the classical CGC algorithm, see Section 3, a vertex represents this "coarse grid" and we construct weighted edges from this vertex to vertices corresponding to candidate coarse grids on adjacent processors. Recall that every strong $F-F$ coupling across the boundary is heavily penalized, so the selection algorithm will not choose the empty grid unless (almost) all points on the respective processor have a strong coupling to coarse grid points on another processor.

An additional issue still needs to be addressed: The CGC selection mechanism does not check if interpolation is possible for the *inner* points. Hence, we only allow the empty grid to the set of empty coarse grids if one of the two following criteria is fulfilled for *all* points on the processors subdomain:

- The point has at least one strong connection to a point on another processor, i.e. the CGC selection mechanism can determine whether the value at this point can be interpolated from the value at a coarse grid point.
- The point has no strong couplings at all, i.e. the error at this point can be reduced efficiently by smoothing only.

In consequence, the resulting graph of admissible coarse grids is slightly larger in the resulting so-called CGC-E scheme than in the original CGC algorithm. The further selection mechanism then takes place as described in the previous section. But in CGC-E the empty grid can additionally be selected as "coarse grid" for a certain processor subdomain, i.e., all points on that subdomain remain fine grid points.

5 Numerical experiments

We now compare this new approach, which we denote by CGC-E, with the original CGC scheme [5] and the parallel Falgout coarsening algorithm [6]. To this end, we introduce two measures for the quality of the constructed hierarchy, the *operator complexity* c_A and the *grid complexity* c_G. They are defined as follows,

$$c_A := \frac{\sum_{l=1}^{L_{\max}} nonzeros(A^l)}{nonzeros(A^1)}, \quad c_G := \frac{\sum_{l=1}^{L_{\max}} |\Omega^l|}{|\Omega^1|}. \quad (2)$$

These numbers give an indication of the memory overhead required by the AMG algorithm.

In all our experiments, we set $\alpha = 0.25$ for the first phase and $\beta = 0.35$ for the second phase of the RSC algorithm, see Section 2. For the setup of the transfer operators, we use the truncated standard interpolation [9] scheme with a truncation parameter $\epsilon_{tr} = 0.2$. The setup phase is stopped if either no strong couplings are present or the coarsening algorithm inserts all fine grid points into the coarse grid.

As an initial guess for the solution phase we use a random-valued vector u_0 with $\|u_0\|_{l_2} = 1$. We perform the $V(1,1)$-cycle, i.e. one pre- and one post-smoothing step per level. The iteration is stopped after it steps if the l^2-norm of the residual $r_{it} = f - Au_{it}$ drops below 10^{-10}. On each level, we employ a (subdomain) block-Jacobi smoother with one inner Gauss–Seidel relaxation step. Note that we do *not* employ a direct solver on the coarsest level. The convergence factor is determined as

$$\rho = \left(\frac{\|r_{it}\|_{l^2}}{\|r_1\|_{l^2}} \right)^{\frac{1}{it-1}}. \quad (3)$$

Table 1 Setup time in seconds for Example 1.

P	Falgout	CGC	CGC-E
1	13.1	13.1	13.1
4	17.5	15.0	15.6
16	18.4	15.7	16.5
64	20.4	18.5	18.6
256	28.5	26.2	25.9

For the first three experiments, we tested the various coarsening schemes on the Himalaya cluster at the department for Scientific Computing at the University of Bonn. This cluster consists of 256 Xeon 3.2 GHz CPUs, connected by Myrinet. The last experiment was carried out on the JUMP supercomputer at the Forschungszentrum Juelich, which consists of 41 IBM p690 with 32 CPUs and 128GB each. There, the interconnect is provided through an HPS network.

Example 1 Laplace operator in two dimensions. In our first experiment we consider the Poisson problem

$$-\Delta u = 0 \text{ in } \Omega = (0,1)^2 \tag{4}$$

with zero Dirichlet boundary conditions on the unit square, discretized with a 5-point finite difference stencil on 512×512 points per processor. In this standard test case, we expect that the convergence rate of the original CGC will deteriorate with increasing number of processors P, since the coarsest level of CGC will be too large, i.e. $O(P)$, to obtain a high quality coarse grid correction by smoothing only. On the other hand, we anticipate that the CGC-E scheme will converge with a rate that is independent of the number of processors P, as the coarsest grid obtained by CGC-E is of size $O(1)$. Furthermore, we expect that the setup times for the CGC and CGC-E schemes are very similar.

In Table 1 we give the setup times for this problem. We see that both CGC variants are faster than the Falgout algorithm. On $P = 4$, 16 and 64 processors, the CGC-E algorithm needs slightly more time than the CGC algorithm due to a somewhat larger coarse grid graph V. On the other hand, the CGC-E algorithm allows an automatic agglomeration and thus a faster setup on coarse levels. On 256 processors this compensates for the slower selection mechanism. This effect can also be seen from the operator complexities given in Table 2.

Table 2 Operator complexity (2) for Example 1.

P	Falgout	CGC	CGC-E
1	2.60	2.60	2.60
4	2.63	2.60	2.60
16	2.64	2.61	2.61
64	2.65	2.63	2.62
256	2.65	2.65	2.62

As the grids produced on the coarsest level are smaller, the operator complexities produced by the CGC-E algorithm are smaller than those obtained by the CGC algorithm. Note that the Falgout algorithm cannot achieve the complexities of the CGC-E scheme even though it also allows an automatic agglomeration. This is due to the employed CLJP algorithm which produces many (unnecessary) points near the processor boundaries.

Table 3 Solution time in seconds for Example 1.

P	Falgout	CGC	CGC-E
1	8.53	8.53	8.53
4	10.8	19.6	9.06
16	11.0	19.5	11.1
64	11.3	24.5	11.5
256	16.8	104	17.2

Table 3 shows the solution times, i.e., the time required by the $V(1,1)$-iteration to reduce the residual to 10^{-10}, for this example. While the Falgout and the CGC-E scheme allow a nearly scalable solution up to 64 processors, this is not true for the CGC algorithm. The explanation can be seen from Table 4: The CGC algorithm cannot coarsen down to a single point, hence a simple relaxation scheme on the coarsest level does not provide a high quality coarse grid correction. In this example the coarsest grid produced by the CGC algorithm still contains 593 unknowns for 64 processors and 1795 points for 256 processors, while the other schemes coarsen down to a single point. In turn, this leads to a deterioration of the overall convergence behavior of the multigrid cycle for the CGC.

Table 4 Convergence factors (3) for Example 1.

P	Falgout	CGC	CGC-E
1	0.13	0.13	0.13
4	0.13	0.38	0.13
16	0.14	0.35	0.13
64	0.14	0.41	0.14
256	0.14	0.73	0.17

Example 2 Grid-aligned anisotropy in two spatial dimensions. We now investigate the parallel coarsening schemes for an example with grid-aligned anisotropy. We consider the equation

$$-\epsilon u_{xx} - u_{yy} = 0$$

with $\epsilon = 0.001$ and zero boundary conditions on the unit square $(0,1)^2$, discretized using a finite difference stencil on 512×512 points per processor.

Again, we expect that the CGC-E scheme will provide small complexities and a converge rate which is independent of the number of processors P.

Table 5 Setup time in seconds for Example 2.

P	Falgout	CGC	CGC-E
1	5.63	5.63	5.63
4	9.41	8.25	8.23
16	10.5	9.07	9.09
64	14.2	12.8	12.3
256	21.9	19.9	19.9

From the setup times displayed in Table 5, we see that the CGC-E algorithm does not need additional time in comparison to the CGC scheme. However, the quality of the grid and operator hierarchy constructed by the CGC-E scheme is better, as can be seen from the operator complexities displayed in Table 6. Out of the three schemes considered, the CGC-E method gives the smallest complexities. The measured solution times and convergence factors are given in Table 7 and Table 8, respectively. We see that the Falgout algorithm and the CGC-E algorithm obtain comparable solution times and scalable convergence factors. As in the isotropic case, the coarsest grid provided by the CGC method is still too large to provide an efficient reduction of the error by smoothing only.

Table 6 Operator complexity (2) for Example 2.

P	Falgout	CGC	CGC-E
1	2.01	2.01	2.01
4	2.14	2.16	2.14
16	2.12	2.11	2.09
64	2.10	2.11	2.08
256	2.09	2.09	2.07

Table 7 Solution time in seconds for Example 2.

P	Falgout	CGC	CGC-E
1	4.28	4.28	4.28
4	8.59	49.2	8.62
16	10.6	23.5	10.7
64	13.5	25.8	13.6
256	13.7	42.1	14.0

Table 8 Convergence factors (3) for Example 2.

P	Falgout	CGC	CGC-E
1	0.14	0.14	0.14
4	0.14	0.75	0.14
16	0.14	0.45	0.14
64	0.14	0.38	0.14
256	0.14	0.56	0.14

Example 3 Non-grid aligned anisotropic problem.

Table 9 Setup time in seconds for Example 3.

P	Falgout	CGC	CGC-E
1	7.02	7.02	7.02
4	10.9	9.99	10.0
16	12.0	11.2	11.1
64	14.5	14.5	13.5
256	23.8	24.7	22.8

Let us now consider the two-dimensional anisotropic problem

$$-(\cos^2\gamma + \epsilon\sin^2\gamma)u_{xx} + 2(1-\epsilon)\sin\gamma\cos\gamma u_{xy}$$
$$-(\sin^2\gamma + \epsilon\cos^2\gamma)u_{yy} = 0$$

on the unit square $(0,1)^2$ with $\epsilon = 0.001$. In this case, the anisotropy is not grid aligned and depends on the angle γ. Geometric multigrid with standard $h \mapsto 2h$ coarsening fails for every value of γ other than 0° or 90°. Sequential AMG can cure this problem, compare [7]. In the following, we concentrate on the case of $\gamma = 45°$.

The problem is discretized using the stencil

$$\frac{1}{h^2}\begin{bmatrix} 0 & -0.001 & -0.4995 \\ -0.001 & 1.003 & -0.001 \\ -0.4995 & -0.001 & 0 \end{bmatrix}_h$$

on 512×512 points per processor.

From the stencil we already see that the strong couplings are diagonally aligned. This means that each processor subdomain has not only strong couplings to the adjacent subdomains in y-direction (as in the previous example), but also to the adjacent subdomains in x-direction and to two subdomains across the corners. Hence, we expect to see larger setup and solution times for this example than in the grid-aligned case.

Table 9 shows that all parallel coarsening techniques require nearly the same setup times. Generally, the times are larger than in the grid-aligned case as there about twice as much strong couplings across the processor subdomains. For the CGC and CGC-E algorithm, this means that more edges are constructed in the coarse grid selection graph, while the Falgout algorithm needs to employ the boundary treatment on twice as many points.

Table 10 Operator complexity (2) for Example 3.

P	Falgout	CGC	CGC-E
1	2.63	2.63	2.63
4	2.70	2.72	2.70
16	2.73	2.79	2.74
64	2.74	2.87	2.76
256	2.75	2.97	2.79

For the operator complexities, c.f. Table 10, we see an almost equal increase for the Falgout and CGC-E algorithm. In contrast, the CGC algorithm shows a larger deterioration than in the grid-aligned case, which can be explained from the large amount of strongly coupled coarse grid points on different processor subdomains.

Table 11 Solution time in seconds for Example 3.

P	Falgout	CGC	CGC-E
1	4.63	4.63	4.63
4	13.4	34.8	14.2
16	14.8	100	15.0
64	15.1	65.4	14.9
256	24.6	107	24.4

Table 12 Convergence factors (3) for Example 3.

P	Falgout	CGC	CGC-E
1	0.14	0.14	0.14
4	0.27	0.63	0.29
16	0.29	0.85	0.29
64	0.30	0.76	0.28
256	0.30	0.75	0.27

From the solution times given in Table 11 and the convergence factors given in Table 12 we see that the classical CGC algorithm again does not provide an efficient multigrid cycle without agglomeration. The other algorithms only exhibit some deterioration as we go from the sequential to the parallel setting, but the parallel convergence factors are independent of the number of processors. Here, we see the impact of an inferior relaxation scheme at the subdomain boundaries.

Example 4 Three-dimensional anisotropic problem.
In the last example we consider the three-dimensional anisotropic problem

$$-\epsilon u_{xx} - u_{yy} - u_{zz} = 0$$

with $\epsilon = 0.001$ and zero boundary conditions on the unit cube $(0, 1)^3$. This equation is discretized using a 7-point finite difference stencil on $40 \times 40 \times 40$ grid points per processor.

Compared to the previous examples, we have a larger surface-to-volume ratio for each processor subdomain. Hence, we expect to see a larger deterioration of the scale-up behavior especially for the solution phase, where smoothing requires a large amount of communication.

In Table 13 we show the setup times for this example. We see that while both CGC variants exhibit comparable times up to 64 processors, this is no longer true for 512 processors any more. In this case, due to the absence of an agglomeration strategy the original CGC coarsening process slows down and the construction of transfer and coarse grid operators takes a significant amount of time. The Falgout algorithm employs the CLJP method to coarsen the boundary planes. This scheme clusters coarse grid points near the subdomain boundary planes,

Table 13 Setup time in seconds for Example 4.

P	Falgout	CGC	CGC-E
1	7.77	7.77	7.77
8	12.5	10.6	11.0
64	19.6	16.7	16.4
512	35.0	38.0	30.8

which even further increases the surface-to-volume ratio on the next level. In contrast, the CGC-E algorithm, which combines an efficient coarse grid matching and fast coarsening, achieves the fastest setup.

Table 14 Operator complexity (2) for Example 4.

P	Falgout	CGC	CGC-E
1	2.40	2.40	2.40
8	2.76	2.54	2.51
64	2.94	2.95	2.66
512	3.02	4.14	2.79

From Table 14 we see that the operator complexity increases for all coarsening methods as the number of processors grows. Hence, we cannot expect solution-time scalability. As pointed out before, the Falgout algorithm constructs larger coarse grids than necessary, while the CGC method does not provide an efficient coarsening as the number of processors grows.

Table 15 Solution time in seconds for Example 4.

P	Falgout	CGC	CGC-E
1	0.84	0.84	0.84
8	3.08	2.90	2.77
64	8.19	9.42	8.65
512	33.6	24.1	15.4

In contrast to the two-dimensional examples, all coarsening schemes achieve similar convergence rates: from 0.13 for one processor up to 0.15 for 512 processors. Hence, the solution time mainly depends on the computational work done per multigrid cycle which in turn is related to the operator complexity. This can be seen from the solution times given in Table 15. As we increase the number of processors from 64 to 512, we see a large jump in the solution times due to hardware and interconnect bandwidth limitations.

In summary, we conclude that the CGC-E algorithm provides setup times compared to that of the original CGC algorithm, while the convergence factors and solution times resemble those of the Falgout algorithm. Hence, this scheme provides both automatic subdomain agglomeration and efficient Ruge-Stüben-like coarse grid construction for parallel algebraic multigrid.

6 Concluding remarks

In this paper we presented an extension to the parallel CGC coarsening algorithm. Our aim was to provide an efficient parallel algebraic multigrid scheme that does not require the use of a parallel or redundant direct (expensive) solver on the coarsest level, which in turn requires that the coarsening process does not stop until there are no strong couplings left regardless of the domain decomposition. In the CGC-E scheme, we first decide whether the value at all points on a certain processor subdomain can be interpolated from the values at points on other processor subdomains (or do not require interpolation at all). If this is the case, we add the empty grid to the set of candidate coarse grids and allow the respective processor to become idle.

The results of our numerical experiments showed that this enhancement reduces both the setup time and operator complexity for problems with anisotropies. In addition, our new CGC-E scheme allows for coarsening up to a single point (globally) which is important for an efficient multigrid cycle.

Overall, the CGC-E algorithm provides a scalable parallel AMG solver which only employs a simple smoothing scheme on all levels. Thus, the use of an expensive (parallel or redundant) direct solver is avoided while we still obtain good convergence rates even in the case of $N \approx P$.

References

1. Alcouffe, R.E., Brandt, A., Dendy, J.E., Painter, J.W.: The multi-grid method for the diffusion equation with strongly discontinuous coefficients. SIAM J. Sci. Stat. Comput. **2**, 430–454
2. Brandt, A.: Algebraic multigrid theory: The symmetric case. Appl. Math. Comput. **19**(1-4), 23–56 (1986)
3. Brandt, A., McCormick, S.F., Ruge, J.: Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations (1982)
4. Brandt, A., McCormick, S.F., Ruge, J.: Algebraic multigrid (AMG) for sparse matrix equations. In: D. Evans (ed.) Sparsity and its Applications, pp. 257–284. Cambridge University Press, Cambridge (1984)
5. Griebel, M., Metsch, B., Oeltz, D., Schweitzer, M.A.: Coarse grid classification: a parallel coarsening scheme for algebraic multigrid methods. Numerical Linear Algebra with Applications **13**(2–3), 193–214 (2006). Also available as SFB 611 preprint No. 225, Universität Bonn, 2005
6. Henson, V.E., Yang, U.M.: BoomerAMG: a parallel algebraic multigrid solver and preconditioner. Tech. Rep. UCRL-JC-141495, Lawrence Livermore National Laboratory (2001)
7. Krechel, A., Stüben, K.: Parallel algebraic multigrid based on subdomain blocking. Tech. Rep. REP-SCAI-1999-71, GMD (1999)
8. Ruge, J.W., Stüben, K.: Algebraic multigid (AMG). In: S.F. McCormick (ed.) Multigrid Methods, *Frontiers in Applied Mathematics*, vol. 5. SIAM (1986)
9. Stüben, K.: Algebraic multigrid (AMG): An introduction with applications. In: U. Trottenberg, C.W. Oosterlee, A. Schüller (eds.) Multigrid, pp. 413–532. Academic Press (2001). Also available as GMD Report 53, GMD - Forschungszentrum Informationstechnik GmbH, March 1999
10. Yang, U.M.: Parallel algebraic multigrid methods - high performance preconditioners. In: A. Bruaset, A. Tveito (eds.) Numerical Solution of Partial Differential Equations on Parallel Computers, *Lecture Notes in Computational Science and Engineering*, vol. 51, pp. 209–236. Springer-Verlag (2006). Also available as technical report UCRL-BOOK-208032, Lawrence Livermore National Laboratory, November 2004
11. Zeeuw, P.d.: Matrix-dependent prolongations and restrictions in a black-box multigrid solver. J. Comp. and Appl. Math. **33**, 1–27 (1990)