Reducing complexity of algebraic multigrid by aggregation

S. Gratton, P. Hénon, P. Jiránek, and X. Vasseur

Technical Report TR/PA/14/18
Reducing complexity of algebraic multigrid by aggregation

Serge Gratton∗ Pascal Hénon† Pavel Jiránek‡ Xavier Vasseur§

Abstract

In order to decrease computational costs and memory requirements of relatively expensive classical algebraic multigrid (AMG) methods, we investigate its combination with aggressive coarsening schemes based on the plain (non-smoothed) aggregation on a fixed number of fine levels. Equivalently, we replace the direct solver on the coarsest level of the aggregation method with an inexact classical AMG solver. In this way, we obtain an efficient preconditioner with improved setup and solution costs and, at the same time, retain the convergence behavior of the exact plain aggregation method. The numerical experiments show the relevance of the proposed combination on both academic and benchmark problems related to reservoir simulation. A reduction factor of up to 2.92 in terms of computational time is obtained with respect to the classical algebraic multigrid method on this last application.

1 Introduction

Algebraic multigrid (AMG) methods [1, 2, 3, 4] are among the most efficient solution and preconditioning algorithms for large and sparse systems of linear equations that arise in a wide range of scientific and engineering applications governed by elliptic partial differential equations. Multigrid methods (including AMG) combine effects of smoothing and coarse-level correction. The smoothing operation usually employs a simple relaxation scheme such as the Jacobi or Gauss-Seidel iterations and attempts to suppress the high-frequency components of the error. The coarse-level correction is designed to eliminate the complementary components of the error, which are not efficiently reduced by smoothing, and consists of transferring the smoothed error on a coarser level with fewer unknowns, solving the associated residual equation (possibly approximately), and interpolating the computed correction back to the fine grid to update the current fine-level solution. In contrast to the geometric approaches (see, for example, [2, 5]), AMG constructs the multigrid hierarchy entirely by algebraic means. The AMG solvers and preconditioners can be consequently employed in a “black-box” fashion and are applicable even for problems defined on complicated geometric domains, where it might be difficult, if possible, to develop a suitable geometric multigrid solver.

In AMG, the multigrid hierarchy is constructed explicitly during the setup phase. Most of the current methods can be classified in terms of coarsening schemes as classical and aggregation AMG. The classical AMG [1, 3, 4] can be considered as an algebraic counterpart of the traditional geometric multigrid. A certain subset of the fine-level variables is identified with the variables

∗INPT-IRIT, University of Toulouse and ENSEEIHT, 2 Rue Camichel, BP 7122, 31071 Toulouse & CERFACS, 42 Avenue Gaspard Coriolis, 31057 Toulouse. E-mail: serge.gratton@enseeiht.fr.
†Total SA, Centre Scientifique et Technique Jean-Féger, Avenue Larribau, 64018 Pau, France. E-mail: pascal.henon@total.com.
‡CERFACS, 42 Avenue Gaspard Coriolis, 31057 Toulouse. E-mail: jiranek@cerfacs.fr.
§CERFACS, 42 Avenue Gaspard Coriolis, 31057 Toulouse. E-mail: vasseur@cerfacs.fr.
on the coarse level and a linear interpolation operator is deduced from the matrix entries to approximate the values in the fine points from the values in the neighboring coarse points. In the aggregation methods [6, 7, 8, 9], the coarse-level variables are associated with contiguous disjoint groups of fine-level variables called aggregates. The interpolation is defined simply by an injection of the values from the coarse-level to the fine-level variables in each aggregate. We recall the basic principles of AMG methods and coarsening schemes in Section 2.

The AMG methods based on the classical approach usually exhibit very good performance in terms of the convergence rates of preconditioned iterative methods (that is, the iteration counts to achieve a certain stopping criterion). The price to pay is however characterized by quite high computational costs and memory requirements associated with the setup and solution phases of the preconditioner. The aggregation AMG methods, on the other hand, are often relatively cheaper with low operator complexities. Nevertheless, they are rarely used in the multigrid setting because of the fact that the convergence rate of an aggregation multigrid method often depends on the problem size [7, 4, 10], that is, on the number of levels. The scalability of the aggregation methods can be improved, for example, by “enriching” the interpolation operators [11, 12, 13, 14, 15] or by using the overcorrection [7, 16] or a certain polynomial acceleration [9, 17, 18] of the coarse-level correction step.

The convergence of aggregation methods can be made independent of the problem size for the two-level method or, more generally, when the number of levels is limited; see, for example, [6, 19]. Such a framework alone is however not practical because it may require to solve possibly large coarse systems to the high level of accuracy, for example, by using a direct solver, which can be unfeasible in practice. In this paper, we consider combining the aggregation and classical AMG approaches within one hierarchy or, equivalently, replacing the “exact” coarse-level solver in the aggregation part of the hierarchy by a classical AMG solver; see Section 3. Since we use a relatively expensive coarsening scheme on a substantially smaller problem, we can achieve a considerable reduction of the overall operator complexity including reduced costs of the setup and solve phases of the resulting preconditioner. In addition, we approximately retain the optimal convergence of the “exact” aggregation method without the need to solve the coarse-level problems to a high level of accuracy.

Our work is motivated by the particular application of AMG in the context of solving linear algebraic systems arising from multi-phase flow models in reservoir simulations [20, 21, 22]. Discretization and linearization of such models lead to a large sequence of systems, which couple together the pressure and saturation or concentration unknowns in the computational cells of the underlying domain discretization and which must be solved efficiently. A possible approach for preconditioning such systems is based on decoupling the dependency of the pressure on the remaining unknowns.

A popular decoupling preconditioning method is the so-called constrained pressure reduction (CPR) [23, 24], which consists of two stages. In the first stage, the pressure subsystem is solved approximately (in this work, by a single V-cycle of the AMG method). In the second stage, an approximate global preconditioner (e.g., based on an incomplete factorization) is applied on the updated residual vector. The setup of the AMG method represents a performance bottleneck and, therefore, our goal is to reduce the costs of the AMG setup in order to improve the overall performance of the CPR preconditioner. We illustrate the performance of our approach on numerical examples in Section 4.
2 Algebraic multigrid

We briefly recall in this section the basic principles of the (algebraic) multigrid methods; see, for example, [3, 4, 5] for more details. We consider a linear algebraic system

\[ Au = f, \tag{2.1} \]

where \( A \) is a real \( N \times N \) symmetric positive definite matrix (SPD) and \( f \) and \( u \) are, respectively, the right-hand side and the unknown solution vectors. A multigrid method attempts to construct the approximations of the solution of the original problem on a hierarchy of levels associated with coarsened spaces in order to accelerate the convergence of solutions on the fine levels. The principal idea is that smooth errors, which cannot be efficiently eliminated by a relaxation method such as Jacobi or Gauss-Seidel iterations, are removed by the coarse-level correction, that is, by solving the residual equation on a coarser level and interpolating the computed correction back to the fine level in order to obtain an improved approximation.

A multigrid hierarchy is defined by a sequence of grids \( \Omega_\ell \), level operators \( A_\ell \), inter-level transfer operators \( P_\ell \) (prolongators) and \( R_\ell \) (restrictors), and the smoothing operators \( S_\ell, \text{pre} \) and \( S_\ell, \text{post} \). In the geometric multigrid, the grids can be constructed by coarsening the initial fine grid \( \Omega \equiv \Omega_1 \), for example, by doubling the mesh size in all spatial directions (or in a subset thereof leading to the so-called semi-coarsening). The operators \( A_\ell \) are then constructed by discretizing the given partial differential equation on the resulting coarse grid and the interpolation operator \( P_\ell \) is defined by the natural injection between the discrete functional spaces defined on \( \Omega_{\ell+1} \) and \( \Omega_\ell \). Finally, the smoothing operators \( S_\ell, \text{pre} \) and \( S_\ell, \text{post} \) are constructed in order to obtain an efficient multigrid cycle (that is, to suppress the error components, which are not efficiently removed by the coarse-level correction). This may be, however, problem dependent. The traditional multigrid V-cycle can be then defined as in Algorithm 2.1 (where \( L \) denotes the index of the coarsest level).

**Algorithm 2.1** Recursive multigrid V-cycle \( \text{MG}_\ell(f_\ell,v_\ell) \)

```plaintext
if \( \ell = L \) then
  solve \( A_\ell v_\ell = f_\ell \)  
else
  \( v_\ell \leftarrow S_\ell, \text{pre}(f_\ell,v_\ell) \)  (Pre-smoothing)
  \( r_{\ell+1} \leftarrow R_\ell(f_\ell - A_\ell v_\ell) \)  (Residual restriction)
  \( c_{\ell+1} \leftarrow 0 \) and do \( \text{MG}_{\ell+1}(r_{\ell+1},c_{\ell+1}) \)  (Coarse-grid correction)
  \( v_\ell \leftarrow v_\ell + P_\ell c_{\ell+1} \)  (Solution update)
  \( v_\ell \leftarrow S_\ell, \text{post}(f_\ell,v_\ell) \)  (Post-smoothing)
end if
```

In AMG, the smoothers are typically fixed to some simple relaxation schemes such as the Jacobi or Gauss-Seidel iterations and the sequence of “grids” (where the grid \( \Omega_\ell \) can be associated simply with the set of level variables \( \{1, \ldots, N_\ell\} \)) and operators is constructed from the given fine-level matrix \( A_1 = A \). It is necessary that the interpolation operators deduced from the coarsening transfer accurately those error components, which are not efficiently eliminated by the smoother. This can be accomplished by using a coarsening analogous to the semi-coarsening in the geometric multigrid, which is however purely algebraic and deduced from the given fine-level matrix.

Computing the AMG hierarchy is carried out during the setup phase, which typically follows a more or less fixed pattern depicted in Algorithm 2.2. Some components of the AMG setup are usually fixed, in particular, one takes the restrictor to be simply given by the transpose of the
Algorithm 2.2 AMG setup

1: set Ω₁ ← Ω, A₁ ← A, ℓ = 1
2: while Ωₖ is not small enough and ℓ < Lmax do
3: Construct the coarse-level variables Ωₖ₊₁
4: Compute the interpolation operator Pₖ
5: Set Rₖ = Pₖᵀ
6: Compute the coarse-level operator Aₖ₊₁ = RₖAₖPₖ
7: ℓ ← ℓ + 1
8: end while

Figure 2.1: Coarsening and interpolation in the classical (left) and aggregation (right) AMG on a regular grid.

The prolongator (Rₖ = Pₖᵀ) and the coarse-level operators are computed by the Galerkin product Aₖ₊₁ = RₖAₖPₖ in order to satisfy certain optimality criteria of the coarse-level correction. The initial two steps of the setup loop, where the coarse-level variables Ωₖ₊₁ and the interpolation operator Pₖ are constructed, give rise to whole families of the AMG methods.

2.1 Coarsening and interpolation in AMG

We briefly describe the classical and aggregation coarsening (see Figure 2.1) and some typical interpolation methods. Throughout this section, we omit the level indices for simplicity, that is, we assume an N × N SPD matrix A is given on some fine level with variables Ω = {1, ..., N}.

Classical AMG

In the classical AMG, which in a sense mimics the coarsening used in the geometric multigrid, the fine-level variables Ω = {1, ..., N} are split into two disjoint sets of fine variables (F-points) F and coarse variables (C-points) C such that Ω = F ∪ C and F ∩ C = ∅. The C-points are then identified with the coarse-level variables and the interpolation is defined by an identity mapping to the C-points and by a linear interpolation rule in F-points from the values in the neighboring C-points. Coarsening schemes in AMG usually utilize the concept of strong coupling between
variables to identify in which directions (algebraically) smooth errors vary slowly and hence in which directions the variables can be coarsened. These directions are identified with in a sense large negative off-diagonal entries. The typical criterion is as follows. With \( A = (a_{ij}) \), the variable \( i \) is said to be strongly dependent on the variable \( j \) (or equivalently that the variable \( j \) strongly influences the variable \( i \)) if

\[
-a_{ij} \geq \theta \max_{k \neq i}(-a_{ik}) \tag{2.2}
\]

for some prescribed strength threshold parameter \( \theta \) (usually, \( \theta = 0.25 \) for 2D problems or \( \theta = 0.5 \) in 3D is recommended [25]). The criterion (2.2) defines for each \( i \) the set \( S_i \) of points \( j \) strongly influencing the point \( i \).

The goal of the classical coarsening is to find a suitable splitting of the variables \( \Omega \) into the sets \( F \) of fine points (F-points) and \( C \) of coarse points (C-points) such that each F-point \( i \in F \) has a prescribed set \( C_i \subset C \) of C-points from which it interpolates, while the values in the C-points are injected from the values on the coarse level. The interpolation operator \( P \) is hence assumed in the form

\[
(Pv)_i = \begin{cases} 
  v_i & \text{if } i \in C, \\
  \sum_{j \in C_i} \omega_{ij}v_j & \text{if } i \in F.
\end{cases} \tag{2.3}
\]

The coarsening generally attempts to fulfill two contradictory criteria. In order to ensure that a chosen interpolation scheme is well-defined and of a good quality, some close neighborhood of each fine point must contain a sufficient amount of coarse points to interpolate from. Hence the set of coarse points must be rich enough. However, the set of coarse points should be sufficiently small in order to achieve a reasonable coarsening rate (the relative decrease of the number of unknowns on the coarse level).

The concrete criteria for coarsening depend on the given interpolation scheme. The interpolation rules (2.3) in classical AMG are typically based on the assumption that the smooth errors \( e \) have small residuals, that is, \( Ae \approx 0 \). Equivalently,

\[
a_{ii}e_i \approx -\sum_{j \in N_i} a_{ij}e_j, \tag{2.4}
\]

where \( N_i = \{ j : j \neq i \text{ and } a_{ij} \neq 0 \} \) is the set of points adjacent to \( i \). For illustration, we consider the classical distance-one interpolation [3], where the interpolation sets \( C_i \) are \( C_i = C \cap S_i \), that is, for each \( i \in F \), \( C_i \) is the set of the adjacent C-points strongly influencing the F-point \( i \). Now with \( F_i = F \cap S_i \) and \( W_i = N_i \setminus S_i \), (2.4) becomes

\[
a_{ii}e_i \approx -\sum_{j \in C_i} a_{ij}e_j - \sum_{j \in F_i} a_{ij}e_j - \sum_{j \in W_i} a_{ij}e_j.
\]

In the classical interpolation, one takes the approximations of the error components in \( W_i \) and \( F_i \) given, respectively, by

\[
e_j \approx e_i, \quad j \in W_i, \quad \text{and} \quad e_j \approx \frac{\sum_{k \in C_i} a_{jk}e_k}{\sum_{k \in C_i} a_{jk}}, \quad j \in F_i,
\]

which leads to the weights \( \omega_{ij} \) in (2.3)

\[
\omega_{ij} = \frac{1}{a_{ii} + \sum_{k \in W_i} a_{ik}} \left( a_{ij} + \sum_{k \in F_i} \frac{a_{ij}a_{kj}}{\sum_{l \in C_i} a_{kl}} \right), \quad i \in F, \ j \in C_i. \tag{2.5}
\]
In order to guarantee that the interpolation weights (2.5) are well defined, the splitting \( \Omega = C \cup F \) must satisfy certain criteria. In particular, for each fine point \( i \in F \) and each point \( j \) strongly influencing \( i \) (\( j \in S_i \)), either \( j \) is a C-point or (if \( j \in F_i \)) there is a C-point \( k \in C \) strongly influencing both points \( i \) and \( j \). Equivalently, each pair of strongly connected F-points must be strongly dependent on a common C-point.

The traditional sequential coarsening process [3] implements essentially a greedy coloring algorithm which attempts to satisfy the aforementioned criterion. In the first pass, an initial splitting is created by, at each step, picking a “good” C-point candidate (where the measure of “goodness” is given by the number of points influenced by the candidate) and marking all its neighbors as F-points. The process is repeated until all points are marked. The second pass forces the condition that each pair of strongly connected F-points has a common C-point to interpolate from by selectively turning certain previously marked F-points in the pairs failing to fulfill this condition to C-points.

Other coarsening schemes exist, some of them designed particularly for parallel AMG. For example, parallel graph coloring algorithms [26, 27] can be adapted for the purpose of the classical AMG coarsening [28, 29]. The advantage here is that the result of the coarsening process does not depend on the number of processors and the domain partitioning; otherwise, a careful treatment of the processor boundaries must be done in order to guarantee the sufficient quality of the coarsening. The traditional coarsening and its variants often create too many coarse points and lead to high operator complexities with slow coarsening rates. This can be avoided by relaxing the coarsening criteria, for example, by requiring only that each F-point is strongly influenced by at least one adjacent C-point. However, the interpolation rules must be carefully adjusted by taking into account wider patches \( C_i \) of neighboring C-points to interpolate from; see, for example, [30]. Further complexity decrease can be achieved by the aggressive coarsening described in [4] combined with a long-range multi-pass interpolation. We consider this scheme for comparison in our numerical experiments in Section 4.

Aggregation AMG

The aggregation methods are based on partitioning the unknowns \( \Omega \) into small disjoint contiguous sets (aggregates) \( G_j \) such that \( \Omega = \bigcup_j G_j \) (and \( G_i \cap G_j \) whenever \( i \neq j \)). The aggregates are usually created by grouping together patches of adjacent nodes of the sub-graph of \( A \) obtained by removing the edges which are not strongly connected with respect to a suitable strength criterion. For SPD problems, the criterion used in the smoothed aggregation [13] defines two distinct points \( i \) and \( j \) to be strongly connected if

\[
|a_{ij}| > \theta \sqrt{a_{ii} a_{jj}}
\]

for some threshold parameter \( \theta \) (in [13], the values \( \theta = 0.08 \cdot 0.5^{\ell-1} \), where \( \ell \geq 1 \) is the index of the level, are recommended).

Essentially any graph coarsening algorithm can be adapted for the purpose of the aggregation AMG. For example, the root point algorithm given in [13] generates aggregates in two passes. In the first pass, the initial aggregation is created from unaggregated strongly connected patches of nodes, while the second pass consists of appending the remaining unaggregated nodes to the adjacent aggregates. We use a modification of this algorithm for aggregation as well; see Section 3. Another possibility is to use a variant of pairwise matching [7, 31, 9] between the adjacent nodes. In parallel implementations, the coarsening by aggregation is often realized simply in the way that all processes aggregate independently their associated local portions of the graph as the aggregation AMG does not necessarily require any special treatment of the coarsening close to the inter-process boundaries in contrast to the classical AMG. The obvious disadvantage is that
the operator size on the coarsest level is bounded from below by the total number of processes (each containing at least one aggregate) and the coarsening rate can be slow in particular on coarser levels. For more information and coupled parallel aggregation algorithms, see [32].

The aggregates are identified with points of the coarse level, that is, the aggregate $G_j$ corresponds to the variable $j$ of the coarse level. The structure of the interpolation operator is given simply by the adjacency between the variables $\Omega$ and the aggregates. For example, in Poisson-like problems, the entries of the prolongator $P = (p_{ij})$ can be defined by

$$p_{ij} = \begin{cases} 1 & \text{if } i \in G_j, \\ 0 & \text{otherwise.} \end{cases}$$

For more general problems, the nonzero entries of $P$ are generally deduced from a provided approximate nullspace basis candidate; see [13] for more details.

The setup of the (plain) aggregation AMG is relatively fast and cheap compared to the classical AMG. The convergence of the V- or W-cycle is however strongly dependent on the problem size or more precisely, on the number of levels in the hierarchy. This dependency can be circumvented by limiting the number of levels, which is impractical due to the possibly large size of the coarsest system. One can construct instead a fixed number of levels using the aggregation and use the scalable classical AMG as the solver on the coarsest level. We investigate this alternative in the next section.

3 Combining aggregation and classical AMG

The classical AMG with a short-distance interpolation as a preconditioner (or stationary solver) often leads to very good V-cycle convergence rates for a wide range of problem sizes. However, the operator complexities characterizing approximately the memory requirements and computational costs of the V-cycle (when ignoring the costs associated with the inter-level operators), defined by

$$C_{op} = \frac{1}{\text{nz}(A_1)} \sum_{\ell=1}^{L} \text{nz}(A_{\ell}),$$

where $\text{nz}(A_{\ell})$ denotes the number of nonzero entries in the operator $A_{\ell}$ on the level $\ell$, can be relatively high. On the other hand, the plain aggregation AMG is relatively fast and cheap but if the number of levels is not kept fixed independently of the problem size, the convergence rate of the V-cycle (even of the W-cycle) may deteriorate quickly for increasingly larger problems.

It is possible to reduce the complexity of the AMG method by first performing the aggressive coarsening with a long-range interpolation [4] on a prescribed number of fine levels and then applying a less aggressive coarsening with a more accurate short-distance interpolation to create the remainder of the multigrid hierarchy. Such an approach is also implemented in the BoomerAMG package of the Hypre library [33]. Although the convergence of preconditioned iterative methods often deteriorates, the substantial decrease of the operator complexity may compensate this drawback and, consequently, the combination of these two approaches in one AMG hierarchy can provide a significant performance improvement.

Motivated by this idea, we consider replacing the classical aggressive coarsening approach by aggregation. The resulting AMG method combines a fixed number of (plain) aggregation levels with the rest of the hierarchy generated by a classical AMG scheme with a short-range interpolation. Alternatively, this can be considered as using an aggregation AMG method (with a limited number of levels) with the coarse-level solver replaced by the classical AMG.
For the aggregation part, we use a modified variant of the aggregation algorithm from [13] with a slightly different definition of the strength of connection. We consider essentially a combination of both criteria (2.6) and (2.2), where for each pair of adjacent nodes $i$ and $j$ in the graph of $A$, we define the weight $\mu_{ij}$ by

$$\mu_{ij} = \frac{|a_{ij}|}{\sqrt{|a_{ii}a_{jj}|}}$$

if $a_{ij} < 0$, $\mu_{ij} = 0$ otherwise,

and say, that $i$ and $j$ are strongly connected if

$$\mu_{ij} > \theta \min\{\max_k \mu_{ik}, \max_l \mu_{lj}\}$$

(3.1)

for some given threshold $\theta$. With this choice, we ensure that the weights are symmetric. For each node $i$, we denote by $S_i^{(1)}$ the set of strongly connected neighbors $j$ including the node $i$, that is, $S_i^{(1)} = \{i\} \cup \{j : j \neq i \text{ satisfies (3.1)}\}$.

The average coarsening rate of the aggregation algorithm used in [13] is limited by the (average) cardinalities of the strongly connected patches $S_i^{(1)}$. In order to allow for more aggressive aggregation, we consider radius-$r$ strong neighborhoods defined recursively by

$$S_i^{(r)} = \bigcup_{j \in S_i^{(r-1)}} S_j^{(1)}, \quad r > 1,$$

that is, the set $S_i^{(r)}$ contains together with the root node $i$ also the nodes contained in “strong paths” starting at $i$ of the distance at most $r$.

The aggregation procedure is described in Algorithm 3.1. It is very similar to the algorithm used in the smoothed aggregation [13] except that in the first pass, we consider the generally larger strong neighborhoods $S_i^{(r)}$. The second pass, where the unaggregated nodes are appended to adjacent aggregates, is repeated $r$-times in order to fill the “holes” created during the first pass which are generally larger than in the original variant in [13]. There are more options how to break ties in the second pass if there are more aggregates $\tilde{G}_j$ strongly connected to the node $i$, for example, one can prefer to append $i$ to the aggregate which has either the maximum number of strong connections to $i$ or contains a node $k$ maximizing the weight $\mu_{ik}$ over all aggregated neighbors of $i$ in $S_i^{(1)}$.

In the solution phase, we attempt to improve the performance on the aggregation part of the hierarchy by using the V-cycle (Algorithm 2.1) with the overcorrection [16]; see Algorithm 3.2. It is equivalent to the one-dimensional minimization of the $A_\ell$-norm of the error in $u_\ell - v_\ell$ along the smoothed correction $c_\ell$ in the last step of the algorithm.

For illustration, we use the 2D Poisson equation in the unit square with the constant right-hand side and the homogeneous Dirichlet boundary conditions discretized by the 5-point finite difference stencil on a uniform grid. We consider eight different schemes: two-level and three-level aggregation methods denoted AGG-2L and AGG-3L (where the exact coarse-level solver is imitated by an iterative method with a tight tolerance), the same configurations with the coarse-level solver replaced by the classical AMG (see the next section for the description of the parameters for the classical AMG) denoted as AGG-2L-CAMG and AGG-3L-CAMG, and the aggregation AMG using the V-cycle and W-cycle with (AGG-V$^+$ and AGG-W$^+$) and without (AGG-V and AGG-W) the overcorrection step. In Table 3.1, we report the number of iterations required to achieve the relative residual norm tolerance $10^{-6}$ in the flexible GMRES (FGMRES) method [34] with the restart parameter set to 50 (we use FGMRES here because the preconditioners with the overcorrection are not generally fixed). The experiments were performed on
Algorithm 3.1 Radius-\( r \) root point aggregation procedure

Set \( U \leftarrow \Omega \setminus \{ i : S_i^{(r)} = \{ i \} \} \), \( k = 0 \)

for \( i \in U \) do 
  if \( S_i^{(r)} \subset U \) then 
    \( k \leftarrow k + 1 \)
    \( G_k \leftarrow S_i^{(r)} \)
    \( U \leftarrow U \setminus G_k \)
  end if
end for

for \( \rho = 1, \ldots, r \) do 
  \( G_j \leftarrow G_j \) for \( j = 1, \ldots, k \)
  for \( i \in U \) do 
    if \( \exists j \) such that \( \bar{G}_j \cap S_i^{(1)} \) then 
      \( G_i \leftarrow G_i \cup \{ i \} \)
      \( U \leftarrow U \setminus \{ i \} \)
    end if
  end for
end for

Algorithm 3.2 Recursive multigrid V-cycle \( \bar{\text{MG}}_\ell (f_\ell, v_\ell) \) with overcorrection

if \( \ell = L \) then 
  solve \( A v_L = f_L \)
else 
  \( v_\ell \leftarrow S_\ell, \text{pre}(f_\ell, v_\ell) \)
  \( r_\ell \leftarrow f_\ell - A v_\ell \)
  \( r_{\ell+1} \leftarrow R \ell r_\ell \)
  \( c_{\ell+1} \leftarrow 0 \) and do \( \bar{\text{MG}}_{\ell+1}(r_{\ell+1}, c_{\ell+1}) \)
  \( c_\ell \leftarrow P_{\ell+1} c_{\ell+1} \)
  \( v_\ell \leftarrow v_\ell + \alpha c_\ell \) with \( \alpha \leftarrow r_\ell^T c_\ell / c_\ell^T A c_\ell \)
end if
a single node of the computer described in the next section. We observe that the performance of both the exact and inexact two- and three-level methods scale very well with the problem size and with the inexact methods we almost achieve the same iteration numbers as with the exact counterparts; see [35, Corollary 5.11] for theoretical justification. It is also clear that using the aggregation AMG with the V- or W-cycle does not provide a scalable preconditioner. The situation can be improved by using the overcorrection, in particular, for the W-cycle. In this case, we actually obtain a cycle, which is very closely related to the K-cycle [18]. We would like however to avoid using the W-cycle or a related type of cycle in this work.

<table>
<thead>
<tr>
<th>Method</th>
<th>Problem size N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64²</td>
</tr>
<tr>
<td>AGG-2L</td>
<td>11</td>
</tr>
<tr>
<td>AGG-2L-CAMG</td>
<td>12</td>
</tr>
<tr>
<td>AGG-3L</td>
<td>12</td>
</tr>
<tr>
<td>AGG-3L-CAMG</td>
<td>12</td>
</tr>
<tr>
<td>AGG-V</td>
<td>25</td>
</tr>
<tr>
<td>AGG-V+</td>
<td>12</td>
</tr>
<tr>
<td>AGG-W</td>
<td>21</td>
</tr>
<tr>
<td>AGG-W+</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 3.1: Dependence of iteration counts of different methods on the problem size.

4 Numerical experiments

We empirically compare in this section the performance of the combination of the aggregation and classical AMG schemes on two types of problems. We test the weak scalability on simple isotropic and anisotropic Poisson problems in 3D and the strong scalability on sequences of linear systems from two benchmark problems arising in reservoir simulations. The test problems were run on the Beaufix computer located at Météo-France in Toulouse, which is a BULL cluster with 1080 computational nodes, each consisting of two sockets equipped with the 12-core Intel Xeon processor (Ivy-Bridge).

We implemented the aggregation algorithms in our AMG framework developed in C++ and based on the parallel algebraic kernels provided by the Epetra package of the Trilinos library [36]. As we apply the aggregation AMG only on a fixed number of fine levels, we consider only the “decoupled” variant of Algorithm 3.1 in our parallel implementation. That is, the aggregation procedure is applied independently to the local portion of the matrix A on each process. The current implementation of our AMG code does not use local threads (partly because of the limited OpenMP support in Epetra). All reported tests were thus executed with only one MPI task per node in order to avoid the excessive amount of communication messages in particular on coarse levels (we plan to implement a threaded version in the future). We denote by “#nodes” the total number of nodes used (the number of processes is hence twice that much).

We consider three configurations of the AMG preconditioner. For the classical AMG (denoted by CAMG), we use the BoomerAMG implementation using the strength threshold parameter $\theta = 0.5$ in (2.2), the PMIS coarsening algorithm [29] together with the classical distance-one interpolation combined with the extended distance-two interpolation [30] in these pairs of F-points, which do not share a common C-point. Compared to the classical coarsening and its variants like CLJP or Falgout-CLJP [33], this choice achieves similar convergence properties...
with relatively faster coarsening rates and lower operator complexities. The coarsening is stopped when the size of the coarse-level matrix does not exceed 9 (the default value in BoomerAMG). We denote by AGG the combined aggregation with the classical AMG, where the first coarse level is created using the aggregation scheme described in the previous section (with the radius of the aggregates equal to two in Algorithm 3.1 and \( \theta = 0.5 \) in (3.1)) and the CAMG scheme on the remainder of the hierarchy. In order to compare our approach with the classical aggressive coarsening, we also consider a variant denoted by AGGRES, where the aggregation method is replaced by this type of coarsening with the multi-pass interpolation.

In the solve phase, the hybrid symmetric Gauss-Seidel relaxation [37] is employed as both the pre-smoother and post-smoother in the whole hierarchy. The hybrid Gauss-Seidel relaxation is essentially the ordinary Gauss-Seidel method run independently on each process exchanging information from the neighboring processors after each iteration (that is, it can be considered as an approximation of the block Jacobi method with the blocks being the block diagonal parts of the local portions of the distributed matrix).

**Weak scalability experiments.** We first test the performance and weak scalability properties of the three AMG preconditioners on the Poisson problem

\[-\nabla \cdot (K \nabla u) = 1\]

in the three-dimensional unit cube with the homogeneous Dirichlet boundary conditions discretized by the 7-point stencil on a uniform regular grid, where each computational node contains \( 10^6 \) grid points (the processes themselves are also organized in a cube). We consider \( K = I \) corresponding to the isotropic problem and \( K = \text{diag}(100, 1, 1) \) for the anisotropic one. The AMG methods are used as preconditioners for FGMRES (with the restart parameter 20) due to the overcorrection used in the part of the multigrid hierarchy created by the aggregation, which leads to a variable preconditioner. The iterations are stopped when the relative residual norm decreases below the tolerance \( 10^{-6} \).

For each test problem, we report in Tables 4.1 and 4.2 the following quantities: the setup time “Setup” of the AMG preconditioner, the solution time “Solve” including the overhead from the FGMRES method, and the overall time “Total”, the number of levels “#lvls”, the operator complexity \( C_{op} \), the number of iterations “#iters” to achieve the prescribed relative residual norm, and the “speed-up factor” \( \tau \) given by the ratio of the total time associated with a given preconditioner and the AGG variant executed on the same number of nodes. In Figure 4.1, we report the setup and solution times with all three AMG configurations with respect to the number of computational nodes.

We observe that using both the aggregation or the aggressive coarsening on the finest level leads to considerable reductions of both the setup and solution times as well as the operator complexities of the multigrid hierarchies with slightly better results for the aggregation. Note that although using both AGG and AGGRES as preconditioners leads to higher numbers of iterations required to achieve the given stopping criterion of FGMRES, the solution phase is faster due to lower operator complexities in comparison with the classical AMG preconditioner.

It is worth pointing out that the reductions of setup and solution times and operator complexities are much more apparent in the isotropic case compared to the anisotropic one essentially for two reasons. Firstly, the classical AMG applied on the anisotropic problem constructs generally much sparser interpolation operators (and consequently the coarse-level operators) on finer levels in contrast to the same method applied to the isotropic problem (although with somewhat slower coarsening in the anisotropic case). Consequently, the operator complexities of the classical AMG corresponding to anisotropic problems are lower than in the isotropic case. Secondly, the behavior of the aggregation method is completely opposite as the presence of the anisotropy
<table>
<thead>
<tr>
<th>Method</th>
<th>#nodes</th>
<th>Setup</th>
<th>Solve</th>
<th>Total</th>
<th>$\tau$</th>
<th>#lvs</th>
<th>$C_{op}$</th>
<th>#iters</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGG</td>
<td>1</td>
<td>0.791</td>
<td>2.671</td>
<td>3.463</td>
<td>1.00</td>
<td>8</td>
<td>1.189</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.976</td>
<td>3.390</td>
<td>4.365</td>
<td>1.00</td>
<td>9</td>
<td>1.204</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>1.302</td>
<td>3.726</td>
<td>5.028</td>
<td>1.00</td>
<td>9</td>
<td>1.210</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>3.455</td>
<td>4.225</td>
<td>7.680</td>
<td>1.00</td>
<td>10</td>
<td>1.213</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>4.201</td>
<td>4.547</td>
<td>8.748</td>
<td>1.00</td>
<td>11</td>
<td>1.215</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>3.124</td>
<td>4.682</td>
<td>7.806</td>
<td>1.00</td>
<td>11</td>
<td>1.216</td>
<td>20</td>
</tr>
<tr>
<td>AGGRES</td>
<td>1</td>
<td>3.116</td>
<td>2.790</td>
<td>5.906</td>
<td>1.70</td>
<td>8</td>
<td>1.457</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3.376</td>
<td>3.470</td>
<td>6.847</td>
<td>1.57</td>
<td>9</td>
<td>1.488</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>4.708</td>
<td>4.061</td>
<td>8.769</td>
<td>1.74</td>
<td>10</td>
<td>1.498</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>5.414</td>
<td>4.325</td>
<td>9.739</td>
<td>1.27</td>
<td>10</td>
<td>1.503</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>5.778</td>
<td>4.613</td>
<td>10.391</td>
<td>1.19</td>
<td>11</td>
<td>1.505</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>6.108</td>
<td>4.946</td>
<td>11.054</td>
<td>1.42</td>
<td>11</td>
<td>1.507</td>
<td>22</td>
</tr>
<tr>
<td>CAMG</td>
<td>1</td>
<td>10.108</td>
<td>3.543</td>
<td>13.650</td>
<td>3.94</td>
<td>8</td>
<td>5.058</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>20.367</td>
<td>4.408</td>
<td>24.776</td>
<td>5.68</td>
<td>9</td>
<td>5.208</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>31.459</td>
<td>5.149</td>
<td>36.607</td>
<td>7.28</td>
<td>10</td>
<td>5.264</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>32.550</td>
<td>6.301</td>
<td>38.851</td>
<td>5.06</td>
<td>11</td>
<td>5.290</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>33.178</td>
<td>6.363</td>
<td>39.541</td>
<td>4.52</td>
<td>11</td>
<td>5.390</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>34.547</td>
<td>6.905</td>
<td>41.452</td>
<td>5.31</td>
<td>12</td>
<td>5.474</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 4.1: Results for the 3D isotropic Poisson problem (weak scalability test, $10^6$ grid points per node).

reduces the sizes of aggregates on fine levels and thus leads to a mild increase of the operator complexity.

**Strong scalability experiments.** Next, we test the strong scalability of the three AMG schemes on public domain matrices arising in reservoir simulation problems. The problems represent coupled pressure-saturation PDE systems and are preconditioned by a variant of the constrained pressure reduction (CPR) preconditioner [23, 24]. The CPR preconditioner is a two-stage preconditioner based on an explicit decoupling of the dependence of the pressure on the saturation. In the first stage, the pressure subsystems are solved approximately by applying a single V-cycle of the AMG preconditioner and, with the updated pressure solution component, the pressure-saturation components are further updated by using a global block ILU preconditioner in the second stage. The FGMRES method (with the restart parameter set to 20) is used as the outer solver with the relative residual tolerance set to $10^{-3}$ as such a relaxed tolerance is usually sufficient in the context of solving linearized problems.

We consider two test problems: the highly heterogeneous and anisotropic SPE10 benchmark problem [38] with 1094421 pressure unknowns and the anisotropic homogeneous problem denoted as QUARTER5SPOT with 2700000 pressure unknowns. We have 8 systems for SPE10 and 5 systems for QUARTER5SPOT arising in the non-linear Newton solver during a single time step of the fully implicit formulation. In addition to the schemes used in the weak scalability experiments, we consider a variant of AGG denoted as AGG* identical to AGG for the first linear system in the sequence but reusing the aggregation part of the hierarchy for updating the AMG solver for each subsequent system (that is, for each subsequent system, we recompute only the coarse-level operators, smoothers, and the coarse-level CAMG solver).

The results are summarized in Tables 4.3 and 4.4. We report both the setup and solution times of the AMG part of the CPR preconditioner. In addition, we report the total setup time.
<table>
<thead>
<tr>
<th>Method</th>
<th>#nodes</th>
<th>Setup</th>
<th>Solve</th>
<th>Total</th>
<th>$\tau$</th>
<th>#lvs</th>
<th>$C_{\text{op}}$</th>
<th>#iters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1.072</td>
<td>3.486</td>
<td>4.559</td>
<td>1.00</td>
<td>8</td>
<td>1.545</td>
<td>15</td>
</tr>
<tr>
<td>AGG</td>
<td>8</td>
<td>2.774</td>
<td>4.576</td>
<td>7.350</td>
<td>1.00</td>
<td>9</td>
<td>1.610</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>5.201</td>
<td>5.097</td>
<td>10.298</td>
<td>1.00</td>
<td>10</td>
<td>1.635</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>6.609</td>
<td>5.599</td>
<td>12.208</td>
<td>1.00</td>
<td>10</td>
<td>1.650</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>7.474</td>
<td>5.973</td>
<td>13.448</td>
<td>1.00</td>
<td>10</td>
<td>1.659</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>7.839</td>
<td>5.910</td>
<td>13.749</td>
<td>1.00</td>
<td>11</td>
<td>1.665</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.701</td>
<td>3.873</td>
<td>5.573</td>
<td>1.22</td>
<td>8</td>
<td>1.688</td>
<td>18</td>
</tr>
<tr>
<td>AGGRES</td>
<td>8</td>
<td>5.848</td>
<td>5.146</td>
<td>10.994</td>
<td>1.50</td>
<td>9</td>
<td>1.744</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>9.328</td>
<td>5.793</td>
<td>15.121</td>
<td>1.47</td>
<td>10</td>
<td>1.781</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>9.128</td>
<td>6.113</td>
<td>15.241</td>
<td>1.25</td>
<td>10</td>
<td>1.781</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>10.722</td>
<td>6.767</td>
<td>17.489</td>
<td>1.27</td>
<td>11</td>
<td>1.794</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.307</td>
<td>1.884</td>
<td>4.191</td>
<td>0.92</td>
<td>9</td>
<td>2.797</td>
<td>6</td>
</tr>
<tr>
<td>CAMG</td>
<td>8</td>
<td>9.365</td>
<td>2.487</td>
<td>11.852</td>
<td>1.61</td>
<td>10</td>
<td>2.887</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>17.101</td>
<td>3.021</td>
<td>20.123</td>
<td>1.95</td>
<td>11</td>
<td>2.923</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>18.611</td>
<td>3.441</td>
<td>22.052</td>
<td>1.81</td>
<td>11</td>
<td>2.941</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>18.801</td>
<td>3.477</td>
<td>22.279</td>
<td>1.66</td>
<td>12</td>
<td>2.952</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>19.716</td>
<td>3.913</td>
<td>23.630</td>
<td>1.72</td>
<td>12</td>
<td>2.959</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.2: Results for the 3D anisotropic Poisson problem (weak scalability test, $10^6$ grid points per node).

of CPR (the construction of the AMG and ILU preconditioners) and the solution time which includes both the solution time of AMG, remainder of the CPR solve, and the overhead in the FGMRES method. The speed-up factor $\tau$ is evaluated similarly as in the weak scalability experiments but with respect to the AGG method, which gives the best overall timing results. We also show the average operator complexity denoted by $C_{\text{op}}$ for each run and the cumulative number of iterations of the FGMRES method “#iters”.

For both problems, we again observe reasonable reductions of the setup and solution times associated with the AMG component of the CPR preconditioner similarly as in the tests with the anisotropic Poisson equation. In the case of the SPE10 problem, even though the aggregation and aggressive coarsening schemes lead to an increase of the (cumulative) iteration numbers, the corresponding solution times are lower or comparable to the variant using the CAMG configuration.

In Figure 4.2, we summarize the timing results from Tables 4.3 and 4.4. Each bar consists of four components. The bottom two components represent the AMG setup time and remainder of the CPR setup time (so that the total height of the two bottom stacks corresponds to the overall setup time of the CPR preconditioner). Similar representation is used for the solution time illustrated by the top two components of each bar. The times in the plots are multiplied by the number of nodes so that the timings associated with a particular method should ideally remain constant. In our tests, the setup times of AGG and AGG* are considerably reduced in comparison with the setup times of the classical AMG method. In terms of total time of AMG, the aggregation leads to improvements of 2.01-2.92 for SPE10 and 1.62-2.54 for QUARTER5SPOT.
## Table 4.3: Results for the SPE10 benchmark (strong scalability tests, 8 linear systems).

<table>
<thead>
<tr>
<th>Method</th>
<th>#nodes</th>
<th>AMG</th>
<th>CPR</th>
<th>$\overline{C_{op}}$</th>
<th>#iters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Setup</td>
<td>Solve</td>
<td>$\tau$</td>
<td>Setup</td>
</tr>
<tr>
<td>AGG</td>
<td>8</td>
<td>1.405</td>
<td>1.804</td>
<td>1.24</td>
<td>2.711</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.845</td>
<td>1.019</td>
<td>1.23</td>
<td>1.499</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.481</td>
<td>0.578</td>
<td>1.19</td>
<td>0.811</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>0.392</td>
<td>0.424</td>
<td>1.19</td>
<td>0.553</td>
</tr>
<tr>
<td>AGG*</td>
<td>8</td>
<td>0.801</td>
<td>1.790</td>
<td>1.00</td>
<td>2.116</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0.502</td>
<td>1.014</td>
<td>1.00</td>
<td>1.165</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.308</td>
<td>0.579</td>
<td>1.00</td>
<td>0.644</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>0.267</td>
<td>0.416</td>
<td>1.00</td>
<td>0.429</td>
</tr>
<tr>
<td>AGGRES</td>
<td>8</td>
<td>2.758</td>
<td>2.369</td>
<td>1.98</td>
<td>4.085</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>1.566</td>
<td>1.278</td>
<td>1.88</td>
<td>2.230</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.816</td>
<td>0.705</td>
<td>1.71</td>
<td>1.154</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>0.682</td>
<td>0.430</td>
<td>1.63</td>
<td>0.842</td>
</tr>
<tr>
<td>CAMG</td>
<td>8</td>
<td>4.605</td>
<td>2.951</td>
<td>2.92</td>
<td>5.910</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2.623</td>
<td>1.555</td>
<td>2.76</td>
<td>3.285</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1.250</td>
<td>0.852</td>
<td>2.37</td>
<td>1.582</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>0.879</td>
<td>0.491</td>
<td>2.01</td>
<td>1.040</td>
</tr>
</tbody>
</table>

## Table 4.4: Results for the QUARTER5SPOT benchmark (strong scalability tests, 5 linear systems).

<table>
<thead>
<tr>
<th>Method</th>
<th>#nodes</th>
<th>AMG</th>
<th>CPR</th>
<th>$\overline{C_{op}}$</th>
<th>#iters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Setup</td>
<td>Solve</td>
<td>$\tau$</td>
<td>Setup</td>
</tr>
<tr>
<td>AGG</td>
<td>8</td>
<td>3.084</td>
<td>2.133</td>
<td>1.19</td>
<td>5.658</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>1.608</td>
<td>1.069</td>
<td>1.13</td>
<td>2.883</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1.018</td>
<td>0.573</td>
<td>1.23</td>
<td>1.657</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>0.672</td>
<td>0.336</td>
<td>1.02</td>
<td>0.996</td>
</tr>
<tr>
<td>AGG*</td>
<td>8</td>
<td>2.424</td>
<td>2.134</td>
<td>1.00</td>
<td>4.992</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>1.285</td>
<td>1.079</td>
<td>1.00</td>
<td>2.561</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.725</td>
<td>0.569</td>
<td>1.00</td>
<td>1.369</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>0.649</td>
<td>0.342</td>
<td>1.00</td>
<td>0.985</td>
</tr>
<tr>
<td>AGGRES</td>
<td>8</td>
<td>3.627</td>
<td>2.838</td>
<td>1.48</td>
<td>6.201</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>1.765</td>
<td>1.350</td>
<td>1.32</td>
<td>3.039</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>0.947</td>
<td>0.693</td>
<td>1.27</td>
<td>1.594</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>0.580</td>
<td>0.394</td>
<td>0.98</td>
<td>0.902</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1.708</td>
<td>1.082</td>
<td>2.16</td>
<td>2.349</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>1.022</td>
<td>0.585</td>
<td>1.62</td>
<td>1.345</td>
</tr>
</tbody>
</table>

Table 4.3: Results for the SPE10 benchmark (strong scalability tests, 8 linear systems).

Table 4.4: Results for the QUARTER5SPOT benchmark (strong scalability tests, 5 linear systems).
5 Conclusions

Classical AMG approaches often lead to preconditioners of good quality for the price of high operator complexities and expensive multigrid cycles. Aggregation methods, on the other hand, are relatively cheap but efficient only with a limited number of levels unless they are combined with a suitable post-processing of the prolongation operators such as in smoothed aggregation leading generally to denser matrices and higher complexities or with more complex multigrid cycles such as K-cycles which often require to be applied in a W-cycle fashion. We proposed to combine both algorithmic approaches in a single multigrid hierarchy, that is, first employing an aggregation coarsening to generate a fixed number of fine levels and continuing further the coarsening process by using a classical AMG scheme. By this strategy, we essentially retained the good convergence rates of the aggregation (with a fixed number of levels) and improved the costs of the setup and solution phases due to the relatively cheap coarsening on the finest levels leading to multigrid hierarchies with lower complexities. In addition, this combination has very
similar effects to the use of the aggressive coarsening combined with the multi-pass interpolation.

Acknowledgements

The authors would like to acknowledge Météo-France for the dotation of computing hours on the Beaufix cluster at Toulouse, France. They also would like to acknowledge GENCI (Grand Equipement National de Calcul Intensif) for the dotation of computing hours on the Curie cluster at CCRT, France. This work was granted access to the HPC resources of CCRT under allocation 2013065068 and 2014065068 made by GENCI.

References


