Scalability of Classical Algebraic Multigrid for Elasticity to Half a Million Parallel Tasks

Allison H. Baker, Axel Klawonn, Tzanio Kolev, Martin Lanser, Oliver Rheinbach, and Ulrike Meier Yang

Abstract The parallel performance of several classical Algebraic Multigrid (AMG) methods applied to linear elasticity problems is investigated. These methods include standard AMG approaches for systems of partial differential equations such as the unknown and hybrid approaches, as well as the more recent global matrix (GM) and local neighborhood (LN) approaches, which incorporate rigid body modes (RBMs) into the AMG interpolation operator. Numerical experiments are presented for both two- and three-dimensional elasticity problems on up to 131,072 cores (and 262,144 MPI processes) on the Vulcan supercomputer (LLNL, USA) and up to 262,144 cores (and 524,288 MPI processes) on the JUQUEEN supercomputer (JSC, Jülich, Germany). It is demonstrated that incorporating all RBMs into the interpolation leads generally to faster convergence and improved scalability.

1 Introduction

Classical Algebraic Multigrid (AMG) methods were originally designed for scalar partial differential equations (PDEs) and usually assume that the nullspace of the operator is one-dimensional and constant. This assumption does not hold for many systems of PDEs. For elasticity problems in particular, the nullspace consists of three (in 2D) or six (in 3D) rigid body modes (RBMs), which comprise translations

A.H. Baker (✉)
National Center for Atmospheric Research, Boulder, CO, USA
e-mail: abaker@ucar.edu

A. Klawonn (✉) • M. Lanser (✉)
Mathematisches Institut, Universität zu Köln, Köln, Germany
e-mail: axel.klawonn@uni-koeln.de; martin.lanser@uni-koeln.de

T. Kolev (✉) • U.M. Yang (✉)
Lawrence Livermore National Laboratory, Livermore, CA, USA
e-mail: tzanio@llnl.gov; umyang@llnl.gov

O. Rheinbach (✉)
Institut für Numerische Mathematik und Optimierung, Technische Universität Bergakademie Freiberg, Freiberg, Germany
e-mail: oliver.rheinbach@math.tu-freiberg.de

H.-J. Bungartz et al. (eds.), *Software for Exascale Computing – SPPEXA 2013-2015*, Lecture Notes in Computational Science and Engineering 113,
DOI 10.1007/978-3-319-40528-5_6

and rotations. Classical AMG methods, including standard approaches modified to handle systems of PDEs, e.g., the unknown approach [23], interpolate translations but not rotations. This limitation will typically result in a loss of optimality and scalability for these approaches when applied to systems problems.

Different approaches to handle linear elasticity problems with AMG methods have been suggested in the last decades, e.g., smoothed aggregation [7, 27], unsmoothed aggregation [3, 4, 8, 19–21], AMGe [6], element-free AMGe [15], local optimization problems to incorporate the RBMs in the interpolation [13], or the global matrix (GM) and local neighborhood (LN) approaches [2].

In this paper, we provide a brief overview of AMG methods and AMG for systems in Sects. 2 and 3. In Sects. 4 and 5, we describe the GM and LN approaches, which were first introduced in [2]. These two approaches explicitly incorporate given smooth error vectors into the AMG interpolation in order to handle the correction of these error components in the coarse grid correction. We note that the descriptions of the AMG methods and interpolations in this paper are based on both [2] (which only considered sequential AMG) and on Chap. 4 of the dissertation [18]. In Sect. 6, we compare the performance of AMG approaches for systems of PDEs and show that the GM and LN approaches can improve convergence and scalability for elasticity problems. The parallel numerical results on up to half a million MPI processes presented in Sect. 6 are new and have not been published elsewhere (as [2] contained only serial results for small problems).

2 Algebraic Multigrid

We first give a brief overview of AMG. Consider the linear system $Au = f$, which is often generated from the discretization of a scalar PDE. We denote with u_i the i-th entry of u. As in geometric multigrid, one needs to define a hierarchy of coarser grids or levels, adequate smoothers or relaxation schemes for each level and restriction, and interpolation operators to move between levels. However, unlike geometric multigrid, algebraic multigrid methods are applied to the linear system without any geometrical or mesh-related information.

Because grid information is not given, one needs to use the linear system to define a "grid". The variables u_i are now the grid points and the non-zero entries a_{ij} of matrix A define the connections between the grid points. Because not all variables are equally important, one defines the concept of *strong dependence*. For a threshold $0 < \theta \leq 1$, a variable u_i *strongly depends* on the variable u_j if

$$-a_{ij} \geq \theta \max_{k \neq i}(-a_{ik}) \ . \tag{1}$$

To determine the coarse-grid variables, which are a subset of the variables u_i, one first eliminates all connections that do not fulfill (1). Then one applies a coarsening algorithm to the remaining "grid". For brevity, we do not describe any coarsening

algorithms here, but note that descriptions of several common coarsening strategies and an investigation of their parallel performance can be found in [28] (e.g., Ruge-Stüben [23, 25], HMIS [11] and Falgout [16]).

In AMG, errors are reduced by two separate operations: the smoothing or relaxation steps and the coarse grid correction. For an optimal AMG method, the coarse grid correction and the relaxation strategy must be chosen carefully to complement each other. While simple pointwise relaxation methods such as Jacobi or Gauß-Seidel rapidly reduce errors in the directions of eigenvectors associated with large eigenvalues, the reduction in directions of eigenvectors associated with small eigenvalues is less optimal; see [6] for details. Errors that are poorly reduced by the smoothing steps are referred to as *smooth errors*. More precisely, *algebraic smooth errors* can be characterized by $Ae \approx 0$, where e is an eigenvector associated with a small eigenvalue. For an effective AMG method, the smooth error must be reduced by the coarse grid correction. Therefore, an interpolation operator P needs to be defined in such a way that the smooth errors are approximately in the range of P [6]. For additional details on interpolation operators, we refer the reader to various publications, e.g. [12, 23, 25, 26, 29]. The restriction operator R is often defined to be the transposed operator P^T, so that in the case of a symmetric positive definite matrix A, the coarse grid operator RAP is also symmetric positive definite. After interpolation, restriction, and coarse grid operators have been defined and a relaxation strategy has been determined, the solve phase can be performed.

For simplicity, consider the two-level case with one fine grid and one coarse grid. For an approximate solution u and the exact solution u^* of the system $Au^* = f$ on the fine grid, we have the relationship $Ae = r$, where $e := u^* - u$ is the error vector and $r := f - Au$ is the residual. One AMG cycle to correct (or update) u is as follows:

(1) Smooth ν_1 times on: $Au = f$
(2) Compute the residual: $r = f - Au$
(3) Solve on the coarse grid: $RAPe_c = Rr$
(4) Correct u: $u = u + Pe_c$
(5) Smooth ν_2 times on: $Au = f$.

T obtain a full multi-level AMG V-cycle, one needs to apply this algorithm recursively, as depicted in Fig. 1. For more details on classical AMG methods, see, e.g., [23, 25].

3 Algebraic Multigrid for Systems of PDEs

We now consider a linear system of equations $Au = f$ derived from the discretization of a system of PDEs with p scalar functions or unknowns. Now, each variable or degree of freedom (dof) of the linear system describes one physical quantity in

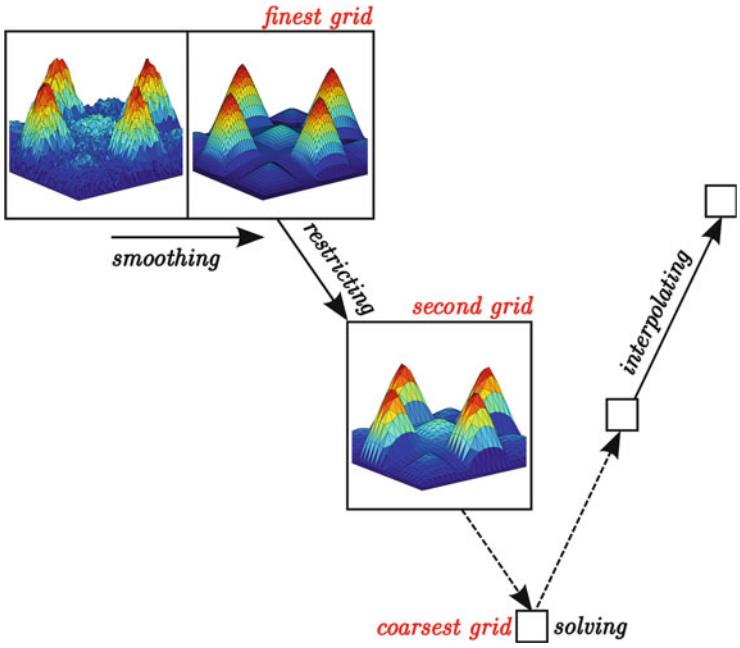

Fig. 1 One AMG V-cycle. Smoothing on the fine grid → Restricting to the coarsest grid → Solving on coarsest grid → Interpolating to the finest grid (Figure from [18])

a grid point or node. For example, in linear or nonlinear elasticity, we have one dof describing one spatial direction in each node. For simplicity, we restrict our presentation here to the two-dimensional case and consider an elasticity problem with two unknowns, x and y, representing the two spatial directions. A detailed three-dimensional description can be found in [2].

For algebraic multigrid methods, the two common approaches to treating systems of PDEs such as $Au = f$ are the *unknown approach (U-AMG)* and the nodal approach, e.g., [1, 10, 14, 22, 23, 25]. While U-AMG completely separates the different physical quantities, the nodal approach considers all unknowns belonging to the same node at once and thus acts on a nodal basis.

We first take a brief look at the U-AMG. Here, we assume an unknown-related ordering of the system matrix (i.e., first all dofs related to the unknown x followed by those associated with y):

$$A = \begin{bmatrix} A_{xx} & A_{xy} \\ A_{yx} & A_{yy} \end{bmatrix} . \tag{2}$$

One now applies classical AMG coarsening and interpolation strategies to the different variables separately, i.e., only to the diagonal blocks A_{xx} and A_{yy}. Note that this strategy ignores couplings between unknowns x and y, which are contained

in A_{xy} and A_{yx}, and leads to an AMG interpolation matrix P that has the diagonal block structure

$$P = \begin{bmatrix} P_x & 0 \\ 0 & P_y \end{bmatrix}. \tag{3}$$

In general, U-AMG is often used to handle systems of PDEs and is quite effective for problems with weak coupling between the different unknowns. Of course, performance also strongly depends on the general quality of the chosen coarsening, interpolation, and smoothing techniques for the diagonal blocks A_{xx} and A_{yy}.

We now describe the nodal approach, which is often a more effective approach for problems with a stronger coupling between the different physical quantities. If we block all unknowns that share the same node and consider a node-related ordering, then the system matrix A can be written as

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{bmatrix}, \tag{4}$$

where the 2×2 blocks A_{ij} connect nodes i and j. Note that if we define N as the number of nodes or grid points, then A is a $N \times N$ block matrix. With the nodal approach, we consider strong dependence between two nodes i and j, instead of between two variables. Therefore, we now have to compare block entries, such as A_{ji} or A_{jj}. This comparison typically involves an appropriate norm such as the Frobenius norm $||\cdot||_F$ or the row-sum norm $||\cdot||_\infty$. Applying the norm to the blocks of the system matrix A results in a condensed $N \times N$ matrix with scalar entries

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \dots & c_{NN} \end{bmatrix} := \begin{bmatrix} ||A_{11}|| & ||A_{12}|| & \cdots & ||A_{1N}|| \\ ||A_{21}|| & ||A_{22}|| & \cdots & ||A_{2N}|| \\ \vdots & \vdots & \ddots & \vdots \\ ||A_{N1}|| & ||A_{N2}|| & \dots & ||A_{NN}|| \end{bmatrix}. \tag{5}$$

The definition of strong dependence in Eq. (1) is based on A or C being an M-matrix, i.e., a matrix whose off-diagonal elements have the opposite sign of the diagonal elements. Therefore, we change the diagonal elements c_{ii} of C to $c_{ii} = -||A_{ii}||$ or

$$c_{ii} = -\sum_{j=1, j\neq i}^{N} ||A_{ij}||. \tag{6}$$

This approach as well as additional options for defining C are further discussed in [10]. In our experiments, we found the latter approach (i.e., the row-sum norm) to give better convergence, and we are using Eq. (6) in the numerical results presented

in Sect. 6. The AMG coarse grids are now obtained by applying classical AMG coarsening techniques to the condensed matrix C. In the nodal coarsening approach, all unknowns on one grid point share the same set of coarse grids. Note the contrast with the unknown approach, which can result in completely different coarse meshes for each unknown. The interpolation matrix in the nodal approach can be obtained by applying scalar AMG interpolation techniques to the blocks (e.g., [14]). Another option, used in our experiments in Sect. 6, is to combine nodal coarsening with unknown-based interpolation. We call this approach the *hybrid approach (H-AMG)*.

4 The Global Matrix Approach

As mentioned in Sect. 2, smooth error vectors should be in the range of the interpolation operator. In the case of linear elasticity, the nullspace of the matrix A consists of the rigid body modes (RBMs), i.e., all rotations and translations of the domain. Since classical AMG interpolation P already interpolates constant vectors exactly, we only have to take care of rotations (i.e., in two dimensions, the single rotation $s(x, y) := [y, -x]$). A possible approach to incorporate an exact interpolation of smooth error vectors in the AMG interpolation is, as already mentioned, the *global matrix* (GM) approach, introduced in [2]. The following description is restricted to two levels. A generalization to the multilevel-case can be found in [2].

The GM approach is based on the idea of augmenting a given global AMG interpolation matrix P with several matrices Q^j. Each matrix Q^j is chosen to exactly interpolate a specified smooth error vector s_j. We designate the rotation $s := [y, -x]$ in two dimensions as the smooth error vector. We define s_C as the restriction of s onto the coarse grid and define a new interpolation matrix $\tilde{P}$ by augmenting P:

$$\widetilde{P} := [P \quad Q], \text{ such that } s \in \text{range}(\widetilde{P}) \,. \tag{7}$$

There are several possibilities to define a matrix Q fulfilling Eq. (7) and also retain the sparsity of P. We will consider both variants suggested in [2]. For *Variant 1* or *GM1* we define $\widetilde{P}$ such that

$$\widetilde{P} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = s \,, \tag{8}$$

whereas for *Variant 2* or *GM2*, $\widetilde{P}$ is defined such that

$$\widetilde{P} \begin{bmatrix} s_C \\ 1 \end{bmatrix} = s \,. \tag{9}$$

For GM1, the coefficients Q_{ij} of Q, where i is the index of a fine grid point and j the index of a coarse grid point, are then defined as

$$Q_{ij} := P_{ij}\Big(\frac{s_i}{\sum\limits_{k\in C_i} P_{ik}}\Big)\,, \tag{10}$$

where C_i is the set of coarse points in the direct neighborhood of i, i.e., the indices of the columns with non-zero entries in row i of the interpolation P. For GM2, the entries Q_{ij}, are given by

$$Q_{ij} := P_{ij}\Big(\frac{s_i}{(\sum\limits_{k\in C_i} P_{ik})} - (s_C)_j\Big)\,. \tag{11}$$

The unknown-based GM interpolation in two dimensions can then be written as

$$\widetilde{P} = \begin{bmatrix} P_x & 0 & Q_x \\ 0 & P_y & Q_y \end{bmatrix}\,,$$

where Q_x and Q_y can be computed independently and have the same sparsity as P_x and P_y. Note that this leads to a coarse grid space with a larger number of degrees of freedom than the coarse grid space generated by the unknown-based or the hybrid approach. The increase in degrees of freedom is even further exacerbated in three dimensions, where one needs to add three rigid body modes. So, while we expect improved convergence, the new method is potentially significantly more expensive, and the increased complexities could prevent better performance. Therefore, to mitigate the increase in complexities, we also truncate the Q matrices (see also our numerical results in Sect. 6). Truncation of Q needs to be done independently from truncation of P, because P-truncation is normalized to interpolate constants whereas the truncated Q matrices need to interpolate the rotations. When truncating Q to $\tilde{Q}$, we adjust the weights of $\tilde{Q}$ so that the row sums of $\tilde{Q}$ equal those of Q.

Interestingly enough, the application of both variants beyond the first level leads to very different algorithms. GM1 needs to only interpolate constants after the first level, whereas GM2 needs to continue to interpolate coarser versions of the rigid body modes, thus requiring the storage of coarse grid versions of the rigid body modes as well as additional computations. More details are available in [2]. However, note that GM2 leads to coefficients of similar size, which is not the case for GM1. It is therefore much more difficult to effectively truncate the Q matrices generated in GM1. This difficulty will become evident in Sect. 6.

5 The Local Neighborhood Approach

We now consider an approach where the rigid body modes are incorporated locally. Because exact local interpolation leads to exact global interpolation, this approach should work at least as well as the global matrix approach. This approach requires looking at interpolation from a different angle. Assume that the error at the fine points, e_F, is interpolated by the error at the coarse points, e_C, such that

$$e_F = W_{FC} e_C \ . \tag{12}$$

Let $\tilde{C}$ be the set of new coarse points that have been introduced by adding new degrees of freedom to the coarse nodes. Further, s is a rigid body mode, s_C is s at the original coarse grid points, and s_F is s at the fine grid points. The idea for the local neighborhood approach is then to exactly interpolate the rigid body mode using an extension operator

$$e_F = W_{FC} e_C + W_{F\tilde{C}} e_{\tilde{C}} \quad s.t. \quad s_F = W_{FC} s_C + W_{F\tilde{C}} s_{\tilde{C}} \ , \tag{13}$$

where $s_{\tilde{C}} = 1$ at the new degrees of freedom in $\tilde{C}$. The LN interpolation matrix needs to be defined by harmonic extension based on the local extension $\tilde{W}_{FC} = [W_{FC}, W_{F\tilde{C}}]$. Let D_s be the matrix with diagonal s. Because W_{FC} interpolates constants, the following definition, which is similar to GM2, satisfies Eq. (13):

$$W_{F\tilde{C}} = [D_s^F W_{FC} - W_{FC} D_s^C] \ . \tag{14}$$

To allow for an arbitrary interpolation matrix P, the implementation of this approach performs a preprocessing step (cf. "iterative weight refinement" [9]) that results in $\bar{P}$ where

$$\bar{P}_{ij} = -\frac{1}{a_{ii}} \Big(a_{ij} + \sum_{k \in F_i} a_{ik} w_{kj} \Big) \ , \tag{15}$$

where F_i is the fine neighborhood of point i and

$$w_{kj} = \frac{P_{kj}}{\sum_{n \in C_i} P_{kn}} \ . \tag{16}$$

Now that $\bar{P}$ is based on harmonic extension, Q can be determined using the following formula

$$Q_{ij} = -\frac{1}{a_{ii}} \sum_{k \in F_i} a_{ik} w_{kj} (s_k - s_j) \ . \tag{17}$$

For k rigid body modes $s_1, \ldots, s_k$, the new LN interpolation operator is given by

$$\tilde{P} = [\ \bar{P}\ Q^1\ \ \ldots\ \ Q^k\ ]\ . \tag{18}$$

Note that this approach assumes that $As = 0$. However, the unknown-based interpolation is not generated from A, but from the block diagonal matrix A_D with block diagonals A_{xx} and A_{yy} in 2D (as well as A_{zz} in 3D). In this situation it is important to modify Eq. (17) by incorporating the non-zero residual. We refer to [2] for further details. In addition, like GM2, the LN approach requires the generation of Q on all coarse levels.

6 Numerical Results

In this section, we present numerical results that compare the performance of the previously described AMG approaches. AMG is here used as a preconditioner to either GMRES or CG. The parallel experiments were conducted on the Vulcan supercomputer (LLNL), except for those presented in Table 6, which were computed on the JUQUEEN supercomputer (JSC) [24]. JUQUEEN and Vulcan were ranked 11th and 12th respectively on the TOP500 list of the world's fastest supercomputers of November 2015. JUQUEEN is a 28,672 node 6 Petaflop Blue Gene/Q system at Jülich Supercomputing Center (JSC, Germany), with a total number of 458,752 processor cores. Vulcan is a 24,576 node 5 Petaflop Blue Gene/Q production system at Lawrence Livermore National Laboratory (USA) with a total number of 393,216 processor cores. Both Blue Gene/Q systems use a Power BQC 16C 1.6 GHz processor with 16 cores and 16 GB memory per node.

We use BoomerAMG [16], the unstructured algebraic multigrid solver in hypre version 2.10.0b [17], which now provides an efficient parallel implementation of the GM and the LN approaches. In hypre version 2.10.0b, the user now simply has to provide the smooth error vectors on the fine grid in addition to the linear system. In our case, we provide the rotations s_j, one in 2D, three in 3D. In order to make efficient use of the hardware threads for the 3D results in Tables 3 and 6, we use oversubscription with 2 MPI ranks for each core of the Power BQC processor. Note that no parallel results are given in [2], as a parallel implementation was not available at that time. To ensure a fair comparison of the different methods, we chose an AMG setup such that all components have shown the potential to scale up to large scales. In particular, for all methods, we use HMIS coarsening, introduced in [11], the *extended+i* interpolation method described in [12, 29] and symmetric SOR/Jacobi smoothing in a V(1,1)-cycle.

We consider the compressible linear elasticity problem

$$-2\mu\ \mathrm{div}(\epsilon(u)) - \lambda \mathrm{grad}(\mathrm{div}(u)) = f\ ,$$

where u is the unknown displacement and $\epsilon(u)$ is the strain. The parameters are $\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \mu = \frac{E}{2(1+\nu)}$ (cf. [5]), where the Young modulus is $E = 210$, and we vary the Poisson ratio ν between 0.3 and 0.49.

More detailed descriptions of the various model problems in two and three dimensions are given in subsequent subsections. The finite element assembly is performed in PETSc, and we also use PETSc's GMRES/CG implementation. In all tables we use the abbreviations *U-AMG* for the unknown approach, *H-AMG* for the hybrid approach with the nodal coarsening strategy in Eq. (6) and the row-sum norm, and *H-AMG-GM1/GM2/LN* for the interpolation approaches GM1, GM2, and LN, respectively. Cop denotes the *operator complexity*, which is defined as the sum of the non-zeros of all matrices A_i on all levels divided by the number of non-zeros of the original matrix A. Operator complexity is an indication of memory usage and the number of flops per iteration and also affects setup times. In order to reduce Cop, we truncate P to at most Pmax non-zero elements per row and use a truncation factor of Q-th (absolute threshold) to truncate Q. In the tables, we mark the fastest time (for the sum of setup and solve) as well as the lowest number of iterations in **bold face**. As a baseline for our weak scalability tests, in order to avoid cache effects, we use the smallest problem which still makes use of at least a single full node.

6.1 Results in Two Dimensions

If a Dirichlet boundary condition is applied to a large portion of the boundary, standard nodal or unknown approaches are known to perform well, and we do not expect any additional benefit from the GM or LN approach. Therefore, we consider an elasticity problem on a rectangular domain $[0, 8] \times [0, 1]$ in 2D, fixed on one of the short sides. A volume force orthogonal to the longer sides is applied. We refer to this problem as *2D beam*, and a solution for a linear elastic material is presented in Fig. 2. We use piecewise quadratic finite elements on triangles in all experiments in two dimensions, and, by reordering the unknowns, we ensure that each MPI rank holds a portion of the beam of favorable shape, i.e., close to a square. We present weak scalability results for the *2D beam* in Tables 1 and 2, comparing the unknown

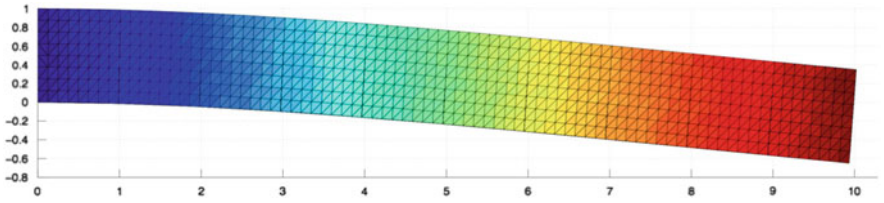

Fig. 2 Solution of the *2D beam* considering linear elasticity with $E = 210$ and $\nu = 0.3$. The color represents the norm of the displacement

Table 1 Weak scalability of the *2D beam* problem with $E = 210$ and $\nu = 0.3$; iterative solver: preconditioned GMRES; stopping tolerance for the relative residual: 1e-8; quadratic triangular finite elements; *Preconditioner* denotes the AMG approach (one V-cycle); *Pmax/Q-th* denotes the truncation of the interpolation operators for P (max non-zeros per row) and Q (absolute threshold); *It.* denotes the number of GMRES iterations and *(Cop)* the operator complexity; *Time GMRES* denotes the runtime of the AMG-GMRES solve phase; *Time BoomerSetup* denotes the time spent in the BoomerAMG setup; *Setup + Solve* denotes the total solution time spent in the AMG setup (BoomerSetup) and the AMG-GMRES (Time GMRES) solve. The fastest variant is marked in *bold face*

#MPI ranks (=#Cores)	Problem size	Preconditioner	Pmax/Q-th	It. (Cop)	Time GMRES (s)	Time BoomerSetup (s)	Time Setup + Solve (s)
32	643,602	U-AMG	– / –	23 (2.4)	4.5	1.1	5.6
		H-AMG	– / –	21 (2.5)	4.0	1.7	5.7
		H-AMG-GM2	– / 0.01	**15** (2.5)	2.9	2.2	**5.1**
128	2,567,202	U-AMG	– / –	26 (2.3)	5.3	1.1	6.4
		H-AMG	– / –	23 (2.3)	4.4	1.7	6.1
		H-AMG-GM2	– / 0.01	**15** (2.4)	3.0	2.3	**5.3**
512	10,254,402	U-AMG	– / –	29 (2.2)	6.0	1.3	7.3
		H-AMG	– / –	25 (2.2)	4.8	1.9	6.7
		H-AMG-GM2	– / 0.01	**16** (2.3)	3.2	2.3	**5.5**
2048	40,988,802	U-AMG	– / –	48 (2.2)	10.2	1.4	11.6
		H-AMG	– / –	26 (2.2)	5.1	1.9	7.0
		H-AMG-GM2	– / 0.01	**18** (2.2)	3.6	2.4	**6.0**
8192	163,897,602	U-AMG	– / –	51 (2.2)	11.0	1.6	12.6
		H-AMG	– / –	26 (2.2)	5.1	2.0	7.1
		H-AMG-GM2	– / 0.01	**18** (2.2)	3.6	2.5	**6.1**
32,768	655,475,202	U-AMG	– / –	54 (2.2)	11.9	1.8	13.7
		H-AMG	– / –	30 (2.2)	5.9	2.0	7.9
		H-AMG-GM2	– / 0.01	**19** (2.2)	3.8	2.5	**6.3**
131,072	2,621,670,402	U-AMG	– / –	59 (2.2)	13.4	2.0	15.4
		H-AMG	– / –	29 (2.2)	5.8	2.1	7.9
		H-AMG-GM2	– / 0.01	**20** (2.2)	4.1	2.7	**6.8**

Table 2 Same problem setup and notation as in Table 1, but larger problem sizes

#MPI ranks (=#Cores)	Problem size	Preconditioner	Pmax/Q-th	It. (Cop)	Time GMRES (s)	Time BoomerSetup (s)	Time Setup + Solve (s)
32	1,644,162	U-AMG	– / –	24 (2.4)	17.5	3.1	20.6
		H-AMG	– / –	23 (2.5)	18.5	4.3	22.8
		H-AMG-GM2	– / 0.01	**14** (2.5)	12.5	5.4	**17.9**
128	6,565,122	U-AMG	– / –	28 (2.3)	20.4	3.1	23.5
		H-AMG	– / –	24 (2.3)	19.9	4.4	24.3
		H-AMG-GM2	– / 0.01	**16** (2.3)	14.0	5.5	**19.5**
512	26,237,442	U-AMG	– / –	44 (2.2)	32.8	3.1	35.9
		H-AMG	– / –	26 (2.2)	21.8	4.5	26.3
		H-AMG-GM2	– / 0.01	**17** (2.3)	15.2	5.6	**20.8**
2048	104,903,682	U-AMG	– / –	51 (2.2)	38.0	3.3	41.3
		H-AMG	– / –	26 (2.2)	21.9	4.6	26.5
		H-AMG-GM2	– / 0.01	**18** (2.3)	16.2	5.7	**21.9**
8192	419,522,562	U-AMG	– / –	54 (2.2)	40.8	3.5	44.3
		H-AMG	– / –	27 (2.2)	23.1	4.6	27.7
		H-AMG-GM2	– / 0.01	**18** (2.2)	16.4	5.8	**22.2**
32,768	1,677,905,922	U-AMG	– / –	58 (2.2)	43.9	3.7	47.6
		H-AMG	– / –	30 (2.2)	25.4	4.8	30.2
		H-AMG-GM2	– / 0.01	**19** (2.2)	17.2	6.0	**23.3**
131,072	6,711,255,042	U-AMG	– / –	83 (2.2)	63.6	3.9	67.5
		H-AMG	– / –	52 (2.2)	44.7	4.9	49.6
		H-AMG-GM2	– / 0.01	**21** (2.2)	19.1	6.2	**25.3**

approach *U-AMG*, the hybrid approach *H-AMG*, and, representing the interpolation approaches, the GM2 approach. The GM1 and LN approaches performed similarly to or worse than GM2 here, but are included in a more detailed discussion on the results in three dimensions, where differences between the approaches are more interesting.

In the weak scaling results in Table 1, the number of GMRES iterations for the unknown approach increases from 23 to 59 iterations, resulting in a noticeable increase in the iteration time as well. However, both the hybrid and GM2 approaches achieve good weak scalability. Comparing the hybrid and the GM2 approaches, the AMG setup times are slightly higher with the GM2 approach. This increased computational cost is expected due to the exact interpolation of the rotation. Since iteration counts and thus the iteration times are lower, the GM2 approach is always the fastest approach in this comparison.

Table 2 also contains weak scaling results for the 2D beam, but the problem sizes are approximately 2.6 times larger per core. Results are similar to the results in Table 1, but here, for the largest problem with 6.7 billion degrees of freedom, the hybrid approach needs 52 compared to only 21 GMRES iterations for the GM2 approach. This improvement leads to a much faster convergence for GM2; see also Fig. 3 for a visualization.

We can conclude that with our settings, all three approaches work well for smaller problems. For larger problems (and larger numbers of cores), the GM2 approach remains scalable whereas U-AMG and H-AMG experience an increase in the number of iterations. The setup cost for the GM2 approach is slightly higher,

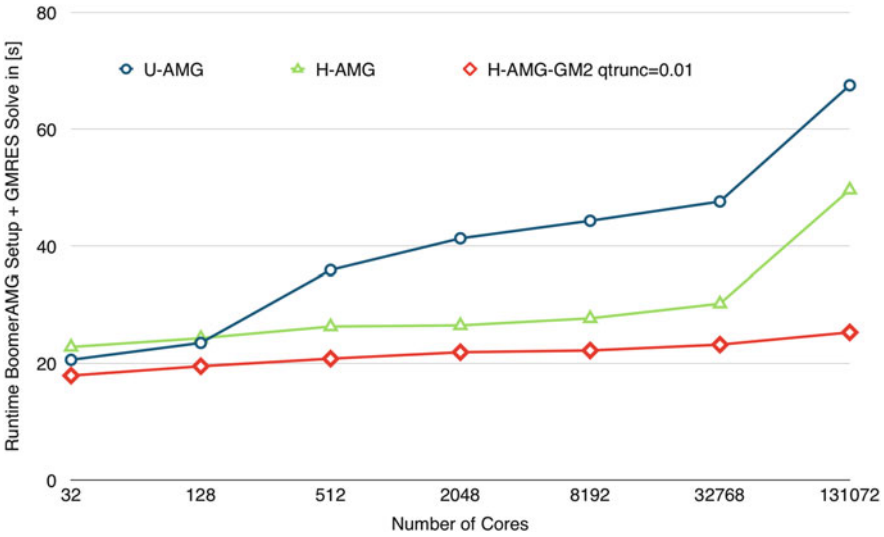

Fig. 3 Weak scalability of total solution time for the two-dimensional beam with $\nu = 0.3$ and $E = 210$; cf. Table 2

compared to the other two approaches, but the setup time is scalable and amortized in the iteration phase; see also Fig. 3.

6.2 Results in Three Dimensions

Now we present results for several three-dimensional domains. In particular, we first investigate weak scalability for a *3D beam* problem. We also investigate the effect of a higher Poisson ratio ν on the *3D beam*, showing scalability results and presenting a small study that increases ν. Second, we examine doubling the beam length. And for a third model problem, we consider a heterogeneous material with different boundary condition, called the *3D cuboid*.

6.2.1 3D Beam Problem

Similar to the *2D beam*, the *3D beam* problem is defined on the domain $[0, 8] \times [0, 1] \times [0, 1]$ for $\nu = 0.3$, $\nu = 0.45$ and $\nu = 0.49$. First, we present weak scalability results in Table 3 for the *3D beam* with $\nu = 0.3$ for all approaches. For the 262K MPI ranks case, we also include a larger problem to show the effect of increasing problem size on performance at large scale.

From the results in Table 3 (see also Figs. 4 and 5), we conclude that for smaller problems, a set of parameters can be found for all approaches such that the results are satisfactory with respect to the numbers of iterations and the solution times. However, for the larger problems (e.g., 262K MPI ranks), the AMG approaches adapted specifically for elasticity, i.e., GM1, GM2, and LN, result in smaller numbers of CG iterations. Note that in the case of the GM1 approach, the low numbers of iterations come at the expense of high complexities because GM1 suffers from the lack of a suitable truncation strategy. As a result, the H-AMG approach is actually faster than GM1. The GM2 and LN approaches achieve the fastest overall total times (with a slight advantage for the LN approach) due to their low iteration counts and acceptable complexities. These considerations also hold when viewing the results for 262K MPI ranks and the increased problem size of 6.3 billion unknowns in Table 3.

Now we increase the Poisson ratio to $\nu = 0.45$ for the *3D beam*. The results in Table 4 (see also Figs. 6 and 7) show that all approaches suffer from a higher number of iterations compared to the case of $\nu = 0.3$. The GM2 and LN approaches remain superior as a result of combining low numbers of iterations with acceptable complexities. For U-AMG and H-AMG, depending on the choice of parameters, either the numbers of iterations are high or the complexities increase substantially. The times are visualized in Figs. 6 and 7. Since GM1 with Pmax=3 requires too much memory, we use it here with Pmax=2. Note that GM1 fails for the largest problem considered.

Table 3 Weak scalability of the *3D beam* problem with $E = 210$ and $\nu = 0.3$; iterative solver: preconditioned CG; stopping tolerance for the relative residual: 1e-6; linear tetrahedral finite elements; 2 MPI ranks per Blue Gene/Q core are used; *Preconditioner* denotes the AMG approach (one V-cycle); *Pmax/Q-th* denotes the truncation of the interpolation operators for P (max non-zeros per row) and Q (absolute threshold); *It.* denotes the number of CG iterations and *(Cop)* the operator complexity; *Time CG* denotes the runtime of the AMG-CG solve phase; *Time BoomerSetup* denotes the time spent in the BoomerAMG setup; *Setup + Solve* denotes the total solution time spent in the AMG setup and the AMG-CG solve

#MPI ranks (= 2×#Cores)	Problem size	Preconditioner	Pmax/Q-th	It. (Cop)	Time CG (s)	Time BoomerSetup (s)	Time Setup + Solve (s)
64	839,619	U-AMG	2 / –	88 (2.76)	23.50	2.42	25.92
		U-AMG	3 / –	58 (2.94)	15.30	3.19	28.49
		U-AMG	4 / –	44 (3.14)	12.25	4.64	16.89
		H-AMG	3 / –	58 (2.42)	12.11	3.44	15.55
		H-AMG	4 / –	50 (2.83)	11.64	5.31	16.95
		H-AMG-GM1	2 / 0.05	52 (2.82)	12.19	5.25	17.44
		H-AMG-GM1	3 / 0.05	**37** (3.61)	10.34	9.18	19.52
		H-AMG-GM2	3 / 0.05	47 (2.45)	10.06	4.54	**14.60**
		H-AMG-LN	3 / 0.05	48 (2.44)	10.26	4.75	15.01
512	6 502,275	U-AMG	2 / –	118 (2.81)	36.27	3.77	40.04
		U-AMG	3 / –	73 (3.02)	23.85	5.21	29.06
		U-AMG	4 / –	54 (3.23)	17.48	6.52	24.00
		H-AMG	3 / –	70 (2.45)	15.22	4.35	19.57
		H-AMG	4 / –	59 (2.87)	14.34	6.39	20.73
		H-AMG-GM1	2 / 0.05	71 (2.84)	18.24	7.17	25.41
		H-AMG-GM1	3 / 0.05	**44** (3.66)	12.81	12.37	25.18
		H-AMG-GM2	3 / 0.05	55 (2.47)	12.35	5.09	**17.44**
		H-AMG-LN	3 / 0.05	57 (2.46)	12.77	5.29	18.06

(continued)

Table 3 (continued)

#MPI ranks (= 2×#Cores)	Problem size	Preconditioner	Pmax/Q-th	It. (Cop)	Time CG (s)	Time BoomerSetup (s)	Time Setup + Solve (s)
4096	51,171,075	U-AMG	2 / –	149 (2.86)	50.64	5.12	55.76
		U-AMG	3 / –	89 (3.09)	33.16	7.14	40.30
		U-AMG	4 / –	67 (3.32)	25.21	8.76	33.97
		H-AMG	3 / –	86 (2.47)	19.34	5.08	24.42
		H-AMG	4 / –	67 (2.89)	16.98	7.39	24.37
		H-AMG-GM1	2 / 0.05	78 (2.84)	20.48	8.25	28.73
		H-AMG-GM1	3 / 0.05	**47** (3.69)	14.05	14.50	28.55
		H-AMG-GM2	3 / 0.05	68 (2.48)	15.83	6.18	22.01
		H-AMG-LN	3 / 0.05	67 (2.48)	15.48	6.38	**21.86**
32,786	406,003,203	U-AMG	2 / –	189 (2.89)	70.94	8.73	79.67
		U-AMG	3 / –	112 (3.13)	49.73	12.90	62.63
		U-AMG	4 / –	86 (3.36)	40.69	15.36	56.05
		H-AMG	3 / –	95 (2.47)	21.97	6.72	28.69
		H-AMG	4 / –	87 (2.90)	22.95	8.98	31.93
		H-AMG-GM1	2 / 0.05	100 (2.84)	26.82	8.89	35.71
		H-AMG-GM1	3 / 0.05	**64** (3.70)	19.54	15.86	35.40
		H-AMG-GM2	3 / 0.05	81 (2.48)	19.53	7.36	26.89
		H-AMG-LN	3 / 0.05	74 (2.48)	17.78	7.60	**25.38**

262,144	3,234,610,179	U-AMG	2 / –	232 (2.90)	95.95	15.36	111.31
		U-AMG	3 / –	135 (3.15)	73.26	27.78	101.04
		U-AMG	4 / –	101 (3.49)	67.16	38.35	105.51
		H-AMG	3 / –	124 (2.48)	29.64	8.53	38.17
		H-AMG	4 / –	106 (2.90)	29.33	9.68	39.01
		H-AMG-GM1	2 / 0.05	138 (2.84)	37.81	8.70	46.51
		H-AMG-GM1	3 / 0.05	**73** (3.70)	22.99	21.39	44.38
		H-AMG-GM2	3 / 0.05	94 (2.48)	23.84	11.01	34.85
		H-AMG-LN	3 / 0.05	84 (2.48)	21.22	11.21	**32.43**
Increased problem size							
262,144	6,312,364,803	U-AMG	3 / –	143 (3.10)	118.72 s	36.94 s	155.66 s
		H-AMG	3 / –	134 (2.52)	62.48 s	13.76 s	76.24 s
		H-AMG-GM2	3 / 0.05	102 (2.53)	48.64 s	16.89 s	65.53 s
		H-AMG-LN	3 / 0.05	**88** (2.53)	41.76 s	16.89 s	**58.65 s**

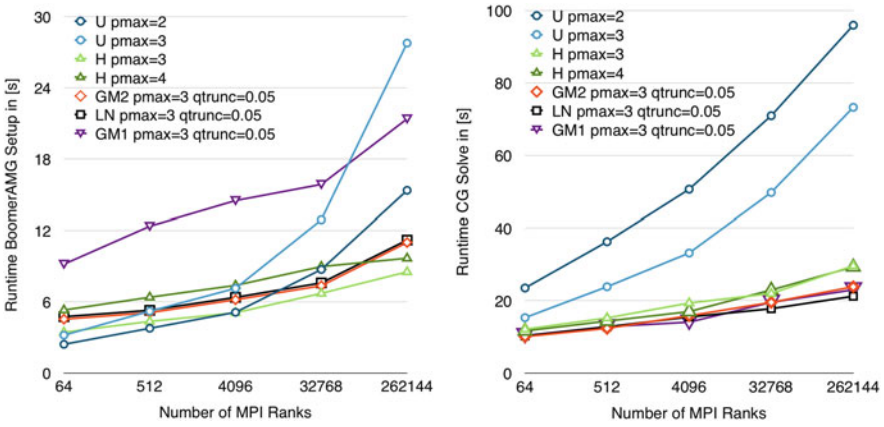

Fig. 4 Weak scalability of the BoomerAMG Setup (*left*) and the time spent in the AMG-CG solve phase (*right*) for the three-dimensional beam with $\nu = 0.3$ and $E = 210$; cf. Table 3

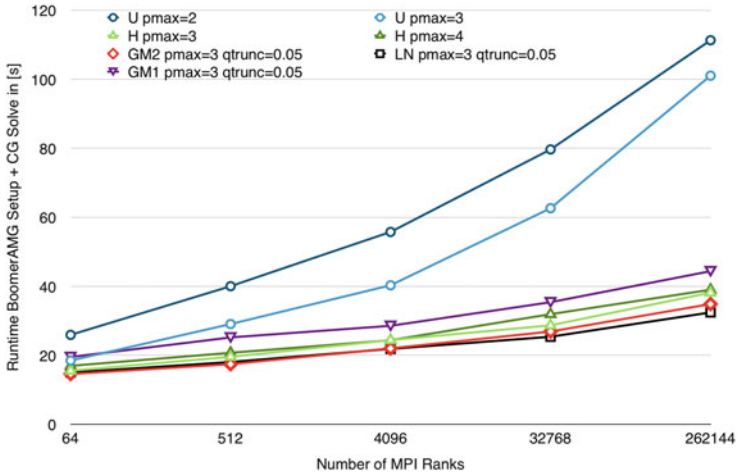

Fig. 5 Weak scalability of total solution time for the three-dimensional beam with $\nu = 0.3$ and $E = 210$; cf. Table 3

Next, in Table 5, the effect of the Poisson ratio on the different AMG approaches is studied. We see that H-AMG does not converge within the limit of 1000 iterations for $\nu = 0.49$. For the other approaches, the convergence rate suffers from an increasing value of ν towards almost incompressibility. This deterioration is also the case for the AMG approaches which are especially adapted for (compressible) elasticity problems, i.e., GM1, GM2, and LN, but which are based on H-AMG. For $\nu = 0.49$, U-AMG, while exhibiting the highest Cop, is the fastest variant in terms of total time.

Table 4 Same problem setup and notation as in Table 3, but larger problem sizes, $\nu = 0.45$. On 32 K MPI ranks H-AMG-GM1 hits the maximal iteration number of 1000 (marked with *max It.*)

#MPI ranks (=2×#Cores)	Problem size	Preconditioner	Pmax/Q-th	It. (Cop)	Time CG (s)	Time BoomerSetup (s)	Time Setup + Solve (s)
64	1,618,803	U-AMG	2 / –	98 (3.18)	56.75	6.10	62.85
		U-AMG	3 / –	69 (3.52)	43.12	9.09	52.21
		U-AMG	4 / –	**54** (3.84)	35.86	12.15	48.01
		H-AMG	3 / –	151 (2.55)	65.27	8.52	73.79
		H-AMG	4 / –	80 (2.97)	38.67	11.86	50.53
		H-AMG-GM1	2 / 0.01	77 (3.07)	38.01	11.08	49.09
		H-AMG-GM2	3 / 0.01	64 (3.07)	31.94	13.52	45.46
		H-AMG-GM2	3 / 0.05	72 (2.56)	31.78	9.52	41.30
		H-AMG-LN	3 / 0.05	69 (2.56)	30.45	10.07	**40.52**
512	12,616,803	U-AMG	2 / –	128 (3.26)	78.80	9.11	87.91
		U-AMG	3 / –	85 (3.59)	56.54	15.10	71.64
		U-AMG	4 / –	**65** (3.94)	46.60	20.58	67.18
		H-AMG	3 / –	232 (2.57)	103.70	9.80	113.50
		H-AMG	4 / –	104 (2.99)	51.60	14.16	65.76
		H-AMG-GM1	2 / 0.01	96 (3.10)	48.72	15.38	64.10
		H-AMG-GM2	3 / 0.01	71 (2.66)	33.29	14.62	**47.91**
		H-AMG-GM2	3 / 0.05	96 (2.58)	43.76	11.45	55.21
		H-AMG-LN	3 / 0.05	89 (2.58)	40.52	11.99	52.51

(continued)

Table 4 (continued)

#MPI ranks (=2×#Cores)	Problem size	Preconditioner	Pmax/Q-th	It. (Cop)	Time CG (s)	Time BoomerSetup (s)	Time Setup + Solve (s)
4096	99,614,403	U-AMG	2 / –	141 (3.30)	95.04	13.13	108.17
		U-AMG	3 / –	106 (3.64)	79.87	21.70	101.57
		U-AMG	4 / –	**85** (4.00)	68.90	27.33	96.23
		H-AMG	3 / –	375 (2.58)	174.54	11.57	186.11
		H-AMG	4 / –	184 (3.01)	95.46	16.32	111.78
		H-AMG-GM1	2 / 0.01	115 (3.12)	59.89	17.58	77.47
		H-AMG-GM2	3 / 0.01	90 (2.60)	42.97	15.81	**58.78**
		H-AMG-GM2	3 / 0.05	125 (2.59)	58.93	13.58	72.52
		H-AMG-LN	3 / 0.05	109 (2.58)	51.15	13.80	64.95
32,768	791,664,003	U-AMG	2 / –	202 (3.31)	146.31	23.41	169.72
		U-AMG	3 / –	128 (3.65)	124.37	48.18	172.55
		U-AMG	4 / –	**102** (4.02)	114.02	54.98	169.00
		H-AMG	3 / –	692 (2.58)	340.75	14.44	355.19
		H-AMG	4 / –	320 (3.01)	174.94	19.26	195.22
		H-AMG-GM1	2 / 0.01	max It.	–	–	–
		H-AMG-GM2	3 / 0.01	124 (2.59)	61.91	19.15	81.06
		H-AMG-GM2	3 / 0.05	146 (2.59)	72.67	17.08	89.75
		H-AMG-LN	3 / 0.05	118 (2.58)	57.24	16.19	**73.43**

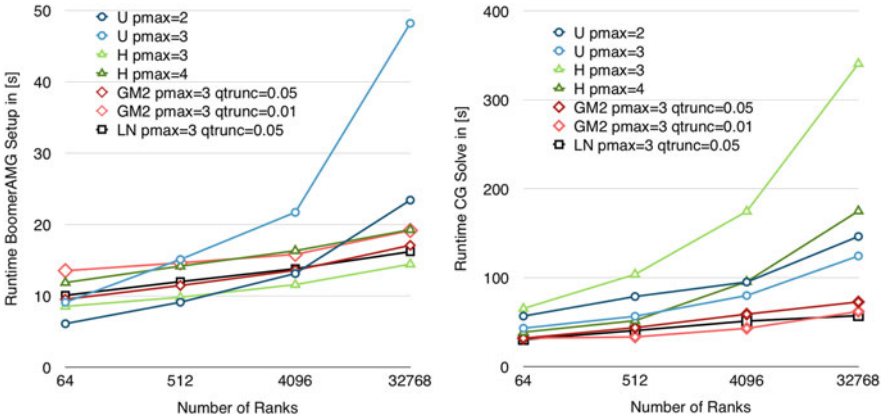

Fig. 6 Weak scalability of the BoomerAMG Setup (*left*) and the time spent in the AMG-CG solve phase (*right*) for the three-dimensional beam with $\nu = 0.45$ and $E = 210$; cf. Table 4

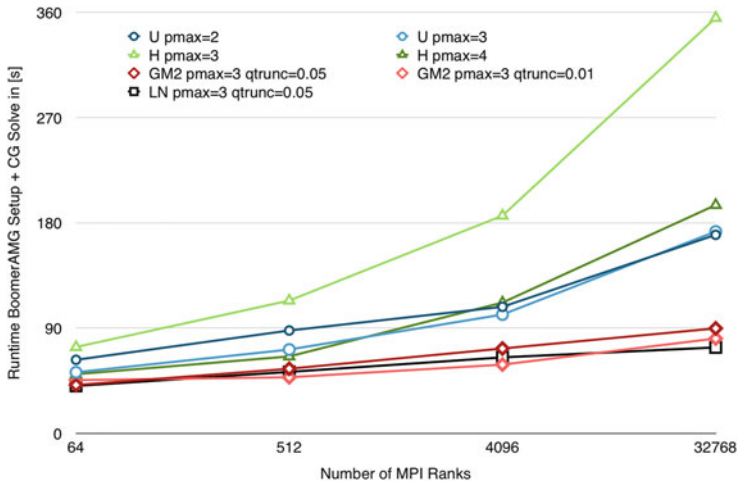

Fig. 7 Weak scalability of total solution time for the three-dimensional beam with $\nu = 0.45$ and $E = 210$; cf. Table 4

6.2.2 3D Beam Problem with Double Length

For $\nu = 0.3$, we examine the effect of doubling the length of the *3D beam* such that the domain is $[0, 16] \times [0, 1] \times [0, 1]$. Table 6 lists the results obtained for the *3D beam* with double the length, using up to 16 of the total 28 racks of the JUQUEEN supercomputer. Again, these experiments show the clear advantage of the GM2 and LN approaches for this problem over the standard methods. The largest three dimensional problem with approximately 13 billion unknowns is solved in less than 81 s using the LN approach. Here, the solve phase time of LN is twice as fast as that of the fastest standard approach H-AMG.

Table 5 Same problem setup and notation as in Table 3. Investigation of the effect of an increasing ν; *Setup + Solve* denotes the total solution time spent in the AMG setup and the AMG-CG solve; H-AMG hits the maximal iteration number of 1000 (marked with *max It.*)

512 MPI ranks, 12,616,803 dofs							
		$\nu = 0.3$		$\nu = 0.45$		$\nu = 0.49$	
Preconditioner	Pmax/Q-th	It. (Cop)	Setup + Solve (s)	It. (Cop)	Setup + Solve (s)	It. (Cop)	Setup + Solve (s)
U-AMG	2 / –	128 (2.79)	76.61	128 (3.26)	87.91	125 (3.60)	102.94
U-AMG	3 / –	79 (2.98)	53.69	85 (3.59)	71.64	89 (3.81)	85.39
U-AMG	4 / –	57 (3.21)	56.16	**65** (3.94)	67.18	**72** (3.89)	**79.10**
H-AMG	3 / –	76 (2.50)	41.47	232 (2.57)	113.50	max It.	–
H-AMG	4 / –	59 (2.94)	41.24	104 (2.99)	65.76	max It.	–
H-AMG-GM1	2 / 0.01	56 (2.88)	39.74	96 (3.10)	64.10	189 (3.49)	127.05
H-AMG-GM2	3 / 0.01	49 (2.59)	34.07	71 (2.66)	**47.91**	159 (2.79)	99.25
H-AMG-LN	3 / 0.01	**47** (2.54)	**32.34**	79 (2.62)	50.06	196 (2.71)	110.85

Table 6 Weak scalability of the larger $[0, 16] \times [0, 1] \times [0, 1]$ *3D beam* problem with $E = 210$ and $\nu = 0.3$. Same notation as in Table 3. Computations carried out on JUQUEEN BlueGene/Q at Jülich Supercomputing Centre (JSC)

#MPI ranks (=2×#Cores)	Problem size	Preconditioner	Pmax/Q-th	It. (Cop)	Time CG (s)	Time BoomerSetup (s)	Time Setup + Solve (s)
16	424,683	U-AMG	3 / –	96 (2.83)	35.18	**3.83**	39.01
		H-AMG	3 / –	92 (2.49)	27.39	4.73	32.12
		H-AMG-GM2	3 / 0.01	**55** (3.50)	21.63	11.38	33.01
		H-AMG-LN	3 / 0.01	56 (3.14)	**20.19**	9.22	**29.41**
128	3,232,563	U-AMG	3 / –	118 (2.93)	64.49	**6.05**	70.54
		H-AMG	3 / –	117 (2.48)	49.88	6.71	56.59
		H-AMG-GM2	3 / 0.01	84 (3.04)	41.66	12.04	43.70
		H-AMG-LN	3 / 0.01	**65** (2.68)	**29.68**	10.23	**39.91**
1024	25,213,923	U-AMG	3 / –	145 (2.99)	82.55	8.79	91.34
		H-AMG	3 / –	138 (2.50)	59.94	**8.57**	68.51
		H-AMG-GM2	3 / 0.01	86 (2.59)	38.88	12.12	51.00
		H-AMG-LN	3 / 0.01	**80** (2.54)	**35.77**	11.41	**47.18**
8192	199,151,043	U-AMG	3 / –	188 (3.04)	113.42	10.61	124.03
		H-AMG	3 / –	160 (2.51)	70.76	**9.01**	79.77
		H-AMG-GM2	3 / 0.01	108 (2.53)	48.94	11.82	60.76
		H-AMG-LN	3 / 0.01	**103** (2.52)	**46.64**	11.94	**58.58**
65,536	1,583,018,883	U-AMG	3 / –	225 (3.11)	157.34	19.84	177.18
		H-AMG	3 / –	195 (2.52)	89.01	**11.32**	100.33
		H-AMG-GM2	3 / 0.01	120 (2.53)	55.63	14.80	70.43
		H-AMG-LN	3 / 0.01	**115** (2.53)	**53.29**	14.86	**68.15**
524,288	12,623,496,963	U-AMG	3 / –	270 (3.06)	229.40	39.18	268.58
		H-AMG	3 / –	253 (2.52)	118.38	**13.34**	131.72
		H-AMG-GM2	3 / 0.01	144 (2.53)	68.56	18.17	86.73
		H-AMG-LN	3 / 0.01	**131** (2.53)	**62.52**	18.11	**80.63**

6.2.3 3D Cuboid Problem

Finally, we consider a *3D cuboid* problem. The cuboid has the same form and size as the original *3D beam*, but is fixed on the two opposite sides with $x = 0$ and $x = 8$. We then compress the cuboid to 95 % of its length. Note that for the *3D cuboid*, we have a core material with $E = 210$ and $\nu = 0.45$ in the part of the cuboid where $0.25 < y < 0.75$ and $0.25 < z < 0.75$. Here (x, y, z) denotes the coordinates in the undeformed reference configuration of the cuboid. In the remaining hull, we have $E = 210$ and $\nu = 0.3$.

The results for the *3D cuboid* problem in Table 7 show that the AMG approaches benefit from the larger Dirichlet boundary as compared to the *3D beam*. However, the GM2 and LN approaches show the best numerical scalability, i.e., the numbers of iterations only increase from 29 to 44 for GM2 and from 24 to 39 for LN when scaling weakly from 64 to 262 K MPI ranks. For this problem, the H-AMG approach remains competitive as well for the largest number of ranks with regard to total times as a result of its low setup time.

6.3 Parallel Problem Assembly and Reordering Process

Although the focus of this paper is on the parallel performance of AMG for linear elasticity problems, we also comment on the parallel problem assembly and setup, presenting timing results in Table 8. In order to assemble the global elasticity problems in two and three dimensions, we first decompose the domain into nonoverlapping parts of equal size, one for each MPI rank. We then assemble local stiffness matrices corresponding to these local parts. These computations are completely local to the ranks and thus perfectly scalable. The local assembly process is denoted as **Local Asm.** in Table 8. To assemble the local stiffness matrices to one global and parallel stiffness matrix, some global communication is necessary. This global assembly process is denoted as **Global Asm.** in Table 8. This process scales fine up to 32 K ranks. Scaling further, the amount of communication and synchronization slows the global assembly down. A classical lexicographical ordering of the global indices is often not optimal for the convergence, especially using hybrid approaches, and we therefore reorder the indices. After the **reordering** process, each rank holds a portion of the global stiffness matrix which has a shape close to a square in two dimensions and a cube in three dimensions. The implementation of the index reordering step is very fast (see Table 8) but makes use of the same communication patterns as the global assembly process leading to the same deterioration on more than 32 K cores.

Table 7 Weak scalability results for the *3D cuboid* problem; notation as in Table 3

#MPI ranks					Time	Time	Time
(=2×#Cores)	Problem size	Preconditioner	Pmax/Q-th	It. (Cop)	CG (s)	BoomerSetup (s)	Setup + Solve (s)
64	1,618,803	U-AMG	2 / –	49 (2.73)	25.50	4.53	30.03
		U-AMG	3 / –	34 (2.95)	18.76	6.93	25.69
		H-AMG	3 / –	34 (2.48)	14.57	7.67	22.24
		H-AMG-GM2	3 / 0.01	29 (3.04)	14.51	12.81	27.32
		H-AMG-LN	3 / 0.01	**24** (2.68)	11.09	10.86	**21.95**
512	12,616,803	U-AMG	2 / –	65 (2.79)	35.83	6.24	42.07
		U-AMG	3 / –	43 (2.98)	24.77	8.57	33.34
		H-AMG	3 / –	41 (2.50)	18.06	8.33	26.39
		H-AMG-GM2	3 / 0.01	31 (2.59)	14.25	11.83	26.08
		H-AMG-LN	3 / 0.01	**23** (2.55)	10.57	11.25	**21.83**
4096	99,614,403	U-AMG	2 / –	84 (2.84)	49.76	8.06	57.82
		U-AMG	3 / –	53 (3.04)	32.02	10.75	42.77
		H-AMG	3 / –	45 (2.51)	20.18	9.23	29.41
		H-AMG-GM2	3 / 0.01	33 (2.53)	15.32	12.28	27.60
		H-AMG-LN	3 / 0.01	**31** (2.53)	14.34	12.30	**26.64**
32,768	791,664,003	U-AMG	2 / –	102 (2.87)	64.11	11.83	75.94
		U-AMG	3 / –	63 (3.09)	43.83	18.31	62.14
		H-AMG	3 / –	49 (2.52)	22.58	10.01	32.59
		H-AMG-GM2	3 / 0.01	38 (2.53)	17.90	13.58	31.48
		H-AMG-LN	3 / 0.01	**34** (2.53)	16.03	14.05	**30.08**
262,144	6,312,364,803	U-AMG	2 / –	126 (2.86)	81.26	17.51	98.77
		U-AMG	3 / –	73 (3.10)	60.98	37.99	98.97
		H-AMG	3 / –	64 (2.52)	30.15	12.60	42.75
		H-AMG-GM2	3 / 0.01	44 (2.53)	21.20	17.85	39.05
		H-AMG-LN	3 / 0.01	**39** (2.53)	18.84	17.83	**36.67**

Table 8 Presentation of problem assembly and setup timings, which are independent of the chosen AMG preconditioner. Values are averages over the measured values in all runs presented in Table 3. The total runtime of the complete *3D beam* application can be obtained by adding these three times to the time *Setup* + *Solve* from Table 3

#MPI ranks	Problem size	Local Asm. (s)	Global Asm. (s)	Reorder (s)
64	839,619	19.10	0.81	0.67
512	6,502,275	19.14	0.86	0.84
4096	51,171,075	19.14	0.93	0.77
32,786	406,003,203	19.05	1.44	1.57
262,144	3,234,610,179	19.03	8.82	9.35

7 Conclusions

We investigated the performance of hypre's AMG variants for elasticity for several 2D and 3D linear elasticity problems with varying Poisson ratios ν. We compared the unknown and hybrid approaches, which use prolongation operators that only interpolate the translations, with three approaches, GM1, GM2 and LN, that are based on the hybrid approach and also incorporate the rotations. In all cases, GM1, GM2 and LN showed improved convergence over the hybrid approach when using the same truncation for P. For $\nu = 0.3$, all hybrid approaches scaled better than the unknown approach, and the GM2 and LN approaches were overall faster for very large problems. For the largest problem in three dimensions with 14 billion unknowns and using the largest number of processes considered, i.e., 524,288 processes, the LN approach was 40 % faster than the standard approaches. For $\nu = 0.45$, GM2 and LN clearly scale better than the other approaches and are more than twice as fast on 32,768 processes with better complexities and five times as fast as the hybrid approach with the same operator complexity.

We also found that the unknown approach was more robust with regard to an increase in ν than the other approaches, solving the problem with $\nu = 0.49$ faster than any of the other approaches, but generally needed larger complexities. While the hybrid approach did not converge within 1000 iterations for $\nu = 0.49$, GM1, GM2 and LN were able to solve the problem in less than 200 iterations.

Overall, our study shows that the inclusion of the rigid body modes into AMG interpolation operators is generally beneficial, especially at large scale. We conclude that, for elasticity problems, using enhancements of the interpolation, parallel AMG methods are able to scale to the largest supercomputers currently available.

Acknowledgements This work was supported in part by the German Research Foundation (DFG) through the Priority Program 1648 "Software for Exascale Computing" (**SPPEXA**) under **KL 2094/4-1** and **RH 122/2-1**. The authors also gratefully acknowledge the use of the **Vulcan** supercomputer at Lawrence Livermore National Laboratory. Partial support for this work was provided through Scientific Discovery through Advanced Computing (**SciDAC**) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (and Basic Energy Sciences/Biological and Environmental Research/High Energy Physics/Fusion

Energy Sciences/Nuclear Physics). This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract **DE-AC52-07NA27344**. The authors gratefully acknowledge the Gauss Centre for Supercomputing (GCS) for providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS share of the supercomputer **JUQUEEN** [24] at Jülich Supercomputing Centre (JSC). GCS is the alliance of the three national supercomputing centres HLRS (Universität Stuttgart), JSC (Forschungszentrum Jülich), and LRZ (Bayerische Akademie der Wissenschaften), funded by the German Federal Ministry of Education and Research (BMBF) and the German State Ministries for Research of Baden-Württemberg (MWK), Bayern (StMWFK) and Nordrhein-Westfalen (MIWF).

References

1. Augustin, C.M., Neic, A., Liebmann, M., Prassl, A.J., Niederer, S.A., Haase, G., Plank, G.: Anatomically accurate high resolution modeling of human whole heart electromechanics: a strongly scalable algebraic multigrid solver method for nonlinear deformation. J. Comput. Phys. **305**, 622–646 (2016)
2. Baker, A.H., Kolev, T.V., Yang, U.M.: Improving algebraic multigrid interpolation operators for linear elasticity problems. Numer. Linear Algebra Appl. **17**(2–3), 495–517 (2010). http://dx.doi.org/10.1002/nla.688
3. Blatt, M., Ippisch, O., Bastian, P.: A massively parallel algebraic multigrid preconditioner based on aggregation for elliptic problems with heterogeneous coefficients. arXiv preprint arXiv:1209.0960 (2013)
4. Braess, D.: Towards algebraic multigrid for elliptic problems of second order. Computing **55**(4), 379–393 (1995). http://dx.doi.org/10.1007/BF02238488
5. Braess, D.: Finite Elemente, vol. 4. Springer, Berlin (2007)
6. Brezina, M., Cleary, A.J., Falgout, R.D., Jones, J.E., Manteufel, T.A., McCormick, S.F., Ruge, J.W.: Algebraic multigrid based on element interpolation (AMGe). SIAM J. Sci. Comput. **22**, 1570–1592 (2000). Also LLNL technical report UCRL-JC-131752
7. Brezina, M., Tong, C., Becker, R.: Parallel algebraic multigrid methods for structural mechanics. SIAM J. Sci. Comput. **27**(5), 1534–1554 (2006)
8. Bulgakov, V.E.: Multi-level iterative technique and aggregation concept with semi-analytical preconditioning for solving boundary value problems. Commun. Numer. Methods Eng. **9**(8), 649–657 (1993). http://dx.doi.org/10.1002/cnm.1640090804
9. Cleary, A.J., Falgout, R.D., Henson, V.E., Jones, J.E., Manteuffel, T.A., McCormick, S.F., Miranda, G.N., Ruge, J.W.: Robustness and scalability of algebraic multigrid. SIAM J. Sci. Comput. **21**, 1886–1908 (2000)
10. Clees, T.: AMG Strategies for ODE Systems with Applications in Industrial Semiconductor Simulation. Shaker Verlag GmbH, Germany (2005)
11. De Sterck, H., Yang, U.M., Heys, J.J.: Reducing complexity in parallel algebraic multigrid preconditioners. SIAM J. Matrix Anal. Appl. **27**(4), 1019–1039 (2006). http://dx.doi.org/10.1137/040615729
12. De Sterck, H., Falgout, R.D., Nolting, J.W., Yang, U.M.: Distance-two interpolation for parallel algebraic multigrid. Numer. Linear Algebra Appl. **15**, 115–139 (2008)
13. Dohrmann, C.R.: Interpolation operators for algebraic multigrid by local optimization. SIAM J. Sci. Comput. **29**(5), 2045–2058 (electronic) (2007). http://dx.doi.org/10.1137/06066103X
14. Griebel, M., Oeltz, D., Schweitzer, A.: An algebraic multigrid for linear elasticity. J. Sci. Comput. **25**(2), 385–407 (2003)
15. Henson, V.E., Vassilevski, P.S.: Element-free AMGe: general algorithms for computing interpolation weights in AMG. SIAM J. Sci. Comput. **23**(2), 629–650 (electronic) (2001). http://dx.doi.org/10.1137/S1064827500372997. copper Mountain Conference (2000)

16. Henson, V.E., Yang, U.M.: BoomerAMG: a parallel algebraic multigrid solver and preconditioner. Appl. Numer. Math. **41**, 155–177 (2002)
17. hypre: High performance preconditioners. http://www.llnl.gov/CASC/hypre/
18. Lanser, M.: Nonlinear FETI-DP and BDDC Methods. Ph.D. thesis, Universität zu Köln (2015)
19. Muresan, A.C., Notay, Y.: Analysis of aggregation-based multigrid. SIAM J. Sci. Comput. **30**, 1082–1103 (2008)
20. Notay, Y.: An aggregation-based algebraic multigrid method. Electron. Trans. Numer. Anal. **37**, 123–146 (2010)
21. Notay, Y., Napov, A.: Algebraic analysis of aggregation-based multigrid. Numer. Linear Algebra Appl. **18**, 539–564 (2011)
22. Ruge, J.W.: AMG for problems of elasticity. Appl. Math. Comput. **19**, 293–309 (1986)
23. Ruge, J.W., Stüben, K.: Algebraic multigrid (AMG). In: McCormick, S.F. (ed.) Multigrid Methods. Frontiers in Applied Mathematics, vol. 3, pp. 73–130. SIAM, Philadelphia (1987)
24. Stephan, M., Docter, J.: JUQUEEN: IBM blue gene/Q®supercomputer system at the Jülich Supercomputing Centre. JLSRF **1**, A1 (2015). http://dx.doi.org/10.17815/jlsrf-1-18
25. Stüben, K.: An introduction to algebraic multigrid. In: Multigrid, pp. 413–532. Academic Press, London/San Diego (2001). also available as GMD Report 70, November 1999
26. Trottenberg, U., Oosterlee, C.W., Schüller, A.: Multigrid. Academic Press, London/San Diego (2001)
27. Vaněk, P., Mandel, J., Brezina, M.: Algebraic multigrid by smooth aggregation for second and fourth order elliptic problems. Computing **56**, 179–196 (1996)
28. Yang, U.M.: Parallel algebraic multigrid methods – high performance preconditioners. In: Bruaset, A., Tveito, A. (eds.) Numerical Solutions of Partial Differential Equations on Parallel Computers. Lecture Notes in Computational Science and Engineering, pp. 209–236. Springer, Berlin (2006)
29. Yang, U.M.: On long-range interpolation operators for aggressive coarsening. Numer. Linear Algebra Appl. **17**, 453–472 (2010)