Comput Mech (2012) 50:321–333
DOI 10.1007/s00466-011-0661-y

ORIGINAL PAPER

Comparison of the deflated preconditioned conjugate gradient method and algebraic multigrid for composite materials

T. B. Jönsthövel · M. B. van Gijzen · S. MacLachlan · C. Vuik · A. Scarpas

Received: 31 August 2010 / Accepted: 26 October 2011 / Published online: 16 November 2011

Abstract Many applications in computational science and engineering concern composite materials, which are characterized by large discontinuities in the material properties. Such applications require fine-scale finite-element meshes, which lead to large linear systems that are challenging to solve with current direct and iterative solutions algorithms. In this paper, we consider the simulation of asphalt concrete, which is a mixture of components with large differences in material stiffness. The discontinuities in material stiffness give rise to many small eigenvalues that negatively affect the convergence of iterative solution algorithms such as the preconditioned conjugate gradient (PCG) method. This paper considers the deflated preconditioned conjugate gradient (DPCG) method in which the rigid body modes of sets of elements with homogeneous material properties are used as deflation vectors. As preconditioner we consider several variants of the algebraic multigrid smoothed aggregation method. We evaluate the performance of the DPCG method on a parallel computer using up to 64 processors. Our test problems are derived from real asphalt core samples, obtained using CT scans. We show that the DPCG method is an efficient and robust technique for solving these challenging linear systems.

Keywords Deflation · Algebraic multigrid · Preconditioners · Conjugate gradients · Rigid body modes · CT scan · Structural mechanics

Mathematics Subject Classification (2000) 65F10 · 65F08 · 65Z05

T. B. Jönsthövel (✉) · A. Scarpas
Department of Structural Mechanics, Faculty of Civil Engineering,
Delft University of Technology, 2628 CN Delft, The Netherlands
e-mail: t.b.jonsthovel@tudelft.nl

A. Scarpas
e-mail: a.scarpas@tudelft.nl

M. B. van Gijzen · C. Vuik
Department of Applied Mathematical Analysis, Faculty
of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology, 2628 CN Delft, The Netherlands
e-mail: m.b.vangijzen@tudelft.nl

C. Vuik
e-mail: c.vuik@tudelft.nl

S. MacLachlan
Department of Mathematics, Tufts University, Bromfield-Pearson
Building, 503 Boston Avenue, Medford, MA 02155, USA
e-mail: scott.maclachlan@tufts.edu

1 Introduction

Finite-element (FE) computations are indispensable for the simulation of material behavior. Recent developments in visualization and meshing software give rise to high-quality but very fine meshes, resulting in large systems, with millions of degrees of freedom, that need to be solved. When choosing a solver for these systems, we distinguish between direct solution methods and iterative methods. By now, it is well established that iterative solution methods are preferable, due to their potential for better algorithmic and parallel scalability than is possible with direct methods. In recent years, parallel computing has become the standard in FE software packages; therefore, only parallel algorithms are considered here. In our application, the FE stiffness matrix is symmetric and positive definite and, therefore, the preconditioned conjugate gradient (PCG) method is the iterative method of choice from a theoretical point of view. Furthermore, the PCG method is well suited for parallel computing.

This paper focuses on the choice of a parallel preconditioner for FE problems in structural mechanics.

Many FE computations involve simulations of *inhomogenous* materials, where the difference in properties of materials leads to large differences in the entries of the resulting stiffness matrices. We have shown in [28] that these jumps in coefficients slow down the convergence of the PCG method using a simple preconditioner. By decoupling regions with homogeneous material properties with a deflation technique, a more robust PCG method has been constructed: the deflated preconditioned conjugate gradient (DPCG) method. The DPCG method proposed in [28] is an extension of the technique of subdomain deflation, introduced in [36], and of the preconditioners to matrices with large differences in the entries, proposed in [50]. There is a correlation between the number of rigid body modes of sub-bodies of materials contained within the FE mesh and the number of small eigenvalues of the scaled stiffness matrix. We used rigid body modes combined with existing deflation techniques to remove those small eigenvalues from the spectrum of the scaled stiffness matrix, yielding a stable and robust adaptation of the PCG method. Like the PCG method, the DPCG method is well suited for parallel computing.

For PDEs with heterogeneous coefficients, there are several state of the art (black box) solvers available. Direct solution methods, the FETI method, and algebraic multigrid methods (AMG) are among the most popular solvers and preconditioners. An important advantage of direct solution methods is their robustness: they can, to a large extent, be used as black boxes for solving a wide range of problems, but they are expensive in terms of computational costs. Several high quality, well parallelisable public domain direct solvers exist [32,38,42,10,6]. The FETI and AMG methods are also robust but are often much less expensive than direct solution methods and have been discussed in [23] and [49]. As a comparison to DPCG, we focus on the best AMG adaptation, smoothed aggregation (SA), as it has been demonstrated to be a successful parallel preconditioner for a number of structural mechanics applications [2,4,14]. The two most relevant studies of SA to the simulations considered here are those of [4,7], both of which focus on micro-FE modeling of bone deformation, based on micro-CT scans of human bones.

In this paper, we will compare the performance of SA using default parameters as a preconditioner for both PCG and DPCG with that of SA using an optimal choice of parameters as a preconditioner to PCG. All methods are implemented within a parallel environment using Trilinos [26]. We will provide an overview of the DPCG method proposed in [28], and discuss the parallel implementation of the DPCG method into an existing FE software package. Finally, we present numerical experiments on FE meshes from real-life cores of asphalt concrete as case studies for this comparison.

2 Problem definition: composite materials

In this paper, we consider asphalt concrete as an example of a composite material. Asphalt concrete consists of a mixture of bitumen, aggregates, and air voids. The difference between the stiffness of bitumen and the aggregates is significant, especially at high temperatures. The surge in recent studies on wheel-pavement interaction show the importance of understanding the component interaction within asphalt concrete, demanding high-quality FE meshes.

Until recently, because of the extremely long execution time, memory, and storage space demands, the majority of FE simulations of composite materials were performed by means of homogenization techniques [18]. Unfortunately, these techniques do not provide an understanding of the actual interaction between the components of the material. It is known, however, that component interaction is the most critical factor in determining the overall mechanical response of the composite material. We obtain accurate FE meshes of the asphalt concrete materials by means of computed tomography (CT) X-ray scans and additional, specialized software tools like Simpleware ScanFE [43].

We use the computational framework described in [18] to simulate the response of a composite material that is subjected to external forces by means of small load steps. We define the relation between the undeformed (reference) state of a volume, V, with the deformed (current) state position vector at a fixed point in time as

$$x = X + \bar{u}, \tag{1}$$

where $x = \left[x_x, x_y, x_z\right]^T$ is the undeformed state position vector, $X = \left[X_x, X_y, X_z\right]^T$ is the deformed state position vector, and $\bar{u} = \left[\bar{u}_x, \bar{u}_y, \bar{u}_z\right]^T$ represents the change of displacement. Changes of the undeformed and deformed state over time are given by,

$$dx = \mathbf{F} dX \tag{2}$$

where $\mathbf{F} = I + \frac{\partial \bar{u}}{\partial X}$ is the deformation gradient. We refer to [18] for details on the balancing of forces within the material. For any given externally applied force, $\bar{f}$, the linearized virtual work equation at equilibrium is given by

$$\begin{aligned} &\int_V \mathbf{S} : \mathrm{sym}\left(\nabla_0 \bar{u}^T \cdot \nabla_0 \delta v\right) dV \\ &\quad + \int_V \mathrm{sym}\left(\mathbf{F}^T \cdot \nabla_0 \delta v\right) : \mathbb{C} : \mathrm{sym}\left(\mathbf{F}^T \cdot \nabla_0 \bar{u}\right) dV \\ &\quad = \delta v \cdot \bar{f} - \int_V \mathbf{S} : \mathrm{sym}\left(\mathbf{F}^T \cdot \nabla_0 \delta v\right) dV \end{aligned} \tag{3}$$

where δv are the virtual velocities, $\nabla_0 \bar{u}$ is the directional derivative in the reference configuration of the displacement field, $\mathbb{C}$ is the fourth-order constitutive tensor, and $\mathbf{S}$ is the second-order Second Piola–Kirchhoff stress tensor. In this paper, we consider hyper elasticity, and use the Neo-Hookean constitutive model,

$$\mathbf{S} = \mu\mathbf{I} - \mu\det(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-\alpha}(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}, \tag{4}$$

and

$$\mathbb{C} = \frac{2\mu\alpha}{\det(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{\alpha}}(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1} \otimes (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1} - \frac{2\mu}{\det(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{\alpha}}\frac{\partial(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}}{\partial(\mathbf{F}^{\mathrm{T}}\mathbf{F})}, \tag{5}$$

where $\alpha = \frac{\nu}{1-2\nu}$ and ν, μ are the Poisson ratio and the Lamé material constant, respectively. By using the FE method, using standard linear basis functions on tetrahedral meshes, we obtain the corresponding stiffness matrix; solving linear system (6),

$$Ku = f, \tag{6}$$

is the most time-consuming computation of the FE simulation. In this equation, u represents the change of displacement of the nodes in the FE meshes, and f is the force unbalance in the system, which is determined by the difference of the internal forces within the system and the external forces exerted on the system. The internal forces are computed by solving non-linear equations for each finite element. The computing time and costs for these steps are negligible compared to solving linear system (6). The stiffness matrix, K, is symmetric and positive definite for elastic, constrained systems; hence, $\forall u \neq 0 : u^T K u > 0$ and all eigenvalues of K are positive. Within the context of mechanics, $\frac{1}{2}u^T K u$ is the strain energy stored within the system for displacement vector u [9]. Energy is defined as a non-negative entity; hence, the strain energy must be non-negative also.

3 Solvers

3.1 Preconditioned conjugate gradient method

Because K is SPD, CG [27] is the method of choice to solve (6) iteratively. The CG method is based on minimizing the energy norm of the error in the k-th approximation to the solution over the Krylov subspace,

$$\mathcal{K}^{k-1}(K; r_0) = \mathrm{span}\{r_0, Kr_0, \ldots, K^{k-1}r_0\}, \tag{7}$$

where the energy norm is defined as $\|u\|_K = \left(u^T K u\right)^{\frac{1}{2}}$. We note that minimizing the error in the K-norm is, in fact, minimizing the strain energy over the Krylov subspace $\mathcal{K}^{k-1}(K; r_0)$. This implies that, for a given distributed static load, we construct a displacement vector that has an optimal distribution of the force over the material.

Theorem 10.2.6 in [22] provides a bound on the error of the approximations computed by CG. Denote the ith eigenvalue of K in nondecreasing order by $\lambda_i(K)$ or, simply, λ_i. After k iterations of the CG method, the error is bounded by

$$\|u - u_k\|_K \leq 2\|u - u_0\|_K \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k, \tag{8}$$

where $\kappa = \kappa(K) = \lambda_n/\lambda_1$ is the spectral condition number of K. While this bound is not always sharp, the error reduction capability of CG is generally limited when the condition number is large. The condition number of K typically increases when the number of elements increases or when the relative stiffnesses of the materials change. For plastic and viscous behavior, this can result in a series of increasing numbers of iterations as the stiffness changes with every load or time step. Here, we focus on a single load and time step, although this is an important question for future research as plasticity and viscosity are key to realistic simulations.

The convergence of CG is dependent not only on the condition number but also on the number and distribution of very small eigenvalues [48]. The eigenvectors corresponding to the smallest eigenvalues have a significant contribution to the global solution but may need a significant number of iterations for convergence locally. Hence, very small eigenvalues can dramatically increase the number of iterations needed to generate a good approximate solution. In our application, the number of aggregates has a direct correlation with the number of smallest eigenvalues of K. Increasing the number of aggregates may, therefore, result in more very small eigenvalues and deterioration of the convergence rates.

To improve the performance of CG, we change the linear system under consideration, resulting in a problem with more favorable extreme eigenvalues and/or clustering. The most efficient way to do this is by preconditioning of the linear system. Preconditioners are essential for the performance of iterative solvers, and no Krylov iterative solver can perform well for these problems without one [41]. The preconditioned equation reads

$$M^{-1}Ku = M^{-1}f, \tag{9}$$

where matrix M is the left preconditioner, which is assumed to also be symmetric and positive definite. The CG iteration bound of Eq. 8 also applies to the preconditioned matrix, replacing $\kappa(K)$ with $\kappa(M^{-1}K)$. Thus, the preconditioning matrix must satisfy the requirements that it is cheap to construct, and that it is inexpensive to solve the linear system $Mv = w$, as preconditioned algorithms need to solve such a linear system in each iteration step. A rule of thumb is that M must resemble the original matrix, K, to obtain eigenvalues that cluster around 1. Obviously, $M = K$ would be the best choice to minimize $\kappa(M^{-1}K)$, but this choice is expensive and equivalent to solving the original system. Common choices of M are the diagonal of K, which is known as diagonal scaling, and the Incomplete Cholesky factorization using a drop tolerance to control the fill-in [41].

Within the field of engineering, the PCG method is widely used because it is easy to implement, PCG iterations are cheap, and the storage demands are modest and fixed. However, there remain pitfalls in its use in practical simulations. As discussed above, its performance depends on the conditioning and/or spectrum of the matrix. This can be improved by the use of an appropriate preconditioner, but this adds to the work and storage required by the algorithm. Consequently, the convergence of low-energy modes can be slow and, more importantly, poorly reflected in the residual associated with an approximate solution. The established alternative is the use of direct solution methods. For our application, the choice of direct or iterative methods should not be made based on the solution of a single linearized system but, rather, based on the full nonlinear, time-dependent equations to be solved. When using simple nonlinear constitutive relations, direct methods may be preferable, if the factorization of one stiffness matrix can be reused many times. If, on the other hand, the stiffness matrix changes significantly with every linearization, the use of PCG may be preferable, particularly if the number of Newton iterations can be controlled. Moreover, many engineering applications do not use an exact evaluation of the Jacobian, hence a relative error of order 10^{-2} of the solution of Eq. 9 would be sufficient. The study presented in this paper assumes that a single factorization cannot be reused enough to make direct methods competitive. This assumption is supported by the experiments that we present in Sect. 5, that show that the costs of factorization are substantially greater than a single iterative solve and, consequently, that iterative solvers can reduce the total required computational time.

3.2 Deflated preconditioned conjugate gradient method

We have shown in [28] that the number of iterations for convergence of PCG is highly dependent on the number of aggregates in a mixture as well as the ratio of the Young's moduli. Increasing the number of aggregates introduces correspondingly more (clustered) small eigenvalues in stiffness matrix K. The jumps in the Young's moduli are related to the size of the small eigenvalues. We know from [48] that the smallest eigenvalues correspond to the slow converging components of the solution. Thus, we look to design a preconditioner that directly addresses these modes.

For any FE computation, we consider subsets of unconstrained elements as rigid bodies. Their corresponding (sub) stiffness matrices are assemblies of the element stiffness matrices. In the context of asphalt concrete, the aggregates are subsets of elements, with their Young's modulus as a shared property; the bitumen and the air voids are defined similarly.

When a matrix, K_{unc}, represents a rigid body, i.e. an unconstrained mechanical problem (with no essential boundary conditions), the strain energy equals zero for the rigid body displacements, as the system remains undeformed, and the matrix is positive semi-definite, $\forall u : u^T K_{\text{unc}} u \geq 0$. More specifically, the number of rigid body modes of any unconstrained volume equals the number of zero-valued eigenvalues of its corresponding stiffness matrix. When a matrix has zero-valued eigenvalues, the kernel $\mathcal{N}(A)$ is non-trivial. Moreover, the basis vectors of the kernel of a stiffness matrix represent the principal directions of the rigid body modes. In general, two types of rigid body modes exist: translations and rotations. In three dimensions, this implies six possible rigid body modes and, hence, six kernel vectors can be associated with the rigid body modes. For the partial differential equations considered in this paper, the physical analogue to these kernels are the rigid body modes of the linear elastic components of the material.

In [28], we conclude that the number of rigid bodies times the number of rigid body modes (six in three dimensions) is equal to the number of small eigenvalues of stiffness matrix K. By using the deflation technique, we deflate the Krylov subspace with pre-computed rigid body modes of the aggregates and remove all corresponding small eigenvalues from the system. As a result, the number of iterations of the DPCG method is nearly not affected by jumps in material stiffness or by the number of aggregates. This is a significant improvement over many other preconditioning techniques whose performance degrades even for simpler heterogenous problems.

To define the deflation preconditioner, we split the solution of (6) into two parts [20],

$$u = \left(I - P^T\right) u + P^T u, \tag{10}$$

where P is a projection matrix that is defined as

$$P = I - KZ(Z^T KZ)^{-1} Z^T, \quad \text{for } Z \in \mathbb{R}^{n \times m}, \tag{11}$$

where $\mathcal{R}(Z)$ represents the deflation subspace, i.e., the space to be projected out of the system, and I is the identity matrix of appropriate size. We assume that $m \ll n$ and that Z has rank m. Under these assumptions, $E \equiv Z^T KZ$ is symmetric and positive definite and may be easily computed and factored. Hence,

$$\left(I - P^T\right) u = ZE^{-1} Z^T Ku = ZE^{-1} Z^T f \tag{12}$$

can be computed directly, and the difficult computation is of $P^T u$. Because KP^T is symmetric,

$$KP^T = PK, \tag{13}$$

Algorithm 1 Deflated preconditioned CG solving $K\mathbf{u} = \mathbf{f}$

```
Select $\mathbf{u}_0$. Compute $\mathbf{r}_0 = (\mathbf{f} - K\mathbf{u}_0)$, set $\hat{\mathbf{r}}_0 = P\mathbf{r}_0$
Solve $M\mathbf{y}_0 = \hat{\mathbf{r}}_0$ and set $\mathbf{p}_0 = \mathbf{y}_0$
for $j = 0, 1, \ldots$ until convergence do
  $\hat{\mathbf{w}}_j = PK\mathbf{p}_j$
  $\alpha_j = \frac{(\hat{\mathbf{r}}_j, \mathbf{y}_j)}{(\hat{\mathbf{w}}_j, \mathbf{p}_j)}$
  $\hat{\mathbf{u}}_{j+1} = \hat{\mathbf{u}}_j + \alpha_j \mathbf{p}_j$
  $\hat{\mathbf{r}}_{j+1} = \hat{\mathbf{r}}_j - \alpha_j \hat{\mathbf{w}}_j$
  Solve $M\mathbf{y}_{j+1} = \hat{\mathbf{r}}_{j+1}$
  $\beta_j = \frac{(\hat{\mathbf{r}}_{j+1}, \mathbf{y}_{j+1})}{(\hat{\mathbf{r}}_j, \mathbf{y}_j)}$
  $\mathbf{p}_{j+1} = \mathbf{y}_{j+1} + \beta_j \mathbf{p}_j$
end for
$\mathbf{u} = ZE^{-1}Z^T\mathbf{f} + P^T\hat{\mathbf{u}}_{j+1}$
```

and P is a projection, we solve the deflated system,

$$PK\hat{u} = Pf, \tag{14}$$

for $\hat{u}$ using the PCG method and multiply the result by P^T giving $u = ZE^{-1}Z^T f + P^T\hat{u}$. We note that (14) is singular; however, the projected solution $P^T\hat{u}$, is unique, as it has no components in the null space, $\mathcal{N}(PK) = \text{span}\{Z\}$. Moreover, from [30,48], the null space of PK never enters the iteration process, and the corresponding zero-eigenvalues do not influence the solution. The DPCG method [46] is given as Algorithm 1.

To obtain a useful bound for the error of DPCG for positive semi-definite matrices, we define the effective condition number of a semi-definite matrix $D \in \mathbb{R}^{n\times n}$ with rank $n - m, m < n$, to be the ratio of the largest and smallest positive eigenvalues; analogous to Eq. 8,

$$\kappa_{\text{eff}}(D) = \frac{\lambda_n}{\lambda_{m+1}}. \tag{15}$$

Theorem 2.2 from [20] implies that a bound on the effective condition number of PK can be obtained.

Theorem 3.1 *Let P be defined as in (11), and suppose there exists a splitting $K = C + R$, such that C and R are symmetric positive semi-definite with null space of C, $\mathcal{N}(C) = span\{Z\}$. Then for ordered eigenvalues λ_i,*

$$\lambda_i(C) \le \lambda_i(PK) \le \lambda_i(C) + \lambda_{max}(PR). \tag{16}$$

Moreover, the effective condition number of PK is bounded by,

$$\kappa_{eff}(PK) \le \frac{\lambda_n(K)}{\lambda_{m+1}(C)}. \tag{17}$$

Proof See [20] (p. 445). □

While the large discontinuities in matrix entries due to strongly varying material properties in the FE discretization induce unfavorable eigenvalues (either large or small) in the spectrum of stiffness matrix K, the effective condition number of PK is bounded by the smallest eigenvalue of C and the largest eigenvalue of K. To remove the discontinuities and, thus, eliminate those unfavorable eigenvalues, we decouple the sub-matrices of stiffness matrix K that correspond to different materials by finding the correct splitting. The eigenvalues of the decoupled sub-matrices then determine the spectrum of PK. However, due to the large differences in stiffness, the values of the eigenvalues for the different sub-matrices can still vary over several order of magnitudes. To achieve a scalable solution algorithm, we couple this deflation procedure with another preconditioner to map the spectra of the sub-matrices onto the same region, around 1. This deflation technique can be used in conjunction with any ordinary preconditioning technique, giving a two-level approach, treating the smallest and largest eigenvalues by deflation and preconditioning, respectively. By choosing a favorable combination of deflation and preconditioning, a better spectrum is obtained, yielding a smaller effective condition number and fewer iterations. For a symmetric preconditioner $M = LL^T$, e.g. diagonal scaling, the result of Theorem 3.1 extends [20, Thm 2.3] to

$$\kappa_{\text{eff}}(L^{-1}PKL^{-T}) \le \frac{\lambda_n(L^{-1}KL^{-T})}{\lambda_{m+1}(L^{-1}CL^{-T})}. \tag{18}$$

Thus, as discussed above, we construct our deflation space Z from the null spaces of the (unconstrained) stiffness matrices of chosen sets of elements. In [29], an algorithm is given for computing rigid body modes of sets of elements. The matrix, C, is then defined by the assembly of all finite elements that belong to each body of material. The matrix, R, consists of the assembly of all finite elements that share nodes with the elements on the boundary of a body of material but that are not contained within the sub-mesh. We note that if some elements of a less stiff material are assigned to the element set of a stiffer material, the material stiffness matrices are not decoupled. So, for instance, when a node belongs to two elements and two different materials and is assigned to the wrong (less stiff) element with respect to the splitting of K, then the preconditioning step will reintroduce the coupling, nullifying the effect of the deflation operator.

The DPCG method extends PCG, enhancing stability and robustness when solving for symmetric, and positive definite systems, but requires extra storage for the deflation matrix Z. Moreover, PKu in Algorithm 1 needs to be computed in every iteration, but the Cholesky decomposition of matrix E and the computation of the matrix-matrix product KZ are done before entering the iteration loop, saving computation time. The unfavorable eigenvalues, due to the discontinuities in the stiffness matrix, are treated by the deflation method, making these costs worthwhile. The convergence of the DPCG method is assured for even highly ill-conditioned problems, and the method yields more accurate solutions than the PCG method.

3.3 Algebraic multigrid method

Multigrid methods [15,47,51] are among the most efficient iterative methods for the solution of the linear systems that arise from the FE discretization of many PDEs. They achieve this efficiency due to the use of two complementary processes, relaxation and coarse-grid correction. In the relaxation phase, a simple stationary iteration, such as the Jacobi or Gauss-Seidel iterations, is used to efficiently damp large-energy errors. Errors associated with small-energy modes are corrected through a coarse-grid correction process, in which the problem is projected onto a low-dimensional subspace (the coarse grid), and these errors are resolved through a recursive approach. This decomposition is, in many ways, the same as that in deflation, $u = (I - P^T)u + P^T u$; the relationship between deflation and multigrid has been explored in [46,45]. For homogeneous PDEs discretized on structured grids, the separation into large-energy and small-energy errors is well-understood, leading to efficient geometric multigrid schemes that offer both optimal algorithmic and parallel scalability.

For PDEs with heterogeneous coefficients that are discretized on unstructured meshes, algebraic multigrid (AMG) approaches [40,44,49] offer similar scalability, although at a higher cost per iteration (see, for example, [34] for a comparison of structured and unstructured multigrid approaches). While the fundamental complementarity of the multigrid approach doesn't change within AMG, the way in which the coarse-grid problems are defined does. In geometric multigrid schemes, the coarse-grid operators and intergrid transfer operators (interpolation and restriction) are determined based on explicit knowledge of the grid geometry and discretized PDE. In contrast, interpolation operators for AMG are defined in matrix-dependent ways [5,40], while the restriction and coarse-grid operators are given by variational conditions (when K is symmetric and positive definite) [35]. Thus, the challenge in achieving efficient multigrid performance is focused on the definition of appropriate matrix-dependent interpolation operators.

In the case of scalar PDEs, there are a wide variety of possible approaches for defining AMG-style interpolation operators [40,49,13,11,33,37]. These approaches are largely based on assumptions about the ellipticity of the underlying differential operator and, for scalar PDEs, they typically result in defining interpolation to closely match a given vector (often the constant vector) with the range of interpolation. For systems of PDEs, such as those that model the displacement of the composite materials considered here, more care must be taken, as the ellipticity of the equations of linear elasticity, for example, depends strongly on both the boundary conditions and Lamé coefficients of the system. As a result, there has been much research into the development of efficient AMG approaches for problems in solid mechanics.

For systems of PDEs, there are several possible AMG approaches. Within the setting of classical AMG (often called Ruge-Stüben AMG) [40,12], these approaches are commonly labeled as the variable-based, point-based (or, equivalently, node-based), and unknown-based approaches [16]. The variable-based approach applies AMG as a black-box, ignoring the structure of the PDE system and, as such, is understood to be effective only for very weakly coupled systems (such as systems with no differential coupling) [16]. The unknown-based approach applies scalar AMG to each component of the system, in a block Jacobi or block Gauss-Seidel manner, and was originally applied to systems of linear elasticity in [39]. Most commonly used is the point-based or node-based approach, where all variables discretized at a common spatial node are treated together in the coarsening and interpolation processes. This approach was first proposed for linear elasticity in [39] and has been extended in [24,31]. An extension framework to improve AMG for elasticity problems was proposed in [8], which uses a hybrid approach with nodal coarsening, but interpolation based on the unknown-based approach.

Despite these recent developments in the use of classical AMG for structural mechanics problems, a much more common algebraic multigrid approach for elasticity problems is in the framework of smoothed aggregation multigrid [49]. In smoothed aggregation, the coarse grid is created by aggregating degrees-of-freedom nodewide, through a greedy process that aims to create aggregates of size 3^d for a d-dimensional problem. Such an aggregation is used to define a partition of unity on the grid. A tentative interpolation operator is then defined in groups of six columns (for three-dimensional mechanics) by restricting the set of global rigid body modes to each aggregate based on this partition. Smoothed aggregation improves this tentative interpolation operator by applying a single relaxation step (or smoother) to it, leading to an overlapping support of the block-columns of interpolation and giving much more robust performance than unsmoothed (or plain) aggregation. As smoothed aggregation has been demonstrated as a successful parallel preconditioner for a number of structural mechanics applications [2,4,14], we focus on this approach here.

4 Parallel computing

4.1 Parallel paradigm: domain decomposition

We use parallelism based on domain decomposition as found in [17]. Global domain Ω is divided into D subdomains, yielding $\Omega = \bigcup_{d=1}^{D} \Omega_d$. Domain Ω holds E elements, each subdomain holds E_d elements, hence $E = \sum_{d=1}^{D} E_d$. Elements can share nodes and the associated degrees of freedom that lie in multiple subdomains, but no element is contained in more than one subdomain. Elementwise operations can be done independently for each subdomain, but the values of any quantity at shared nodes must be synchronized between subdomains after finishing the operation. The synchronization yields communication between the boundaries of the subdomains. Examples of elementwise operations are numerical integration, matrix–vector multiplications etc.

4.2 Parallel implementation of PCG

The PCG algorithm is constructed from basic linear algebraic operations. The matrix–vector operation and inner product require communication between neighboring and all subdomains respectively. All other linear algebraic operations (e.g., vector scaling and addition) can be done locally; i.e., there is no communication with other subdomains. This makes the PCG method easy to parallelize. The other operation that needs to be taken care of explicitly is the preconditioner. In this research, we consider AMG and for comparison also diagonal scaling. We note that diagonal scaling is an inherently parallel operation. We elaborate on the parallel implementation of AMG below.

4.3 Parallel implementation of DPCG

The DPCG method given by Algorithm 1 is similar to the standard PCG algorithm, but the parallelization of the DPCG method involves two extra steps. First, the construction of the deflation matrix, Z, on each subdomain and, second, the evaluation of PKp_j for each iteration of DPCG.

Obviously, the mapping of the deflation matrix, Z, onto the subdomains is defined by the partitioning of elements over the subdomains. The computation of rigid body modes only requires the element matrices; hence, no communication is needed for the assembly of the distributed deflation matrix. We only store the non-zero elements of Z, giving a small memory overhead.

The evaluation of PKp_j can be optimized. Consider

$$PKp_j = Kp_j - KZE^{-1}Z^TKp_j,$$

where $K \in \mathbb{R}^{n\times n}$, $Z \in \mathbb{R}^{n\times k}$. Here, $Kp_j = y$ is computed as usual, while $KZ = \tilde{Z} \in \mathbb{R}^{n\times k}$ and $E^{-1} = (Z^TKZ)^{-1}$ are computed only once, before entering the Krylov process (iteration loop). Hence, for each iteration of DPCG, we have three extra operations compared to PCG, computing

$$\begin{aligned} &Z^Ty = \tilde{y}, \quad \tilde{y} \in \mathbb{R}^{k\times 1},\\ &E^{-1}\tilde{y} = \hat{y}, \quad \hat{y} \in \mathbb{R}^{k\times 1},\\ &\tilde{Z}\hat{y} = \bar{y}, \quad \bar{y} \in \mathbb{R}^{n\times 1}. \end{aligned}$$

Communication between subdomains is needed for the computation of KZ, E, and Z^T. The entries of the $k \times k$ matrix E are distributed over the subdomains, and its decomposition is determined in parallel. The computation of Z^Ty requires one communication involving all subdomains to compute the k parallel inner products of $k \times 1$ vectors.

4.4 Parallel AMG

In recent years, two general-purpose parallel algebraic multigrid codes have been developed, alongside a number of other codes aimed at specific applications. One of these codes, BoomerAMG [25] (included in the Hypre package [19]), focuses on classical (Ruge–Stüben) AMG algorithms and their variants, while the other, ML [21] (included in the Trilinos project [26]), focuses on the smoothed aggregation setting. In our experiments below, we make use of ML and Trilinos for the parallel SA implementation.

There have been a number of studies of the performance of parallel AMG codes for solid mechanics operations. Initial two-dimensional results were reported in [52], based on an AMG treatment that first coarsens appropriately along boundaries shared between processors and then treats the processor interiors. Scalability studies for a simplified AMG approach, based on maximal independent set coarsening of nodes, remeshing the coarse node set, and using geometric grid-transfer operators, for both linear elasticity and nonlinear elastic and plastic solid mechanics are detailed in [1]. This method was compared with smoothed and unsmoothed aggregation in [2], where it was found that the simplified approach was less robust than the aggregation approaches, and that smoothed aggregation was most robust and typically not much more expensive than the other two approaches. A comparison of Ruge–Stüben AMG, smoothed aggregation, and a generalized smoothed aggregation approach (using information from local eigensolves to complement the restricted rigid-body modes) was performed in [14], where smoothed aggregation was shown to generally outperform Ruge–Stüben AMG. The generalized form offers even greater robustness, but relies on an expensive preprocessing step. One important issue is the choice of parallel smoother; this was studied in depth in [3], comparing parallel hybrid Gauss-Seidel orderings with polynomial (Chebyshev) smoothers and concluding that polynomial smoothers offer many advantages.

For our study, we investigate primarily two options within the ML software. In the first, denoted by SA (AMG), we treat the smoothed aggregation solver as a “black box”, providing the minimal amount of information required and following the default software options. In this case, we provide just the matrix (in node-ordered form) and the geometric coordinates of the mesh nodes. In the second, denoted by SA (AMG) / VBMetis, we explicitly provide complete information about the PDE structure and null space. In particular, we provide the degree-of-freedom to node map, including full details about eliminated degrees of freedom due to boundary conditions, and we give directly the rigid body modes of the entire solid body. In both cases, we make use of Chebyshev smoothers.

5 Numerical experiments

The meshes of experiments 1 and 2, given by Figs. 1 and 5 are derived from real-life samples of asphaltic material obtained by CT scan. Both experiments involve different mesh sizes, yielding 230.000 and 2,9 million degrees of freedom, respectively. The meshes contain a mixture of materials that are subjected to an external force applied to the upper boundary of the volume. Zero displacement boundary conditions are imposed on three sides of the volume; that is, homogenous Dirichlet boundary conditions are given for all degrees of freedom in the x, z-, x, y- and y, z- planes for $y = 0$, $z = 0$ and $x = 0$, respectively. These materials give rise to the coupled partial differential equations given in [18]. In both experiments, we make use of the same set of material parameters. We distinguish between three materials: aggregates, bitumen, and air voids. The corresponding stiffness coefficients (Young's moduli) are given in Table 1 and are the dominating contributions to the entries of the stiffness matrix.

As described in Sect. 2, the underlying PDEs of both experiments come from the computational framework in [18]. In this paper, only hyperelastic materials are considered. The constitutive model involves non-linear material behavior; hence, the stiffness matrix of Eq. 6 is a tangential stiffness matrix derived from the linearization of the virtual work equation.

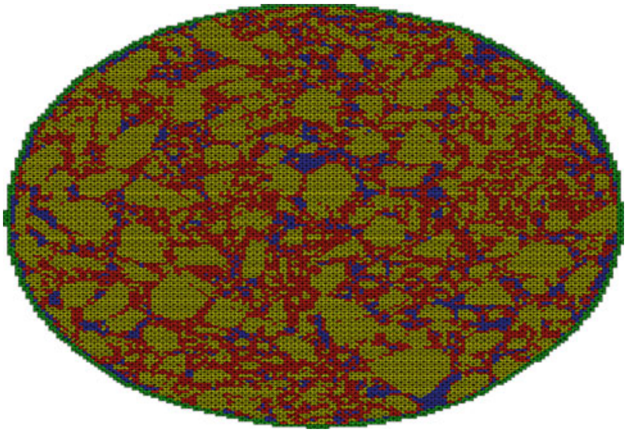


Fig. 1 Experiment 1: FE mesh representing cube of asphaltic material containing aggregates (*yellow*), bitumen (*red*) and air voids (*blue*). (Color figure online)

Table 1 Young's moduli for different materials

Aggregate	Bitumen	Air voids
69,000	5,000	100

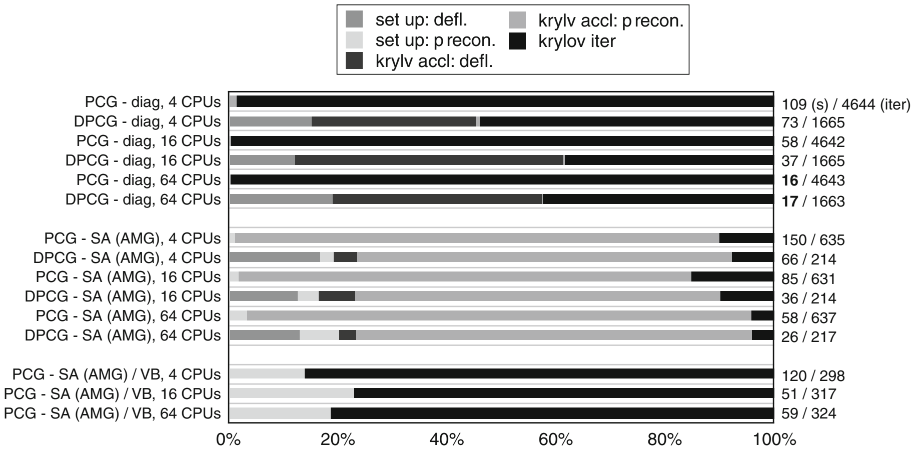
The deflation vectors are constructed by computing the rigid body modes of the aggregation of stones, bitumen, and air voids, as proposed in [28]. The number of deflation vectors for the experiments are 162 and 342, respectively. All vectors are sparse and Z is of full rank. As described in Sect. 4.3, the parallel implementation involves the distribution of the degrees of freedom, thus vectors; hence, the rigid body modes of an arbitrary aggregation of materials may be spread over multiple domains. Apart from variations due to the round-off errors induced by the domain decomposition, the parallel implementation should yield the same number of iterations as a sequential implementation given the same deflation space. As a result, the number of iterations of the DPCG method for a given problem should be invariant under an increasing number of subdomains.

The aim of the experiments is to compare the performance and robustness of our deflation method and (optimized) SA for mechanical problems with heterogeneous material coefficients. In Sect. 3.2, we have argued that DPCG is a two-level approach, and that we need a preconditioner to treat both ends of the spectrum of the stiffness matrix. We have seen that SA is designed to solve homogeneous elastic equations in an optimal way. A natural choice for preconditioning of DPCG would be using a "black box" implementation of SA for the "decoupled" stiffness matrices. Hence, we compare PCG and DPCG preconditioned by SA (AMG) as well as PCG preconditioned by SA (AMG) / VBMetis. We also include diagonal scaling as preconditioner to have a point of reference from which to compare all methods. The stopping criterion for all computations is $\frac{\|r_i\|}{\|r_0\|} < 10^{-6}$ where r_i is the residual vector at ith iteration. Although we have argued that direct solution algorithms are not the methods of choice for solving these large, 3D problems, we have included the run time of the decomposition of the stiffness matrix using SuperLU 2.5 (distributed memory) [32]. For a fair comparison of PCG and DPCG, we have implemented the methods in Trilinos [26]. All experiments were done at Tufts University (USA) on an IBM 60-node cluster with over 500 64-bit cores (16–48 GB RAM per node, Infiniband interconnect). Due to the complexity of the meshes and limitations of our meshing software, we only take into consideration the strong scaling effect of the solvers. In Experiment 1, we compare results for 4, 16 and 64 subdomains, where each subdomain corresponds to a computing core. In Experiment 2, we compare results for 4, 8 and 64 subdomains, because of memory limitations of the SA (AMG) / VBMetis solver.

5.1 Experiment 1

This experiment involves a slice of asphaltic material, represented by a mesh containing 315270 4-noded tetrahedral elements yielding roughly 230000 DOF. The wall clock times as well as the number of iterations of all solvers for all domain

Fig. 2 Experiment 1: Wall clock time and number of iterations of (D)PCG



decompositions are given in Fig. 2. Running with 64 subdomains, the PCG and DPCG solver with diagonal scaling are the fastest methods, requiring 16 and 17 seconds respectively. The number of iterations for PCG and DPCG for both diagonal scaling and SA (AMG) is essentially invariant with the number of subdomains, as expected. The strong scaling is clearly not optimal for this number of unknowns. For instance, the wall clock time for DPCG with diagonal scaling goes from 73 to 37 and down to 17 seconds, reducing by a factor of two when the number of subdomains increases by a factor of four. The costs of the deflation operator for diagonal scaling are almost equal to the costs of the matrix–vector products together with the inner products. The setup time of the deflation operator lies around 20% of the total cost and does not significantly increase when increasing the number of subdomains.

The DPCG method combined with the SA (AMG) preconditioner has a good performance in terms of iterations, but shows poor parallel performance. Going from 16 subdomains to 64 subdomains gains very little in terms of speed-up. Also, the SA (AMG) preconditioner has three times the setup cost and ten times the operational cost compared to the deflation operator.

The PCG method combined with the SA (AMG) / VBMetis preconditioner performs worse than the DPCG method with SA (AMG) in terms of iterations as well as wall clock time. Although the ratio between setup time and cost per iteration is almost equal for both methods, the overall cost of the SA (AMG)/VBMetis preconditioner is much higher. Again, due to the small problem size, there is no benefit from the parallel implementation. Overall, these results are not surprising. In terms of iterations, we see strong improvement as we improve the preconditioner for PCG, from over 4,600 iterations for diagonal scaling, to about 630 for SA(AMG), to about 320 for SA(AMG)/VBMetis. Similarly, we see uniform improvements adding deflation to PCG, reducing the iteration counts for both the diagonal scaling and SA(AMG) preconditioners by almost a factor of three. In terms of wall-clock time to solution, however, the much simpler approach of diagonal scaling becomes the clear winner over the approaches based on SA for this example; even though many more iterations of PCG or DPCG are needed, the lack of a setup cost and the much lower cost per iteration of the simpler approach pay off in the end.

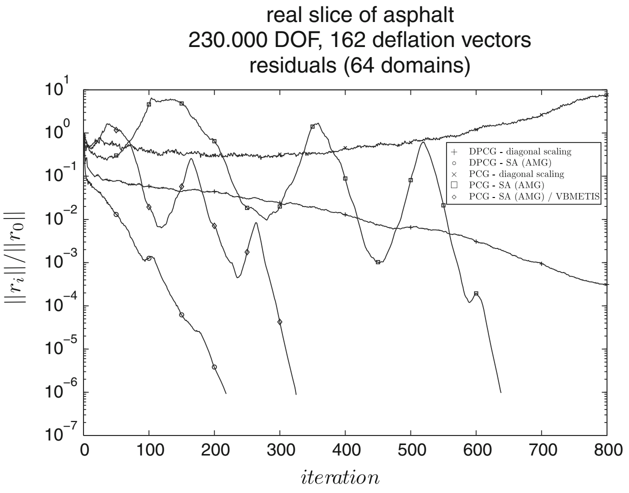


Fig. 3 Experiment 1: L_2 norms of the residuals

The L_2 norms of the residuals of Experiment 1 are given in Fig. 3. We observe that without deflation, i.e., PCG preconditioned by SA (AMG) or SA (AMG)/VBMetis, not all unfavorable eigenvalues have been removed from the spectrum of $M^{-1}K$. This can also be seen from Fig. 4, where the 50 smallest Ritz values of $M^{-1}K$ and $M^{-1}PK$ are given. Clearly, the deflated systems have no clustering of very small (approximated) eigenvalues whereas the non-deflated systems, even when preconditioned by SA (AMG)/VBMetis, still contain some unfavorable eigenvalues.

5.2 Experiment 2

This experiment involves a slice of asphaltic material, represented by a mesh containing 4531353 4-noded tetrahedral elements, yielding roughly 3 million degrees of freedom shown in Fig. 5. The wall clock time as well as the number

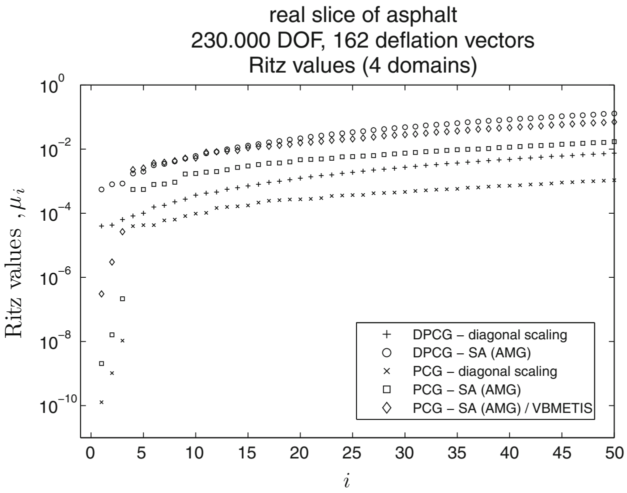


Fig. 4 Experiment 1: Ritz values derived from (D)PCG

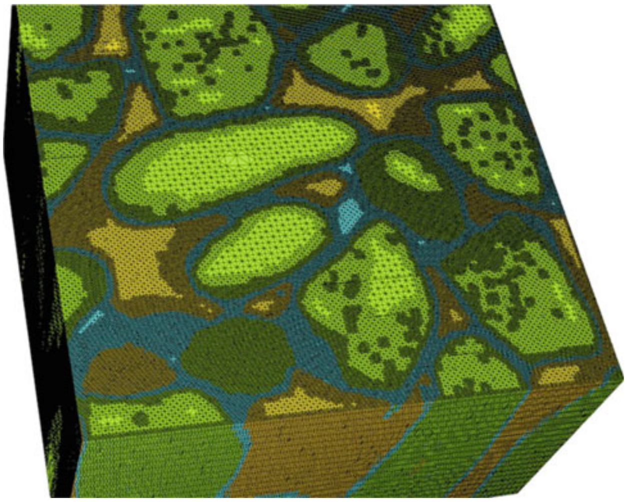
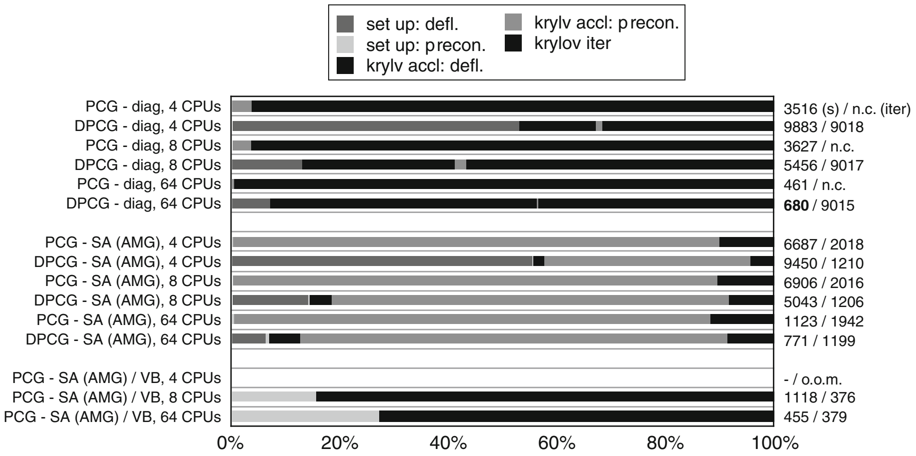


Fig. 5 Experiment 2: mesh representing cube of asphaltic material containing aggregates (*light green*), bitumen (*dark green*) and air voids (*blue*). (Color figure online)

of iterations of all solvers for all domain decompositions are given in Fig. 6. Again, we see expected performance from the iteration counts. For PCG, these improve from not converging (within 10,000 iterations) with diagonal scaling, to roughly 2,000 iterations with SA (AMG), to roughly 380 iterations with SA (AMG)/VBMetis. Here, the added costs of SA(AMG)/VBMetis are very notable, giving an out-of-memory error on four CPUs. Also as before, we see the immediate advantage of DPCG, leading to convergence in about 9,000 iterations for diagonal scaling, and about 1,200 iterations for SA (AMG). In this case, however, the added expense of SA (AMG)/VBMetis pays off, yielding about a 50% speedup over DPCG with diagonal scaling on 64 CPUs (and a greater speedup on eight CPUs). We note that the SA (AMG)/VBMetis approach is far from a "black-box" preconditioner, and is available only as beta software within the ML package; as such, further improvement may be possible by refining this software and addressing memory issues that were encountered in obtaining these results.

Several interesting observations are possible about the parallel scaling. For the DPCG approaches, the relative cost of the deflation operator increases with the number of subdomains, becoming as expensive the total cost of matrix–vector and inner products; however, the setup time decreases, due to the parallel Cholesky decomposition of E. Excellent strong scaling is observed for DPCG with diagonal scaling, giving speedup of 1.8 and 8.0 for increases in the number of subdomains by factors of 2 and 8, respectively. Slightly less impressive speedup of DPCG preconditioned by SA (AMG) is observed, with factors of 1.87 and 6.54 for increases in the number of subdomains by factors of 2 and 8, respectively. This sub-optimal speed-up is due to the SA (AMG) preconditioner, which involves an extra set-up phase that scales poorly. The speedup of the SA (AMG)/VBMetis approach shows the poorest results, with a factor of only 2.45 when the number of subdomains increases by a factor of eight. This, again, is due to poor scaling of the SA(AMG)/VBMetis

Fig. 6 Experiment 2: Wall clock time and number of iterations of (D)PCG



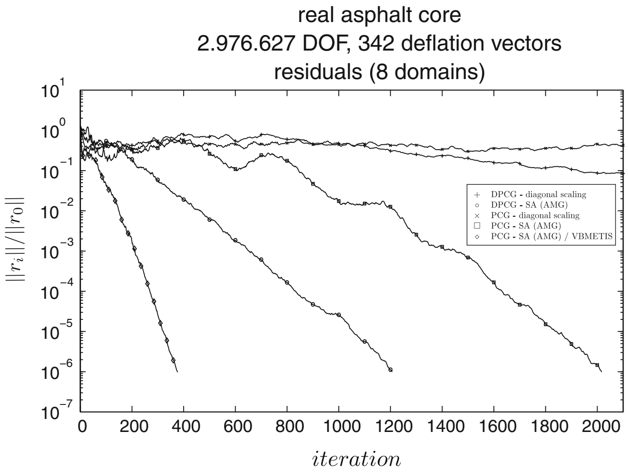


Fig. 7 Experiment 2: L_2 norms of the residuals

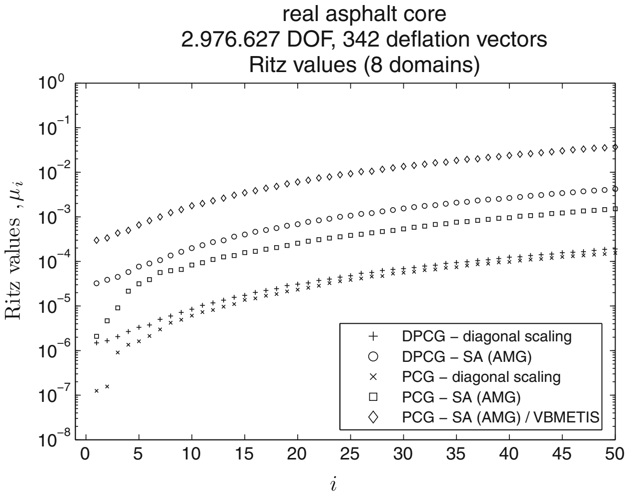


Fig. 8 Experiment 2: Ritz values derived from (D)PCG

preconditioner, although better scaling might be observed with more degrees-of-freedom per subdomain on the 64 CPU scale.

The L_2 norms of the residuals of Experiment 2 are given in Fig. 7, and the Ritz values of the preconditioned stiffness matrix derived form the (D)PCG iterations are given in Fig. 8. We observe that the stiffness matrix preconditioned by SA (AMG)/VBMetis yields a more favorable spectrum of eigenvalues compared to preconditioning with the deflation operator combined with SA (AMG). This can also be observed in Fig. 7, as the residual curve of PCG preconditioned by SA (AMG)/VBMetis is steeper than the curve of DPCG preconditioned by SA (AMG), which indicates that the eigenvalues of the spectrum of preconditioned K lie relatively closer to one. However, compared to PCG preconditioned by diagonal scaling and SA (AMG), all unfavorable eigenvalues have been removed from the spectrum by DPCG and by PCG preconditioned by SA (AMG)/VBMetis. For this reason we do not consider the obvious combination of DPCG and SA (AMG)/VBMetis; there are no small eigenvalues left to be treated by DPCG.

Table 2 Experiment 1 & 2: wall-clock times (s) using SuperLU, distributed memory version 2.5

	4 CPUs	8 CPUs	16 CPUs	64 CPUs
Experiment 1	317	114	69	56
Experiment 2	*o.o.m*	*o.o.m*	*o.o.m*	3979

Remark The results for a direct solver for both experiments are given in Table 2. Using the parallel direct solver, wall-clock times are uniformly worse than those of DPCG for the smaller test problem; for the larger test problem, memory issues prevented convergence on all but the largest numbers of cores, where the wall-clock time was again substantially worse than all of the iterative approaches considered here.

6 Conclusion

We have compared the PCG method and the DPCG method for the solution of large linear systems from mechanical problems with strongly varying stiffnesses of materials. We have compared three preconditioners with PCG and two with DPCG, using two implementations of smoothed aggregation (SA), an adaptation of algebraic multigrid designed for solving elasticity equations, and diagonal scaling. For one implementation of SA, we choose the default parameter set and, for the other implementation, we choose an optimal parameter set for the experiments involved. DPCG offers clear enhancements over standard PCG in terms of both the number of iterations required for convergence and the wall-clock time to solution. It is well-suited for parallel computing and can be easily implemented within any existing FE software package with basic parallel linear algebraic operations. The combination of DPCG with diagonal scaling offers an exceptionally low cost per iteration, giving much better wall-clock performance than PCG, even with the SA (AMG) preconditioner. We suspect that DPCG and diagonal scaling combine so well because deflation and scaling are complementary operations, working on the lower and upper part of the spectrum respectively. DPCG with the SA (AMG) preconditioner strongly improves the iteration counts over diagonal scaling but, for our experiments, does not improve the wall-clock time to solution. For our larger test problem, the optimized SA (AMG)/VBMetis preconditioner does outperform the much simpler DPCG with diagonal scaling approach; however, this approach requires significantly more software development effort and, as such, may not be readily available for all simulation codes. Thus, we have demonstrated that the DPCG approach is efficient, scalable, and robust and, as such, is an effective tool in large-scale simulation of the mechanics of composite materials.

References

1. Adams MF (2000) Parallel multigrid solvers for 3d unstructured finite element problems in large deformation elasticity and plasticity. Int J Numer Methods Eng 48(8):1241–1262
2. Adams MF (2002) Evaluation of three unstructured multigrid methods on 3D finite element problems in solid mechanics. Int J Numer Methods Eng 55(5):519–534
3. Adams M, Brezina M, Hu J, Tuminaro R (2003) Parallel multigrid smoothing: polynomial versus Gauss-Seidel. J Comput Phys 188:593–610
4. Adams MF, Bayraktar HH, Keaveny TM, Papadopoulos P (2004) Ultrascalable implicit finite element analyses in solid mechanics with over a half a billion degrees of freedom. In: ACM/IEEE proceedings of SC2004: high performance networking and computing. Institute of Biomedical Engineering, Umeå
5. Alcouffe RE, Brandt A, Dendy JE, Painter JW (1981) The multigrid method for the diffusion equation with strongly discontinuous coefficients. SIAM J Sci Stat Comput 2:430–454
6. Amestoy PR, Duff IS, L'Excellent J-Y (1998) Multifrontal parallel distributed symmetric and unsymmetric solvers. Comput Methods Appl Mech Eng 196(8), 1429–1435
7. Arbenz P, Harryvan Lenthe G, Mennel U, Müller R, Sala M (2008) A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures. Int J Numer Methods Eng 73(7):927–947
8. Baker AH, Kolev TZV, Yang UM (2010) Improving algebraic multigrid interpolation operators for linear elasticity problems. Numer Linear Algebra Appl 17(2–3):495–517
9. Bathe KJ (1995) Finite element procedures. Prentice Hall, Upper Saddle River
10. Bollhöfer M, Saad Y (2006) Multilevel preconditioners constructed from inverse-based ilus. SIAM J Sci Comput 27(5):1627–1650
11. Brandt A, Brannick J, Kahl K, Livshits I (2011) Bootstrap AMG. SIAM J Sci Comput 33(2):612–632
12. Brandt A, McCormick SF, Ruge JW (1984) Algebraic multigrid (AMG) for sparse matrix equations. In: Evans DJ (ed) Sparsity and its applications. Cambridge University Press, Cambridge
13. Brannick J, Zikatanov L (2007) Algebraic multigrid methods based on compatible relaxation and energy minimization. In: Domain decomposition methods in science and engineering XVI. Lect Notes Comput Sci Eng 55:15–26. Springer, Berlin
14. Brezina M, Tong C, Becker R (2006) Parallel algebraic multigrids for structural mechanics. SIAM J Sci Comput 27(5):1534–1554
15. Briggs WL, Henson VE, McCormick SF (2000) A multigrid tutorial, second edn. SIAM Books, Philadelphia
16. Clees T (2005) AMG strategies for PDE systems with applications in industrial semiconductor simulation. PhD thesis, Universität zu Köln, Köln
17. Douglas CC, Haase G, Langer U (2003) A tutorial on elliptic Pde solvers and their parallelization (software, environments, and tools). Society for Industrial and Applied Mathematics, SIAM Books, Philadelphia
18. Drescher A, Kringos N, Scarpas T (2009) On the behavior of a parallel elasto-visco-plastic model for asphaltic materials, mechanics of materials. Delft University of Technology, Delft
19. Falgout RD, Yang UM (2002) Hypre: a library of high performance preconditioners. In computational science—ICCS 2002: international conference, Amsterdam. In: Proceedings, part III, number 2331 in lecture notes in computer science. Springer, New York, pp 632–641
20. Frank J, Vuik C (2001) On the construction of deflation-based preconditioners. SIAM J Sci Comput 23(2):442–462
21. Gee MW, Siefert CM, Hu JJ, Tuminaro RS, Sala MG (2006) ML 5.0 smoothed aggregation user's guide. Technical Report SAND 2006–2649. Sandia National Laboratories, Sandia
22. Golub GH, Van Loan CF (1996) Matrix computations (Johns Hopkins studies in mathematical sciences). The Johns Hopkins University Press, Baltimore
23. Graham IG, Scheichl R (2007) Robust domain decomposition algorithms for multiscale pdes. Numer Methods Part Differ Equ 23:859–878
24. Griebel M, Oeltz D, Marc Alexander S (2003) An algebraic multigrid method for linear elasticity. SIAM J Sci Comput 25(2):385–407
25. Henson VE, Yang UM (2002) BoomerAMG: a parallel algebraic multigrid solver and preconditioner. Appl Numer Math 41:155–177
26. Heroux MA, Bartlett RA, Howle VE, Hoekstra RJ, Hu JJ, Kolda TG, Lehoucq RB, Long KR, Pawlowski RP, Phipps ET, Salinger AG, Thornquist HK, Tuminaro RT, Willenbring JW, Williams A, Kendall SS (2005) An overview of the trilinos project. ACM Trans Math Softw 31(3):397–423
27. Hestenes MR, Stiefel E (1952) Methods of conjugate gradients for solving linear systems. J Res Natl Bur Stand 49:409–436
28. Jönsthövel TB, van Gijzen MB, Vuik C, Kasbergen C, Scarpas A (2009) Preconditioned conjugate gradient method enhanced by deflation of rigid body modes applied to composite materials. Comput Model Eng Sci 47:97–118
29. Jönsthövel TB, van Gijzen MB, Vuik C, Scarpas A (2011) On the use of rigid body modes in the deflated preconditioned conjugate gradient method. Technical Report 11-04. Delft University of Technology, Delft
30. Kaasschieter EF (1988) Preconditioned conjugate gradients for solving singular systems. J Comput Appl Math 24(1-2):265–275
31. Karer E, Kraus JK (2010) Algebraic multigrid for finite element elasticity equations: determination of nodal dependence via edge-matrices and two-level convergence. Int J Numer Methods Eng 83(5):642–670
32. Li X, Demmel JW (2003) Superlu dist: a scalable distributed-memory sparse direct solver for unsymmetric linear systems. ACM Trans Math Softw 29:110–140
33. MacLachlan S, Manteuffel T, McCormick S (2006) Adaptive reduction-based AMG. Numer Linear Algebra Appl 13:599–620
34. MacLachlan SP, Tang JM, Vuik C (2008) Fast and robust solvers for pressure-correction in bubbly flow problems. J Comput Phys 227(23):9742–9761
35. Nicolaides RA (1979) On some theoretical and practical aspects of multigrid methods. Math Comput 33:933–952
36. Nicolaides RA (1987) Deflation of conjugate gradients with applications to boundary value problems. SIAM J Numer Anal 24(2):355–365
37. Olson LN, Schroder JB, Tuminaro RS (2011) A general interpolation strategy for algebraic multigrid using energy minimization. SIAM J Sci Comput 33(2):966–991
38. Polizzi E, Sameh AH (2005) A parallel hybrid banded system solver: the spike algorithm abstract. Parallel Comput 32:177–194
39. Ruge J (1986) Amg for problems of elasticity. Appl Math Comput 19(1–4):293–309

40. Ruge JW, Stüben K (1987) Algebraic multigrid (AMG). In: McCormick SF (ed) Multigrid methods, volume 3 of frontiers in applied mathematics. SIAM, Philadelphia, pp 73–130
41. Saad Y (2003) Iterative methods for sparse linear systems, second edn. Society for Industrial and Applied Mathematics, Philadelphia
42. Schenk O, Gärtner K, Fichtner W, Stricker A (2001) Pardiso: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. Future Gener Comput Syst 18(1):69–78
43. Simpleware (2009). http://www.simpleware.com. Accessed 11 December 2009
44. Stüben K (2001) An introduction to algebraic multigrid. In: Trottenberg U, Oosterlee C, Schüller A (eds) Multigrid. Academic Press, San Diego, pp 413–528
45. Tang JM, MacLachlan SP, Nabben R, Vuik C (2010) A comparison of two-level preconditioners based on multigrid and deflation. SIAM J Matrix Anal Appl 31:1715–1739
46. Tang JM, Nabben R, Vuik C, Erlangga YA (2009) Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. J Sci Comput 39:340–370
47. Trottenberg U, Oosterlee CW, Schüller A (2001) Multigrid. Academic Press, London
48. Van der Sluis A, Van der Vorst HA (1986) The rate of convergence of conjugate gradients. Numer Math 48(5):543–560
49. Vaněk P, Mandel J, Brezina M (1996) Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. Computing 56:179–196
50. Vuik C, Segal A, Meijerink JA (1999) An efficient preconditioned cg method for the solution of a class of layered problems with extreme contrasts in the coefficients. J Comput Phys 152:385–403
51. Wesseling P (1992) An introduction to multigrid methods. Wiley, Chichester
52. Wriggers P, Boersma A (1998) A parallel algebraic multigrid solver for problems in solid mechanics discretisized by finite elements. Comput Struct 69(1):129–137