INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2011; **88**:1112–1127
Published online 28 April 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.3209

Deflated preconditioned conjugate gradient solvers for linear elasticity

R. Aubry[1,2,*,†], F. Mut[1], S. Dey[2] and R. Löhner[1]

[1]*CFD Center, Department of Computational and Data Science, M.S. 6A2, College of Science, George Mason University, Fairfax, VA 22030-4444, U.S.A.*
[2]*U.S. Naval Research Laboratory, 4555 Overlook Ave SW, Washington, DC 20375, U.S.A.*

SUMMARY

Extensions of deflation techniques previously developed for the Poisson equation to static elasticity are presented. Compared to the (scalar) Poisson equation (*J. Comput. Phys.* 2008; **227**(24):10196–10208; *Int. J. Numer. Meth. Engng* 2010; DOI: 10.1002/nme.2932; *Int. J. Numer. Meth. Biomed. Engng* 2010; **26**(1):73–85), the elasticity equations represent a system of equations, giving rise to more complex low-frequency modes (*Multigrid*. Elsevier: Amsterdam, 2000). In particular, the straightforward extension from the scalar case does not provide generally satisfactory convergence. However, a simple modification allows to recover the remarkable acceleration in convergence and CPU time reached in the scalar case. Numerous examples and timings are provided in a serial and a parallel context and show the dramatic improvements of up to two orders of magnitude in CPU time for grids with moderate graph depths compared to the non-deflated version. Furthermore, a monotonic decrease of iterations with increasing subdomains, as well as a remarkable acceleration for very few subdomains are also observed if all the rigid body modes are included.

Received 15 October 2010; Revised 15 March 2011; Accepted 18 March 2011

KEY WORDS: iterative solvers; preconditioned conjugate gradient; deflation; elasticity; subdomain agglomeration

1. INTRODUCTION

This paper represents the first and natural step towards an iterative solver for coupled fluid structure scattering. After having applied deflation to scalars in [1–3], the next step consists in extending it to systems. The numerical solution of the Navier equations describing elasticity for complex 3D domains is required for many computational predictions in geomechanics, civil engineering, turbomachinery and aeronautics. Owing to the very large equation systems that arise in such problems, only iterative methods are considered in this paper. In order to fix the notation, we write the Navier equations as

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}) = \mathbf{f} \tag{1}$$

where $\mathbf{u}$ is the displacement, $\mathbf{f}$ is the force applied to the solid and $\boldsymbol{\sigma}$ is the stress tensor

$$\boldsymbol{\sigma}(\mathbf{u}) = \lambda \mathrm{Tr}(\boldsymbol{\varepsilon}(\mathbf{u})) + 2\mu\boldsymbol{\epsilon} \tag{2}$$

*Correspondence to: R. Aubry, CFD Center, Department of Computational and Data Science, M.S. 6A2, College of Science, George Mason University, Fairfax, VA 22030-4444, U.S.A.
†E-mail: raubry@gmu.edu

λ and μ are the Lamé parameters of the material, and the strain tensor $\boldsymbol{\epsilon}$ is defined as:

$$\boldsymbol{\epsilon}(\mathbf{u}) = \frac{\nabla\mathbf{u} + \nabla^{\mathrm{T}}\mathbf{u}}{2} \tag{3}$$

After discretization, these equations give rise to large sparse matrices and their inversion may be time consuming. For large three-dimensional problems, iterative methods in a broad sense (geometric multigrid, algebraic multigrid, iterative solvers, domain decomposition methods) represent the methods of choice mainly due to memory requirements, but possibly also due to CPU requirements. However, even in the symmetric positive-definite case that is under consideration here, a large number of costly iterations is often necessary to reach convergence. Therefore, different solutions have been proposed to solve them efficiently.

One of the first attempts to use iterative methods for the Navier equations is reported by Axelsson *et al.* [4], where the influence of the Poisson parameter on the convergence of the iterative solver is studied. Numerous preconditioners, such as exact and approximate block diagonal preconditioners, and full block factorization are attempted. Dickinson [5] considers high aspect ratio elements. The preconditioned conjugate gradient (PCG) is preconditioned by incomplete factorization based on levels of fill-in, drop tolerance and hierarchical basis, and compared to direct solvers. Saint-George *et al.* [6] highlight the advantages of the PCG applied to the Navier equations and confront to the performances of the direct solvers, which were for a long time the method of choice in structural analysis with finite elements. They stress the fact that only low-order factorizations are relevant, as iterations decrease slowly with added fill-in. The preconditioners studied in this reference come from advanced incomplete factorizations such as Incomplete Cholesky (IC), Modified Incomplete Cholesky (MIC), or sometimes based on a dynamic factorization of the Stieltjes matrix associated with the original matrix. The superiority of the PCG over direct frontal and skyline methods is clearly shown, even in a two-dimensional context. More recently, Kilic *et al.* [7] also compare performances between various preconditioned conjugate gradients and conjugate residuals against sparse direct solvers for structural dynamics problems, highlighting the importance of iterative solvers in terms of storage and CPU time in a three-dimensional context. Preconditioners include IC, block Jacobi and SSOR. Finally, Hladik *et al.* [8] explore the possibilities of a relaxed version of IC and scaling of an element-by-element preconditioner.

Once direct solvers have been discarded for three-dimensional applications, there are still numerous choices between PCG, geometric or algebraic multigrids and domain decomposition methods. For this latter class of method, the so-called FETI method [9] represents the archetype example. This reference proposes a remarkable manner of extracting the rigid body modes. When the direct solver encounters a null pivot, the column is kept as a basis of the null space. Later, this method was extended in [10] to provide a more robust version. Recently, Dohrmann [11] improve the balancing preconditioner proposed in [12] for three-dimensional elasticity and non-homogeneous materials, and Dostál *et al.* [13] introduce an easier version of FETI, where Dirichlet boundary conditions are also weakly imposed to maintain the same dimension of the null space for all subdomains. In Jouglard *et al.* [14], a comparison between PCG and geometric multigrid is conducted, showing that PCG with a good multilevel preconditioner outperforms the geometric multigrid on a CPU time basis. Similar conclusions were obtained by Waltz [15]. A very interesting reference is Bulgakov *et al.* [16], where preconditioning is sought as a coarse grid correction. This reference incorporates already the rigid body modes in the iterative process. The introduction of the rigid body modes accelerates drastically the iterative procedure. However, the coarse grid correction, namely the action of the preconditioner considered in this paper is non-symmetric and a conjugate gradient squared (CGS) is used for the iterative solver. This reference is very close to a deflation process. As a matter of fact, as shown by Saad in [17], the deflation process may be interpreted as a preconditioned conjugate gradient with a singular symmetric preconditioning matrix. Recently, Jönsthövel *et al.* [18] apply a deflated conjugate gradient to static elasticity with deflation based on rigid body modes for non-homogeneous materials. They use deflation to remove the smallest eigenvalues associated with each physical aggregate. Owing to the large jump in material properties, each physical aggregate is more and more disconnected from the bitumen, tending

DOI: 10.1002/nme

towards Neuman boundary conditions and singular submatrices. However, they do not show that deflation of the rigid body modes not only removes the smallest eigenvalues, but also provides an accurate approximation of the eigenvectors associated with the lowest part of the spectrum, giving rise to a dramatic acceleration. Furthermore, the large CPU gains are not clearly seen on large models. Boersma *et al.* [19] use an algebraic multigrid technique to solve the Navier equations coupled with a PCG in a three-dimensional context. No special procedure is carried out in order for the vectors that almost belong to the kernel of the fine grid to also belong to the kernel of the coarse grid [20]. The coupling with the PCG may certainly provide an added robustness so that special interpolation may not be required. Similarly, Feng *et al.* [21] couple a non-linear geometric multigrid with some Krylov methods as smoothers and outer iterative solvers, without treatment for rigid body modes. At the opposite end of possible techniques, Griebel *et al.* [22] show in an algebraic multigrid context that its interpolation operator represents exactly rigid body modes on a certain class of coarse grids, relying on a block interpolation. Along the same lines, Xiao *et al.* [23] explicitly builds the interpolation of the AMG solver in order for the rigid body modes to be reproduced in a two-dimensional context. Recently, Baker *et al.* [24] improve these techniques further by introducing approximately the rigid body modes through a least-square approach and exactly through additional unknowns on the coarse grid.

To sum up, there are two categories of possibly optimal methods for elasticity. Iterative methods represent the first category and deliver high performance if well preconditioned, which typically involves a non-negligible cost. This may be due to the fact that spectral information is not straightforwardly accessible during a factorization process. Multigrid methods constitute the second category and have found a sound theoretical basis to tackle the problem. However, the algebraic coarsening process and the building of the Galerkin operator are slow, and the multiple geometrical meshes are awkward. Furthermore, it is not obvious that a couple of smoothing iterations on the intermediate grids is worth the effort compared to the highly efficient preconditioned conjugate gradient smoother on the finest grid. There is therefore room for improvement by taking advantage of the best of both approaches. This is exactly what the deflated preconditioned conjugate gradient pretends to achieve.

After this introduction, deflation applied to preconditioned conjugate gradient solvers is presented. First, one particular derivation of the algorithm is presented. Then, the critical construction of the deflation space is discussed. Finally, numerical examples illustrate the behavior of the algorithm in a serial and a parallel context.

2. DEFLATED CONJUGATE GRADIENT

Deflation is an old and common technique in iterative solvers for eigenvalues [25, 26]. In his seminal paper [27], Nicolaides accelerates an iterative solver for symmetric positive-definite matrices, the widely utilized preconditioned conjugate gradient [28], through a deflation technique (see [1] for numerous references). More recently, deflation has been extended to non-symmetric solvers with success in [29, 30].

2.1. Derivation of the algorithm

There are many ways of deriving the deflated conjugate gradient [1, 17, 27, 31]. In this section, we rely on the approach proposed in [31], which may be more amenable to extensions in the non-symmetric case and is fully equivalent to the one proposed in [1, 17] if no additional preconditioner is applied. Given a deflation space $\mathbf{W}$, let us define the projector $\mathbf{P}$ as:

$$\mathbf{P}=\mathbf{I}-\mathbf{AW}(\mathbf{W}^{\mathrm{T}}\mathbf{AW})^{-1}\mathbf{W}^{\mathrm{T}} \tag{4}$$

$\mathbf{P}$ is an $\mathbf{A}^{-1}$-orthogonal projector onto $\mathbf{W}^{\perp}$ along span $\{\mathbf{AW}\}$ as:

$$(\mathbf{PX},\mathbf{AW})_{\mathbf{A}^{-1}}=(\mathbf{PX},\mathbf{W})=(\mathbf{X},\mathbf{P}^{\mathrm{T}}\mathbf{W})=0 \quad \forall \mathbf{X} \tag{5}$$

DOI: 10.1002/nme

Its transpose is

$$\mathbf{P}^{\mathrm{T}} = \mathbf{I} - \mathbf{W}(\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{A} \tag{6}$$

and is an $\mathbf{A}$-orthogonal projector onto $\mathbf{W}^{\perp_{\mathbf{A}}}$ along span $\{\mathbf{W}\}$ as:

$$(\mathbf{P}^{\mathrm{T}}\mathbf{X}, \mathbf{W})_{\mathbf{A}} = (\mathbf{A}\mathbf{P}^{\mathrm{T}}\mathbf{X}, \mathbf{W}) = (\mathbf{X}, \mathbf{P}\mathbf{A}\mathbf{W}) = 0 \quad \forall \mathbf{X} \tag{7}$$

It is easily verified that $\mathbf{P}$ and $\mathbf{P}^{\mathrm{T}}$ are projectors:

$$\mathbf{P}^2 = \mathbf{P} \tag{8}$$

As $\mathbf{A}$ is symmetric, the following relation holds true:

$$\mathbf{A}\mathbf{P}^{\mathrm{T}} = \mathbf{P}\mathbf{A} \tag{9}$$

The solution $\mathbf{x}$ to the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{10}$$

is obtained as [32]

$$\mathbf{x} = \mathbf{P}^{\mathrm{T}}\mathbf{x} + (\mathbf{I} - \mathbf{P}^{\mathrm{T}})\mathbf{x} = \mathbf{P}^{\mathrm{T}}\mathbf{x}_1 + \mathbf{x}_2 \tag{11}$$

with

$$\mathbf{P}^{\mathrm{T}}\mathbf{x}_1 = \mathbf{P}^{\mathrm{T}}\mathbf{x} \tag{12}$$

such that

$$\mathbf{P}\mathbf{A}\mathbf{x}_1 = \mathbf{P}\mathbf{b} \tag{13}$$

and

$$\mathbf{x}_2 = (\mathbf{I} - \mathbf{P}^{\mathrm{T}})\mathbf{x} = \mathbf{W}(\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{b} \tag{14}$$

In detail, $\mathbf{x}_2$ is computed as

$$\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W}\mu = \mathbf{W}^{\mathrm{T}}\mathbf{b} \tag{15}$$

and

$$\mathbf{x}_2 = \mathbf{W}\mu \tag{16}$$

In order to obtain $\mathbf{x}$, $\mathbf{x}_1$ is multiplied by $\mathbf{P}^{\mathrm{T}}$ and is added to $\mathbf{x}_2$ to form the solution:

$$\mathbf{x} = \mathbf{P}^{\mathrm{T}}\mathbf{x}_1 + \mathbf{W}(\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{b} \tag{17}$$

In this form, the deflated preconditioned conjugate gradient looks slightly different than the algorithm presented in [17]. Furthermore, this form may not be of the most optimized one from a computational viewpoint. For example, it is not feasible to build the operator $\mathbf{PA}$ in a three-dimensional context. However, it is fully equivalent in exact arithmetic. After some algebraic

DOI: 10.1002/nme

manipulations, the final algorithm can be rewritten as follows:

Given $\mathbf{x}_{-1}$:

$$\mathbf{r}_{-1} = \mathbf{b} - \mathbf{A}\mathbf{x}_{-1} \tag{18a}$$

$$\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W}\mu = \mathbf{W}^{\mathrm{T}}\mathbf{r}_{-1} \tag{18b}$$

$$\mathbf{x}_0 = \mathbf{x}_{-1} + \mathbf{W}\mu \tag{18c}$$

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 \tag{18d}$$

$$\mathbf{z}_0 = M^{-1}\mathbf{r}_0 \tag{18e}$$

$$\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W}\mu = \mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{z}_0 \tag{18f}$$

$$\mathbf{p}_0 = \mathbf{z}_0 - \mathbf{W}\mu \tag{18g}$$

while(not converged):

$$\alpha_j = (\mathbf{r}_j, \mathbf{z}_j)/(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j) \tag{18h}$$

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j \tag{18i}$$

$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j \tag{18j}$$

$$\mathbf{z}_{j+1} = M^{-1}\mathbf{r}_{j+1} \tag{18k}$$

$$\beta_j = (\mathbf{r}_{j+1}, \mathbf{z}_{j+1})/(\mathbf{r}_j, \mathbf{z}_j) \tag{18l}$$

$$\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W}\mu = \mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{z}_{j+1} \tag{18m}$$

$$\mathbf{p}_{j+1} = \mathbf{z}_{j+1} + \beta_j \mathbf{p}_j - \mathbf{W}\mu \tag{18n}$$

which is exactly the algorithm proposed in [17] in exact arithmetic.

2.2. *Deflation space*

The deflated preconditioned conjugate gradient is at the crossroads of various iterative solvers for large matrices such as multigrid (either geometric or algebraic), domain decomposition and Krylov subspace methods, as all these methods may be interpreted as projection methods [20, 33, 34]. Although the core algorithm is constituted by a Krylov iterative solver, its main aim is to remove from the residual eigenvector components that are difficult to remove by standard iterative solvers. As a matter of fact, given $\mathbf{W}$, the deflated CG algorithm generates a sequence $\mathbf{x}_1$, $\mathbf{x}_i$ such that [17]:

$$\mathbf{x}_i \in \mathbf{x}_0 + \mathbf{K}_i \tag{19}$$

with $\mathbf{K}_i = \mathrm{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{i-1}\mathbf{r}_0, \mathbf{W}\}$ and:

$$\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i \perp \mathbf{K}_i \tag{20}$$

The error is characterized by:

$$\|\mathbf{x} - \mathbf{x}_i\|_{\mathbf{A}} = \min_{u \in \mathbf{x}_0 + \mathbf{K}} \|\mathbf{x} - \mathbf{u}\|_{\mathbf{A}} \tag{21}$$

Finally, convergence is estimated with

$$\|\mathbf{e}^i\|_{\mathbf{A}} \leqslant 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \|\mathbf{e}^0\|_{\mathbf{A}} \tag{22}$$

DOI: 10.1002/nme

where κ is the condition number of the deflated matrix $\mathbf{PAP}^{\mathrm{T}}$ and $\mathbf{e}^i = \mathbf{x} - \mathbf{x}_i$ is the error. Therefore, the convergence speed improves as soon as the condition number improves [35], although clustering of eigenvalues may also be relevant [36], and noting that only the symmetric case is considered here [37]. From a practical viewpoint, modes associated with low eigenvalues should be identified, either analytically or algebraically, and should be well represented in the deflated subspace. Compared to a multigrid approach [38], the deflated subspace plays the role of the prolongation in a two-level coarse grid correction. The use of zero energy modes in a multigrid context is discussed in [39]. In [1], the deflated subspace was constructed by subdomain agglomeration on the mesh to try to represent constant modes in each subdomain, as constant modes belong to the kernel of the continuous operator. Subdomain agglomeration or plain aggregation has been used since the fifties (see [40, p. 68], [41] and the references therein) in economic modeling and was also present in the original paper of Nicolaides [27]. A natural extension to the system of equations given by the static elasticity equations would be to build a constant representation of the displacement for each direction. However, it is well known [16, 22, 23, 42, 43] that apart from translations, which would be accurately represented by constant displacements, rigid body modes, which are in the kernel of the continuous operator, are also constituted by rotational modes. An accurate representation of these modes is therefore mandatory too. Writing the rigid body modes as

$$\mathbf{u} = \mathbf{u}_0 + \mathbf{w} \wedge \mathbf{OM} \tag{23}$$

where $\mathbf{u}_0$ is the translation vector, $\mathbf{w}$ is the rotation vector and $\mathbf{OM} = (x, y, z)$ is the position of point $\mathbf{M}$ with its coordinates, an accurate representation of $\mathbf{W}$, the deflation subspace would be

$$\mathbf{W}_j = \begin{pmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{pmatrix} \tag{24}$$

where point $\mathbf{M}$ belongs to group j. As stated in the introduction, this deflation or augmented space has already been proposed in [16, 24]. Therefore, the small system to be formed after an agglomeration of size `ngroup` has dimension `6*ngroup`, compared to a dimension of `ngroup` for the scalar case. Inclusion of only translation modes gives rise to a small system with dimension `3*ngroup`. However, the results in the numerical section will show that this kind of deflation is not reliable, and that the introduction of the rotational modes in the deflated subspace is well worth the extra effort of duplicating the unknowns in the small system.

3. EXAMPLES

This section illustrates the behavior of the deflated conjugate gradient for various examples. The two first examples show what the deflation can achieve with and without rotations included in the deflated subspace. Then, large-scale examples highlight its behavior in an industrial context. Direct solves are performed by a primitive symmetric skyline direct solver so that gains much be still higher with a state-of-the-art sparse direct solver [44, 45]. For all these examples, a Young's modulus of 2.1×10^{11} and a Poisson ratio of 0.3 were used. Convergence is met if the norm of the residual is less than 10^{-7} of the norm of the right-hand side. All the meshes used in this work are unstructured meshes of tetrahedra. Although the element type may affect the accuracy of the solution, the solver behavior should be very similar with hexahedra. An implementation based on an edge-based data structure [46] has been chosen. For the parallel examples shown below, a renumbering technique for shared memory has been used in order for the same point not to be accessed by edges belonging to different threads [47, 48]. Owing to the large differences obtained in iteration number, all the plots have been performed in log–log scale. The non-deflated results correspond to the arbitrary value 0.1 in the group number. In this section, Kpoints denote 10^3 points, Mpoints denote 10^6 points. The same nomenclature is used for elements.

DOI: 10.1002/nme

Figure 1. Norm of the displacement for the clamped beam.

Figure 2. Subdomains obtained for 10 groups after agglomeration.

3.1. Clamped beam in tension

This example considers a long beam clamped on the one side, with imposed displacements on the other side along the beam axis of 0.3. The solution is given by a linear variation of the displacement along the beam main axis. The beam dimensions are $10 \times 0.1 \times 0.1$. An unstructured mesh of tetrahedra has been generated with a uniform size of 0.02. The final mesh contains 30 Kpoints and 141 Kelems. This academic example has been chosen for its deep graph due to the elongated geometry, and to contrast the behavior of the deflation with and without rotation. The norm of the displacement is displayed in Figure 1. The subdomains obtained for 10 groups are represented in Figure 2.

Comparisons of deflation with and without rotations are shown in Figure 3. As a first remark, it is very surprising that if the iteration number decreases for 10 groups, it increases again for 50 and 100 groups without rotations. This kind of behavior shows a lack of robustness in the solver and was not observed in the scalar case. The explanation of this behavior may be inferred by looking at the subdomains obtained for 100 groups, which are displayed in Figure 4. A zoom around some groups is highlighted in Figure 5. One can clearly see that subdomain boundaries are not perfectly aligned with the plane normal to the main axis of the beam. Therefore, some subdomains may trigger a rotational mode not present in the solution. This low-frequency mode may impede convergence. With the introduction of the rotations, iterations decrease monotonically with the group number, a feature shared with the scalar case. Furthermore, very few groups are necessary to drastically reduce the iteration number. Eigenvalues are not evenly distributed for an elliptic operator, and low eigenvalues are much less dense than high eigenvalues. The deflation technique is therefore in the optimal context as a few groups will already approximate well the worst eigenmodes from a convergence viewpoint. This very important aspect, which provides the robustness of the method, has been recovered due to the introduction of the rotational modes.

3.2. Clamped beam in torsion

The previous geometry is now used with a pure rotation context. The beam is still clamped at one extremity, and a rotation is imposed with a rotation vector along the beam axis. Displacements are displayed in Figure 6. As displacements without rotations are represented by constant modes, the average displacements due to rotation are zero. Constant displacement deflation is therefore completely blind to rotations.

DOI: 10.1002/nme

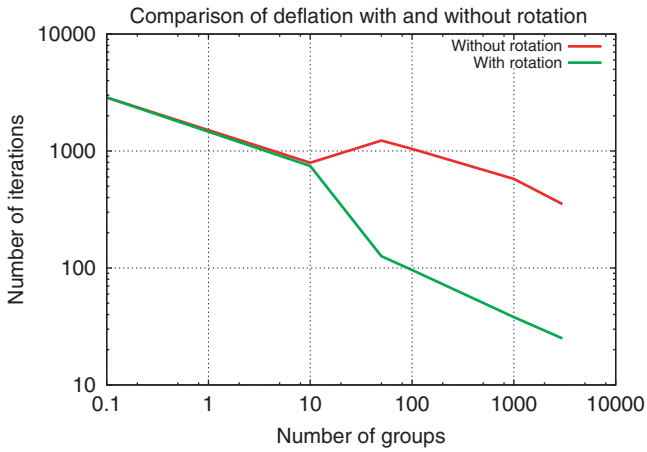

Figure 3. Comparison of deflation with and without rotation for the tension example.

Figure 4. Subdomains obtained for 100 groups after agglomeration.

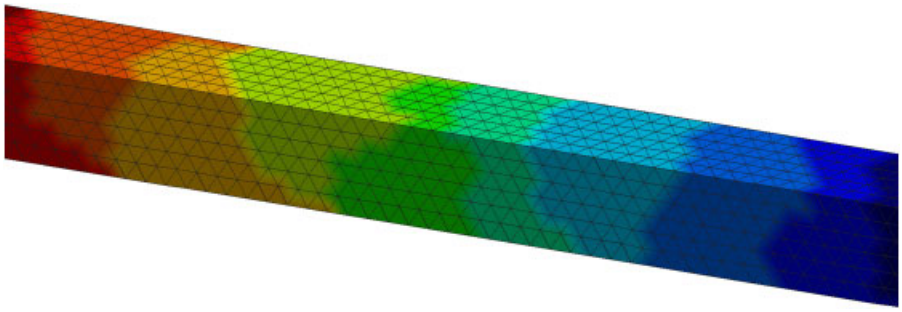

Figure 5. Zoom for 100 subdomains and mesh.

Figure 7 compares the number of iterations for the deflated solver with and without rotations. The difference between both results is dramatic. An unappealing feature of deflation without rotations is the slow decrease in the number of iterations when increasing the numbers of groups. Again, the introduction of the rotations gives rise to a very steep decrease in the iterations, even for a small number of groups. The iterations still decrease but the most important low-frequency modes have already been removed.

3.3. Hook

This example represents a solid hook. The mesh contains 187 Kpoints and 1 Melem. The hook is clamped at the top, and a displacement is imposed at the bottom. The norm of the displacements is shown in Figure 8.

The iteration number obtained versus the subdomain number is displayed in Figure 9 and the CPU time corresponding to this case is depicted in Figure 10. The drastic decrease in iterations when increasing the number of groups is clearly observed. Even with 10 groups, for this example

DOI: 10.1002/nme

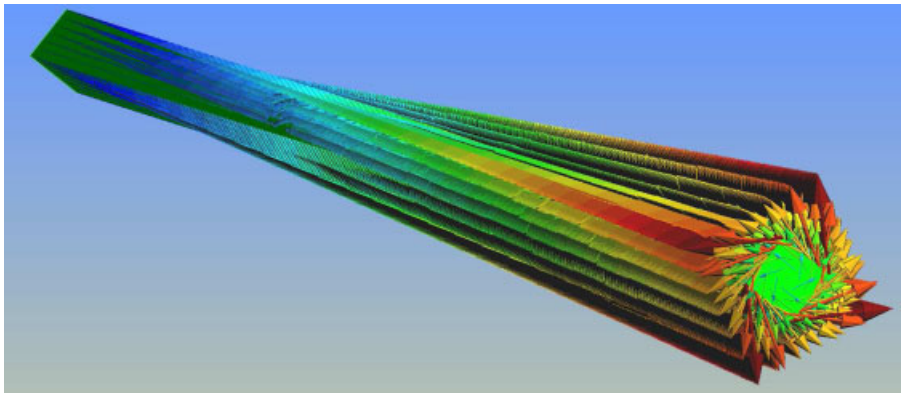

Figure 6. Norm of the displacement for the clamped beam.

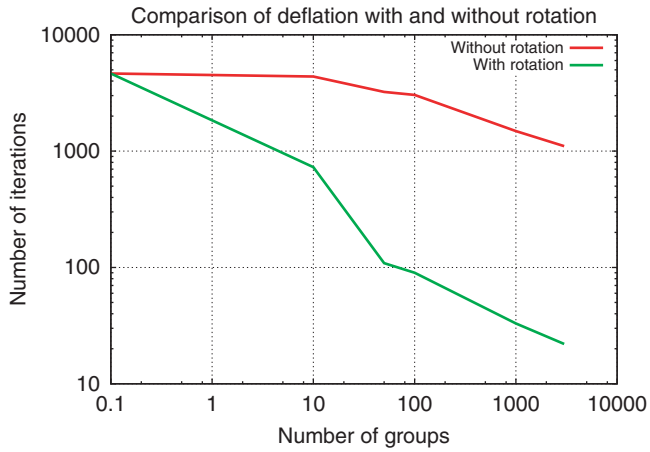

Figure 7. Comparison of deflation with and without rotation for the torsion example.

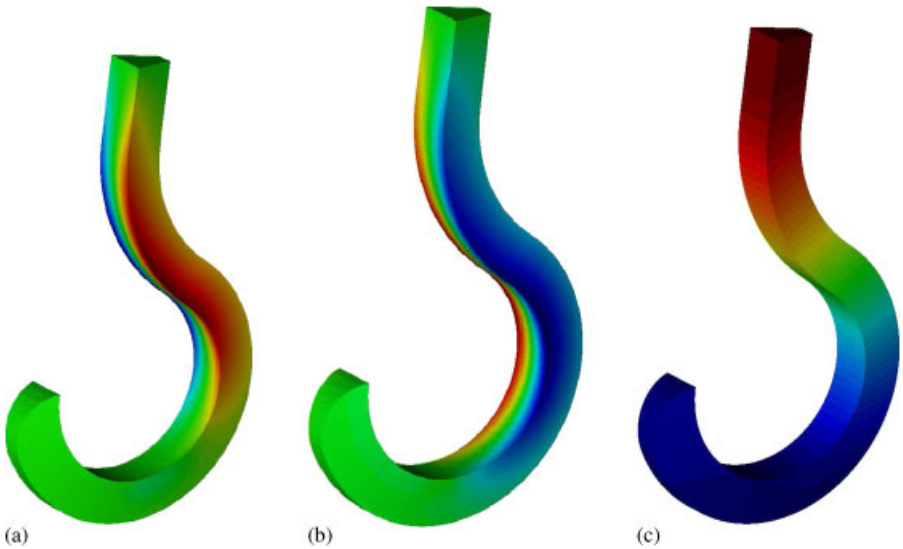

Figure 8. Displacement for the hook example: (a) Ux; (b) Uy; and (c) Uz.

DOI: 10.1002/nme

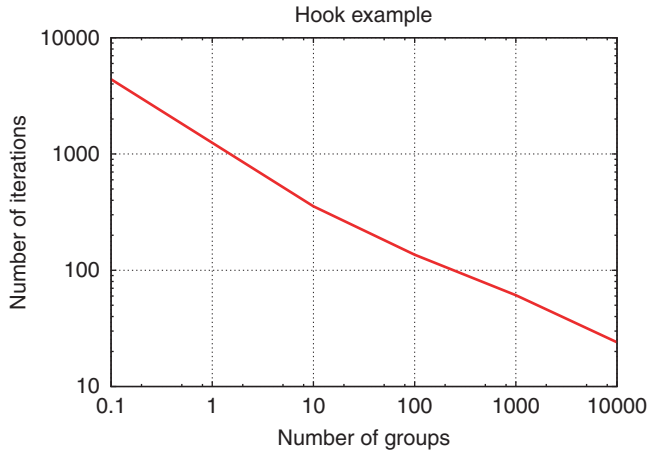

Figure 9. Iterations versus group number.

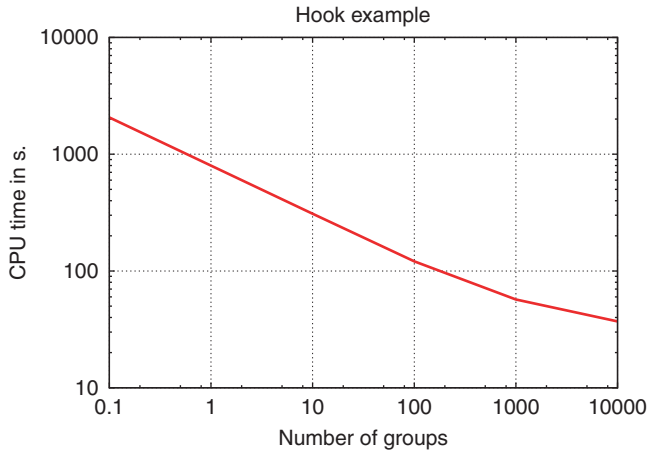

Figure 10. CPU time versus group number.

of 560 Kdof, the iteration number has been reduced by more than an order of magnitude. For 10^4 groups, an almost two orders of magnitude gain in CPU time is achieved compared to the simple PCG. However, 10^4 groups represent a system of already 10% of the whole system. For this small example, this is still affordable in terms of CPU and memory, but can certainly not be maintained for larger examples. Furthermore, although the CPU time is still decreasing, an inflexion appears in Figure 10 due to the CPU cost of the direct solver, which begins to be substantial compared to the iterative solver. This inflexion does not appear in Figure 9 as only iterations are taken into account.

3.4. Shovel

A typical excavator shovel is considered. The mesh contains 91 Melems and 16 Mpoints. The displacement at the top of the shovel is imposed to be zero and the displacement at the handle is imposed to be -0.1 in the vertical direction. The solution obtained is displayed in Figure 11. For this large example, convergence as well as scalability has been tested.

Figure 12 displays the iteration number against the number of groups. It should be noted that the PCG without deflation did not converge in 10^4 iterations and 12 h of CPU. However, to show the orders of magnitude of gains, 10^4 iterations have been reported in the figure for 0 groups. The same behavior as for the previous examples is obtained for this larger case: a strong decrease in the number of iterations with few groups, and a global monotonic decrease with increasing number of

DOI: 10.1002/nme

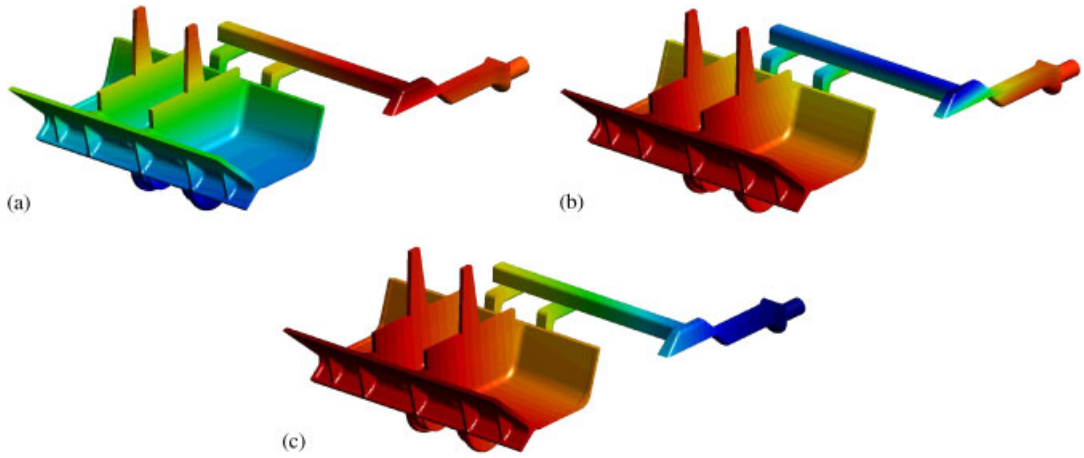

Figure 11. The shovel example: (a) Ux; (b) Uy; and (c) Uz.

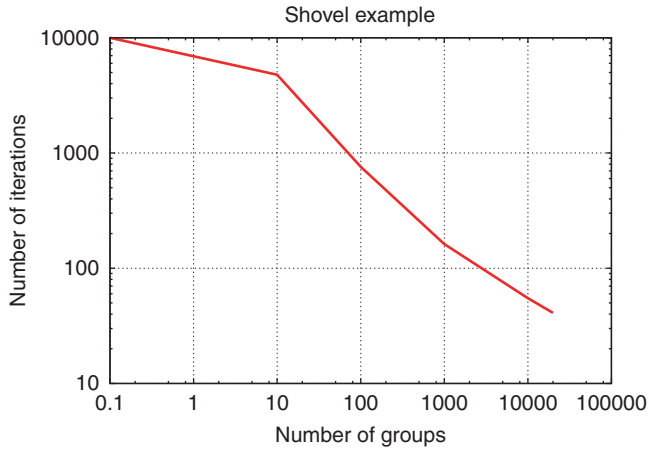

Figure 12. Iterations versus group number for the shovel example.

groups. Note that the storage required for such a small number of groups is negligible compared to the edge-based data structure, which is also negligible compared to the storage needed by a sparse direct solver. For this example, the whole solver required around 14 GB of RAM until 10^4 groups. Then, the storage needs of the direct solver rapidly increase. For 10^6 groups, 42 GB of RAM are necessary for the whole solver, meaning that 28 GB are required for the direct solver. As noted before, there are obviously much more efficient solvers than the simple one used in this work [44, 45]. However, as the problem size grows, the same tendencies will emerge due to the ever increasing fill-in, particularly in three dimensions.

The CPU times are shown in Figure 13. The plot is basically the same than the one reported in [1] for the scalar case. As the number of groups increases, the CPU time decreases faster and faster, until the cost of the direct solver becomes relevant. However, a clear gain of two orders of magnitude appears compared to the non-deflated PCG which did not even converge for that CPU time.

The parallel behavior of the algorithm is studied from a production viewpoint. The group number corresponding to the fastest serial case is considered and the processors are increased up to eight. The run is performed on two Intel quadcores running at 2.6 GHz in shared memory (OpenMP) mode. Although this is a small number of processors, it is typical for commodity PC's and reflects a production environment where wall clock time is relevant. Figure 14 illustrates the CPU time for the building of the deflated matrix and its factorization with the iterative solver time on the

DOI: 10.1002/nme

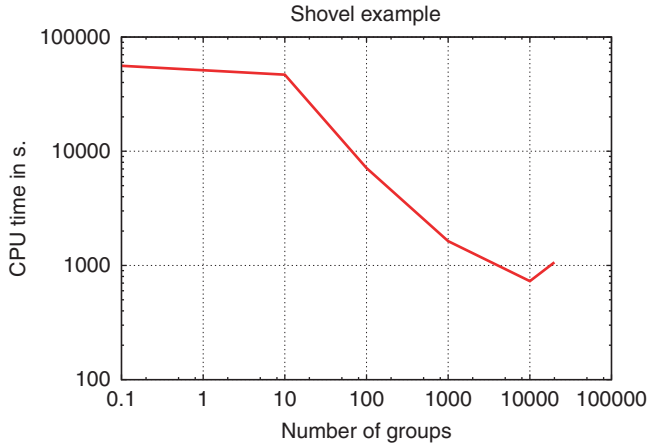

Figure 13. CPU time versus group number for the shovel example.

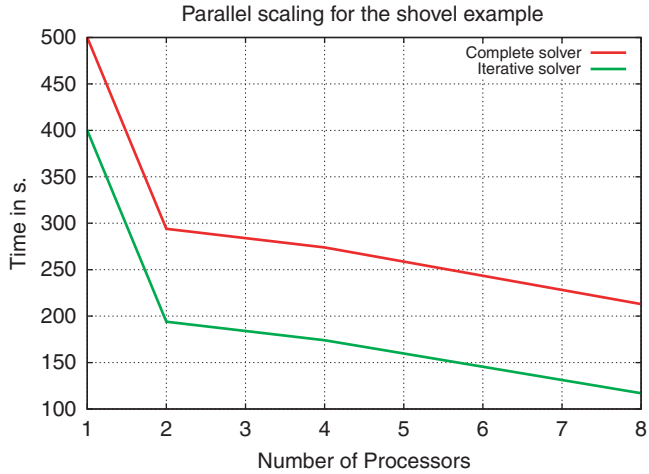

Figure 14. Scaling of the shovel example for eight processors.

one hand, and the iterative solver time on the other. The gap between both curves is constant as it represents the building and factorization time. As the restriction and the triangulation solves are serial, Amdahl's law affects the scalability of the whole solver. However, CPU is still decreasing, achieving a little bit more than 3 min for 45 Mdofs. This is not considered to be an impediment for large-scale parallel computing. It simply illustrates that in a large parallel context it may be more efficient to reduce the size of the direct solve, as the complexity of the direct solver is much higher than the scalability of the pure iterative solver. This claim is furthermore strengthened by the fact that the deflation effect is already noticeable with few groups. Should the subdomain number increase with the processor number, a superlinear effect would be observed.

3.5. Generic cylinder

This example represents a generic hollow cylinder. The mesh contains 147 Melems and 26 Mpoints. The cylinder is clamped at the bottom and a displacement is imposed at the top. Figure 15 displays the displacements obtained in the three directions.

Figure 16 displays the iteration number against the group number, and Figure 17 the CPU time. Compared to the previous example, the same characteristics are observed for this larger problem. The iteration curve shows a monotonic decrease with the number of groups, and the CPU plot

DOI: 10.1002/nme

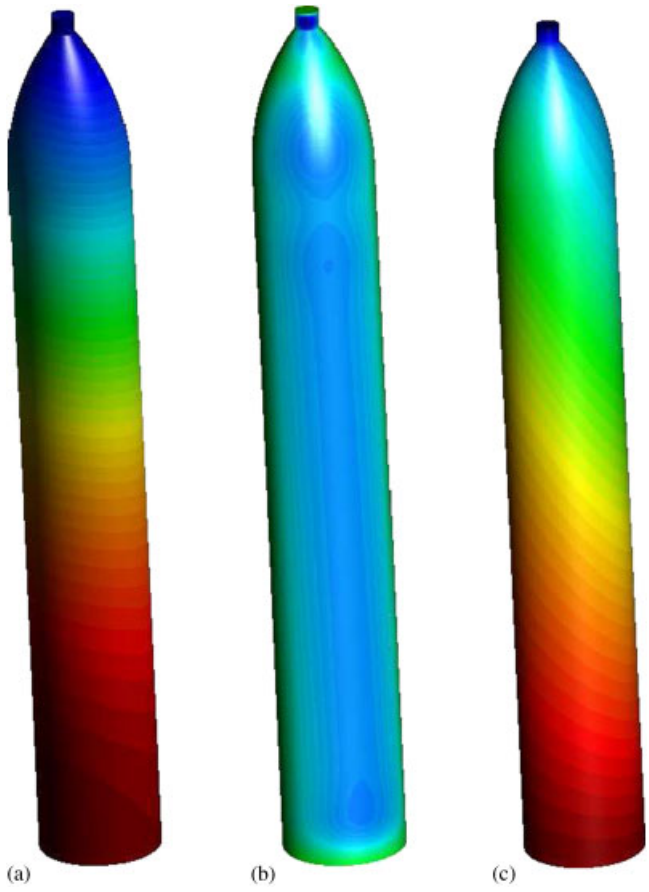

Figure 15. The hollow cylinder example: (a) Ux; (b) Uy; and (c) Uz.

shows an increasingly faster acceleration until the direct solver cost becomes non-negligible. The decrease in the slope of both curves with the group number also reflects what happens in the convergence of the PCG [35]. The natural deflation of the PCG for the rest of the components of the error is stronger as the group number increases. The main difference appears in the point of inflexion of the CPU plot. For the last example, it appears around 10^4 groups while for this example it is close to 3×10^4. As the problem size increases, the iterative solver cost per iteration increases too, so that a decrease in the iteration number is still more beneficial. Therefore, the weight of the direct solver is less relevant and a larger number of groups is more effective on a CPU time basis. Again, a clear gain of at least two orders of magnitude is achieved compared to the non-deflated PCG.

Figure 18 illustrates the CPU time for the building of the deflated matrix and its factorization with the iterative solver time on the one hand, and the iterative solver time on the other. The same general behavior as in the previous example is observed. For eight cores, the final CPU time takes around 7 min for this 78 Mdofs example.

4. CONCLUSION

A deflation technique applied to the static Navier equations of linear elasticity has been presented for the preconditioned conjugate gradient. As highlighted, the system case gives rise to a larger kernel than its scalar counterpart. Therefore, the addition of the rotational modes to the translation modes provides the decisive ingredient for a robust behavior and fast convergence. The deflation

DOI: 10.1002/nme

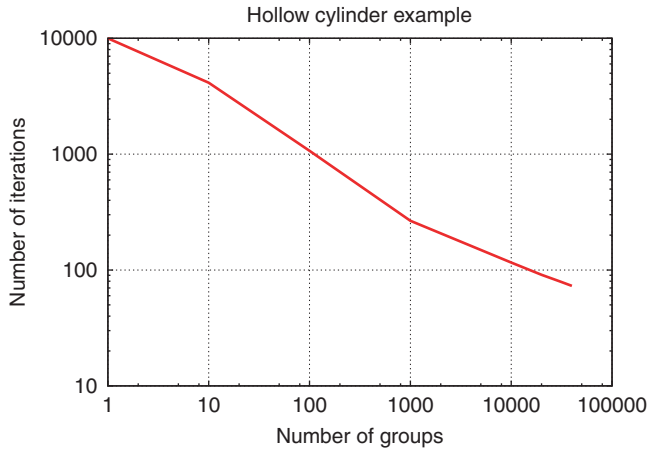

Figure 16. Iterations versus group number for the hollow cylinder example.

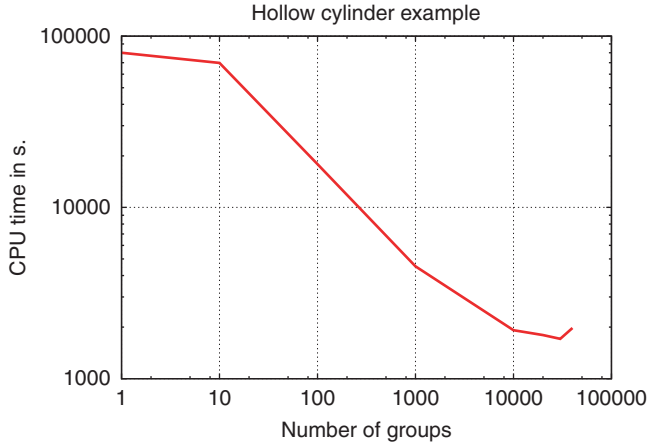

Figure 17. CPU time versus group number for hollow cylinder example.

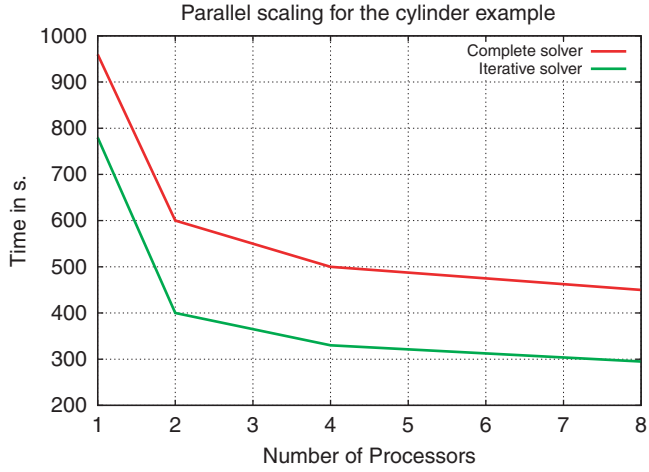

Figure 18. Scaling of the cylinder example for eight processors.

DOI: 10.1002/nme

technique is easy to program, is almost a black box solver as it only needs the coordinates of the vertices, is robust with respect to the number of groups and the increase in the problem size, allows for moving objects and is easily parallelized while providing flexibility for efficiency in a serial and parallel context. From a computational viewpoint, it requires minimal additional storage and CPU overhead. CPU time savings may exceed two orders of magnitude even for grids with relatively moderate graph-depth, which is still more impressive than the scalar case.

Neither anisotropic meshes nor variable material properties have been considered in this work. However, it was shown in [1] that linelet preconditioners could easily and efficiently complement deflation for these meshes, while a subdomain construction was proposed in [3] to take into account variable material properties. Incompressible or quasi-incompressible materials represent another difficulty for linear solvers and deserve a fully different treatment.

Finally, as noted at the beginning, extensions to acoustic and elastic scattering are currently under way with encouraging preliminary results, although it may already be noted that monotonic decrease in the iteration number with an increase in the subdomain number may be lost.

ACKNOWLEDGEMENTS

This work was partly supported by the Office of Naval Research and the High Performance Computer Modernization Program.

REFERENCES

1. Aubry R, Mut F, Löhner R, Cebral JR. Deflated preconditioned conjugate gradient solvers for the pressure-Poisson equation. *Journal of Computational Physics* 2008; **227**(24):10196–10208.
2. Mut F, Aubry R, Löhner R, Cebral JR. Fast numerical solution of patient based blood flow in 3D arterial systems. *International Journal for Numerical Methods in Biomedical Engineering* 2010; **26**(1):73–85.
3. Löhner R, Mut F, Cebral JR, Aubry R, Houzeaux G. Deflated preconditioned conjugate gradient solvers for the pressure-Poisson equation: extensions and improvements. *International Journal for Numerical Methods in Engineering* 2010; DOI: 10.1002/nme.2932.
4. Axelsson O, Gustafsson I. Iterative methods for the solution of the Navier equations of elasticity. *Computer Methods in Applied Mechanics and Engineering* 1978; **15**:241–258.
5. Dickinson JK, Forsyth PA. Preconditioned conjugate gradient methods for three dimensional linear elasticity. *Journal of Computational and Applied Mathematics* 1994; **37**:2211–2234.
6. Saint-Georges P, Warzee G, Beauwens R, Notay Y. High-performance PCG solvers for FEM structural analysis. *International Journal for Numerical Methods in Engineering* 1996; **39**:1313–1340.
7. Kilic SA, Saied F, Sameh A. Efficient iterative solvers for structural dynamics problems. *Computers and Structures* 2004; **82**:2363–2375.
8. Hladik I, Reed MB, Swoboda G. Robust preconditioners for linear elasticity FEM analysis. *International Journal for Numerical Methods in Engineering* 1997; **40**:2109–2127.
9. Farhat C, Roux FX. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering* 1991; **32**:1205–1227.
10. Farhat C, Lesoinne M, LeTallec P, Pierson K, Rixen D. FETI-DP: a dual–primal unified FETI method part I. A faster alternative to the two-level FETI method. *International Journal for Numerical Methods in Engineering* 2001; **50**:1523–1544.
11. Dohrmann CR. A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal on Scientific Computing* 2003; **25**:246–258.
12. Mandel J. Balancing domain decomposition. *Communications in Numerical Methods in Engineering* 1993; **9**:233–241.
13. Dostál Z, Horák D, Kuçera R. Total FETI—an easier implementable variant of the FETI method for numerical solution of elliptic PDE. *Communications in Numerical Methods in Engineering* 2006; **22**:1155–1162.
14. Jouglard CE, Coutinho ALGA. A comparison of iterative multi-level finite element solvers. *Computers and Structures* 1998; **69**(5):655–670.
15. Waltz J. Unstructured Multigrid Methods. *Ph.D. Thesis*, George Mason University, 2000.
16. Bulgakov VE, Kuhn G. High-performance multilevel iterative aggregation solver for large finite-element structural analysis problems. *International Journal for Numerical Methods in Engineering* 1995; **38**:3529–3544.
17. Saad Y, Yeung M, Erhel J, Guyomarc'h F. A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing* 2000; **21**(5):1909–1926.
18. Jönsthövel TB, van Gijzen MB, Vuik C, Kasbergen C, Scarpas A. Preconditioned conjugate gradient method enhanced by deflation of rigid body modes applied to composite materials. *Computer Modeling in Engineering and Sciences* 2009; **47**:97–118.

19. Boersma A, Wriggers P. Algebraic multigrid solver for finite element computations in solid mechanics. *Engineering Computations* 1997; **14**:202–215.
20. Shapira Y. *Matrix-based Multigrid*: *Theory and Applications*. Springer: Berlin, 2008.
21. Feng YT, Peric D, Owen DRJ. A non-nested Galerkin multi-grid method for solving linear and nonlinear solid mechanics problems. *Computer Methods in Applied Mechanics and Engineering* 1997; **144**:307–325.
22. Griebel M, Oeltz D, Schweitzer MA. An algebraic multigrid method for linear elasticity. *SIAM Journal on Scientific Computing* 2003; **25**(2):385–407.
23. Xiao YX, Zhang P, Shu S. An algebraic multigrid method with interpolation reproducing rigid body modes for semi-definite problems in two-dimensional linear elasticity. *Journal of Computational and Applied Mathematics* 2007; **200**(2):637–652.
24. Baker AH, Kolev TzV, Yang UM. Improving algebraic multigrid interpolation operators for linear elasticity problems. *Numerical Linear Algebra with Applications* 2010; **17**:495–517.
25. Wilkinson JH. *The Algebraic Eigenvalue Problem*. Oxford University Press, Inc.: New York, NY, U.S.A., 1988.
26. Parlett B. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics. SIAM: Philadelphia, Philadelphia, 1998.
27. Nicolaides RA. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis* 1987; **24**(2):355–365.
28. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* 1952; **49**:409–436.
29. Erlangga YA, Nabben R. Deflation and balancing preconditioners for Krylov subspace methods applied to nonsymmetric matrices. *SIAM Journal on Matrix Analysis and Applications* 2008; **30**(2):684–699.
30. Erlangga YA, Nabben R. Multilevel projection-based nested Krylov iteration for boundary value problems. *SIAM Journal on Scientific Computing* 2008; **30**(3):1572–1595.
31. Frank J, Vuik C. On the construction of deflation-based preconditioners. *SIAM Journal on Scientific Computing* 2001; **23**:442–462.
32. Vermolen F, Vuik C, Segal A. Deflation in preconditioned conjugate gradient methods for finite element problems. In *Conjugate Gradient and Finite Element Methods*, Křižek M, Neittaanmäki P, Glowinski R, Korotov S (eds). Springer: Berlin, 2004; 103–129.
33. Tang JM, Nabben R, Vuik C, Erlangga YA. Theoretical and numerical comparison of various projection methods derived from deflation, domain decomposition and multigrid methods. *Report 07-04*, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2007.
34. Saad Y. *Iterative Methods for Sparse Linear Systems* (2nd edn). SIAM: Philadelphia, 2003.
35. van der Sluis A, van der Vorst HA. The rate of convergence of conjugate gradients. *Numerische Mathematik* 1986; **48**:543–560.
36. Elman HC, Silvester DJ, Wathen AJ. *Finite Elements and Fast Iterative Solvers*. Oxford University Press: Oxford, 2005.
37. Greenbaum A, Pták V, Strakous Z. Any nonincreasing convergence curve is possible for GMRES. *SIAM Journal on Matrix Analysis and Applications* 1996; **17**(3):465–469.
38. Trottenberg U, Oosterlee CW, Schuller A. *Multigrid*. Elsevier: Amsterdam, 2000.
39. Vanek P, Brezina M, Mandel J. Convergence of algebraic multigrid based on smoothed aggregation. *Computing* 1996; **56**:179–196.
40. Brezina M. Robust Iterative Methods for Unstructured Meshes. *Ph.D. Thesis*, University of Colorado, 1997.
41. Gravemeier V, Gee MW, Wall WA. An algebraic variational multiscale–multigrid method based on plain aggregation for convection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering* 2009; **198**:3821–3835.
42. Vanek P, Mandel J, Brezina M. Two-level algebraic multigrid for the Helmholtz problem. *Tenth International Conference on Domain Decomposition*, *Volume 218 of Contemporary Mathematics*. American Mathematical Society, 1998; 349–356.
43. Farhat C, Li J, Avery P. A FETI-DP method for the parallel iterative solution of indefinite and complex-valued solid and shell vibration problems. *International Journal for Numerical Methods in Engineering* 2005; **63**:398–427.
44. Amestoy PR, Duff IS, Koster J, L'Excellent J-Y. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications* 2001; **23**(1):15–41.
45. Schenk O, Gärtner K, Fichtner W, Stricker A. Pardiso: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems* 2001; **18**(1):69–78.
46. Löhner R. *Applied Computational Fluid Dynamics Techniques*: *An Introduction Based on Finite Element Methods* (2nd edn). Wiley: New York, 2008.
47. Löhner R. Renumbering strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**:95–109.
48. Aubry R, Houzeaux G, Vázquez M, Cela JM. Some useful strategies for unstructured edge-based solvers on shared memory machines. *International Journal for Numerical Methods in Engineering* 2011; **85**(2):537–561.

DOI: 10.1002/nme