A scalable Schwarz method for 3D linear elasticity problems on domains with complex geometry

Fande Kong, Xiao-Chuan Cai

Department of Computer Science, University of Colorado Boulder, Boulder, Colorado 80309-0430, ({fande.kong, cai}@colorado.edu)

Introduction

Several iterative algorithms for solving the elasticity equation discretized by the finite element method are *theoretically* scalable in the sense that the number of iterations does not grow much when the mesh is refined for better accuracy or when the number of processors is increased. However, the theoretically optimal scalability does not translate into linear scalability in *total compute time*, especially when the number of processors is large and the computational domain is complex, because the coarse level solvers are not scalable in terms of the compute time. We introduce a new way to construct coarse level spaces that preserve the geometric feature of the computational domain, but give up accuracy in the interior of the domain. As it turns out the tradeoff in accuracy provides the high scalability in terms of the total compute time. The low accuracy coarse solver does not change the overall accuracy since it is part of the preconditioner. We show numerically that such a new preconditioner is highly scalable for solving linear elasticity equation discretized on unstructured 3D meshes with hundreds of millions of unknowns on a supercomputer with over 10,000 processors.

Linear elasticity equation

The following linear elasticity equation [3] is used to calculate the displacement $\boldsymbol{u}$ of a body Ω in R^3 which is fixed along a portion of its boundary, Γ_d, and is subject to a surface force $\boldsymbol{g}$ along the rest of the boundary, $\Gamma_n = \partial\Omega \backslash \Gamma_d$.

$$\begin{cases} -\mu\Delta\boldsymbol{u} - (\lambda+\mu)\nabla(\nabla\cdot\boldsymbol{u}) = \boldsymbol{f} & in \;\; \Omega \\ \boldsymbol{\sigma n} = \boldsymbol{g} & on \;\; \Gamma_n \\ \boldsymbol{u} = 0 & on \;\; \Gamma_d. \end{cases} \tag{1}$$

Here $\boldsymbol{f}$ is the given body force, $\boldsymbol{\sigma}$ is a stress tensor, $\boldsymbol{n}$ is the outward unit normal to the boundary Γ_n, μ and λ are Lamé coefficients expressed as functions of Young's modulus, E, and Poisson's ratio, ν, by

$$\mu = \frac{E}{2(1+\nu)} \text{ and } \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}.$$

The variational formulation of (1) reads as

$$\begin{cases} \text{Find } \;\; \boldsymbol{u} \in \boldsymbol{H}^1_{\Gamma_d}(\Omega), \;\; \text{such that} \\ a(\boldsymbol{u},\boldsymbol{v}) = f(\boldsymbol{v}), \;\; \forall \boldsymbol{v} \in \boldsymbol{H}^1_{\Gamma_d}(\Omega), \end{cases} \tag{2}$$

where

$$a(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})\boldsymbol{\varepsilon}(\boldsymbol{v}) + \lambda \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{v})\, \mathrm{d}x, \;\; \boldsymbol{\varepsilon}(\boldsymbol{u}) = 1/2(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T),$$

$$f(\boldsymbol{v}) \;\; = \int_{\Gamma_n} \boldsymbol{g}\boldsymbol{v}\, \mathrm{d}S + \int_\Omega \boldsymbol{f}\boldsymbol{v}\, \mathrm{d}x, \text{ and } \boldsymbol{H}^1_{\Gamma_d}(\Omega) = \{\boldsymbol{v} \in H^1(\Omega)^3 : \boldsymbol{v}|_{\Gamma_d} = 0\}.$$

Approximating $\boldsymbol{H}^1_{\Gamma_d}(\Omega)$ by

$$\boldsymbol{V}^h = \{\boldsymbol{v} : \boldsymbol{v} \text{ is continuous on } \Omega, \boldsymbol{v}|_K \text{ is linear on each element}, \boldsymbol{v} = 0 \text{ on } \Gamma_d\},$$

the corresponding discretized version of (2) is: Find $\boldsymbol{u}_h \in \boldsymbol{V}^h$ such that

$$a(\boldsymbol{u}_h, \boldsymbol{\phi}) = f(\boldsymbol{\phi}), \;\; \forall \boldsymbol{\phi} \in \boldsymbol{V}^h,$$

which is a large and often ill-conditioned linear system of equations

$$Ax = b. \tag{3}$$

Three-level restricted Schwarz preconditioner

We solve (3) with an overlapping Schwarz preconditioned iterative method. To save space, we do not discuss the iterative method itself and only focus on the preconditioner since the parallel scalability of the method depends almost completely on how the preconditioner is constructed. For related information, please see [5] and [6].

Through a large number of numerical experiments, we find that the one-level method scales well when the number of processors is small, and the two-level method scales well when the number of processors is in the order of a few thousands. To go beyond 10,000 processors, we need at least a three-level method.

For convenience, three meshes from fine to coarse are denoted by Ω_{h_i}, and the corresponding matrices and vectors are denoted by A_{h_i} and $x_{h_i}, i = 0, 1$ and 2. In order to define an one-level method, we first partition the finite element mesh Ω_{h_i} into np (np is the number of processors) subdomains $\Omega_{h_i,j}$, then we extend each subdomain $\Omega_{h_i,j}$ to overlap with its neighbors by a user-specified amount δ, and we denote the overlapping subdomain as $\Omega^\delta_{h_i,j}$. Let $R^\delta_{h_i,j}$ [2] be a rectangular matrix that returns the vector of coefficients defined in the interior of $\Omega^\delta_{h_i,j}$ that is,

$$x^\delta_{h_i,j} = R^\delta_{h_i,j} x_{h_i} = (I \;\; 0) \begin{pmatrix} x^\delta_{h_i,j} \\ x_{h_i} \backslash x^\delta_{h_i,j} \end{pmatrix}$$

where $x^\delta_{h_i,j}$ is the vector of the coefficients associated with subdomain $\Omega^\delta_{h_i,j}$, and $x_{h_i} \backslash x^\delta_{h_i,j}$ is the vector for the components in $\Omega_{h_i} \backslash \Omega^\delta_{h_i,j}$. Similarly, we define $R^0_{h_i,j}$ as a restriction that returns the vector of coefficients defined in the interior of $\Omega_{h_i,j}$ without any overlap. Thus, the one-level methods can be written as

$$B^{h_i}_{one} = \sum_{j=1}^{np} (R^0_{h_i,j})^T A^{-1}_{h_i,j} R^\delta_{h_i,j}, \;\; A_{h_i,j} = R^\delta_{h_i,j} A_{h_i} (R^\delta_{h_i,j})^T, i = 0, 1, 2, \tag{4}$$

where $A^{-1}_{h_i,j}$ is an approximate subdomain solve. If a coarse space is introduced, the two-level methods can be written as:

$$B^{h_i}_{two} = B^{h_i}_{one} + I^{h_i}_{h_{i+1}} A^{-1}_{h_{i+1}} (I^{h_i}_{h_{i+1}})^T - I^{h_i}_{h_{i+1}} A^{-1}_{h_{i+1}} (I^{h_i}_{h_{i+1}})^T A_{h_i} B^{h_i}_{one}, i = 0, 1, \tag{5}$$

where $I^{h_i}_{h_{i+1}}$ is an interpolation operator from $\Omega_{h_{i+1}}$ to Ω_{h_i}; $(I^{h_i}_{h_{i+1}})^T$ is the corresponding restriction operator, and $A^{-1}_{h_{i+1}}$ represents an approximate solve on $\Omega_{h_{i+1}}$. Similarly, if the approximate solve is replaced by another two-level preconditioner recursively, the three-level method can be written as:

$$B^{h_i}_{three} = B^{h_i}_{one} + I^{h_i}_{h_{i+1}} B^{h_{i+1}}_{two} (I^{h_i}_{h_{i+1}})^T - I^{h_i}_{h_{i+1}} B^{h_{i+1}}_{two} (I^{h_i}_{h_{i+1}})^T A_{h_i} B^{h_i}_{one}, i = 0. \tag{6}$$

Surface geometry preserving coarse space

The three-level preconditioner defined above is able to keep the number of iterations small if the coarse meshes are fine enough, but to keep the total compute time down is tricky. We have to coarsen the coarse meshes to reduce the computational cost. In the case of simple geometry (such as a cube), we can uniformly coarsen the coarse meshes, but for complex geometry (Fig 2), the uniform coarsening changes the geometry of the coarse spaces and will result in a large increase of the number of iterations. In this poster, we propose a boundary geometry preserving coarsening idea. For complex geometry problems, a coarse space with the right geometric features is vital for guaranteeing that the algorithm is scalable for a large number of processors. A boundary geometry preserving coarse mesh is shown in Fig 1.

Re-engineered ParMETIS for large number of processors

ParMETIS is used for the partition of the meshes. When the number of processors is small, ParMETIS is great, but our experience shows that when the number of processors is large the partition is far from ideal. We developed a simple but very effective way to use ParMETIS, that is, we first use ParMETIS to partition the mesh into N_1 (N_1 is the number of compute nodes) subdomains, and then use METIS to further partition each of the subdomains to N_2 (N_2 is the number of the cores on each compute node) smaller subdomains. The resulting partition is significantly better than obtaining a large number of subdomains all at once from a single use of ParMETIS.

Numerical results

We consider a three dimensional lever, which is part of a dental CT machine, as shown in Fig 2, where a bearing load of 5000 (N/mm^2) is applied to the leftmost cylinder, and the second and the last cylinders are all fixed. The Young's modulus E and Poisson ratio ν are assigned as $2.15 \times 10^{11} Pa$ and 0.29 respectively. The algorithms are implemented on top of the PETSc [1] and the numerical experiments are carried on a cluster of IBM servers.

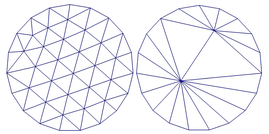

Figure 1: The left figure is the fine mesh, and the right figure is a "boundary" geometry preserving coarse mesh. In 3D, "boundary" is changed to "surface".

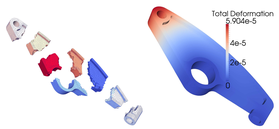

Figure 2: Left: a sample partition with 8 subdomains; right: numerical solution.

The strong scalability for up to 10240 processor cores is listed in Table 1. It can be seen clearly that the proposed approach is scalable in terms of the number of iterations (it is nearly a constant for all processor counts), and the total compute time. The compute time and speedup are also shown in Fig 3. To further understand the scalability of the algorithm, we plot the compute time spent on different levels and on the interpolation/restriction in Fig 3. Level 2 is the coarsest level, level 1 is the second coarse level, and the level 0 is the fine level. The compute time on level 1 is not scalable, but it counts for only a few percentage of the total time.

Table 1: Strong scalability results. The problem has 260,998,902 degrees of freedom and is solved by FGMRES with the three-level restricted Schwarz preconditioner. np is the number of processors and $iter$ is the number of FGMRES iterations.

np	iter	time	speedup	ideal speedup	efficiency
4096	51	44.1	1	1	100%
6144	51	29.1	1.5	1.5	100%
8192	51	23.6	1.9	2	95%
10240	49	18.4	2.4	2.5	96%

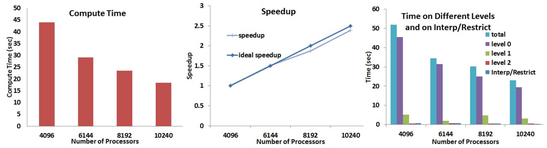

Figure 3: Left: total compute time; middle: speedup; right: time spent on different levels and on the interpolation/restriction.

Some final remarks

A highly scalable three-level Schwarz method with geometry preserving coarse spaces was introduced and studied for solving 3D linear elasticity equation on an unstructured mesh. Coarse meshes were carefully constructed so that they preserve boundary geometry which is very important for guaranteeing that the algorithm is scalable for a large number of processors. A re-engineered ParMETIS was developed to partition meshes into a large number of subdomains. Numerical experiments show that the algorithm developed in the poster is highly scalable for linear systems with more than 260 millions unknowns, and on a machine with over 10,000 processors.

References

[1] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. F. Smith and H. Zhang, PETSc Users Manual, Argonne National Laboratory, 2013.

[2] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput., 21 (1999), pp. 792-797.

[3] P. Howell, G. Kozyreff and J. Ockendon, Applied Solid Mechanics, Cambridge University Press, Cambridge, 2009.

[4] G. Karypis, K. Schloegel, ParMETIS - Parallel Graph Partitioning and Sparse Matrix Ordering Library Version 4.0, University of Minnesota, 2013.

[5] B. F. Smith, P. E. Bjørstad and W. D. Gropp, Domain Decomposition Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, Cambridge, 1996.

[6] A. Toselli, O. Widlund, Domain Decomposition Methods - Algorithms and Theory, Springer-Verlag, Berlin, 2005.