Performance Analysis of SA-AMG Method by Setting Extracted Near-kernel Vectors

Naoya Nomura[1], Akihiro Fujii[1], Teruo Tanaka[1], Kengo Nakajima[2], and Osni Marques[3]

[1] Kogakuin University, Tokyo, Japan
em15016@ns.kogakuin.ac.jp
[2] The University of Tokyo, Tokyo, Japan
[3] Lawrence Berkeley National Laboratory

Abstract. The smoothed aggregation algebraic multigrid (SA-AMG) method is among the fastest solvers for large-scale linear equations. The SA-AMG method achieves good convergence by generating small-sized matrices from the original matrix problem. However, the convergence of the method can be further improved by setting near-kernel vectors. Our research investigates the effectiveness of setting multiple near-kernel vectors and finds the important near-kernel vectors for fast convergence. Our method is applied to the 3-dimensional elastic problem. The known near-kernel vectors in this problem (the parallel translation and rotation vectors) improve the convergence and execution time of the SA-AMG method. In the present study, we extract multiple near-kernel vectors by an iterative process known as the V-cycle. In numerical experiments, suitable choice of the near-kernel vectors reduced the iteration number by nearly two-thirds and halved the execution time, relative to the known near-kernel vectors.

Keywords: Linear solver · Algebraic multigrid method · Near-kernel vectors · Performance evaluation

1 Introduction

Iterative solutions to large-scale linear equations $\boldsymbol{Ax} = \boldsymbol{b}$ are often required in scientific computing. Among the fastest solvers for these equations is the algebraic multigrid (AMG) method [1], a multi-level method that builds smaller matrices from the matrix problem. A variant called smoothed aggregation AMG (SA-AMG) [2][3][4] is effective for solving various problems and is widely used.

The SA-AMG method comprises a setup part and a solve part. The setup part creates a graph structure based on the matrix problem and defines a coarse problem based on aggregates of unknowns. Recursive application of this process generates multiple small-sized matrices. The solve part repeatedly applies relaxation (e.g., the Jacobi method) to the hierarchical matrices constructed in the setup part, thereby achieving rapid convergence. The structure is hierarchical, with fine levels (large matrices) and coarser levels (small matrices). The finest level corresponds to the original matrix problem.

The SA-AMG method can incorporate error components which are difficult to correct by ordinary relaxation methods. These error components typically correspond to the near-kernel vectors, defined as non-zero vectors $\boldsymbol{x}$ satisfying $A\boldsymbol{x} \approx 0$. The SA-AMG method sets these error components, and efficiently corrects them by moving them to coarser levels.

In this research, we focus on the 3-dimensional elastic problem, in which the near-kernel vectors are the parallel translation and rotation vectors. Setting these near-kernel vectors improves the convergence of the SA-AMG method in the target problem. To our knowledge, methods for extracting near-kernel vectors and their efficiency have been rarely reported other than αSA [5], which is described in section 4. Thus, this study reports a simple extraction method for multiple near-kernel vectors using V-cycles, and numerically evaluates its performance.

2 SA-AMG method

This section describes the SA-AMG method, which creates and solves hierarchical matrices from the matrix problem.

The SA-AMG method is among the fastest solvers for large-scale linear equations. It creates multiple small sized matrices from the matrix problem, and uses them to solve the matrix problem.

Fig. 1 shows the solve part of the SA-AMG method, which performs matrix vector multiplication and relaxation. The top level is the original matrix problem, and progressively coarser levels are represented by progressively smaller matrices. The number of levels depends on the size of the matrix problem. Data are moved between levels by the inter-level *prolongation* and *restriction* matrices. First, the relaxation method is applied to the finest problem. Next, the residual vector is calculated and multiplied by the *restriction* matrix, which moves it to a coarser level. At the coarser level, the coarsened vector is set as the right-hand-side vector. After applying relaxation at the coarser level, the corrected vector is solved and multiplied by the *prolongation* matrix, which moves it to a finer level. The solution is added to the solution vector at the finer level. Finally, relaxation is repeated at the finer level. The solve part is called the V-cycle.

There are many types of AMG methods with different ways of creating the inter-level matrices [6]. In this research, we use the SA-AMG method, which creates coarser level matrices from a graph structure based on the matrix problem. The problem unknowns and non-zero elements correspond to nodes and edges, respectively. The SA-AMG method then aggregates the unknowns, and assigns each aggregate to a node at the coarser level. Each node at the finer level belongs to a corresponding aggregate. For interpolation, the SA-AMG method sets weight values on each of the aggregate nodes. The weights are stored in an inter-level matrix. The coarser level matrix is calculated as a 3-matrix product RAP, where R and P are *restriction* and *prolongation* matrices respectively. To improve the convergence of SA-AMG, the inter-level matrix can be constructed from the near-kernel vector information.

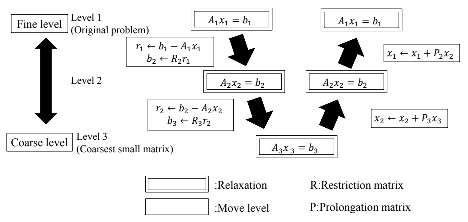

Fig. 1. V-cycle of SA-AMG

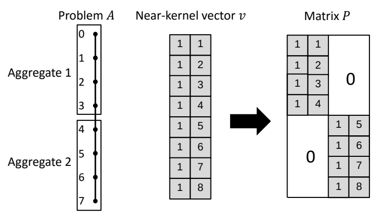

Fig. 2. Construction of the *prolongation* matrix from the near-kernel vectors

3 Near-kernel vector

In this section, we explain the near-kernel vector and its setting in the SA-AMG method.

The near-kernel vector is a vector $\boldsymbol{x}$ that satisfies $A\boldsymbol{x} \approx 0(\boldsymbol{x} \neq 0)$. The equation $A\boldsymbol{x} = \boldsymbol{b}$ is solved by a regular iterative solver, and the solution vector is updated using the vector $\boldsymbol{b}$. However, the error components of the near-kernel vectors cannot be corrected by this vector. Consequently, the convergence stagnates. For moving the near-kernel vector components to coarser levels, the SA-AMG method calculates inter-level operators from the near-kernel vectors. Thus, these components are efficiently corrected at coarser levels, and rapid convergence is achieved [2][3][4]. In some cases, the near-kernel vectors can be specified from the problem settings. For example, the near-kernel vectors of the elastic problem are the translation and rotation vectors.

Fig. 2 illustrates the setting of the near-kernel vectors in the inter-level matrix. In this figure, the *prolongation* matrix is constructed from two near-kernel vectors and two aggregates. In particular, the *prolongation* matrix is created by selecting the corresponding elements from the near-kernel vectors for each aggregate. In the SA-AMG method, the number of columns in the *prolongation* matrix increases proportionally to the number of near-kernel vectors, increasing the calculation costs. Therefore, this method has a trade-off between the number of near-kernel vectors and the execution time.

Fig. 3 shows how the number of iterations for convergence depends on the number of near-kernel vectors in the elastic problem. The left panel illustrates the target problem of this experiment. The matrix structure of the elastic problem is described in section 5.1. The object in this problem comprises a soft upper part and hard lower part. A force is applied over a small area on the upper side. The Young's modulus ratio is 1:0.8, and the Poisson's ratio is 0.3:0.3. The right panel of Fig. 3 graphs shows the number of iterations to convergence. The legend "Number of kernel vector" denotes the designated number of near-kernel vectors. "Number of kernel vectors:1" uses a constant vector, "Number of kernel vectors:3" uses only translation vectors, "Number of kernel vectors:6" uses both translation and rotation vectors. This experiment is a weak scaling test

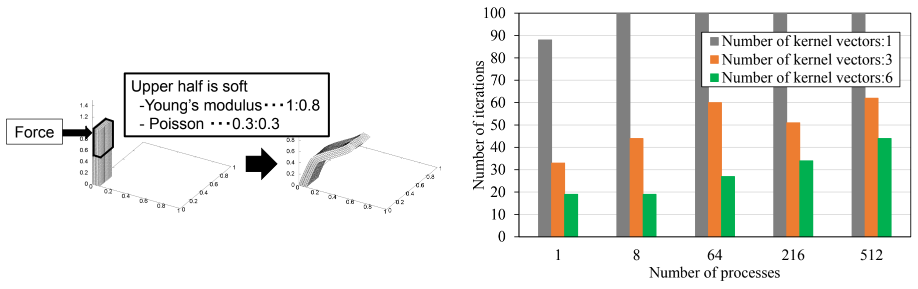

Fig. 3. Effect of multiple near-kernel vectors on the number of iteration in the SA-AMG method

(with a local domain size per process of 6×15×60). The problem domain size increases proportionally to the number of processes. We observe that setting more near-kernel vectors reduces the number of iterations because the near-kernel vectors are known from the problem settings. In the next section, we consider the extraction of additional near-kernel vectors that are not known from the problem settings.

4 Near-kernel vector extraction

The section describes the method for near-kernel vector extraction. There is a related work which proposes near-kernel vector extraction using V-cycles [5]. The authors of [5] calculate the coarser level near-kernel vectors at first, then interpolate them to the finest level and use them as near-kernel vectors. On the other hand, we use V-cycle iterations to calculate near-kernel vectors directly. In this paper, the near-kernel vectors are extracted by the following procedure:

1. Initialize the vector $\boldsymbol{x}$ by assigning random numbers.
2. Iterate the V-cycle μ times to solve $A\boldsymbol{x} = 0$
3. Set the solution vector $\boldsymbol{x}$ in Step 2 as an additional near-kernel vector. If the number of near-kernel vectors is insufficient, return to Step 1.
4. Output the extracted near-kernel vectors.

After solving $A\boldsymbol{x} = 0$ through V-cycle iterations, we are left with the near-kernel error component, which cannot be solved by the V-cycle. The V-cycle iterations are executed on near-kernel vectors extracted in previous iteration steps. Thus, the new near-kernel vector is considered to differ from the previously found near-kernel vectors. It must be noted that V-cycle needs near-kernel vector setting from the beginning. At first, those vectors must be found through the problem settings. In subsequent processes, the method extracts independent near-kernel vectors. In Step 2, the parameter μ is provided as an input. This parameter specifies the number of V-cycle iterations and largely determines the performance of SA-AMG. The optimization of μ will be considered in future works. In the present study, we set μ to 20.

5 Experimental results

5.1 Experimental environments and problem setting

Experiments were performed on an FX10 supercomputer system (Oakleaf-FX) [7] at the University of Tokyo. Each node of the FX10 is equipped with one SPARC64 IXfx processor (1.848GHz, 16 cores) and 32 GB memory and is connected through the 6D mesh/tour network (5GB/s/link, bidirectional). We launched one process per core (Flat MPI model), employing up to 512 cores.

The experimental subject was the 3-dimensional elastic problem, in which an elastic object is pressed under a constant force, and the displacements of all parts of the object are to be determined. The problem was constructed as a 3×3 block matrix at each node. This problem solves linear elasticity problems in simple cube geometries of media with heterogeneous material properties using FEM. The object consisted of an inner hard cube and an outer soft cube, and a force was applied to a small area on the upper face (see Fig. 4). The Young's modulus ratio (indicating the stiffness of the materials) was 5:0.5, where the high and low values correspond to hard and soft materials, respectively. And Poisson's ratio is 0.3:0.3. Moreover, the experiment was performed as a weak scaling (with a local domain per process of 15×15×15). The problem domain was divided into several subdomains with equal intervals on each axis.

The experiment was implemented in the AMGS library [8], which solves large-scale linear equations by the AMG method. The solve part is executed by the GPBiCG method [9], and one iteration of V-cycle is used as a preconditioner. The relaxation procedure of the solve part was performed twice per level by the Gauss-Seidel method, ignoring the data dependency beyond the border of the processing domain. The coarsest level was determined as less than 100 unknown blocks. That is, there are less than 100 × k unknowns at the coarsest level, when it is set with k near-kernel vectors. The termination criterion for the 2-norm of the relative residuals was set to 1.0×10^{-7}. The maximum number of iterations was set to 500.

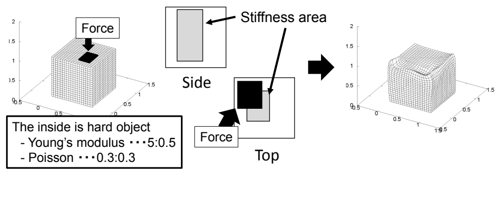

Fig. 4. Setup of the experimental elastic problem

5.2 Experimental results

In this experiment, we investigated the performance of the SA-AMG method by changing the number of near-kernel vectors. The near-kernel vector setups are

Table 1. A target of comparison

near-kernel vectors	Details
3 provided	Parallel translation in each axis direction (X,Y,Z)
6 provided	Parallel translation + rotation on each axis (X,Y,Z)
3+1, 3+2, ...	Parallel translation + extracted near-kernel vectors (up to 7)

compared in Table 1. The near-kernel vectors in the extraction process were set as the parallel translation at first.

Fig. 5 shows the results. The five graphs correspond to results with 1, 8, 64, 216 and 512 processes, respectively. The X axis corresponds to different settings for the near-kernel vectors. The bar indicates the time, while the line indicates the number of iterations for convergence. The bar's element "Setup" and "Solve" report the execution time of the setup part and solve part, respectively. If the number of iteration exceeds 500, the execution time is not plotted. In this experiment, we disregarded the time of extracting the near-kernel vectors (In single process, the time to extract 7 vectors was 17 seconds). As shown in Fig. 5, the number of iterations and execution time were lower for the best extracted near-kernel vectors than when 3 and 6 near-kernel vectors were provided. However, setting many near-kernel vectors does not always improve the convergence. For example, when running 512 processes, the 3+7 near-kernel vector setting failed to converge. In 64 processes, the lowest iteration number was achieved for 6 near-kernel vectors, whereas 3+1 near-kernel vectors achieved the lowest execution time. This result can be explained by the larger processing time for 6 than for 3+1 provided near-kernels.

Fig. 6 plots the best results of Fig. 5. The left and right panels plots the number of iterations and the execution time, respectively. This figure shows that by appropriately determining the near-kernel vectors, we can dramatically reduce the number of iterations. Even in the largest problem (512 processes), the best of the extracted vectors approximately halved the number of iterations and reduced the execution time by approximately 40%, relative to the case of 6 provided near-kernel vectors.

6 Conclusion

In this paper, we investigated whether extracting the multiple near-kernel vectors by V-cycle iterations improved the performance of the SA-AMG method. Evaluations were performed on the 3-dimensional elastic problem. The results of multiple extracted near-kernel vectors were compared with those of the known near-kernel vectors in the elastic problem (parallel translation and rotation vectors). In the largest problem (512 processes), setting the extracted near-kernel vectors halved the iteration number and decreased the solving time by approximately 40%, relative to setting the ordinary translation and rotation vectors.

Thus, the ordinary V-cycle iterations turned out to extract effective near-kernel vectors for convergence. Another problem is the scalability for cases with many nodes. Fig 6 shows that both the number of iterations and the computation time increase, as the problem size increases in weak scaling computation. This is mainly because of localized block-Jacobi-type Gauss-Seidel smoothers in SA-AMG procedure.The increasing in the number of iterations is very significant for ill-conditioned problems, while this effect is not so large in Fig 3 where condition number is close to 1. Stabilization of the localized smoother is a very critical issue. Some remedies described in [10], such as extension of overlapped zones, will be introduced in the future.

Let us conclude with remarks on future directions. First, setting many near-kernel vectors did not always improve the performance; in particular, the method sometimes extracted inappropriate near-kernel vectors that failed to converge. Therefore we must investigate the extracted near-kernel vectors. The relationships between the residual history of the near-kernel vector extraction and the effectiveness of the near-kernel vectors will also be examined in future study.

References

1. Pereira, F. H., Verardi, S. L. L., and Nabeta, S. I.: A fast algebraic multigrid preconditioned conjugate gradient solver, Applied Mathematics and Computation 179, pp.344-351 (2006)
2. Vanek, P., Brezina, M. and Mandel, J.: Convergence of Algebraic Multigrid Based on Smoothed Aggregation, Numerische Mathematik, vol 88, pp.559-579 (2001)
3. Vanek, P., Mandel, J. and Brezina, M.: Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems, Computing, Vol.56, pp.179-196 (1998)
4. Chan, T. F. and Vanek, P.: Multilevel algebraic Elliptic Solvers, UCLA Math, Dept. CAM Report (1999)
5. Brezina, M., Falgout, R., Maclachlan, S., Manteuffel, T., Mccormick, S. and Ruge, J.: Adaptive Smoothed Aggregation (αSA), SIAM J. Sci. Comput, Vol. 25, No.6, pp.1896-1920 (2004)
6. Fujii, A. and Oyanagi, Y.: Evaluation of Algebraic Multi-grid Method: An Efficient Linear Solver for Scientific Simulations, Simulations, 28(4), 149-154, 2009-12-15, pp.9-14 (2009)
7. Information Technology Center: The University of Tokyo, http://www.cc.u-tokyo.ac.jp/
8. AMGS Library: http://hpcl.info.kogakuin.ac.jp/lab/software/amgs
9. Zhang, S.-L.: GPBi-CG: Generalized Product-type Methods Based on Bi-CG for Solving Nonsymmetric Linear Systems SIAM J. Sci. Comput, Vol. 18, No.2, pp.537-551 (1997)
10. Nakajima, K.: Strategies for Preconditioning Methods of Parallel Iterative Solvers in Finite-Element Applications on Geophysics, Advances in Geocomputing, Lecture Notes in Earth Science 119, pp.65-118 (2009)

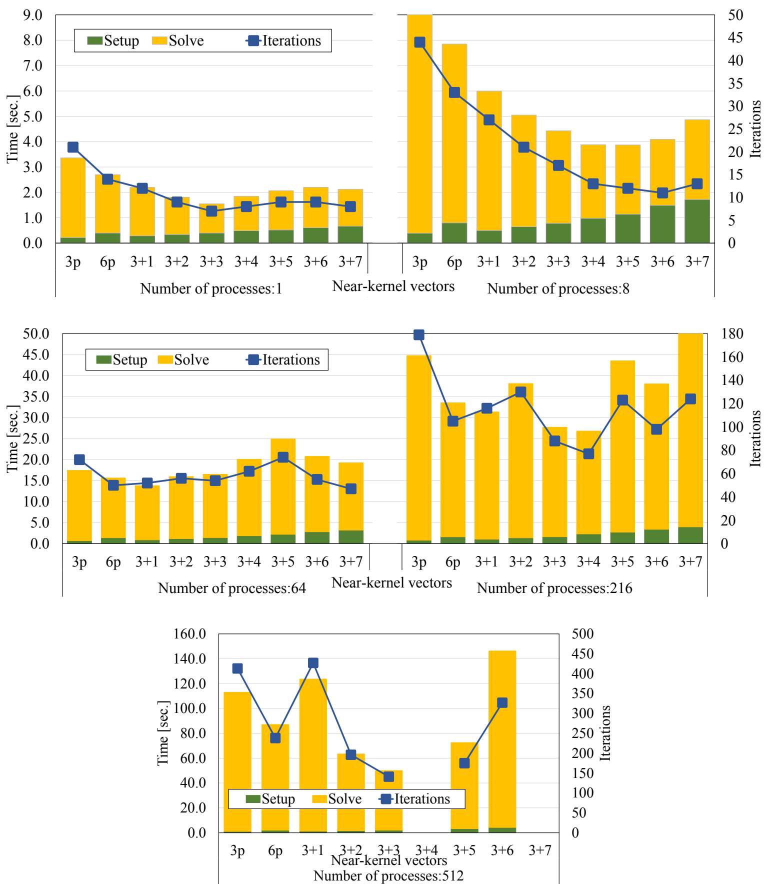

Fig. 5. Experimental results of various settings of near-kernel vectors

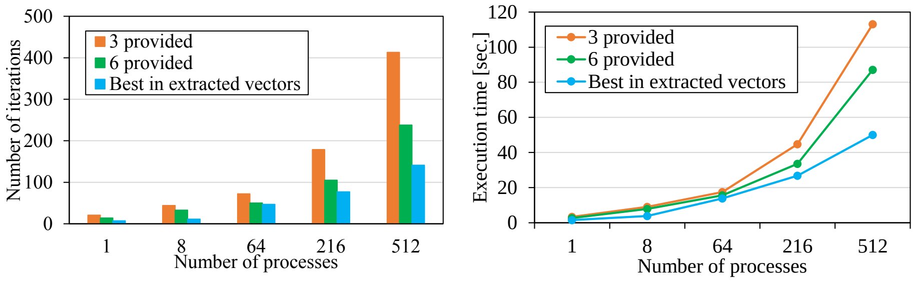

Fig. 6. Number of iterations(left) and execution time (right) for various near-kernel vectors (3 provided, 6 provided, and the best number of extracted near-kernel vectors)