Adaptive preconditioning for a stream of SLAEs

A P Karpov, V A Erzunov, E B Shchanikova and Yu G Bartenev
FSUE “RFNC-VNIIEF”, Mira ave., 37, Sarov, 607188, Russia

E-mail: apkarpov@vniief.ru

Abstract. The paper considers the way of reducing the time consumed to solve SLAEs with iterative methods by reusing the data structures obtained in the solution of a previous SLAE, or selecting a preconditioner from the available set of preconditioners to minimize the time of solving the next SLAEs. Such adaptive preconditioning is used to solve time-dependent nonlinear problems. SLAEs generated at the Newton iteration n-1 of every computation step are solved using the SLAE structure of the first Newton iteration and the selection of a preconditioner from the given set allows reducing the time of solving SLAEs of a varying complexity at different time steps. The adaptive preconditioning idea and its application are demonstrated for a stream of SLAEs in some RFNC-VNIIEF’s codes.

1. Introduction

In contrast to the use of an adaptive solver for a single System of Linear Algebraic Equations (SLAE) specifying its parameters in the course of iteratively solving the system [1] to reduce the problem runtime, the paper considers a similar task for a stream of SLAEs. The task is resolved by reusing the data structures accumulated in solving some previous SLAE and selecting the most appropriate one from the available set of preconditioners. Data for a SLAE with a permanent matrix pattern may include the earlier done optimizing permutations, the structure of preconditioning matrices, a preconditioner, the construction and solution time, and the number of iterations with the earlier used preconditioners.

The adaptive preconditioning of such a type has been developed in the LParSol set of libraries [2], [3] and published in [4]. It is used to solve nonlinear problems in parallel with a number of codes. In one step of computations, Newton’s iterations usually generating a stream of SLAEs with the same matrix pattern are performed. So, the selection of the most suitable one among the given number of preconditioners and the repeated use of the data structures of an iterative solver (CG, BiCGStab) are performed in one step according to formula (1) for 1 ≤ i ≤ n:

\[
\min_i T(i) = \min_i \{ N_{it}(i) \times T_{it1}(i) + T_{pf}(i) \times NSLAEs + (T_{pf}(i) - T_{pl}(i)) \},
\]

where:
- n is the number of preconditioners ordered to solve SLAEs of a given problem.
- NSLAEs is the number of SLAEs (Newton’s iterations) solved in the previous step.
- \( N_{it}(i) \) is the expected number of iterations of the solver used to solve all SLAEs in the step with Preconditioner(i).
- \( T_{it1}(i) \) is the expected time of one iteration of the SLAE solver with Preconditioner(i).
- \( T_{pf}(i) \) is the expected time of the full construction of Preconditioner(i) in a Newton’s iteration.
- $T_{pl}$ is the expected time of the ‘light’ (partial) construction of Preconditioner(i) in the rest Newton iterations.

The expected values are based on tests, i.e. on periodically solving some SLAEs with each of the selected preconditioners during the problem runtime.

The preconditioner selection according to formula (1) without the last term can be further used until the matrix pattern is changed, or the SLAE solution time differs too much from the maximum SLAE solution time in one of the previous steps. Testing of preconditioners which induce a relatively long SLAE solution time, or unresolved SLAE is blocked, or rarely performed. If several physical processes are simulated concurrently, several streams of SLAEs are generated and its own set of preconditioners is admitted for each stream to select one of them that speeds up the solution more efficiently.

The adaptive preconditioning technique for a stream of SLAEs allowed reducing the SLAE solution time and preliminary experiments with a variety of preconditioners in a number of software packages used at FSUE “RFNC-VNIIEF” allowed identifying a set of the fastest SLAE solution methods (among those available in LParSol) in various physical process simulation parts.

2. **Application conditions and experiments**

One of the conditions for this technique application is a rare strong change of the SLAE matrix conditioning of the first Newton iteration (which is usually the most hard to perform) in a time step in the total stream of SLAEs such as those, for example, in one of the RFNC-VNIIEF codes.

![Figure 1. The time spent for SLAE solution in time steps (from paper [4]).](image)

In solving this problem, the list of tested preconditioners of BiCGStab solver included AMG [2], the additive Schwarz method with in-domain ILU$_0$-factorization and the method of matching in overlap according to I E Kaporin and I N Konshin [5], and the block Jacobi method with in-domain ILU$_0$. Each of these methods showed itself as the best one in certain stages of the simulation process. In this particular case, AMG was sufficient to solve all SLAEs in the problem, with the time spent for solution being almost equal to the time of solving SLAE with the use of all the three preconditioners.
above. Here, one should note the effectiveness of adaptively using even the selective-type AMG preconditioner alone [2] which consists in partially constructing AMG to solve most SLAEs. The method consists in reusing restriction matrices, prolongation matrices, and coarse matrices of the previous SLAE and is reduced to the replacement of coefficients of coarse matrices in the course of operations:

\[ A_c = R_c \times A_f \times P_c , \]  

(2)

where \( R_c \) and \( P_c \) are the restriction and prolongation matrices in the previous step, or in the current step. Paper [3] demonstrates the usefulness of such ‘light’ construction for the aggregative-type AgAMG preconditioner [2] used to solve a stream of SLAEs generated by the LOGOS Aero-Hydro software package [6] in solving aerodynamics problems on a hybrid node of the OOO “CKO” cluster [7] that contains two eight-core Intel Xeon CPUs (Sandy Bridge type) and one Intel Xeon Phi co-processor (KNC type). This is illustrated by figure 2.

Figure 2. Solution of 100 SLAEs with AgAMG method (99 SLAEs were solved with light construction).

The problem was solved on an irregular grid of ~ 214 thousand cells with 5 unknowns in each cell. AgAMG formed aggregates of 4 cells and performed 3 post-smoothing iterations using a multithreaded multicolored symmetrical Seidel-Gauss smoother (SGS) at each level. Dashed lines are for the time of solving all of the 100 SLAEs with fully constructing AgAMG. Solid lines are for the time of solving all SLAEs in the problem with full construction of AgAMG for the first SLAE only and with ‘light’ construction of AgAMG for the rest SLAEs (2-100). The total number of AgAMG iterations in the both cases is almost the same in this problem (usually, the difference is very small).

X-axis shows the number of threads (the number of processes multiplied by the number of threads in each process), Y-axis shows the time. One can see that the full construction of AMG gives a positive result, which is most noticeable on a co-processor.

These preconditioning techniques have certain overheads, including the need in testing all preconditioners in the set of ordered preconditioners (if several preconditioners were ordered) and such tests should be performed from time to time by solving SLAEs with one of the given methods, which application consumed more time in comparison with the method used to solve SLAE in the previous step. In this case, the repeated test of such method is performed rarely, or delayed till the time, when all the rest methods fail to solve some SLAE [4].

The adaptive preconditioning for a stream of SLAEs is used in the LOGOS Heat software package [8] to simulate the heat transfer in a solid using a simpler algorithm of selecting CG solver’s
preconditioner. The method that demonstrated its efficiency for various problems is used: Block Jacobi method with ILU0, or AgAMG with SGS smoother, or AMG2 with Chebyshev’s smoother. Each of them was better than the two other methods in solving certain problems. So, in the very first steps of solving the problem the preconditioner is selected with which CG solver solved one of the three SLAEs faster than with the others and the rest SLAEs of the problem are solved using this preconditioner.

The adaptive preconditioning was tested on the problem of simulating the thermal state of a package with fissile materials under the fire accident conditions. SLAEs of diffusion contain 1623844 equations and are distributed over 20 single-thread MPI-processes. 108 time steps were performed, 206 SLAEs were solved within the accuracy of 10-8 under criterion of the ratio between the residual norm and the right hand side norm. Table 1 gives data on the time spent to solve SLAEs with the basic method without the adaptive mechanism and with the adaptive mechanism including the three methods above.

| Preconditioning       | SLAE solution/  | Total number of |
|-----------------------|-----------------|-----------------|
| BlockJacobi_mpi(ILU0_float) | 878/16          | 35721/311       |
| Adaptive              | 642/40          | 27582/378       |

Here is the order of solving SLAEs with an adaptive preconditioner of CG solver: the basic method (the first row in table 1) was used in the first 15 time steps and AgAMG was used in all the rest time steps. In total (with regard to results of testing the methods during the problem solution), 61 SLAEs were solved with the basic preconditioner and 145 SLAEs were solved with AgAMG. The adaptive preconditioning provided a 1.37 times reduction of the time spent to solve SLAEs.

3. Conclusion
The adaptive preconditioning for a stream of SLAEs is useful to solve both steady and unsteady physical process simulation problems reducing the time expenditures. It is especially useful for developers of applications, who have small knowledge of the SLAE solution methods and use LParSol as a black box. This allows them to use the fastest methods to solve SLAEs in a wide variety of problems.

References
[1] Zhukov V T, Krasnov M M, Novikova N D and Feodoritova O B 2016 An algebraic multigrid method with adaptive smoothers on the base of Chebyshev’s polynomials Preprints of M.V. Keldysh IAM 113 10.20948/prepr-2016-113
[2] Karpov A P, Bartenev Yu G, Erzunov V A, Petrov D A, Stakanov A N and Shchanikova E B 2019 Solving SLAEs with algebraic multigrid methods in software packages of ITMF Proc. of XVII International Conference “Supercomputing and Mathematical Modeling” ed. R M Shagaliev (Sarov: FSUE “RFNC-VNIIEF”) pp 300-10
[3] Bartenev Yu G and Karpov A P 2019 Repeated use of a hierarchical data structure in an algebraic multigrid solver for systems of linear algebraic equations Voprosy Atomnoy Nauki i Tekhniki. Ser.: Matematicheskoe modelirovanie fizicheskikh protsesov 4 pp 66-77 (in Russian)
[4] Erzunov V A and Bartenev Yu G 2021 Solver adaptation to a SLAE flux Voprosy Atomnoy Nauki i Tekhniki. Ser.: Matematicheskoe modelirovanie fizicheskikh protsesov 1 pp 68-79 (in Russian)
[5] Kaporin I E and Konshin I N 2001 Parallel solution of symmetric positive definite systems
based on decomposition into overlapping blocks Computational Mathematics and Mathematical Physics 41 pp 481-93

[6] Deryugin Yu N, Zelenskiy D K, Glazunov V A et al 2013 Multifunctional LOGOS software package: physical-and-mathematical models for solving problems of aero-, hydrodynamics and heat transport Preprint 111 (in Russian)

[7] OOO "CKO" https://compcenter.org/

[8] Vishnyakov A Yu, Deryugin Yu N, Glazunov V A and Chistyakova I N 2013 LOGOS software package. A computational module for conjugate and coupled heat transfer problems Proc. of XIV International Conference “Supercomputing and Mathematical Modeling” ed. R M Shagaliev (Sarov: FSUE “RFNC-VNIIEF”) pp 154-61 (in Russian)