A simplified AINV method based on nonzero element positions of a coefficient matrix

Kengo Suzuki
Hokkaido University
Japan
kengo.suzuki@eis.hokudai.ac.jp

Takeshi Fukaya
Hokkaido University
Japan
fukaya@iic.hokudai.ac.jp

Takeshi Iwashita
Hokkaido University
Japan
iwashita@iic.hokudai.ac.jp

1 INTRODUCTION

In this research, we consider solving sparse linear systems of equations: $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}(\in \mathbb{R}^{n\times n})$ is a symmetric and positive definite matrix. In the last few years, GPUs have been utilized to efficiently solve a linear system of equations by means of preconditioned Krylov subspace methods. To maximize the potential of GPUs for massive data processing, the preconditioner is desired to have a high degree of parallelism. Whereas incomplete factorization preconditioners involve forward/backward substitutions that are not preferable for implementation on GPUs, sparse approximate inverse (SAI) preconditioners are suited for GPUs because their preconditioning operations are performed by sparse matrix-vector multiplications. However, SAI algorithms tend to take more time than incomplete factorization to construct the preconditioners. Therefore, it is desirable to speed up the preconditioner construction part of the SAI preconditioners.

In this research, we focus on the approximate inverse (AINV) preconditioner [1]. Simplifying some procedures of the AINV algorithm, we propose a new AINV variant, named PS-AINV. Thanks to the simplification, it is expected that the PS-AINV algorithm constructs the preconditioner faster than the standard AINV algorithm.

2 POSITION-BASED SIMPLIFIED AINV

The standard AINV algorithm is performed as in Algorithm 1, based on incomplete conjugation (**A**-orthogonalization) of the standard basis. By adding some dropping techniques to **A**-orthogonalization, the sparse approximate inverse of the form $\mathbf{A}^{-1} \approx \mathbf{Z}\mathbf{D}^{-1}\mathbf{Z}^T$ is obtained, where $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n]$ and $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$.

Algorithm 1 The AINV algorithm

1: **for** $i = 1, \cdots, n$ **do**
2: $\quad \mathbf{z}_i = \mathbf{e}_i$
3: $\quad$ **for** $j = 1, \cdots, i-1$ **do**
4: $\quad\quad p_j = \mathbf{a}_i^T \mathbf{z}_j$
5: $\quad\quad \mathbf{z}_i = \mathbf{z}_i - \frac{p_j}{d_j}\mathbf{z}_j$
6: $\quad\quad$ Drop some elements from $\mathbf{z}_i$.
7: $\quad$ **end for**
8: $\quad d_j = \mathbf{a}_i^T \mathbf{z}_j$
9: **end for**

In the AINV algorithm, p_j is frequently equal to 0, because both $\mathbf{a}_i$ and $\mathbf{z}_j$ are sparse vectors. To exploit this property, it is advisable to calculate the inner product only if there is a possibility that $p_j \neq 0$. However, judging whether p_j can be nonzero also takes non-negligible time. To reduce this additional execution time, we propose a new method, which is named Position-based Simplified AINV (PS-AINV). In the PS-AINV algorithm, the judging process is simplified based on the positions of the nonzero elements of **A**. Specifically, we modify the AINV algorithm so that p_j is calculated only if the (i, j)th element of **A**, a_{ji}, is not equal to 0. The modified algorithm is given by replacing the third line of Algorithm 1 with "**for** $j = 1, \cdots, i-1$ && $a_{ji} \neq 0$ **do**."

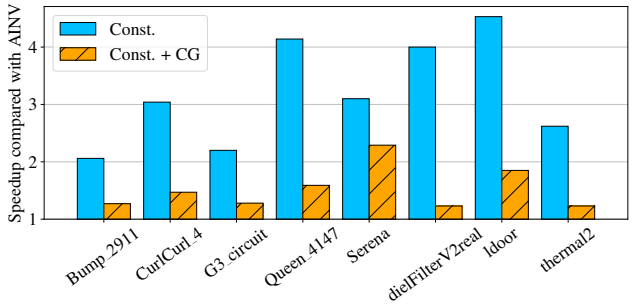

Figure 1: Computational time comparison between SP-AINV and AINV. The results for the preconditioner construction, "Const.," and the combined results for the preconditioner construction and the CG computation, "Const. + CG" are shown.

3 NUMERICAL TESTS

We conducted numerical tests using a computational node that is equipped with Intel Xeon Gold 6230 CPUs and NVIDIA Tesla V100 GPUs. We selected eight datasets from the SuiteSparse Matrix Collection [2] for the test problems. In the tests, we examined the performance of CG solvers preconditioned by AINV and PS-AINV. The preconditioner construction part ran on a CPU, and the iterations of the CG solvers were executed on a GPU.

Figure 1 compares the execution time for PS-AINV with that for AINV for all test cases. This comparison shows that PS-AINV reduced the execution time taken for the preconditioner construction to at least 1/2 for all test matrices. Furthermore, for the three test matrices: Queem_4147, dielFilterV2real, and ldoor, PS-AINV reduced that time to less than 1/4. As a result of this reduction, PS-AINV also shortened the total execution time taken for the preconditioner construction and the CG computation.

REFERENCES

[1] Michele Benzi, Carl D Meyer, and Miroslav Tůma. 1996. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing* 17, 5 (1996), 1135–1149.
[2] Timothy A Davis and Yifan Hu. 2011. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)* 38, 1 (2011), 1–25.