Performance Evaluation of the ES-SC Preconditioned CG Method on GPU

Yuya Kudo[1], Yuki Satake[1], Takeshi Fukaya[1], Takeshi Iwashita[2]
1: Hokkaido University 2: Kyoto University

1. Introduction

- In scientific applications, systems of linear equations are often solved.
  - ✓ In particular, some problems (such as time-dependent analysis) require solving a **sequence of linear systems with the same coefficient matrix**.
- For such problems, Iwashita et al. proposed the **Error vector Sampling-based Subspace Correction (ES-SC)** preconditioner [1].
  - ✓ ES-SC uses information obtained from the initial solve to **accelerate subsequent solutions**.
  - ✓ On CPUs, it has been confirmed that ES-SC is an efficient preconditioner for the CG method.

Contributions

- We **redesigned the ES-SC preconditioned CG (ES-SC-CG) method for GPUs**.
  - ✓ We combined the ES-SC and SDAINV [2] preconditioners.
- We evaluated the ES-SC-SDAINV-CG method on a GPU.
  - ✓ To conduct a more detailed evaluation, **we also evaluated an approach that combines ES-SC with an IC-based preconditioned CG** (ES-SC-MCIC-CG), as in previous work.

2. Problem Settings

We solve a sequence of linear systems:

$$A\boldsymbol{x}^{(k)} = \boldsymbol{b}^{(k)} \quad (k = 1,2,\dots,k_t)$$

- $A \in \mathbb{R}^{n\times n}$ is a large, sparse, and symmetric positive-definite (SPD) matrix.
- $\boldsymbol{x}^{(k)}, \boldsymbol{b}^{(k)} \in \mathbb{R}^n$ are the solution vector and right-hand side vector at step k, respectively.

Note: We assume that $\boldsymbol{b}^{(k)}$ depends on $\boldsymbol{x}^{(k-1)}$.

→ **Therefore, these systems must be solved sequentially**.

3. The ES-SC-CG method for GPUs

Overview of the ES-SC-CG method

- Initial solve ($k = 1$)

Step1: Sample approximate solution vectors during the iterative process.

Step2: Compute error vectors.

✓ Subtract each sampled approximation from the converged solution.

Step3: Construct an auxiliary matrix B from the error vectors.

- Subsequent solves ($k \geq 2$)

The following preconditioner is used for preconditioning in the CG method:

$$M_{\mathrm{E}}^{-1} = \underbrace{M^{-1}}_{\text{standard preconditioner}} + \underbrace{B\left(B^{\mathrm{T}}AB\right)^{-1}B^{\mathrm{T}}}_{\text{SC preconditioner}}$$

→ **Accelerate subsequent solves**.

GPU Implementation of the ES-SC-CG Method

- It is **important to choose a standard preconditioner** for use with ES-SC.

Standard preconditioner	Building the preconditioner	Features of calculation	GPU-friendliness
MCIC	**Low cost**	Forward and backward substitution	△
SDAINV	High cost	**Sparse matrix-vector multiplication (SpMV)**	◎

← Parallelized incomplete Cholesky (IC) preconditioner

→ **We combined ES-SC and SDAINV**.

- We implemented our solver using CUDA.
- Construction of the MCIC and SDAINV preconditioner matrices is performed on the CPU.
- To efficiently execute SpMV operations, we used **Sliced-ELL** [3] as the sparse matrix storage format.
- We used cuBLAS and cuSOLVER for inner products and ES-SC–specific computations (e.g., auxiliary matrix construction); all other computations are implemented using our own CUDA kernels.

4. Numerical experiments

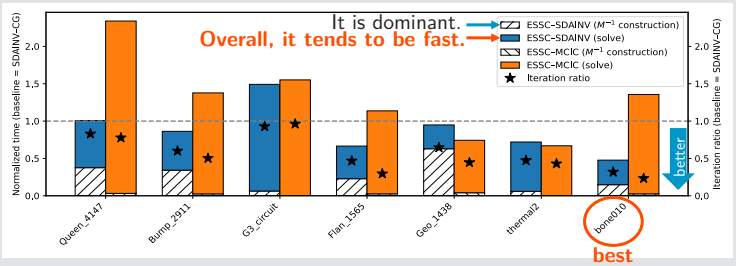
Experimental setup

- **Platform**: NVIDIA H100 GPU (Hokkaido University)
- **Test data**: seven SPD matrices
- **Methods compared**:
  - ✓ **Baseline:** SDAINV-CG (without ES-SC)
  - ✓ ES-SC-SDAINV-CG
  - ✓ ES-SC-MCIC-CG

Results

1. Total number of iterations and computation time

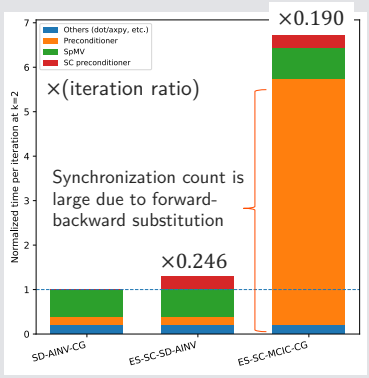
- The iteration ratios (star markers) are below 1.0 for all tested matrices, showing that **ES-SC combined with SDAINV consistently reduces iterations**.
- The **total computation time is reduced for 5/7 matrices**.
- **When k_t increases, ES-SC-SDAINV-CG may become faster in terms of total computation time** because the preprocessing construction cost (the hatching area) becomes relatively smaller.



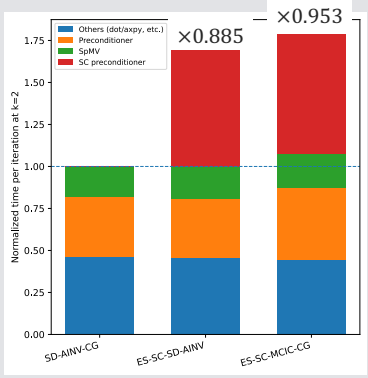
Note: The total computation time consists of the time required for building the preconditioner on a CPU and the time required for solving k_t systems on a GPU.

2. Breakdown of computation time per iteration for $k = 2$

- If the computational cost of the **SC preconditioning is low and the number of iterations is sufficiently reduced, the total computation time will be shorter**.
- Conversely, if the computational cost of the SC preconditioning is high and the number of iterations is not reduced enough, the total computation time increases.



(a) Speedup case: bone010



(b) Slowdown case: G3_circuit

5. Conclusion

Summary

We implemented ES-SC-SDAINV-CG for a GPU and confirmed its effectiveness.

- ✓ **Number of iterations**: reduced for all test matrices
- ✓ **Computation time**: reduced for five test matrices

Future work

The validity has been confirmed on CPUs for a sequence of linear systems with the same asymmetric matrix [4].

→ Redesign for GPUs.

References

1. Takeshi Iwashita et al. Convergence acceleration of preconditioned conjugate gradient solver based on error vector sampling for a sequence of linear systems. Numerical Linear Algebra with Applications 30 (2023), e2512
2. Kengo Suzuki et al. A New AINV Preconditioner for the CG Method in Hybrid CPU-GPU Computing Environment. Journal of Information Processing 30 (2022), 755–765
3. Alexander Monakov et al. Automatically tuning sparse matrix-vector multiplication for GPU architectures. Proceedings of the 5th International Conference on High Performance Embedded Architectures and Compilers (HiPEAC '10) (2010), 111-125
4. Hirotoshi Tamori et al. Subspace Correction Preconditioning for Solving a Sequence of Asymmetric Linear Systems Using the Bi-CGSTAB Method, Journal of Information Processing 31 (2023), Pages 875-884