A GPU-Friendly SSOR Preconditioner Based on an Approximate Inverse of Triangular Matrices

Yuya Kudo[1)], Yuki Satake[1)] (Advisor), Takeshi Fukaya[1)] (Advisor), Takeshi Iwashita[2)] (Advisor) 1) Hokkaido University 2) Kyoto University

1. Introduction

Problem: large sparse system of linear equations

$$A\boldsymbol{x} = \boldsymbol{b}$$

- A: symmetric positive definite (SPD) matrix
- Solver: preconditioned conjugate gradient (PCG) method
- Applications: structural analysis, electromagnetics, etc.

Motivation: design a GPU-friendly solver

- Preconditioning techniques are important for efficient solving
- **Approximate inverse approaches are promising for GPUs** [1]

Goal: develop a GPU-friendly SSOR preconditioner

(SSOR: Symmetric Successive Over-Relaxation [2])

- **Incorporate an approximate inverse into SSOR**
- **Improve the construction of an approximate inverse**
- **Demonstrate effectiveness of the proposed method on a GPU** (compare with existing parallel SSOR preconditioners)

2. Proposed method

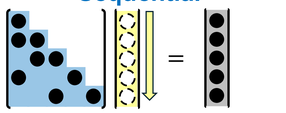
SSOR preconditioner: calculate $M_{\mathrm{SSOR}}\boldsymbol{z} = \boldsymbol{r}$ for $\boldsymbol{z}$ in each PCG iteration, where $M_{\mathrm{SSOR}} = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega L)^{\mathrm{T}}$

ω: relaxation parameter $(0 < \omega < 2)$, L: the strictly lower triangular part of A, D: the diagonal part of A

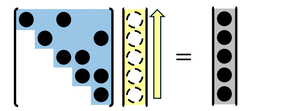
Original

Forward substitution

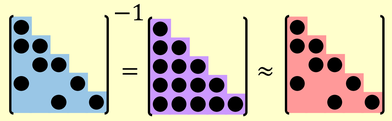
Sequential



Backward substitution



Key idea: incorporate an approximate inverse of $D+\omega L$



W: **same sparsity pattern as** $D+\omega L$

Impact of an approximate inverse

- **Construction cost of W occurs**
- **Affects solve time**, due to:
  - $M_{\mathrm{SSOR}}\boldsymbol{z} = \boldsymbol{r}$ replaced by SpMV
  - Change in iteration count

Construction of W

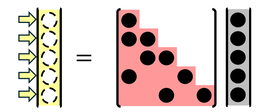
Note: Based on [3]

for $i = 1,2,\dots,n$ **do**
  $w_{ii} \leftarrow 1/l'_{ii}$ (Parallelizing ⇒ **setup cost ↓**)
  for $k = i+1,\dots,n \wedge l'_{ki} \neq 0$ do
    $w_{ki} \leftarrow 0$ ($L' = D + \omega L$)
    for $j = i,\dots,k-1 \wedge l'_{kj} \neq 0$ do
      $w_{ki} \mathrel{-}= l'_{kj} w_{ji}$ (Dropping nonzero entries ⇒ **SpMV cost ↓**)
    $w_{ki} \leftarrow w_{ki}/l'_{kk}$
    if $|w_{ki}| < \tau$ **then** $w_{ki} \leftarrow 0$

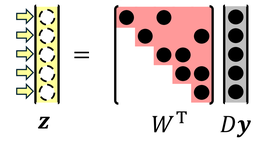
Proposal

SpMV

Parallel



SpMV



SpMV: Sparse Matrix-Vector multiplication

GPU-friendly

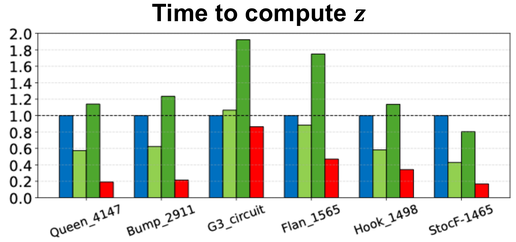
3. Numerical experiments

Objective: comparison with other parallel SSOR preconditioners for PCG

- MC (**baseline**): forward and backward substitutions parallelized by coloring (ordering) [4]
- TNS ($p = 1$), TNS ($p = 2$): $(D+\omega L)^{-1}$ approximated by p-term truncated Neumann series (**SpMV-based calculation**) [5,6]
- **Proposed method**: incorporating an approximate inverse of $D+\omega L$ (**SpMV-based calculation**)

Note: all values are normalized by MC. Smaller values are better.

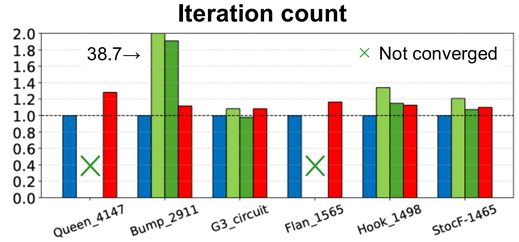
Time to compute z



vs. MC: **faster** (since SpMV-based)

vs. TNS: **faster** (since nonzero reduction)

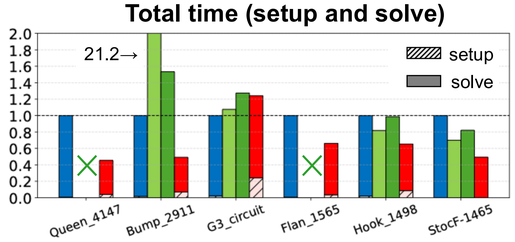
Iteration count



vs. MC: **slightly more iterations**

vs. TNS: **fewer iterations in some cases**

Total time (setup and solve)



vs. MC and TNS: **faster except for one case**, despite including setup cost

Experimental Conditions

- **Environment**: Grand Chariot 2 (@Hokkaido Univ., H100 GPU and Xeon Gold 6548Y+ CPU), CUDA 12.8.0 (nvcc -O3 -Xcompiler -fopenmp), FP64
- **Matrix data**: matrices from SuiteSparse Matrix Collection (dimension $\geq 10^6$), pre-scaled so that $D = I$, stored in CRS format
- **SSOR parameters**: $\omega = 0.8$, $\tau = 0.01$
- **PCG settings**: stopped when relative residual norm $\|\boldsymbol{r}_i\|/\|\boldsymbol{b}\| \leq 10^{-10}$ or max 100,000 iterations, initial approximate solution $\boldsymbol{x}_0 = \boldsymbol{0}$, $\boldsymbol{b} = [1,1,\dots,1]^{\mathrm{T}}$
- **Execution**: **setup on a single CPU (with 16 threads), solve on a single GPU**

4. Conclusion

Summary

- **Developed a GPU-friendly variant of the SSOR preconditioner based on an approximate inverse of triangular matrices**
- **Demonstrated its effectiveness on an NVIDIA H100 GPU**

Future work

- Optimize parameters (ω, τ) of the proposed method
- Evaluate the proposed method against other preconditioners (e.g., IC, AINV)
- Exploit mixed-precision computing
- Extend to nonsymmetric linear systems

References

1. K. Suzuki et al., "A New AINV Preconditioner for the CG Method in Hybrid CPU-GPU Computing Environment," J. Inf. Process., vol. 30, pp. 755-765, 2022.
2. Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed. Philadelphia, PA: SIAM, 2003.
3. H. Anzt et al., "Incomplete Sparse Approximate Inverses for Parallel Preconditioning," Parallel Comput., vol. 71, pp. 1-22, 2018.
4. C. H. Wu, "A Multicolour SOR Method for the Finite-Element Method," J. Comput. Appl. Math., vol. 30, no. 3, pp. 283-294, 1990.
5. M. Benzi et al., "A Comparative Study of Sparse Approximate Inverse Preconditioners," Appl. Numer. Math., vol. 30, no. 2, pp. 305-340, 1999.
6. R. Helfenstein et al., "Parallel Preconditioned Conjugate Gradient Algorithm on GPU," J. Comput. Appl. Math., vol. 236, no. 15, pp. 3584-3590, 2012.