Performance Comparison of Linear System Solvers for CMFD Acceleration on GPU Architectures

Namjae Choi, Junsu Kang, Han Gyu Joo*
Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 08826, Korea
**Corresponding author: joohan@snu.ac.kr*

1. Introduction

Graphic Processing Units (GPUs) are increasingly becoming the main means of calculation in the scientific fields, due to its cost-effectiveness and massive vector processing capability. The potential of employing GPUs in the reactor physics calculations is also being explored by the nTRACER [1] developer group at Seoul National University. nTRACER is now one of the members of the high performance computing consortium in Korea, and the whole code is being renovated for the application to the heterogeneous high performance computers as a 4-year research project.

nTRACER uses the method of characteristics (MOC) coupled with the coarse mesh finite difference (CMFD) acceleration for solving the transport equation. So far the focus of optimization was on the MOC calculation since the majority of the calculation time is spent on the ray tracing calculation. As the result, we have recently observed a promising result on the feasibility of GPU acceleration of the MOC calculation [2]. Afterwards the optimization of the CMFD acceleration became critical. Therefore, performance of several linear system solvers on GPU under the CMFD framework are studied in this work. Especially, the numerical performance under the single precision arithmetic as well as the feasibility of massive parallelization are examined.

2. Theoretical Backgrounds

2.1 CMFD Power Iteration

Generalized CMFD eigenvalue problem has the form:

$$\mathbf{M\Phi} = \frac{1}{k_{eff}}\mathbf{F\Phi} \tag{1}$$

where $\mathbf{M}$ is the migration matrix which represents the migration of neutrons through energy and space due to scattering and diffusion. $\mathbf{F}$ is the production matrix that incorporates the effect of neutron production caused by fission. The migration matrix is ordered such that dense block matrices which incorporate the scattering terms are laid along the diagonal, as depicted in **Figure 1**.

The scaled power iteration of Eq. (1) is formulated as follows:

$$\mathbf{\Phi}^{(n)} = \frac{1}{k_{eff}^{\ (n-1)}}\mathbf{M}^{-1}\mathbf{F\Phi}^{(n-1)} \tag{2}$$

where n is the power iteration index. However, because the direct inversion of $\mathbf{M}$ is expensive, it is replaced by solving the linear system using iterative methods:

$$\mathbf{M\Phi}^{(n)} = \frac{1}{k_{eff}^{\ (n-1)}}\boldsymbol{\chi\psi}^{(n-1)}. \tag{3}$$

where $\boldsymbol{\psi}$ is the fission source vector and $\boldsymbol{\chi}$ is the fission spectrum matrix.

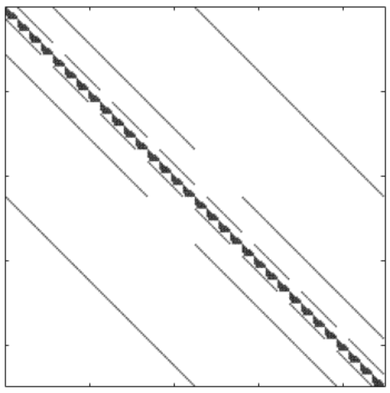
Figure 1. Migration matrix structure.

The solution of this linear system does not require full convergence since the fission source is still inaccurate. Such unnecessity of full convergence makes the linear system solution in the CMFD problem differ from other ordinary linear system problems.

Once the linear system is solved, the multiplication factor is updated using:

$$k_{eff}^{\ (n)} = k_{eff}^{\ (n-1)}\frac{\left\langle \boldsymbol{\psi}^{(n)}, \boldsymbol{\psi}^{(n)} \right\rangle}{\left\langle \boldsymbol{\psi}^{(n)}, \boldsymbol{\psi}^{(n-1)} \right\rangle} \tag{4}$$

2.2 Linear System Solution Methods

2.2.1 Bi-Conjugate Gradient Stabilized (BiCGSTAB)

BiCGSTAB is one of the most widely employed Krylov method for solving linear systems. The Krylov methods are normally augmented by a preconditioning scheme, which accelerates the convergence by improving the condition number of the matrix. The algorithm of the preconditioned BiCGSTAB is described in **Figure 2**, in which the matrix $\mathbf{K}$ is the preconditioner.

In this paper, *Incomplete LU (ILU)* preconditioner will be studied. The ILU preconditioner preserves the sparsity pattern of the original matrix in the factorized matrices. The inversion of the preconditioner then can be done by the LU solve with a small computation load. However, the forward and the backward substitution are basically serial, so the parallel performance may be poor.

```
1. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
2. $\rho_0 = \alpha = \omega_0 = 1$
3. $\mathbf{v}_0 = \mathbf{p}_0 = 0$
4. For $i = 1, 2, 3, ...$
    4.1 $\rho_i = (\mathbf{r}_0, \mathbf{r}_{i-1})$
    4.2 $\beta = (\rho_i / \rho_{i-1})(\alpha / \omega_{i-1})$
    4.3 $\mathbf{p}_i = \mathbf{r}_{i-1} + \beta(\mathbf{p}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1})$
    4.4 $\mathbf{y} = \mathbf{K}^{-1}\mathbf{p}_i$
    4.5 $\mathbf{v}_i = \mathbf{A}\mathbf{y}$
    4.6 $\alpha = \rho_i / (\mathbf{r}_0, \mathbf{v}_i)$
    4.7 $\mathbf{h} = \mathbf{x}_{i-1} + \alpha\mathbf{y}$
    4.8 $\mathbf{s} = \mathbf{r}_{i-1} - \alpha\mathbf{y}$
    4.9 $\mathbf{z} = \mathbf{K}^{-1}\mathbf{s}$
    4.10 $\mathbf{t} = \mathbf{A}\mathbf{z}$
    4.11 $\omega_i = (\mathbf{t}, \mathbf{s}) / (\mathbf{t}, \mathbf{t})$
    4.12 $\mathbf{x}_i = \mathbf{h} + \omega_i\mathbf{z}$
    4.13 $\mathbf{r}_i = \mathbf{s} - \omega_i\mathbf{t}$
    4.14 If $\|\mathbf{r}_i\| / \|\mathbf{r}_0\| < \varepsilon$, stop the iteration
```

Figure 2. Preconditioned BiCGSTAB algorithm.

2.2.2 Block Successive Over-relaxation (BSOR)

SOR is a famous conventional stationary method for solving linear systems. For the parallelization of SOR, red-black ordering strategy is used. The structure of the red-black ordered CMFD linear system can be found in **Figure 3** and can be expressed as:

$$\begin{pmatrix} \mathbf{D}_R & \mathbf{M}_R \\ \mathbf{M}_B & \mathbf{D}_B \end{pmatrix}\begin{pmatrix} \mathbf{\Phi}_R \\ \mathbf{\Phi}_B \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_R \\ \mathbf{Q}_B \end{pmatrix} \tag{5}$$

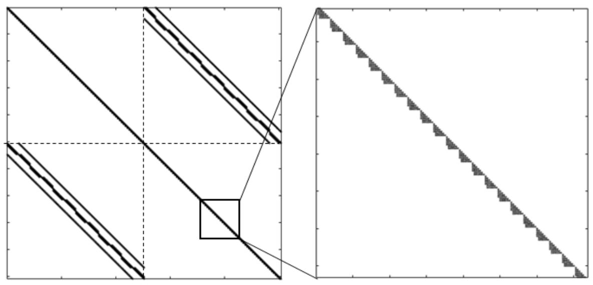

Figure 3. Red-black ordered CMFD linear system structure.

However, the existence of the scattering terms prevents the point SOR scheme from parallelization. Hence, the block SOR scheme is introduced:

$$\begin{aligned} \mathbf{\Phi}_R^{(n)} &= \omega\mathbf{D}_R^{-1}(\mathbf{Q}_R^{(n-1)} - \mathbf{M}_R\mathbf{\Phi}_B^{(n-1)}) + (1-\omega)\mathbf{\Phi}_R^{(n-1)} \\ \mathbf{\Phi}_B^{(n)} &= \omega\mathbf{D}_B^{-1}(\mathbf{Q}_B^{(n-1)} - \mathbf{M}_B\mathbf{\Phi}_R^{(n)}) + (1-\omega)\mathbf{\Phi}_B^{(n-1)} \end{aligned} \tag{6}$$

Still the direct inversion of the diagonal blocks is not preferable, so the source iteration is employed and the up-scattering terms are removed from the linear system so that the diagonal blocks have lower triangular shape, as illustrated in **Figure 3**. Then, the diagonal blocks can be inverted readily using the forward substitution.

SOR requires an estimate of the spectral radius ρ_{GS} of the Gauss-Seidel iteration matrix. Hence, a dummy Gauss-Seidel step should be carried out initially. Once it is known, the optimal relaxation factor is found by the Young's formula:

$$\omega = \frac{2}{1+\sqrt{1-\rho_{GS}}}. \tag{7}$$

And the number of SOR iterations n per outer iteration is pre-determined using

$$n = \frac{\log \varepsilon}{\log \rho_{SOR}} \tag{8}$$

where ε is the desired error reduction ratio, and

$$\rho_{SOR} = \omega - 1 \tag{9}$$

2.2.3 Alternating Anderson-Jacobi (AAJ)

Jacobi iteration is the simplest form of the fixed-point iteration. In the Jacobi scheme, the matrix $\mathbf{A}$ of a linear system $\mathbf{Ax = b}$ is split as

$$\mathbf{A = D + R} \tag{10}$$

where $\mathbf{D}$ is the diagonal matrix and $\mathbf{R}$ consists of the off-diagonals. Now, define the residual as

$$\mathbf{f}(\mathbf{x}) = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{R}\mathbf{x}) - \mathbf{x}. \tag{11}$$

However, the Jacobi scheme typically suffers from slow convergence, so the Anderson extrapolation technique was introduced by P. Pratapa *et al* [4]. In the Anderson-Jacobi scheme, the fixed-point iteration is generalized to

$$\mathbf{x}_{k+1} = \bar{\mathbf{x}}_k + \beta\mathbf{f}(\bar{\mathbf{x}}_k) \tag{12}$$

where $\bar{\mathbf{x}}_k$ denotes the weighted average of the previous $m+1$ iterates

$$\bar{\mathbf{x}}_k = \mathbf{x}_k - \sum_{j=1}^{m}\gamma_j(\mathbf{x}_{k-m+j} - \mathbf{x}_{k-m+j-1}). \tag{13}$$

The weighting factors $\Gamma_k = [\,\gamma_1 \ \cdots \ \gamma_m\,]^T$ are chosen so as to minimize the l^2-norm of the residual

$$(\mathbf{F}_k^T\mathbf{F}_k)\Gamma_k = \mathbf{F}_k^T\mathbf{f}(\mathbf{x}_k) \tag{14}$$

where

$$\mathbf{F}_k = \left[\left(\mathbf{f}(\mathbf{x}_{k-m+1}) - \mathbf{f}(\mathbf{x}_{k-m})\right) \ \cdots \ \left(\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}_{k-1})\right)\right]. \tag{15}$$

Then, the update formula of Eq. (12) can be written as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \beta\mathbf{f}(\mathbf{x}_k) - (\mathbf{X}_k + \beta\mathbf{F}_k)(\mathbf{F}_k^T\mathbf{F}_k)^{-1}\mathbf{F}_k^T\mathbf{f}(\mathbf{x}_k) \tag{16}$$

where

$$\mathbf{X}_k = \left[(\mathbf{x}_{k-m+1} - \mathbf{x}_{k-m}) \ \cdots \ (\mathbf{x}_k - \mathbf{x}_{k-1}) \right]. \qquad (17)$$

This extrapolation replaces the ordinary Jacobi scheme every p times of iteration, and this alternation process is repeated until the relative residual reduction reaches ε.

The calculation of $\mathbf{F}_k^T \mathbf{F}_k$ is sensitive to the round-off error. Therefore, it should be computed with the double precision arithmetic, which is the main drawback of this method. Still, the extrapolation does not have to be done frequently and the dimension of $\mathbf{F}_k^T \mathbf{F}_k$ is restricted; our sensitivity test told us that setting $m = p = 10$ is enough. Thus, AAJ still holds an advantage over other methods that the majority of the calculation can be carried out with the naturally parallelizable Jacobi scheme.

2.3 Iterative Refinement

Even the commercial GPUs have far larger single precision computing power than CPUs. However, it is discouraged to solve the CMFD problem directly by the single precision arithmetic, since the achievable source residual level has a limited lower bound imposed by the round-off error. Therefore, to solve the CMFD problem effectively on GPUs exploiting the single precision, the iterative refinement technique is introduced. It allows to use the single precision arithmetic for the linear system solver while still ensuring the double precision accuracy. The procedure is explained in **Figure 4** in which the subscript indicates the floating point precision.

1. Compute residual: $\mathbf{r}_{(8)} = \mathbf{b}_{(8)} - \mathbf{A}_{(8)}\mathbf{x}_{(8)}$
2. Solve for correction: $\mathbf{A}_{(4)}\mathbf{d}_{(4)} = \mathbf{r}_{(4)}$
3. Add correction: $\mathbf{x}^*_{(8)} = \mathbf{x}_{(8)} + \mathbf{d}_{(8)}$

Figure 4. Iterative refinement technique procedure.

3. Results and Discussion

The implementation of the solvers was done using the basic routines provided by CUBLAS and CUSPARSE libraries, to ensure a high level of optimization. All of the sparse matrices were saved in the CSR format to use the libraries, even though a specific saving format may be more efficient for highly structured matrices such as the CMFD matrix [4]; however, the penalty of using the general sparse format applies to all solvers commonly, so it does not affect the fairness of the comparison.

The VERA problem 5A-2D [5] was chosen as the test problem. Its configuration is illustrated in **Figure 5**. The number of unknowns of this problem is 879,511 (18,713 pins × 47 energy groups) and the number of nonzero is 26,182,017. It is effectively the largest dimension that a single device has to handle in the applications, because the 2D/1D method parallelizes the problem into planes.

Table 1 shows the performances of the linear system solvers. Three inner iteration tolerances were tested: 0.1, 0.05, and 0.01. Each line in the table corresponds to each tolerance. The CMFD outer iteration exits when the relative source residual reduction reaches 0.1. For the MOC solver, P_1 scattering was used. The number of outers indicates the CMFD outer iteration count, and the Ax = b time represents the total time spent for the solver, including the refinement step. For AAJ, $\beta = 0$ was used so that a few terms are cancelled out, because according to our sensitivity tests, giving β a nonzero value did not result in a meaningful convergence enhancement. For the GPU, GeForce GTX 1070 was used.

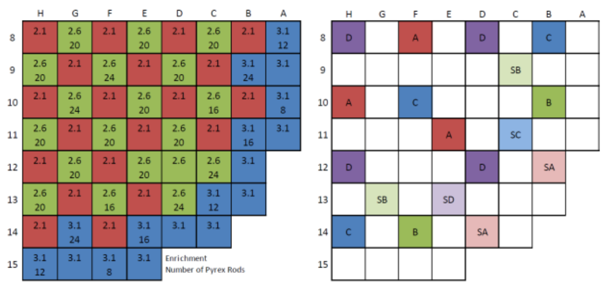

Figure 5. VERA problem 5A-2D configuration [5].

Table 1. Performance of the solvers w/o group condensing.

Description	BiCGSTAB		BSOR	AAJ
	ILU	None		
k_{eff}	1.00270			
# of MOCs	7			
# of Outers	283 284 276	288 279 279	955 931 847	310 273 277
Ax = b Time (s)	38.2 50.4 59.8	62.1 66.1 83.5	42.7 50.4 65.8	59.5 62.4 83.8

From **Table 1**, following interpretations can be made:

1. Even though the triangular solver is not effective, the unpreconditioned BiCGSTAB consumes much more time than the ILU preconditioned BiCGSTAB due to increased inner iterations.
2. The reason for significantly large number of outer iterations of BSOR compared to other solvers is the up-scattering source iteration. Increase of the outer iterations results in not only more inner iterations but also more iterative refinement steps which are performed by the double precision arithmetic.
3. Poor performance of AAJ is contributed by large number of inner iterations and the extrapolation cost. Even with the extrapolation, still dozens of Jacobi iterations are required in AAJ.

With group condensation, however, the performance of the solvers can be improved. The group condensation accelerates the convergence of the global fission source distribution in a multigrid manner – by collapsing the multi-group problem into a few-group problem. In this paper, two-group condensation was employed. The two-group problem does not include up-scattering.

Table 2 shows the performance of the solvers with the condensed two-group CMFD. The entire iteration procedure is as follows:

1. Do 5 initial multi-group CMFD iterations.
2. Do two-group CMFD iterations until the relative source residual reduction is smaller than 0.1.
3. Do multi-group CMFD iterations until the relative source residual reduction is smaller than 0.1.

Table 2. Performance of the solvers w/ group condensing.

Description	BiCGSTAB		BSOR	AAJ
	ILU	None		
k_{eff}	1.00270			
# of MOCs	7			
# of MG Outers	56 56 56	57 56 56	59 60 56	56 56 56
# of 2G Outers	446 444 441	453 442 442	852 712 509	483 457 440
Ax = b Time (s)	24.3 30.5 46.9	19.9 22.8 28.9	5.2 5.7 6.7	14.9 20.0 29.8

The performance of the linear system solvers after incorporating the group condensation differ largely from the multi-group stand-alone result. Such differences can be explained as follows:

1. The group condensation compensated the increase of outer iterations in BSOR from the up-scattering source iteration. It reduced the number of multi-group outer iterations to the same level with other solvers, making BSOR the most efficient solver.
2. Total computation time of the ILU preconditioned BiCGSTAB is not reduced as much as other solvers, which pushes the ILU preconditioned BiCGSTAB to the least efficient solver. It is due to the overhead imposed by the triangular solver employed to solve the preconditioner of the group-condensed system.
3. BSOR also employs a lower triangular solver, but the bottleneck is not observed. It is because of the structure of the triangular matrices. The triangular matrix of BSOR is composed of many independent block matrices (scattering blocks), which allows high degree of parallelization. On the other hand, the ILU preconditioner matrices contain the global spatial coupling terms which deteriorate parallelity. Same applies to the multi-group problem as well.

4. Conclusion

Several stationary and Krylov linear system solution methods were implemented employing CUDA, and the performance was examined under the CMFD framework. Following missions were imposed on the solvers: 1. High parallel efficiency on the GPU architectures, 2. Numerical stability under the single precision arithmetic and iterative refinement, and 3. Effectiveness under the CMFD power iteration and group-condensation. And it appeared that BSOR is the most amenable of the solvers to the GPU acceleration of the CMFD power iteration. Especially, the group-condensation compensated the up-scattering source iteration of BSOR. Meanwhile, ILU preconditioned BiCGSTAB could not solve the small linear systems occurring from the group condensation effectively, resulting in even worse performance than the unpreconditioned version.

Nonetheless, preconditioned Krylov methods are still preferred in that it can solve various linear systems with a wide range of numerical properties. The utilization of unpreconditioned BiCGSTAB is discouraged because of its poor convergence behavior. Also, the convergence of stationary methods is only assured when the system is diagonally dominant. However, the diagonal dominance may be broken in the CMFD linear system due to the existence of the correction factors. Therefore, use of highly parallelizable preconditioners, such as the *Sparse Approximate Inverse (SPAI)* preconditioner, should be investigated. Also, the convergence property of BSOR should be examined with more various type of problems, including three-dimensional problems that require axial calculations.

Furthermore, this research limits the scope on the use of a single device. However, to retain the scalability to large-scale parallel machines, performance of the linear system solvers should be examined under the presence of communication as well.

ACKNOWLEDGEMENTS

This research is supported by National Research Foundation of Korea (NRF) Grant No. 2016M3C4A7952631 (Realization of Massive Parallel High Fidelity Virtual Reactor)

REFERENCES

[1] Y. S. Jung, C. B. Shim, C. H. Lim, H. G. Joo, “Practical Numerical Reactor Employing Direct Whole Core Neutron Transport and Subchannel Thermal/Hydraulics Solvers,” Annals of Nuclear Energy, Vol. 62, pp. 357–374 (2013).
[2] N. J. Choi, J. S. Kang, H. G. Joo, “Massively Parallel Method of Characteristics Neutron Transport Calculation with Anisotropic Scattering Treatment on GPUs,” International Conference on High Performance Computing in Asia-Pacific Region, Tokyo, Japan, Jan 28–31 (2018).
[3] P. Pratapa, P. Suryanarayana, J. Pask, “Anderson Acceleration of the Jacobi Iterative Method: An Efficient Alternative to Krylov Methods for Large, Sparse Linear Systems,” Journal of Computational Physics, Vol. 306, pp. 43–54 (2016).
[4] J. S. Kang, H. G. Joo, “GPU-based Parallel Krylov Linear System Solvers for CMFD Calculation in nTRACER,” Transactions of the Korean Nuclear Society Spring Meeting, Jeju, Korea, May 17-18 (2018).
[5] A. T. Godfrey, “VERA Core Physics Benchmark Progression Problem Specifications,” Rev. 4, CASL-U-2012-0131-004 (2014).