# GPU Accelerated $S_N$ Transport Simulation for Shielding Analysis using STRAUM

Ao Zhang[a,b], Ser Gi Hong[a*]
[a]*Dept. of Nuclear Engineering, Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul, Korea*
[b]*Shanghai Institute of Applied Physics, Chinese Academy of Sciences, 2019 Jialuo-ro, Shanghai, China*
[*]*Corresponding author: hongsergi@hanyang.ac.kr*

***Keywords** :* STRAUM, GPU parallel computing, $S_N$ transport, shielding analysis

## 1. Introduction

Radiation shielding is critical for the safety of nuclear reactor operations, radioactive medical treatments, transportation of spent nuclear fuels, and so on. To evaluate radiation distributions in nuclear devices, the steady-state Boltzmann transport equation necessitates being solved accurately and efficiently. STRAUM ($S_N$ Transport for Radiation Analysis with Unstructured Meshes), under the development at Hanyang University, has been developed to simulate radiation transport in complex geometries with good accuracy and efficiency.

STRAUM utilizes the multi-group method, the discrete ordinate ($S_N$) method, the Linear Discontinuous Expansion Method with Subcell Balances (LDEM-SCB) method [1] for energy, angular, spatial discretizations, respectively. In STRAUM, the Krylov subspace methods combined with preconditioners involving DSA (Diffusion Synthetic Acceleration) and TSA (Transport Synthetic Acceleration) , along with a Gauss Seidel iteration scheme for energy group sweep, have been incorporated to enhance convergence [2]. Additionally, the computing performance of STRAUM was improved through CPU parallelization using shared memory programming models, including Taskflow and OpenMP [3]. However, the simulation efficiency remains a challenge, especially for problems involving a large amount of physical unknowns, necessitating further improvements.

By leveraging parallel computing powers from multiple consumer-grade GPUs, the computing efficiency of STRAUM has been greatly improved in this work. STRAUM is then applied to the shielding analysis of a Pressurized Water Reactor (PWR), showing its good accuracy and efficiency from the code-to-code comparisons with MCNP.

## 2. Theory and methodology

The steady-state multi-group Boltzmann transport equation solved in STRAUM is discretized using $S_N$ method, which is given by

$$\vec{\Omega}_n \nabla \psi_n^g(\vec{r}) + \Sigma_t^g(\vec{r}) \psi_n^g(\vec{r}) = Q_{n,\mathrm{ex}}^g(\vec{r}) + \sum_{g'=1}^{G} \sum_{l=0}^{L} \sum_{m=-l}^{l} (2l+1) \Sigma_{s,l}^{g' \to g}(\vec{r}) R_{n,l,m} \phi_{l,m}^{g'}(\vec{r}), \quad (1)$$

where $\vec{\Omega}_n$ is the $n$-th angular direction, $\psi_n^g$ is the $g$-th group angular flux, $\vec{r}$ is the position vector. $\Sigma_t^g$ is the macroscopic total cross sections, $\Sigma_{s,l}^{g' \to g}$ is the $l$-moment of the group transfer scattering cross section from $g'$ to $g$. $R_{n,l,m}$ is the real spherical harmonics. $Q_{n,\mathrm{ex}}^g$ is the external source, while $\phi_{l,m}^g(\vec{r})$ is the flux moments, evaluating by

$$\phi_{l,m}^g(\vec{r}) \cong \sum_n w_n R_{n,l,m} \psi_n^g(\vec{r}), \quad (2)$$

where $w_n$ is the corresponding quadrature weight for the $n$-th discrete ordinate direction. For the spatial discretization, the LDEM-SCB method is used in STRAUM, resulting in four unknowns for each computing cell [1].

To apply the Krylov subspace methods, Eq. (1) can be written as a compact operator form, i.e.,

$$\boldsymbol{L}\psi = \boldsymbol{M}\boldsymbol{S}\phi + Q_{\mathrm{ex}}, \quad (3)$$

where $\boldsymbol{L}$ is the transport operator composed of streaming and collision terms, $\boldsymbol{M}$ is the moment-to-discrete operator for converting harmonic moments to discrete angles, $\boldsymbol{S}$ is the scattering operator. By introducing an operator $\boldsymbol{D}$ to covert the angular flux to flux moments using Eq. (2), one can get a new formula from Eq. (3) as

$$(\boldsymbol{I} - \boldsymbol{D}\boldsymbol{L}^{-1}\boldsymbol{M}\boldsymbol{S})\phi = \boldsymbol{D}\boldsymbol{L}^{-1} Q_{\mathrm{ex}}, \quad (4)$$

In this work, all energy groups are divided into one or multiple group chunks, even in the absence of up-scattering, while all the up-scattering energy groups are typically assigned to the last group chunk. Eq. (4) for one energy group chunk can be written as

$$(\boldsymbol{I} - \boldsymbol{D}\boldsymbol{L}^{-1}\boldsymbol{M}\boldsymbol{S}_{\mathrm{GC}})\phi_{\mathrm{GC}} = \boldsymbol{D}\boldsymbol{L}^{-1}(\boldsymbol{M}\boldsymbol{S}_{\mathrm{to\_GC}}\phi_{\mathrm{to\_GC}} + Q_{\mathrm{ex}}), \quad (5)$$

where $\phi_{\mathrm{GC}}$ and $\phi_{\mathrm{to\_GC}}$ represent the vectors of the flux moments for the current group chunk and the group chunks scattering down to the current group. On the right-hand side of Eq. (5), the flux moments are derived from the converged high-energy group chunks, under the assumption that all up-scattering groups are assigned to the last group chunk. $\boldsymbol{S}_{\mathrm{GC}}$ means the scatterings between all energy groups in the current group chunk, and $\boldsymbol{S}_{\mathrm{to\_GC}}$ means the scattering from the upper group chunks to the current group chunk. Eq. (5), the linear system of the flux moments, is solved by the Krylov subspace methods in a matrix-free manner using the basic algebra operations in STRAUM.

## 3. GPU parallel schemes

The GPU-based $S_N$ transport sweep algorithm is presented in Algorithm 1, representing the inversion of

transport operator. The transport sweep is executed sequentially, octant by octant, to handle reflective boundary conditions. Within the calculation loop of each octant, the transport kernel is launched for the current energy group chunk, the angular directions within the octant, and all cells, after initializing the angular fluxes of boundary cells. The grid size of transport kernel is determined as the product of the number of energy groups within the group chunk and the number of angular directions within the octant, with one thread block assigned to process all cells. This design exposes the parallelism across energy groups and angles to thread blocks, while the cell-level parallelism is handled by threads within each block. The unknown equation for each cell is solved by one CUDA thread along with calculating the coefficient matrix and the incoming angular flux. To maintain the required serial sweep order between the sweep levels, the low-cost thread synchronization is used within each thread block.

**Algorithm 1** Transport sweep for one group chunk

```
Input: vector V_in
Output: vector V_out = DL^{-1} V_in
Set current energy group chunk: {g^1_GC, g^2_GC, ...}
Initialize flux moments and angular flux
for oct ∈ octants do
    Copy boundary incoming flux to angular flux
    for g ∈ {g^1_GC, g^2_GC, ...} do                ▷ block-level
        for n ∈ oct do                              ▷ block-level
            for lev ∈ levels do
                for cel ∈ lev do                    ▷ thread-level
                    Solve unknowns using one thread
                end for
                Thread synchronizations within a block
            end for
        end for
    end for
    Record outgoing flux at reflective boundary
    Calculate flux moments by Eq. (2)
end for
Free memories for flux moments and angular flux
```

The non-symmetric linear system of Eq. (5) is solved by a multi-group BiCGSTAB method, derived from the standard BiCGSTAB method. This method handles energy group chunks one at a time, while the energy dependency over the group chunks is managed at a higher level from the highest-energy group chunk and proceeding to the lowest-energy group chunk. All the matrix and vector operations are implemented by the self-written GPU kernels. Especially, the $\boldsymbol{DL}^{-1}$ operation represented by Algorithm 1 is called twice by each BiCGSTAB iteration. The $L^2$ -norm of residual error vector and the $L^\infty$-norm of absolute error vector of solution are used for convergence check together.

A group chunk decomposition method is proposed and illustrated on dual-GPU systems using 14 energy groups with two energy group chunks, as presented in Fig. 1. This method partitions each group chunk into multiple subgroup chunks, with the number of subgroup chunks matching the number of GPU devices. The multi-group BiCGSTAB solver is performed over energy group chunks from high energy to low energy. The variables and Krylov subspace vectors for each energy group chunk are partitioned for subgroup chunks within the current group chunk. These data for different subgroup chunks are distributed on their corresponding GPU devices, enabling every energy group chunk utilizes all GPU devices for parallel computing. For this parallel scheme, a small amount of additional communication is required to calculate scattering sources using the flux moments across energy group or subgroup chunks. With the calculated total source, communication between GPUs is not required since the calculations among energy groups are independent in Algorithm 1. Therefore, the group chunk decomposition method is feasible to give a linear speedup in Algorithm 1. Importantly, the group chunk decomposition does not cause degradation in convergence.

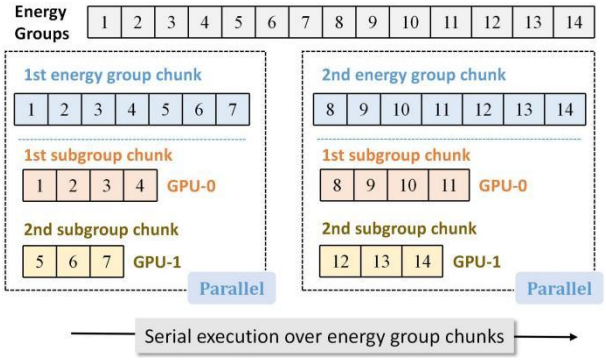


Fig. 1. Illustration of the group chunk decomposition method on dual-GPU systems using 14 energy groups with two energy group chunks.

## 4. Code parallel performance

In order to verify the new features implemented in STRAUM, a Kobayashi-like problem, derived from the Kobayashi benchmark problems which are designed for the verification of radiation transport codes, was simulated using STRAUM. The detailed parameters of the Kobayashi-like problem can be found in the reference [2]. It has an isotropic and uniform neutron source with 27-group neutron and 19-group gamma cross sections. The cross sections of the realistic materials were processed with the scattering anisotropy up to $P_3$ order but up-scattering was not considered. The 46 energy groups were partitioned into two group chunks for the multi-group BiCGSTAB solver: 27 neutron groups in the first group chunk and 19 gamma groups in the second group chunk. Three angular divisions, including 4, 16, and 24 angular directions per octant, were employed to test the performance of the multi-group Krylov subspace GPU solver on two machines equipped with RTX 3080 Ti and RTX 4090 GPUs, respectively. The block sizes of the transport kernel for all the three simulations are set as 128.

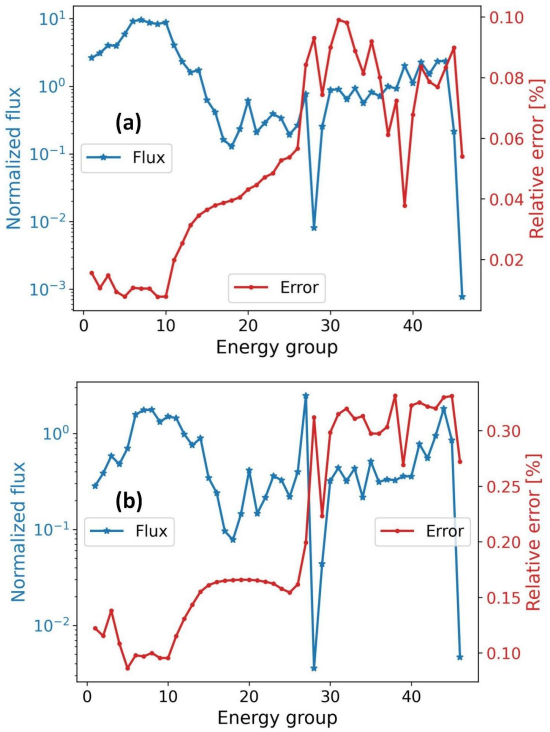


Fig. 2. Neutron spectra and their relative errors between the multi-group BiCGSTAB GPU solver and the within-group CPU solver in STRAUM at region 1 (a) and region 2 (b).

Fig. 2 presents the neutron spectra and their relative errors at region 1 using 4 angles per octant and at region 2 using 16 angles per octant. The reference results were obtained using the within-group CPU solver with serial double-precision computation, while the multi-group BiCGSTAB GPU solver was executed with single-precision computation. The flux errors mainly come from the different solving methods and the numerical calculation using $10^{-3}$ as the tolerance for $L^{\infty}$-norm of absolute flux errors. The maximum flux error for all energy groups is less than 0.4%, which is acceptable for radiation shielding analysis.

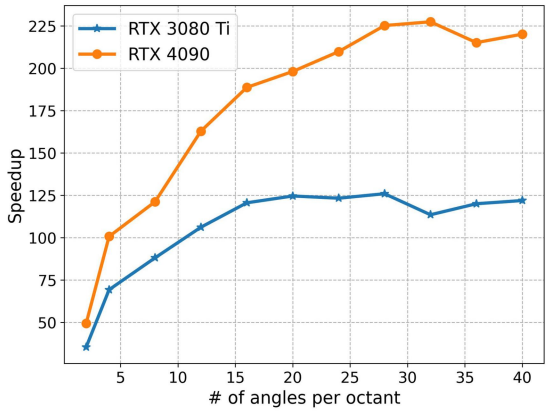


Fig. 3. Speedup for single-GPU systems using different angular divisions.

Fig. 3 illustrates the parallel speedup for the single-GPU systems as the number of angles per octant is scaled from 2 to 40. The results demonstrate nearly 225x and 125x speedups for the simulations on the RTX 4090 and 3080 Ti GPUs over the serial CPU simulation, respectively. With a fixed thread block size of 128, the total number of threads launched for the transport kernel increases as the number of angles per octant grows. Therefore, the advantage for GPU parallel scheme grows with the problem size as the angular quadrature order increases. For the two single-GPU systems, the maximum speedup is achieved when the number of threads issued is more than six times the number of streaming processors.

Table I: Parallel computing efficiency and memory reduction ratio for the dual-GPU simulations (46 energy groups in one group chunk).

| System | Parameters | Angles per octant | | |
|---|---|---|---|---|
| | | 4 | 16 | 24 |
| Single-GPU | Memory | 5.3 GB | 7.2 GB | 8.4 GB |
| | $T_{3080}$ | 12.7 s | 30.5 s | 43.6 s |
| | $T_{4090}$ | 7.33 s | 17.8 s | 26.2 s |
| Dual-GPU | $\xi_{mrr}$ | 40.3% | 42.5% | 45.8% |
| | $\xi_{pce}^{3080}$ | 54.7% | 82.7% | 91.4% |
| | $\xi_{pce}^{4090}$ | 56.7% | 87.8% | 97.8% |

$\xi_{mrr}$: memory reduction ratio on per-GPU.
$\xi_{pce}$: parallel computing efficiency.

The performance of the group chunk decomposition was tested on two dual-GPU systems. The Kobayashi-like problem with only one group chunk, containing all the 46 energy groups, was adopted to ensure that all the tests on one or two GPUs have same iteration times. As illustrated in Fig. 1, this configuration creates two subgroup chunks, each of which contains 23 energy groups and is executed on its corresponding GPU. For the simulation using 24 angles per octant, a large number of threads are launched for the transport kernel executed on dual GPUs, and the calculation time of kernels requiring GPU communication accounts for less than 2% of the total elapsed time. Therefore, the parallel computing efficiency of dual GPUs for this simulation achieves more than 90%, as showed in Table I. Importantly, the memory usage on per-GPU for the dual-GPU systems is reduced by more than 40% compared to single-GPU simulations, which enables STRAUM to simulate larger problems on multi-GPU systems.

## 5. Applied to shielding analysis

A typical PWR derived from the Korean Next Generation Reactor (KNGR) was used for shielding analysis using GPU-version STRAUM. Besides, the GPU-version STRAUM is verified by code-to-code comparisons with MCNP. The detailed geometry and

material parameters can be found in the references [2][4]. The external source in fuel assemblies were determined using their power distribution, while the Watt fission spectrum was used for all assemblies. Several material-wise 47-group cross sections with $P_3$ anisotropic scattering were generated using the open-source Monte Carlo code OpenMC [5] for STRAUM based on a highly simplified 1D model. For this PWR model, approximately 620,205 tetrahedral cells shown in Fig. 4 along with the Gauss-Chebyshev angular quadrature were used, employing 4 polar and 4 azimuthal angles per octant.

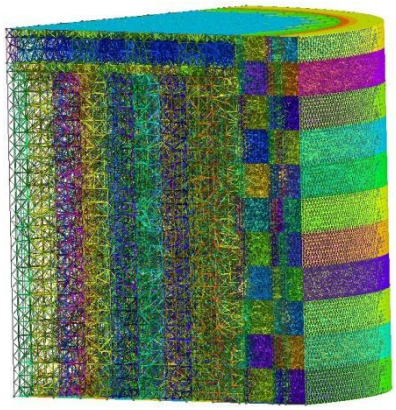

Fig. 4. Division the PWR model with 620,205 tetrahedral meshes for STRAUM simulation.

The reference flux results were obtained from MCNP with continuous-energy nuclear data by modeling the 3D reactor core. For this deep penetration problem, it is challenging for the Monte Carlo codes to tally high-energy group fluxes with low statistical uncertainties in the regions far from the external fixed source. To address this problem, the ADVANTG cod was used to generate energy and space-dependent source biasing and particle importance parameters for effective variance reductions in the MCNP simulation. As a result, all MCNP tallied group-wise fluxes in the reactor pressure vessel exhibit uncertainties less than 2% by simulating $5.0\times10^9$ particle histories.

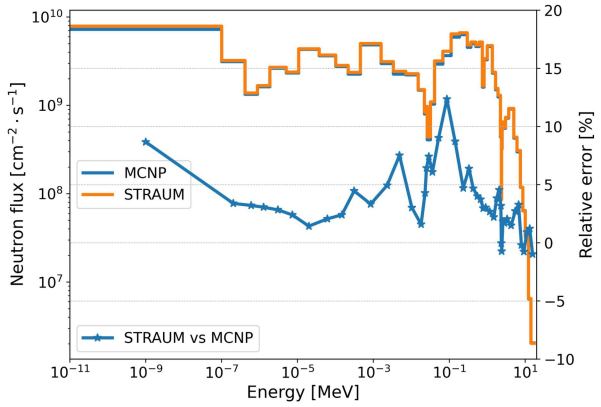


Fig. 5. Neutron energy spectra calculated by MCNP and STRAUM in the reactor pressure vessel.

The neutron spectra in the reactor pressure vessel, obtained from MCNP and STRAUM, are presented in Fig. 5. One can see that only one group flux at approximately 0.1 MeV has an error slightly greater than 10% and the lowest energy group has an error around 8.7%, while the flux errors for most of the thermal and fast groups are relatively small. These differences are acceptable because STRAUM utilizes 47-group cross sections while MCNP uses continuous-energy cross sections. It took 28 minutes for STRAUM on two RTX 4090 GPUs and 7.26 hours for MCNP on two Intel Xeon Gold 6226R CPUs, each of which contains 16 CPU cores.

## 6. Conclusions

A GPU parallel multi-group Krylov subspace solver has been developed and implemented in STRAUM. Compared to serial execution using a single CPU core, the GPU parallel scheme gives more than 200x speedup on a single RTX 4090 GPU, with a parallel computing efficiency of more than 90% on a dual-GPU system. For a practical PWR model, GPU-version STRAUM presents good accuracy and efficiency by code-to-code comparisons with MCNP.

## ACKNOWLEDGMENTS

This work was supported by the Institute for Korea Spent Nuclear Fuel (iKSNF) and Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government (Ministry of Trade, Industry and Energy (MOTIE)) (RS-2021-KP002656).

## REFERENCES

[1] Hong, S.G., 2013. Two Subcell Balance Methods for Solving the Multigroup Discrete Ordinates Transport Equation with Tetrahedral Meshes. Nuclear Science and Engineering 173:2, 101-117.
[2] Woo, M., Hong S.G., 2022. STRAUM-MATXST: A code system for multi-group neutron-gamma coupled transport calculation with unstructured tetrahedral meshes. Nuclear Engineering and Technology 54, 4280-4295.
[3] Jeong, S., Hong, S.G., 2022. Angular Direction Parallelization on STRAUM code using OpenMP. Transactions of the Korean Nuclear Society Autumn Meeting Changwon, Korea, October 20-21.
[4] Kim, J. K., Shin, C. H., Park, S. H., et al., 2001. An Evaluation of Korean Next Generation Reactor Pressure Vessel Neutron Fluence by Monte Carlo Simulations. International Associations for Structural Mechanics in Reactor Technology, Washington, DC, USA.
[5] Romano, P. K., Horelik, N. E., Herman, B. R., et al., 2015. OpenMC: A state-of-the-art Monte Carlo code for research and development. Annals of Nuclear Energy, 82, 90-97.