# Feasibility of Fast Pinwise Core Simulation Using GPUs

Seoyoon Jeon, Namjae Choi, and Han Gyu Joo*
*Department of Nuclear Engineering, Seoul National University,*
*1 Gwanak-ro, Gwanak-gu, Seoul 08826, Korea*
**Corresponding author: joohan@snu.ac.kr*

## 1. Introduction

Assembly-wise two-step core calculation using nodal methods has been adopted as the primary core design method due to its low computing cost compared to the direct whole-core calculation (DWCC). It compromises accuracy for computing time. Nonetheless the solution accuracy is regarded sufficient for the practical applications in the industry because the predictions determined with bias and uncertainties are within the allowed range around the measured values.

However, the need for obtaining detailed pin-level solutions is continuously increasing due to the tightened safety regulations and it becomes more difficult to meet the design criteria with the conventional two-step method. Yet none of the high-fidelity core analysis codes employing DWCC is able to meet the computing cost requirements for the nuclear design and analyses which involves repeated core calculations for fuel loading pattern search and design data generation. In this regard, pinwise two-step calculations are considered as a compromise for DWCCs. However, even the core calculation using the pin-homogenized multigroup (MG) cross sections (XSs) is still computationally demanding to be executed practically on personal computers (PCs), thus a proper measure to reduce the computing time should be taken.

In this regard, we decided to explore the use of GPUs for the pinwise nodal calculation. Modern consumer-grade GPUs equipped in PCs come out with significant computing power, and we already have successfully applied GPU acceleration to the direct whole core calculation code nTRACER [1] and the Monte Carlo code PRAGMA [2] that achieved remarkable speedups. Such experiences, along with the preliminary works of SCOPE2 [3] on GPU acceleration, gave us the confidence of the feasibility of accelerating the pinwise nodal calculation with GPUs.

We are developing VANGARD (Versatile Advanced Neutronics code for GPU-Accelerated Reactor Designs) which is a GPU-based pinwise nodal design code, and this work is the initial phase of the development. This paper will introduce the nodal algorithm in VANGARD and the principles of GPU acceleration, along with some preliminary results obtained with a benchmark problem.

## 2. GPU-based Pinwise Nodal Calculation

VANGARD employs Simplified $P_3$ ($SP_3$) form of the source expansion nodal method (SENM) [4, 5]. SENM was chosen as the primary nodal kernel in VANGARD in that the existence of hyperbolic terms helps capturing severe flux gradients in the pinwise multi-group core calculation. In this section, the formulation of SENM is briefly explained and the strategies of GPU acceleration are described. Detailed derivations and the definition of coefficients can be found in the references.

### *2.1 One-node SENM Kernel for Pinwise Calculation*

In the following, subscripts for groups and directions will be omitted for brevity. Transverse-integrated one-dimensional within-group neutron diffusion equation is written as follows:

$$-\frac{4D}{h^2}\frac{d^2}{d\xi^2}\phi(\xi)+\Sigma_r\phi(\xi)=Q(\xi). \tag{1}$$

In SENM, as the name implies, the source term $Q$ is expanded up to 4th order using Legendre polynomials as:

$$Q(\xi)=\frac{\chi}{k_{eff}}\psi(\xi)+S(\xi)-L(\xi)\cong\sum_{i=0}^{4}q_iP_i(\xi). \tag{2}$$

Plugging Eq. (2) in Eq. (1) and analytically solving for the flux yields the following solution:

$$\phi(\xi)=A\sinh(\kappa\xi)+B\cosh(\kappa\xi)+\sum_{i=0}^{4}c_iP_i(\xi) \tag{3}$$

where

$$\begin{gathered}\kappa=\frac{h}{2}\sqrt{\frac{\Sigma_r}{D}},\\ c_1=\frac{1}{\Sigma_r}\left(q_1+\frac{15}{\kappa^2}q_3\right),\ c_2=\frac{1}{\Sigma_r}\left(q_2+\frac{35}{\kappa^2}q_4\right),\\ c_3=q_3/\Sigma_r,\ c_4=q_4/\Sigma_r.\end{gathered} \tag{4}$$

The flux term in Eq. (3) is also expanded using the Legendre polynomials as follows:

$$\phi(\xi)=\bar{\phi}+\sum_{i=1}^{4}a_iP_i(\xi) \tag{5}$$

where

$$\begin{aligned}a_1&=c_1+\frac{3}{\kappa}\left(\cosh(\kappa)-\frac{\sinh(\kappa)}{\kappa}\right)A,\\ a_2&=c_2-5\left(\frac{3\cosh(\kappa)}{\kappa^2}-\left(1+\frac{3}{\kappa^2}\right)\frac{\sinh(\kappa)}{\kappa}\right)B,\\ a_3&=c_3+\frac{7}{\kappa}\left(\left(1+\frac{15}{\kappa^2}\right)\cosh(\kappa)-\left(6+\frac{15}{\kappa^2}\right)\frac{\sinh(\kappa)}{\kappa}\right)A,\\ a_4&=c_4-9\left(\frac{5}{\kappa^2}\left(2+\frac{21}{\kappa^2}\right)\cosh(\kappa)-\left(1+\frac{45}{\kappa^2}+\frac{105}{\kappa^4}\right)\frac{\sinh(\kappa)}{\kappa}\right)B.\end{aligned} \tag{6}$$

Once the coefficients are obtained, node average flux is calculated as follows:

$$\bar{\phi} = \frac{\bar{s} + \sum_{u=x,y,z} \left( \Sigma_D^u \left( \begin{array}{l} \frac{1}{2}\tau^u \left( J_u^{l-} + J_u^{r-} \right) \\ +\left(3 - \tau^u(\mu + 3\beta^u)\right)c_2^u \\ +(10 - \tau^u(\mu + 10\beta^u))c_4^u \end{array} \right) \right)}{\Sigma_r + \sum_{u=x,y,z} \Sigma_D^u \mu \tau^u}. \quad (7)$$

With the updated node average flux, outgoing current can be determined as follows:

$$\begin{aligned} J^{r+} &= \mu\phi\ (1) + \frac{1}{2} J\ (1) \\ &= \alpha^s A\ + \alpha^c B\ + \mu(\bar{\phi}\ + \phi^P(1)) + \frac{1}{2} J^P(1), \\ J^{l+} &= \mu\phi\ (-1) + \frac{1}{2} J\ (-1) \\ &= -\alpha^s A\ + \alpha^c B + \mu(\bar{\phi}\ + \phi^P(-1)) + \frac{1}{2} J^P(-1). \end{aligned} \quad (8)$$

The SP$_3$ SENM equations are defined for the summed flux and the 2nd flux moment. The solution procedure is analogous to the diffusion case except for some changes in the definitions of coefficients: the value of $\mu$ is set to 1/4 and 7/16 for the summed flux and the 2nd moment equations, respectively, and diffusion coefficients and removal cross sections are defined differently for the 2nd moment equation. The SP$_3$ SENM equations are given as follows:

$$\begin{aligned} -\frac{4D_0}{h^2}\frac{d^2}{d\xi^2}\hat{\phi}_0 + \Sigma_{r0,g}\hat{\phi}_0 = Q_0, \\ -\frac{4D_2}{h^2}\frac{d^2}{d\xi^2}\phi_2 + \Sigma_{r2,g}\phi_2 = Q_2 \end{aligned} \quad (9)$$

where

$$\begin{aligned} &\hat{\phi}_0 = \phi_0 + 2\phi_2, \\ &D_0 = \frac{1}{3\Sigma_{tr}}, \quad D_2 = \frac{3}{7\Sigma_t}, \\ &\Sigma_{r0} = \Sigma_r, \quad \Sigma_{r2} = \frac{4}{3}\Sigma_r + \frac{5}{3}\Sigma_t. \end{aligned} \quad (10)$$

The source terms are also defined separately for the summed flux and the 2nd moment equations:

$$\begin{aligned} Q_0 &= \bar{s} + 2\Sigma_r\phi_2 - L_0, \\ Q_2 &= -\frac{2}{3}\bar{s} + \frac{2}{3}\Sigma_r\hat{\phi}_0 - L_2. \end{aligned} \quad (11)$$

The SENM kernel can be derived up to the 4th order, but for the solution of radial direction where the mesh size is small, the expansion is truncated at the 2nd order. Applying the 4th order expansion for the pin-sized mesh is an overkill and induces stability issues [5]. On the other hand, axial meshes are thick and the full 4th order kernel is used in the axial solution. That is, kernels of different orders are used in hybrid for efficient three-dimensional pinwise nodal core calculation.

## *2.2 GPU Acceleration*

### 2.2.1 Suitability of SENM for GPU Acceleration

GPU is a subset of vector processor and is specialized at SIMD (Single Instruction Multiple Data) parallelism; namely, adjacent threads should perform same operation on coalesced data. A GPU contains substantial number of simple arithmetic cores, which can deliver significant floating point computing power. However, it is bounded by the performance of the relatively slow main memory, so a GPU contains several types of small but fast local memories, such as register, to buffer the main memory accesses. Therefore, the number of operations per each main memory access (= operational intensity) should be maximized to have high performance. To summarize, an algorithm should satisfy the following properties to be efficiently accelerated on GPUs:

1. Branchless (single instruction)
2. Contiguousness of data access (coalescing)
3. Exploitation of local variables (local memory)
4. Compute-intensive (operational intensity)

The properties of the nodal method make it suitable for the GPU acceleration. First of all, the data structure and the iteration scheme of the nodal method are highly regular; each cell is solved independently with the same algorithm and the cells and the energy groups are fully contiguous, which enables branchless calculations and coalesced memory accesses.

Second, the nodal method involves a lot of arithmetic operations. In SENM, calculation of coefficients such as $\kappa$, $\beta$, $\Sigma_D$, and $c_i$ is computationally intense. However, these variables are declared locally and are likely to be stored in registers. That is, the nodal kernels have high operational intensity and can exploit the local memories.

The frequent exponential (hyperbolic) calculation in the SENM kernels is also where GPUs have strengths. GPUs have hardware-level SFUs (special function unit) and corresponding fast math functions working in single precision, which provides fast approximate estimations of some special math functions including exponentials. That is, SENM can take more advantage from the GPU acceleration than other nodal methods.

### 2.2.2 Parallelization and Precision Issues

Currently, all the routines of the nodal solver, which serves as the largest computational hotspot, had been ported to GPU. Parallelization is applied for cells and energy groups; a Jacobi scheme is applied in energy and cells are arranged in red-black ordering so that each thread can take an energy group of a cell.

Additionally, the homogenization routine for CMFD acceleration had been ported to GPU so far. From the profiling result after the first porting of the nodal solver, it was observed that the time portion of the data copy between host and device was not negligible, reaching

40% of the total nodal solution time. It then turned out that copying out the outgoing current array was causing unnecessary overheads. Since the outgoing current is only used to determine $\hat{D}$ in the CMFD acceleration, it was worthwhile to remove the copy of outgoing currents by simply calculating the CMFD coupling coefficients directly on GPU. Note, however, that eventually all the routines of CMFD will be ported to GPU as well; due to the cumbersomeness of constructing the linear systems in sparse format in parallel on GPU, and considering the small time portion of assembly-wise CMFD, the CMFD acceleration is temporarily being performed on CPU.

Since VANGARD targets to be applied in PC-level, employing consumer-grade GPUs is of interest. The key characteristics of the consumer-grade GPUs is that they are specialized for single precision (FP32) arithmetic, as graphics visualization does not require high precisions. As the result, consumer-grade GPUs have only minimal support for double precision (FP64) arithmetic.

Therefore, single precision is utilized in every places of the GPU nodal solver. However, this raises a stability issue caused by the hyperbolic functions. The term $\kappa$ inserted in the hyperbolic functions is proportional to the mesh size. In the axial direction where the mesh sizes are relatively thick, the hyperbolic function values have chance to grow extremely large to the range which single precision cannot cover properly. As the result, a cascade effect occurs in the coefficient calculations by the numerical errors and makes the kernel diverge.

To resolve this, a clever mixed precision scheme is employed. For the radial 2nd order kernels which do not suffer from numerical errors, single precision is used. On the other hand, precisions are mixed in the axial 4th order kernel. For the hyperbolic functions whose double precision versions are computationally expensive, single precision is used. But for the other arithmetic including coefficient calculations, double precision is used. In this way, the performance penalty of using double precision can be minimized while retaining accuracy and stability. **Table 1** summarizes the selection of kernel expansion orders and precisions in each direction.

**Table 1.** Order of kernels and precisions in each direction.

| Direction | Radial | Axial |
|---|---|---|
| Expansion Order | 2 | 4 |
| Arithmetic Precision | FP32 | FP64 |
| Hyperbolic Precision | FP32 | FP32 |

## 3. Results

To verify the GPU acceleration capability, NEACRP benchmark problems [6] were analyzed with full-core configurations and the results were compared with the CPU solvers. Five cases including ARO case with T/H feedback were analyzed. In this section, consistency of the GPU routines with the CPU routines was examined and the performance of GPU routines was investigated. In addition, sensitivity studies regarding the precisions were performed.

### *3.1 Accuracy Evaluation*

The soundness of the GPU routines was assessed by comparing the critical boron concentrations (CBC) and pin powers with the CPU results. **Table 2** summarizes the results. In all cases, CBCs matched exactly between CPU and GPU, and the pin power differences were also negligible, showing maximum relative error being lower than 0.005%. The peak errors occur at the peripheral pins adjacent to the reflector. Except for those locations, most pins have almost exact agreement in power, which is shown by the RMS values.

**Table 2.** Comparison of results between CPU and GPU.

| Case | CBC (ppm) | RMS (%) | Max. Rel. ΔP (%) |
|---|---|---|---|
| ARO | 1221.98 | 2.51E-5 | 3.67E-3 |
| A1 | 547.26 | 1.48E-4 | 3.29E-3 |
| A2 | 1168.89 | 2.41E-4 | 4.01E-3 |
| B1 | 1256.45 | 1.25E-4 | 2.81E-3 |
| B2 | 1197.04 | 3.63E-5 | 4.19E-3 |

### *3.2 Computing Time Analysis*

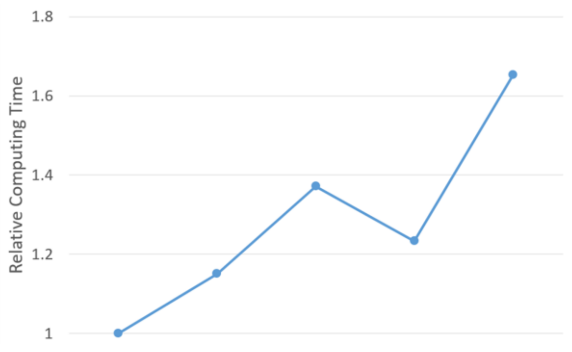
**Table 3** shows the computing resources used for the calculations. CPU calculation was run in parallel using OpenMP with two CPU processors containing 10 cores. GPU calculation was performed with a single consumer-grade GPU.

**Table 3.** Computing resources.

| Type | Processor |
|---|---|
| CPU | Intel Xeon E5-2630 v4 (2EA) |
| GPU | NVIDIA GeForce RTX 2080 Ti |

**Figure 1** illustrates the relative computing time of the nodal solvers with different nodal kernel precisions with respect to the full single precision calculation. Note that all the operations outside the nodal kernel (e.g. source calculation) is done with single precision. It is obvious that using full single precision has the best performance, but it sacrifices stability and is not considered optimal. Thus, the suggested mixed precision scheme, which is the fastest except for the full single precision case, is the optimal choice. Even though it loses the performance by 15%, it is still significantly faster than the blinded use of double precision, and this shows the effectiveness of the adaptive selection of precision.

With the optimal choice of precisions, the GPU nodal calculation time was compared to the CPU calculation using single core and 20 cores, which are summarized in **Table 4**. Compared to a single CPU core, a single GPU can achieve over 210 times of speedup, and the speedup is still 20 times compared to 20 CPU cores due to the limited parallel efficiency of CPU. This demonstrates that the GPU acceleration was very successful and the developed routines are highly efficient.



| Radial FP | 32 | 32 | 64 | 32 | 64 |
|---|---|---|---|---|---|
| Axial FP | 32 | 64 | 64 | 64 | 64 |
| Radial Hyperbolic FP | 32 | 32 | 32 | 32 | 64 |
| Axial Hyperbolic FP | 32 | 32 | 32 | 64 | 64 |

**Figure 1.** Effect of precisions on the nodal calculation time.

**Table 4.** Nodal calculation time comparison with CPU.

| Case | CPU (s) | | GPU (s) | Speedup | |
|---|---|---|---|---|---|
| | 1 Core | 20 Cores | | | |
| ARO | 1349.5 | 127.5 | 6.41 | 210.7 | 19.9 |
| A1 | 1103.9 | 103.1 | 5.19 | 212.7 | 19.9 |
| A2 | 1350.5 | 123.8 | 6.35 | 212.8 | 19.5 |
| B1 | 962.1 | 88.6 | 4.52 | 212.8 | 19.6 |
| B2 | 1268.8 | 116.4 | 5.94 | 213.5 | 19.6 |

**Figure 2** compares the computing time share of CPU and GPU calculations for the A2 case. The nodal solver, which tended to dominate the CPU calculation by taking 93% of the total computing time, is reduced to less than half of the total time by GPU acceleration. Still, the speedup in terms of total computing time is limited to 9.5 times due to the calculations still performed on CPUs, such as T/H feedback. It is thus expected that the total computing time will be further reduced by applying the GPU acceleration to the remaining calculations.

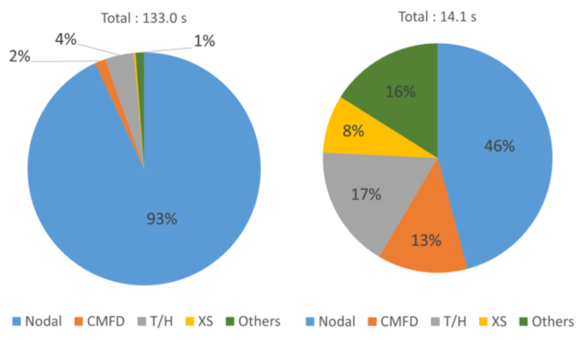


**Figure 2.** Computing time share comparison between CPU (left) and GPU (right).

## 4. Conclusions

In order to perform pinwise nodal core simulations at PC-level practically, GPU acceleration was applied to the pinwise nodal solver. The compute-intensive characteristics of the nodal method makes it suitable for the GPU acceleration.

In this work, pinwise one-node $SP_3$ SENM kernels were accelerated on GPU. A radial 2nd – axial 4th order hybrid expansion was used for efficient pinwise core simulation. A consumer-grade GPU was chosen as the acceleration platform considering that the code targets to be used in PC environments. The use of consumer-grade GPUs necessitated using single precision, which in turn caused stability issues induced by the numerical errors in treating the large hyperbolic terms in the axial kernel. This was resolved by adaptively using double precision in the axial kernel such that the performance regarding both stability and time cost is optimal.

The soundness and the performance of the developed GPU acceleration module were assessed by solving the NEACRP benchmark problems. The relative pin power errors between CPU and GPU solvers did not exceed 0.005% while a single GPU achieved 20 times speedup over 20 CPU cores in the nodal calculation.

Still, this work is preliminary and substantive amount of works need to be done. Especially, treating the group constants on GPU is necessary. Other calculations such as T/H feedback and CMFD acceleration should be also ported to GPU. Namely, more extensive application of GPU acceleration remains as the future work.

## REFERENCES

[1] N. Choi, J. Kang, H. G. Joo, “Preliminary Performance Assessment of GPU Acceleration Module in nTRACER,” Transactions of the Korean Nuclear Society Autumn Meeting, Yeosu, Korea, Oct. 25-26 (2018).
[2] N. Choi, K. M. Kim, H. G. Joo, “Initial Development of PRAGMA – A GPU-Based Continuous Energy Monte Carlo Code for Practical Applications” Transactions of the Korean Nuclear Society Autumn Meeting, Goyang, Korea, Oct. 24-25 (2019).
[3] Y. Kodama, M. Tatsumi, Y. Ohoka, “Study on GPU Computing for SCOPE2 with CUDA,” Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011), Rio de Janeiro, Brazil, May 8-12 (2011).
[4] J. I. Yoon, H. G. Joo, “Two-Level Coarse Mesh Finite Difference Formulation with Multigroup Source Expansion Nodal Kernels,” Journal of Nuclear Science and Technology 45(8), pp. 668-682 (2008).
[5] H. Hong, H. G. Joo, “Source Expansion Nodal Kernel for Multi-Group Pin-by-Pin SP3 Core Calculation,” Transactions of the Korean Nuclear Society Spring Meeting, Jeju, Korea, June 9-10 (2020).
[6] H. Finnemann, A. Galati, “NEACRP 3-D LWR CORE TRANSIENT BENCHMARK – Final Specifications,” NEACRP-L-335 (Revision 1), OECD Nuclear Energy Agency (1992).