공학박사 학위논문

Development of GPU-Based Deterministic and Probabilistic Direct Whole-core Calculation Systems

GPU 기반 결정론적 및 확률론적 직접 전노심 계산 체계 개발

2021년 2월

서울대학교 대학원

에너지시스템공학부

최 남 재

Development of GPU-Based Deterministic and Probabilistic Direct Whole-core Calculation Systems

GPU 기반 결정론적 및 확률론적 직접 전노심 계산 체계 개발

지도 교수 주 한 규

이 논문을 공학박사 학위논문으로 제출함
2021년 1월

서울대학교 대학원
에너지시스템공학부
최 남 재

최남재의 공학박사 학위논문을 인준함
2021년 1월

위 원 장 심 형 진 (인)
부위원장 주 한 규 (인)
위 원 이 재 진 (인)
위 원 이 덕 중 (인)
위 원 이 은 기 (인)

Abstract

The significant advances in CPU computing power through the past decades have enabled solving the entire reactor core directly with the transport methods, which is referred to as direct whole-core calculation. Since the direct whole-core calculation can provide high-fidelity and detailed solutions for the safety analysis of operating reactors as well as for the design of advanced reactors, there have been continued high demands for fast and accurate direct whole-core calculations. The CPU-based computing platforms are, however, not yet fast enough to employ the direct whole-core calculation methods in routine design analyses. Therefore, the legacy two-step methods are still used as the primary nuclear design tools in the industries.

Unfortunately, further improvements of the CPU-based computing platforms to make the direct whole-core calculations affordable in the industry is hard to expect because of the limitations imposed by power and memory barriers. Thus, a complete changeover to a new computing platforms is required to achieve a paradigm shift to utilize the direct whole-core calculation methods in routine core designs.

In this regard, this research develops GPU acceleration methods and frameworks for the 2D/1D and continuous-energy Monte Carlo (MC) methods, which are the representative deterministic and probabilistic direct whole-core calculation methods, and lays the foundations for the practical use of direct whole-core calculations. The recent rise of artificial intelligence (AI) and big data industries and enhancements of display resolutions are all causing tremendous amount of computing power demands in both sides of scientific computing and graphics processing, which is boosting the advances in GPU computing technologies. A single consumer-grade GPU is already as powerful as hundreds of server CPU cores, and cutting-edge supercomputers are relying on the high power efficiency of GPUs to achieve their target computational capacity. By taking advantage of this megatrend in the computing paradigm, this research aims to achieve not only time-wise but also cost-wise feasibility of the direct

whole-core calculation methods by exploiting consumer-grade GPUs for the GPU acceleration.

This research suggests algorithms and schemes for an efficient GPU acceleration of the 2D/1D and continuous-energy MC methods. However, this research does not end up as a mere collection of fragmentary algorithms, but integrates the algorithms to constitute complete solution frameworks. For this, this research is performed with production-grade codes as opposed to the previous researches which ended up with limited implementations on mock-up codes or proxy applications. The study on the GPU acceleration of 2D/1D method is performed with the nTRACER code, and the entire solution procedure of continuous-energy MC method is accelerated by GPUs through the development of the GPU-based continuous-energy MC code PRAGMA. Then, the effectiveness of developed algorithms is demonstrated by the applications to real engineering problems.

Specifically, the steady-state calculation module of nTRACER encompassing the planar MOC, CMFD, and axial solvers becomes the target of GPU acceleration in the 2D/1D method. MOC ray tracing and CMFD linear system solution schemes are optimized for massive parallelization, and CPU − GPU concurrency is exploited in the MOC calculation to take advantage of the heterogeneous computing environment. In the CMFD calculation, massively parallelizable DSPAI preconditioner substitutes for the LU-type preconditioners and iterative refinement technique is introduced to implement mixed precision arithmetic. In the axial solver, an axial MOC solver with improved parallel efficiency, accuracy, and stability is developed.

In the continuous-energy MC method, namely in the development of PRAGMA, optimization of cross section look-up and vectorization of random walk becomes the key of GPU acceleration. The unionized grid method is improved by a linear hashing scheme and nuclide-wise temperature-dependent grid collapse. A vectorized event-based tracking algorithm is developed, and the region partitioning and energy sort schemes are employed together with the event-based tracking algorithm to increase

the chance of memory coalescing significantly in the cross section look-up for the depleted fuel calculations.

Various schemes for the application of GPU-based continuous-energy MC method to real operating reactors are also developed. For an effective treatment of resonance scattering, the RST target velocity sampling scheme which resolves the drawback of DBRC and WCM is developed, and a domain decomposition scheme to realize large-scale power reactor calculations with limited GPU memory capacity is introduced. In addition, a unique scheme named MSC is employed for a practical MC depletion calculation, and the localized delta-tracking scheme can treat exactly the temperature distributions and material variations in the fuel pellets without additional cost, which enables efficient thermal feedback and depletion calculations. CMFD and ramp-up fission source convergence acceleration schemes are also introduced to improve the practicality of massive particle simulations.

The results of this research could demonstrate a high potential of the direct whole-core calculation methods as the practical nuclear design tools. A whole-core steady-state calculation employing the 2D/1D method could be performed in a few minutes, and whole-core massive particle MC simulations employing billions of particles has become a routine task that can be done in minutes. All these accomplishments were made on a practical computing cluster mounting dozens of consumer-grade GPUs. The performance of GPUs is still under an exponential growth and the practicality of developed frameworks will be continuously improved as the time goes.

Keywords: GPU Acceleration
Direct Whole-core Calculation
2D/1D Method
Continuous-energy Monte Carlo Method
PRAGMA

Student Number: 2016-29920

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1. Introduction

1.1 Overview and Motivation

Nuclear design has been traditionally performed based on the two-step methods. However, development of advanced reactors, deterioration of operating reactors, and tightening of safety regulations all gave rise to the demands on more precise but still fast simulation of reactors. As the result, direct application of transport methods to whole-core analysis and their coupling with the multi-physics solvers has been the trend of the reactor physics researches in the last decade. As the representative case, United States, whose average age of nuclear reactors reaches 40 years, invested 10 years and 2.5 billion dollars to the Consortium of Advanced Simulation of Light Water Reactors (CASL) led by the Oak Ridge National Laboratory (ORNL) and developed the Virtual Environment for Reactor Applications (VERA) [1] to renovate legacy reactor analysis methods and retain sustainability of their nuclear industries.

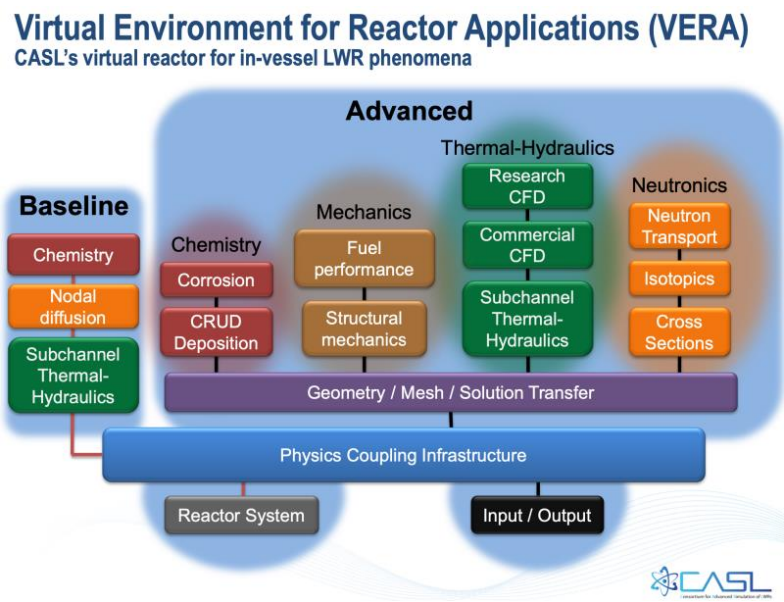


Figure 1.1 Constitution of the VERA code package.

However, the practicality of such direct whole-core calculation tools is challenged, because they strongly rely on high performance computing (HPC). VERA had been developed based on a leadership-class supercomputer Titan [2] and is targeted to be executed on clusters with thousands of CPU cores which the CASL team expect to become common in the industries within a few years.

In fact, it is natural that increasing the simulation fidelity comes with additional expenses of computational cost. The underlying expectation of developing the direct whole-core calculation codes is that the exponential increase of the computing power will naturally drag down the cost of direct whole-core calculations to a practical level. However, advances in the CPU processing power for the last decades have eventually hit the thermal barrier and are being challenged nowadays. As the result, the focus of CPU development is now shifting towards having more parallelism from having higher single-thread performance. Pat Gelsinger, who was used to be an executive of the Intel Corporation, stated in 2001:

"If current trends continue, you would have a processor with 1.8 billion transistors by 2010. You'd also have a heat generator with the intensity of a nuclear reactor."

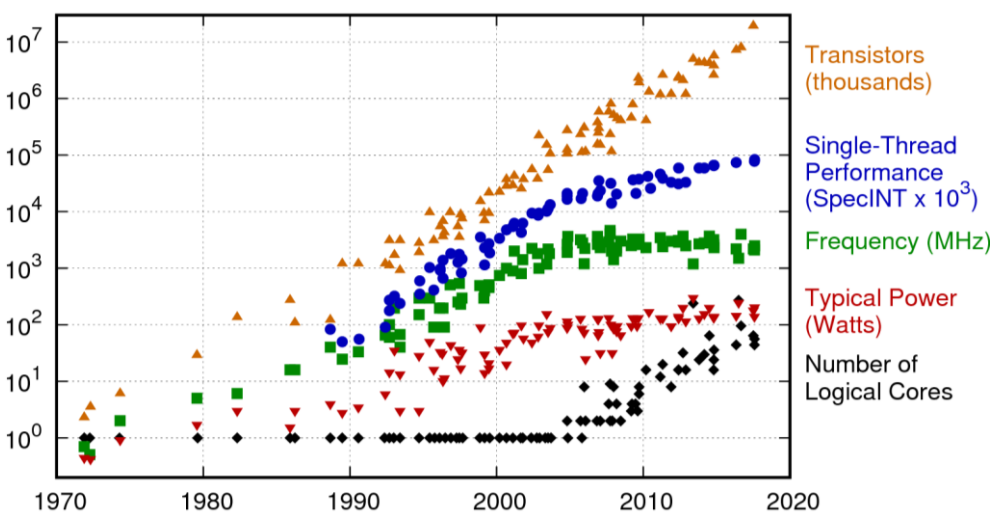


Figure 1.2 Improvement trend of CPU processing power [3].

Another obstacle of modern HPC is that the improvement of DRAM performance is much slower than the improvement of CPU processing power. In fact, CPUs are already fast enough if they can actually render their full performances. However, the performance of DRAM has improved by only ~ 7% per year, which is approximately twice improvement every seven years, while the performance of CPUs had doubled every two years by the Moore's law until relatively recently. As the result, modern CPUs are bound to the performance of DRAM rather than their own performances. Therefore, modern CPUs tend to mount larger cache memories made of SRAM to buffer the accesses to DRAM and try to elaborate caching mechanisms, but it has limitations.

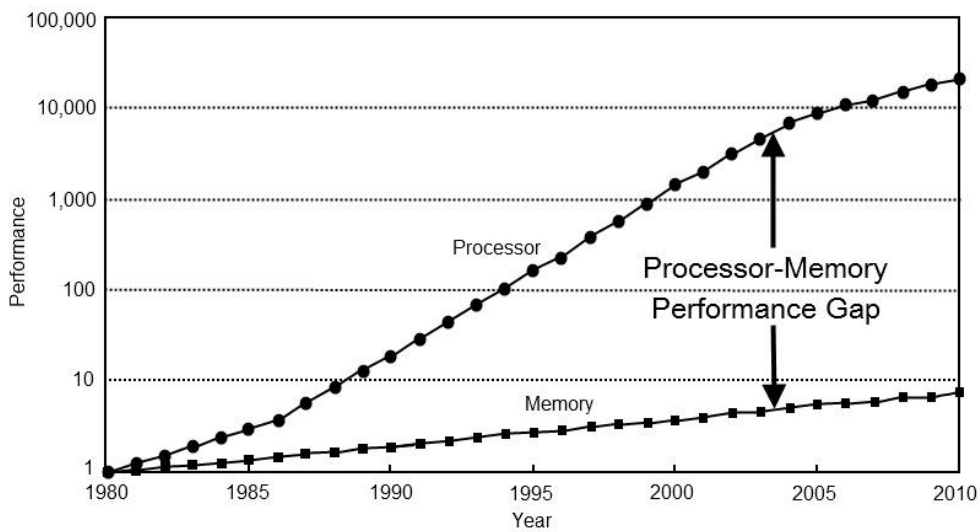


Figure 1.3 CPU – memory performance gap over time [4].

As the result, the expectations that the direct whole-core calculation methods will naturally substitute the legacy two-step methods in nuclear design with continuous reduction of processor costs will likely be a daydream, and the paradigm shift from the two-step approach to the direct whole-core calculation is difficult to be realized as far as the reactor physics codes remain in the conventional CPU platforms. That is, a complete changeover of reactor analysis methods to a new computing platform is required.

In this regard, we consider the graphics processing units (GPU) as a powerful and compelling alternative to realize practical direct whole-core calculations. Utilizing GPUs in computing is referred to as General-Purpose computing on GPUs (GPGPU). It is well-known that the artificial intelligence (AI) has become one of the greatest interest of this era which encompasses all the industrial fields. While the algorithmic innovations made by Geoffrey Hinton that resolved the divergence, overfitting, and vanishing gradient problems in neural networks had also contributed largely to the revival of AI research, the biggest contributor of the advent of AI era is the GPGPU technology which has enabled processing massive amount of data to train the neural networks.

GPGPU is the main driver of modern HPC, and it is getting increasingly popular in the scientific computing fields as well as in the AI researches. Increasingly many supercomputers are employing GPUs as the main computing resources, especially in the leadership-class supercomputers due to the superior power efficiency of GPUs. According to the TOP500 and Green500 lists [5], 6 out of top 10 supercomputers in performance and 20 out of top 25 supercomputers in power efficiency are mounting GPUs.

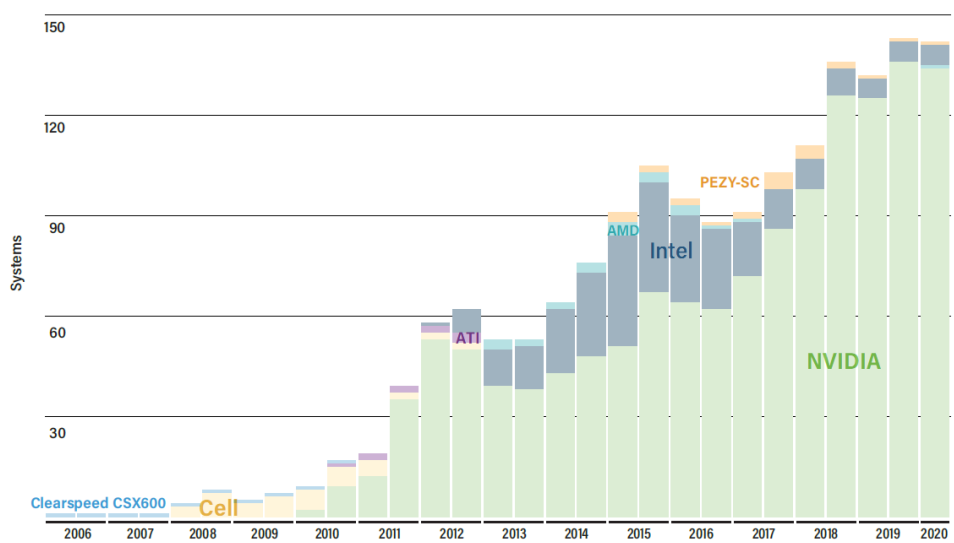


Figure 1.4 Trend of the number of TOP500 supercomputers mounting GPUs [5].

GPUs will continue to play a key role in the upcoming exascale computing era. The performance of GPUs is still scaling exponentially, and NVIDIA expects to have a GPU whose performance is equivalent to a thousand CPU cores by 2025. What is more compelling is the performance improvement of consumer-grade GPUs, which is being boosted by the enhancements of display technologies which are now moving from full HD to 4K or even 8K. That is, tremendous amount of computing power demands are being caused by graphics processing as well as computing. As the result, recent flagship consumer-grade GPUs are already as powerful as hundreds of CPU cores, and GPUs have literally become 'portable' supercomputers.

It opens up the possibility of performing direct whole-core calculations employing consumer-grade GPUs. Namely, consumer-grade GPUs which are cheap and readily obtainable in marketplaces can become the main computing resources to perform the direct whole-core calculations instead of the expensive server processors, which will make the direct whole-core calculations substantially more practical and affordable for academia and industries.

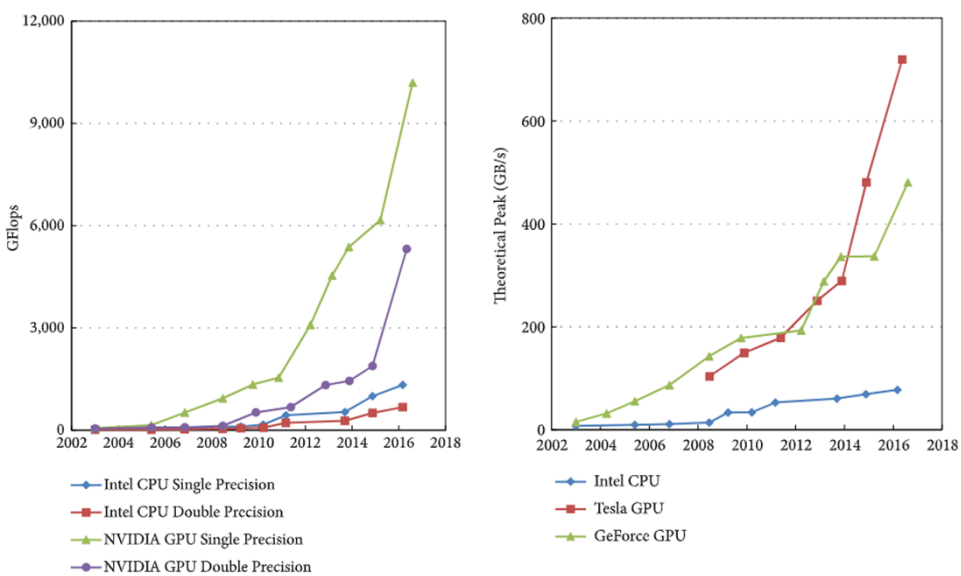


Figure 1.5 FLOPS (left) and memory bandwidth (right) improvement trends in time of CPU and GPU [6].

1.2 Literature Overview

While various approaches exist in the deterministic direct whole-core calculation, 2D/1D method is considered to be the most feasible approach among the currently available methods. The 2D/1D method was proposed contemporarily by the research groups from Korea Advanced Institute of Science and Technology (KAIST) [7] and Korea Atomic Energy Research Institute (KAERI) [8], and was put into practice in whole-core level for the first time by the DeCART [9] code. Since then, the method has been elaborated into various forms by many follow-up codes such as nTRACER [10], MPACT [11], NECP-X [12], and nTER [13], and has become the most famous direct whole-core calculation method.

The probabilistic direct whole-core calculation employs the Monte Carlo (MC) method. MC method is one of the fundamental computational methodology utilized ubiquitously in many disciplines of physics and mathematics. MC radiation transport has an extensive history beginning from 1946 when John von Neumann had first suggested its concept in Los Alamos National Laboratory (LANL). By compiling the decades of researches since then, the MCNP [14] code was first released in 1983 by LANL and has established the standard methodologies of MC radiation transport calculation. MCNP is still acknowledged as the standard of MC radiation transport codes. Beginning from 2010, more modernized MC radiation transport codes have been developed, including McCARD [15], OpenMC [16], Serpent [17], Shift [18], SuperMC [19], MCS [20] and such. Among the codes, OpenMC and Shift achieved massive parallelism employing more than 100,000 CPU cores. Especially, the Shift team demonstrated simulating a trillion particles for an AP1000 reactor within three hours using 240,000 CPU cores in the Titan supercomputer [21], which is considered to be a milestone of high performance computing application in reactor physics.

While the direct whole-core calculation methods have been established through decades of developments, however, application of GPGPU to the direct whole-core

calculations is still rudimentary. It is largely contributed by the fact that GPGPU is a relatively new technology. Current production direct whole-core calculation codes are the results of decades of man-years, but migrating the codes to GPUs will likely require rewriting many if not all parts of the codes. Furthermore, GPUs are not as versatile as CPUs, so dedicated tuning works are needed to achieve high performance. As the result, researches on the GPU acceleration of direct whole-core calculations are mostly performed with proxy applications or mock-up codes, as modifying the production codes for the GPU acceleration is cumbersome.

The first researches on the GPU acceleration of Method of Characteristics (MOC) neutron transport calculation were performed contemporarily by Boyd et al. [22] for the OpenMOC [23] code and by Zhang et al. [24] for the TCM code. Boyd focused on accelerating the 2D MOC calculation which is the primary workload of the 2D/1D method, while Zhang targeted to accelerate the 3D MOC calculation for generalized geometries. Following their works, Han et al. [25] developed a GPU-accelerated 2D MOC calculation algorithm for hexagonal geometries. The main difference between the works of Boyd and Han is the sweep algorithm; Boyd employed the Jacobi sweep algorithm while Han adopted the Gauss-Seidel sweep algorithm. Later, Song et al. [26] had performed parametric studies in detail on the GPU acceleration of 2D MOC calculation and concluded that the Jacobi sweep algorithm is more suitable for GPUs. They also developed a CPU/GPU hybrid MOC calculation scheme which assigns the ray tracing workloads dynamically to different types of processors [27]. While the works aforementioned are focusing on applications, Tramm et al. [28] performed an in-depth performance analysis of the 3D MOC calculation on cutting-edge HPC architectures including GPUs using a proxy application SimpleMOC. It is intended to provide a foundational understanding of the performance of 3D MOC calculations on HPC architectures and offer guidelines for algorithmic and architecture choices.

While a number of researches had presented GPU-based MOC calculations and demonstrated their high potential, however, little work has been done to incorporate

them into the 2D/1D calculation framework or into the production codes. Only one research by Song et al. [29] on the GPU porting of the Coarse Mesh Finite Difference (CMFD) acceleration and paring with the GPU-based MOC calculation can be found, and none on the 3D extension.

For the MC method, the first literature that discussed the GPU acceleration of MC neutron transport can be found in 2009 by Nelson [30]. However, the history of vectorizing MC neutron transport goes up to 1980s when vector processing became popular in the contemporary supercomputing arena. Brown and Martin [31] had first introduced the concept of event-based neutron tracking, and the continuous-energy MC code MVP of Japan Atomic Energy Research Institute (JAERI) was developed targeting the vector supercomputer [32]. At that time, their works quickly grew out of interest as the trend of supercomputing had moved from the vector processing to massive parallelization. However, with the recent rise of GPUs which are also vector processors, their works have started to be revisited.

Due to the complexity of continuous-energy MC calculation, early researches on the GPU acceleration of the MC method had either limited the scope to multi-group mock-up problems [30] [33] [34] [35] [36] [37] [38] or to a specific algorithm. For example, Liu et al. [39] performed GPU acceleration of the proxy XSBench which mimics the cross section look-up calculation of continuous-energy MC simulation. Brun et al. [40] developed a portable continuous-energy MC code PATMOS, which only offloads the cross section calculation to GPUs and performs tracking on CPUs. Sweezy [41] suggested a volumetric-ray-casting (VRC) estimator to offload the tally process of MCNP to GPUs. The VRC tally is performed on GPU concurrently with the main MCNP kernel on CPU.

In contrast to the 2D/1D method, however, the researches on the GPU acceleration of MC calculation are more progressive and are going production-grade. The WARP code developed by Bergmann and Vujic [42] had demonstrated a continuous-energy MC calculation in a general 3D geometry for the first time using GPUs. They utilized

the ray tracing engine NVIDIA OptiX [43] for the geometry calculations and made elaborate considerations for the stream compaction and sorting of random particles. Since the release of WARP, more full-featured GPU-based MC codes have started to be released. The GUARDYAN code developed by Molnar et al. [44] is a GPU-based MC code for a direct time-dependent simulation of research reactors, and Hamilton and Evans [45] developed a GPU acceleration module for the production MC code Shift. Recently, LLNL also started to port their Mercury and Imp codes [46] to the GPUs of the Sierra supercomputer [47].

However, no consensus has been made on what is the optimal way to implement continuous-energy MC method on GPUs. For instance, WARP and Shift had chosen to employ the event-based tracking algorithms for GPUs, while GUARDYAN claims that the conventional history-based tracking algorithm turned out to be more efficient than the event-based algorithm. It is due to the complexity of the continuous-energy MC calculation, which makes the performance strongly implementation-dependent. The features each code has are also very diverse. Therefore, still substantially more researches need to be performed by a variety of groups, so that sufficient amount of experiences by independent groups to systematically analyze various factors of the performance of continuous-energy MC calculation on GPUs and to draw a consensus can be accumulated.

1.3 Objectives and Scopes

This research aims to develop GPU acceleration algorithms for the 2D/1D method and the MC method for direct whole-core calculation. This research does not end up by merely suggesting individual algorithms, but integrates the algorithms to establish complete frameworks. Especially, this research focuses on retaining the practicality of the frameworks by employing consumer-grade GPUs so that it will facilitate the paradigm shift of the nuclear design from the legacy two-step calculation to the direct whole-core calculation.

This thesis contains two major chapters and one introductory chapter. Chapter 2 briefly overviews the architectures and optimization rules of CUDA GPUs and states the rationale of using consumer-grade GPUs for acceleration. The general principle of the mixed precision technique, which is the fundamental strategy running through the entire research and the key technique to exploit the consumer-grade GPUs, is introduced.

Chapter 3 is devoted to the GPU acceleration of 2D/1D method. Accelerating the steady-state module of the legacy direct whole-core calculation code nTRACER [10] with GPUs is the target of this research. This chapter presents, but not limited to, the following unique works of this research:

1. Utilization of CPU − GPU concurrency by task parallelism in MOC calculation for mixed precision arithmetic and calculation overlapping [48].
2. Development of GPU acceleration algorithm of P_L MOC calculation [48].
3. Introduction of Dropout Sparse Approximate Inverse (DSPAI) preconditioner for efficient parallelization of CMFD linear system solution [49].
4. Introduction of iterative refinement technique in the CMFD power iteration for mixed precision arithmetic [50].
5. Development of augmented axial MOC solver to enhance stability, accuracy, and parallel computing performance over the conventional axial nodal kernels [51].

6. Integration of the GPU-based MOC, CMFD, and axial kernels into a complete 2D/1D steady-state calculation framework [52].

Chapter 4 covers the development of the GPU-based continuous-energy MC code PRAGMA [53]. In contrast to nTRACER, PRAGMA is a newly developed code with the consideration of GPU acceleration from the very base. This chapter spells out the full implementation details of the PRAGMA code, including the uniquely developed algorithms of PRAGMA, in the following categories:

1. Cross section look-up methods: unionized grid method and hashing, treatment of temperature dependency, and optimization of data layout.
2. Vectorized tracking methods: event-based and modified history-based tracking algorithms, sorting schemes, and array-based massively parallel bank algorithm.
3. Relative Speed Tabulation (RST) method for target velocity sampling.
4. Domain decomposition algorithm [54].
5. Feedback calculation schemes.
6. Multilevel Spectral Collapse (MSC) scheme for depletion calculations [55].
7. Localized delta-tracking scheme for analytic fuel temperature profile treatment and efficient depletion calculation [56].
8. Hybrid CMFD and ramp-up acceleration for massive particle simulations [57].

Chapter 2. GPGPU Primer

GPU is a subset of vector processor and has different design characteristics with the conventional CPUs. In contrast to the CPUs which have sophisticated pipelines to perform complicated tasks, GPUs are specialized for a massive parallel processing of simple tasks, which requires elaborate tuning works for optimization. However, once a GPU application is optimized, it renders unbeatable performance and power efficiency. As the result, GPGPU has become the most rapidly growing computing technic in this era of big data as the primary tool to satisfy the tremendous computing power demands despite its difficulties in development and optimization.

For the GPGPU programming, developers are left with only two choices available: CUDA (Compute Unified Device Architecture) [58] and OpenCL (Open Computing Language) [59]. CUDA has been developed by NVIDIA and is designed exclusively for the use on their GPUs, while OpenCL is an open standard for a cross-platform heterogeneous computing maintained by the Khronos Group which aims at unifying the programming languages of all types of processors into a single standard.

We had chosen CUDA as the development framework for high productivity. While OpenCL claims to be cross-platform, its portability is still far from ideal in terms of performance. In fact, it is virtually impossible to have a single code for all kinds of computing architectures as they have very different characteristics. Furthermore, the current spread of the GPGPU technology is being led by the NVIDIA Corporation and they have a monopolistic position in the market. Therefore, we do not expect a significant disadvantage by limiting the target platform to the NVIDIA GPUs.

This chapter briefly explains the CUDA architecture and the rules for optimization. In addition, this chapter discusses the fundamental strategies of GPU acceleration that will run through the entire research, which is the use of consumer-grade GPUs and mixed precision arithmetic.

2.1 CUDA Architecture

In contrast to the CPU threads that have a single-level, one-dimensional mapping, the GPU threads have two-level hierarchy and three-dimensional mapping, as shown in Figure 2.1. A number of threads, not more than 1024, mapped in three dimensions constitute a thread block, which is again mapped in three dimensions. By combining the block and thread indices, maximum of six-dimensional indices can be used.

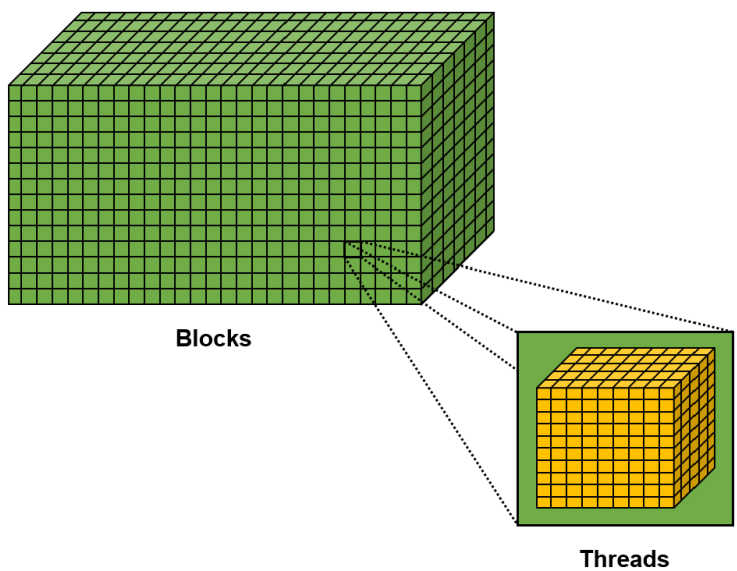


Figure 2.1 Illustration of three-dimensional thread blocks and threads.

The reason of having such two-level hierarchy has to do with the scheduling policy of the underlying hardware architectures. Figure 2.2 illustrates the overall layout of a CUDA GPU card. A CUDA GPU consists of multiple streaming multiprocessors (SM), each of which is equivalent to a CPU core in terms of functionality. Each SM processes a group of thread blocks which are scheduled by the GigaThread engine. Namely, the basic unit of scheduling is the thread block, not the individual threads. A thread block is terminated only when all the threads in the block are complete.

Each SM processes threads of a thread block in a vectorized manner. Figure 2.3 illustrates the structure of an SM. The smallest unit of an SM is a CUDA core which

is merely an arithmetic logic unit (ALU) that executes primitive arithmetic and logic instructions. Each SM contains 32 to 192 CUDA cores depending on the architecture generation, which can yield hundreds to thousands of CUDA cores for an entire GPU. The power of GPUs comes from such massive number of CUDA cores.

However, although each CUDA core can operate individually, it cannot process instructions on its own; each CUDA core can only execute the instructions, not fetch and decode them. Therefore, an instruction dispatch unit groups several CUDA cores and controls them collectively. As the result, every CUDA core in a group executes same type of instruction while the target data for operation are different. Such way of parallelization is referred to as single instruction multiple data (SIMD) parallelism. In CUDA, 32 threads form a warp and each warp is assigned to the group of CUDA cores. In other words, the threads in a warp should be organized such that the work items are SIMD-parallelizable.

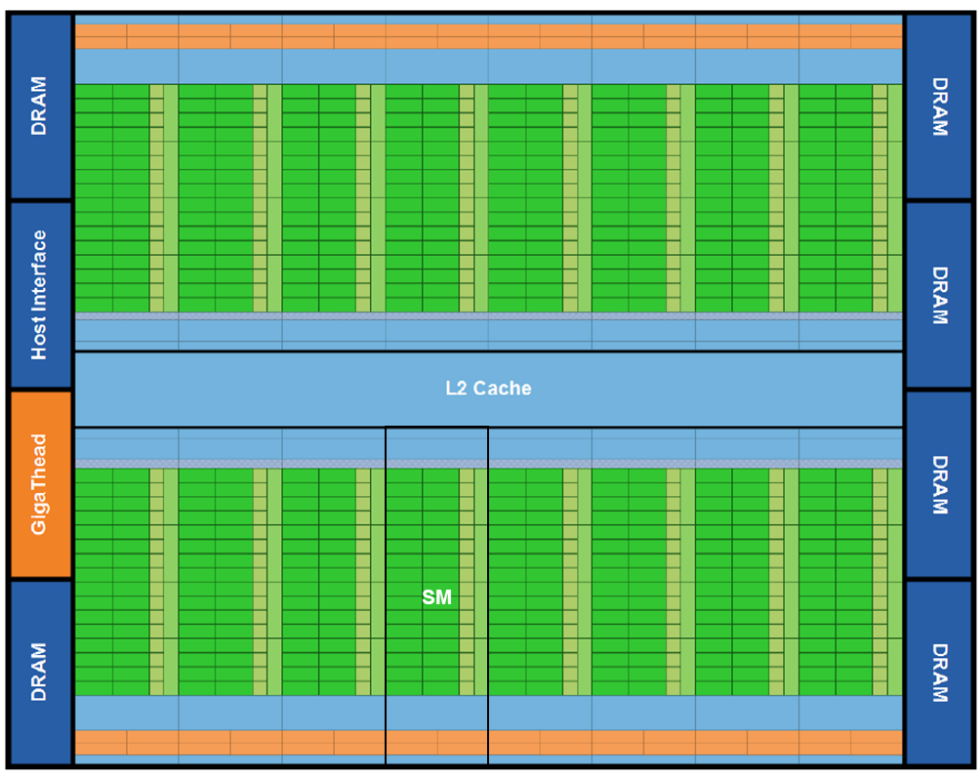


Figure 2.2 A Fermi architecture GPU card containing 16 SMs [60].

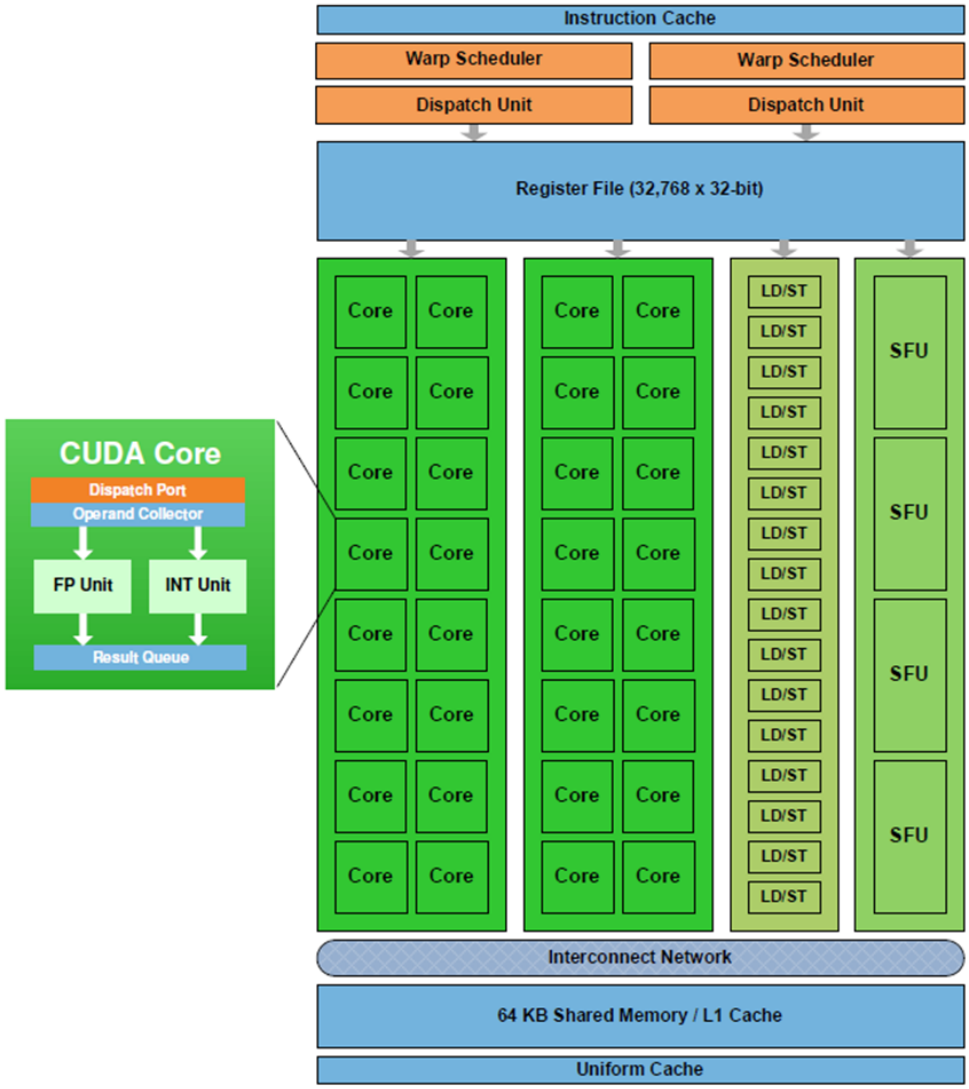


Figure 2.3 Streaming multiprocessor of the Fermi architecture [60].

Each SM allows to have much more threads than the number of cores. It is enabled by a flexible scheduling of warps, as shown in Figure 2.4. At every instruction cycle, the scheduler selects a warp which can be executed right away. As the result, a warp is not executed continuously; when the warp stalls due to some reasons such as the memory dependency, another warp is processed in that instruction cycle instead.

By dynamically switching the warps, such latencies can be effectively hidden and the resources can be kept busy. Thus, the thread occupancy which is the ratio of the

actual number of active threads in each SM to the maximum allowed threads is one of the important factors to be considered for optimization, as achieving higher thread occupancy will likely enable more latency hiding.

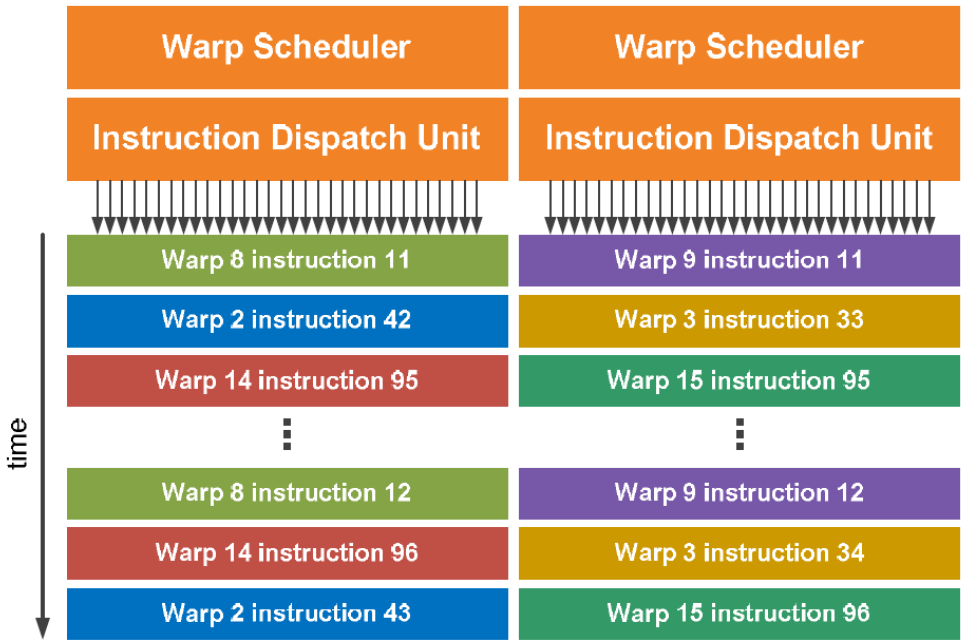


Figure 2.4 Schematic diagram of the scheduling of warps [60].

Memory hierarchy of GPUs which is illustrated in Figure 2.5 is also an important aspect to be considered in the GPU programming. In every SM, registers and caches are located closely to the operating units to reduce the memory access latencies. The fastest memory is register, which is used to store contexts and local variables. The local variables that cannot be stored in the registers are spilled to the local memory. L1 and read-only data caches, whose roles will be explained below, are also mounted SM-wise. Shared memory is a programmable cache memory; namely, the data in the shared memory can be managed by programmer. It is as fast as L1, and the data in the shared memory is block-private; it is shared by the threads in a thread block.

Outside the SMs, a shared L2 cache and the DRAM which is the main storage of a GPU are located. DRAM is reserved for storing global, local, constant, and texture

memories. While they are all stored in the same location, their usages and caching mechanisms are very different. Global memory is the heap memory of a GPU which can be dynamically allocated. The access to the global memory is cached to L2 only. Local memory is a compiler-managed memory; the spilled thread local variables are stored here. The addressing of the local memory is managed by the compiler which allows very efficient accesses. Furthermore, access to the local memory is cached to L1, and due to the compiler-managed addressing, spilled local variables are likely to be found in L1. Constant memory is a memory which stores global read-only data. It has a fixed size of 64KB regardless of the architectures and cannot be dynamically allocated; namely, arrays in the constant memory must be static whose size is known in the compile time. The data in the constant memory is read-only for GPU, but can be modified by CPU. Access to the constant memory is cached to the read-only data cache. Lastly, texture memory is another read-only memory specialized for the data which exhibits two-dimensional spatial locality such as images. Access to the texture memory is cached to the read-only data cache as well.

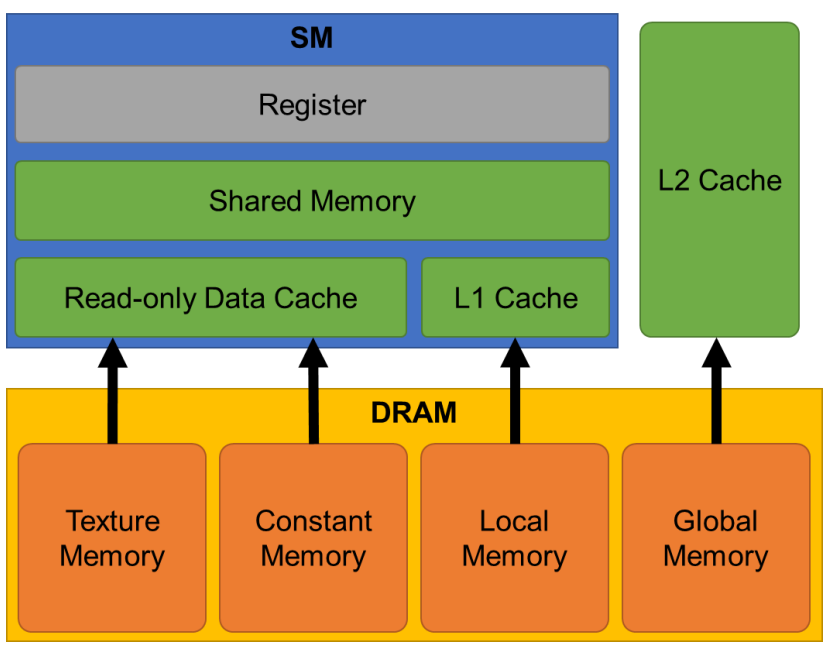


Figure 2.5 Illustration of CUDA memory hierarchy.

2.2 GPU Optimization Rules

In the performance of a GPU application, the memory optimization has a dominant impact. For example, GeForce RTX 2080 Ti, a flagship gaming GPU, mounts 4,352 single precision cores and the theoretical peak single precision performance is 11.75 TFLOPS. Assuming that each operation loads two 4-byte values and writes the result into a 4-byte value, data transaction of 141TB per second is required to achieve the peak FLOPS. However, the global memory bandwidth of the GPU is only 672GB/s, so even drawing 1% of the peak performance is difficult by merely using the global memory.

Of course there exist latencies in the execution of operations, so the peak FLOPS is only theoretical and cannot be achieved anyway. However, it is definite that high performance cannot be achieved with the global memory bandwidth. Therefore, the buffer memories aforementioned such as registers and cache memories should be exploited. Figure 2.6 illustrates the bandwidth of each memory type in the GeForce GTX 480 GPU, from which it can be seen that the registers have more than 60 times higher bandwidth than the global memory and L1 caches also deliver the bandwidth of a severalfold of the global memory. Although the figure describes a specific GPU model, this characteristic is common for all GPUs and only the values are different. Therefore, to increase the utilization of registers and L1 caches, a prefetch technique which migrates frequently used data from the global memory to the local variables or the shared memory should be actively utilized.

However, it should be noted that the use of buffer memories has a trade-off with the thread occupancy. The capacity of buffer memories in each SM is limited, so if each thread uses more buffer memories, there would be less active threads per SM. Having higher thread occupancy sometimes enhances the performance through the context switching mechanism, so the balance between the thread occupancy and the buffer memory usage per thread should be examined in programming.

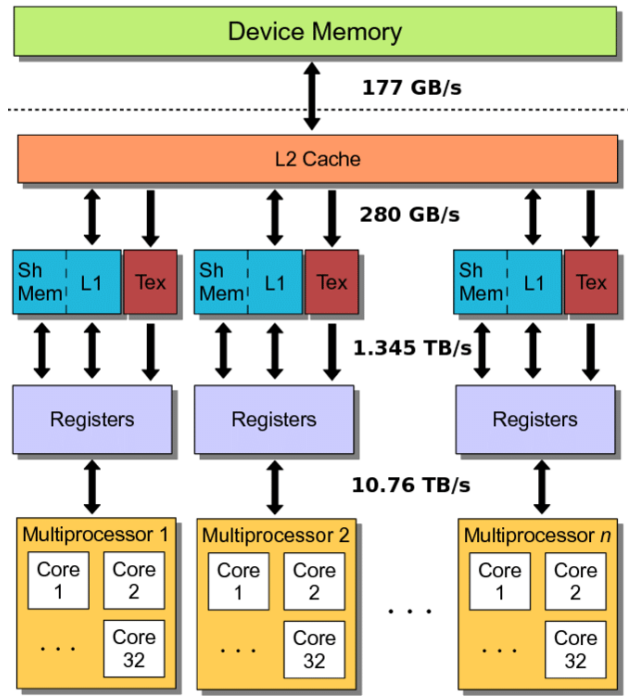


Figure 2.6 Memory hierarchy and bandwidth of each memory type in the GeForce GTX 480 GPU [61].

But unfortunately, not all data are prefetchable, and there are always cases when the global memory must be accessed. In this case the *memory coalescing* which is to make the threads in each warp access contiguous memory becomes important. Figure 2.7 shows schematically the difference of coalesced and strided memory accesses.

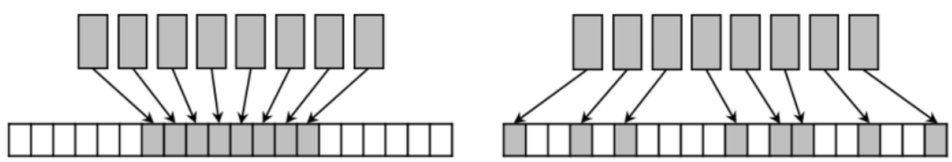


Figure 2.7 Schematic diagram of coalesced (left) and strided (right) memory accesses.

The reason why the memory coalescing is crucial is related to the cache memory policy. In CPUs, each core has its own L1 and L2 caches and therefore each thread does not have to care about the access patterns of adjacent threads. On the other hand, GPUs have shared caches and the memory read/write operations are executed warp-

wise; namely, multiple threads access to the caches simultaneously. All the accesses to the global memory goes through the L2 cache, whose cache lines are of 128-byte width. Each cache line is then composed of four 32-byte sectors, which becomes the unit of memory access. As the result, the cache memory on GPU is generally not for the temporal locality of accesses in each thread, but for the spatial locality of accesses over multiple threads.

Hence, the memory accesses in each warp should be arranged such that each cache line sector can serve as many threads as possible; namely, the memory accesses in a warp should be 'concentrated.' If all the threads in a warp are reading the same data, loading a single cache line sector will be able to serve all the requests. On the other hand, if all the threads in a warp are accessing the memory with 32-byte interval, 32 cache lines will have to be loaded and it will likely cause invalidation of cache lines; replacing each cache line requires 128-byte load from the global memory. Figure 2.8 schematically illustrates some example of memory access patterns and the cache line sectors being loaded.

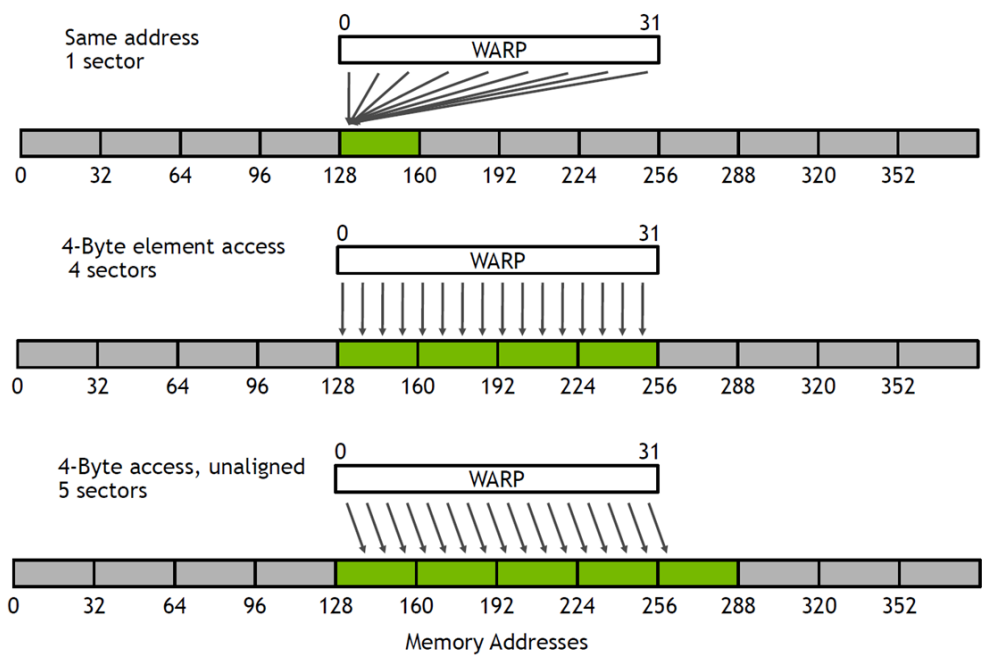


Figure 2.8 Example of memory access patterns [62].

Finally, there exists a very fundamental optimization rule which is vectorization; it is the working principle of GPUs. Due to the collective core control mechanism, a warp should be composed of threads performing SIMD work items. If branches exist in a warp, however, every branch is visited one at a time because only a single type of instruction can be performed at once, and the threads that do not participate in the branch remains idle, as illustrated in Figure 2.9. It is called branch divergence and should be avoided since it causes inactive cores during the execution. Branches can have several forms; the representative form of branches is the conditional statement, and the difference in the size of loops can be also a form of branches.

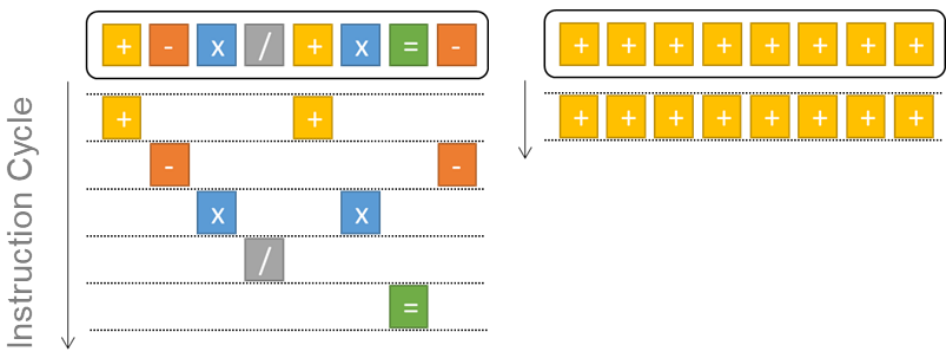


Figure 2.9 Illustration of the processing of branched (left) and vectorized (right) work items on GPU.

2.3 Strategies for the Research

As far as personal computers, workstations, and servers are considered, NVIDIA releases products in three lineups: GeForce, Tesla, and Quadro. Each lineup has its own purposes and therefore their characteristics are also different. The representative GPU products in each lineup and their specifications are shown in Table 2.1.

Table 2.1 List of representative NVIDIA GPUs in each lineup.

Lineup	GeForce	Tesla	Quadro
Name	RTX 2080 Ti	V100	RTX 8000
Generation	7.5 (Turing)	7.0 (Volta)	7.5 (Turing)
Picture			
FP32	11,750 GFLOPS	14,899 GFLOPS	16,300 GFLOPS
FP64	367 GFLOPS	7,450 GFLOPS	510 GFLOPS
Memory	11GB (GDDR6)	32GB (HBM2)	48GB (GDDR6)
Bandwidth	616GB/s	900GB/s	672GB/s
Power	250W	250W	295W
MSRP	$ 999	$ 10,000	$ 5,500

GeForce lineup is used for gaming and equipped to personal computers. Because graphics processing does not necessarily require high precision, GeForce GPUs only mount minimal number of double precision computing units while having substantial single precision computing capability. Furthermore, the size of the built-in memory is limited. However, their prices are relatively cheap and can be acquired readily.

Tesla lineup is specially designed for scientific computing and installed on servers or workstations. Tesla GPUs do not have display output and only have the function of computing. Instead, Tesla GPUs have desirable properties for scientific computing: 1. significant amount of double precision computing units, 2. large high-bandwidth memory to handle scientific data, and 3. fast inter-GPU communication capabilities

such as peer-to-peer (P2P) data transfer and GPUDirect RDMA. In addition, Tesla GPUs come out with the Error Correction Code (ECC) which prevents the memory corruption by the cosmic rays. Thus, Tesla GPUs are suitable for supercomputers or large data centers for which stability is crucial and which have more chances to be bombarded by the cosmic rays due to their volume. However, their prices are fairly expensive, which may be financially burdening for academia and industries to utilize.

Quadro lineup is suited to computer assisted design (CAD) and other professional visualization works, and it has an intermediate position of GeForce and Tesla lineups. While the Quadro GPUs share most of the features with the Tesla GPUs, they do not mount high-bandwidth memories and lack double precision computing power like the GeForce GPUs. The price range of Quadro GPUs is also located at the middle of GeForce and Tesla GPUs. Quadro GPUs are adequate for large-scale visualizations which should be done by multiple GPUs cooperatively.

If an unlimited budget is allowed, using professional GPUs such as the Tesla series would be beneficial. However, often academia and industries are constrained by the budget for the purchase and the maintenance of the servers. Thus, they will likely be left with the legacy computing platforms if using expensive professional GPUs is the only available choice for them.

Then, a question arises on whether we really have to use the professional GPUs for scientific computing. The underlying fact in this doubt is that many modern large-scale scientific calculations tend to pursue data-driven simulations and are mostly bound to memory bandwidth rather than computation. Namely, operations are simple while a lot of data are required to perform the operations. Certainly, little data reuse is available under these conditions. This is well-explained by the roofline model of Figure 2.10; unless certain degree of operational intensity (ridge point) is retained, the performance of the applications are limited by the memory bandwidth. However, the ridge point of GPUs is very high for ordinary scientific calculations, and only a few specific algorithms can reach the ridge point.

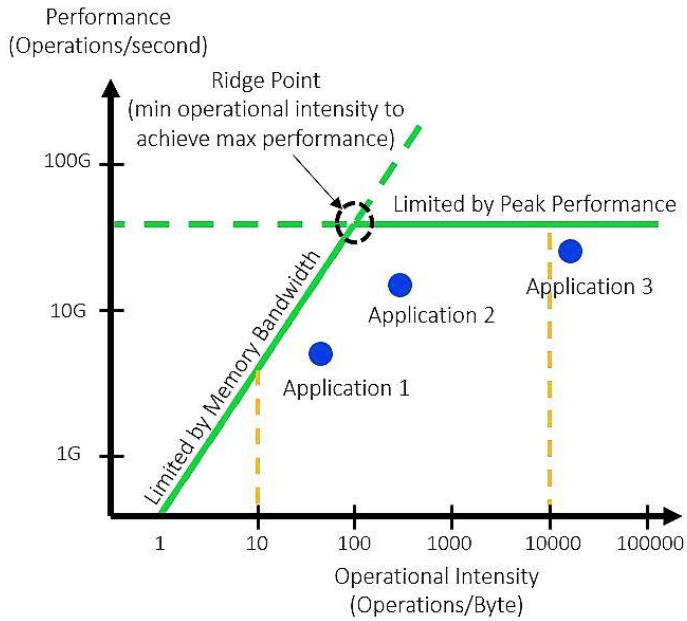


Figure 2.10 An example of the roofline model [63].

Of course, if an algorithm is compute-intensive and sensitive to the floating point precision, Tesla series would be the only choice. Also, if an algorithm is intensively performing data communications, Quadro or Tesla series would be more suitable as they provide efficient data communication functions.

However, if that is not the case, which is the case of direct whole-core calculation methods, professional GPUs are not worth their cost. First, as far as single precision is concerned, GeForce GPUs render comparable computing power with Tesla and Quadro GPUs. Second, GeForce GPUs have sufficiently large memory bandwidth; Tesla GPUs do have larger bandwidth than the GeForce GPUs, but not proportionally to the price gap. If an algorithm is bandwidth-bound, not much performance gain is expected by using Tesla GPUs instead of GeForce GPUs.

Shortage of memory in GeForce GPUs can be overcome by algorithmic measures such as domain decomposition or batch schemes. In addition, although it will not be utilized in this research, there exists a CUDA feature named unified memory which allows to create a virtual address space weaving all the memory devices in the node such that they can be seen as a single memory device, as illustrated in Figure 2.11.

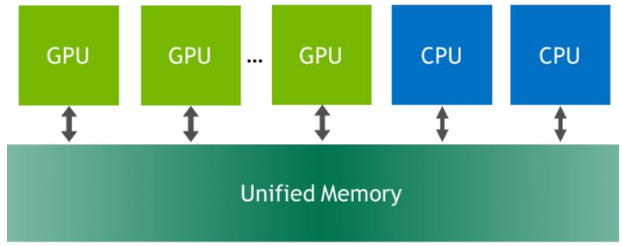


Figure 2.11 Concept of CUDA unified memory [64].

Lack of dedicated communication functions is not crucial unless the data transfer is heavily performed. Without the use of P2P or GPUDirect, the data transfer should be done via the host memory. However, the PCI bus has sufficiently large bandwidth, and the data transfer between host and GPU can be done relatively quickly. Once the data is on the host memory, high performance interconnects such as Infiniband can rapidly transfer the data. Furthermore, recent MPI implementations support CUDA-aware communication APIs which efficiently manage the underlying data copies, as shown in Figure 2.12.

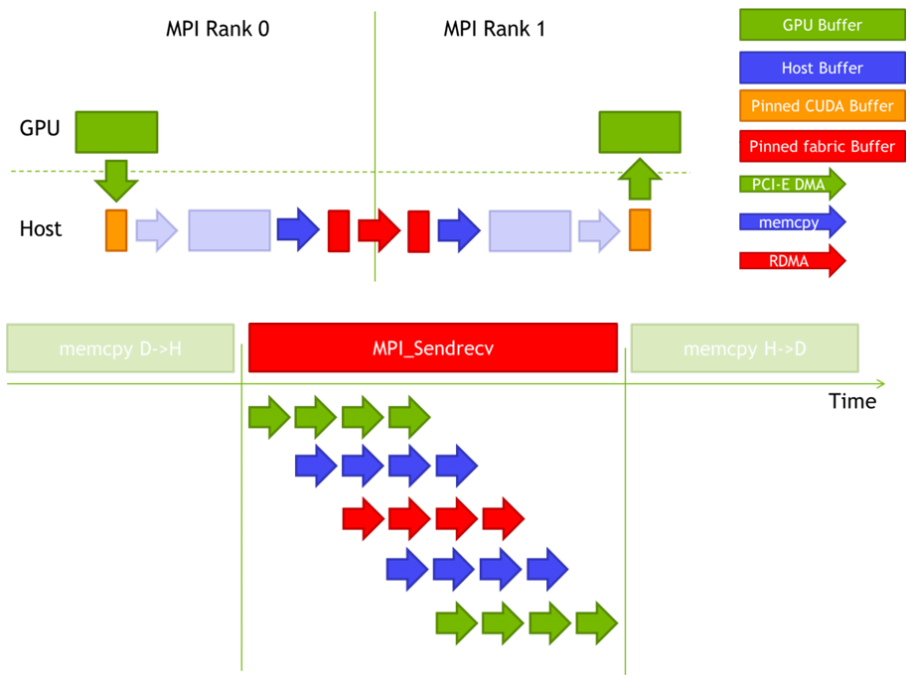


Figure 2.12 Schematic diagram of CUDA-aware MPI communication for GPUs without GPUDirect RDMA [65].

In this regard, the focus of this research is on exploiting the consumer-grade GPUs, namely the GeForce GPUs, to retain practicality, and the mixed precision technique becomes the fundamental strategy of GPU acceleration running through this research. Mixed precision technique is to maximize the use of single precision and restrict the use of double precision to the data or calculation that are critical to the accuracy so that the double precision accuracy is preserved. Utilizing single precision has several benefits not only for the consumer-grade GPUs but also for the professional GPUs:

1. In terms of memory throughput, twice more data can be loaded or stored with a given bandwidth and twice more data can be stored in registers and caches.
2. In terms of storage, twice more data can be stored with a given memory size.
3. In terms of floating point arithmetic, single precision cores can be exploited. Even for the Tesla GPUs, single precision computing power is twice larger than double precision.

The specific implementation of the mixed precision strategy will differ algorithm by algorithm, but the general principles are as follows:

1. Use single precision for read-only data and temporary variables.
2. For the read-only data that require preprocessing, operate with double precision on host, do typecasting, and store with single precision on device.
3. Use double precision for accumulating data.
4. Use double precision for precision-sensitive operations (e.g. division by a small value).

Chapter 3. 2D/1D Method

nTRACER [10] is one of the original 2D/1D direct whole-core calculation code. nTRACER claims to be a practical numerical reactor and has been developed to run on computing clusters with hundreds of CPU cores. nTRACER has served its main purpose as the practical numerical reactor decently for the last decade, demonstrating successes in several applications including YGN3C1 and UCN5C1 [10], BEAVRS two-cycle depletion [66], AP1000 [67], APR1400 [68], KRITZ and B&W 1810 [69], and SPERT III E-core [70].

However, nTRACER has gradually lost competitiveness in terms of performance by the development of newer codes adopting more scalable parallelization schemes. In addition, the amount of computing resources required by nTRACER, which was expected to become practical within a few years from the initial development of the nTRACER code, is still found impractical by the industries. Therefore, we conclude that for nTRACER to become a true practical numerical reactor and to reclaim the leadership, it has to embrace GPU computing.

In this regard, a 5-year government project to renovate nTRACER to a GPU-based code had initiated in November 2016 and runs through June 2021 (Grant No. NRF-2016M3C4A7952631). This chapter covers the achievements in the first stage of the project which is the GPU acceleration of steady-state module, including the planar MOC, CMFD, and axial solvers. The second stage which targets to accelerate a full cycle depletion calculation is under way.

This chapter consists of 5 sections except Section 3.1 which is merely an overview of the basic theories and methodologies of the 2D/1D method. Section 3.2 presents the GPU acceleration of planar MOC calculation. The conventional Gauss-Seidel sweep algorithm which performs spatial sweep at the innermost loop has little chance of vectorization due to the spatially random memory accesses and the divergence of

loop sizes. Therefore, the Jacobi sweep algorithm is adopted for the GPU-based ray tracing calculation, and detailed parallelization algorithms for P_0 and P_L calculations are explained.

Especially, CPU − GPU concurrency is utilized by a task-based parallelization of source update and ray tracing, which is enabled by a block energy decomposition of the Jacobi sweep algorithm. It was introduced mainly to overcome the limitation of working with nTRACER which is a legacy code containing more than 150,000 code lines. Namely, porting the entire code is extremely difficult and existing CPU-based routines − mostly cross section related routines − should be utilized. As the result, having certain degree of CPU dependency is indispensable, and the concurrency can hide the CPU overheads by overlapping calculations. We consider that utilizing the existing routines with the concurrency is a good strategy for offloading legacy codes to GPUs while avoiding the burden to refactor the codes. In addition, by leaving the memory-bound and precision-sensitive calculations to CPUs, spare double precision computing power of CPUs can be exploited to realize mixed precision arithmetic and plentiful main memory can be utilized to reduce the memory pressure on GPUs.

Section 3.3 explains the GPU acceleration of CMFD calculation. The conventional group major ordering which employs a group-wise downward solution does not lend itself to massive parallelization, and the LU-type preconditioners are inappropriate for GPUs. Thus, the node major ordering which solves the entire linear system as a whole is used to expose more parallelism and Dropout Sparse Approximate Inverse (DSPAI) preconditioner which is massively parallelizable is introduced. In addition, iterative refinement technique is introduced to realize mixed precision arithmetic in the CMFD power iteration, in which the linear system solution is performed in single precision while the rest of the power iteration is carried out in double precision.

Section 3.4 covers the newly developed axial solver named augmented axial MOC. It was discovered that the conventional axial SENM solver cannot retain accuracy, parallel efficiency, and stability at once. For the axial SENM solver to be sufficiently

accurate, SP_3 solution should be used. The SP_3 nodal calculation is only viable with one-node or whole-node kernel because two-node kernel will require an SP_3 CMFD solver. Since the one-node kernel is unstable, whole-node kernel remains as the only choice, which is not efficiently parallelizable.

The axial MOC solver was developed with the purpose to enhance the stability and accuracy of the axial solution as well as the parallel performance. After a profound study on the instability problem, we had concluded that the polynomial expansion of flux and the occurrence of negative flux are the main causes. In this regard, the axial MOC solver employs step characteristics and subgrid scheme with form functions instead of resorting to high-order spatial expansions, and selectively applies leakage splitting, limited transport correction, and P_L method in non-fuel regions to suppress the occurrence of negative flux. In addition, the axial MOC solver can be effectively parallelized with axial domain decomposition.

Section 3.5 integrates the individual solvers into the steady-state solution module and presents the global calculation scheme. The distributed memory parallelization of the GPU-based steady-state solution module employs a plane-wise decomposition in which each GPU is assigned to an MOC plane. The steady-state calculation flow is also explained. Then, performance analyses using the APR1400 [67] whole-core problem are performed in Section 3.6, which concludes the chapter.

3.1 Theory and Methodology

This section introduces the fundamental concept of 2D/1D method starting from the overview of governing equations, and explains the methods which constitute the 2D/1D solution, namely MOC and CMFD.

3.1.1 Governing Equations

Steady-state neutron transport equation is given as follows:

$$(\Omega \cdot \nabla + \Sigma_t(\vec{r}, E))\varphi(\vec{r}, E, \Omega) = Q(\vec{r}, E, \Omega) \tag{3.1}$$

where the source term Q is expressed as:

$$\begin{aligned} Q(\vec{r}, E, \Omega) = & \int dE' \int_{4\pi} d\Omega' \Sigma_s(\vec{r}, E' \to E, \Omega' \to \Omega)\varphi(\vec{r}, E, \Omega) \\ & + \frac{1}{k_{eff}} \frac{\chi(\vec{r}, E)}{4\pi} \int dE' \nu\Sigma_f(\vec{r}, E')\phi(\vec{r}, E'). \end{aligned} \tag{3.2}$$

In actual calculations, continuous treatment of energy is not available, so a multi-group approximation is made:

$$\int_{E_{g+1}}^{E_g} (3.1)\, dE \to (\Omega \cdot \nabla + \Sigma_{t,g}(\vec{r}))\varphi_g(\vec{r}, \Omega) = Q_g(\vec{r}, \Omega) \tag{3.3}$$

where E_g is the upper-bound energy of the energy group g.

For practical calculations, further approximation is introduced to Eq. (3.3), which is the diffusion approximation. In the diffusion approximation, angle dependencies are neglected and Eq. (3.3) is integrated over the angles:

$$\int_{4\pi} (3.3) d\Omega \to \nabla \cdot \int_{4\pi} d\Omega \Omega\, \varphi_g(\vec{r}, \Omega) + \Sigma_{t,g}(\vec{r})\phi_g(\vec{r}) = Q_g(\vec{r})\ . \tag{3.4}$$

Then, the Fick's law is introduced to express the current in terms of scalar flux:

$$\int_{4\pi} d\Omega\Omega\, \varphi_g(\Omega) = \mathbf{J}_g = -D_g \nabla \phi_g \,. \tag{3.5}$$

Substituting Eq. (3.5) into Eq. (3.4) yields the multi-group diffusion equation:

$$-D_g(\vec{r})\nabla^2 \phi_g + \Sigma_{t,g}(\vec{r})\phi_g(\vec{r}) = Q_g(\vec{r}) \,. \tag{3.6}$$

3.1.2 2D/1D Method

Solving the transport equation in Eq. (3.3) directly on a three-dimensional whole-core is not practical. Therefore, noting that a reactor has severe radial heterogeneity while the axial heterogeneity is relatively small, a clever approximation is made as depicted in Figure 3.1. This approximation converts the 3D transport equation to the combination of two integral equations by integrating the transport equation in radial and axial directions as in Eq. (3.7) and Eq. (3.8):

$$\begin{aligned}(\Omega\cdot\nabla)_{xy}\bar{\varphi}_g(\vec{r},\Omega) + \Sigma_{t,g}(\vec{r})\bar{\varphi}_g(\vec{r},\Omega) \\ = \bar{Q}_g(\vec{r},\Omega) - \underbrace{\frac{\cos\theta}{h_z}(\varphi^{z+}(\vec{r},\Omega) - \varphi^{z-}(\vec{r},\Omega))}_{L_z}\end{aligned} \tag{3.7}$$

$$\begin{aligned}\cos\theta \frac{d\bar{\bar{\varphi}}_g(\vec{r},\Omega)}{dz} + \Sigma_{t,g}(\vec{r})\bar{\bar{\varphi}}_g(\vec{r},\Omega) \\ = \bar{\bar{Q}}_g(\vec{r},\Omega) - \underbrace{\frac{1}{A_{xy}}\oint_l (\Omega\cdot\hat{n}_l)_{xy}\varphi_g(\vec{r},\Omega)dl}_{L_{xy}}\end{aligned} \tag{3.8}$$

where the overbars $\bar{\cdot}$ and $\bar{\bar{\cdot}}$ denote that a variable is axially and radially averaged, respectively. The terms L_z and L_{xy} which couple two integral equations by passing the information of other direction as the source are called axial and radial transverse leakages, respectively.

In the 2D/1D approximation, the radial equation is solved rigorously considering the sub-pin level details by employing MOC, while the axial equation is solved only approximately. The bi-directional solutions are then cast into the global 3D CMFD acceleration. nTRACER employs SP_3 whole-node Source Expansion Nodal Method (SENM) [71] for the axial solver, and there are lots of variants of the 2D/1D method in terms of the choice of axial solver.

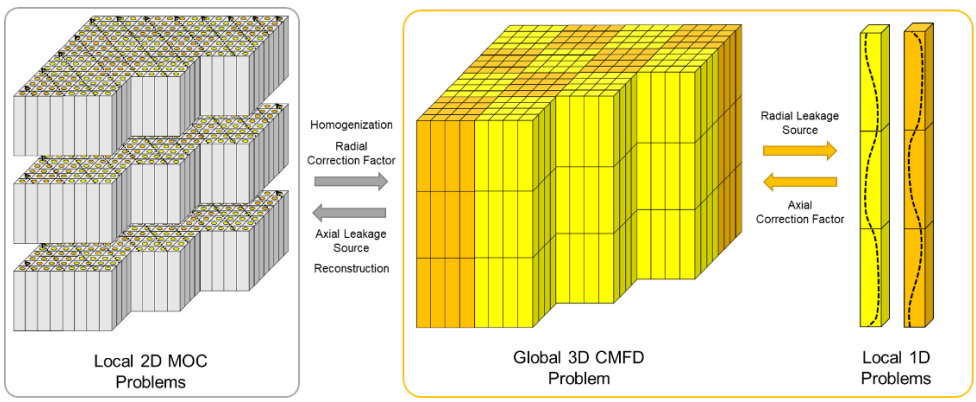


Figure 3.1 Schematic diagram of 2D/1D approximation.

3.1.3 MOC Calculation

MOC is a mathematical scheme to solve a partial differential equation (PDE) by reducing the PDE to a family of ordinary differential equations (ODE). In the 2D/1D method, MOC is used to solve the radial equation rigorously. For the MOC solution of the neutron transport equation, following characteristic curve is defined:

$$\vec{r} = s\Omega . \tag{3.9}$$

The characteristic curve is viewed as a ray flying in a fixed direction, and from this, the computational scheme to solve the transport equation by MOC is referred to as ray tracing calculation. Plugging Eq. (3.9) into Eq. (3.3) leads to the characteristics equation:

$$\frac{d}{ds}\varphi_g(s) + \Sigma_{t,g}(s)\varphi_g(s) = Q_g(s,\Omega). \tag{3.10}$$

For the numerical solution of the characteristics equation, flat source region (FSR) approximation is introduced as the spatial discretization scheme. In each FSR, it is assumed that the cross section and source are constant. With this assumption, the solution of the characteristics equation for a characteristic curve passing FSR i can be obtained readily by the method of integrating factors:

$$\varphi_g(s,\Omega) = \varphi_g(s_0,\Omega)\exp(-\Sigma_{t,g,i}s) + \frac{Q_{g,i}(\Omega)}{\Sigma_{t,g,i}}(1-\exp(-\Sigma_{t,g,i}s)). \tag{3.11}$$

The ray tracing calculation solves Eq. (3.11) over the entire phase space involving space, energy, and angle. Figure 3.2 illustrates a typical domain on which ray tracing is performed. Note that the rays are not created in axial direction; the rays in the polar direction are only the projections from the plane. The MOC calculation in the 2D/1D solution solves plane-wise axially integrated transport equations, so it is referred to as planar MOC calculation. By taking average of Eq. (3.11) over the segment r in the polar direction θ, the average angular flux of segment is obtained:

$$\bar{\varphi}_{g,r}(\Omega) = \frac{1}{L_r}\int_0^{L_r}\varphi_g(s,\Omega)ds = \frac{\varphi_g(0,\Omega)-\varphi_g(L_r,\Omega)}{\Sigma_{t,i}L_r} + \frac{Q_{g,i}(\Omega)}{\Sigma_{t,g,i}} \tag{3.12}$$

where $L_r = l_r / \sin\theta$. By taking the areal average of segment-averaged angular flux in the FSR, the region-averaged angular flux is obtained:

$$\Phi_{g,i}(\Omega) = \frac{\Delta_\alpha}{A_i}\sum_{r\in i} L_r\bar{\varphi}_{g,r}(\Omega) \tag{3.13}$$

where Δ_α is the ray spacing of azimuthal angle α and A_i is the area of FSR i.

The scalar flux of the FSR is then calculated as the weighted sum of angular flux:

$$\phi_{g,i} = \sum_{\Omega} \omega(\Omega)\Phi_{g,i}(\Omega) \tag{3.14}$$

where ω is the angular quadrature weight.

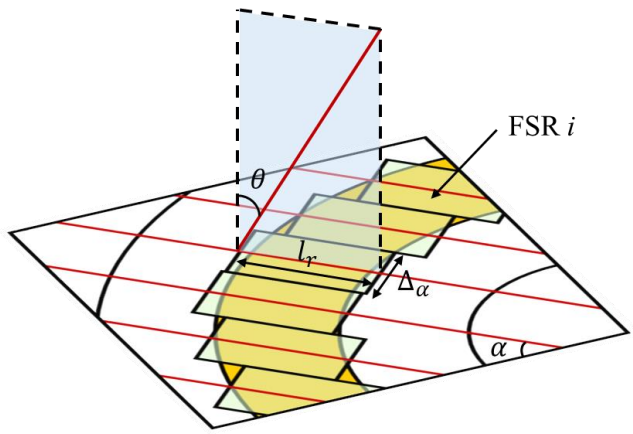


Figure 3.2 Illustration of the ray tracing domain.

In practical calculations, the transport correction is applied and isotropic scattering is assumed, which is called (transport-corrected) P_0 calculation. For more rigorous consideration of the scattering anisotropy, however, a spherical harmonics expansion can be introduced to the angular flux, which is referred to as the P_L method.

The definitions of the spherical harmonics and its conjugate complex are as follows:

$$Y_l^m(\Omega) = Y_l^m(\mu,\alpha) = \sqrt{\frac{(l-m)!}{(l+m)!}} P_l^m(\mu) e^{im\alpha}, \tag{3.15}$$

$$Y_l^{m*}(\Omega) = Y_l^{-m}(\Omega) \tag{3.16}$$

where

$$P_l^m(\mu) = (-1)^m \sqrt{1-\mu^2}^m \frac{d^m P_l(\mu)}{d\mu^m}; m \le l \tag{3.17}$$

is the associated Legendre polynomial. P_l is the ordinary Legendre polynomial of order l, and μ is the cosine of polar angle θ. Then, the angular flux is expanded using the *L*-th order spherical harmonics as follows:

$$\varphi_g(\Omega) = \sum_{l=0}^{L} \sum_{m=-l}^{l} \frac{2l+1}{4\pi} \phi_g^{lm} Y_l^m(\Omega) \tag{3.18}$$

where

$$\phi_g^{lm} = \int_{-1}^{1} d\mu' \int_{0}^{2\pi} d\alpha' \varphi_g(\mu', \alpha') Y_l^{m*}(\mu', \alpha') \tag{3.19}$$

is the angular flux moment. The high-order scattering source is then computed as:

$$Q_g^s(\Omega) = \sum_{g'=1}^{G} \sum_{l=0}^{L} \sum_{m=-l}^{l} \frac{2l+1}{4\pi} \Sigma_{g'g}^l \phi_{g'}^{lm} Y_l^m(\Omega) \tag{3.20}$$

where

$$\Sigma_{g',g}^l = \int_{-1}^{1} d\mu_s 2\pi \Sigma_{g',g}(\mu_s) P_l(\mu_s) \tag{3.21}$$

is the *l*-th order Legendre angular expanded scattering cross section, and μ_s is the cosine of the scattering angle.

In the P_0 calculation, it is more efficient to calculate the region-wise scalar flux directly during the ray tracing calculation using the segment-averaged angular flux. However, P_L calculation explicitly stores the region-wise angular flux by the angular flux saving scheme [72] to compute the region-wise angular flux moments as follows:

$$\phi_{g,i}^{lm} = \sum_{\Omega} \omega(\Omega) \Phi_{g,i}(\Omega) Y_l^{m*}(\Omega). \tag{3.22}$$

3.1.4 CMFD Acceleration

CMFD acceleration is basically a spatial multi-grid acceleration scheme. As shown in Figure 3.3, CMFD acceleration involves a successive feedback between the fine mesh solution of high-order calculations and the coarse mesh solution of low-order calculations. The primary role of the CMFD acceleration is to accelerate the global fission source convergence by globally rebalancing the fission source distribution on a coarse mesh basis, and the secondary role in the 2D/1D method is to tightly couple the bidirectional solvers under a global 3D solution framework.

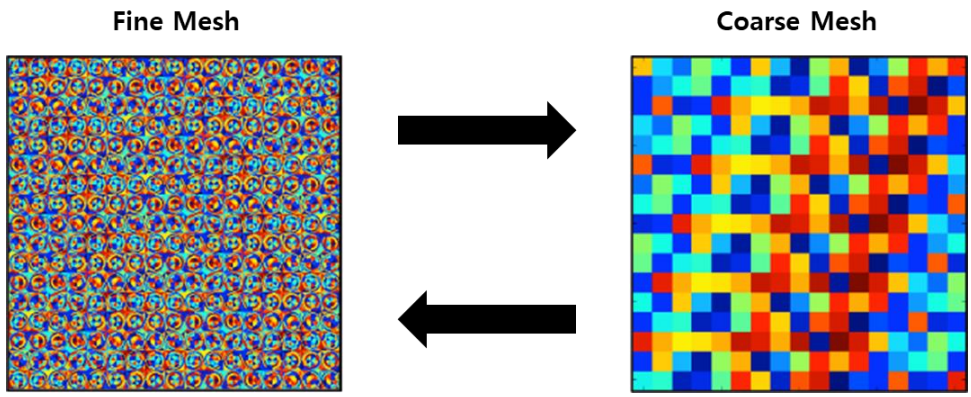


Figure 3.3 Concept of the CMFD acceleration.

The process to collapse the fine mesh to the coarse mesh is called homogenization. The cross sections in the coarse meshes are calculated as the flux-volume-weighted average of the fine mesh cross sections:

$$\bar{\Sigma}_I = \frac{\sum_{i \in I} \Sigma_i \phi_i V_i}{\sum_{i \in I} \phi_i V_i} \tag{3.23}$$

where I is the coarse mesh index.

CMFD acceleration performs finite difference calculation on the coarse mesh, as its name implies. Therefore, the formulation of CMFD acceleration starts from the

derivation of the finite difference neutron diffusion equation. Assume there are three nodes as illustrated in Figure 3.4. The within-group neutron balance of node i will be expressed as:

$$\left(J^L + J^R\right) + h_i \Sigma_{r,i} \phi_i = h_i Q_i \,. \tag{3.24}$$

The goal of the finite difference discretization of neutron diffusion equation is to construct a system with respect to the node flux, and therefore the currents should be expressed in terms of the node flux. It is achieved by imposing the current continuity on the surface using the surface flux:

$$J^R = -D_i \frac{\phi_s - \phi_i}{\frac{h_i}{2}} = -D_{i+1} \frac{\phi_{i+1} - \phi_s}{\frac{h_{i+1}}{2}} \,. \tag{3.25}$$

We now define the term $\frac{D}{h}$ as diffusivity and denote it as β. Taking out the surface flux in Eq. (3.25) yields:

$$\phi_s = \frac{\beta_i \phi_i + \beta_{i+1} \phi_{i+1}}{\beta_i + \beta_{i+1}} \,. \tag{3.26}$$

Plugging Eq. (3.26) into Eq. (3.25) again yields:

$$J^R = -2\beta_i (\phi_s - \phi_i) = -\tilde{D}^R (\phi_{i+1} - \phi_i) \,. \tag{3.27}$$

where

$$\tilde{D}^R = \frac{2\beta_i \beta_{i+1}}{\beta_i + \beta_{i+1}} \tag{3.28}$$

is the diffusion coupling coefficient. As the result, the nodal balance equation can be written in terms of the node flux, which can be formulated into a linear system:

$$-\tilde{D}^L\phi_{i-1} - \tilde{D}^R\phi_{i+1} + (\tilde{D}^L + \tilde{D}^R + h_i\Sigma_{r,i})\phi_i = h_iQ_i\,. \tag{3.29}$$

Extension of this 1D balance equation to 3D is done by simply superimposing each direction and changing length to volume.

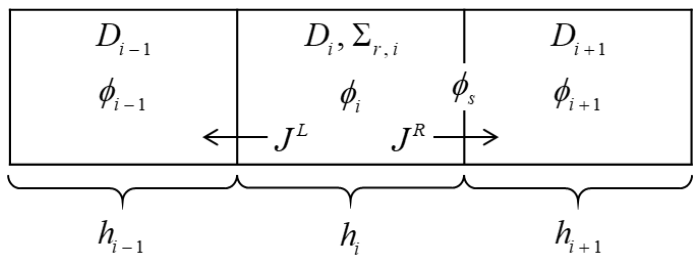


Figure 3.4 Illustration of node discretization.

The CMFD acceleration is merely the modification of the current relation from the finite difference diffusion relation, as follows:

$$J^R_{ref} = -\tilde{D}^R(\phi_{i+1} - \phi_i) - \hat{D}^R(\phi_{i+1} + \phi_i) \tag{3.30}$$

where J^R_{ref} is the reference current calculated from high-order calculations such as MOC, and $\hat{D}^R$ is the current correction factor defined as follows:

$$\hat{D}_R = \frac{\tilde{D}_R(\phi_{i+1} - \phi_i) - J_R^{ref}}{\phi_{i+1} + \phi_i}\,. \tag{3.31}$$

Namely, the CMFD acceleration introduces a correction factor to the finite difference diffusion formulation which preserves the current of high-order calculations, so that the finite difference calculation can produce equivalent solutions to the high-order calculations.

3.2 Planar Method of Characteristics

The planar MOC calculation that is responsible for the pin-resolved high-fidelity transport solutions in the radial direction is extremely compute-intensive owing to the necessity of a very fine discretization. The rays are generated with the spacing far less than a millimeter and the number of angles is usually in the order of hundreds. As the result, the total number of ray segments for a typical 3D quarter core problem reaches hundreds of billions, and the total number of unknowns can become trillions including the energy groups. Furthermore, the planar MOC calculation is an iterative calculation which has to be performed repeatedly.

Therefore, an efficient GPU acceleration of the planar MOC calculation is crucial for retaining the practicality of the 2D/1D method. This section describes the GPU acceleration algorithm of the planar MOC calculation in nTRACER. Starting from the ray data structure, changes in the sweep algorithms, utilization of CPU − GPU concurrency and mixed precision, and the design of GPU-based ray tracing kernels are explained in full detail in this section, and the performance analysis is presented in the last part of the chapter.

3.2.1 Ray Structure

The ray structure of nTRACER is implemented in a hierarchical manner, as shown in Figure 3.5. The ray hierarchy consists of four levels: rotational ray, core ray, pin ray, and ray segment. These rays are all defined on a radial plane. Rotational rays are defined for the problems that have reflective boundaries. A rotational ray departs from a point on a vacuum boundary and keeps reflecting at the reflective boundaries until it reaches another vacuum boundary. It is the unit of ray-based parallelism and composed of core rays. A core ray starts from a core boundary point and ends at the other side of the boundary regardless of the boundary condition, and it consists of

multiple of pin rays. A pin ray is the part of the core ray that lies between the two pin surfaces, which are needed to tally the pin surface currents for the CMFD parameter generation. A pin ray is composed of ray segments which are formed between the intersections of different FSRs, and the ray segments become the basic unit of the MOC calculation.

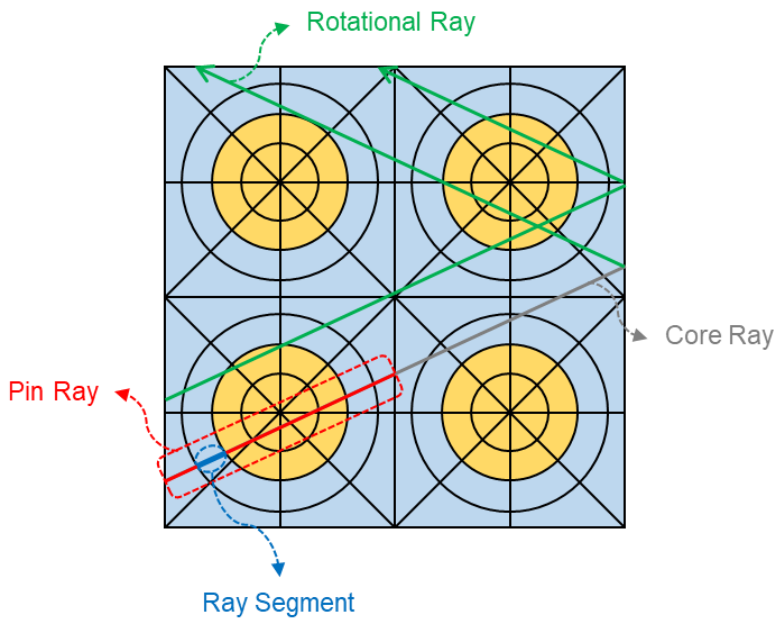


Figure 3.5 Hierarchal structure of rays.

In the host side, the ray information is stored with abstract data types which reflect the hierarchical structure of the rays. For efficient data access of ray information on a GPU, however, the structure of the arrays storing the ray information was changed such that the ray segment data are saved in one-dimensional arrays. The ray data are retrieved through the displacement vectors. As illustrated in Figure 3.6**,** each element of a displacement vector at a certain level points to the beginning position of the subsequent level's displacement vector or data, which constructs a tightly nested top-down sweep loop; i.e., the displacements of each level becomes the beginning and end indices of the subsequent loop.

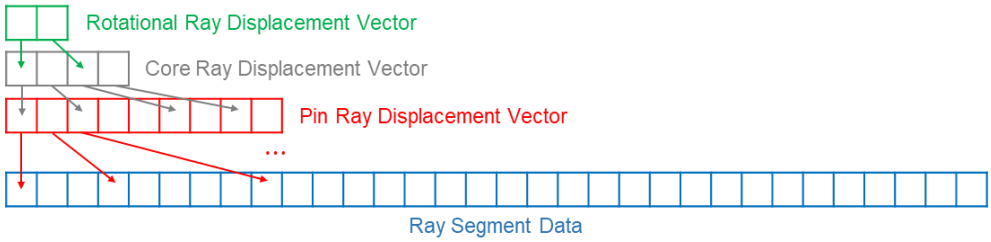


Figure 3.6 Ray storage layout.

3.2.2 Sweep Algorithm

As presented in Algorithm 3.1, there are two schemes in the ray sweep algorithm as far as the group update of the scattering source is concerned: Gauss-Seidel and Jacobi. The Gauss-Seidel scheme sweeps the energy groups at the outermost loop. The inner loops are for spatial and angular sweeps. This enables on-the-fly update of the scattering source during each iteration and therefore the updated flux information at high energy groups quickly propagate to the lower energy groups, which improves the inner iteration convergence. In addition, the Gauss-Seidel scheme is favorable in terms of memory usage. nTRACER has been employing ray-based parallelism and a temporary flux storage buffer is required for each thread to avoid data races. In the Gauss-Seidel scheme, the buffer is only required for a single energy group and can be reused for other groups. Therefore, nTRACER has been using the Gauss-Seidel scheme as the default solver for the conventional multi-core CPU calculations.

On the other hand, the Jacobi scheme has the energy group sweep at the innermost loop so that the scattering source is fixed for all groups during the ray tracing. Since the Jacobi scheme uses the scalar flux solution of the previous iteration to determine the scattering sources of the current iteration, the overall convergence is slower. In addition, the temporary flux storage buffer should be allocated for all energy groups and it results in a significantly more memory usage.

Nonetheless, the Jacobi scheme has computationally compelling aspect. First, it can achieve higher memory throughput than the Gauss-Seidel scheme. The Gauss-

Seidel scheme has the spatial sweep at the inner loop and the rays drive over the problem domain rather chaotically, which makes the contiguous access on FSR data barely expectable. On the other hand, the Jacobi scheme has the energy sweep at the inner loop and the energy data can always be contiguously accessed, which increases the cache hit.

Second, the inner loop of the Jacobi scheme can be vectorized, which is crucial for the GPU acceleration. Performing the same operation (ray sweep) on contiguous data (energy) is the very characteristics of the SIMD parallelism. The inner loop of the Gauss-Seidel scheme can be also vectorized, but the memory accesses are not coalesced and the loop size varies thread-by-thread, which would likely cause branch divergences.

Therefore, the Jacobi scheme is adopted for GPU acceleration. Note that the issue of the buffer memory size is not valid anymore for the GPU implementations, as the flux accumulation on GPUs should rely on the atomic addition regardless of the ray sweep algorithms due to massive parallelism. Furthermore, it turned out that the slow convergence of the Jacobi scheme is compensated by the CMFD acceleration. As the result, both of the demerits of the Jacobi scheme are gone and only the computational advantage of the Jacobi scheme remains.

Algorithm 3.1 Gauss-Seidel and Jacobi ray sweep algorithms.

Gauss-Seidel	**Jacobi**
FOR $g \in G$ **DO** Group Sweep	Calculate source for all $g \in G$
Calculate source for group g	**FOR** r **DO** Rotational Ray Sweep
FOR r **DO** Rotational Ray Sweep	**FOR** $l \in r$ **DO** Pin Ray Sweep
FOR $l \in r$ **DO** Pin Ray Sweep	**FOR** $s \in l$ **DO** Ray Segment Sweep
FOR $s \in l$ **DO** Ray Segment Sweep	**FOR** $g \in G$ **DO** Group Sweep
FOR p **DO** Polar Angle Sweep	**FOR** p **DO** Polar Angle Sweep
Solve equations over the rays	Solve equations over the rays
END DO	**END DO**
END DO	**END DO**
END DO	**END DO**
END DO	**END DO**
END DO	**END DO**

3.2.3 Asynchronous Calculation and Mixed Precision

The main advantage of using a heterogeneous computing platform is that multiple types of computing devices can run asynchronously. If task parallelism is exploited with each task being properly assigned to a suitable device, the overall runtime can be substantially reduced. In the MOC calculation, there are two major tasks: source calculation and ray tracing.

The ray tracing calculation is compute-intensive, vectorizable, and insensitive to the arithmetic precision, so it is appropriate for the GPU acceleration and can be run with single precision arithmetic efficiently. In contrast, the source calculation is computationally less intense, but it is data-intensive and sensitive to the arithmetic precision. It requires the calculation of region-wise macroscopic cross sections using the microscopic cross sections and the regional material properties. The data used for the source calculation is very extensive and irregular, and the order of the number densities and the microscopic cross sections vary in magnitude. Therefore, it is more suitable for CPU which has plenty of memory and double precision computing power, and which can decently handle complicated data.

However, the computational burden of source calculation is not negligible as it involves region-wise scattering matrix multiplication. Thus, it is better to overlap the CPU-based source calculation and the GPU-based ray tracing using the concurrency of CPU and GPU. The problem is that the source calculation and the ray tracing calculation are essentially sequential as the ray tracing calculation requires the source to be known.

Therefore, a block decomposition strategy is introduced to the energy sweep such that the entire energy groups are divided into a number of blocks and processed sequentially, but without the source update between blocks (fully Jacobi). It enables overlapping CPU source calculation and GPU ray tracing by letting the host proceed without waiting for the result from the device, as illustrated in Figure 3.7. Depending

on the relative execution speed of host and device, two types of execution modes shown in the figure can be observed.

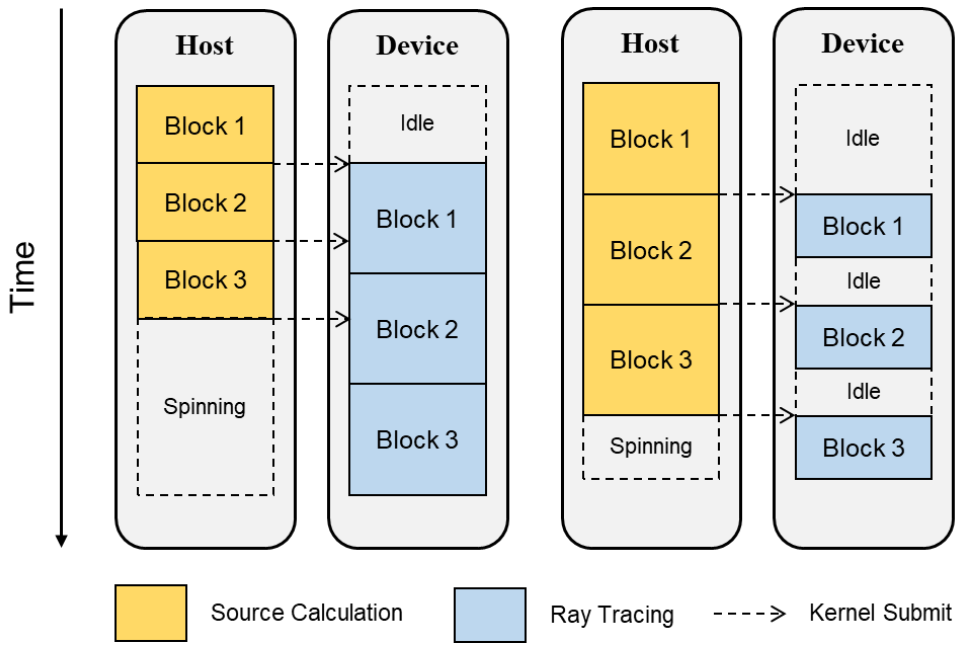


Figure 3.7 Schematic diagram of CPU – GPU asynchronous execution modes.

The schematic diagram of the asynchronous execution is illustrated in Figure 3.8. Once the source of the current block is ready, the host driver fires the ray tracing task of the current group block to the device, forgets about its progress, and proceeds to the source calculation of the next group block. This is repeated until the sweep over the group blocks is completed.

The data copy between the host and the device are done through the buffers on the pinned memory which allows Direct Memory Access (DMA) and does not require synchronization for the copies. The source is first calculated on the pageable memory with double precision and is typecasted to single precision while copying the source to the pinned memory buffers. Then, the device can copy the source asynchronously with the host whenever needed. The ray tracing results are also stored on the pinned memory asynchronously and collected by the host after the completion of the group block sweep.

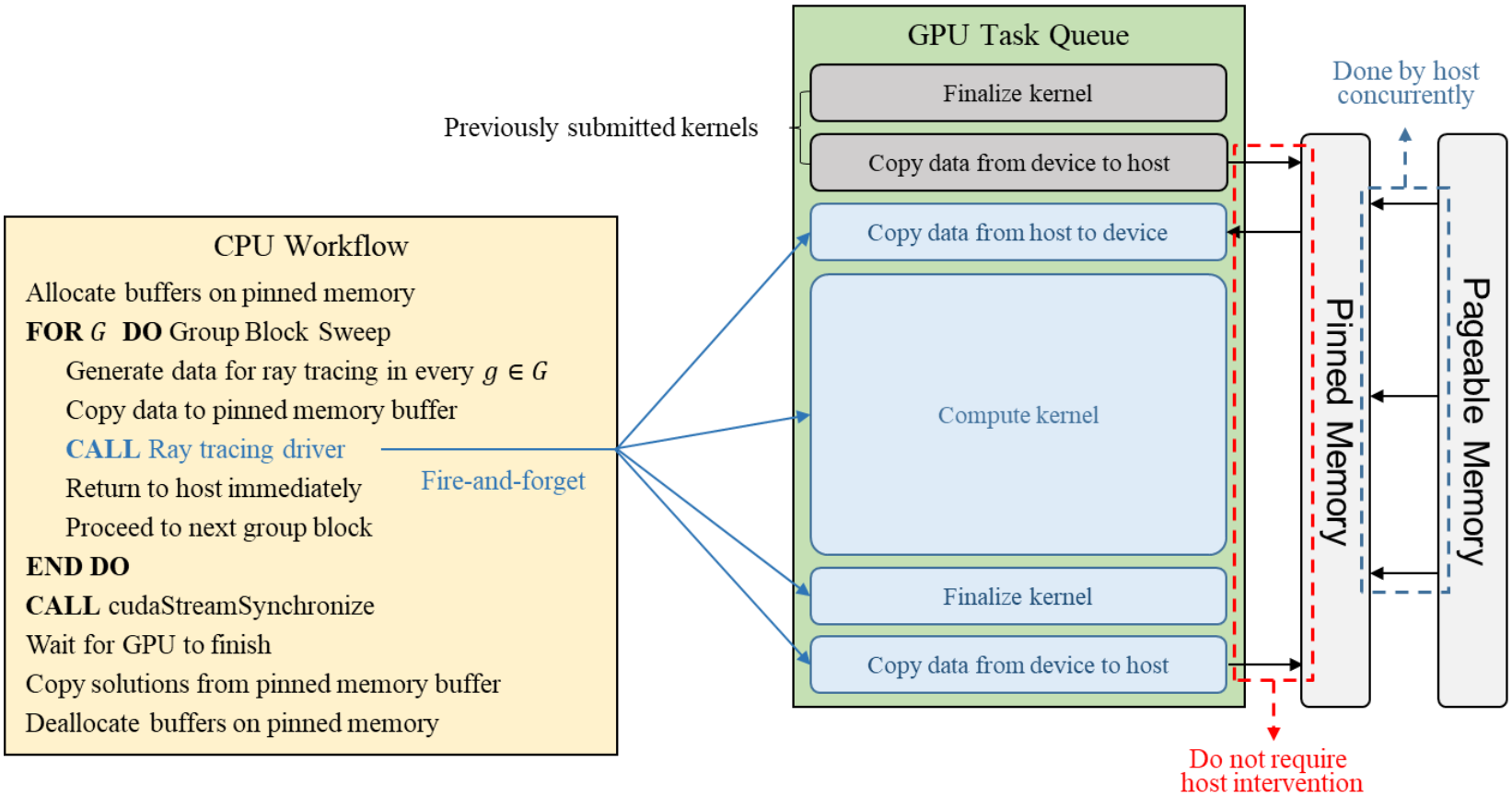


Figure 3.8 Illustration of CPU – GPU asynchronous MOC calculation scheme.

3.2.4 P_0 Ray Tracing Kernel

The GPU acceleration of P_0 ray tracing calculation was first explored by Boyd et al. [22], and nTRACER mostly adopts the algorithm with further considerations of the mixed precision arithmetic and the CPU − GPU concurrency. P_0 ray tracing is relatively easy to implement on GPU because of the isotropic assumption; explicit storage of angular flux is unnecessary. Thus, the angular flux of each segment can be saved in a local variable and can be discarded after the accumulation to the global scalar flux array.

The GPU-based P_0 MOC module consists of two kernels: ray tracing calculation and true scalar flux calculation. Description below explains the operations that each kernel performs. Note that the angular quadrature weights in nTRACER are defined such that the sum over 4π becomes unity. The equations appearing from now on will follow the modified quadrature.

[Kernel 1] Ray tracing calculation

Instead of calculating the true scalar flux directly during the ray tracing calculation, the pseudo scalar flux $\tilde{\phi}$, which is the weighted sum of the angular flux changes in each FSR, are calculated, which will be converted to the true scalar flux after the ray tracing kernel by a mathematical manipulation of the equations:

$$\varphi_{out,g}(\Omega) = \varphi_{in,g}(\Omega)e^{-\Sigma_{tr,g}l} + \tilde{Q}_g(1-e^{-\Sigma_{tr,g}l}), \tag{3.32}$$

$$\tilde{\phi}_g \mathrel{+}= \omega(\Omega)\sin\theta\,\Delta_\alpha(\varphi_{in,g}(\Omega) - \varphi_{out,g}(\Omega)) \tag{3.33}$$

where the source is divided by the transport cross section in advance:

$$\tilde{Q}_g = \frac{Q_g}{\Sigma_{tr,g}}. \tag{3.34}$$

[Kernel 2] True scalar flux calculation

Pseudo scalar flux is converted to the true scalar flux by the following equation:

$$\phi_g = \frac{\tilde{\phi}_g}{\Sigma_{tr,g} A} + \tilde{Q}_g \,. \tag{3.35}$$

The parallel block Jacobi ray tracing algorithm for the P_0 calculation is described in Algorithm 3.2. The atomic accumulation of neutron currents at the pin boundaries is for the use in the CMFD acceleration. Following considerations were made for the optimization:

1. The polar angle loop is not parallelized because the polar angle components can be discarded once its contribution to the scalar flux is known; namely, for each segment the polar angular flux can be accumulated to a local variable. Exploiting local variables is more favorable than generating more threads with more global memory accesses unless the thread occupancy is very low.
2. Before the polar angle sweep, transport cross section, source, and segment length, which are shared over the polar angles, are prefetched to the local variables.
3. To save outgoing angular flux that will be fed to the subsequent segment as the incoming angular flux, shared memory is used. This quantity is also temporary and thread-private, but it cannot utilize registers since the number of polar angles is unknown at the compile time. If a local array is dynamically indexed, it goes to the global memory instead of the register because register does not support dynamic addressing.
4. The angular quadrature weight, trigonometric function value, and ray spacing of each angle are saved in the constant memory as a single term; namely, they are multiplied in advance as they always appear together.
5. For the exponential calculation, the intrinsic fast math function __expf() is used. The common practice for this is to tabulate the exponential values, but on GPU, accessing tables is considered more expensive.

Algorithm 3.2 Parallel block Jacobi P_0 ray tracing algorithm.

```
FOR G DO Group Block Sweep
  Copy in pseudo source and total cross section
  [Kernel 1] Ray tracing
  FOR r PARALLEL DO Rotational Ray Sweep
    FOR l ∈ r DO Pin Ray Sweep
      FOR g ∈ G PARALLEL DO Group Sweep
        FOR p DO Polar Angle Sweep
          Accumulate pin incoming current on local variable
        END DO
        Atomically accumulate pin incoming current
      END PARALLEL DO
      FOR s ∈ l DO Ray Segment Sweep
        FOR g ∈ G PARALLEL DO Group Sweep
          Prefetch FSR and ray data to local variables
          FOR p DO Polar Angle Sweep
            Save outgoing track angular flux on shared memory
            Accumulate angular flux change on local variable
          END DO
          Atomically accumulate region pseudo scalar flux
        END PARALLEL DO
      END DO
      FOR g ∈ G PARALLEL DO Group Sweep
        FOR p DO Polar Angle Sweep
          Accumulate pin outgoing current on local variable
        END DO
        Atomically accumulate pin outgoing current
      END PARALLEL DO
    END DO
  END PARALLEL DO
  [Kernel 2] Calculate true scalar flux
  Copy out solution
END DO
```

Figure 3.9 shows the threading scheme, where the rotational rays are distinguished by colors. Each thread takes an energy group of a rotational ray, and the rays and the energy groups are distributed using the global thread indices. In certain warps, more than two rotational rays with different lengths can be interleaved, which will likely cause thread divergences. However, this is inevitable unless the number of energy groups becomes a multiple of 32. In nTRACER, a 47-group energy group structure is used and the group block size for the CPU − GPU asynchronous calculation is set to 16 for P_0 MOC.

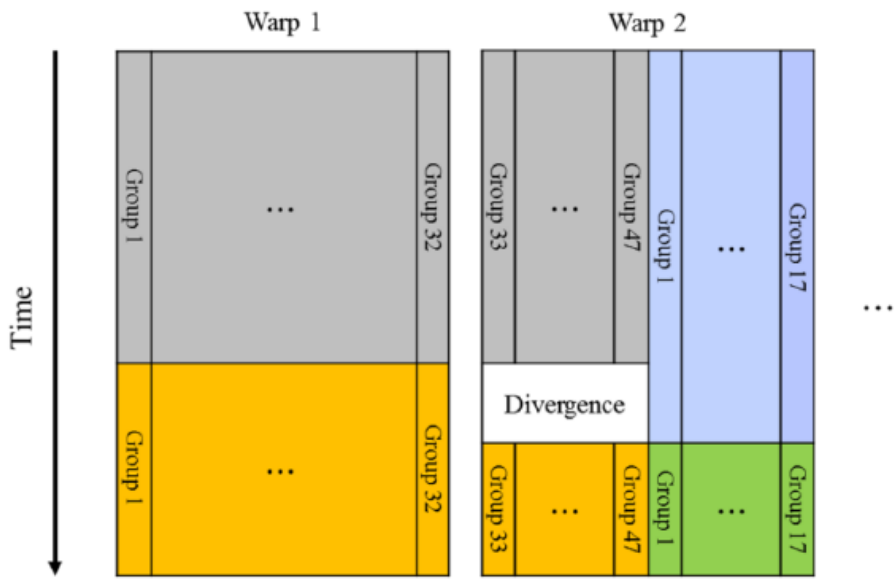


Figure 3.9 P_0 ray tracing calculation threading scheme.

3.2.5 P_L Ray Tracing Kernel

P_L ray tracing calculation is more cumbersome than the P_0 counterpart due to the larger requirements of memory caused by the storage of angular flux. Recall that as opposed to the P_0 MOC, angular flux of each FSR is stored during ray tracing and the angular flux moments are evaluated based on the FSR-wise angular flux. Hence, in addition to the energy decomposition, angular decomposition is indispensable to reduce the memory requirements.

The P_L MOC module is composed of four kernels: angular source calculation, ray tracing calculation, pseudo angular flux moment accumulation, and true angular flux moment calculation. The role of each kernel is explained below.

[Kernel 1] Angular source calculation

The angle-dependent source is computed by multiplying the spherical harmonics components to the angular source moments:

$$\tilde{Q}_g(\Omega) = \sum_{l=0}^{L} \sum_{m=-l}^{l} Y^{lm}(\Omega) \tilde{Q}_g^{lm} \tag{3.36}$$

where the angular source moments are defined as:

$$\tilde{Q}_g^{00} = \frac{1}{\Sigma_{t,g}} Q_g = \frac{1}{\Sigma_{t,g}} \left(\frac{\chi_g}{k_{eff}} \sum_{g'} \nu\Sigma_{f,g'} \phi_{g'} + \sum_{g'} \Sigma_{g' \to g} \phi_{g'} \right), \tag{3.37}$$

$$\tilde{Q}_g^{lm} = \frac{2l+1}{\Sigma_{t,g}} Q_g^{lm} = \frac{2l+1}{\Sigma_{t,g}} \left(\sum_{g'} \Sigma_{g' \to g}^{l} \phi_{g'}^{lm} \right). \tag{3.38}$$

Note that the angular source moments are divided by the total cross sections similarly to the P_0 case.

[Kernel 2] Ray tracing calculation

Following the rays, the angular flux changes $\tilde{\Phi}$ are accumulated in each FSR, but without weight:

$$\varphi_{out,g}(\Omega) = \varphi_{in,g}(\Omega) e^{-\Sigma_{t,g} l} + \tilde{Q}_g(\Omega)(1 - e^{-\Sigma_{t,g} l}), \tag{3.39}$$

$$\tilde{\Phi}_g(\Omega) += \varphi_{in,g}(\Omega) - \varphi_{out,g}(\Omega). \tag{3.40}$$

[Kernel 3] Pseudo angular flux moment accumulation

Region-averaged angular flux changes are accumulated to the pseudo angular flux moments with the angular quadrature weights and spherical harmonics components:

$$\tilde{\phi}_g^{lm} = \sum_{\Omega} \omega(\Omega)\Delta_\alpha Y^{lm,*}(\Omega)\tilde{\Phi}_g(\Omega)\,. \tag{3.41}$$

[Kernel 4] True angular flux moment calculation

True angular flux moment is calculated from the pseudo angular flux moment:

$$\phi_g^{lm} = \frac{\tilde{\phi}_g^{lm}}{\Sigma_{t,g} A} + \frac{\tilde{Q}_g^{lm}}{2l+1}\,. \tag{3.42}$$

The parallel ray tracing algorithm for the P_L calculation is described in Algorithm 3.3. There are two major differences in the algorithm from the P_0 counterpart. First, the azimuthal angles are decomposed and processed sequentially due to the storage limitation. At each ray tracing kernel, four azimuthal angles that form a rotation in 2π are grouped together as shown in Figure 3.10, and the rotational rays belonging to the azimuthal group are processed in parallel. After the ray tracing of an azimuthal group, their contributions to the pseudo angular flux moments are calculated and the angular flux storage is initialized for the next use.

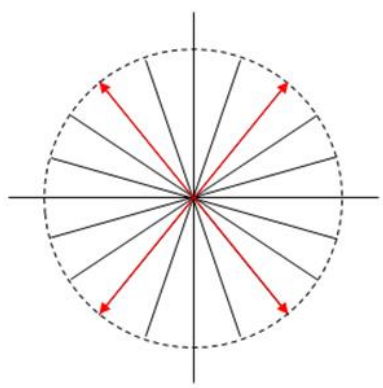


Figure 3.10 Illustration of angles constituting an azimuthal group.

Second, parallelization is applied to the polar angle loop. Since the polar angular flux should be stored explicitly, the optimization by accumulating the polar angular flux on the local variable in the P_0 case is no more valid. In addition, the source is now anisotropic and prefetching the source has no effect. Therefore, polar angle loop is also parallelized to extract more parallelism which compensates for the reduction of parallelism by the azimuthal angle decomposition. It is expected that the multicast mechanism of the global memory load will optimize the accesses to the shared data over the polar angles such as the total cross section and segment length.

Figure 3.11 illustrates the threading scheme. Each thread takes one energy group and one polar angle of a rotational ray. Even though the number of polar angles can be user-dependent, in most cases a constant value of 4 is used in nTRACER with the Tabuchi-Yamamoto polar angle quadrature [73]. Thus, the size of energy group block is set to 8 in P_L calculations, which can fit a rotational ray into the warp width.

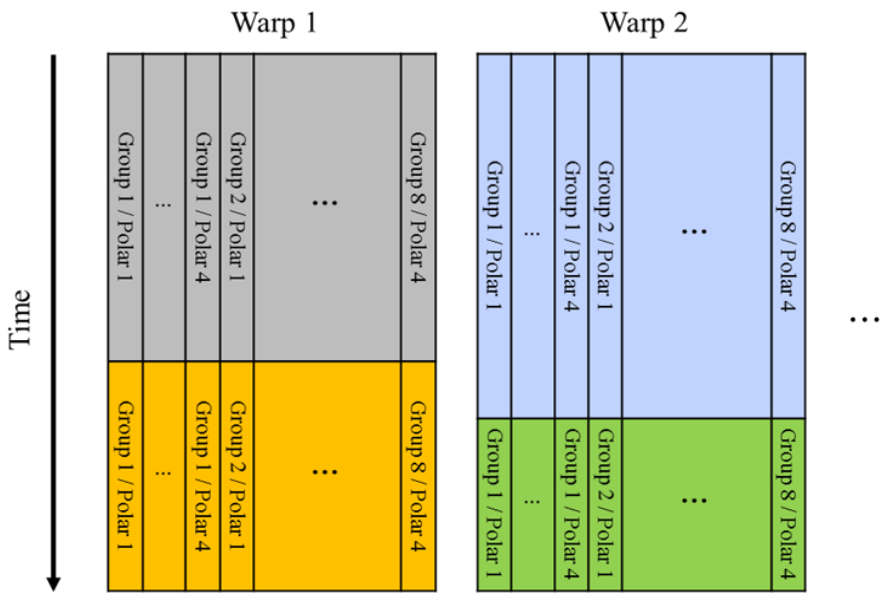


Figure 3.11 P_L ray tracing calculation threading scheme.

Algorithm 3.3 Parallel block Jacobi P_L ray tracing algorithm.

```
FOR g DO Group Block Sweep
  Copy in pseudo angular source moment and total cross section
  FOR a DO Azimuthal Angle Sweep
    [Kernel 1] Compute pseudo angular source
    [Kernel 2] Ray tracing
    FOR r ∈ a PARALLEL DO Rotational Ray Sweep
      FOR l ∈ r DO Pin Ray Sweep
        FOR g ∈ G PARALLEL DO Group Sweep
          FOR p PARALLEL DO Polar Angle Sweep
            Atomically accumulate pin incoming current
          END PARALLEL DO
        END PARALLEL DO
        FOR s ∈ l DO Ray Segment Sweep
          FOR g ∈ G PARALLEL DO Group Sweep
            FOR p PARALLEL DO Polar Angle Sweep
              Save outgoing track angular flux on local variable
              Atomically accumulate region pseudo angular flux
            END PARALLEL DO
          END PARALLEL DO
        END DO
        FOR g ∈ G PARALLEL DO Group Sweep
          FOR p PARALLEL DO Polar Angle Sweep
            Atomically accumulate pin outgoing current
          END PARALLEL DO
        END PARALLEL DO
      END DO
    END PARALLEL DO
    [Kernel 3] Accumulate pseudo angular flux moment
    Initialize angular flux storage
  END DO
  [Kernel 4] Calculate true angular flux moment
  Copy out solution
END DO
```

3.3 CMFD Acceleration

The CMFD acceleration compensates notoriously slow convergence of the planar MOC calculation. It is also a key operation which couples radial and axial solutions in the 3D solution framework. The CMFD calculation entails a repeated solution of a large sparse linear system whose dimension can be tens of millions in whole-core problems, where the number of nonzeros is more than a billion. Therefore, a proper offloading of the CMFD calculation is also important.

This section covers the strategies for the GPU acceleration of CMFD calculation. The linear system structure of the legacy nTRACER solver was changed to a suitable form for parallelization, and a massively parallelizable preconditioner appropriate for the characteristics of the CMFD linear system was devised. The mixed precision strategy for the power iteration is also introduced.

3.3.1 Linear System Structure

In the conventional CPU-based CMFD solvers, so-called group major ordering is considered effective. In the group major ordering scheme, energy group becomes the primary sort variable and spatial node becomes the secondary. Resultantly, the linear system is ordered such that each group forms a block matrix:

$$\begin{pmatrix} \mathbf{M}_1 & -\mathbf{S}_{2,1} & \cdots & -\mathbf{S}_{G-1,1} & -\mathbf{S}_{G,1} \\ -\mathbf{S}_{1,2} & \mathbf{M}_2 & \ddots & -\mathbf{S}_{G-1,2} & -\mathbf{S}_{G,2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ -\mathbf{S}_{1,G-1} & -\mathbf{S}_{2,G-1} & \ddots & \mathbf{M}_{G-1} & -\mathbf{S}_{G,G-1} \\ -\mathbf{S}_{1,G} & -\mathbf{S}_{2,G} & \cdots & -\mathbf{S}_{G-1,G} & \mathbf{M}_G \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2 \\ \vdots \\ \boldsymbol{\phi}_{G-1} \\ \boldsymbol{\phi}_G \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \\ \vdots \\ \mathbf{Q}_{G-1} \\ \mathbf{Q}_G \end{pmatrix} \tag{3.43}$$

where $\mathbf{M}_g$ is the septa-diagonal block matrix containing the removal cross sections and the coupling coefficients between the nodes of group g, $\mathbf{S}_{g,g'}$ is the diagonal

matrix containing the scattering cross sections from group g to g', and $\boldsymbol{\phi}_g$ is the flux vector of group g. This allows decomposing a whole problem into group-wise local problems as follows:

$$\mathbf{M}_g \boldsymbol{\phi}_g = \mathbf{Q}_g + \sum_{g' \neq g} \mathbf{S}_{g',g} \boldsymbol{\phi}_{g'} \quad (g = 1 : G) \tag{3.44}$$

It reduces the computational cost significantly as it eliminates the scattering terms from each local linear system by moving them to the right hand side. This approach is valid because the migration of neutrons through energy by scattering is dominated by the down-scattering.

However, for the GPU-based CMFD calculation, such approach is inefficient. First, each block matrix is rather small to extract enough parallelism. Second, the solution of each group is serialized by the scattering source update. Therefore, the node major ordering is applied for the GPU application. In this ordering scheme, as opposed to the group major, the primary sort variable is the spatial node and the secondary is the energy group. As the result, dense block matrices which describe removal through collision and energy migration through scattering in each node are formed along the diagonal:

$$\mathbf{M}_i = \begin{pmatrix} \sum_{s=1}^{6} (\tilde{D}_1^s - \hat{D}_1^s) + \Sigma_{r,1} & -\Sigma_{2,1} & \cdots & -\Sigma_{G,1} \\ -\Sigma_{1,2} & \sum_{s=1}^{6} (\tilde{D}_2^s - \hat{D}_2^s) + \Sigma_{r,2} & \ddots & -\Sigma_{G,2} \\ \vdots & \ddots & \ddots & \vdots \\ -\Sigma_{1,G} & -\Sigma_{2,G} & \cdots & \sum_{s=1}^{6} (\tilde{D}_G^s - \hat{D}_G^s) + \Sigma_{r,G} \end{pmatrix} \tag{3.45}$$

Then, a global linear system including the spatial migration is constructed, and the problem is solved as a whole.

Since the node major migration matrix contains the scattering terms, they should be computed during the inner iteration and the required FLOPs is significantly larger than the group major scheme. However, the convergence of the node major scheme is more robust than the group major scheme because of the explicit incorporation of the up-scattering terms in the inner iteration, and it can achieve higher throughput on GPU by exposing more parallelism. In other words, the CMFD calculation runs more efficiently on CPU than on GPU due to the inherent physics of neutrons. However, massive parallelization with GPU can still be beneficial for the CMFD calculation.

3.3.2 Dropout Sparse Approximate Inverse (DSPAI) Preconditioner

BiCGSTAB with Sparse Approximate Inverse (SPAI) preconditioner is used as the linear system solver for the inner iteration. The SPAI preconditioner is appropriate for GPUs because the preconditioning is done by massively parallelizable matrix − vector multiplication. The SPAI preconditioning calculates an approximate inverse matrix $\mathbf{K}$ of the coefficient matrix $\mathbf{M}$, which has the same sparsity pattern with the original matrix, in a way that the following condition is satisfied:

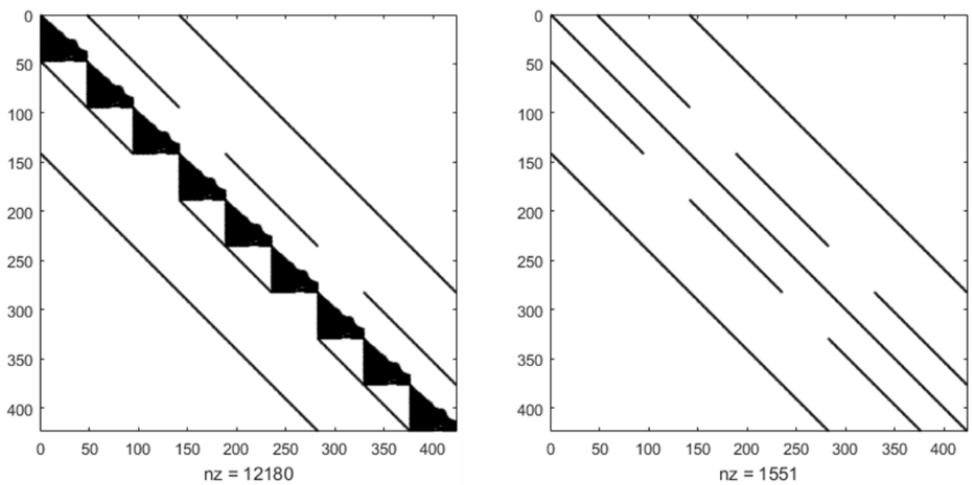
$$\mathbf{K} \leftarrow \arg\min_{\mathbf{K}} \left\| \mathbf{I} - \mathbf{MK} \right\| \tag{3.46}$$

where $\left\| \cdot \right\|_F$ is the Frobenius norm and $\mathbf{I}$ is the identity matrix. This minimization problem can be reduced to minimizing the Euclidean norm of each column vector:

$$\min \left\| \mathbf{I} - \mathbf{MK} \right\| = \sum_j \min \left\| \mathbf{e}_j - \mathbf{Mk}_j \right\| \tag{3.47}$$

where $\mathbf{e}_j$ and $\mathbf{k}_j$ are the column j of the identity matrix $\mathbf{I}$ and the preconditioner matrix $\mathbf{K}$, respectively.

Generation of a SPAI preconditioner is rather expensive, and thus, instead of using the full coefficient matrix, only the dominant nonzero terms are considered to reduce the preconditioner setup cost [49]. That is, out-scattering terms which are relatively small in the magnitude are 'dropped out' and only the spatial coupling terms and the removal terms are considered, as shown in Figure 3.12. Here, this preconditioning technique will be referred to as the Dropout SPAI preconditioner.



(a) (b)

Figure 3.12 Sparsity pattern of (a) the original linear system and (b) the dropout SPAI preconditioner.

The minimization problem of Eq. (3.47) for each column with the sparsity pattern shown in Figure 3.12(b) generates a set of least square problems sharing the same coefficient matrix $\mathbf{M}$ with different right hand sides which are merely the column vectors of the identity matrix. The solution of the least square problems to determine the sparse element in each column of $\mathbf{K}$ can be conveniently obtained by employing the QR factorization of $\mathbf{M}$, which turns each least square problem into the following linear system that can be solved quickly by a forward substitution:

$$\mathbf{R}\mathbf{k}_j = \mathbf{Q}^T\mathbf{e}_j \tag{3.48}$$

However, $\mathbf{M}$ is too large to be factorized directly. Thus, for each column vector $\mathbf{k}_j$, a reduced matrix is formed based on the sparsity pattern of the vector.

Although the CMFD matrix changes during the global solution process due to the continuous update of cross sections and surface currents, the preconditioner is not updated so that the QR factorization is performed only once. This is justified because the preconditioner is only approximate and the convergence of the preconditioned Krylov linear system solver would not be significantly affected by the slightly less accurate preconditioner.

3.3.3 Iterative Refinement

In the later stage of power iteration, the solution vectors are updated only slightly as the error reduces continuously. As the result, if the power iteration is performed in single precision, the error cannot be reduced below a certain level due to the round-off error lingering at the last significant digit. The remedy for this problem which is inevitable with the use of single precision for the iterative linear system solver is to introduce the iterative refinement technique [74] in the power iteration framework.

As shown in Algorithm 3.4, the linear system at each outer iteration is now solved with respect to the source residual which is obtained as the imbalance between the current fission source and the removal rates (including leakage) calculated with the previous flux solution. The subscripts denote the byte size of each variable's data type. The iterative linear system solution which takes most of the computing time in the power iteration is done in single precision, while the rest of the calculations are done in double precision including the residual vector calculation.

The validity of the mixed precision power iteration scheme is demonstrated in Figure 3.13. It can be seen that the lower bound of the power iteration error reduction is determined by the round-off error originating from the linear system solver. The single precision behavior starts to deviate from the double precision behavior around

the 10^{-6} error level and reaches the lower bound above 10^{-8}, which is not enough to guarantee a tight convergence. However, with the iterative refinement scheme, the double precision convergence behavior can be followed by the single precision linear system solver and thus a strict CMFD convergence can be ensured.

Algorithm 3.4 CMFD power iteration with iterative refinement.

DO WHILE Not Converged

Compute total source: $\mathbf{Q}_{(8)}^{(n)} = \frac{1}{k^{(n)}} \boldsymbol{\chi}_{(8)} \boldsymbol{\psi}_{(8)}^{(n)}$

Compute residual: $\mathbf{r}_{(8)} = \mathbf{Q}_{(8)}^{(n)} - \mathbf{M}_{(8)} \boldsymbol{\phi}_{(8)}^{(n)}$

Solve (iteratively) for correction: $\mathbf{M}_{(4)} \mathbf{d}_{(4)} = \mathbf{r}_{(4)}$

Add correction: $\boldsymbol{\phi}_{(8)}^{(n+1)} = \boldsymbol{\phi}_{(8)}^{(n)} + \mathbf{d}_{(8)}$

Update fission source: $\boldsymbol{\psi}_{(8)}^{(n+1)} = \mathbf{F}_{(8)} \boldsymbol{\phi}_{(8)}^{(n+1)}$

Update eigenvalue

Check convergence

END DO

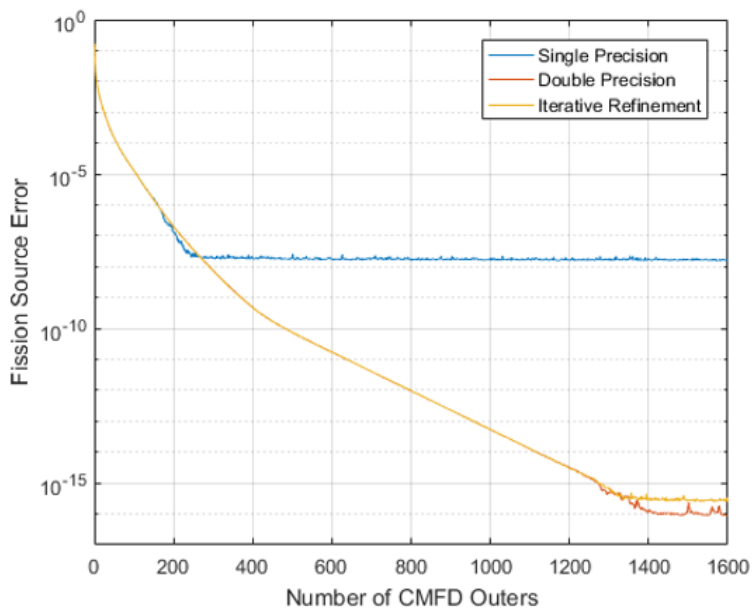


Figure 3.13 CMFD power iteration error behaviors of different precision schemes.

3.4 Axial Solver

The 2D/1D scheme is only conditionally stable and considerable efforts have been made by many research groups to stabilize the scheme. Mostly the instability has to do with the divergence of the CMFD calculation, which is largely contributed by the use of optically thick cells and the generation of negative flux in high-order solvers. The polynomial expansion (usually 4^{th} order) techniques employed axially to capture the spatial variation of flux effectively in the coarse mesh also deteriorates stability due to the oscillatory evolution of the axial flux distribution.

The issue of optical thickness has been well-addressed by several researches [75] [76] [77]; the solution is to simply increase the number of transport sweeps per each CMFD calculation or to modify the CMFD current relations. The instability incurred by the high-order polynomial expansion was also addressed by the subplane CMFD method [78] embedded by the Low Order Polynomial Expansion (LPEN) axial nodal solver [79]; LPEN expands the flux with a quadratic function by taking the advantage of thinner subplane sizes, which makes the axial flux distribution less oscillatory and evolve more stably. However, the negative flux problem is still an ongoing issue.

The negative flux occurs mainly from two causes: transverse leakage and transport correction. In the cells where fission source does not exist, the transverse leakage can make the total source negative. This can cause the flux to become negative unless enough streaming sources are provided from neighboring cells, which frequently occurs in the reflectors, and such circumstance is more frequent in the axial solver than the radial solver. In case of transport correction, the weighted sum of the first moment components of the scattering matrix is subtracted from the self-scattering cross section so that the self-scattering terms are often negative for light nuclei such as hydrogen. In transport solutions, the negative self-scattering source can make the total source negative, especially in the fast groups where the in-scatter source is not sufficient.

In this regard, a new axial solver employing 1D MOC was developed, which will be used as the main axial solver for the GPU acceleration in substitution for the built-in whole-node SENM solver in nTRACER. The axial MOC solver has the purpose of achieving higher accuracy, stability, and parallel performance than the whole-node SENM solver. The axial MOC solver evicts the polynomial expansion technique and employs a subgrid scheme to suppress severe axial flux variations while resolving detailed flux distributions. For stability and accuracy, leakage splitting [80], limited transport correction, and P_L methods are augmented. This section covers the detailed methods of the axial MOC solver and its parallelization.

3.4.1 Subgrid Scheme

For stability and simplicity of implementation, step characteristics (SC) scheme is employed in the axial MOC solver. A linear source MOC was developed by Ferrer et al. [81] and Hursin et al. [82] derived an NEM formulation of 1D S_N for the axial solver. In addition, Stimpson et al. [83] suggested a cubic Legendre expansion of the characteristic equation. Namely, the spatial expansion of the characteristic equation is viable. However, we stick to SC in that the axial expansion is one of the sources of instability, and that SC has a benefit that the positivity of solution is assured as far as the source term remains positive.

Since SC requires sufficiently fine meshes to retain accuracy, a subgrid structure is introduced axially as shown in Figure 3.14. The 1D MOC problems are formed for each homogenized pin. Since thick planes are used in planar MOC and 3D CMFD, the axial shape of the sources should be properly reconstructed. It is done by using the flux form function, which is defined as follows:

$$f_i = \frac{\phi_i}{\bar{\phi}_I} = \frac{H_I \phi_i}{\sum_{i \in I} \phi_i h_i} \tag{3.49}$$

where I and i are the coarse mesh and fine mesh indices, respectively, and H is the coarse mesh thickness.

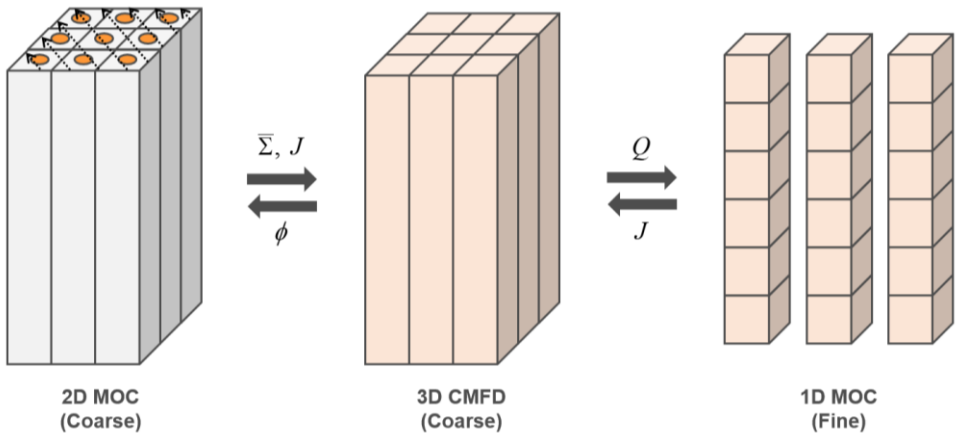


Figure 3.14 Axial grid structures for a radial plane.

The form function is evaluated at the end of the previous axial sweep and is used to reconstruct the initial fine mesh flux and fission source distributions in the next axial sweep using the CMFD updated coarse mesh flux. The solution procedure is shown in Figure 3.15 and proceeds as follows:

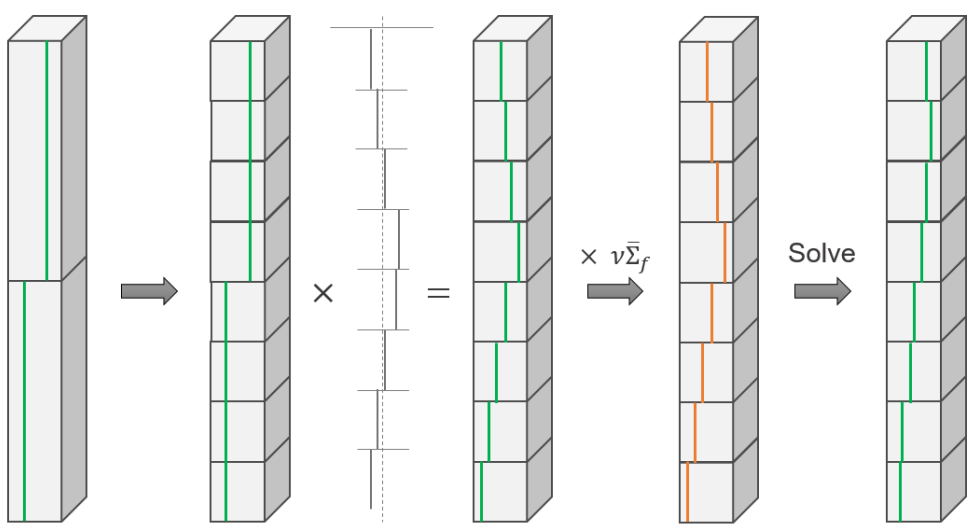


Figure 3.15 Iteration sequence of the 1D MOC axial solver.

1. The coarse mesh average flux is obtained from the CMFD solution.
2. The coarse mesh average flux is multiplied by the flux form function which was determined at the previous axial sweep to reconstruct the fine mesh flux.
3. The fine mesh fission source is calculated by using the reconstructed fine mesh flux and is fixed during the axial sweep. The average fission source of the fine meshes within a coarse mesh preserves the coarse mesh average fission source.
4. Given the reconstructed fines mesh fission source, fixed source iterations (inner iterations) are performed in each pin by successively updating the fine mesh flux and scattering source. The flux form function is updated after performing the designated number of inner iterations (default is 10).

Initially, two types of form functions were considered: fission source and flux. If the fission source form function is used, the memory requirement will be largely reduced compared to using flux form function which is energy-dependent. However, it turned out that a proper reconstruction of the fine mesh flux is necessary after the coarse mesh flux update, due to the existence of self-scattering source in the transport fixed source problem. Because MOC is inherently a slow converging method, wrong initial scattering source results in a substantial increase of the inner iterations. Thus, the flux form function was chosen.

3.4.2 Transverse Leakage Treatment

The radial transverse leakage determined by the 3D CMFD calculation is given in the coarse mesh basis and is treated as isotropic. The shape of the transverse leakage within a coarse mesh is determined by fitting three coarse mesh values by a quadratic polynomial as illustrated in Figure 3.16, which is a conventional way of considering the shape of the transverse leakages in the nodal methods. The coefficients of the quadratic polynomial are given as follows:

$$L(z) = az^2 + bz + c \quad (0 \le z \le h_C) \tag{3.50}$$

where

$$a = 3\frac{\bar{L}_L^{xy}(h_C + h_R) - \bar{L}_C^{xy}(h_L + 2h_C + h_R) + \bar{L}_R^{xy}(h_L + h_C)}{(h_L + h_C)(h_C + h_R)(h_L + h_C + h_R)}, \tag{3.51}$$

$$b = -2\frac{\bar{L}_L^{xy}(2h_C^2 + 3h_C h_R + h_R^2) + \bar{L}_C^{xy}(h_L^2 - 3h_C^2 - 3h_C h_R - h_R^2) - \bar{L}_R^{xy}(h_L^2 - h_C^2)}{(h_L + h_C)(h_C + h_R)(h_L + h_C + h_R)}, \tag{3.52}$$

$$c = \frac{\bar{L}_L^{xy} h_C(h_C^2 + 2h_C h_R - h_R^2) + \bar{L}_C^{xy} h_L(2h_L h_C + 3h_C + h_L^2 h_R + 3h_C h_R + h_R^2) + \bar{L}_R^{xy} h_L h_C(h_L + h_C)}{(h_C + h_R)(h_L^2 + 2h_L h_C + h_C^2 + h_L h_R + h_C h_R)}. \tag{3.53}$$

The quadratic transverse leakage polynomial is then integrated in each fine mesh to form a piecewise constant distribution. It is fed into the source term of the 1D MOC solver.

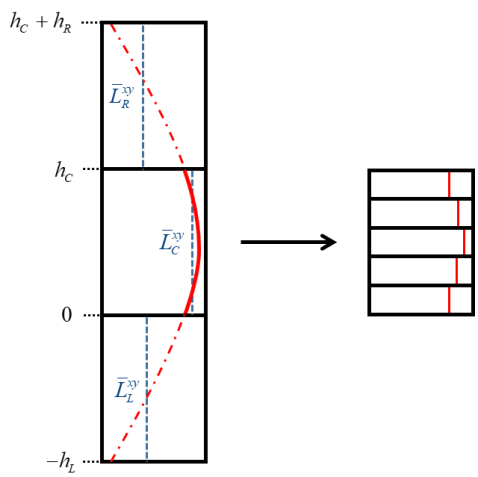


Figure 3.16 Representation of the axial profile of the radial transverse leakage.

3.4.3 Treatment to Prevent Negative Flux

There are two main causes of the negative flux. One is the radial transverse leakage making the total source negative, and the other is the negative self-scattering cross section of hydrogen. Leakage splitting, limited transport correction, and P_L methods can alleviate the problem, but they either deteriorate accuracy or add computational

burdens. Thus, by noting that there are sufficient fission sources in the fuel that will compensate the negative terms, the treatments are selectively applied to the non-fuel regions such that their impacts can be minimized. Details of the schemes and their selective applications are explained in the following.

Transverse Leakage Splitting

Instability can be caused if the transverse leakage is negative since the positivity of flux is not assured with the negative source. It is often observed in non-fuel regions where no fission source can compensate the negative source. Therefore, the leakage splitting scheme is introduced to avoid the negative source. This scheme is to move the leakage term to the left hand side and add to the collision term. Specifically, the leakage term is treated as the absorption term by introducing an equivalent pseudo absorption cross section. The pseudo absorption cross section can be easily obtained by dividing the leakage by the scalar flux. The original leakage splitting scheme is to perform the treatment only when the total source is negative, which leads to the following conditional equations:

$$\begin{cases} \dfrac{d\varphi_g}{ds} + \left(\Sigma_{tr,g} + \dfrac{L_g - Q_g}{\phi_g} \right) \varphi_g = 0 & (L_g > Q_g) \\ \dfrac{d\varphi_g}{ds} + \Sigma_{tr,g} \varphi_g = Q_g - L_g & (L_g \le Q_g) \end{cases} \tag{3.54}$$

However, we perform the splitting with the sign of the transverse leakage instead of the total source, as shown in Eq. (3.55). Since the total source includes scattering, it keeps changing during the fixed source iteration. Since the treatment modifies the total cross section, the exponential term which appear in the characteristic equation should be re-evaluated at each inner iteration. It will increase the computational cost considerably and therefore not preferable. Thus, in order to calculate the exponential terms only once at each axial sweep, only the transverse leakages are split:

$$\begin{cases} \dfrac{d\varphi_g}{ds} + \left(\Sigma_{tr,g} + \dfrac{L_g}{\phi_g} \right) \varphi_g = Q_g & (L_g > 0) \\ \dfrac{d\varphi_g}{ds} + \Sigma_{tr,g} \varphi_g = Q_g - L_g & (L_g \le 0) \end{cases} \tag{3.55}$$

In fact, this scheme has been already applied in the planar MOC calculation for the axial transverse leakage, but it could not be applied to the axial nodal solver since it treats the transverse leakage in a functionalized form. As the axial MOC solver uses subgrids and piece-wise averaged leakages, application of the leakage splitting in the axial solver becomes more viable.

The leakage splitting can be problematic if the flux in the denominator is negative. In such case, the modified total cross section might become negative and it will result in exponentially increasing flux. To prevent this, a positivity condition is imposed to the flux in the treatment; if the flux is negative, the treatment is not performed.

Elimination of Negative Scattering Cross Sections

Due to transport correction, the scattering matrix for hydrogen contains negative self-scattering cross sections, and they can also incur negative sources. Since the 1D MOC problems are formed for each homogenized pin, this issue does not arise in the fuel region where the positive self-scattering terms of fuel materials compensate for the negative terms of hydrogen by homogenization. In addition, this might not be the problem for low energy groups that will have sufficient in-scatter source to make the total source positive. However, for high energy groups in water-dominant cells, the negative self-scattering cross sections can make the source negative. Once negative flux occurs in the high energy groups, cascading effects will occur towards the lower energy groups through the negative down-scatter source. Therefore, two remedies to prevent negative scattering sources are used.

The first remedy is to treat the anisotropic scattering explicitly using the P_L method. The P_L anisotropic scattering source in 1D is expressed as follows:

$$Q_s(\mu) = \sum_{g'} \sum_{l=0}^{L} \frac{2l+1}{2} \Sigma^l_{g',g} P_l(\mu) \phi^l_{g'} \tag{3.56}$$

where the angular flux moment in 1D is written as:

$$\phi^l_g = \int_{-1}^{1} P_l(\mu) \varphi(\mu) d\mu \,. \tag{3.57}$$

Except that the source is now anisotropic and that the angular flux moment needs to be calculated along with the scalar flux, the solution scheme remains the same.

The main problem of applying the P_L method in the homogeneous pin-basis axial solver is that rigorous homogenization of high-order scattering matrix is not possible because it requires higher order angular flux moments. Two difficulties arise when using angular flux moments as the weighting function in the homogenization process: 1. it requires region-wise angular flux moments to be calculated during the planar MOC calculation which is costly, and 2. the angular flux moments can be negative which makes it improper to be used as the weighting function.

However, it should be noted that the regions where the negative self-scattering of hydrogen causes problems are the reflector regions, and the reflector cells are mostly homogeneous. Therefore, by applying the P_L method selectively to the homogeneous water cells in the reflector region, the difficulties regarding homogenization vanishes and the stability issue coming from the negative self-scattering cross section can be effectively eliminated. In addition, it will improve the solution accuracy of peripheral regions where scattering anisotropy is high.

The second remedy is the limited transport correction ($LTCP_0$), which is to move negative self-scattering terms to the left hand side similarly to the leakage splitting scheme as:

$$\begin{cases} \dfrac{d\varphi_{g,p}}{ds} + (\Sigma_{tr,g} - \Sigma_{gg})\varphi_{g,p} = Q_{f,g} + \dfrac{1}{2}\displaystyle\sum_{g' \neq g} \Sigma_{g'g}\phi_{g'} & (\Sigma_{gg} < 0) \\ \dfrac{d\varphi_{g,p}}{ds} + \Sigma_{tr,g}\varphi_{g,p} = Q_{f,g} + \dfrac{1}{2}\displaystyle\sum_{g'} \Sigma_{g'g}\phi_{g'} & (\Sigma_{gg} \geq 0) \end{cases} \tag{3.58}$$

This scheme would ruin the solution accuracy so that the usage should be limited. In MPACT [84], it was applied globally including the planar MOC, but the accuracy degradation was avoided by limiting the application range to over 1 MeV. In our case, no restriction is set to the energy range, because the negative self-scattering term of hydrogen appears in almost all energy groups. Instead, it is applied only to the axial MOC solver and to the cells whose major constituent is water but which are not fully homogeneous; e.g. empty guide tubes or shroud cells, as illustrated in Figure 3.17. For such cells, the P_L method cannot be applied directly due to the heterogeneity and thus $LTCP_0$ is the only practical treatment available. It is expected that the $LTCP_0$ scheme will perform better in such cells than in the homogeneous water cells as the positive self-scattering terms in the structure materials will compensate the negative self-scattering of hydrogen by homogenization.

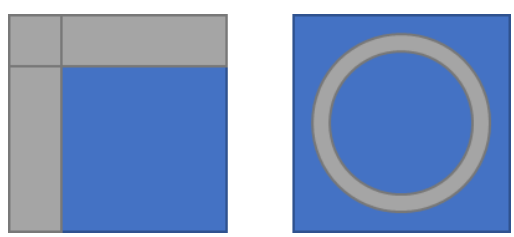


Figure 3.17 Example of heterogeneous water-dominant cells.

3.4.4 Parallelization

The strength of MOC as the axial solver compared to the whole-node SENM is that the axial decomposition is straightforward. For the whole-node SENM solver, a domain shift between the plane-wise decomposition to the pin-wise decomposition is required as illustrated Figure 3.18, which entails heavy MPI communications. On

the other hand, the axial MOC solver employs the same domain with the planar MOC solver and the outgoing angular flux of each domain at the current inner iteration is fed to the neighboring domains as the incoming angular flux of the subsequent inner iteration. Namely, the boundary angular flux data is repeatedly interchanged.

One may think that the nodal methods can be also axially decomposed if one-node or two-node formulations are used. However, the one-node kernel is rather unstable and requires lots of iterations, and the two-node kernel would require an SP_3 CMFD solver to update the second order flux moments, which is unstable and not practical.

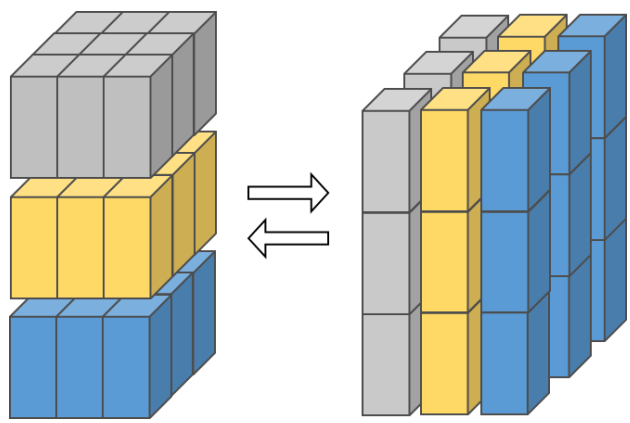


Figure 3.18 Illustration of domain shift between the planar MOC solver and the axial solver for the whole-node SENM axial solution.

GPU acceleration of the axial MOC solver goes as Algorithm 3.5 and Algorithm 3.6 which describe the parallel axial sweep algorithm and the structure of the source update loop for the 1D P_1 MOC case, respectively. The parallelization of 1D P_1 MOC is different from the 2D P_1 MOC. First, the angular flux saving scheme is not used; angular flux moments are computed on-the-fly. Second, the angle-dependent source is not saved; they are calculated on-the-fly using the isotropic sources and the angular source moments. Third, source update is performed on GPU as well, because source update is a major computational overhead in the 1D MOC calculation. To optimize the memory access pattern of the source update kernel, the pin index is placed inner than the group index. Otherwise, the innermost scattering source update loop will

suffer from large strided memory accesses on the flux variable. The indexing scheme is shown in Algorithm 3.6 in Fortran column-major style.

Algorithm 3.5 Parallel axial sweep algorithm.

FOR *g* **PARALLEL DO** Group Sweep
  FOR *xy* **PARALLEL DO** Pin Sweep
    FOR *z* **DO** Axial Mesh Sweep
      FOR *p* **DO** Polar Angle Sweep

$$\tilde{Q}_g(\mu) = \tilde{Q}_g + 3P_1(\mu)\tilde{Q}_g^1$$

$$\Delta\varphi_g(\mu) \;=\; (\tilde{Q}_g(\mu) - \varphi_{in,g}(\mu))(1 - e^{-\Sigma_{t,g}l})$$

$$\phi_g \;+=\; \omega(\mu)(-\Delta\varphi_g(\mu))$$

$$\phi_g^1 \;+=\; P_1(\mu)\omega(\mu)(-\Delta\varphi_g(\mu))$$

      END DO
      Convert to true scalar flux and P_1 flux moment
    END DO
  END PARALLEL DO
END PARALLEL DO

Algorithm 3.6 Parallel source update algorithm.

FOR *z* **PARALLEL DO** Axial Mesh Sweep
  FOR *g* **PARALLEL DO** Group Sweep
    FOR *xy* **PARALLEL DO** Pin Sweep

$$\tilde{Q}(xy, g, z) \;=\; \frac{1}{k_{eff}}\chi(xy, g, z)\psi(xy, z) - L$$

      FOR g' **DO** Group Sweep

$$\tilde{Q}(xy, g, z) \;+=\; \Sigma_s(xy, g, z, g')\phi(xy, g', z)$$

$$\tilde{Q}^1(xy, g, z) \;+=\; \Sigma_s^1(xy, g, z, g')\phi^1(xy, g', z)$$

      END DO

$$\tilde{Q}(xy, g, z) \;/=\; \Sigma_t(xy, g, z)$$

$$\tilde{Q}^1(xy, g, z) \;/=\; \Sigma_t(xy, g, z)$$

    END PARALLEL DO
  END PARALLEL DO
END PARALLEL DO

3.4.5 Initial Verification

Before proceeding to the GPU application of the axial MOC solver, verifications of the accuracy of the method itself and the schemes introduced for stabilization were performed in this subsection.

Verification of P_L and $LTCP_0$ Scattering Treatments

To examine the effect of P_L method and $LTCP_0$ after isolating the effect of radial transverse leakage, a fictitious fuel pin problem with axial reflectors was designed as illustrated in Figure 3.19. The size of subgrids used in the calculations is 0.487cm, which is the nearest lower value of 0.5cm that can divide each coarse mesh.

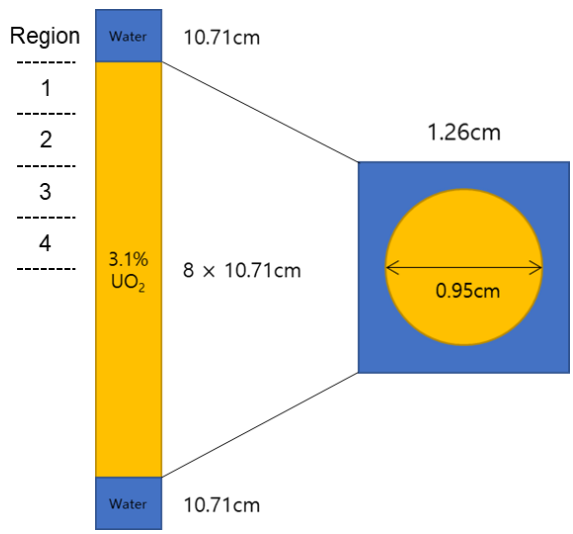


Figure 3.19 Configuration of the fictitious fuel pin problem.

First, the effect of P_L method for the water reflector was examined. Table 3.1 shows the region-wise relative pin power errors of different axial solvers compared to the reference continuous-energy MC result. It can be seen that the power distribution is not sensitive to the scattering order in the reflector, and P_1 shows sufficient accuracy. Therefore, P_1 scattering in the reflector is taken as the default.

Table 3.1 Region-wise relative pin power errors with different axial solvers and scattering expansion orders.

Region	MC	SP_3 Error (SENM)	P_1 Error (MOC)	P_2 Error (MOC)	P_3 Error (MOC)
1	0.6789	-0.78%	0.23%	0.12%	0.10%
2	0.9039	-0.13%	-0.07%	-0.01%	-0.01%
3	1.1454	0.20%	-0.04%	-0.02%	-0.02%
4	1.2718	0.32%	-0.03%	-0.03%	-0.03%

Next, the validity of applying $LTCP_0$ to the heterogeneous non-fuel cells was tested. Stainless steel and zirconium alloy were smeared with various volume fractions in the reflector region, which represents the homogenized structure and guide tube cells to which $LTCP_0$ will be actually applied. Figure 3.20 shows region-wise power errors of different scattering treatments for various reflector compositions. $LTCP_0$ is much better than P_0, but it is still detrimental if it is applied to the pure water. However, as the structural materials get smeared, the negative impacts are reduced. P_1 would be the best choice if available, but $LTCP_0$ can be a decent alternative that can be applied without significant accuracy loss, if used very limitedly.

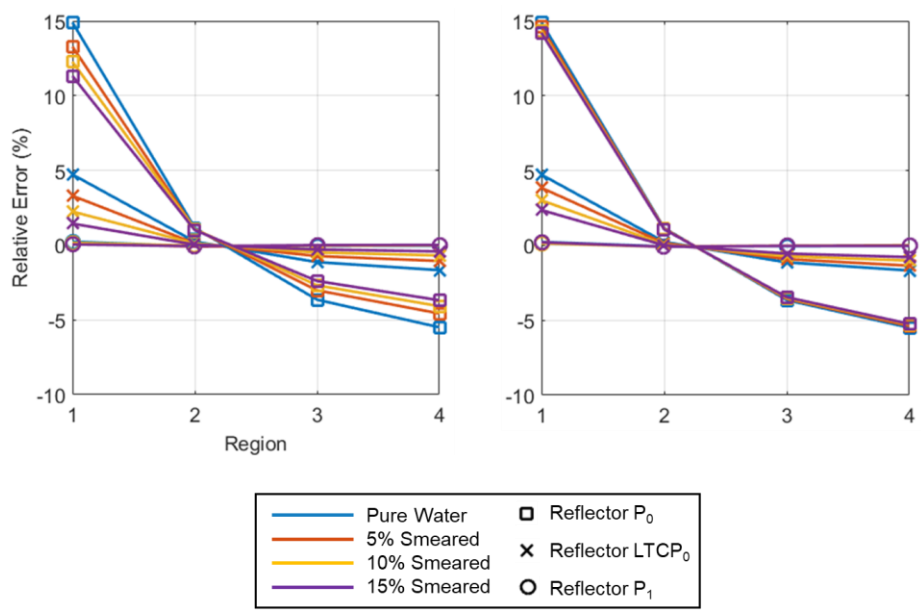


Figure 3.20 Region-wise relative pin power errors of stainless still smeared cases (left) and zirconium alloy smeared cases (right).

Sensitivity Study of Subgrid Size and Verification of Leakage Splitting

The C5G7MOX benchmark [85] is a famous benchmark problem widely used for the verification of transport codes. The cross section set given for the problems does not contain any negative terms, so it is suitable to set a reference case and to perform sensitivity studies. The geometrical description of the Rodded B case is illustrated in Figure 3.21. The core was sliced into 9 planes such that each plane becomes 7.14cm. The plane index is numbered from the top fuel plane to the bottom.

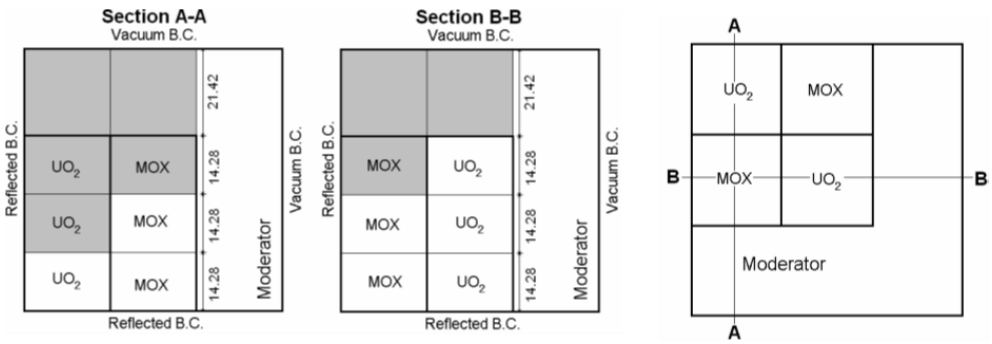


Figure 3.21 Geometry of the C5G7MOX Rodded B case [85]

With the Rodded B case, a parametric study on the size of the subgrids was carried out. As the reference, a multi-group MC solution was obtained employing 100 billion active histories, which gives the 3D power with RMS uncertainty of 0.017%. Figure 3.22 demonstrates the radially integrated axial power errors of different subgrid sizes compared to the MC solution. Four cases were analyzed: 0.1cm, 0.25cm, 0.5cm, 1cm. Note that the actual subgrid size is determined as the nearest lower value which can divide the coarse mesh, and actual subgrid sizes are indicated in the figure. It appears that for sufficiently accurate results, subgrids of less than 0.25cm were required, but 0.5cm subgrid also showed practically acceptable accuracy.

Local 3D power errors of different subgrid sizes were also analyzed, both with the MC solution and the finest mesh solution, which are shown in Table 3.2. In terms of the errors with the MC solution, subgrid sizes below 0.5cm performed comparably,

although it would be the result of error cancellations. When compared to the 0.1cm subgrid case, however, 0.5cm subgrid showed over 2% of relative 3D power error in maximum. However, the peak error occurs at the tip of the core where the absolute power level is very low, and the RMS error is in an acceptable range. Thus, 0.5cm subgrid will be used as the default in the subsequent analyses for practical reasons. Using finer subgrids would require too much memory for GPUs when it goes to core problems.

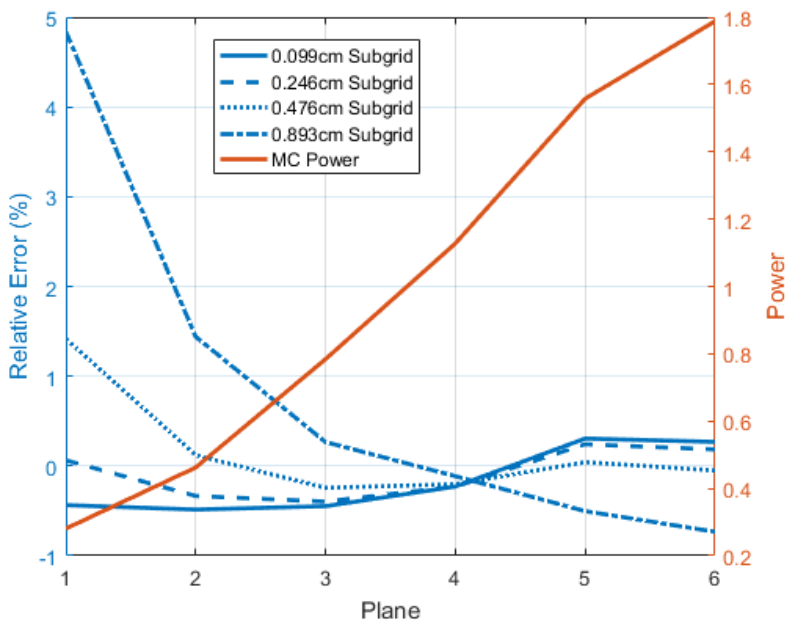


Figure 3.22 Axial power relative errors of different subgrid sizes for the C5G7MOX 3D Rodded B case.

Table 3.2 3D power RMS and maximum relative errors of different subgrid sizes for the C5G7MOX 3D Rodded B case.

Subgrid Size	vs. MC		vs. the Finest Mesh	
	RMS	Max	RMS	Max
0.099cm	0.80%	2.43%	-	-
0.246cm	0.73%	2.38%	0.12%	0.62%
0.476cm	0.63%	3.08%	0.47%	2.30%
0.893cm	1.12%	6.73%	1.43%	6.49%

This problem does not suffer from the negative flux due to the large positive self-scattering cross sections of moderator. Therefore, it was used to investigate the effect of the leakage splitting on the power distributions as well. Table 3.3 summarizes the errors due to using the leakage splitting scheme compared to the normal case without leakage splitting, whereas Table 3.4 provides the plane-wise local pin power errors. The errors are all relative, and the maximum errors occur at the peripheries where the power is low. It is confirmed that applying leakage splitting in the reflector region has negligible impacts on the result.

Table 3.3 Summary of errors due to the leakage splitting scheme for the C5G7MOX 3D Rodded B case.

Eigenvalue Error	11 pcm
Radial Pin Power RMS Error	0.023%
Radial Pin Power Max. Relative Error	0.076%
Axial Power RMS Error	0.041%
Axial Power Max. Relative Error	0.282%

Table 3.4 Plane-wise local pin power errors due to the leakage splitting scheme for the C5G7MOX 3D Rodded B case.

Plane	1	2	3	4	5	6
RMS	0.088%	0.068%	0.029%	0.032%	0.022%	0.010%
Max	0.300%	0.197%	0.092%	0.058%	0.028%	0.024%

Whole-core Solution Stability and Accuracy

The performance of the axial MOC solver for whole-core problems was examined with OPR1000 [10], APR1400 [68], and BEAVRS [66] 3D core problems. All the problems were solved at HZP conditions. The two versions of built-in SENM axial solvers of nTRACER (two-node diffusion and whole-node SP_3) are to be compared. For each core problem, reference continuous-energy MC solution using 19.2 billion histories was obtained.

In order to stabilize the axial SENM solvers, nTRACER has been employing the following damping scheme for the CMFD nonlinear correction factor $\hat{D}$:

$$\begin{aligned} &\text{If } |\hat{D}^{(n+1)} - \hat{D}^{(n)}| > \alpha\tilde{D}, \\ &\hat{D}^{(n+1)} \leftarrow \hat{D}^{(n)} + \frac{\tilde{D}}{|\hat{D}^{(n+1)} - \hat{D}^{(n)}| - \tilde{D}}\left(\hat{D}^{(n+1)} - \hat{D}^{(n)}\right). \end{aligned} \tag{3.59}$$

That is, when the update rate of $\hat{D}$ between two iterations is too drastic compared to the magnitude of the diffusion coupling coefficient $\tilde{D}$, the update is conditionally damped. The value of α is set to 10, which is experimentally chosen.

In addition, there exists an aggressive stabilization technique which skips the axial calculations in the radial reflector pins where the flux levels are very low. In those regions, axial $\hat{D}$ is forced to be zero and the axial solutions are only determined by uncorrected CMFD which would be highly inaccurate. This option is referred to as the ‘reflector FDM’ option.

Table 3.5 summarizes the calculation results of the core problems. ‘Div.’ indicates that the calculation had diverged. Without damping, the axial nodal solvers present fairly unstable behavior, but at least convergence can be achieved in OPR1000 and APR1400 cases as far as damping is used. In case of BEAVRS, however, two-node diffusion SENM fails to converge with any kinds of stabilization options, and only whole-node SP_3 SENM converges with the aggressive ‘reflector FDM’ option. Such unstable behaviors originate from the detailed plenum and vessel descriptions of the BEAVRS core. Namely, the reflector are extended, and the flux levels are extremely low in those regions, which makes the axial solvers unstable.

On the other hand, axial MOC was able to converge all the tested problems without additional stabilization schemes, which demonstrates the robustness of axial MOC. By the eviction of the polynomial expansion, the solution evolves stably, which acts as a natural damping, and the occurrence of negative flux is effectively suppressed

in reflectors by the selective application of leakage splitting, P_L method, and limited transport correction.

Table 3.5 Summary of whole-core calculation results with axial MOC.

Problem	Stabilization	SP_3	Diffusion	MOC	MC
OPR1000	None	Div.	1.03056	1.03060	1.03066
	Damping	1.03059	1.03056		
APR1400	None	Div.	Div.	1.00090	1.00088
	Damping	1.00092	1.00090		
BEAVRS	None	Div.	Div.	1.04324	1.04333
	Damping	Div.	Div.		
	Reflector FDM	1.04380	Div.		
	Damping + Reflector FDM	1.04380	Div.		

Table 3.6 shows the summary of axial power errors and Figure 3.23 to Figure 3.25 illustrate problem-wise detailed axial power errors. In ORP1000 and APR1400 cases, SP_3 SENM performs decently and presents comparable accuracy with axial MOC. In the BEAVRS case, however, SP_3 SENM could not converge normally and shows large errors near the axial reflector interfaces due to the stabilization scheme which deteriorates the accuracy. On the other hand, axial MOC shows only similar degree of errors with other problems. Although SP_3 SENM has lower RMS error than axial MOC, it is likely the result of error cancellation.

Table 3.6 Summary of whole-core calculation axial power errors with axial MOC.

Problem	Error	MOC	SP_3	Diffusion
OPR1000	RMS	0.54%	0.61%	1.02%
	Max.	1.50%	1.97%	3.47%
APR1400	RMS	0.11%	0.18%	0.30%
	Max.	2.33%	1.94%	2.42%
BEAVRS	RMS	0.59%	0.50%	N/A
	Max.	1.82%	6.06%	N/A

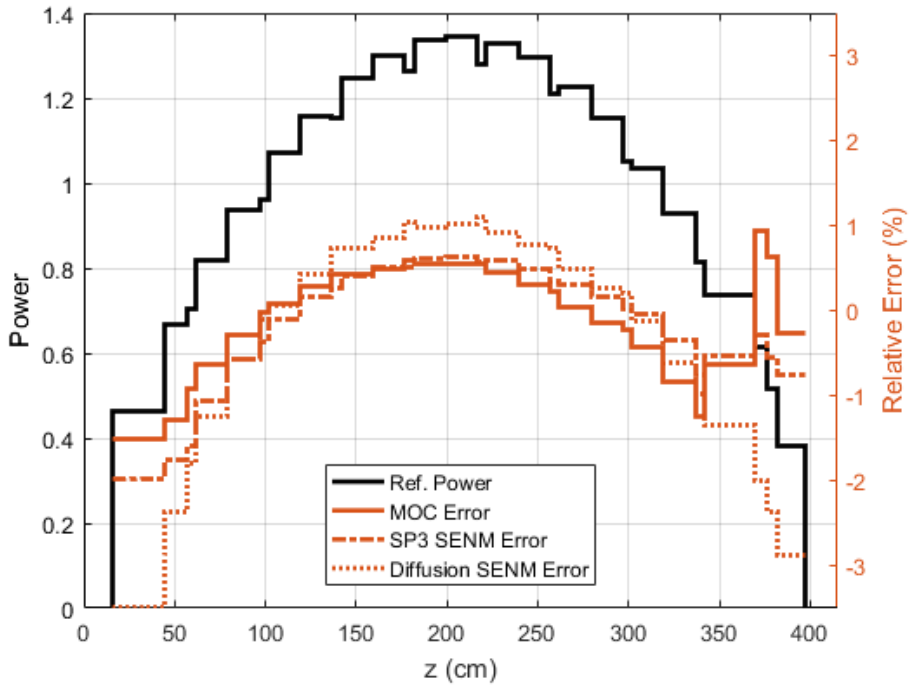


Figure 3.23 Axial power errors of different axial solvers for OPR1000.

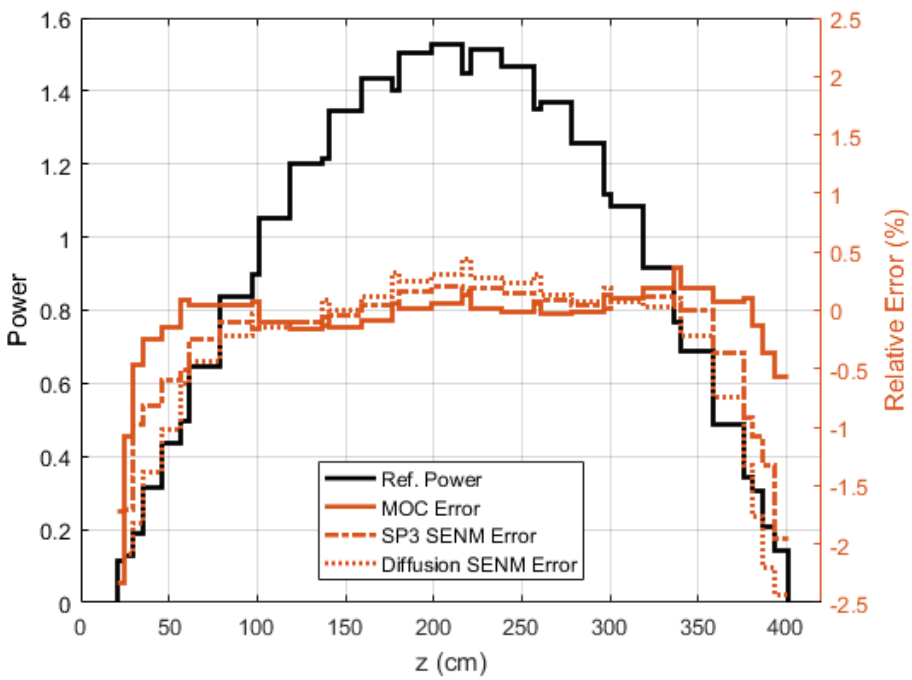


Figure 3.24 Axial power errors of different axial solvers for APR1400.

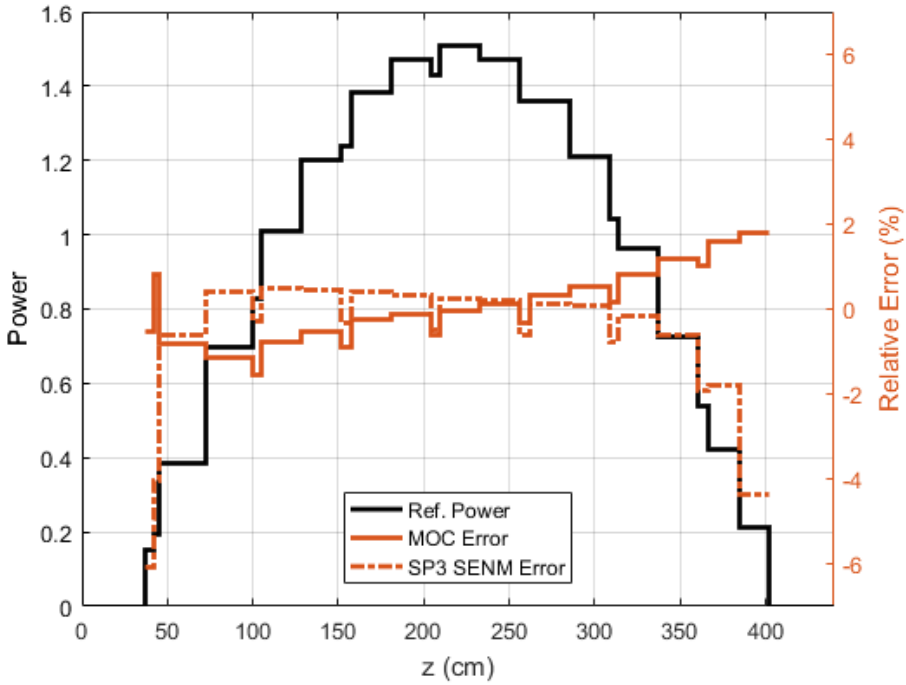


Figure 3.25 Axial power errors of different axial solvers for BEAVRS.

3.5 Global Calculation Scheme

3.5.1 Distributed Parallelization

The distributed memory parallelization of the GPU-accelerated 2D/1D calculation takes advantage of the plane-wise formulation originating from the decoupled planar MOC solution which is the most expensive task. Each plane is decoupled from others and solved independently. It enables a plane-wise distributed parallelization where each GPU takes an MOC plane. The distributed parallelization across multiple GPUs is performed by the MPI constructs while auxiliary CPU calculations are parallelized on shared memory using OpenMP directives. This three-level hybrid parallelization scheme is illustrated in Figure 3.27.

The MPI process topology is illustrated in Figure 3.26. The global communicator is used to communicate data and a local communicator is defined for setting the local ranks which correspond to the device IDs. By using the *MPI_COMM_SPLIT_TYPE* function with *MPI_COMM_TYPE_SHARED* as the argument, MPI can detect which processes belong to the same shared memory node and creates local communicators for each node. Using multiple processes in a shared memory node might result in a redundant allocation of common data, but it reduces the complexity of programming and it is more performance-friendly to the modern Non-Uniform Memory Access (NUMA) computer architectures.

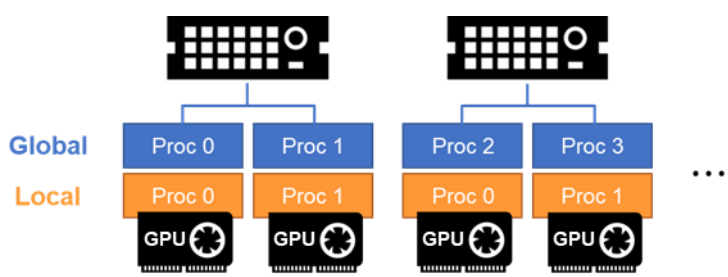


Figure 3.26 MPI process topology.

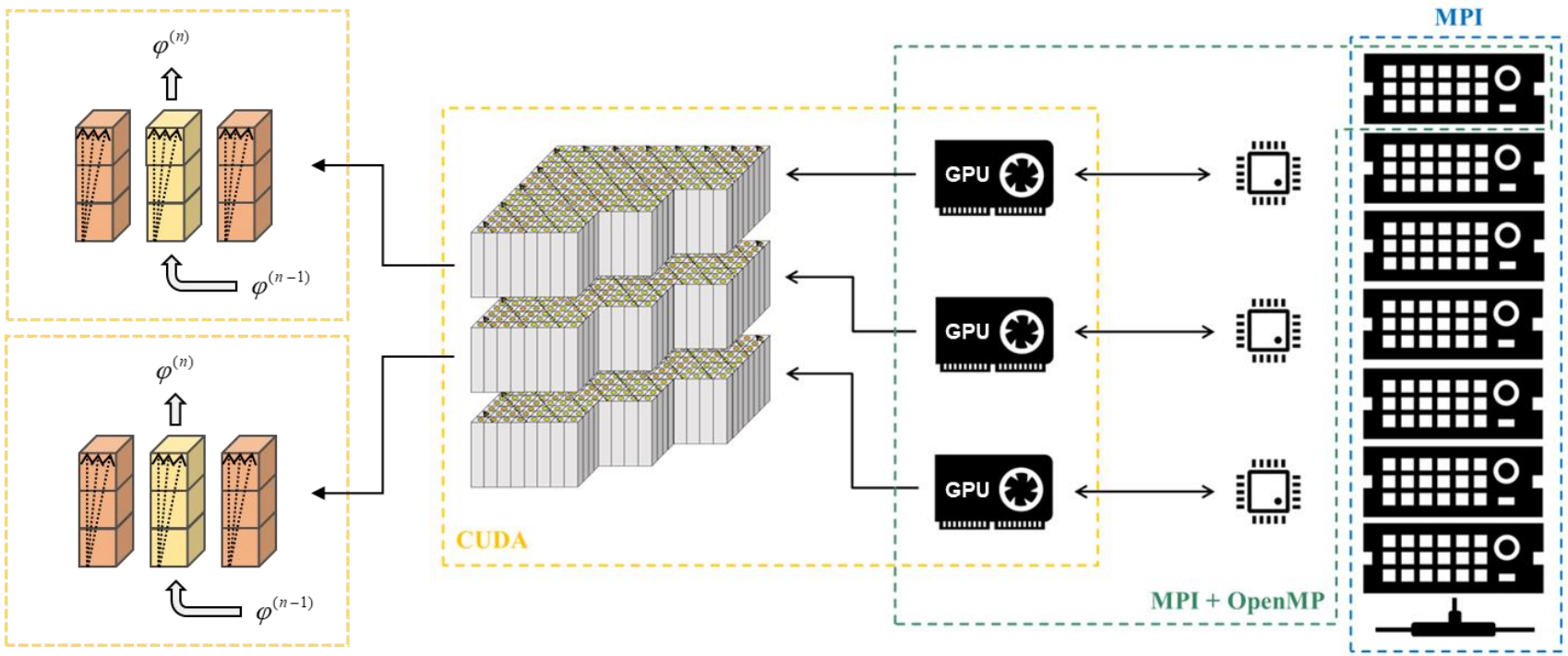


Figure 3.27 Three-level hybrid parallelization based on plane-wise decomposition.

3.5.2 Calculation Flow

The complete calculation flow of the GPU-based steady-state calculation is shown in Figure 3.28. Blue boxes are the calculations solely performed by CPU, while green boxes are the GPU-accelerated ones. The resonance treatment is performed first with Macro-Level Grid (MLG) subgroup method [86]. The ray tracing calculation in the subgroup fixed-source problem (FSP) is offloaded to GPU, while the other resonance treatments are still performed by CPU utilizing the existing routines in nTRACER. After the subgroup calculation, initial CMFD calculation is performed to provide a rough initial guess. In the first CMFD calculation, the axial solver is not called.

After the first CMFD calculation, the global iteration initiates. The 2D MOC solver calculates the sub-pin level flux to calculate the pin-homogenized cross sections and radial currents to be fed to the subsequent 1D MOC and CMFD calculations. Each MOC outer iteration is composed of four ray tracing calculations: two for the entire 47 groups, and the other two for the thermal groups starting from group 24.

Based on the detailed flux solution provided from the 2D MOC calculation, cell homogenization is performed. Note that this process is performed on CPU since it requires access to the cross section data. Then, the 1D MOC calculation is performed on the homogenized pin cells to calculate the axial interface currents for the CMFD calculation. For each 1D MOC sweep, 10 inner iterations are performed.

After the bi-directional MOC sweeps, the surface currents become available and now the CMFD acceleration is performed. The linear system is constructed by CPU in the Compressed Sparse Row (CSR) format and is passed to the GPU-based power iteration module written with cuSPARSE [87] and cuBLAS [88] APIs. The CMFD calculation is further accelerated by two-group condensing; multi-group CMFD only serves as pre- and post-smoothers and the global fission source convergence is driven by the two-group condensed CMFD.

After five times of the multi-group CMFD outers as pre-smoothing, another 1D MOC sweep is performed to update the axial currents by reflecting the updated radial transverse leakages. Then, the linear system is updated and condensed to a two-group system. The two-group CMFD calculation employs Wielandt shift and is converged quickly. The two-group CMFD is escaped when the relative reduction of the fission source error from the first outer iteration is below 0.1, and the accelerated flux is fed to the multi-group CMFD to carry out post-smoothing outer iterations. The global CMFD calculation is also terminated when the relative fission source error reduction is below 0.1. Then, a global convergence check is performed, and if the calculation has not converged, the updated flux is fed to the next 2D MOC calculation, and this loop continues until the global convergence is reached.

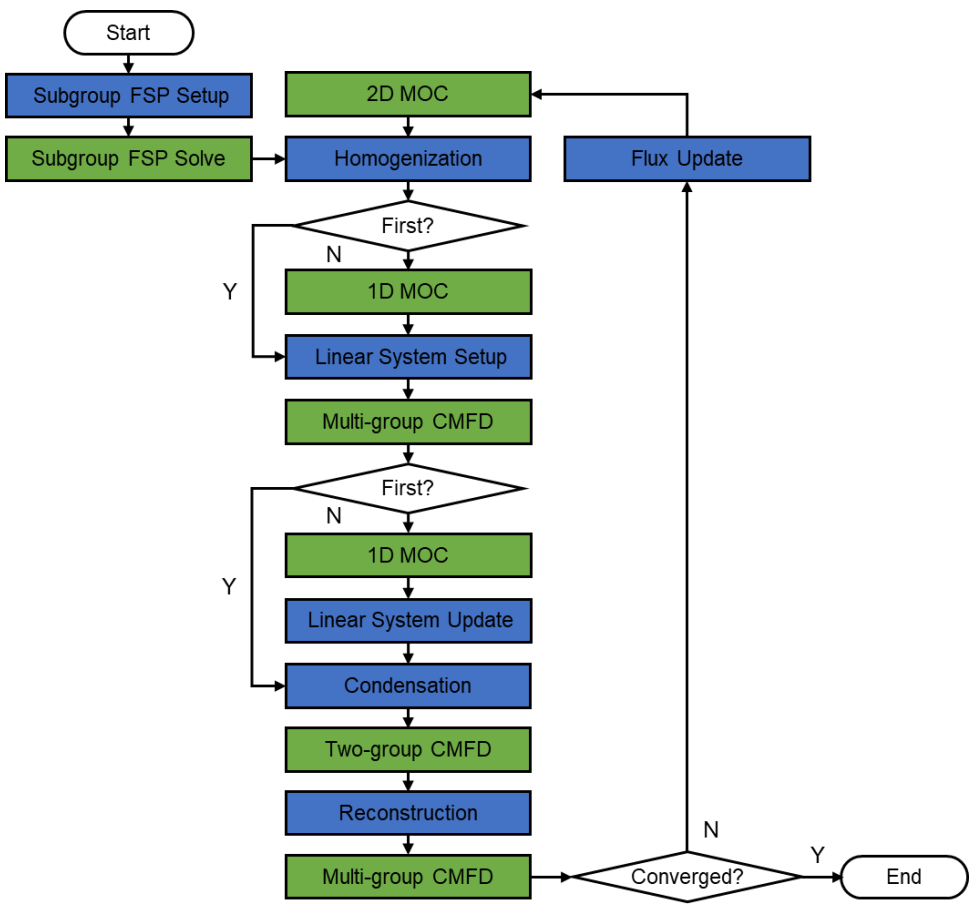


Figure 3.28 Calculation flowchart of GPU-based steady-state calculation.

3.6 Verification and Performance Analysis

In this subsection, the performance of GPU acceleration algorithms is examined with 2D and 3D APR1400 initial core problems [68]. In-house computing clusters named Soochiro are the computing platforms of which the specifications are shown in Table 3.7. Soochiro 3 is a moderate-sized CPU cluster, and Soochiro 4 is a small heterogeneous cluster equipped with consumer-grade GPUs to which academia and industries can easily afford. In the followings, the reference CPU-based calculations will be performed on Soochiro 3 and their performance will be compared to the GPU-based calculations on Soochiro 4. Note that the GPU configuration of the Soochiro 4 cluster is not uniform; first three nodes are equipped with the GTX 1080 GPUs while the other six nodes are mounting the RTX 2080 Ti GPUs which are newer and more powerful. This non-uniformness becomes important in 3D calculations.

Table 3.7 Specification of computing clusters.

Cluster	Soochiro 3	Soochiro 4
of Nodes	27	9
CPU / Node	2 × Xeon E5-2640 v3 (16 Cores, 2.8GHz)	2 × Xeon E5-2630 v4 (20 Cores, 2.4GHz)
GPU / Node	-	4 × GTX 1080 (Node 0 - 2) 4 × RTX 2080 Ti (Node 3 - 8)
RAM / Node	128GB DDR4	128GB DDR4
Interconnect	Intel Omni-path (58Gbps)	Mellanox Infiniband (56Gbps)
Compiler	Intel Fortran 14.0.3	NVFORTRAN 20.7

3.6.1 Planar MOC Calculation

Performance Analysis of the Ray Tracing Kernel

First, the effectiveness of GPU acceleration for the ray tracing calculation, which has been the largest computational burden in the 2D/1D calculation, was investigated for the 2D core problem. Figure 3.29 presents the ray tracing time for the subgroup

calculation on different processors with various ray parameters, and the speedups of GPUs compared to the single-thread case are presented in Figure 3.30. In the MLG subgroup method of nTRACER, a total of 128 independent MOC FSP problems are defined for the fuel, and they are solved as a whole on GPU employing the Jacobi ray tracing kernel as if a single 128-group problem is solved. That is, the ray tracing calculation in subgroup FSP can be considered equivalent to that of a 128-group P_0 neutronics problem.

At the finest ray discretization condition (0.05cm ray spacing, 16 azimuthal angles and 4 polar angles in octant) which is employed in actual calculations, approximately 260 million ray segments are generated, and the ray tracing performance of a single RTX 2080 Ti GPU at this parameter is equivalent to that of 500 CPU cores. The less performant GTX 1080 GPU also shows about 180 times of speedup, which is already very effective in terms of cost.

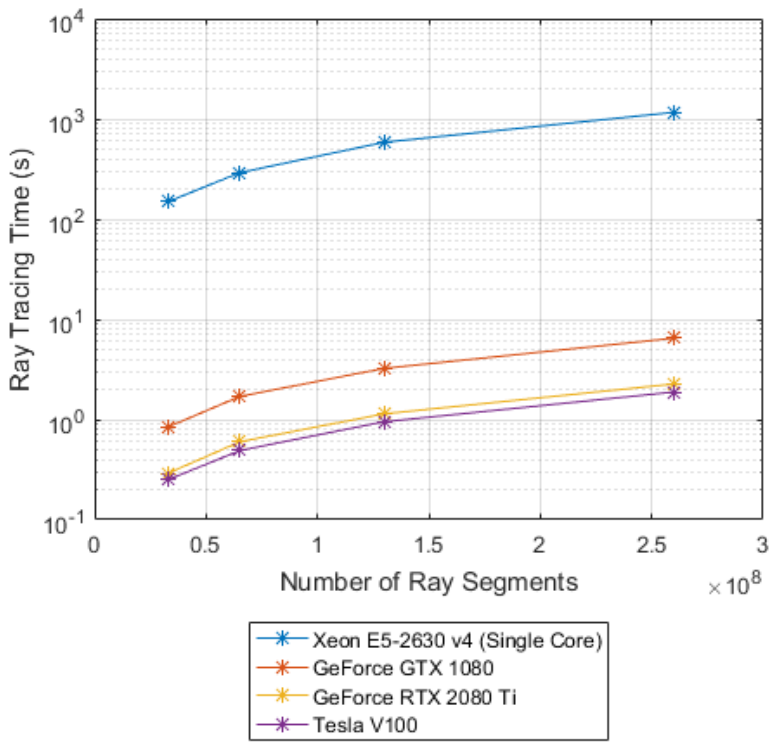


Figure 3.29 Comparison of ray tracing times on different processors.

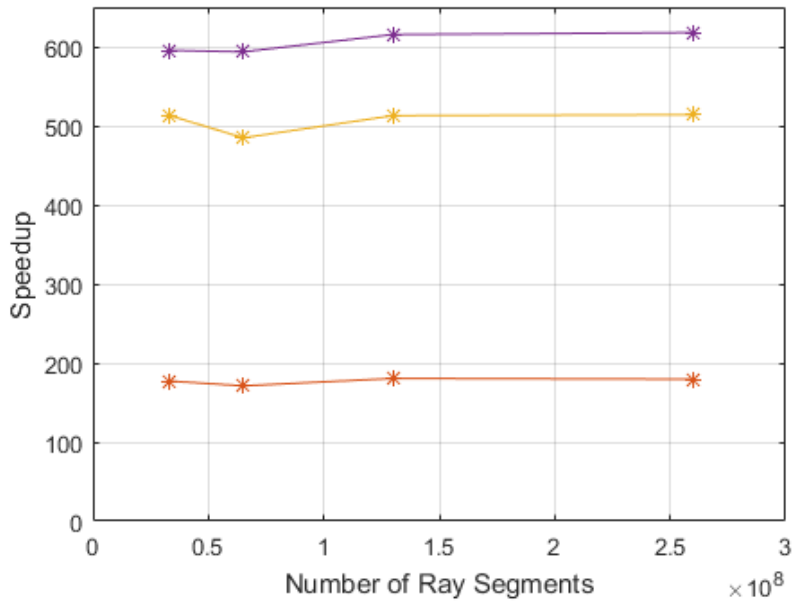


Figure 3.30 Ray tracing speedup ratios of different GPUs.

Next, the computing time of the entire MOC sweep including the source updates was investigated and the results are illustrated in Figure 3.31. In this comparison, the CPU-based calculations employed a single Soochiro 3 node (16 cores) whereas the GPU-based calculations used a single RTX 2080 Ti GPU paired with 20 CPU cores.

While all cases present substantial speedup ratios, much higher speedup ratios are observed in the P_L cases compared to the P_0 case; the speedup ratio of P_0 MOC sweep is 17, while that of P_1 MOC reaches 30. It must be noted, however, that the sweep algorithms of P_0 and P_L cases are different in the CPU-based calculations; P_0 case employed the Jacobi scheme while P_L cases employed the Gauss-Seidel scheme.

The inability of employing the Jacobi scheme in the P_L calculations is the current limitation of nTRACER which employs a ray-based parallelism that replicates flux storage buffers in every thread. Namely, only the Gauss-Seidel scheme can be used for the CPU-based P_L calculations due to the excessive memory requirement in the Jacobi algorithm when it comes to saving all the angular flux thread-by-thread. Thus, the speedups of the P_L cases are overestimated due to the inefficiency of the legacy nTRACER solver.

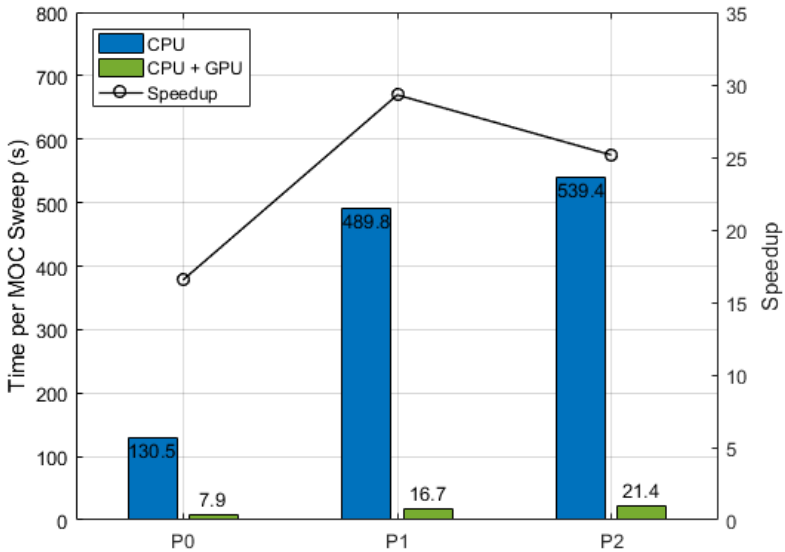


Figure 3.31 Comparison of MOC sweep times between CPU-based and GPU-based calculations.

Then, the remaining issue is whether the use of Jacobi scheme is numerically valid. Figure 3.32 demonstrates the CMFD-accelerated MOC convergence behaviors of the Gauss-Seidel and Jacobi schemes for P_0 and P_2 calculations. For P_0 calculations, the Gauss-Seidel and Jacobi schemes are equivalent as the only quantity being calculated is the scalar flux which is effectively converged by the CMFD acceleration. For P_2 calculations, however, the angular flux moments which the CMFD acceleration is not updating have to be calculated as well, and the Jacobi scheme lags one or two sweeps due to the slower convergence of angular flux moments in the MOC source iteration.

For P_0 calculations, therefore, it is definite that the Jacobi scheme is superior to the Gauss-Seidel scheme as far as the CMFD acceleration is used, which is always the case in practical applications. For P_L calculations, however, the effectiveness of the Jacobi scheme might be debated as there exists an increase of iterations. Nonetheless, we consider that the increase of iterations is tolerable given the high computational efficiency of the Jacobi scheme.

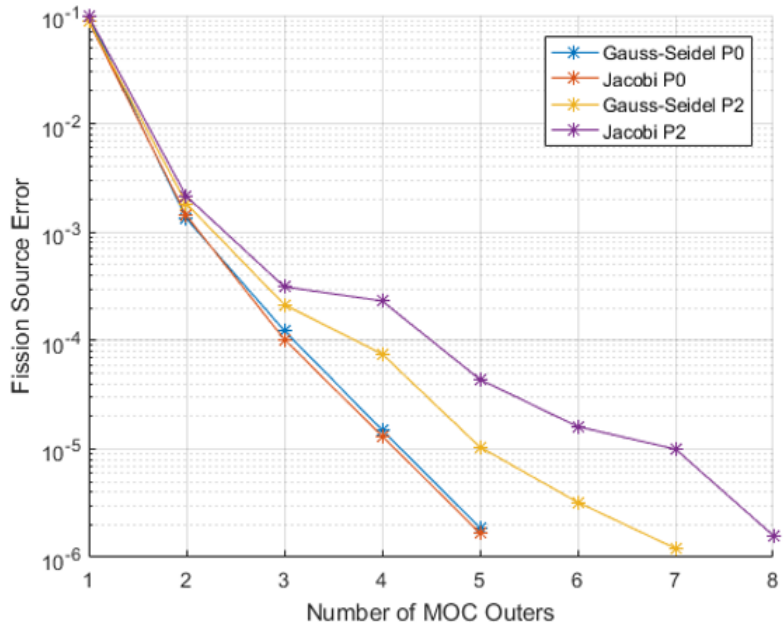


Figure 3.32 Comparison of convergence behaviors of Gauss-Seidel and Jacobi sweep algorithms.

Performance Analysis of the Asynchronous Calculation Scheme

So far the superior performance of GPU-based MOC calculations compared to the CPU-based calculations had been demonstrated. From now on, the effectiveness of CPU − GPU asynchronous calculation scheme is investigated by the performance comparison with the synchronous calculation. Note that the previous results already imply the use of asynchronous calculation scheme.

Figure 3.33 presents the time per MOC sweep of asynchronous and synchronous calculations. The relative computing time of asynchronous calculations compared to the synchronous calculations are indicated. The results show that the computational portion of the source calculation on CPU becomes non-negligible as the ray tracing calculation is effectively accelerated by GPU and that the CPU overheads are being effectively hidden by the asynchronous calculation scheme. The overlapping fraction tends to be larger in the P_L calculations than the P_0 calculation, which is due to the additional calculation of angular source moments by CPU in the P_L calculations.

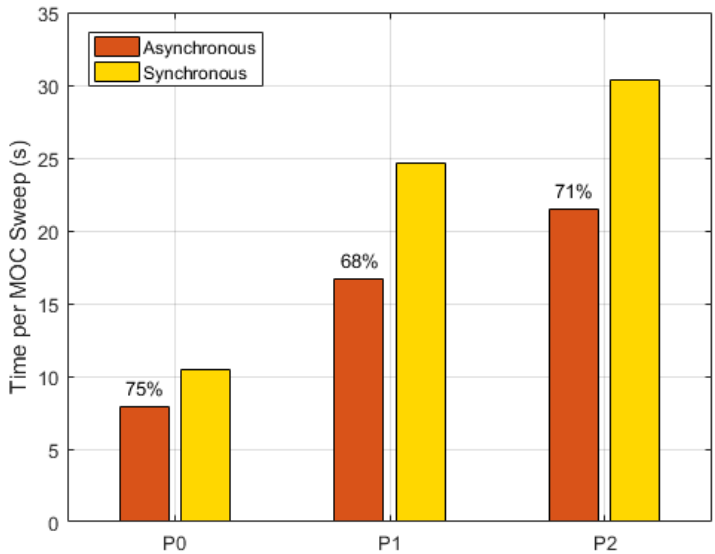


Figure 3.33 Comparison of MOC sweep times of asynchronous and synchronous calculations.

Figure 3.34 and Figure 3.35 present the GPU timelines which show visually how the calculations are proceeding in time. Figure 3.34 is the comparison of timelines of different GPUs for P_1 MOC with asynchronous calculation, and Figure 3.35 shows the timelines of asynchronous and synchronous calculations for P_0 MOC calculation.

From Figure 3.34, the two execution modes depicted in Figure 3.7 can be observed. For the RTX 2080 Ti GPU, the calculation speed of ray tracing for each group block is faster than that of source calculation, and the spinning time of GPU between each group block can be seen. On the other hand, the situation is opposite for the relatively less performant GTX 1080 GPU; CPU finishes its source calculation way before the completion of ray tracing calculation on GPU and begins to spin from the middle of the sweep. As the result, the GPU calculation can proceed without stop and it can be seen that the kernels are being executed consecutively.

From Figure 3.35, the effect of asynchronous calculation can be observed. Without the asynchronous calculation, the ray tracing calculation goes intermittent and almost half of the total computing time is spent by CPU. On the other hand, the GPU can be kept busy during each group block sweep with the asynchronous calculation scheme.

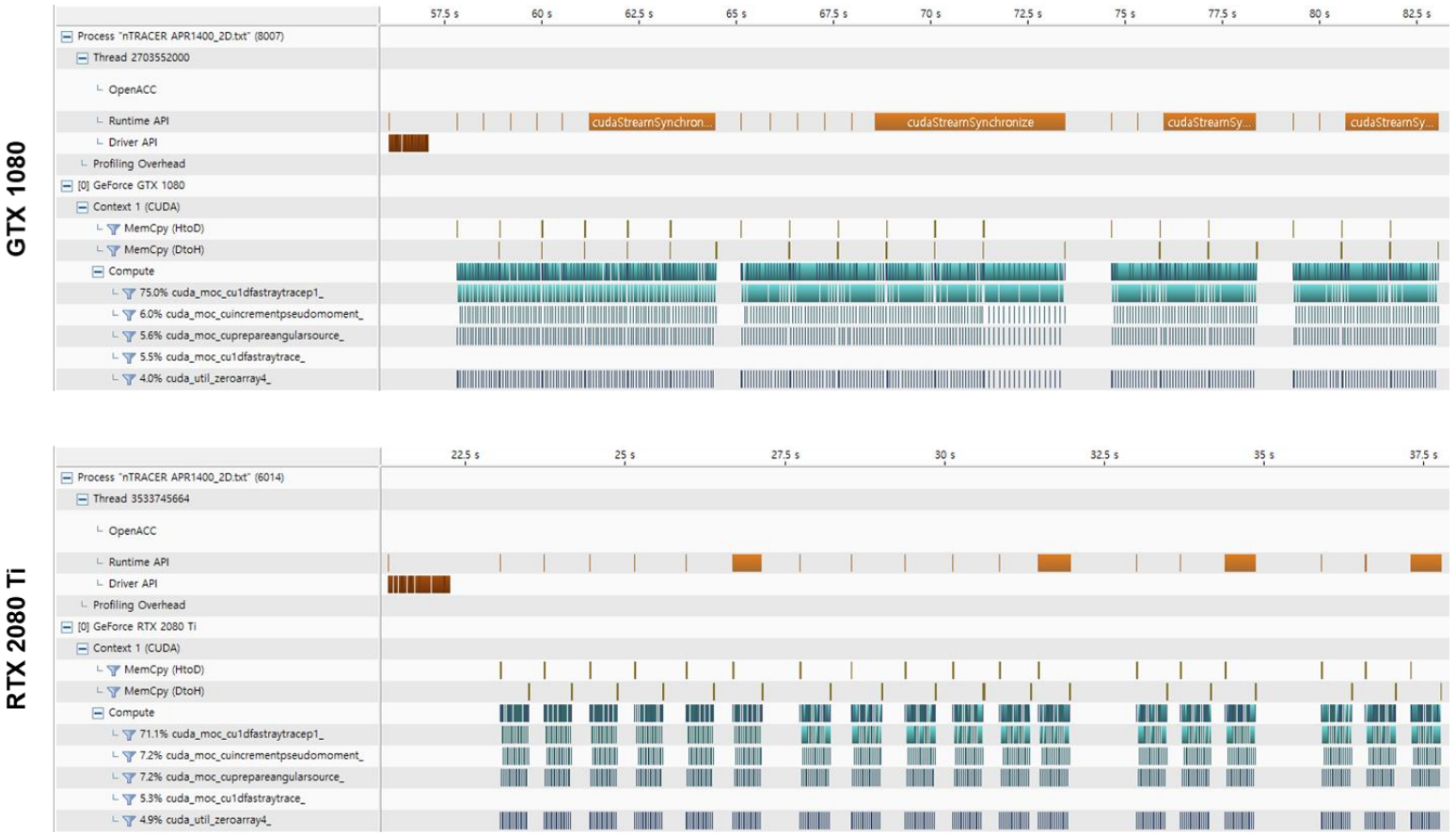


Figure 3.34 Timelines of RTX 2080 Ti and GTX 1080 for P_1 calculation.

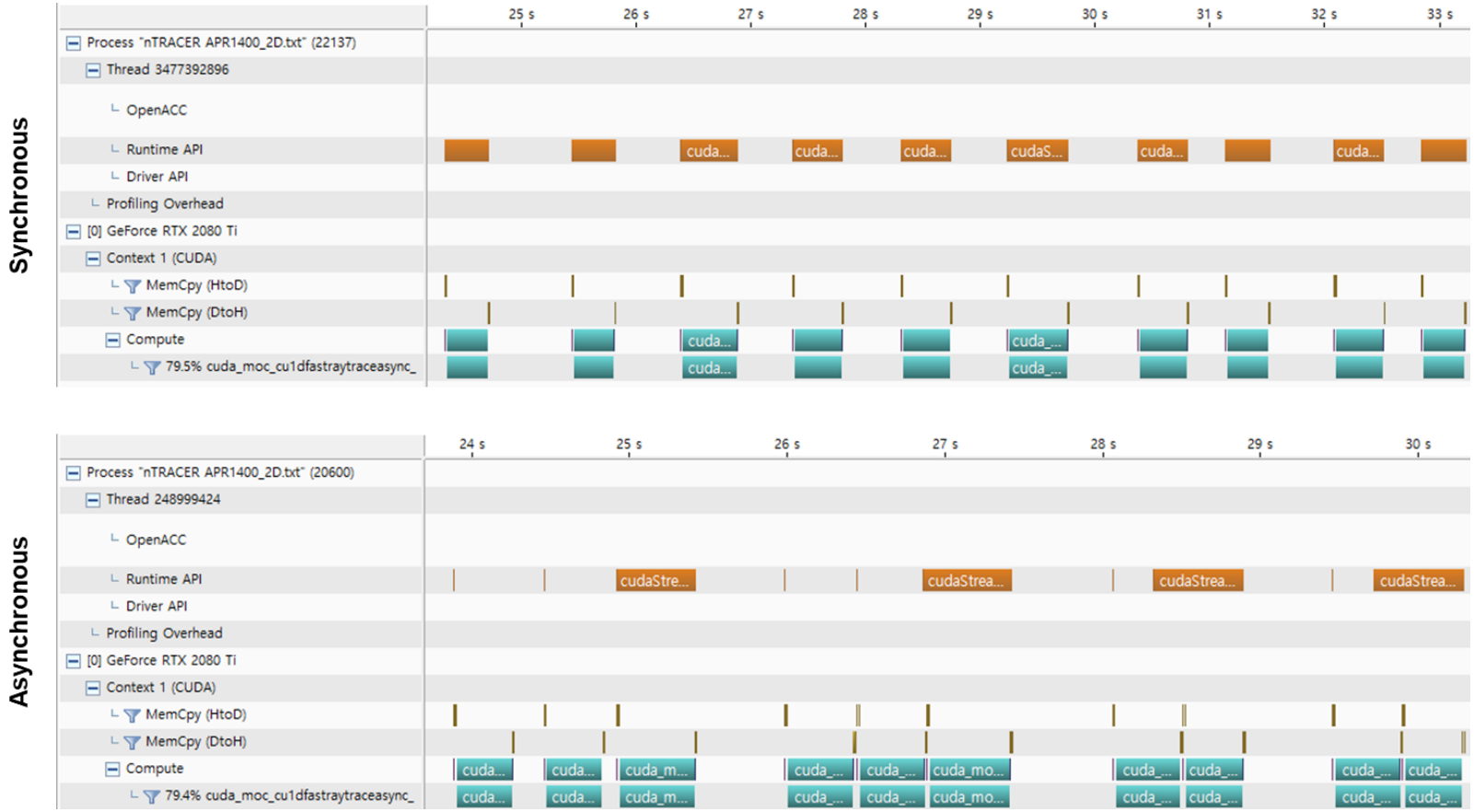


Figure 3.35 Timelines of asynchronous and synchronous modes for P_0 calculation.

3.6.2 Comprehensive Analysis

In this subsection, the performance of P_0 steady-state module is comprehensively studied. The overall computing time comparison of the 2D core calculation is shown in Table 3.8. The CPU-based calculation was performed on a single Soochiro 3 node with 16 cores, and the GPU-based calculation employed a single RTX 2080 Ti GPU paired with 20 CPU cores on Soochiro 4. In both calculations, five MOC sweeps were required for convergence. Note that the ray tracing time in the MOC calculation cannot be measured independently due to the asynchronous execution scheme. The speedups are given with respect to the CPU-based calculation which employs the Jacobi scheme for MOC and the group major ordering scheme for CMFD. The CPU-based CMFD solver was implemented using the Intel Math Kernel Library [89].

Table 3.8 Comparison of computing times (s) of the CPU solver and the GPU solver for the APR1400 2D case.

Architecture	CPU	CPU + GPU	Speedup
Subgroup (Ray Tracing)	97.0 (95.5)	7.1 (2.6)	13.7 (36.7)
MOC	706.1	43.0	16.4
CMFD (Power Iteration)	38.1 (29.7)	15.8 (2.2)	2.4 (13.5)
Total	841.2	65.9	12.8

As a whole, a speedup ratio of 13 is achieved in the total computing time primarily owing to the reduction in the MOC calculation time. In case of subgroup and CMFD calculations, speedups are limited even though their main workloads − ray tracing and power iteration − that had been offloaded to GPUs show considerable speedups. This indicates that the auxiliary CPU operations are now dominating the computing time. In the subgroup calculation, auxiliary resonance stuffs performed by CPUs now take most of the time. For the CMFD acceleration, overall speedup is the lowest as most of the time is spent for cross section homogenization and linear system setup

which are performed by CPUs. Additionally, the inferiority of the NVFORTRAN (previously PGI Fortran) compiler, which is the only compiler that supports CUDA Fortran, compared to the Intel compiler in terms of CPU performance optimization contributes in some degree to reducing the speedup.

In addition, it should be reminded that the speedup of the GPU-based CMFD solver is algorithmically limited due to the adoption of the node major ordering scheme. The linear system of the node major ordering scheme contains approximately 41.5 million nonzeros elements while there exist only 4.7 million nonzeros in the group major ordering scheme because the off-diagonal elements of the scattering matrices are not contained in the linear system.

In terms of accuracy, the validity of the mixed precision approach is confirmed. The relative pin power difference of the CPU-based and GPU-based calculations at fully converged states (relative *l*2-norm of the MOC fission source change being less than 5×10^{-7}) is illustrated in Figure 3.36, which presents only negligible level of errors; the RMS of the difference is only 0.0013%.

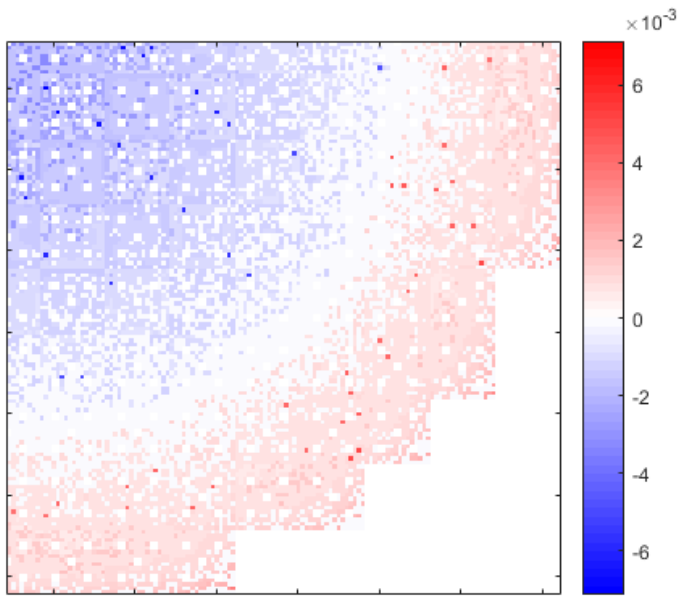


Figure 3.36 Pin power relative difference (%) of the CPU solver and the GPU solver at fully converged states for the APR1400 2D case.

Next, the performance of 3D core calculation is presented. The 3D core consists of 36 MOC planes, which is the maximum size which can be handled by Soochiro 4 containing 36 GPUs in total. The CPU-based calculation was performed by 18 nodes of Soochiro 3 which results in 288 cores and two MOC planes per node, whereas the GPU-based calculation assigns one MOC plane per GPU such that each node solves four MOC planes. Table 3.9 presents the computing times of the 3D core calculations in which five MOC sweeps were required for convergence.

Table 3.9 Comparison of computing times (s) of the CPU solver and the GPU solver for the APR1400 3D case.

Architecture	CPU	CPU + GPU	Speedup
Subgroup	186.0	14.5	12.8
MOC	1315.0	112.7	11.7
CMFD (Power Iteration)	66.0 (49.5)	51.2 (16.7)	1.3 (3.0)
Axial	139.0	25.1	5.5
Total	1706.0	203.5	8.4

The total computing time is reduced by more than 8 times by GPU acceleration which enables to perform direct whole-core calculation based on the 2D/1D method in four minutes with practical amount of resources. The overall speedup, however, is limited compared to the 2D core case.

There are several reasons for this lower performance. First, the performance of the GTX 1080 GPUs which are mounted on the first three nodes of Soochiro 4 are much lower than the RTX 2080 Ti GPUs mounted on the other nodes, and they bound the total computing time. Recall from Figure 3.30 that the ray tracing performance of a GTX 1080 GPU is about three times slower than an RTX 2080 Ti GPU. Second, the number of CPU cores assigned to each GPU for auxiliary operations is reduced. In the 2D calculation, all the available cores in a node were exploited by a single GPU, but now they are shared by four GPUs, which increases the CPU calculation time.

The performance degradation of the CMFD power iteration is the most apparent. It is because the data communication between GPUs has higher latency compared to the host data communication due to the additional transfers through PCI bus. CMFD linear system is fully coupled between processes and frequent data communication is needed for the linear system solution. It is the drawback of using consumer-grade GPUs which lack dedicated data communication functions. However, the computing time portion of the linear system solution is small and the use of consumer-grade GPUs is still justified.

In the aspect of solution accuracy, it is verified that the mixed precision GPU solver yields sufficiently accurate solutions in the 3D calculation. The errors distributions of the pin power and axial power obtained using the same tight convergence criteria with the 2D case are illustrated in Figure 3.37 and Figure 3.38, respectively. The pin power RMS difference is only 0.01%. The axial solver is entirely in single precision except for the homogenized cross sections that are provided by the CMFD module. Therefore, the errors might be larger than the 2D case, but the degree of errors is at a practically negligible order.

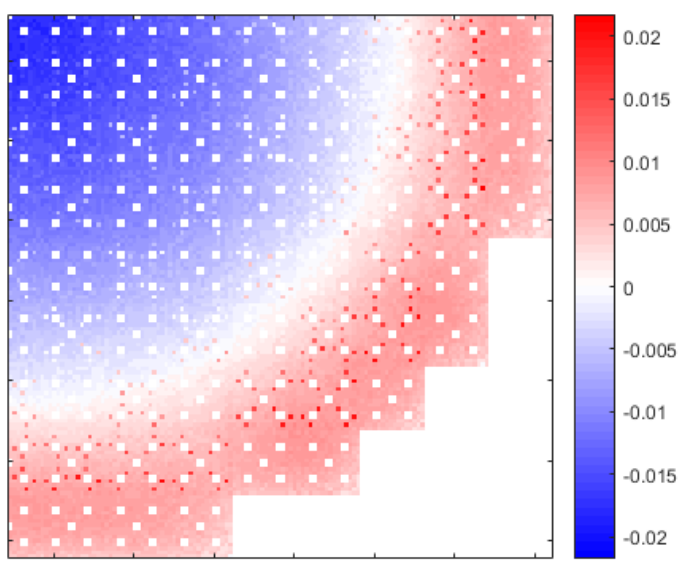


Figure 3.37 Pin power relative difference (%) of the CPU solver and the GPU solver at fully converged states for the APR1400 3D case.

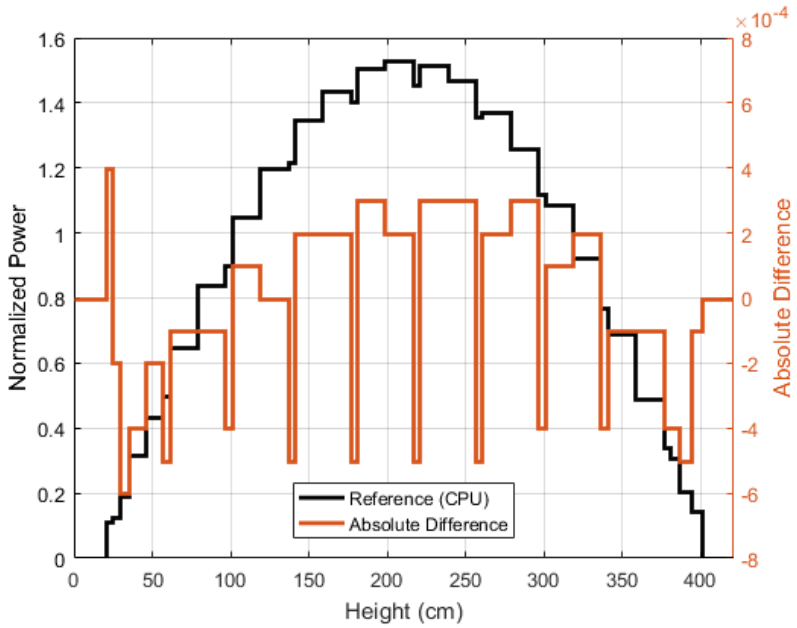


Figure 3.38 Axial power absolute difference of the CPU solver and the GPU solver at fully converged states for the APR1400 3D case.

Lastly, a very rough comparison of performance in terms of system cost and power consumption was made. It must be noted that this comparison is not rigorous as many assumptions are made. Table 3.10 shows the MSRPs and power consumptions of the processors employed for the calculations. The power consumptions of CPUs are the thermal design powers (TDP) which are typically 20 - 30% lower than the peak power, and the power consumptions of GPUs are the actual peak power. The prices and power consumptions of all the other components such as barebones, DRAMs, and interconnects are not included here due to the difficulties in quantifying them.

Table 3.10. Summary of MSRPs and power consumptions of the processors.

Processor	MSRP ($)	Power (W)
Intel Xeon E5-2640 v3	944	90
Intel Xeon E5-2630 v4	671	85
NVIDIA GeForce GTX 1080	599	180
NVIDIA GeForce RTX 2080 Ti	999	250

Based on the computing time in Table 3.9 and the information of Table 3.10, total processor costs and power consumptions were calculated as presented in Table 3.11. The total power consumption was calculated with the conservative assumption that both CPUs and GPUs are always at full load during the calculations, which is never true. Especially for the GPU-based calculation, only one side is at full load for most of the time.

Even with this conservative assumption, the GPU-based calculation shows almost three times higher power efficiency than the CPU-based calculation. In terms of the processor cost, the Soochiro 4 cluster is 1.3 times more expensive than the Soochiro 3 cluster, while the performance gap is more than 8 times. If the costs of barebones, which take a dominant portion in the entire cluster cost, and other components such as DRAMs and interconnects are considered, the price gap will be smaller or might even be reversed.

Table 3.11. Total processor cost and power consumption of each calculation.

Cluster	Soochiro 3 (18 Nodes)	Soochiro 4 (9 Nodes)
Total Processor Cost ($)	33,984	43,242
Total Power (W)	3,240	9,690
Total Power Consumption for the Calculation (MJ)	5.53	1.97

Chapter 4. Monte Carlo Method

Deterministic methods have limitations in the treatment of energy and angle which are essentially continuous. Especially, there exists a fundamental contradiction in the multi-group approximation of deterministic calculations which is that the flux has to be known in advance to generate multi-group cross sections which are in turn used to calculate the flux. As the result, numerous assumptions have to be introduced in the generation of multi-group cross sections. Such limitations in the energy and angle treatments incurs errors which are difficult to characterize and resolve.

On the other hand, continuous-energy MC calculation can treat continuous energy and angle domains without any restrictions and is growing increasingly attractive for direct whole-core calculations. The only obstacle which prevents the practical use of the MC method is the inherent uncertainty, which requires a substantial amount of computing power to be overcome.

In this regard, we have been developing a GPU-based continuous-energy MC code PRAGMA (Power Reactor Analysis using GPU-Based Monte Carlo Algorithm) [53] under the support of the Korea Hydro and Nuclear Power (KHNP) Company (Grant No. 2018-Tech-09). The primary goal of PRAGMA is to enable routine direct whole-core MC calculations with practical time and computing resources by exploiting the power of GPUs, and to realize the goal, PRAGMA employs a specialized geometry model and algorithms for the power reactor analysis. Namely, PRAGMA sacrifices generality in order to deliver an optimal performance for power reactor calculations, but the physics models are implemented as rigorously as possible.

This chapter covers the algorithms of PRAGMA in full detail. More specifically, this chapter consists of 10 sections. Section 4.1 introduces the fundamental theories and methodologies, including the basic MC tracking algorithm, continuous-energy physics, and the mixed precision strategy for MC calculation. Initial verifications on

the soundness of continuous-energy module in PRAGMA and the validity of mixed precision approach are also performed.

Section 4.2 covers the optimization of the cross section look-up algorithm which is the major overhead in the continuous-energy MC calculation. The unionized grid becomes the base look-up algorithm. This method is efficient in that the grid search for each nuclide is replaced by table look-ups after a single search on the unionized grid, but the double index table becomes voluminous when the number of nuclides increases. The strided accesses to the double index is also harmful to the performance. Therefore, a hashing scheme which combines table look-up and linear search in each local grid is introduced to reduce the size of double index table. In addition, within-nuclide temperature-dependent energy grids are collapsed and cross sections are pre-calculated so that the structure of the unionized grid over the temperatures is unified. Interpolation of temperature-dependent cross sections is performed by the stochastic mixing technique, and an interpolation factor is found.

Section 4.3 presents the vectorized tracking algorithms for GPUs. An event-based tracking algorithm is developed and the traditional history-based tracking algorithm is modified by limiting the number of transitions at each kernel. Both the event-based and modified history-based algorithms split the tracking loop into successive stages of kernels, which allows to introduce stream compaction and sorting schemes on the neutrons. Region partitioning and energy sort schemes are introduced to increase the chance of coalescing in the cross section calculations, and the sorting operations are performed more efficiently by the reference remapping scheme which introduces an indirection between threads and the neutrons via a remapping vector. When sorting, the remapping vector is sorted instead of the neutron data so that the data movement is minimized. The conventional bank algorithm using queues is also unavailable on GPUs, and thus an array-based bank algorithm was developed. Fission site buffers and posterior fission neutron sampling scheme are introduced, and a parallel fission site indexing scheme using atomic addition was devised.

Section 4.4 explains a new target velocity sampling scheme named Relative Speed Tabulation (RST) method for an efficient resonance scattering treatment on GPUs. Constant Cross Section (CXS) method cannot treat the resonance scattering properly due to the underlying assumption that the cross section is constant at the vicinity of the incident energy. Doppler Broadening Rejection Correction (DBRC) method and Weight Correction Method (WCM) are the alternatives which can treat the resonance scattering exactly, but they fail to render good performance. DBRC is especially poor on GPUs due to its significantly varying rejection efficiency, and WCM perturbs the neutron weights too severely so that the statistics is spoiled. The RST scheme is a rejection-free method and does not adjust the neutron weights either. Therefore, it can resolve the problems that DBRC and WCM have. It reverses the sampling order such that the relative speed is sampled first from tabulated probability tables and then the target speed is sampled analytically from a truncated CDF which guarantees the cosine to be in the physical range.

Section 4.5 explains the domain decomposition scheme. Large-scale power reactor calculation involving depletion with the ordinary particle decomposition scheme is prevented by the limited memory of GPUs, which necessitates an explicit domain decomposition. The domain decomposition algorithm in PRAGMA automatically partitions domains by wheel clustering scheme which is optimized for power reactor geometries, and employs an inner − outer iteration which overlaps communication and computation and naturally integrates with the vectorized tracking algorithms.

Section 4.6 covers the feedback algorithms. T/H feedback, xenon equilibrium, and critical search algorithms using soluble boron and control rod banks are presented. Especially, the control rod bank search is a unique feature of PRAGMA which cannot be found in other MC codes.

Section 4.7 describes the depletion algorithm. Directly tallying the reaction rates incurs too large overheads and memory requirements. Thus, calculating the reaction rates using the flux spectra becomes the primary approach of PRAGMA. Tallying an

ultra-fine flux spectrum in each region is prohibited by the limited memory of GPUs, so Multilevel Spectral Collapse (MSC) scheme is devised. This scheme tallies fine-group flux spectra of tens of thousands of groups for a coarse geometry domain such as pin, and tallies multi-group flux spectra of hundreds of groups in each region. The fine-group flux spectra are used to generate multi-group cross sections in each coarse geometry domain, and the regions inside employs the multi-group cross sections and their own multi-group flux spectra to calculate the reaction rates.

Section 4.8 presents the localized delta-tracking scheme. It restricts the range of delta-tracking kernel to a fuel pellet, and employs temperature majorant microscopic cross sections and local maximum number densities to determine the maximum cross sections in the pellet. The temperature distribution in each pellet is functionalized by polynomials and treated analytically. The number density variations in the pellet is also treated exactly by retaining the internal submeshes, but the neutrons do not stop at the internal surfaces by delta-tracking and the tracking performance is virtually unchanged from using a single mesh for the pellet.

Section 4.9 explains the fission source convergence acceleration using CMFD and ramp-up schemes. PRAGMA targets to deploy hundreds of millions of particles per cycle to realize massive particle simulations, and at this rate of particles the number of active cycles can be reduced to dozens even for whole-core problems. To perform such massive particle simulations practically, inactive cycle costs should be reduced effectively, and CMFD and ramp-up accelerations are used in hybrid to minimize the inactive cycle costs.

Lastly, Section 4.10 demonstrates an extensive application of PRAGMA to whole-core calculations integrating all the algorithms, which finalizes the chapter. The core problems to be solved include APR1400 [67], VERA problem 5 [90], and BEAVRS [91].

4.1 Theory and Methodology

This section explains the fundamental theories and methodologies of the neutron MC simulation, which includes the basic MC algorithm, continuous-energy physics, and the mixed precision scheme. Initial verification on the soundness of the physics module in PRAGMA and the validity of the mixed precision scheme is also presented.

4.1.1 Algorithms

Implicit Capture

Implicit capture is a variance reduction technique by allowing the existence of 'half neutrons.' In the analog simulation, the contribution of each neutron will be binary, namely one or zero. In the implicit capture method, instead, each neutron is assigned with an initial weight and it is reduced at each collision by the expected absorption probability. The weight of each neutron becomes the contribution of the neutron to the simulation:

$$w \leftarrow w \times \left(1 - \frac{\sigma_a}{\sigma_t}\right). \tag{4.1}$$

As the absorption reaction includes fission, sampling of fission neutrons is also done implicitly. At each collision, the expected fission neutron yield is computed and the fission neutrons are sampled accordingly. However, as the fission neutron yield must be an integer, one of the nearest two integers is used probabilistically such that the average preserves the expected yield. Using the floor operator, this procedure can be expressed as follows:

$$\nu = \left\lfloor \frac{w}{k_{eff}} \frac{\nu\sigma_f}{\sigma_t} + \xi \right\rfloor. \tag{4.2}$$

However, with this scheme the neutron weights never reach zero, so the Russian Roulette technique is introduced to kill the neutron tracking without biasing the game. Specifically, a weight threshold $w_{\min}$ and a probability p to kill the neutrons whose weights are below the threshold are determined. If a neutron has weight lower than the threshold, the neutron is killed by the prescribed probability. If a neutron survives from the roulette, its weight is multiplied by $1/(1-p)$ such that the total simulation weight is preserved. This process can be described by a single equation as follows:

$$w \leftarrow \frac{w}{1-p} \times \lfloor 1 - p + \xi \rfloor . \tag{4.3}$$

Tracking Algorithm and Cycles

Assume that a neutron is flying in a region whose macroscopic total cross section is Σ_t. The probability of the neutron to survive up to the distance l without collision is given as follows:

$$P(l) = \int_0^l \Sigma_t e^{-\Sigma_t x} dx . \tag{4.4}$$

The first thing to do before migrating the neutron is to determine the macroscopic total cross section of the region:

$$\Sigma_t = \sum_{i=1}^{n} N_i \sigma_{t,i} . \tag{4.5}$$

Once the macroscopic total cross section is known, next collision location of the neutron should be determined. The distance to the next collision location is referred to as distance-to-collision (DTC) and can be sampled by the inverse transformation of Eq. (4.4) as follows:

$$l_{\mathrm{DTC}} = -\frac{\ln(1-\xi)}{\Sigma_t} = -\frac{\ln \xi}{\Sigma_t} . \tag{4.6}$$

However, it should be noted that Eq. (4.4) is only valid when the cross section is constant. The neutron may arrive at the material interface where the cross section changes before making a collision. In that case, DTC has to be re-sampled from the interface. The distance to the material interface is called distance-to-surface (DTS), and both DTC and DTS have to be calculated at each neutron migration. The neutron flight length is determined as the smaller value between DTC and DTS:

$$l = \min(l_{\mathrm{DTC}}, l_{\mathrm{DTS}}) . \tag{4.7}$$

If DTS is smaller than DTC, the neutron is moved to the surface and the procedure is repeated from Eq. (4.5). If DTC is smaller than DTS, however, the neutron makes a collision, and the target nuclide with which the neutron has the reaction should be sampled. The probability of each nuclide to make a reaction with the neutron is the relative contribution of the nuclide to the macroscopic total cross section. Namely, the sample formula can be expressed as:

$$\sum_{i=1}^{j-1} \Sigma_{t,i} < \xi \Sigma_t \le \sum_{i=1}^{j} \Sigma_{t,i} . \tag{4.8}$$

After determining the target nuclide j, fission neutrons are sampled by Eq. (4.2) and the reaction type is determined. The probability of which reaction to undergo is given as the relative contribution of the cross section of each reaction to the total cross section of the target nuclide. Due to the implicit capture, however, absorption reactions are excluded from sampling and the absorption cross section is subtracted from the total cross section, since the absorption reactions are treated implicitly by the weight reduction. Therefore, the sampling formula is written as:

$$\sum_{x=1,\, x\neq a}^{y-1} \sigma_{x,\,j} < \xi(\sigma_{t,\,j} - \sigma_{a,\,j}) \leq \sum_{x=1,\, x\neq a}^{y} \sigma_{x,\,j} \,. \tag{4.9}$$

As the summary, the tracking algorithm can be expressed as a flowchart, as shown in Figure 4.1.

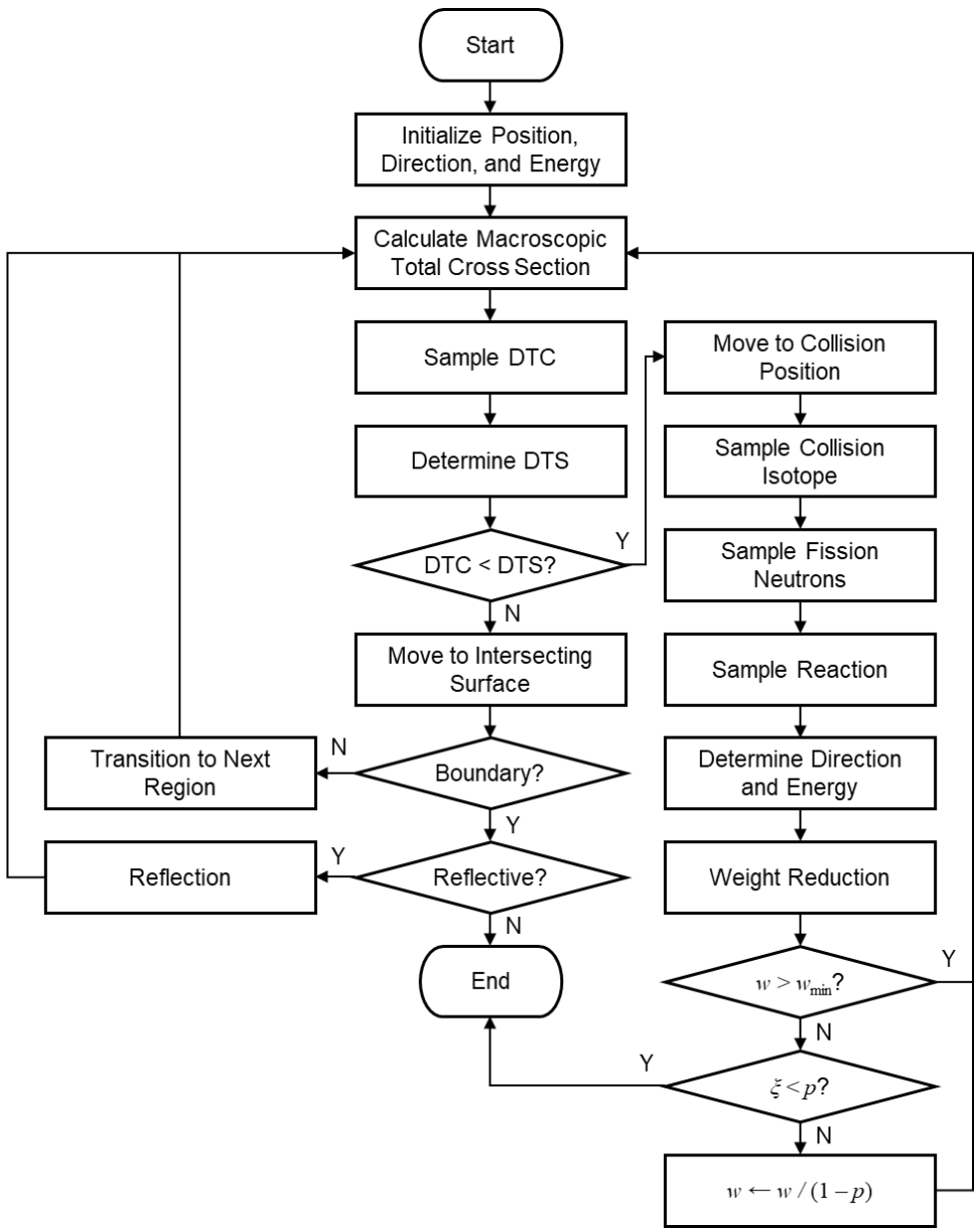


Figure 4.1 MC simulation flowchart.

Simulating a group of neutrons from birth to death according to the aforementioned procedure is called a cycle, which corresponds to a neutron generation in a reactor. The fission neutrons generated during a cycle becomes the source neutrons of next cycle, which continues for the prescribed number of cycles.

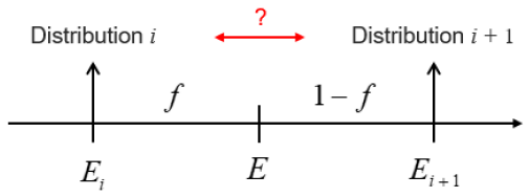
At first, the calculation will begin with some arbitrarily distributed neutrons, which will eventually reach a fundamental mode distribution. Tallies during this evolution of the neutron distribution will give misleading results. Therefore, tallies should be performed after the neutron distribution reaches the fundamental mode. The cycles needed to converge the source distribution are called inactive cycles, and the cycles for producing tallies which are continued after the completion of inactive cycles are called active cycles.

4.1.2 Continuous-Energy Physics

PRAGMA is employing the ACE (A Compact ENDF) format cross section library used by MCNP [14]. The physics of PRAGMA is therefore following the physics of MCNP. Note that many parts of this subsection are referencing the documentation of OpenMC [92] which employs the same ACE format libraries.

Treatment of Tabular Distributions

In the nuclear data library processed by NJOY [93], all the quantities are given as tables. For each nuclide, the outgoing energy and angle distributions are given as probability density functions (PDF) and cumulative density functions (CDF) that are given for several incident neutron energies. If the incident energy E is between E_i and E_{i+1} for which the probability distributions are given, either of the distributions is selected with the probability proportional to the energy distances as illustrated in Figure 4.2, which is called stochastic mixing.



$$f = \frac{E - E_i}{E_{i+1} - E_i} \qquad \begin{array}{ll} \text{if } \xi < f, & \ell = i+1 \\ \text{else}, & \ell = i \end{array}$$

Figure 4.2 Schematic diagram of stochastic mixing.

Once a distribution ℓ is selected for sampling, the next step is to sample a random variable from the distribution. However, the probability distributions are tabular, and to sample a continuous random variable from a discrete probability distribution, appropriate interpolation schemes are necessary.

For the interpolation, either histogram interpolation or linear-linear interpolation is used. Histogram interpolation assumes that the PDF between two points is flat as Eq. (4.10) so that the CDF becomes piece-wise linear function as Eq. (4.11).

$$P_{l,j}(x) = P_{l,j}, \tag{4.10}$$

$$C_{l,j}(x) = C_{l,j} + (x - x_{l,j})P_{l,j}, \tag{4.11}$$

$$x = x_{l,j} + \frac{\xi - C_{l,j}}{P_{l,j}}. \tag{4.12}$$

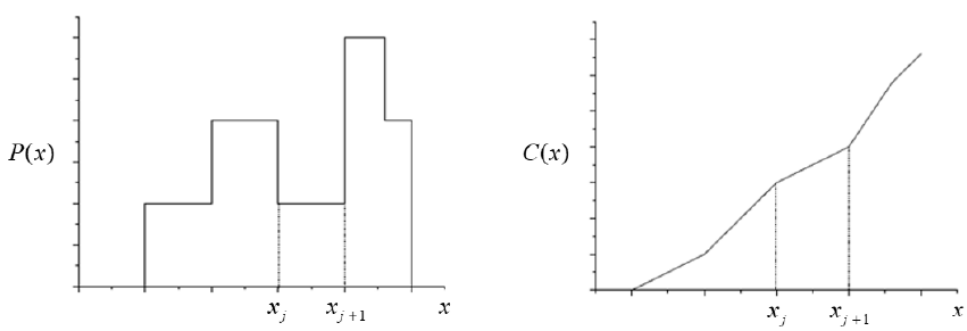


Figure 4.3 Example of PDF and CDF for histogram interpolation.

On the other hand, linear-linear interpolation assumes that the PDF between two points changes linearly as Eq. (4.13) so that the CDF becomes piece-wise quadratic function as Eq. (4.14).

$$P_{l,j}(x) = P_{l,j} + \frac{P_{l,j+1} - P_{l,j}}{x_{l,j+1} - x_{l,j}}(x - x_{l,j}) = P_{l,j} + m(x - x_{l,j}), \tag{4.13}$$

$$C_{l,j}(x) = C_{l,j} + \frac{m}{2}x^2 + (P_{l,j} - mx_{l,j})x + \frac{m}{2}x_{l,j}^2 - P_{l,j}x_{l,j}, \tag{4.14}$$

$$x = x_{l,j} + \frac{1}{m}\left(\sqrt{P_{l,j}^2 + 2m(\xi - C_{l,j})} - P_{l,j}\right). \tag{4.15}$$

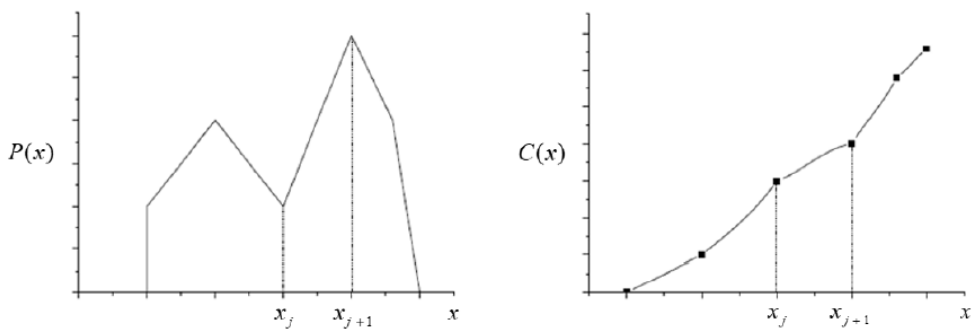


Figure 4.4 Example of PDF and CDF for linear – linear interpolation.

For non-probability distributions such as cross sections, there are five interpolation schemes: histogram, linear-linear, linear-log, log-linear, and log-log. Cross section tables always employ linear-linear interpolation, while it is case-dependent for other distributions.

Determination of Outgoing Energy and Angle

1. Determination of Scattering Angle

The scattering angle of a reaction is given as the cosine of the incoming and the outgoing directions of the neutron. Unless isotropic, the scattering angle distributions

can be given in three different forms: probability tables, S(α, β) tables, and Kalbach-87 formalism.

2. Elastic Scattering

Elastic scattering is a scattering reaction which preserves the kinetic energy of the neutron in the center-of-mass system (CMS). The outgoing energy of a neutron after elastic scattering can be calculated using the classical mechanics. Denoting the velocity of neutron and target nucleus as $\mathbf{v}_n$ and $\mathbf{v}_t$, respectively, the velocity of the center-of-mass $\mathbf{v}_{cm}$ becomes:

$$\mathbf{v}_{cm} = \frac{\mathbf{v}_n + A\mathbf{v}_t}{A+1} \tag{4.16}$$

where A is the atomic weight ratio (AWR). Using the center-of-mass velocity, the velocity of neutron in CMS is calculated as:

$$\mathbf{v}_{n,cm} = \mathbf{v}_n - \mathbf{v}_{cm} . \tag{4.17}$$

The scattering angle distribution of elastic scattering is given from external tables. Assuming for now that the direction of neutron in CMS after scattering was already determined as Ω_{cm}, the outgoing velocity of neutron in CMS can be expressed as:

$$\mathbf{v}'_{n,cm} = \left\|\mathbf{v}_{n,cm}\right\| \Omega_{cm} . \tag{4.18}$$

Since $\mathbf{v}_{cm}$ is invariant in the elastic scattering, the outgoing velocity of neutron in the lab system can be obtained as:

$$\mathbf{v}'_n = \mathbf{v}'_{n,cm} + \mathbf{v}_{cm} . \tag{4.19}$$

3. Discrete Level Scattering (ACE LAW = 3)

Inelastic scattering is a reaction where the neutron is absorbed and re-emitted with some of its energy being used to excite the target nucleus. Technically all the energy levels of a nucleus are discrete, but high energy levels are so dense that they are not distinguishable. Therefore, the descriptions for inelastic scattering are given in two forms: discrete level scattering (MT = 51 ~ 90) and continuum level scattering (MT = 91). For the continuum level scattering, the outgoing energy distributions are given as tables or spectra. For the discrete level scattering, however, the outgoing energy can be determined exactly, as follows:

$$E' = \left(\frac{A}{A+1} \right)^2 \left(E - \frac{A+1}{A} Q \right) \tag{4.20}$$

where Q is the Q-value of the reaction which corresponds to the excitation energy of the level.

4. Tabular Outgoing Energy (ACE LAW = 4)

This law describes the outgoing energy distributions as probability tables, and most reactions follow this description. As explained, the stochastic mixing is performed to choose the distribution ℓ, and the outgoing energy is sampled using appropriate interpolation rules. However, stochastic mixing might induce non-physical results. Each distribution specifies the lower and upper bounds of the outgoing energy which are physically determined for a specific incident energy. Hence, using a distribution which was intended to be used for a different incident energy will likely result in an outgoing energy physically out-of-range for the given incident energy.

In this regard, the scaled interpolation is introduced to rescale the outgoing energy. First, the lower and upper energy bounds are adjusted by linear interpolation of the energy bounds in two adjacent distributions:

$$E_{\min} = E_{i,\min} + f(E_{i+1,\min} - E_{i,\min}), \tag{4.21}$$

$$E_{\max} = E_{i,\max} + f(E_{i+1,\max} - E_{i,\max}). \tag{4.22}$$

Then, the sampled outgoing energy is rescaled proportionally within the adjusted energy bounds:

$$E' \leftarrow E_{\min} + \frac{E' - E_{\ell,\min}}{E_{\ell,\max} - E_{\ell,\min}}(E_{\max} - E_{\min}). \tag{4.23}$$

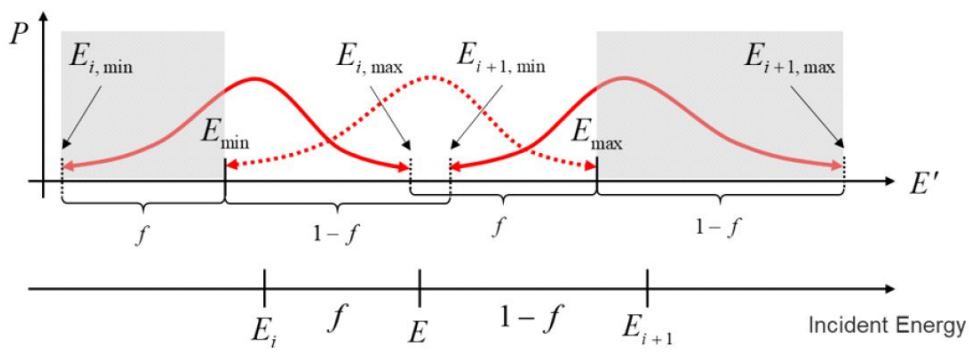


Figure 4.5 Schematic diagram of scaled interpolation.

5. Maxwell Spectrum (ACE LAW = 7)

This law describes the outgoing energy distributions by the following Maxwell spectrum:

$$P(E') = C\sqrt{E'}e^{-E'/T} \tag{4.24}$$

where T is the nuclear temperature, which is given as a table with respect to the incident energy.

The sampling scheme is adopted from C64 of the Monte Carlo sampler [94]. Using three independent random numbers, the outgoing energy is sampled as Eq. (4.25), but the outgoing energy is accepted only when Eq. (4.26) is satisfied, where U is the restriction energy. If rejected, the sampling is repeated.

$$E' = -T\left[\log(\xi_1) + \log(\xi_2)\cos^2\left(\frac{\pi}{2}\xi_3\right)\right], \tag{4.25}$$

$$0 \le E' \le E - U\ . \tag{4.26}$$

6. Evaporation Spectrum (ACE LAW = 9)

This law describes the outgoing energy distributions by the following Evaporation spectrum:

$$P(E') = CE'e^{-E'/T}\ . \tag{4.27}$$

The sampling is performed by a rejection sampling scheme which is adopted from LA-UR-14-27694 [95]. With two independent random numbers, the outgoing energy is sampled as follows:

$$E' = -T\log((1-g\xi_1)(1-g\xi_2)) \tag{4.28}$$

where

$$g = 1 - \exp\left(-\frac{E-U}{T}\right). \tag{4.29}$$

The sampled outgoing energy is accepted only when Eq. (4.26) is satisfied.

7. Energy-Dependent Watt Spectrum (ACE LAW = 11)

This law describes the outgoing energy distributions by the Watt spectrum in the following:

$$P(E') = Ce^{-E'/a}\sinh\sqrt{bE'} \tag{4.30}$$

where a and b are given as the tables with respect to the incident energy.

The sampling scheme was suggested by Romano [96]. First, W is sampled from Eq. (4.25) by setting $T = a$. Then the outgoing energy is sampled as follows:

$$E' = W + \frac{a^2 b}{4} + (2\xi - 1)\sqrt{a^2 b W} \,. \tag{4.31}$$

The sampled outgoing energy is accepted only when Eq. (4.26) is satisfied.

8. Kalbach-87 Formalism (ACE LAW = 44)

This law describes correlated outgoing energy-angle distributions. The outgoing energy distributions are given as probability tables, and its sampling is analogous to LAW = 4. Once the outgoing energy is determined, the probability distribution of the outgoing angles is expressed by the following Kalbach-Mann distribution:

$$P(\mu) = \frac{A}{2\sinh(A)}[\cosh(A\mu) + R\sinh(A\mu)] \tag{4.32}$$

where R is the precompound factor and A is the angular distribution slope, which are both given as tables with respect to the outgoing energy.

The sampling of Eq. (4.32) adopts C39 and C40 of the Monte Carlo sampler [94]. First, a random number is sampled, and depending on its relative value with *R*, either of the formula in the following is used:

$$\mu = \begin{cases} \dfrac{1}{A}\log\left(T + \sqrt{T^2 + 1}\right) & (\xi_1 > R) \\ \dfrac{1}{A}\log\left(\xi_2 e^A + (1 - \xi_2)e^{-A}\right) & (\xi_1 \le R) \end{cases} \tag{4.33}$$

where

$$T = (2\xi_2 - 1)\sinh(A) \,. \tag{4.34}$$

9. Correlated Tabular Outgoing Energy-Angle (ACE LAW = 61)

This law also describes correlated outgoing energy-angle distributions, but in this law the scattering angle distributions are given as probability tables. Namely, for each outgoing energy, tabular outgoing angle distributions are specified. Sampling of outgoing energy is done by LAW = 4, and the outgoing angle is sampled from the probability table defined for the sampled outgoing energy.

10. N-Body Phase Space Distribution (ACE LAW = 66)

This law describes the outgoing energy distributions by the N-Body Phase Space distribution in the following:

$$P(E') = C\sqrt{E'}(E^{\max} - E')^{(3n/2)-4} \tag{4.35}$$

where $E^{\max}$ is the maximum energy a particle can have in CMS. It is calculated as follows:

$$E^{\max} = \frac{A_p - 1}{A_p}\left(\frac{A}{A+1}E + Q\right) \tag{4.36}$$

where A_p is the sum of the AWR of outgoing particles. For $n = 3$, the sampling is done by R28 and C64 of the Monte Carlo sampler [94]. First, x and y are sampled from Eq. (4.25) by setting $T = 1$. Then, the outgoing energy is calculated as:

$$E' = \frac{x}{x+y}E^{\max}. \tag{4.37}$$

11. S(α, β) Tables

The scattering reactions below 4 eV tend to be affected by the chemical bindings of the target nuclei, which has large impacts especially for moderating materials such

as hydrogen. S(α, β) table provides the thermal scattering data considering the effect of chemical bindings. Since the base library only provides the cross sections of free atoms, the free atom scattering cross section is replaced with the thermal scattering cross section and the total cross section is recalculated as follows:

$$\sigma_t \leftarrow \sigma_t - \sigma_s + \sigma_{thermal} . \tag{4.38}$$

S(α, β) table describes three types of thermal scattering reactions: coherent elastic scattering, incoherent elastic scattering, and inelastic scattering. However, only the inelastic scattering treatment was implemented in PRAGMA and will be explained here, as the other reactions are not important in LWRs. Coherent elastic scattering is significant only in crystalline materials such as graphite or beryllium, and incoherent elastic scattering is important only in hydrogenous solids such as polyethylene.

The outgoing energy distribution of thermal inelastic scattering can be represented in three ways depending on the ACER input parameter of NJOY [93]: continuous, equiprobable, and skewed equiprobable tables. For the continuous table formulation, the sampling scheme is analogous to LAW = 61. For the equiprobable and skewed equiprobable table formulations, a preset number of discrete outgoing energies are given for each incident energy, whose probabilities follow Table 4.1.

Table 4.1 Outgoing energy probability table for inelastic scattering in S(α, β) table.

Outgoing Energy	E_1	E_2	E_3	…	E_{n-2}	E_{n-1}	E_n
Equiprobable	1	1	1	…	1	1	1
Skewed Equiprobable	1	4	10	…	10	4	1

Using the prescribed probability table, an outgoing energy bin is sampled and then interpolated between values corresponding to neighboring incident energies:

$$E' = E_{i,j} + f(E_{i+1,j} - E_{i,j}) \tag{4.39}$$

where i and j are incident and outgoing energy bin indices, respectively.

For each outgoing energy, a preset number of equiprobable outgoing angles are given regardless of the outgoing energy representation. An outgoing angle bin is sampled uniformly and, if the outgoing energy formulation is either equiprobable or skewed equprobable table, the outgoing angle is interpolated between values corresponding to neighboring incident energies as was done for the outgoing energy:

$$\mu = \mu_{i,\,j,\,k} + f\,(\mu_{i+1,\,j,\,k} - \mu_{i,\,j,\,k}) \tag{4.40}$$

where k is the scattering cosine bin index.

12. Coordinate Transformation

If the outgoing angle and energy distributions are given in CMS, a transformation to the lab system is required. Denoting the scattering angle and the outgoing energy determined in CMS as μ_{cm} and E'_{cm}, respectively, the outgoing energy in the lab system E' is computed as:

$$E' = E'_{cm} + \frac{E + 2\mu_{cm}(A+1)\sqrt{EE'_{cm}}}{(A+1)^2}. \tag{4.41}$$

Then, the scattering angle in the lab system μ is calculated as:

$$\mu = \mu_{cm}\sqrt{\frac{E'_{cm}}{E'}} + \frac{1}{A+1}\sqrt{\frac{E}{E'}}. \tag{4.42}$$

Once the scattering angle in the lab system is determined, it should be converted to the directional vector. First, ϕ is sampled uniformly in [0, 2π). Then, denoting the incoming direction vector as $\Omega = (u, v, \omega)$, the outgoing vector $\Omega = (u', v', \omega')$ is determined as:

$$u' = \mu u + \frac{\sqrt{1-\mu^2}\,(u\omega\cos\phi - v\sin\phi)}{\sqrt{1-\omega^2}} \tag{4.43}$$

$$v' = \mu v + \frac{\sqrt{1-\mu^2}\,(v\omega\cos\phi + u\sin\phi)}{\sqrt{1-\omega^2}} \tag{4.44}$$

$$\omega' = \mu\omega - \sqrt{1-\mu^2}\sqrt{1-\omega^2}\cos\phi \tag{4.45}$$

4.1.3 Mixed Precision Technique

The basic strategy for the mixed precision calculation is to use double precision only for the tallies and use single precision for the remaining operations. However, blindly using single precision causes instabilities induced by numerical errors. Thus, special treatments are needed to make the MC solution stable under single precision arithmetic:

1. If a 32-bit Mersenne twister is used as the random number generator, the random bits should be typecasted from unsigned integer to float by dividing the unsigned integer with *UINT_MAX*. During this process, the random number can become unity, and thus the random number should be adjusted so that it does not exceed 1 − *FLT_EPSLION*.
2. The direction vector of neutron is vulnerable to numerical errors that accumulate during the coordinate transformation, and the size of the direction vector deviates from unity. Hence, renormalization of the direction vector is needed after every coordinate transformation.
3. The linear − linear interpolation of CDF in Eq. (4.15) can be erroneous when the two adjacent PDF values are very close. If the difference of the two PDF values are below a certain threshold (in PRAGMA, 0.0001), the interpolation scheme should fall back to the histogram interpolation.
4. Functions such as sqrt() and log() are recommended to be overridden by ‘safe’ versions which can treat zero or negative inputs appropriately.

Additionally, the ray tracing calculation must be performed in double precision if general geometry is considered. Single precision ray tracing is vulnerable to the self-intersection problem and particles get stuck on surfaces. PRAGMA is employing a specialized geometry module which stores the line, ring, and cell indices of a neutron, which is free from the self-intersection problem even though single precision is used. However, it is not valid for generalized geometry treatments.

4.1.4 Initial Verification

Before proceeding to the discussion of GPU acceleration algorithms in PRAGMA, initial verifications on the soundness of the continuous-energy physics module and on the accuracy of the mixed precision approach were carried out. The verification of the continuous-energy physics module was performed with McCARD [15], and the accuracy of the mixed precision approach was examined by switching PRAGMA between the mixed precision mode and the double precision mode.

Verification of Continuous-energy Physics Module

Table 4.2 and Table 4.3 shows the eigenvalues of McCARD and PRAGMA for pin cells and assemblies in APR1400 [68], and Figure 4.6 illustrates the flux spectra of McCARD and PRAGMA for the 3.65% pin cell. All the calculations were performed with the cross section libraries based on ENDF/B-VII.1, but the libraries of the two codes are not necessarily identical. All the standard deviations are 2 pcm.

A positive bias is observed in the PRAGMA eigenvalues against the McCARD eigenvalues, but the degree of the bias is extremely small for all cases, and there is a perfect agreement between the spectra of the two codes. Considering that the cross section libraries are not identical, having such degree of errors is considered natural and we conclude that the continuous-energy physics module in PRAGMA was well-implemented.

Table 4.2 Comparison of pin cell multiplication factors with McCARD.

Pin	Without S(α, β)			With Hydrogen S(α, β)		
	McCARD	PRAGMA	Error (pcm)	McCARD	PRAGMA	Error (pcm)
1.72%	1.20020	1.20026	6	1.19932	1.19940	8
2.64%	1.31391	1.31396	5	1.31304	1.31307	3
3.65%	1.38181	1.38184	3	1.38091	1.38096	5

Table 4.3 Comparison of assembly multiplication factors with McCARD.

Assembly	Without S(α, β)			With Hydrogen S(α, β)		
	McCARD	PRAGMA	Error (pcm)	McCARD	PRAGMA	Error (pcm)
A0	0.99322	0.99329	7	0.99123	0.99128	5
B0	1.19767	1.19777	10	1.19548	1.19553	5
B1	1.01883	1.01888	5	1.02174	1.02180	6
B2	1.00685	1.00692	7	1.00988	1.00989	1
B3	0.97013	0.97020	7	0.97420	0.97422	2
C0	1.23107	1.23116	9	1.22874	1.22880	6
C1	1.07006	1.07009	3	1.07254	1.07262	8
C2	1.02370	1.02374	4	1.02731	1.02732	1
C3	1.01269	1.01272	3	1.01642	1.01646	4

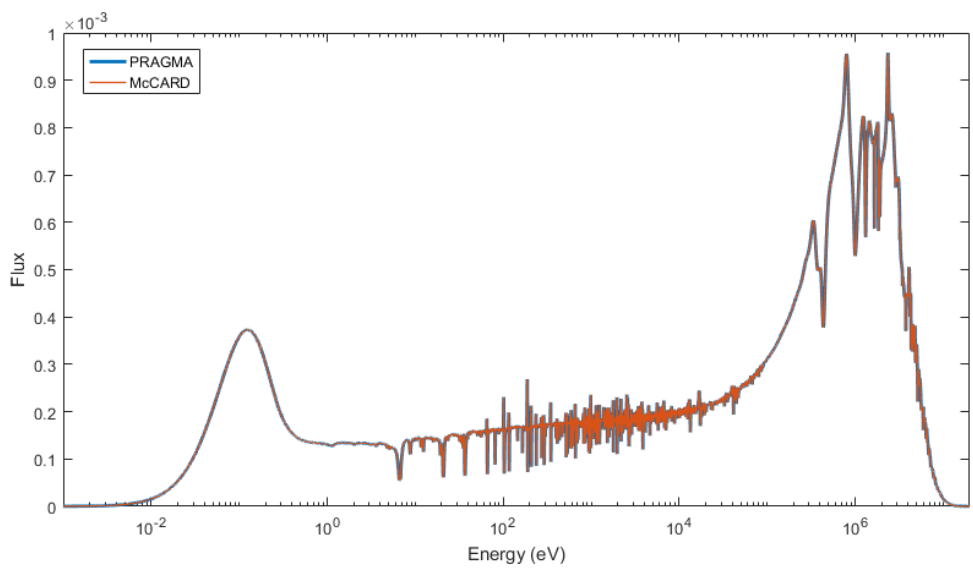


Figure 4.6 3.65% pin flux spectra of McCARD and PRAGMA.

Validity of Mixed Precision Approach

The APR1400 2D quarter core problem [68] was solved with mixed precision and double precision modes and the solutions were compared. Each case was executed for 20 times with 7.2 billion active histories per execution, and the 20-times averaged pin power distributions and eigenvalues were compared, as shown in Figure 4.7 and Table 4.4. Note that the pin power and the error distributions were octant-folded. No specific tendency could be found in the error distribution in relation to the power distribution; namely, the errors are random, which is likely due to the uncertainty. In addition, the RMS error of the pin power is much smaller than the RMS of pin power uncertainty. Namely, the pin power error is not distinguishable from the uncertainty and therefore negligible. Regarding the eigenvalue, although the error is larger than the uncertainty, its magnitude is practically negligible.

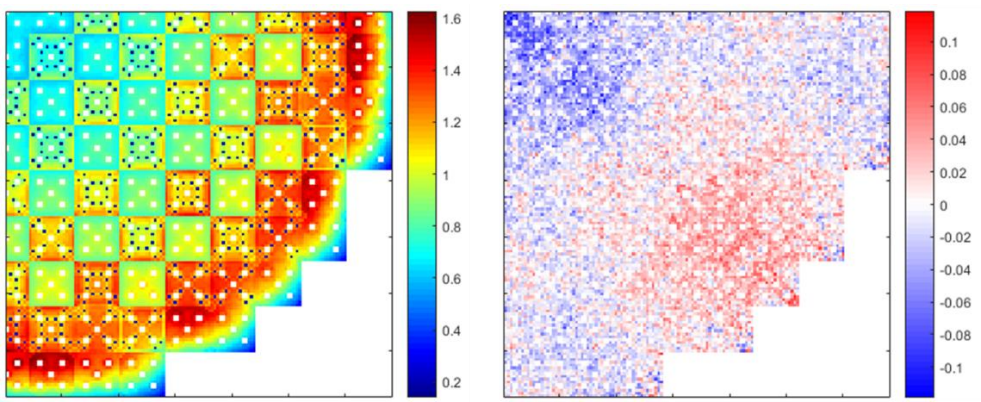


Figure 4.7 Average pin power distribution (left) and the relative difference (%) of the average pin powers (right).

Table 4.4 Summary of the errors between mixed and double precision results.

Eigenvalue Difference	2.4 pcm
Eigenvalue Uncertainty	0.7 pcm
RMS Pin Power Difference	0.028%
RMS Pin Power Uncertainty	0.086%

4.2 Optimization of Cross Section Lookup

Most of the computing time in the continuous-energy MC calculation is consumed for the calculation of macroscopic cross sections. Calculating the macroscopic cross sections requires summing up all the microscopic cross sections of the nuclides in the region, which involves significant amount of memory transactions which are all composed of random accesses. Thus, optimizing the cross section lookup process is crucial for achieving high performance on GPUs, and this section describes the cross section lookup optimization techniques used in PRAGMA. The performance of the optimization techniques presented in this section will be analyzed comprehensively in conjunction with the tracking algorithm in the subsequent section.

4.2.1 Hashed Unionized Grid Method

The unionized grid method is one of the energy grid lookup optimization schemes which was first introduced in the Serpent code [97]. Each nuclide possesses its own unique energy grid structure on which its cross sections are tabulated. Thus, when calculating the macroscopic cross section, the energy grids of every nuclide should be searched to retrieve the cross sections.

The unionized grid method treats this problem by defining an artificial energy grid which contains all the non-overlapping grid points of every nuclide. The unionized grid method also generates a large lookup table named double index, which stores pointers from the unionized grid points to the local grid points of each nuclide. Once the unionized grid point of an incident neutron energy is found, the local grid point of each nuclide can be found by simply reading the corresponding value of the double index table. Although a single search on the unionized grid point is more expensive than searching on each nuclide's grid as the unionized grid is much larger, it is only required once and the binary search algorithm has an $O(\log n)$ complexity. Therefore,

it is eventually much cheaper than performing searches for every nuclide. Figure 4.8 illustrates an example of the unionized grid and the double index table.

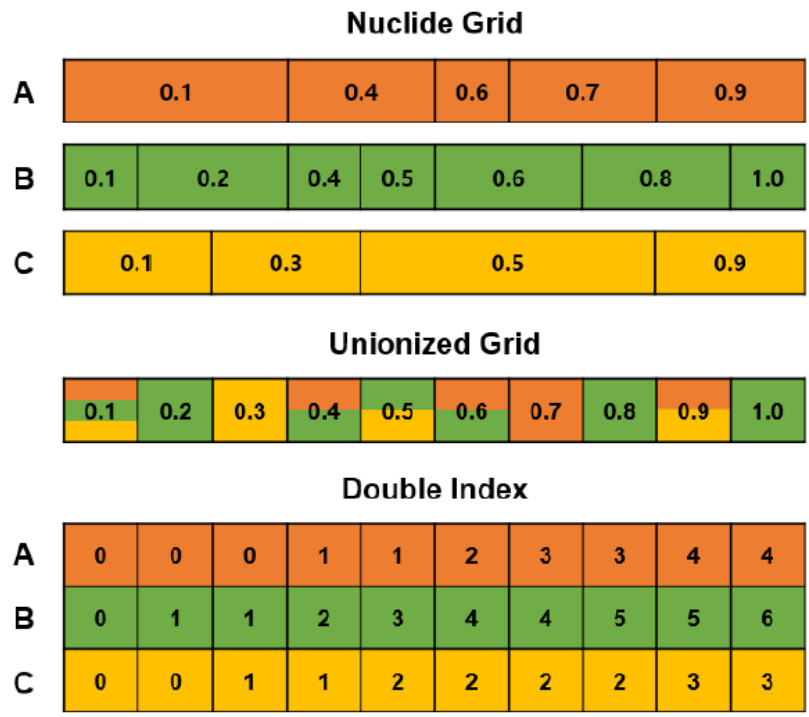


Figure 4.8 Example of the unionized grid and the double index table.

However, the size of double index table is determined as the number of unionized grid points multiplied by the number of nuclides, and this can grow excessively large in depleted fuel problems where hundreds of nuclides are contained and the number of the unionized grid points is of millions order. Accesses to such vast table are very sparse and strided, which does not promise a good performance.

Thus, a linear-interval hashing scheme is introduced, whose procedure is shown in Figure 4.9 and explained in the following descriptions. In this scheme, the double index is set only for every N unionized grid point. As the result, there is no more a direct guidance from a unionized grid point to the corresponding local grid point of each nuclide. Instead, one should refer to the double index of the nearest lower energy in the hashed double index table, which will only guide to a certain local grid

point near the target point. Finding the target point is then achieved by performing a linear search starting from the guided point.

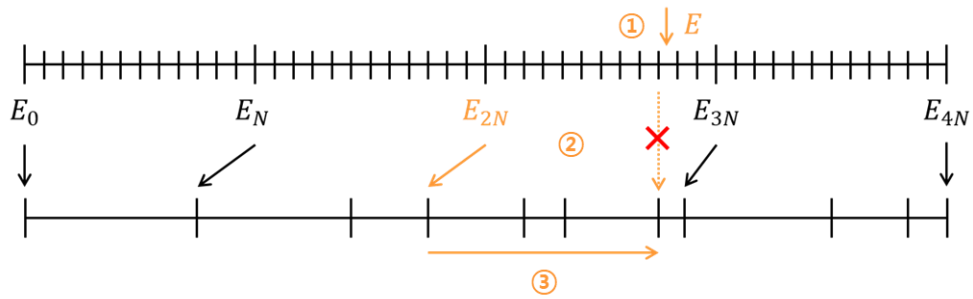


Figure 4.9 Schematic diagram of double index hashing scheme.

① A neutron comes in with energy *E* and the corresponding unionized grid point is found.

② Without hashing, the double index will directly guide to the local grid point. But with hashing, the double index will guide to a certain point near the target point.

③ Linear search is performed on the local grid and the target point is found.

The hashing scheme utilizes the fact that the unionized grid is much denser than the local grid of each nuclide. That is, even though the double index table is hashed, each hash will mostly contain only one or two local grid points. Figure 4.10 shows the distribution of the hash size, namely the number of local grid points contained, for a double index table containing 294 nuclides and 3,496,072 unionized grid points and hashed by 20. It can be seen that 94% out of the 51 million hashes contain only a single local grid point and 99.7% of the hashes contain no more than three local grid points. That is, the size of the double index table is reduced to 1/20 (3920MB to 196MB) while the linear search overhead is kept minimal. As the result, the size of the double index table becomes manageable with the limited GPU memory, and the performance is even improved by the reduction of the strided memory accesses to the double index table.

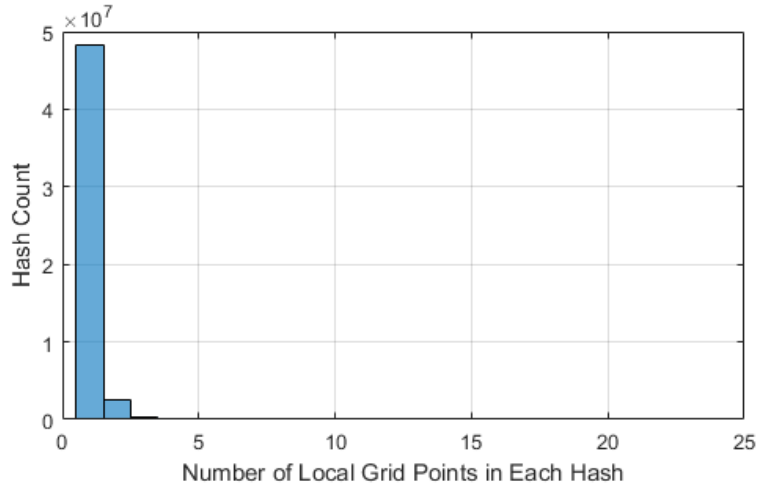


Figure 4.10 Histogram of the number of local grid points in each hash.

4.2.2 Temperature-Dependent Grid Collapse

Another problem of the unionized grid method is that the double index table has to be prepared for every temperature point. It is due to the fact that even for the same nuclide, the grids vary depending on the temperature, which were optimally chosen by NJOY [93]. Aside from the double index table, having different grids for a single nuclide serves as an obstacle of optimizing the schemes.

Therefore, the temperature dependent grids of each nuclide are unified into a single grid, as depicted in Figure 4.11. The cross sections at the newly added points during the grid unification are linear interpolated during the initialization phase.

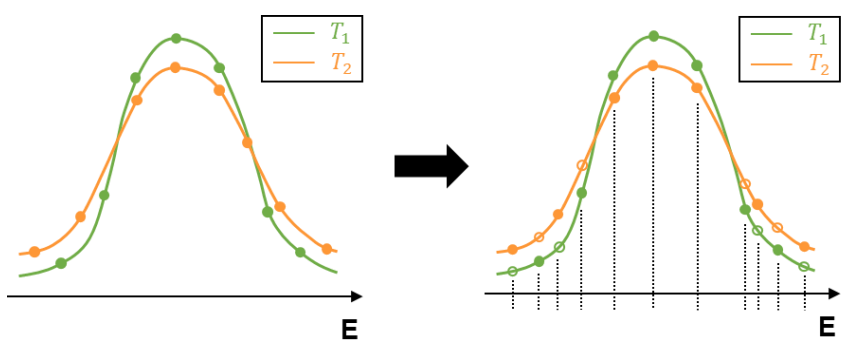


Figure 4.11 Illustration of temperature dependent grid unification.

This process increases the memory usage as it adds the cross section points by the pre-interpolation, but it is not burdensome as the majority of the grid points overlap. Rather, it obviates the need for storing the grids separately for each temperature point, which compensates the increased memory consumption.

Table 4.5 shows the number of grid points of U-238 before and after the unification process where 13 equidistant temperature points from 600K to 1800K are used. As can be seen, the number of grid points after unification is much smaller than the sum of each temperature set, which indicates that many of the points are superposed. As the result, the number of cross section points was increased by only 23%. Assuming that four types of principal cross sections (scattering, absorption, fission, and total) are additionally stored for each point, the number of additional variables is around 1.4 million. However, as the grid goes unified, storing only a single grid is enough and the number of grid points that must to be stored is reduced from 1.5 million to 0.14 million, which compensates the increase in the cross section points.

Table 4.5 Comparison of the number of grid points before and after the unification process for U-238.

Temperature (K)	Number of Grid Points		
	Before Unification	After Unification	Ratio
600	133,953	144,003	1.075
700	129,496		1.112
800	125,674		1.146
900	122,570		1.175
1000	119,680		1.203
1100	117,369		1.227
1200	115,350		1.248
1300	113,424		1.270
1400	111,585		1.291
1500	110,052		1.309
1600	108,676		1.325
1700	107,377		1.341
1800	106,198		1.356
Total	1,521,394	1,872,039	1.230

4.2.3 Stochastic Mixing of Temperature Dependent Cross Sections

Cross section temperature interpolation is also an important issue. Deterministic interpolation of the cross sections requires the cross sections of both temperatures to be known, which will double the amount of memory accesses. Therefore, a stochastic mixing approach is introduced for the temperature interpolation of the cross sections. Namely, if a temperature lies between the two given temperature points T_i and T_{i+1}, either temperature point is selected stochastically.

Then, the issue would be the choice of the interpolation factor. The convolutional nature of the Doppler broadening kernel does not give an easy way of interpolation. Therefore, it is typically assumed that the cross sections change either linearly (T) or proportionally to the square root $(\sqrt{T})$. If sufficiently fine temperature points are used, the linear interpolation will yield accurate results. However, it is prevented by the memory limitation of GPUs. Thus, PRAGMA uses the following interpolation formula which was empirically found to be optimal:

$$f = \frac{T^{2/3} - T_i^{2/3}}{T_{i+1}^{2/3} - T_i^{2/3}} \tag{4.46}$$

Table 4.6 shows the accuracy of different interpolation schemes for a pin problem in terms of the eigenvalues. The reference eigenvalues were obtained using the cross sections generated exactly at the target temperatures, and the standard deviations are all within 2 pcm. It can be seen that the suggested $T^{2/3}$ interpolation gives the most decent results among the interpolation schemes. The linear interpolation should use sufficiently fine temperature interval (less than 100K) to be accurate. Interpolation with $\sqrt{T}$ performs better, but still has some errors at 300K interval and a negative bias is observed. On the other hand, using $T^{2/3}$ not only shows the lowest error level, but also retains accuracy even at 300K interval.

Table 4.6 Accuracy comparison of various temperature interpolation schemes.

Temp (K)	Reference k_{eff}	Interpolation Point (K)		k_{eff} of Different Interpolation Schemes (Error)		
		Lower	Upper	T	$\sqrt{T}$	$T^{2/3}$
350	1.39863	300	400	1.39869 (6)	1.39860 (-3)	1.39862 (-1)
		300	500	1.39880 (17)	1.39858 (-5)	1.39866 (3)
		300	600	1.39891 (28)	1.39853 (-10)	1.39868 (5)
450	1.39179	300	500	1.39189 (10)	1.39173 (-6)	1.39178 (-1)
		300	600	1.39220 (41)	1.39164 (-15)	1.39185 (6)
		400	500	1.39185 (6)	1.39178 (-1)	1.39181 (2)
		400	600	1.39191 (12)	1.39175 (-4)	1.39178 (-1)
550	1.38510	300	600	1.38533 (23)	1.38500 (-10)	1.38510 (0)
		400	600	1.38521 (11)	1.38507 (-3)	1.38509 (-1)
		500	600	1.38515 (5)	1.38507 (-3)	1.38511 (1)

4.2.4 Optimization of Cross Section Memory Layout

The random nature of the MC calculation makes it difficult to achieve the memory coalescing between threads. If that is the case, it is necessary to increase the memory throughput of each thread to compensate the uncoalesced memory accesses. For such purpose, CUDA provides special built-in data types called vector types. They have the form of a typical C struct that contains two or four scalar variables as its members *x*, *y*, *z*, and *w*; for example, a *float4* vector type variable carries four *float* variables.

The vector types use a special memory alignment and the NVCC compiler uses a wider memory load instruction. When reading a single 4-byte scalar variable such as *int* or *float*, the compiler uses an ordinary *LD* instruction which is 32-bit wide. On the other hand, when reading a 16-byte vector variable containing four *int* or *float* variables, the load instruction is substituted by *LD.128* which is 128-bit wide. Recall that the size of a cache line sector is 32 bytes. To fill a cache line sector with *LD*, accesses to eight scalar variables should be coalesced. However, filling a cache line sector with *LD.128* only require two accesses to be coalesced, which would greatly increase the memory throughput under random access conditions.

Therefore, PRAGMA utilizes the vector types for the data that are always read in pair or that are heavily loaded. The most representative data are the principal cross sections and the number densities that are extensively accessed in the macroscopic cross section calculation. In case of the principal cross sections, there are four major reactions − scattering, absorption, fission, and total − taken into account in the MC simulation which perfectly cast into the *float4* vector type as illustrated in Figure 4.12; namely, the cross sections at each energy point are carried as a single vector variable.

However, expressing the number densities by the vector types requires some effort. The number density list in each region is split by four and each chunk is saved as a *float4* vector variable as illustrated in Figure 4.13, but padding is required because

the number of nuclides contained in each region is not necessarily divided by four. When calculating the macroscopic cross section, the sweeping loop is unrolled by the batch size of four and every time four packed number densities are pre-fetched to the local memory.

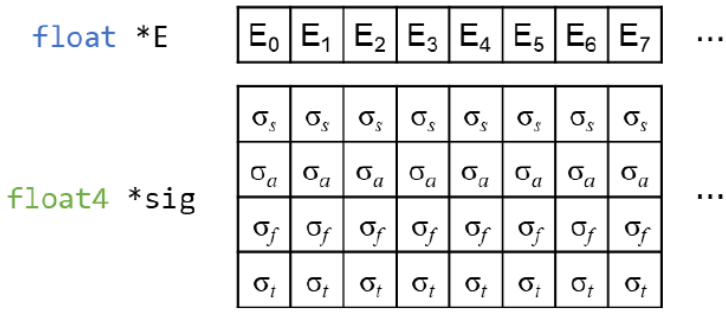


Figure 4.12 Layout of principal cross section data.

U234	2.926465E-06	pnum[0]
U235	3.420932E-04	
U236	5.583304E-06	
U237	4.486001E-08	
U238	2.259984E-02	pnum[1]
U239	1.675936E-08	
U240	5.104557E-12	
Np234	2.453305E-24	
Np235	6.276698E-17	pnum[2]
Np236	2.938223E-14	
Np237	9.724436E-08	
Np238	3.591513E-10	
Np239	2.426722E-06	pnum[3]
Pu236	1.615628E-19	
Pu237	1.678892E-14	
Pu238	1.362319E-09	
Pu239	1.915216E-05	pnum[4]
Pu240	6.129561E-07	
Pu241	5.219927E-08	
Pu242	7.259716E-10	
Pu243	2.281731E-13	pnum[5]
Pu244	7.481629E-16	
	Dummy	
	Dummy	

Figure 4.13 Layout of number density data.

4.3 Vectorization of Neutron Tracking

The random nature of the MC method is unfavorable to the characteristics of GPUs which process work items of the same operation in groups. To perform the random walk process of neutrons on GPUs, stream compaction and sorting algorithms should be exploited to regularize neutrons, and a new tracking algorithm which can properly cast the regularized neutrons into the vector processing pipelines of GPUs has to be developed.

Hence, an event-based tracking algorithm and a modified history-based tracking algorithm were developed. Sorting algorithms such as region partitioning and energy sort and the reference remapping scheme which makes the sorting operation easier and more efficient were also introduced. In addition, an array-based bank algorithm which replaces the conventional queue-based algorithm for massive parallelization was devised. All these works will be explained in detail throughout this section, and an extensive performance analysis will be performed at the end of this section.

4.3.1 Tracking Algorithm

History-based Tracking Algorithm

History-based tracking algorithm is the most conventional and widely used form of the neutron tracking algorithm, in which the neutron history from its birth to death becomes the unit of parallelization. Namely, each neutron is traced for its entire life, one at a time.

In the GPU implementation of the history-based algorithm, one neutron is assigned to a thread in contrast to the ordinary CPU implementations which assign a batch of neutrons to a thread, as illustrated in Algorithm 4.1. It is the most straightforward algorithm and easy to program, but it has been considered inappropriate for GPUs due to the excessive branches. Every neutron undergoes its own sequence of ‘events’

which is different from others. An event is a category of neutron actions, and the choice of event is randomly determined. Each event is then processed by conditional branches. As the result, the vectorization of threads and the memory coalescing are barely expectable.

However, in terms of the local memory use, the history-based algorithm possesses strengths. Since the tracking loop exists as a large monolithic kernel which is active for an entire cycle, the local variables also stay in the scope during the cycle. Thus, frequently used data can be pre-fetched to the local memory in the beginning of the cycle and can be reused for the remaining cycle.

Algorithm 4.1 History-based tracking algorithm.

```
Launch kernel with num_neutron threads
    parallel foreach num_neutron neutrons
     while (alive)
       Calculate macroscopic cross section
       Calculate DTS and DTC
       Move neutron to next position
       if (DTS < DTC) continue
       else
         Sample target nuclide
         Store fission source sites
         Sample collision reaction
         Sample outgoing energy-angle pair
         Update weight and perform Russian roulette
       end if
     end while
    end parallel foreach
Generate fission neutrons from source sites
```

Event-based Tracking Algorithm

To overcome the drawbacks of the conventional history-based tracking algorithm in terms of vector processing, the event-based tracking algorithm was first suggested by Brown and Martin in 1980s [31] when the vector processing became popular in the contemporary supercomputers. As the GPUs are increasingly gaining attention nowadays, the event-based algorithm has also started to be revisited.

The event-based algorithm can expose more refined parallelism. In contrast to the history-based algorithm where the basic unit of work is a neutron history from birth to death, the event-based algorithm treats a particular event in a neutron history as the unit of work. That is, the events are processed one at a time, which allows to trace a collection of neutrons that are about to undergo the same event. This is completely opposite from the history-based algorithm which processes a sequence of events for each neutron history.

In the GPU implementation, each event constitutes a kernel and the tracking loop consists of thousands of kernel calls. For each event kernel, a sorting scheme can be paired to collect the neutrons that are undergoing the event. The implementation of the event-based algorithm may vary depending on how the events are defined and how the sorting operations are performed. Maximizing the granularity of events will result in branchless event kernels that will enjoy a high vectorization efficiency, but as it will likely come with additional expenses of sorting costs, it is important to find a balance between reducing the branches and increasing the sorting overheads.

PRAGMA's event-based algorithm employs three kernels, as shown in Algorithm 4.2: macroscopic cross section calculation, tracing, and collision. Macroscopic cross section calculation kernel delivers the macroscopic cross sections needed for DTC calculations and tallies. This kernel also pre-samples target nuclides assuming that every neutron will make collisions, because sampling the target nuclide requires the CDF of each nuclide's contribution to the macroscopic total cross section. It takes the highest computational cost, and thus special sorting algorithms will be introduced. Tracing kernel finds intersections with the geometry to calculate DTS and migrates each neutron. Track-length and macroscopic collision tallies are also performed here. Collision kernel is where the reactions are processed; fission source sites are stored, reaction types are determined, and energy and angle pair of each neutron is newly determined. Before entering the collision kernel, a binary sort on the neutrons with the collision flag is performed.

After the sequence of the kernels, the dead neutrons are sorted out and the sequence is repeated until there is no more alive neutron left; the number of active threads is gradually reduced during a cycle.

Algorithm 4.2 Event-based tracking algorithm

```
while (num_alive > 0)
  Launch macroscopic cross section reconstruction
  kernel with num_alive threads
  |   parallel foreach num_alive neutrons
  |     Calculate macroscopic cross section
  |     Sample target nuclide assuming collision
  |   end parallel foreach
  Launch tracing kernel with num_alive threads
  |   parallel foreach num_alive neutrons
  |     Calculate DTS and DTC
  |     Move neutron to next position
  |     if (DTS < DTC) collision = false
  |   end parallel foreach
  Sort colliding neutrons
  Update num_collision
  Launch collision kernel with num_collision threads
  |   parallel foreach num_collision neutrons
  |     Store fission source sites
  |     Sample collision reaction
  |     Sample outgoing energy-angle pair
  |     Update weight and perform Russian roulette
  |   end parallel foreach
  Sort alive and dead neutrons
  Update num_alive
  Other posterior sorting processes
end while
Generate fission neutrons from source sites
```

In fact, the collision kernel contains a lot of branches due to various reaction modes and energy-angle sampling laws. However, the majority of the reactions are thermal and elastic scattering, as shown in Table 4.7. In addition, the computing time portion of the collision kernel turned out to be small, taking less than 10% of the total time. Therefore, sorting neutrons with refined reaction types will not be beneficial as the sorting cost will outweigh the benefit, and we consider that binary sort is sufficient and the branches in the collision kernel is tolerable.

Table 4.7 Portion of reactions in a typical LWR pin cell (10 million histories).

Reaction	Count	Portion
Hydrogen S(α, β)	1.34E+08	37.12%
Elastic Scattering	2.23E+08	61.72%
Discrete Level Scattering	2.84E+06	0.79%
Continuum Level Scattering	1.28E+06	0.36%
(n, $2n$) and (n, $3n$)	3.81E+04	0.01%

Modified History-based Tracking Algorithm

To take advantage of the sorting schemes while keeping the strength of the history-based algorithm in the local memory utilization and the easiness of programming, a modified history-based tracking algorithm outlined in Algorithm 4.3 is proposed. In this algorithm, the tracking loop is still constituted as a single kernel as the ordinary history-based algorithm, but the number of transitions − either surface crossing or collision − that can occur at each kernel call is limited. That is, the entire tracking loop is constructed by repeated calls of the flight kernel which drives each neutron's migration up to a preset *max_transition* times. Between the kernel calls, the sorting schemes can be introduced to increase the chance of vectorization as is done in the event-based algorithms. As the result, the modified history-based tracking algorithm is essentially a hybrid form of the history-based and the event-based algorithms.

In fact, setting the maximum transition to infinity makes the algorithm identical to the ordinary history-based algorithm, which will be referred to as the history-based limit. On the other hand, setting the maximum transition to unity results in a similar algorithm with the event-based algorithm, which then becomes the event-based limit. That is, the modified history-based algorithm can be tuned to have both history-based and event-based characteristics by simply adjusting the maximum transition.

The idea of limiting the number of events that can occur at each kernel was first proposed by Hamilton et al. [38] for accelerating a multi-group MC simulation with GPUs, which restricted the number of collisions. However, limiting the number of

transitions turned out to be more effective than limiting the number of collisions. In most cases, the surface crossing event is much more frequent than the collision event, so only limiting the number of collisions will result in a very different number of transitions between the threads, which will cause many idle threads.

Algorithm 4.3 Modified history-based tracking algorithm.

```
while (num_alive > 0)
  Launch kernel with num_alive threads
    parallel foreach num_alive neutrons
      for i = 1 : max_transition
        if (!alive) break
        Calculate macroscopic cross section
        Calculate DTS and DTC
        Move neutron to next position
        if (DTS < DTC) continue
        else
          Sample target nuclide
          Store fission source sites
          Sample collision reaction
          Sample outgoing energy-angle pair
          Update weight and perform Russian roulette
        end if
      end for
    end parallel foreach
  Sort alive and dead neutrons
  Update num_alive
  Other posterior sorting processes
end while
Generate fission neutrons from source sites
```

4.3.2 Array-Based Bank Algorithm

In the conventional CPU algorithms, neutron bank is implemented with the queue structure, which is a flexibly resizable first-in first-out (FIFO) data structure shown in Figure 4.14. The neutron to be traced is dequeued from the front of the queue, and the fission neutrons are enqueued to the back. By using a dual counter, the neutrons of current and next generations are distinguished. As the result, fission neutrons fill the queue completely after a cycle, and the queue is naturally reused in the next cycle. Since each thread can have its own queue, the data race is not a concern.

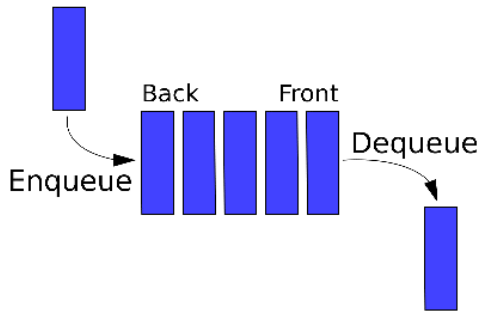


Figure 4.14 Queue structure.

On GPUs, however, such approach is not valid; there are millions of threads which make the privatization inadequate, and dynamic memory allocation on device is very expensive. Therefore, arrays should be used to implement the bank algorithm. As the result, the neutrons are stored as a large global array in the Array of Structure (AOS) fashion, and each thread has a unique index on the neutron array. A separate array to store the fission neutrons, which is called source sites, is also required. The source site array must be allocated with a redundant size compared to the neutron population to prevent overflow. As the result, twice of memory is consumed.

To reduce the memory waste and to enhance efficiency, the posterior fission source sampling technique is introduced. Instead of sampling the fission neutrons during the tracking, only essential information needed to sample the fission neutrons, such as the location, target nuclide, and incident energy and angle are stored in the source site array. Then, the fission neutrons are sampled collectively from the source sites. In this way, the size of the source site array can be minimized, and by separating the fission source sampling process, improvement in vectorization can be expected.

When storing the source sites in parallel, the atomic addition function of CUDA *atomicAdd* is utilized to prevent the data race. *atomicAdd* receives two arguments: pointer to the buffer on which the atomic addition is performed and the accumulating value. The return value of *atomicAdd* is the value of buffer right before performing the addition operation.

```
old_buffer = atomicAdd(&buffer, value);
```

At every fission event, each thread accumulates its neutron yield to a global offset buffer via *atomicAdd*. The return value is the value of the offset before adding the yield, which becomes the saving index of the thread on the source site array. It can be inferred that the value of the offset buffer after the completion of a cycle becomes the total number of neutrons produced. An example of fission source site filling by atomic addition is illustrated in Figure 4.15.

There was a concern for the performance of this scheme; atomic addition requires memory locks which will introduce synchronization of threads. However, the atomic functions were well-implemented by the GPU hardware and it eventually turned out that the overhead is negligible. It was just as effective as storing the yields to a vector and performing an exclusive scan on the yield vector to calculate the indices.

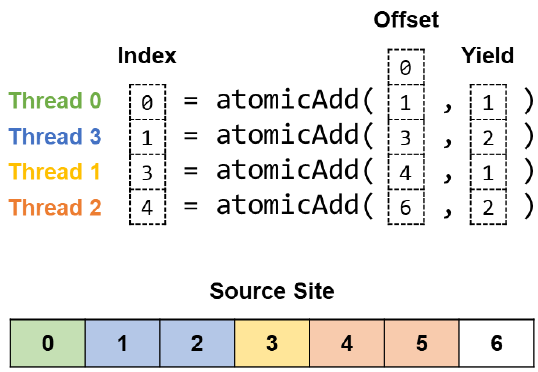


Figure 4.15 Example of parallel indexing of the source site by atomic addition.

4.3.3 Neutron Sorting Algorithm

Region Partitioning and Energy Sort

It is evident that the largest overhead in the continuous-energy MC calculation is caused by the macroscopic cross section calculation. It involves significant amount of random memory access, which makes the MC calculation to be bandwidth-bound.

Therefore, it is crucial to optimize the access patterns to the cross section data during the calculation of macroscopic cross sections.

The most straightforward sorting scheme that can be thought of is to sort neutrons with respect to their energies such that the chance of memory access coalescing to the cross section data is increased, as the cross section data are arranged by energy in ascending order. PRAGMA uses the unionized grid method and each neutron has the unionized grid index as integer. Therefore, the radix sort algorithm can be utilized to sort the neutrons by energy.

Another sorting algorithm to be introduced is region partitioning, which collects the neutrons that are about to pass the regions of same type. This idea originates from the fuel-sort specialization approach suggested by the NVIDIA researcher Scudiero [98], which tried to resolve an inefficiency in depleted fuel calculations. Figure 4.16 shows the typical number of nuclides contained in the fuel, cladding, and moderator of a depleted pin cell. As the fuel depletes, the number of nuclides explodes by the generation of fission products and actinides, and this results in a significantly larger number of nuclides contained in the fuel than in the other regions.

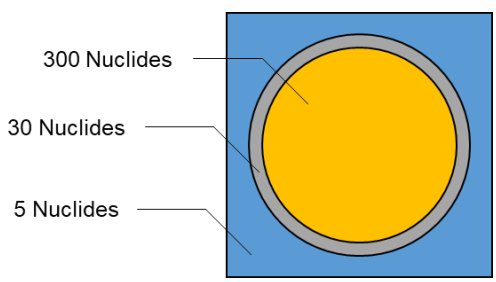


Figure 4.16 Typical number of nuclides in each region of a depleted pin cell.

It results in a severe branch divergence in the macroscopic cross section calculation kernel as the number of nuclides to be processed varies drastically between threads, as shown in the left figure of Figure 4.17. The idea of the fuel-sort specialization is to collect the neutrons that will pass the fuel region in the next event, such that the macroscopic cross section calculation kernel can expect a vectorized execution as

illustrated in the right figure of Figure 4.17, as some warps get fully composed of threads processing the fuel. The constituent nuclides of the depleted fuels are more-or-less the same, so by using the energy sort for the fuel threads, memory coalescing efficiency can be also improved.

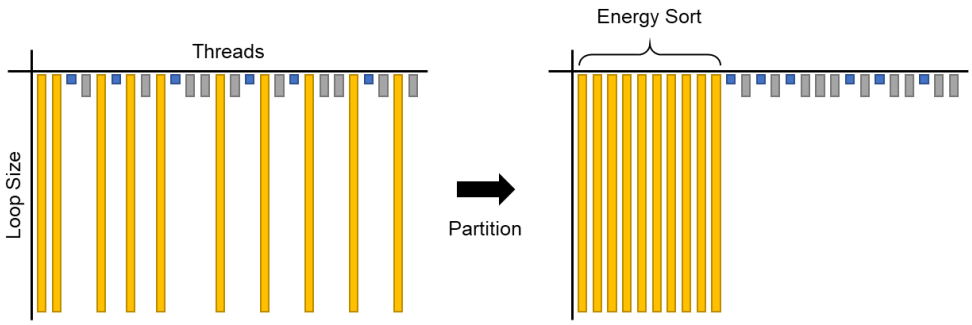


Figure 4.17 Schematic diagram of fuel partitioning and energy sort.

The fuel-sort specialization method only performed sorting block-wide. However, we perform the sorting device-wide, namely, for the entire active threads. In a more general sense, this approach can be referred to as region partitioning, as the sorting region does not necessarily have to be the fuel. For a fresh fuel problem, cladding becomes the region which contains the largest number of nuclides, and in this case neutrons in the cladding can be partitioned, which would be called clad partitioning.

Reference Remapping

When sorting, it is costly to rearrange the neutron data itself due to its large size. In addition, it is difficult to utilize highly efficient external libraries as the neutron data is a derived struct type which would not be supported by the libraries. Therefore, a remapping vector, which is an array of indices that maps each thread to a neutron, is introduced. The usage of the remapping vector is illustrated in Figure 4.18, where each thread refers to the remapping vector of its position and retrieves the neutron index. The elements colored in grey indicate terminated neutrons. This idea was first introduced in WARP [42] as reference remapping.

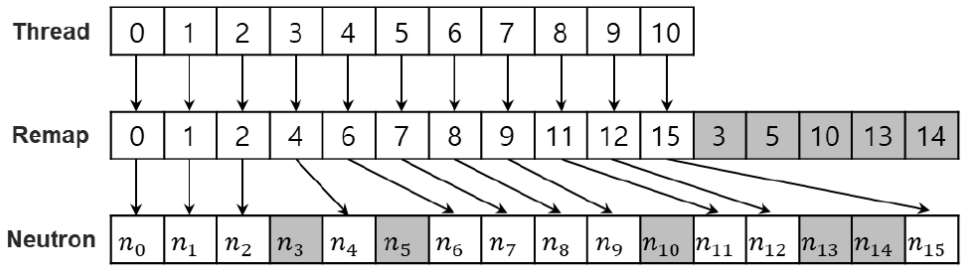


Figure 4.18 Example of reference remapping.

Instead of sorting the neutron data directly, the remapping vector is sorted. Figure 4.19 shows an example of binary sorting (flagged partition) with respect to the alive status of neutrons after passing the tracking kernel. Note that the neutron data remain unchanged and only the remapping vector is rearranged.

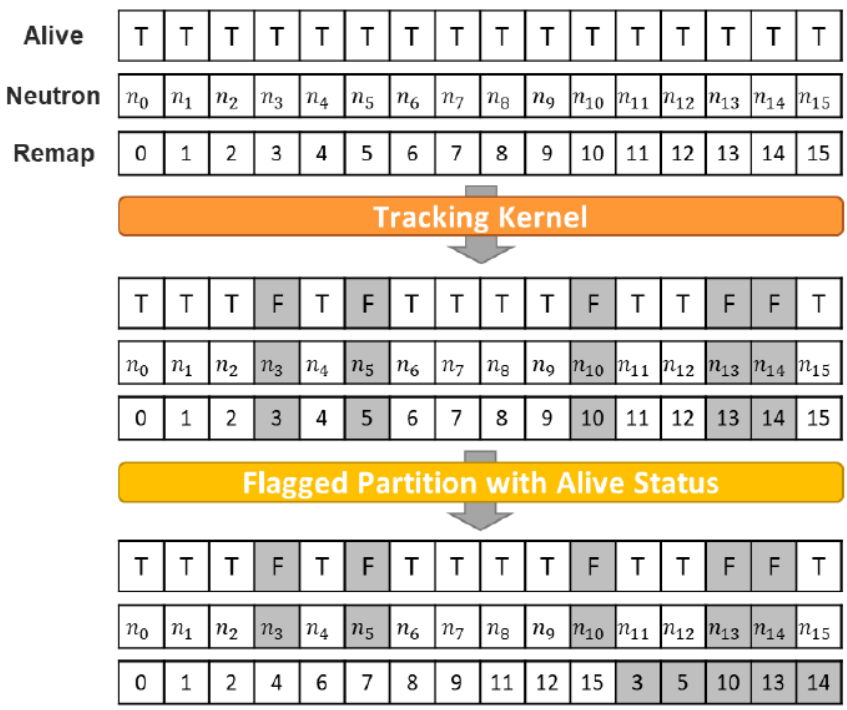


Figure 4.19 Example of the sorting using the remapping vector.

For the implementation of sorting algorithms, CUB library [99] is utilized, which provides parallel algorithms for handling arrays. The energy sort employs the radix sort algorithm of CUB, and all kinds of binary sorting operations such as sorting

alive neutrons and the region partitioning utilize the flagged partition algorithm of CUB. The data vector and the remapping vector are provided as the key and the value to the CUB function, respectively. Since CUB functions also alter the keys, a local copy of the data vector is passed to the sorting functions to prevent corruptions.

4.3.4 Comprehensive Performance Analysis

This subsection presents comprehensive parametric and performance analyses of the tracking algorithms. The target problems for the study are fresh and depleted 1.6% pin cells in the BEAVRS [91] benchmark. The composition of the depleted fuel was shared by Romano and Walsh [100]. A detailed description of the problem is shown in Figure 4.20 and Table 4.8. Unless specified, all the pin cell calculations employed three million neutrons per cycle and a single NVIDIA GeForce RTX 2080 Ti GPU.

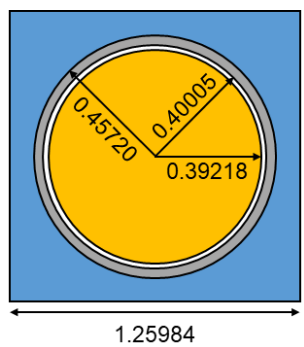


Figure 4.20 Geometry of the BEAVRS pin cell problem.

Table 4.8 Material composition of the BEAVRS pin cell problem.

Region	Number of Nuclides
Fresh Fuel	5 (U, O)
Depleted Fuel	290 (U, Np, Pu, …)
Gas Gap	2 (He)
Cladding	25 (Cr, Fe, Zr, Sn, O)
Moderator	6 (B, H, O)

Comparison of the Tracking Algorithms and the Effect of Sorting Schemes

First, the effect of sorting schemes was investigated with the event-based algorithm. The event-based algorithm can provide a detailed time share of the kernels, from which an insight in examining the performance can be obtained. For the fresh fuel problem, four cases were compared: 1. no extra sorting except for the basic sorting schemes (alive/dead sort and collision sort), 2. energy sort, 3. clad partition, and 4. clad partition with energy sort for cladding. Figure 4.21 illustrates per-cycle-average computing time of kernels for each case, and Table 4.9 compares the total average cycle time.

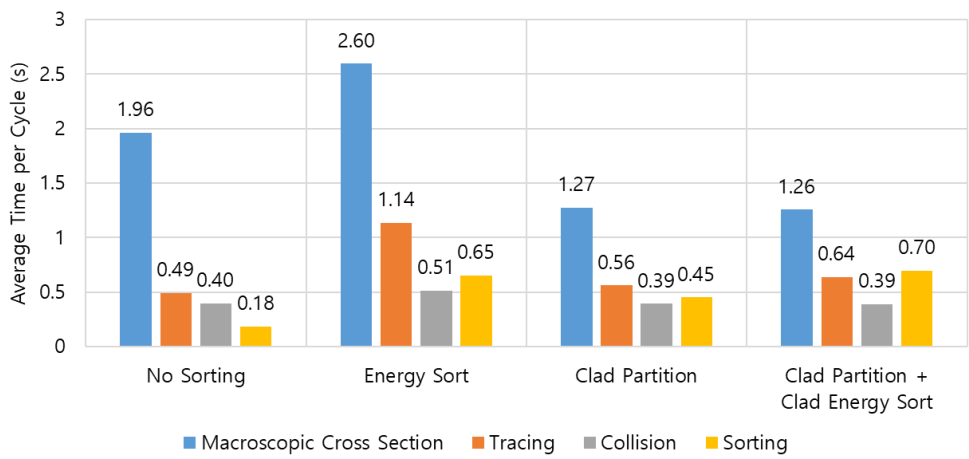


Figure 4.21 Time share of kernels of the event-based algorithm for the fresh fuel case with different sorting options.

Table 4.9 Average cycle time of the event-based algorithm for the fresh fuel case with different sorting options.

Option	No Sorting	Energy Sort	Clad Partition	Clad Partition + Clad Energy Sort
Time	3.03s	4.90s	2.68s	2.98s

For the fresh fuel case, introducing sorting schemes had little benefit. Only the clad partition case presented slightly improved performance compared to the base case, which would be strongly case-dependent and therefore not meaningful. The benefit

of sorting is not sufficiently large to compensate the sorting cost itself for the fresh fuel case. The sorting schemes are to reduce the computing time for the macroscopic cross section calculation, and they do have effect as far as the clad partition cases are considered, but basically the number of nuclides used in the fresh fuel case is small and the effect is limited.

A noteworthy result is that the energy sort had detrimental effect on every kernel. It is because of the chaotic memory access pattern to the neutron data after the radix sort. The energy sort completely mixes up the neutrons as the energy distribution of neutrons is fully randomized. As each kernel has to retrieve the neutron data from the global memory every time in the event-based algorithm, the access pattern to the neutron data also becomes important. Note that the energy sort after clad partition have the same drawback; the tracing kernel time is notably increased for the same reason compared to the base case. However, the impact is smaller in the clad partition cases, since the number of neutrons passing the cladding region at each iteration is only a fraction of the total population, as illustrated in Figure 4.22. That is, the sorting is performed on a limited range.

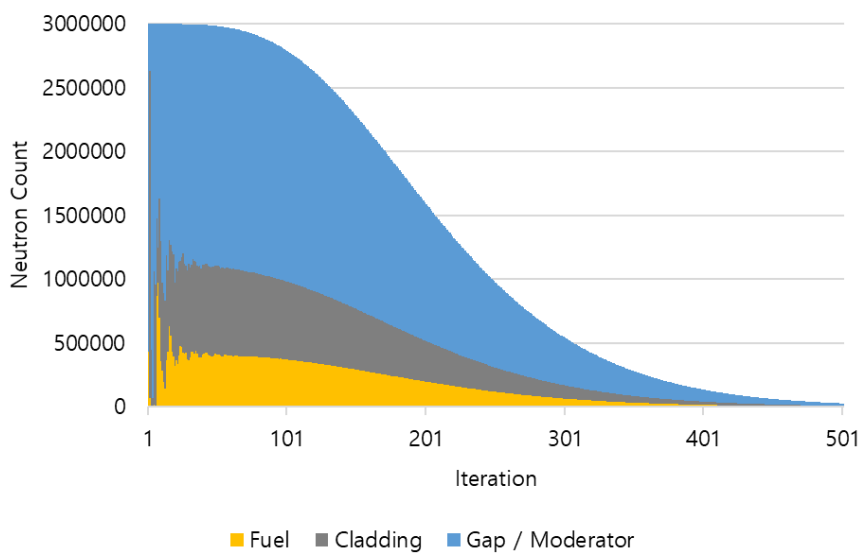


Figure 4.22 Distribution of the location of neutrons at each iteration of a cycle.

Same tendency is observed in the depleted fuel case illustrated in Figure 4.23 as well, but in this case the sorting schemes are extremely effective, as the reduction of the macroscopic cross section calculation time due to the enhanced access pattern to the cross section data outweighs any other detrimental effects. With the fuel partition paired by the energy sort, the macroscopic cross section calculation time is reduced to one-fourth from the base case, and the total runtime is reduced by 70% as shown in Table 4.10.

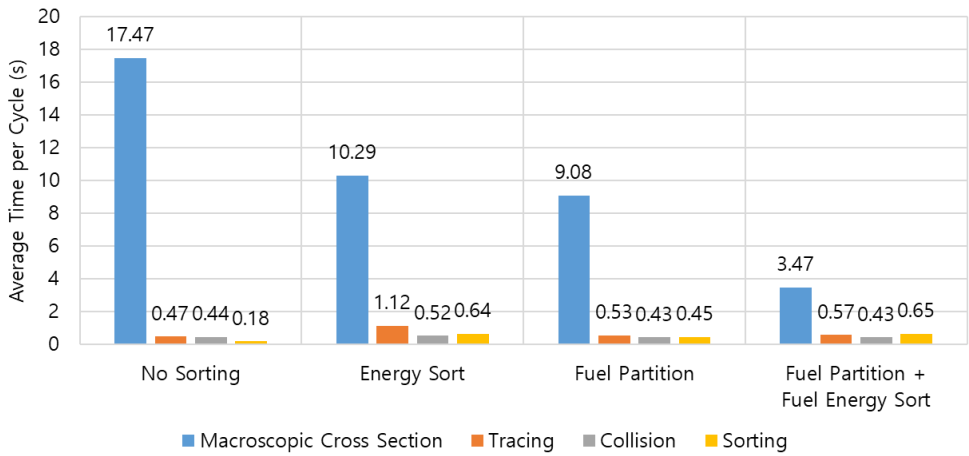


Figure 4.23 Time share of kernels of the event-based algorithm for the depleted fuel case with different sorting options.

Table 4.10 Average cycle time of the event-based algorithm for the depleted fuel case with different sorting options.

Option	No Sorting	Energy Sort	Fuel Partition	Fuel Partition + Fuel Energy Sort
Time	18.57s	12.57s	10.49s	5.11s

Next, the performance of the history-based algorithm was examined. Note that the term history-based algorithm means the modified history-based algorithm from now on. In the history-based algorithm, there is a hyper parameter, namely the transition limit, which should be examined. Figure 4.24 and Figure 4.25 shows the average cycle time for various transition limits and sorting schemes. The clad partition was

not tried for the fresh fuel case. It is quite evident that the clad partitioning will not be effective, as the clad region is very thin and most of the neutrons will penetrate. That is, the effect of sorting will only remain for a single transition after the sorting. Since the history-based algorithm simulates multiple transitions in each kernel, the effect of clad partitioning will be limited.

For a similar reason, the effectiveness of fuel partitioning for the depleted fuel case is maximized at the event-based limit and quickly diminishes as the transition limit is increased. The mean-free-path of a neutron in the fuel is typically larger than the dimension of the fuel pellet except for thermal neutrons, and therefore the effect of the fuel partitioning does not last long.

On the other hand, the energy sort was effective for the fresh fuel case in the history-based algorithm as opposed to the event-based algorithm. It is because, in the history-based algorithm, each thread makes a local copy of the neutron and uses it for several transitions until it is written back to the global memory. Thus, although the access pattern to the neutron data is spoiled by the energy sort, its impact is limited.

Another reason is that the surface intersection event is considerably more frequent than the collision event, as illustrated in Figure 4.26. That is, the effect of the energy sort can last for several transitions as the neutrons keep their energies in transit, and thus there exists an optimal frequency of energy sort. In contrast to the event-based algorithm, the history-based algorithm the sorting frequency can be adjusted flexibly by tuning the transition limit.

In any case, the results clearly demonstrate that setting the maximum transition is effective for the GPU implementation of the history-based algorithm. Limiting the number of transitions allows introducing the sorting schemes for vectorization while the modification of the original algorithm is minimized. Compared to the original algorithm (infinite transition limit), the modified history-based algorithm reduces the simulation time by 45% for the fresh fuel case and takes only one-fifth of the time for the depleted fuel case, at the optimal combination of algorithms and parameters.

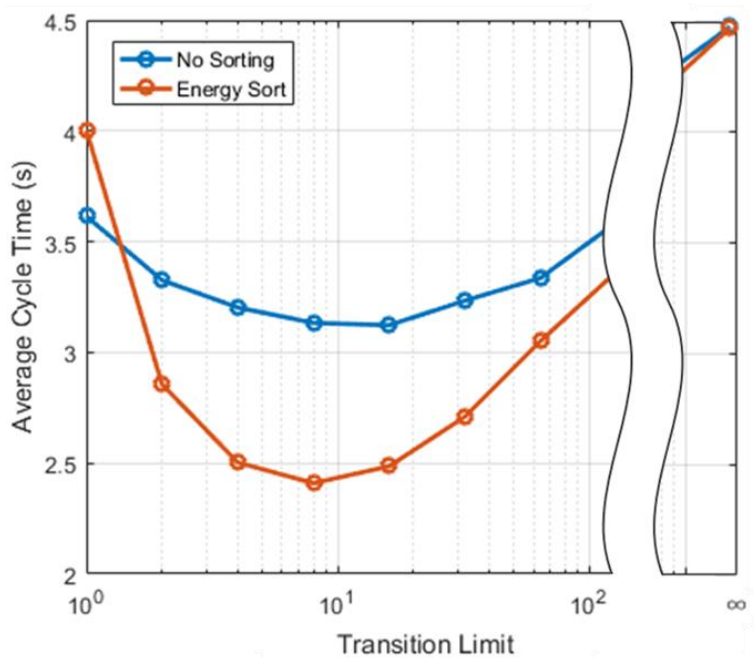


Figure 4.24 Performance of the history-based algorithm with respect to the transition limit and the use of sorting algorithms for the fresh fuel case.

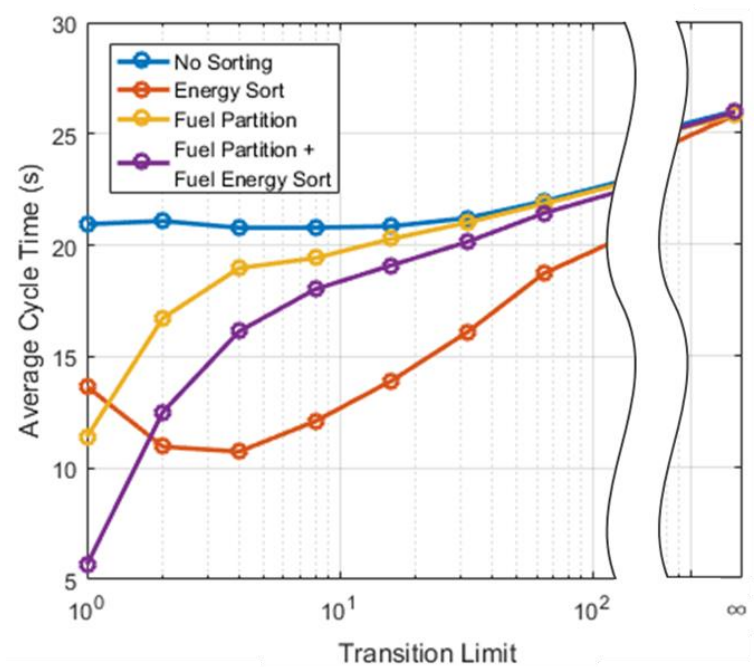


Figure 4.25 Performance of the history-based algorithm with respect to the transition limit and the use of sorting algorithms for the depleted fuel case.

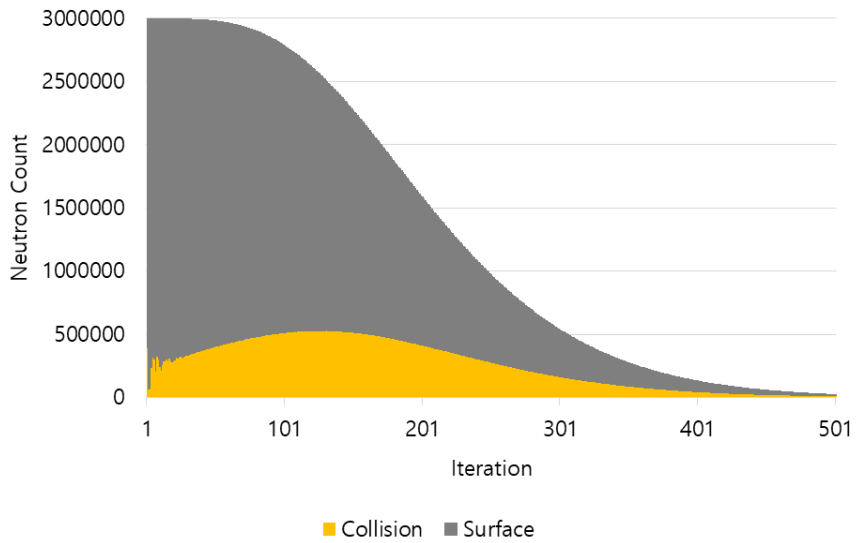


Figure 4.26 Distribution of the event of neutrons at each iteration of a cycle.

Effect of the Unionized Grid Method

Next, the effect of unionized grid method was examined. The number of nuclides used in the fresh fuel case is 34 and the number of unionized grid points is 628,695, and those of the depleted fuel case are 294 and 3,496,072. Figure 4.27 compares the relative macroscopic cross section calculation time of the energy look-up schemes. In the figure, the brute-force case is where the grid of each nuclide is searched one by one. For the hashed unionized grid method, the hash size was set to 50.

For the fresh fuel case, it was observed that the unionized grid method is even less efficient than the brute-force method. The fresh fuel case does not use much nuclides, so the strided access to the large double index table has more detrimental effects than searching a few number of grids. However, with the hashed unionized grid method, the double index table is compressed and the macroscopic cross section calculation time is reduced by 28% compared to the brute-force method and by 34% compared to the ordinary unionized grid method, respectively.

For the depleted fuel case, however, the unionized grid method is more efficient than the brute-force method due to the explosion of nuclides. Still, the most efficient

method is the hashed unionized grid method which reduces the macroscopic cross section calculation time by 64% compared to the brute-force method and by 35% compared to the ordinary unionized grid method, respectively.

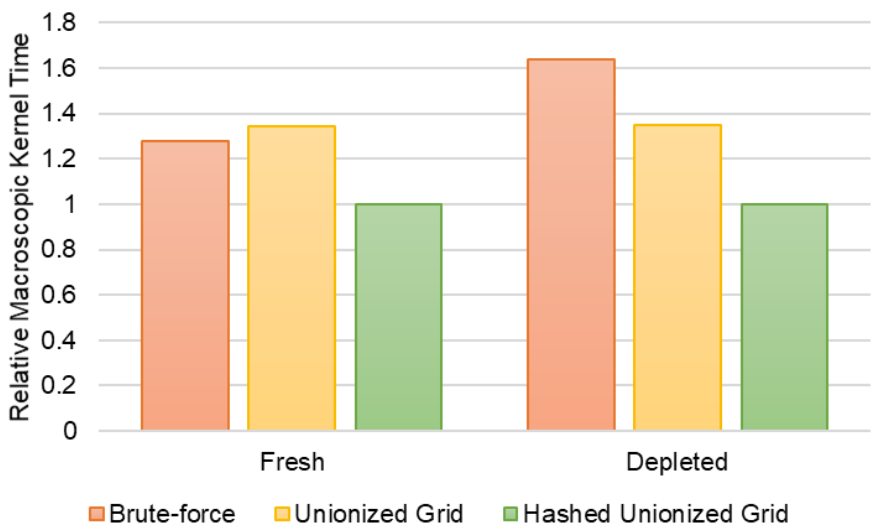


Figure 4.27 Performance comparison of the energy look-up schemes for fresh and depleted fuel conditions.

The sensitivity of hash size was also examined. The tracking rates of the history-based and the event-based algorithms depending on the hash size are shown in Figure 4.28 and Figure 4.29, respectively. The hash size was changed from 1 to 50. It turned out that the hashing scheme is more effective for the event-based algorithm than the history-based algorithm; the largest performance gain of the event-based algorithm was 17% for the fresh fuel case and 25% for the depleted fuel case, while those of the history-based algorithm were 4% and 17%, respectively.

The results support our initial claim that the performance will be enhanced by the hashing scheme due to the reduction of the strided accesses to the double index table. In other words, the hashing scheme not only saves the memory but also enhances the performance. Note that the previous results already imply the use of optimal hashing parameters.

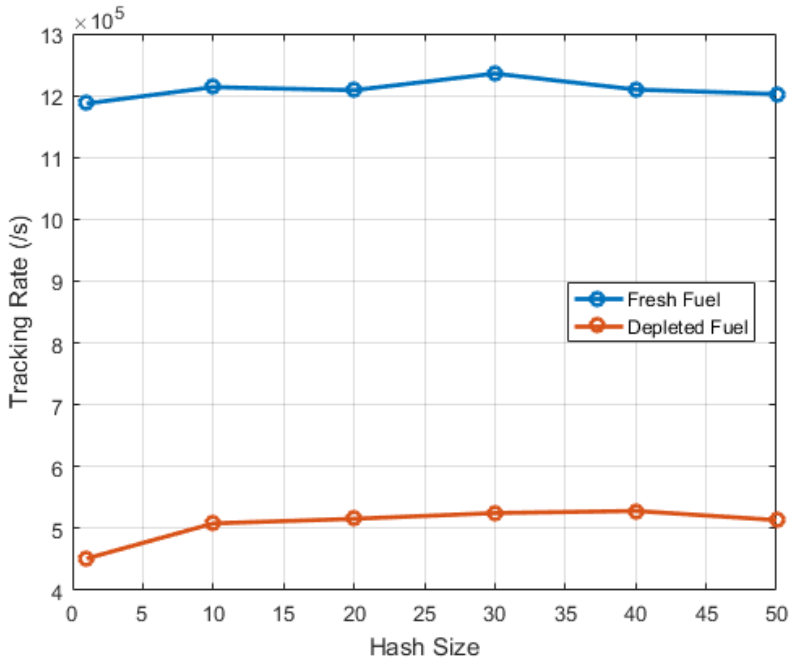


Figure 4.28 Tracking rate with respect to the hash size in the history-based algorithm.

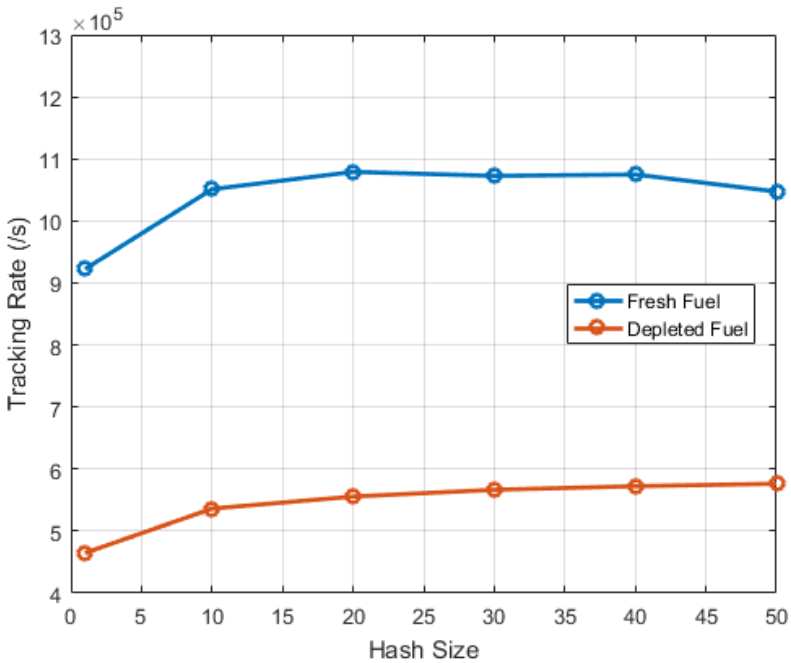


Figure 4.29 Tracking rate with respect to the hash size in the event-based algorithm.

Scalability of the Tracking Algorithms

In order to exploit the performance of massive parallel processors like GPUs, they should be exposed to a sufficient degree of parallelism. In case of the MC simulation, the number of neutrons assigned to each device becomes the metric of parallelization. Therefore, the scaling of the tracking algorithms against the number of histories per cycle was examined for the pin cell problems. The number of histories per cycle was changed from 0.1 million to 6.4 million.

Figure 4.30 and Figure 4.31 present the tracking rates as the function of the number of neutrons per cycle for the history-based and the event-based tracking algorithms, respectively. It is observed that more than three million neutrons should be deployed at each cycle to saturate the performance of the tracking algorithms. Especially for the event-based algorithm, the tracking rates still do not level off at 6.4 million per cycle.

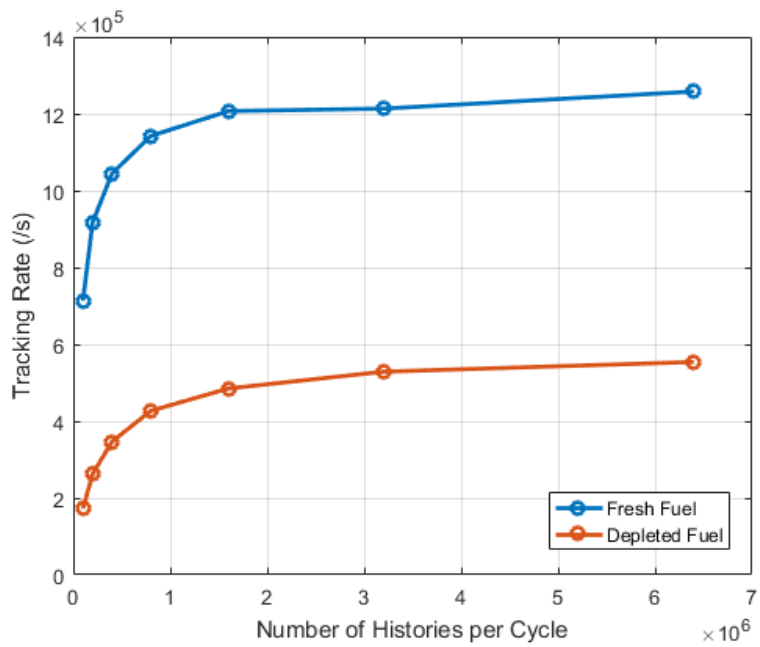


Figure 4.30 Tracking rate with respect to the number of histories per cycle in the history-based algorithm.

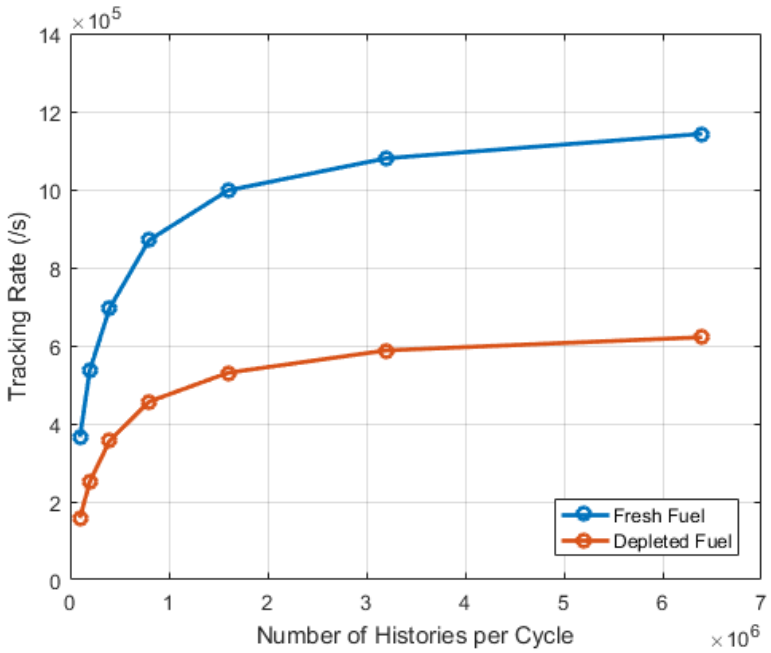


Figure 4.31 Tracking rate with respect to the number of histories per cycle in the event-based algorithm.

The cause of this problem is the population tail effect. From Figure 4.22 and Figure 4.26, it can be inferred that the majority of the neutrons die in less than a few hundred events, and the remaining cycle proceeds with a few number of neutrons. Figure 4.32 presents the within-cycle particle processing rates as the function of the fraction of terminated particles for the event-based algorithm. The depleted fuel problem was tested and the number of particles was varied from 1.6 million to 6.4 million.

In the beginning of cycle, namely below the 40% point, the processing rates are more-or-less the same regardless of the number of particles since there are enough particles alive for the GPU to extract parallelism. As the cycle proceeds, however, less particles become available and the processing rates begins to decrease. For the 6.4 million case, the performance is maintained until the 90% point since the number of particles alive is still kept large up to that point, whereas the performance drops after the 70% point due to the lack of particles for the 1.6 million case. In other words, the length of the tail where the tracking proceeds under the deficiency of parallelism becomes shorter with larger number of particles per cycle.

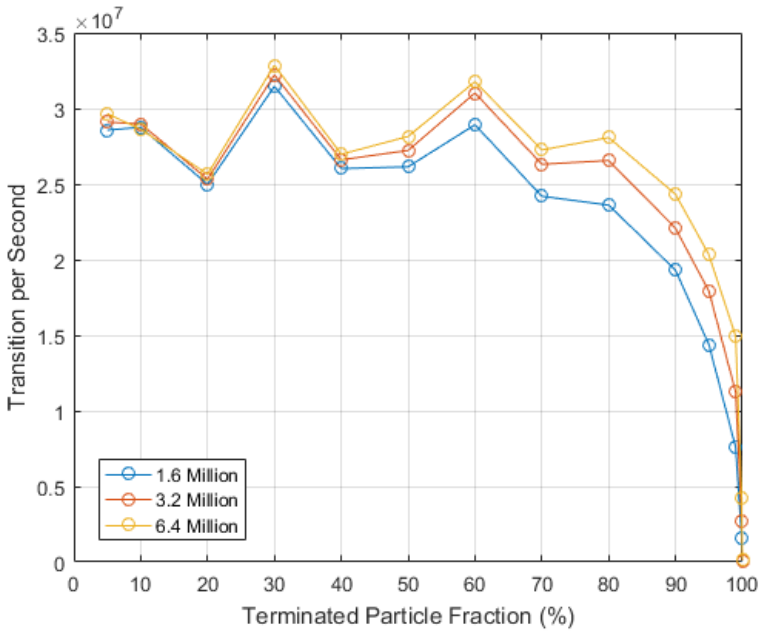


Figure 4.32 Within-cycle particle processing rates of the event-based algorithm depending on the number of particles for the depleted fuel case.

Performance Assessment with Production Codes

To objectively assess how fast the developed algorithms are, the performance of PRAGMA was compared with OpenMC for the same pin cell problems [100]. The paper provides the tracking rates of OpenMC for the pin cell problems under various conditions. The OpenMC calculations were performed with two Intel Xeon Platinum 8176 CPUs, which are a flagship model containing 38.5MB last-level cache and 28 cores each. PRAGMA calculations were performed with a single GeForce RTX 2080 Ti GPU and used optimal algorithms and parameters determined from the parametric studies; the fresh fuel case was solved by the modified history-based algorithm, and the depleted fuel case was solved by the event-based algorithm.

The comparison might not be considered fair as OpenMC employs a constructive solid geometry (CSG) representation for generality while the PRAGMA geometry module is dedicated for square lattices. That is, OpenMC may spend more time for the geometry treatments. However, the target problem is a two-dimensional pin cell

and the difference in the geometry treatment will have a marginal effect.

Table 4.11 shows the comparison result, which demonstrates the effectiveness of the developed algorithms. While substantial speedups are observed in all cases, the speedups for the depleted fuel cases which reach up to 500 are especially remarkable. Comparing the power consumption (165W per CPU, 250W per GPU) and the price ($8,719 per CPU, $999 per GPU) of the processors clearly demonstrates the cost-effectiveness of PRAGMA.

The significance of this result is that PRAGMA does not lose much performance in solving depleted fuel problems as opposed to the conventional MC codes, and this is due to the optimized macroscopic cross section calculation exploiting the vector processing capability of GPUs. Namely, the real strength of employing GPUs for the continuous-energy MC calculation is revealed in treating depleted fuels, and this will serve as a significant benefit in actual cycle depletion calculations.

Table 4.11 Comparison of the tracking rates (kiloneutrons per second) of OpenMC and PRAGMA.

Case	Temperature	OpenMC	PRAGMA	Speedup	Equivalent Cores
Fresh	300K	373.5	1206.7	3.23	181
	600K	362.3	1145.1	3.16	177
	1000K	354.2	1125.5	3.18	178
Depleted	300K	66.4	585.7	8.82	494
	600K	64.1	552.3	8.62	483
	1000K	63.1	553.6	8.77	491

Next comparison is made with the GPU version of the Shift code [45]. In the paper, the tracking rates of the Shift code achieved on the Tesla V100 GPU mounted on the Summit supercomputer [101] are presented. The results in the paper were obtained with so called RTK geometry option of the Shift code, which is a dedicated geometry module for power reactors and has an analogy to the specialized geometry module of PRAGMA. Therefore, the comparisons can be made under more fair conditions.

The Shift calculations were performed with the NuScale SMR benchmark [102], but the documents were not accessible. Therefore, we solved a similar-sized SMART 3D quarter core problem. According to the paper, the NuScale SMR benchmark fresh fuel problem models the full core, which creates approximately 142,000 mesh cells. The SMART 3D quarter core problem consists of around 160,000 mesh cells, so the problem sizes are equivalent.

Figure 4.33 compares the tracking rates of PRAGMA and Shift at inactive cycles. Both codes used the event-based algorithm, though the detailed implementations can differ. PRAGMA was run on several types of GPUs while only the result on the Tesla V100 GPU is available for Shift. On the same Tesla V100 GPU, PRAGMA achieves about four times higher tracking rate over Shift. However, the significance of this result is that PRAGMA is capable of achieving almost equivalent performance on a flagship consumer-grade GPU, namely GeForce RTX 2080 Ti, with the professional GPU. Even with the lower-grade GeForce GTX 1080 GPU, PRAGMA can achieve comparable tracking rate to that of Shift with the professional GPU. In other words, PRAGMA is well-optimized on consumer-grade GPUs, which enhances practicality.

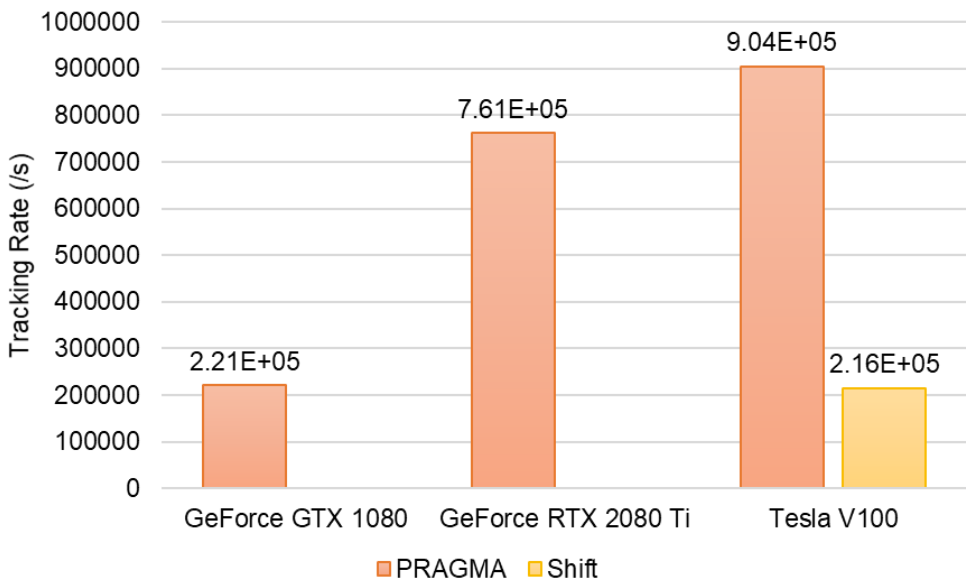


Figure 4.33 Comparison of the tracking rates of Shift and PRAGMA.

4.4 Target Velocity Sampling

Target Velocity Sampling (TVS) is an important procedure in the MC method to properly consider the elastic scattering of neutrons and target nuclei. The elastic scattering is described by the classical two-body kinematics, but determining the velocity of the target is not straightforward. Mostly a nucleus is assumed to be a free atom; namely, its velocity distribution follows the Maxwell-Boltzmann distribution. However, simply sampling the velocity from the Maxwell-Boltzmann distribution fails to preserve the reaction rates, as the cross sections used in the calculations had already been Doppler-broadened.

Using a proper TVS scheme is especially important for the treatment of epithermal resonance scattering. The resonance scattering tends to up-scatter the neutrons, and thereby lowering the reactivity. U-238, which is the dominant nuclide in the power reactors, is known to have very large epithermal resonance scattering cross sections as illustrated in Figure 4.34, and it has been observed that incorrectly treating these resonances may result in a substantial overestimation of reactivity as well as a wrong estimation of actinide inventories in depletion.

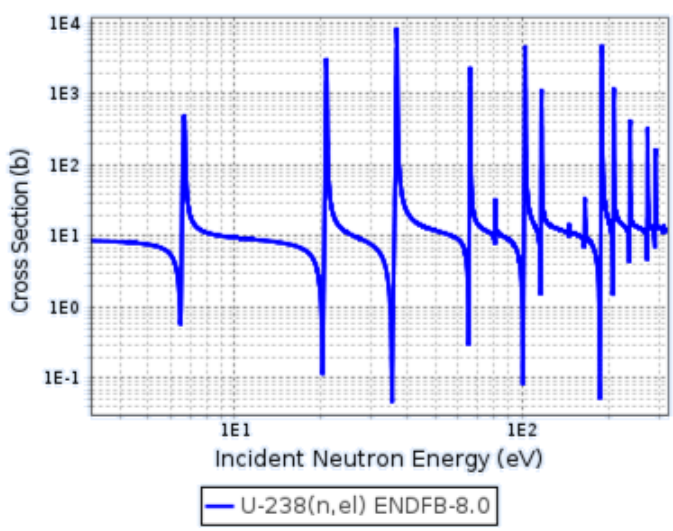


Figure 4.34 Elastic scattering cross sections of U-238.

The probability distribution of the target velocity $\mathbf{V}$ which is consistent to the Doppler-broadened cross sections is given as the Maxwell-Boltzmann distribution weighted by the reaction rates:

$$P(\mathbf{V},T)d\mathbf{V}=\frac{v_r\sigma_0(v_r)M(\mathbf{V},T)d\mathbf{V}}{\int v_r\sigma_0(v_r)M(\mathbf{V}',T)d\mathbf{V}'}=\frac{v_r\sigma_0(v_r)M(\mathbf{V},T)}{v\sigma_T(v)}d\mathbf{V} \qquad (4.47)$$

where

$M(\mathbf{V},T)$ = Maxwell-Boltzmann velocity distribution,

σ_0 = Zero-kelvin scattering cross section,

σ_T = Doppler-broadened scattering cross section at temperature T,

v = Neutron speed,

v_r = Relative speed of neutron and target.

Since the nuclei agitate isotopically, the Maxwell-Boltzmann velocity distribution can be written as follows:

$$M(\mathbf{V},T)d\mathbf{V}=M(V,T)f(\mu)g(\alpha)dVd\mu d\alpha=\frac{1}{4\pi}M(V,T)dVd\mu d\alpha\,. \quad (4.48)$$

Substituting Eq. (4.48) into (4.47) and integrating over the azimuthal angle yields:

$$P(V,\mu,T)dVd\mu=\frac{v_r\sigma_0(v_r)M(V,T)}{2v\sigma_T(v)}dVd\mu\,. \qquad (4.49)$$

Sampling of Eq. (4.49) is not trivial in that v_r is dependent to both V and μ, which makes the probability distribution bivariate, and that σ_0 also has distributions. Conventional TVS schemes to cope with this problem, which are either approximate or exact, are explained in the following. Then, a newly developed TVS scheme for the application in PRAGMA is presented.

4.4.1 Existing Methods

Asymptotic Scattering Assumption

In this assumption, it is considered that the target nucleus is virtually stationary, as neutrons are moving significantly faster than nuclei. This assumption is valid when a neutron has sufficiently high energy, while it is not valid in the thermal energies. Therefore, MCNP [14] applies the asymptotic scattering assumption for the neutrons having energies higher than $400k_BT$, where k_B is the Boltzmann constant. Below this threshold, the target velocities are explicitly sampled. Note that exclusively for hydrogen, the threshold is not set; namely, asymptotic scattering is not used for hydrogen. PRAGMA applies the same strategy with MCNP.

Constant Cross Section (CXS) Method

This method assumes that σ_0 in Eq. (4.49) is constant at the vicinity of incident neutron energy. As the result, it can be rewritten as follows:

$$P_{CXS}(V,\mu,T)=\frac{v_r\sigma_0(v_r)M(V,T)}{2v\sigma_T(v)}=Cv_r\beta^3V^2e^{-\beta^2V^2} \tag{4.50}$$

where $\beta=\sqrt{\frac{m}{2k_BT}}$ and m is the mass of the target nucleus. The following lemma [103] is then introduced to establish the sampling scheme:

Lemma. Any density function which has the form $f(x)=cg(x)h(x)$ where $g(x)$ is a density function and $h(x)$ is [0, 1]-valued can be sampled by drawing x' from $g(x)$ and accepting it with probability $h(x')$.

Using the fact that the relative speed cannot exceed the sum of two speeds, namely:

$$h(V) = \frac{v_r}{v+V} \leq 1 \tag{4.51}$$

Eq. (4.50) is reformulated so that the lemma can be utilized:

$$\begin{aligned} P_{CXS}(V, \mu, T)dVd\mu &= \frac{\sigma_0}{2v\sigma_T(v)} v_r M(V,T)dVd\mu \\ &= C\frac{v_r}{v+V}\left(v\beta^3 V^2 e^{-\beta^2 V^2} + \beta^3 V^3 e^{-\beta^2 V^2}\right)dVd\mu \\ &= Ch(V)g(V)dVd\mu. \end{aligned} \tag{4.52}$$

First, the target speed is sampled from $g(V)$, and then μ is sampled uniformly in $[-1,1]$. Then, the $V-\mu$ pair is accepted with the probability $\frac{v_r}{v+V}$, where

$$v_r^2 = V^2 - 2\mu vV + v^2. \tag{4.53}$$

CXS is valid for most of the nuclides; for light nuclei, the scattering cross sections change slowly with energy, and for heavy nuclei, their contributions to the neutron moderation are negligible so that the errors in the scattering kernels are tolerable. However, CXS becomes invalid for the nuclides that have large resonance scattering cross sections, especially U-238. Therefore, most MC codes employ CXS for the majority of the nuclides and apply elaborated TVS schemes for a few exceptional nuclides which require special treatments.

Doppler Broadening Rejection Correction (DBRC) Method

The DBRC method first conceptualized by Rothenstein [104] and popularized by Becker et al. [105] allows to retain the energy dependence of the cross sections in

explicitly by introducing a secondary rejection step to the CXS procedure. In this scheme, the maximum scattering cross section $\sigma_{\max}$ is set prior to the sampling. Instead of considering the entire energy range for finding the maximum, $\sigma_{\max}$ is set in the range of $\left[v - \frac{4}{\beta}, v + \frac{4}{\beta} \right]$; it is the four times of the most probable speed in the Maxwell-Boltzmann distribution and it is safe to assume that the relative speed will be contained in the range. Then, the probability distribution for DBRC is written as follows:

$$P_{DBRC}(V, \mu, T) dV d\mu = C' \frac{\sigma_0(v_r)}{\sigma_{\max}} \frac{v_r}{v+V} \left(v\beta^3 V^2 e^{-\beta^2 V^2} + \beta^3 V^3 e^{-\beta^2 V^2} \right) dV d\mu. \quad (4.54)$$

The relative speed is sampled by the ordinary CXS procedure, and then it is accepted with the probability $\frac{\sigma_0(v_r)}{\sigma_{\max}}$ which is guaranteed to be less than unity.

The major problem of DBRC is, however, that the secondary rejection step suffers from a very low acceptance rate at certain energies, especially when a neutron falls near a resonance dip. In that case, $\sigma_{\max}$ will likely be the resonance peak, which will result in a very low acceptance probability. At some energies near the resonances, it is reported that the average number of rejections per acceptance reaches tens of thousands.

Weight Correction Method (WCM)

This method was first introduced in the MVP code [106]. Like the DBRC method, the sampling of the target velocity is done by the ordinary CXS scheme. However, WCM avoids the rejection step by manipulating the weights of neutrons. Denoting $\sigma_0(v_r) = \sigma_0$ which is the assumption of CXS, $\sigma_T(v)$ can be Doppler-broadened

analytically as follows:

$$\sigma_T(v) = \sigma_0 \frac{\left(\beta^2 v^2 + \frac{1}{2}\right)\mathrm{erf}\beta v + \sqrt{\frac{1}{\pi}}\beta v e^{-\beta^2 v^2}}{\beta^2 v^2} = \sigma_0 g(v). \quad (4.55)$$

Plugging Eq. (4.55) into Eq. (4.52) yields:

$$P_{CXS}(V, \mu, T)dVd\mu = \frac{v_r}{2vg(v)} M(V, T)dVd\mu. \quad (4.56)$$

Resultantly, the ratio of the actual probability to the probability of CXS becomes the weight adjustment factor f, as follows:

$$f = \frac{P(V, \mu, T)}{P_{CXS}(V, \mu, T)} = \frac{\sigma_T(v)}{\sigma_0(v_r)} g(v). \quad (4.57)$$

Since WCM merely adds the weight correction to the CXS scheme, the calculation speed remains virtually unchanged from CXS. The critical drawback of WCM is, however, that the adjustment factor can become excessively large at certain energies, especially near the sharp resonances. If the neutron speed corresponds to the peak of a resonance and the relative speed is located at the resonance dip, the factor may become tens of thousands, which will likely ruin the calculation.

4.4.2 Developed Method: Relative Speed Tabulation (RST)

The developed scheme named Relative Speed Tabulation (RST) is motivated from the routine derivation of the Doppler broadening kernel. The expected reaction rate per incident neutron can be written as:

$$v\sigma_T(v) = \int_{\mathbf{V}} v_r \sigma_0(v_r) M(\mathbf{V}, T) d\mathbf{V}. \quad (4.58)$$

By the azimuthal symmetry, Eq. (4.58) becomes:

$$v\sigma_T(v) = \frac{1}{2}\int_0^\infty \int_{-1}^1 v_r \sigma_0(v_r) M(V,T) d\mu dV . \tag{4.59}$$

From Eq. (4.53), we can obtain the Jacobian transformation between v_r and μ:

$$d\mu = \frac{v_r dv_r}{vV};\ [-1,\ 1] \rightarrow \left[\left|v-V\right|,\ \left|v+V\right|\right]. \tag{4.60}$$

Inserting Eq. (4.60) to Eq. (4.59) gives:

$$v\sigma_T(v) = \frac{1}{2v}\int_0^\infty \left[\int_{|v-V|}^{|v+V|} \left[v_r \sigma_0(v_r)\right] v_r dv_r\right] \frac{M(V,T)}{V} dV . \tag{4.61}$$

Converting the order of integrals in Eq. (4.61) yields:

$$v\sigma_T(v) = \frac{1}{2v}\int_0^\infty \left[\int_{|v-v_r|}^{|v+v_r|} \frac{M(V,T)}{V} dV\right] \left[v_r \sigma_0(v_r)\right] v_r dv_r . \tag{4.62}$$

Plugging in the Maxwell-Boltzmann distribution in Eq. (4.62) leads to the final form of the Doppler broadening kernel:

$$\begin{aligned} v\sigma_T(v) &= \frac{1}{2v}\int_0^\infty \left[\int_{|v-v_r|}^{|v+v_r|} \frac{4}{\sqrt{\pi}} \beta^3 V e^{-\beta^2 V^2} dV\right] \left[v_r \sigma_0(v_r)\right] v_r dv_r \\ &= \frac{1}{v}\frac{\beta}{\sqrt{\pi}}\int_0^\infty \left[e^{-\beta^2 (v-v_r)^2} - e^{-\beta^2 (v+v_r)^2}\right] \left[v_r \sigma_0(v_r)\right] v_r dv_r . \end{aligned} \tag{4.63}$$

The RST scheme was motivated by the conversion of the order of integrals in Eq. (4.62) and the idea of the Relative Velocity Sampling (RVS) scheme suggested by Romano and Walsh [100] to change the order of sampling such that the relative speed is sampled first instead of the target speed. The conversion can be seen as the shift from a Riemann integral to a Lebesgue integral as illustrated in Figure 4.35.

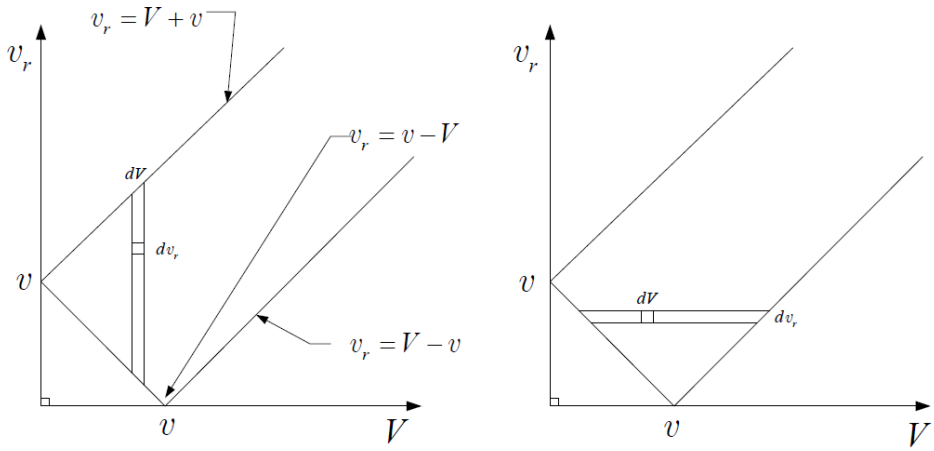


Figure 4.35 Schematic diagram of the Riemann (left) and the Lebesgue (right) integrals.

That is, the expected reaction rate can be considered as the sum of the contributions of reactions having certain relative speeds, and the contribution of each relative speed is described by the term $\left[e^{-\beta^2(v-v_r)^2} - e^{-\beta^2(v+v_r)^2}\right]\left[v_r\sigma_0(v_r)\right]v_r$ in the integral of (4.63). In addition, for each relative speed, Eq. (4.62) implies that the distribution of the target speed follows $\frac{M(V,T)}{V}$. This leads to an idea of sampling the relative speed first and then sample the target speed for the fixed relative speed, as opposed to sampling the target speed first and performing rejections in the conventional TVS schemes.

Therefore, as the first step of RST, the term $\left[e^{-\beta^2(v-v_r)^2} - e^{-\beta^2(v+v_r)^2}\right]\left[v_r\sigma_0(v_r)\right]v_r$ is tabulated into N points at each energy point in advance to the calculation, which becomes the probability distribution of the relative speed $P(v_r \mid v,T)$ for a given incident neutron energy and temperature. Like the DBRC scheme, the relative speeds are tabulated in the range of $\left[v - \frac{4}{\beta}, v + \frac{4}{\beta}\right]$.

At each elastic scattering, the relative speed is sampled from the preset $P(v_r \mid v, T)$. The table at the nearest energy point of the incident neutron energy is used. Once a relative speed is sampled, the target speed is sampled from $\frac{M(V,T)}{V}$ analytically. The CDF of $\frac{M(V,T)}{V}$ can be obtained as follows:

$$C(V) = 1 - e^{-\beta^2 V^2}. \tag{4.64}$$

By the inverse transformation of Eq. (4.64), the sampling formula for target speed is obtained as follows:

$$V = \frac{\sqrt{-\log(1-\xi)}}{\beta}. \tag{4.65}$$

Note that the physically available range of the target speed $[V_{\min}, V_{\max}]$ for a given relative speed can be determined as follows:

$$V_{\min} = |v_r - v|, \tag{4.66}$$

$$V_{\max} = v_r + v. \tag{4.67}$$

Using this fact, sampling of the target speed is done with a truncated CDF to avoid the rejection step on μ. First, the corresponding CDF values $C_{\min}$ and $C_{\max}$ for $V_{\min}$ and $V_{\max}$ are obtained:

$$C_{\min} = 1 - e^{-\beta^2 V_{\min}^2}, \tag{4.68}$$

$$C_{\max} = 1 - e^{-\beta^2 V_{\max}^2}. \tag{4.69}$$

Then, the random number is scaled in the range $[C_{\min}, C_{\max})$:

$$\xi \leftarrow C_{\min} + \xi(C_{\max} - C_{\min}). \tag{4.70}$$

Using the scaled random number, the target speed is sampled from Eq. (4.65). Once the target speed is sampled, μ can be calculated directly from Eq. (4.53).

To summarize, the sampling algorithm of the RST scheme is given as follows:

0. (Preprocessing) Tabulate $P(v_r \mid v, T) = \left[e^{-\beta^2 (v - v_r)^2} - e^{-\beta^2 (v + v_r)^2} \right] \left[v_r \sigma_0(v_r) \right] v_r$ in the range $\left[v - \frac{4}{\beta}, v + \frac{4}{\beta} \right]$ for each energy point.
1. Sample relative speed from $P(v_r \mid v, T)$.
2. With the sampled relative speed, determine $V_{\min}$ and $V_{\max}$.
3. Sample V from the truncated CDF.
4. Determine μ.

4.4.3 Comparison of the TVS Schemes

Comparison of Scattering Kernels

First of all, the indispensableness of using dedicated TVS schemes is presented by comparing the CXS scheme and the DBRC scheme near the scattering resonance of U-238. Figure 4.36 and Figure 4.37 compare the scattering kernels of CXS and DBRC for different temperatures at 36.25eV and 28.78eV, respectively. 36.25eV is the vicinity of the representative 36.682eV resonance of U-238, and 28.78eV is the middle of 20.871eV and 36.682eV resonances.

As can be seen, CXS largely underestimates the up-scattering reactions caused by the resonance scattering, especially at higher temperatures. While CXS is valid for energies away from the resonances as shown in the 28.78eV case, its inaccurateness around the resonance energies at high temperatures results in the overestimation of reactivity and yields misleading temperature coefficients which are the key safety factors in the power reactors. Furthermore, due to the wrong flux spectra around the resonances, the resonance absorptions are not calculated properly, and this incurs errors in the nuclide inventory during depletion. Therefore, a dedicated TVS scheme should be used for U-238.

Meanwhile, it is confirmed that DBRC and WCM yield equivalent results, as can be seen in Figure 4.38 which compares the resonance scattering kernels of DBRC and WCM at 36.25eV. It had been known that WCM is equivalent to DBRC [107], and this confirms that the implementation of WCM in PRAGMA is correct. However, WCM is inappropriate to be applied due to the issues with the weight adjustment, which will be shown in the upcoming paragraphs.

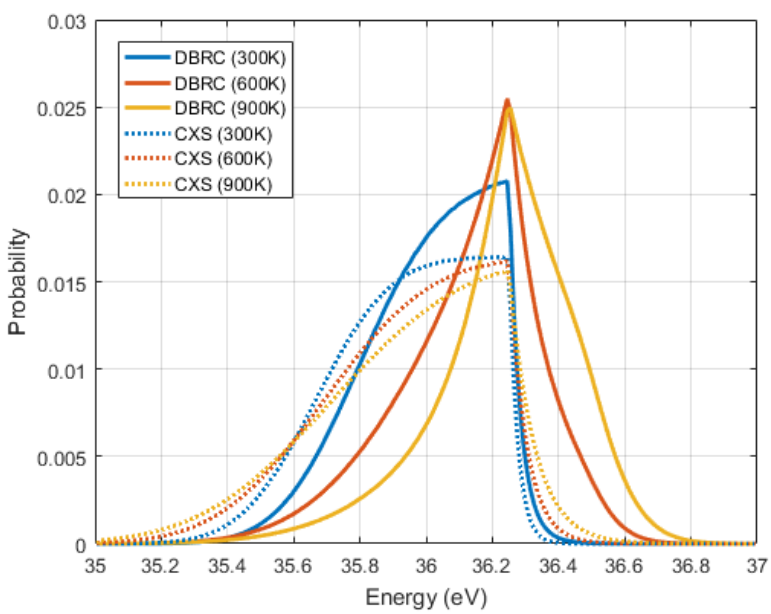


Figure 4.36 Comparison of U-238 scattering kernels of DBRC and CXS for neutrons injected at 36.25eV.

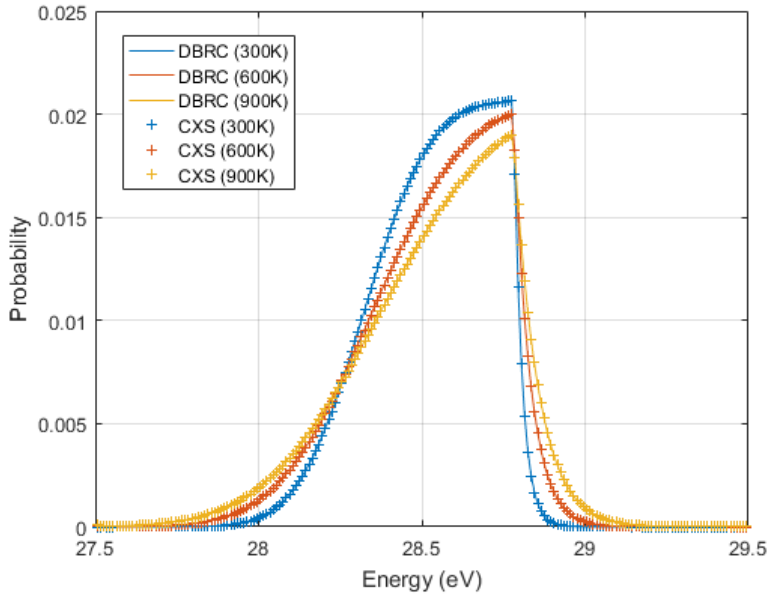


Figure 4.37 Comparison of U-238 scattering kernels of DBRC and CXS for neutrons injected at 28.78eV.

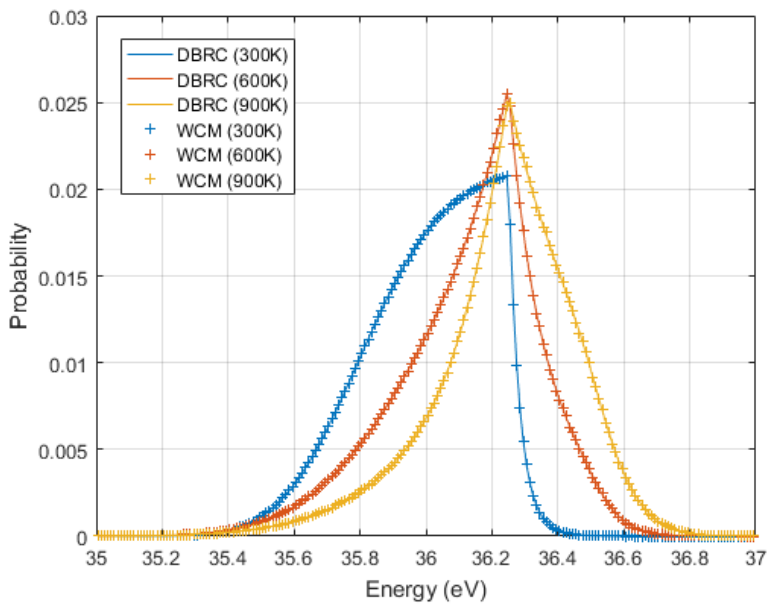


Figure 4.38 Comparison of U-238 scattering kernels of DBRC and WCM for neutrons injected at 36.25eV.

Next, the scattering kernels of RST were compared with those of DBRC to verify the accuracy of RST. The up-scattering percentages of the scattering kernels near the first four scattering resonances of U-238 were compared, as shown in Table 4.12. N is the number of tabulation points for the relative speed. 100 million samples were used and the uncertainties in the up-scattering percentages can be neglected.

Table 4.12 Comparison of up-scattering percentages of DBRC and RST at different energies and temperatures.

Energy (eV)	Temperature (K)	DBRC	RST (N = 200)	RST (N = 400)
6.52	300	62.2	61.5	62.1
	600	82.8	82.2	82.6
	900	84.4	83.8	84.1
20.2	300	5.67	5.28	5.38
	600	15.5	14.5	14.7
	900	26.8	25.5	25.8
36.25	300	7.14	7.06	7.21
	600	30.6	30.4	30.9
	900	51.5	50.6	51.3
65.8	300	4.50	4.36	4.42
	600	8.10	7.79	7.93
	900	10.8	10.3	10.5

Since RST is a tabulation method, it is inevitable to have inherent errors coming from the discretization. The maximum error in the up-scattering percentage was 1.3%p observed at 20.2eV, and there is a trend of underestimation in the up-scattering percentages compared to DBRC. In addition, changing N makes slightly different results. Nonetheless, the errors are not significant and RST gives almost equivalent results for the scattering kernels with DBRC, as illustrated in Figure 4.39. Therefore, it can be insisted that RST and DBRC are consistent. Regarding the choice of N, using 400 gives slightly more accurate results than using 200, but the differences are marginal. Therefore, we choose N as 200 for the subsequent calculations.

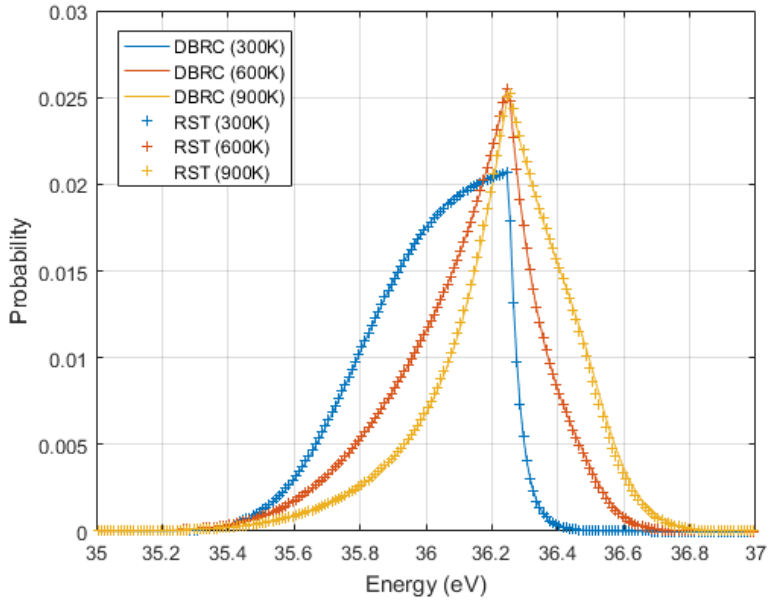


Figure 4.39 Comparison of U-238 scattering kernels of RST and DBRC for neutrons injected at 36.25eV.

The major limitation of RST, however, is that the relative speed probabilities are tabulated at discrete temperature points. Therefore, the accuracy of RST under the temperature interpolation should be examined. As mentioned in the previous chapter, the temperature interpolation in PRAGMA is done by stochastic mixing. Using the same strategy, the interpolation of the RST scattering kernels was investigated. Using the tables generated at 800K and 1000K, it was examined whether the scattering kernel of 900K can be reproduced. Same test was done for the 600K scattering kernel using the tables generated at 500K and 700K.

As can be seen in Figure 4.40, the interpolated scattering kernels closely follow the scattering kernels generated exactly at the specified temperatures. In the actual calculations of PRAGMA involving temperature variation, cross sections are given with less than 100K interval, so the interpolation with 200K interval is already a very coarse interpolation. Therefore, it is expected that there will be no accuracy issue regarding the temperature interpolation in actual power reactor calculations.

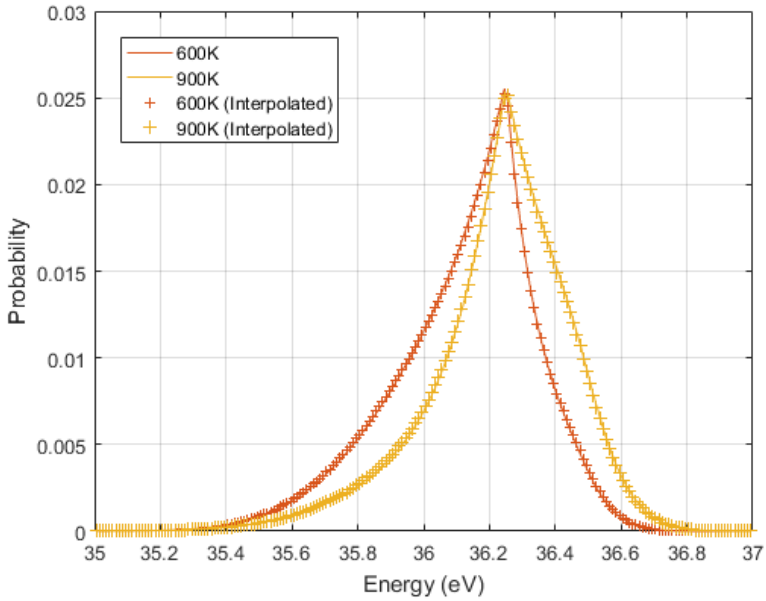


Figure 4.40 Comparison of exact and interpolated scattering kernels for neutrons injected at 36.25eV.

Comparison of Reactivity

The Mosteller Doppler reactivity defect benchmark problem [108] was tested for the verification of RST scheme in terms of the reactivity calculation. The RST results were compared with the DBRC results, and corresponding McCARD [15] DBRC results were included as the reference. McCARD used 500 active cycles and 200,000 neutrons per cycle which results in around 4 pcm of standard deviations. Meanwhile, PRAGMA employed 1,000 active cycles and 1,000,000 neutrons per cycle and the standard deviations of PRAGMA results are lower than 2 pcm.

Either DBRC or RST was applied to U-238 only, and the CXS scheme was applied for other nuclides. In addition, the TVS schemes are employed only below 210 eV; beyond that energy, asymptotic scattering assumption was used. Table 4.13 and Table 4.14 presents the eigenvalues of several enrichment cases at Hot Zero Power (HZP) and Hot Full Power (HFP) conditions, respectively.

Table 4.13 Eigenvalues of different TVS schemes for the HZP cases.

Enrichment	McCARD (DBRC)	CXS	DBRC	RST	RST − DBRC (pcm)
0.7%	0.66559	0.66631	0.66561	0.66563	2
1.6%	0.96074	0.96180	0.96073	0.96079	6
2.4%	1.09907	1.10011	1.09893	1.09896	3
3.1%	1.17688	1.17825	1.17693	1.17698	5
3.9%	1.23959	1.24089	1.23957	1.23967	10
4.5%	1.27499	1.27631	1.27501	1.27508	7
5.0%	1.29935	1.30063	1.29929	1.29943	14

Table 4.14 Eigenvalues of different TVS schemes for the HFP cases.

Enrichment	McCARD (DBRC)	CXS	DBRC	RST	RST − DBRC (pcm)
0.7%	0.65906	0.66045	0.65915	0.65922	7
1.6%	0.95169	0.95358	0.95165	0.95175	10
2.4%	1.08898	1.09095	1.08882	1.08891	9
3.1%	1.16629	1.16858	1.16627	1.16640	13
3.9%	1.22865	1.23091	1.22853	1.22868	15
4.5%	1.26382	1.26619	1.26378	1.26394	16
5.0%	1.28815	1.29039	1.28799	1.28813	14

A consistent positive reactivity bias is observed in RST, and it is thought that the tendency of RST to underestimate the up-scattering percentage, which was shown in Table 4.12, is related with the bias. However, the degree of the bias is considered tolerable and the difference in the Doppler coefficient illustrated in Figure 4.41 is negligible. In addition, both DBRC and RST results agree with the McCARD results, which demonstrates the soundness of implementation. Meanwhile, it can be seen that the CXS scheme produces about 200 pcm reactivity errors in HFP cases due to the mistreatment of the resonance scattering of U-238, and this results in an incorrect estimation of the Doppler coefficients.

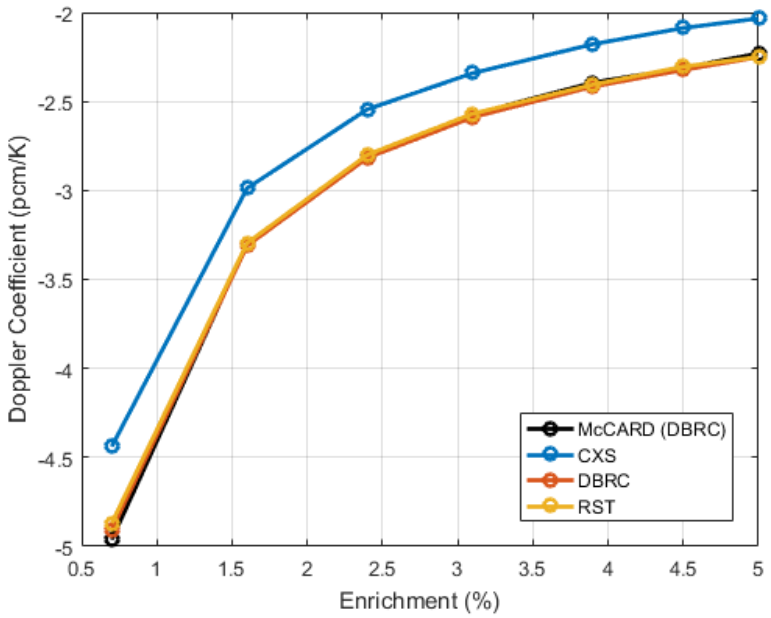


Figure 4.41 Comparison of Doppler coefficients of different TVS schemes.

Comparison of Performance

The computing time for simulating 120 million histories (20 inactive cycles, 100 active cycles, and 1M neutrons per cycle) with various TVS schemes for the 5.0% enrichment problem is presented in Table 4.15. A single NVIDIA GeForce RTX 2080 Ti GPU was used for the calculations.

WCM and RST do not add any overhead compared to CXS, while DBRC induces significant overhead to make the calculation infeasible. The extraordinarily poor performance of DBRC is due to the peculiar characteristic of GPUs that the entire GPU hangs by a few threads suffering from high rejection rates. Note that in the conventional MC calculations using CPUs, the DBRC scheme is known to incur about 10% overhead in the total computing time. Therefore, this result does not mean that the DBRC scheme is impractical, but that the DBRC scheme is not suitable for the GPU-based MC calculation, and this demonstrates the motivation of developing the RST scheme.

Table 4.15 Comparison of computing time (s) of different TVS schemes for the 5.0% enriched case.

Case	CXS	DBRC	WCM	RST
HZP	96.2	2633.7	97.0	95.8
HFP	95.0	4143.8	98.5	97.9

WCM was once considered as the main TVS scheme in PRAGMA as it does not rely on rejection, whose advantage was shown through the computing time. However, it turned out to be inadequate due to its low statistical quality and its possible hazard to cause extremely large weights.

Figure 4.42 illustrates the behavior of eigenvalues at each cycle of different TVS schemes. It can be clearly seen that WCM shows significantly larger deviation of eigenvalue than the other two schemes, and this is due to the occurrence of large weights. Although WCM is computationally efficient, it can harm the stability of the simulation and therefore cannot be utilized. As the result, neither DBRC nor WCM is suitable for GPU applications, and this shows the strength of RST.

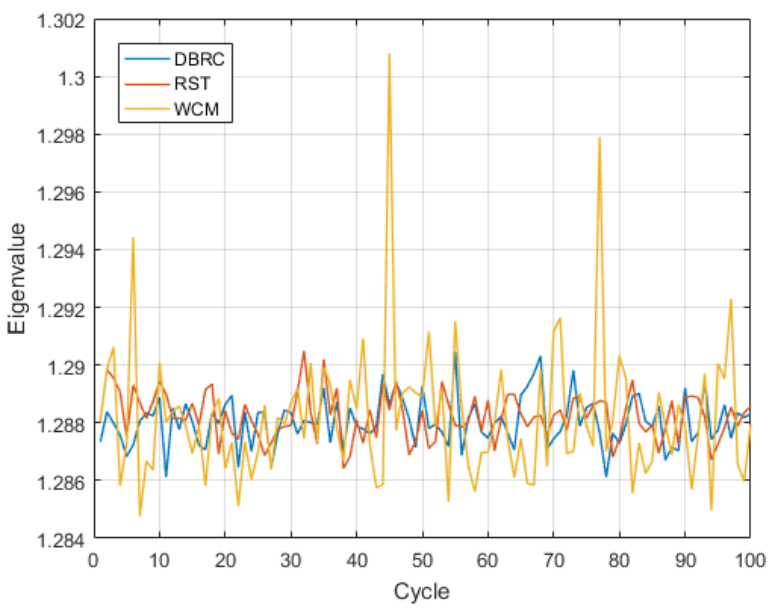


Figure 4.42 Trend of eigenvalues at each cycle of different TVS schemes for the 5.0% HFP case.

Memory Requirement

In the current implementation of RST in PRAGMA, the relative speed probabilities are tabulated at the temperatures for which the cross sections are given, and the energy points are also set identically to the cross section grids. As far as the point-wise cross section library of PRAGMA, which is based on ENDF/B-VII.1 and was processed by NJOY [93], is considered, it is observed for U-238 that around 8,000 energy points out of 170,000 exist below 210 eV. Since each energy point contains 200 points of relative speed points, the size of the table is 8,000 × 200 = 1,600,000. PRAGMA uses single precision by default and the memory size becomes 6.4 MB. Since both PDF and CDF have to be stored, the final memory requirement is 12.8 MB per temperature. For double precision, the requirement will be twice. PRAGMA typically employs 14 temperature-dependent cross section sets (550, 600, 700 … 1700, 1800) for power reactor calculations, and thus the total memory requirement is about 180 MB for U-238.

Therefore, the RST scheme will be burdensome if it is applied to all the nuclides. However, the common practice is to apply elaborated TVS schemes only for U-238, or for a few major actinides at most. In this regard, it is considered that there will be no memory issues in practical uses. Furthermore, using the 210 eV threshold might be too conservative and using the cross section grids for the tabulation might not be necessarily optimal, so there exist more rooms for reducing the memory usage.

4.5 Domain Decomposition

The MC method is ‘embarrassingly parallel’ if the problem can be copied in every process, since the particle parallelization can be naturally done by the independence of particles. However, GPUs have limited memory capacity and replicating the entire problem in each GPU is not feasible for large-scale core calculations. As the result, an explicit decomposition of problem domains is required to carry out whole-core MC simulations with GPUs.

This section presents about the domain decomposition algorithm of PRAGMA. An automated domain partitioning algorithm considering the characteristics of power reactors was developed, and an inner − outer iteration algorithm which overlaps communication and tracking was implemented. In addition, a surface source update algorithm for the peculiar bank structure of GPUs was devised.

4.5.1 Domain Partitioning Algorithm: Wheel Clustering

How to partition the domains has a significant implication to the load balance. In case of the deterministic methods, the computational load of each process is largely predictable through the number of discretization variables that each process has to calculate. On the other hand, the load balance in the MC domain decomposition is governed by the number of particles in each domain, and the source distribution is basically unknown until an actual calculation is performed. Therefore, having a good domain partitioning algorithm is crucial.

The domain partitioning can be considered as clustering unit geometry objects into a number of domains, and in this aspect, there exists the *k*-means clustering algorithm [109] which is a representative data clustering algorithm. As depicted in Figure 4.43, the algorithm aims to partition a given set of data objects into *k* clusters which are represented by centroids. Each data object is bound to a cluster whose centroid has

the nearest Euclidean distance. The centroids are relocated autonomously such that the squared sum of the distances to the data objects in each cluster is minimized. Once the centroids are repositioned, the data objects change their affiliations as well, and this iterative process continues until the centroids no longer move.

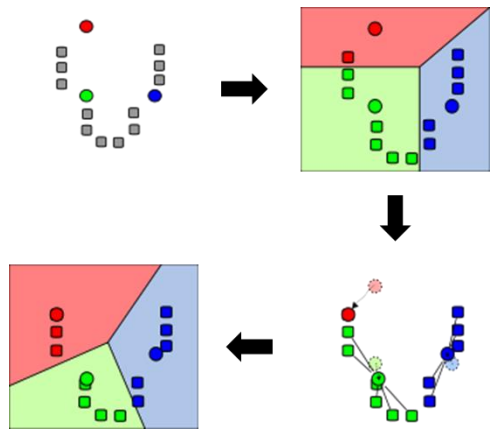


Figure 4.43 Example of *k*-means clustering [110].

The major drawback of the *k*-means algorithm is, however, that the results are not deterministic because the algorithm is basically an unsupervised learning algorithm and the initialization of the centroid positions rely on random numbers. In addition, there is no global minimum of the distance minimization problem in our application since the data objects (fuel assemblies) are uniformly distributed. As the result, the *k*-means algorithm results in a completely random clustering if it is applied to the domain partitioning problem. Therefore, the *k*-means clustering algorithm itself is inappropriate for domain partitioning, but it provides a useful idea.

By adopting the concept of the *k*-means algorithm that the centroids are iteratively relocated and each data object is repeatedly bound to its nearest centroid, a clustering algorithm named wheel clustering method was developed, which is deterministic and appropriate for the power reactor geometry. In this method, each assembly is treated as a point object. The method is illustrated in Figure 4.44 and the procedure is given in the description below:

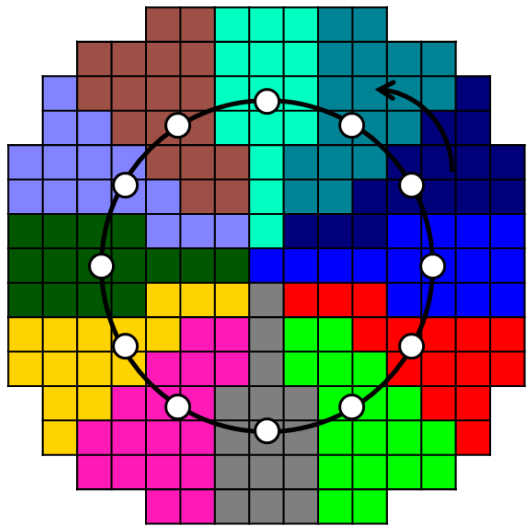


Figure 4.44 Illustration of domain partitioning with wheel clustering.

1. A ‘wheel’ centered at the core is defined. On the wheel, centroids corresponding to the number of domains are located with uniform spacing.
2. Each assembly is bound to the nearest centroid.
3. The wheel is rotated slightly and the assembly binding process is repeated.
4. Find the angle of wheel rotation in which the fuel assemblies are most uniformly distributed among the clusters, which become the domains.

In this implementation, it had been assumed that every fuel assembly has the same importance with each other. However, technically it is possible to assign appropriate weights to each assembly to have better load balance, for instance with the amount of fissile contents in each assembly. In addition, the wheel radius is set to 1.5 times of the assembly pitch, which is a quite small wheel, but one can end up with slightly different decompositions for different radii selections.

In fact, it is impossible to quantitatively prove that this scheme yields an optimal decomposition. One may consider decomposing domains to have high surface-to-volume ratios to be optimal in that it minimizes the number of escaping neutrons, which is not achievable in the wheel clustering algorithm. However, for the reactor analyses, the communication costs are a small fraction of the total computing time

and considerably more time is spent to the physics calculations [111]. Therefore, the priority should be put on minimizing the load imbalance rather than minimizing the communications. In this regard, the wheel clustering algorithm possesses a strength in that it exploits the characteristic azimuthal symmetry of the reactors and finds a decomposition map that has the most uniform fuel assembly distribution.

4.5.2 Inner − Outer Iteration Algorithm

Another issue would be on how to communicate efficiently the particles between processes. In the domain decomposed MC calculation, the tracking of a neutron must end when the neutron reaches the domain boundary. This neutron should serve as the source for the neighboring domain. As the result, the communication has to intervene the tracking loop, which cannot be done naturally in the conventional history-based tracking algorithm.

However, the GPU-based tracking algorithms divide the tracking loop into stages and therefore the intervention of communication is straightforward. Exploiting this feature, an inner − outer iteration algorithm was developed as illustrated in Figure 4.45, which overlaps the tracking and the surface source communications. The outer iteration is to drive a cycle; at each cycle, the outer iteration is initiated and continues until there is no more alive neutrons. At each outer iteration step, the tracking kernel is invoked for a fixed number (N) of times, which is the inner iteration. If a neutron escapes a domain during the inner iteration, it is stashed separately into the boundary source bank. After the inner iteration, the stashed neutrons are sorted according to their destinations, and non-blocking communications for sending and receiving the domain crossing neutrons are submitted to the network buffer.

The escaping neutrons return back to the tracking after two outer iteration steps as described below. Without synchronizing for the completion of the communications at the current outer iteration step, the next outer iteration step proceeds such that the

communications are overlapped with the subsequent tracking calculation. Instead, the communications which were submitted at the end of the previous outer iteration step finalized at the end of the current outer iteration step and the received neutrons are merged to the bank, which will be simulated at the next outer iteration step.

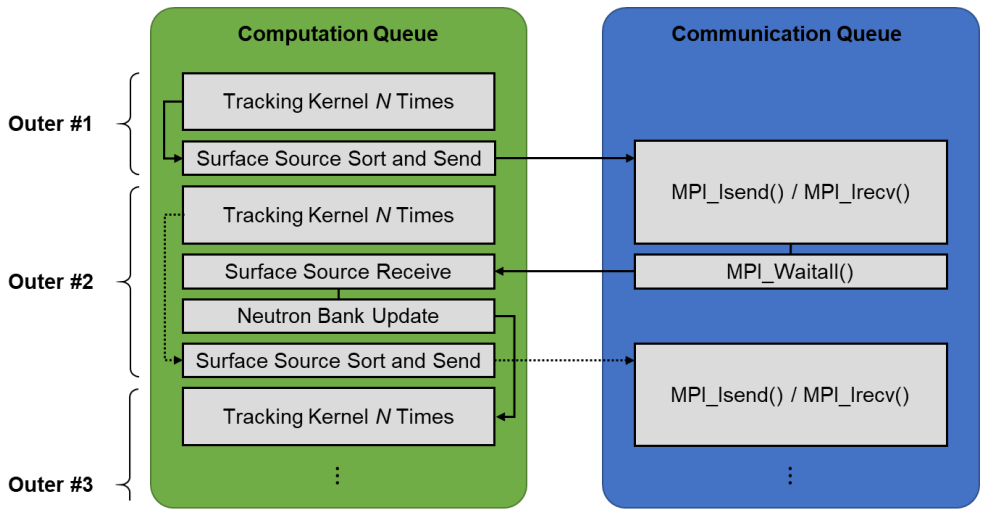


Figure 4.45 Schematic diagram of the inner – outer iteration scheme.

4.5.3 Surface Source Update Algorithm

The last issue is on how to bring back the surface source into the bank. The bank is implemented using arrays for the GPU-based MC calculation due to the sequential nature of the queue structure. This prevents a flexible resizing of the bank and thus a proper way to receive the incoming neutrons into the bank should be devised.

The solution for this problem is to recycle the storage of terminated neutrons. Since the neutron array does not resize in a cycle, terminated neutrons are still occupying their spaces. Finding the locations of terminated neutrons and replacing them with the surface sources is done by utilizing the remapping vector, as illustrated in Figure 4.46. By the flagged partitioning for the alive status, the remapping vector is split into two parts. The elements of the former and the latter parts point to the positions of alive and terminated neutrons, respectively. In updating the neutron bank with the

surface source neutrons, the latter part is referenced. After the incoming neutrons completely substitute the terminated neutrons, the alive status flags of the replaced positions are reset. Then, the remapping vector is partitioned again using the updated alive status vector.

The problem of the recycling is that there may be a chance of bank overflow at the very early stage of a cycle when more surface sources are incoming than the amount of available storage. In that case, the neutron array has to be resized and the surface sources should be appended to the end of the array, which incurs overheads. However, it is observed that such circumstances rarely occur due to the lagged update of the surface sources in the inner − outer iteration scheme.

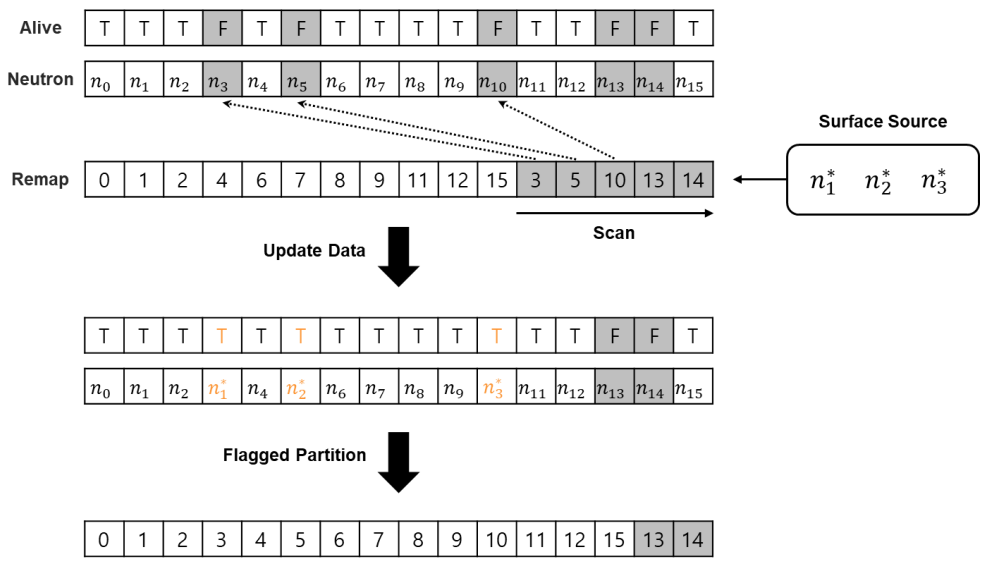


Figure 4.46 Example of the neutron bank update with surface sources.

4.5.4 Initial Verification

A mock-up depleted core problem with the depleted fuel composition of Table 4.8 was designed as illustrated in Figure 4.47, and it was examined whether the memory burden to simulate a full depleted core can be handled by the domain decomposition scheme. The performance investigations will be made at the end of this chapter.

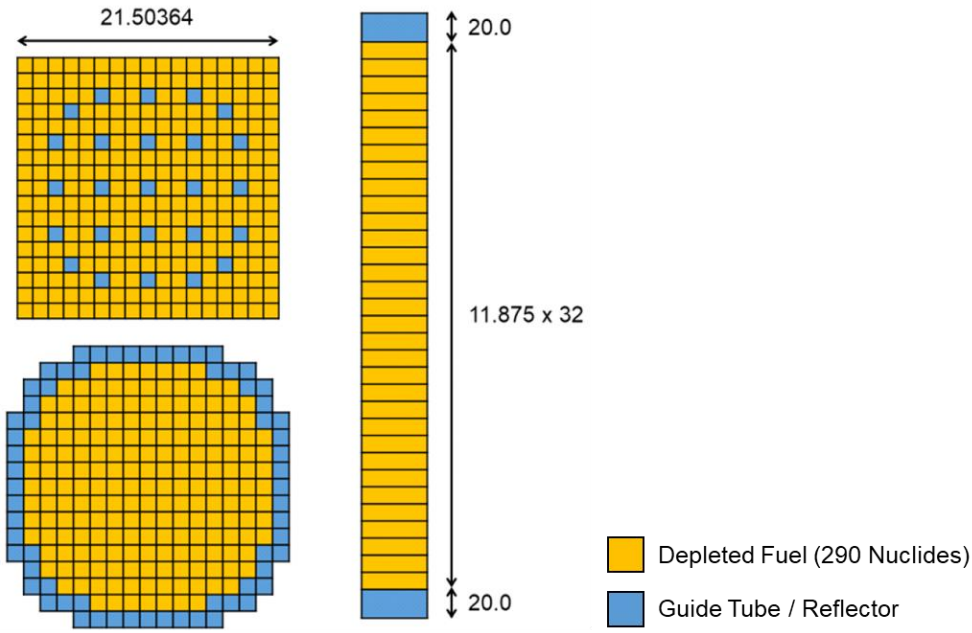


Figure 4.47 Configuration of the mock-up depleted core.

Table 4.16 shows the memory requirements depending on the problem size and the use of the domain decomposition scheme. The used memory include the storage for 300 million particles and cross sections at 14 temperature points. The problem was distributed to 24 GPUs whose total memory is 258.3GB. Without the discretization of the fuel pellets, the ordinary scheme can barely hold the memory requirement, but it quickly goes out of memory (OOM) with the realistic discretization condition. On the other hand, domain decomposition largely alleviates the memory burden for the geometry handling, and sufficient amount of free memory which will be used for the tallies can be retained even with fine discretization.

Table 4.16 Comparison of memory consumptions depending on the application of domain decomposition.

Number of Fuel Subdivisions	1		5	
Number of Fuel Regions	1,630,464		8,152,320	
Number of Material Regions	8,466,184		14,988,020	
Domain Decomposition	On	Off	On	Off
Used Memory (GB)	130.7	185.7	138.2	OOM

4.6 Feedback Calculations

This section briefly explains the feedback capabilities of PRAGMA for the power reactor applications. The feedback capabilities in PRAGMA include T/H feedback, xenon equilibrium, and critical searches with boron concentration and control rod position, which are all essential for analyzing power reactors at operating conditions.

4.6.1 T/H Feedback

The T/H feedback model in PRAGMA is so-called 1D single-phase close-channel model, which has the following assumptions:

1. Fuel rod is an azimuthally symmetric cylinder with no axial heat transfer.
2. Coolant is single-phase and coolant channels do not interact with each other.

For the heat conduction in pellet and cladding, the 1D heat conduction equation with respect to r is solved for each axial segment of a pin:

$$-\frac{1}{r}\frac{\partial}{\partial r}\left(rk(r)\frac{\partial T}{\partial r}\right) = (1-\gamma)q''' . \tag{4.71}$$

Note that the spatial dependence of heat source is neglected; only pellet-averaged power is provided from the MC solution. In addition, 4.5% of the heat is assumed to be directly transferred to moderator as gamma rays. However, the spatial dependence of conductivity is retained, which uses the FRAPCON-4.0 [112] correlations.

For the gap heat transfer, on the other hand, the heat convection equation is solved:

$$q'' = h\Delta T . \tag{4.72}$$

The conduction equation is solved with the point-scheme finite difference method. The number of discretization points in gap and cladding are fixed to two and three,

respectively, and only the number of fuel discretization points can be adjusted. The discretization of the fuel rod is illustrated in Figure 4.48.

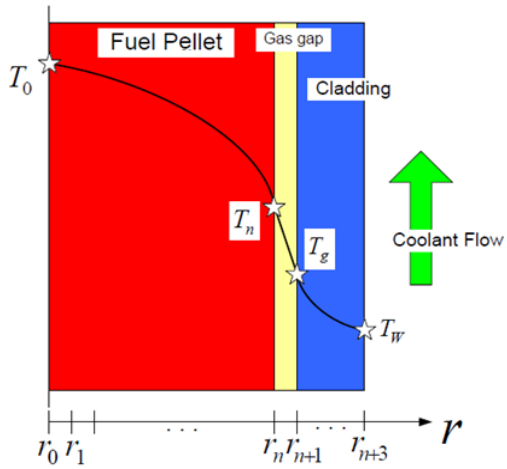


Figure 4.48 Illustration of fuel rod discretization.

For the coolant, there is no hydraulics calculation; only the enthalpy conservation is considered in determining the temperature of the coolant:

$$[\rho hv]^i_{out} = [\rho hv]^i_{in} + \bar{q}_i \Delta z_i \,. \tag{4.73}$$

Since the hydraulics is neglected, the lateral mixing effect of the coolant cannot be considered properly. To compensate this deficiency, pin-wise flow channels in each assembly are averaged to a single channel, as illustrated in Figure 4.49. Namely, the moderator temperature is assumed to be uniform among the pins in each assembly.

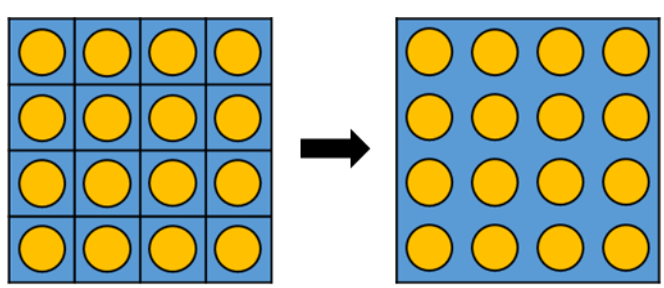


Figure 4.49 Illustration of lumping pin-wise flow channels.

4.6.2 Xenon Equilibrium

Xenon equilibrium is a stabilization technique for the depletion calculation. As the depletion calculation often uses large time steps, numerical xenon oscillation occurs and this incurs a cataclysmic change of the power distribution over time. To prevent the xenon oscillation, the Xe-135 number density is forced to be at equilibrium state during the MC simulation, as follows:

$$N_{Xe} = \frac{\sum_i \gamma_{Xe}^i \Sigma_f^i \phi}{\lambda_{Xe} + \sigma_a^{Xe} \phi} \tag{4.74}$$

where γ_{Xe}^i is the cumulative fission product yield of Xe-135 of fissile *i*, which is provided from MT = 459 data of the ENDF library.

The xenon equilibrium calculation is performed by updating the number densities of Xe-135 after each cycle with the flux and cross sections tallied during that cycle. The updated number densities are used in the next cycle and so on, which results in a successive iteration between neutronics and the Xe-135 number density update.

4.6.3 Critical Search

PRAGMA is able to perform critical searches with boron concentration and control rod bank position. Critical boron concentration (CBC) search is a routine procedure of the nuclear design and is one of the essential capabilities of the design codes. It is done by iteratively updating the boron concentration towards criticality by observing the eigenvalue changes and recalculating the boron number densities in the coolant with the updated boron concentration. Specifically, the boron worth δ is estimated using the difference of average eigenvalues $\bar{k}$ and boron concentrations c between two adjacent batches:

$$\delta_i \ (/\text{ppm}) = \frac{\Delta \bar{k}}{\Delta c} = \frac{\bar{k}_i - \bar{k}_{i-1}}{c_i - c_{i-1}} \tag{4.75}$$

where i is the current batch index. Using the boron worth, the boron concentration of the subsequent batch is updated as follows:

$$c_{i+1} = c_i + \frac{1 - \bar{k}_i}{\delta_i}. \tag{4.76}$$

However, naively applying the algorithm will likely fail in the MC method. Once the calculation is converged, only stochastic perturbations exist in the eigenvalue. Especially when it comes to massive particle simulations, the average eigenvalues of the adjacent batches will become very close to each other. As the result, the worth calculation can break down when either the boron concentration difference or the eigenvalue difference happens to be too small. In the former case, a very large worth is induced and the boron concentration cannot be updated anymore. In the latter case, the boron concentration can overshoot. To avoid this problem, the absolute value of the boron worth is forced to be within 5 to 20 pcm/ppm, which is an ad hoc treatment but still effective:

$$0.00005 \ \le \ |\delta| \ \le \ 0.0002 \, . \tag{4.77}$$

Performing critical searches with control rods is not a common practice. However, determining the critical position of control rods is an important task in actual reactor operations. Figure 4.50 shows the daily load change of the nuclear power plants in France who is famous for performing aggressive daily load following of their nuclear power plants. For this kind of operation, control rod is the only measure to adaptively change the reactor power in short period of time, as changing the power of reactors using soluble boron is slow and produces a lot of radioactive wastewater. While such

daily load following has not been common so far, it will likely be needed in the future as the capacity of renewable sources increases. In this regard, we expect that the load following scenarios will be considered in the future nuclear design, and the control rod search capability was implemented in PRAGMA.

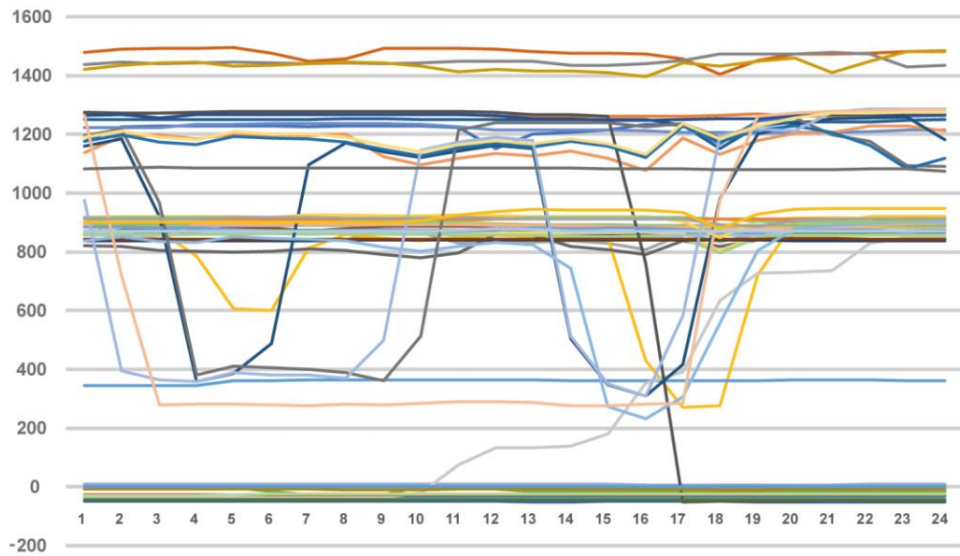


Figure 4.50 Unit-wise power production (MW) by the hour on the 13th of September 2015 for the entire nuclear fleet in France [113].

To perform the control rod search, user must specify the reference bank step s_{ref} and the reference eigenvalue k_{ref}. The remaining search process is analogous to the CBC search except that the derivative is now calculated with respect to the reference step and eigenvalue instead of using the difference of the two batches, as follows:

$$\delta_i \text{ (/step)} = \frac{\bar{k}_i - k_{ref}}{s_i - s_{ref}} \tag{4.78}$$

$$s_{i+1} = s_i + \frac{1 - \bar{k}_i}{\delta_i} \tag{4.79}$$

4.7 Depletion Calculation

Depletion calculation is important for predicting the long-term behavior of reactors. The nuclear energy originates from the reactions between neutrons and nuclei, and the nuclei undergoes transmutations as the time goes by the neutron-induced nuclear reactions. This process is referred to as burnup and is simulated by the core depletion calculation.

To perform depletion calculation, reaction rates of every nucleus should be known. Tallying and storing all the reaction rates entails enormous amount of computational overheads and memory requirements which reach terabytes. Thus, the conventional way of calculating reaction rates cannot be employed practically neither in terms of time nor resources.

In this regard, an alternative reaction rate generation technique is developed, which does not add tally overheads and can be realized with limited GPU memories. The developed scheme named Multilevel Spectral Collapse (MSC) [55] condenses point-wise cross sections into one group using flux spectra, but the condensation is done in two-level to reduce the memory usage. This section introduces the MSC scheme and presents initial verification results on its accuracy.

4.7.1 Theory and Methodology

The burnup process is described by the Bateman equation:

$$\frac{dN_i}{dt} = -(\lambda_i + \sigma_i \phi) N_i + \sum_j (\gamma_{j \to i} \lambda_j + \sigma_{j \to i} \phi) N_j \tag{4.80}$$

where

N_i = Number density of nuclide i,

λ_i = Decay constant of nuclide i,

$\gamma_{j\to i}$ = Decay fraction of nuclide j leading to nuclide i,

σ_i = Microscopic absorption cross section of nuclide i,

$\sigma_{j\to i}$ = Microscopic cross section of nuclide j leading to nuclide i.

The Bateman equation can be expressed as a time-dependent linear system:

$$\frac{d\mathbf{N}}{dt} = \mathbf{AN} \tag{4.81}$$

which gives the solution in terms of matrix exponential:

$$\mathbf{N}(t) = \exp(\mathbf{A}t)\mathbf{N}(0)\,. \tag{4.82}$$

Eq. (4.82) is solved by the Chebyshev Rational Approximation Method (CRAM) [114], which reduces Eq. (4.82) to:

$$\mathbf{N}(t) = \alpha_0\mathbf{N}(0) + 2\,\mathrm{Re}\left(\sum_{i=1}^{N/2}\alpha_i\left(\mathbf{A}t - \theta_i\mathbf{I}\right)^{-1}\mathbf{N}(0)\right) \tag{4.83}$$

where N is the order of CRAM and α and θ are the complex residues and poles, respectively, whose values are given as Table 4.17 and Table 4.18:

Table 4.17 Residues of CRAM of order 14.

	Real Part	Imaginary Part
α_0	1.832174378254041E−14	0.000000000000000E+00
α_1	- 7.154288063589067E−05	1.436104334854130E−04
α_2	9.439025310736168E−03	- 1.718479195848301E−02
α_3	- 3.763600387822696E−01	3.351834702945010E−01
α_4	- 2.349823209108270E+01	- 5.808359129714207E+00
α_5	4.693327448883129E+01	4.564364976882776E+01
α_6	- 2.787516194014564E+01	- 1.021473399901565E+02
α_7	4.807112098832508E+00	- 1.320979383742872E+00

Table 4.18 Poles of CRAM of order 14.

	Real Part	Imaginary Part
θ_1	- 8.897773186468888E+00	1.663098261990209E+01
θ_2	- 3.703275049423448E+00	1.365637187148327E+01
θ_3	- 0.208758638250130E+00	1.099126056190126E+01
θ_4	3.993369710578568E+00	6.004831642235037E+00
θ_5	5.089345060580624E+00	3.588824029027006E+00
θ_6	5.623142572745977E+00	1.194069046343966E+00
θ_7	2.269783829231112E+00	8.461797973040221E+00

The matrix inversion in Eq. (4.83) is performed iteratively using the Gauss-Seidel method. The matrix $\mathbf{A}t - \theta_i\mathbf{I}$ is basically sparse and has large diagonal terms owing to the subtraction of poles from the diagonal elements. Thus, the convergence of the Gauss-Seidel iteration is mostly guaranteed and also it converges very quickly. As the result, using the Gauss-Seidel method is much cheaper than the direct inversion.

As the time marching scheme, predictor-corrector method is employed. Using semi predictor-corrector method to reduce neutronics solution time is a common practice. However, unless short time steps are used, full predictor-corrector method should be used in MC depletion calculations for the sake of stability, especially in 3D problems. A small asymmetry in the power distribution caused by uncertainty can be amplified through the large time steps (typically weeks), which induces oscillations. In the full predictor-corrector scheme, the neutronics solution obtained after the corrector step counteracts the asymmetry and can stabilize the depletion calculation.

4.7.2 Multilevel Spectral Collapse (MSC) Scheme

Tallying all the reaction rates online is extremely time- and memory-consuming as it requires calculating all the required cross sections at each migration of neutron and the voluminous buffers to store the reaction rates. Thus, PRAGMA takes a different approach, which utilizes flux spectra to calculate the reaction rates posteriorly by the group condensation of point-wise cross sections. However, the flux-based approach is still prohibitive in terms of memory usage as it requires the flux spectra of tens of thousands of groups to be tallied in each depletion region.

In order to significantly reduce the memory usage of the flux-based approach, the MSC scheme specialized for the power reactor characteristics was developed. The idea of MSC is to employ two-level flux tally domains with different geometry and energy granularities. The first-level tally domain is the monolithic axial pin on which ultra-fine-group flux spectra are tallied. The second-level tally domain becomes the unit depletion region and multi-group flux spectra with hundreds of groups are tallied. The pin-wise ultra-fine-group spectra capture the detailed resonance shapes that are expected to be common along the depletion regions in the pin, and the region-wise multi-group flux spectra provide the information of actual flux levels. This two-level flux spectra tally is schematically illustrated in Figure 4.51.

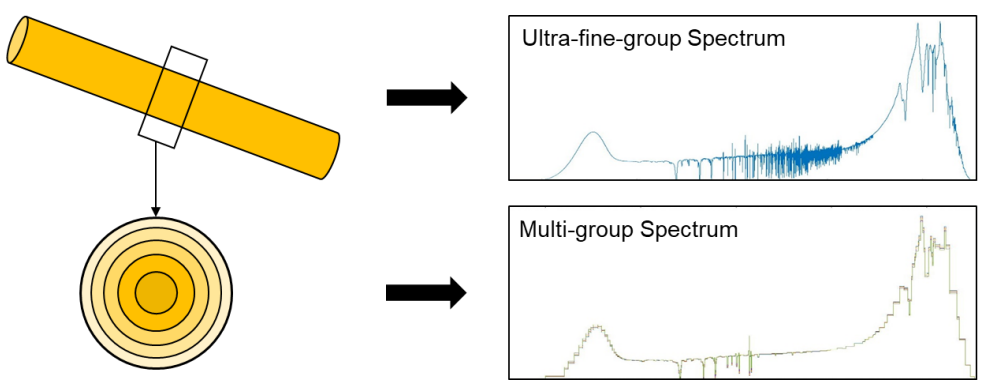


Figure 4.51 Schematic diagram of two-level flux spectra tally in the MSC scheme.

After tracking, multi-group cross sections are generated in each axial pin using the tallied ultra-fine-group flux spectra:

$$\sigma_g(T) = \frac{\int_{E_g}^{E_{g-1}} \sigma(E,T)\phi(E)dE}{\int_{E_g}^{E_{g-1}} \phi(E)dE}. \tag{4.84}$$

Then, for each depletion region in the pin, one-group microscopic reaction rates are calculated using the shared multi-group cross sections and locally tallied multi-group flux spectra:

$$\sigma\phi = \sum_g \sigma_g(T)\phi_g . \tag{4.85}$$

In the multi-group flux spectra, the majority of the energy groups are placed in the thermal energy range, because the thermal flux spectra can vary along the depletion regions in a pin. For example, build-up of plutonium or depletion of burnable poison which affects the thermal flux spectra significantly is highly dependent to the burnup exposure which is different region-by-region. In that case, the thermal flux shape of the pin-averaged ultra-fine-group spectrum cannot represent the thermal flux shape of the actual spectrum of the region, which will lead to large errors in the thermal reaction rates. Thus, the thermal energy range is finely discretized in the multi-group flux spectra in order to minimize the error from using the incorrect flux shapes.

This concept is fairly simple, but it is very effective in terms of computational cost and memory usage. It is estimated that a typical 3D full-core power reactor model will generate approximately 10 million depletion regions and 50 thousand pins. For a reactor of this size, the memory requirement for tallying the flux spectra of 57,000 groups in each depletion region reaches 4 TB, while the corresponding MSC scheme using 500-group flux spectra in each depletion region can be realized with merely 60 GB of memory.

4.7.3 Initial Verification

First of all, the validity of using ultra-fine-group flux spectra to generate reaction rates was examined. To determine how many groups are required for the ultra-fine-group flux spectra to obtain accurate reaction rates, a sensitivity test on the number of ultra-fine groups was performed. Figure 4.52 illustrates the microscopic capture reaction rate of U-238 depending on the number of ultra-fine groups, which indicates that approximately 57,000 groups are required to obtain a converged reaction rate.

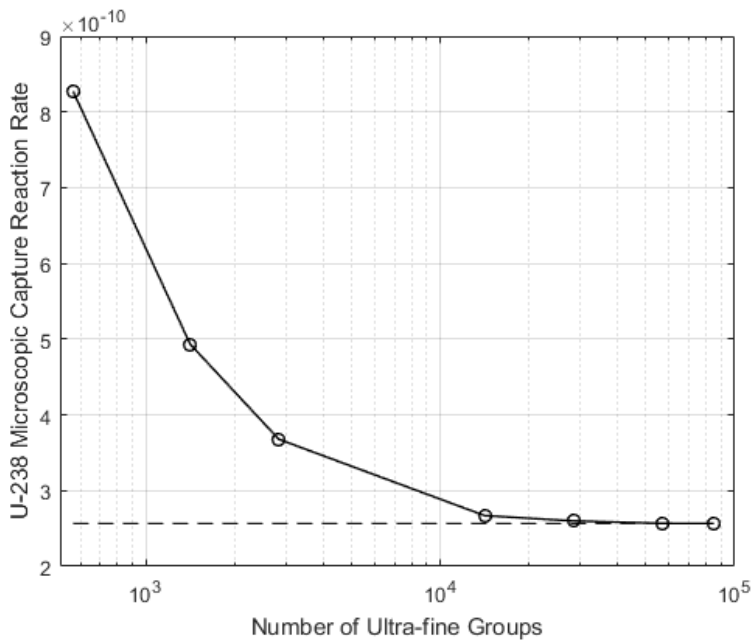


Figure 4.52 Reaction rate sensitivity to the number of ultra-fine groups.

Using the 57,000 group flux spectra, APR1400 [67] pin cell depletion calculations were performed and the results were compared with McCARD which employs the online reaction rate tally scheme. Figure 4.53 shows the letdown curves of McCARD and PRAGMA for three pin cells with different enrichments. The two codes agreed within 20 pcm, where the standard deviations were 2 pcm. This result demonstrates the soundness of the depletion solver in PRAGMA and the validity of using ultra-fine-group flux spectra for reaction rate calculation.

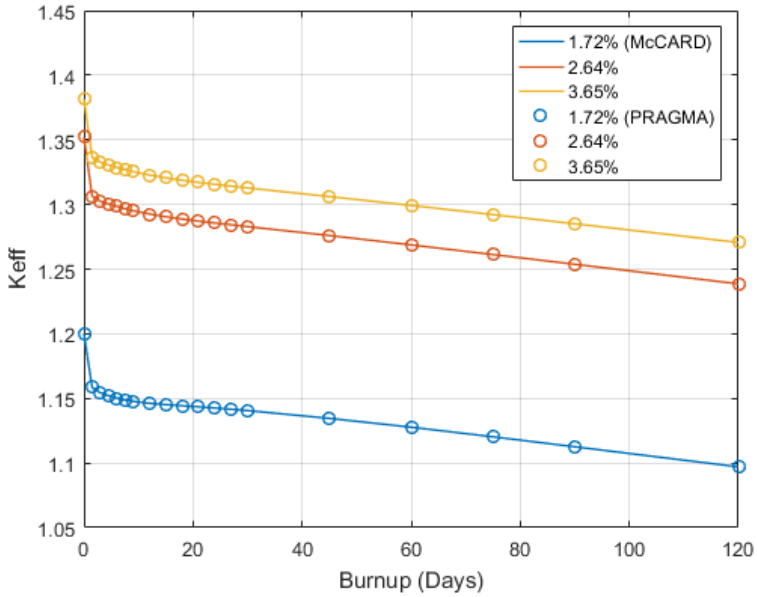


Figure 4.53 APR1400 pin cell depletion letdown curves of PRAGMA with ultra-fine-group spectra and McCARD.

Next, assembly depletion calculations were performed using the MSC scheme and the results were compared with the ultra-fine-group reference which employs ultra-fine-group flux spectra in every depletion region. The number of groups in the multi-group flux spectra is set to 500, where more than half of the groups are placed below 1 eV. Figure 4.54 and Figure 4.55 illustrate the letdown curves of APR1400 2D C0 and C3 assembly depletion calculations. The standard deviations are 2 pcm.

For the C0 assembly that does not contain Gadolinium (Gd) burnable poisons, the error is well-contained within 15 pcm. On the other hand, for the C3 assembly which has the highest Gadolinium content, the error increases up to 30 pcm and a trend in the error is observed, which implies the existence of biases. It is due to the rim effect of Gd; the depletion regions in a Gd-bearing pin have varying thermal flux spectra depending on the amount of Gd residues, which cannot be properly reflected in the pin-averaged ultra-fine-group flux spectra. However, the degree of error is kept at a tolerable level, which is due to the fine division of thermal groups in the multi-group flux spectra.

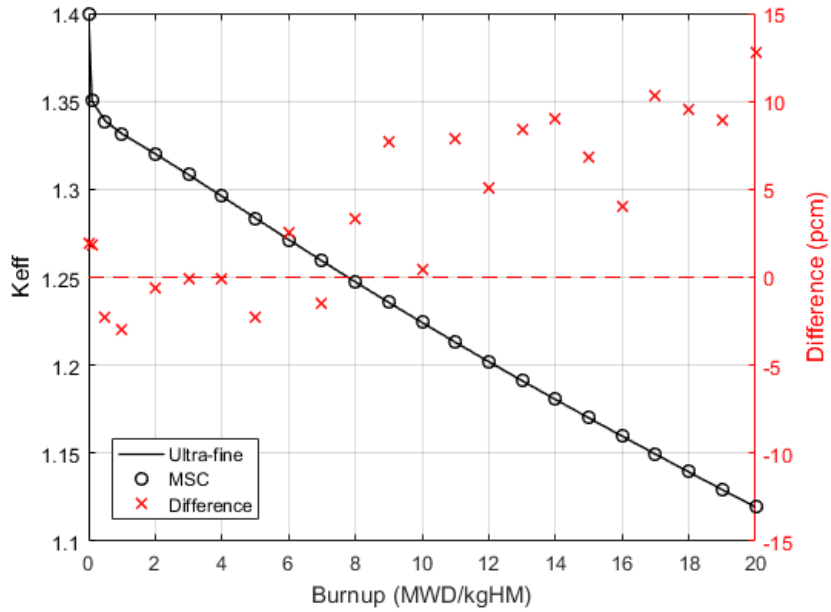


Figure 4.54 APR1400 2D C0 assembly depletion letdown curves of the ultra-fine-group reference and MSC.

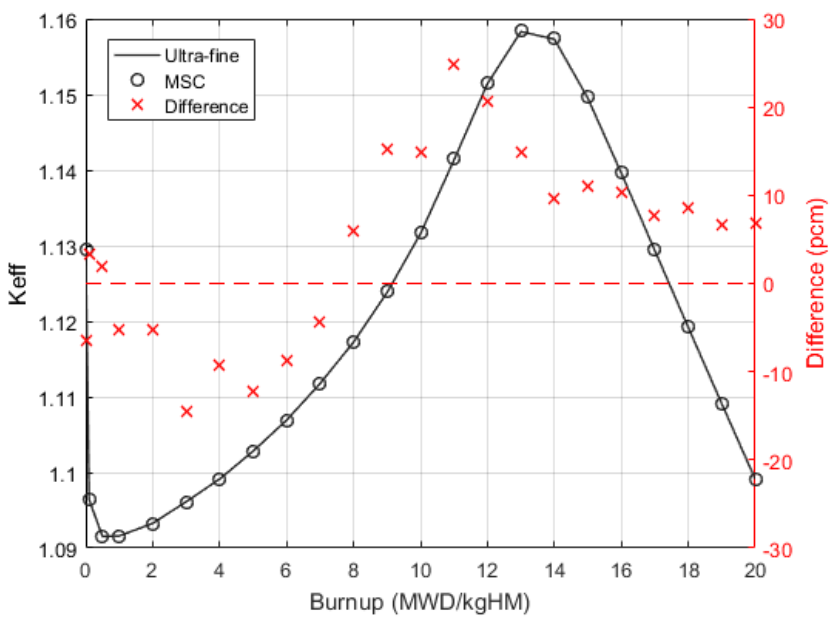


Figure 4.55 APR1400 2D C3 assembly depletion letdown curves of the ultra-fine-group reference and MSC.

The MSC scheme was also valid in 3D calculations. Figure 4.56 and Figure 4.57 demonstrate the letdown curves of 3D C0 and C3 assembly depletion calculations, where the degree of errors is more-or-less the same with the 2D cases. The primary concern of the MSC scheme is that the scheme may yield large errors if the spectral characteristics vary axially in the pin. However, the 3D assembly depletion results indicate that the spectral characteristics along axial direction in each pin stay uniform during depletion, which validates the MSC scheme.

Figure 4.58 illustrates the RMS error of axial powers between the ultra-fine-group reference and MSC solution at each burnup step of the 3D C3 assembly. In fact, it is difficult to rigorously investigate the axial power errors due to the uncertainties and oscillations. Thus, the difference of the two independent ultra-fine-group references was also examined. It can be seen that the degree of error between the reference and the MSC solution is not different from the errors between the two independent ultra-fine-group references. Namely, the error of the MSC solution is small enough such that it is not distinguishable from the effect of uncertainties and oscillations.

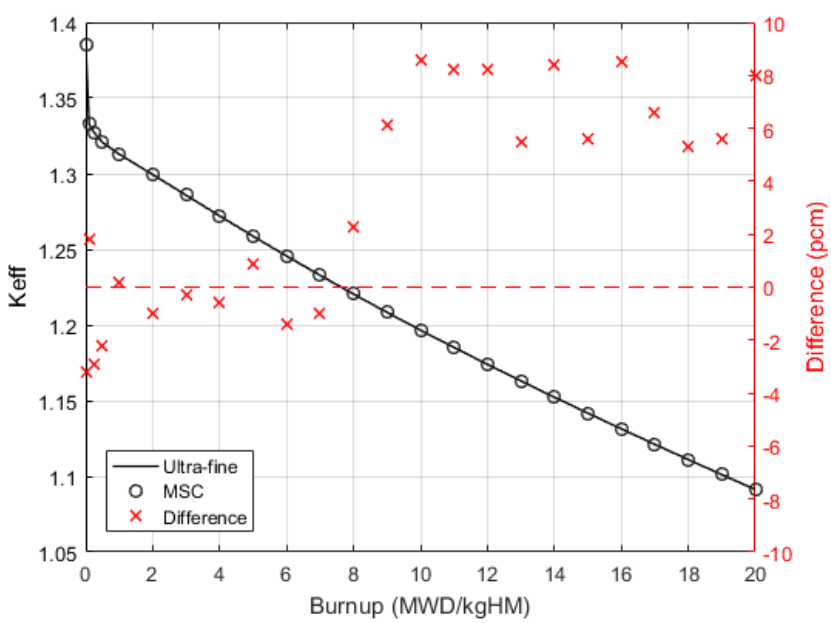


Figure 4.56 APR1400 3D C0 assembly depletion letdown curves of the ultra-fine-group reference and MSC.

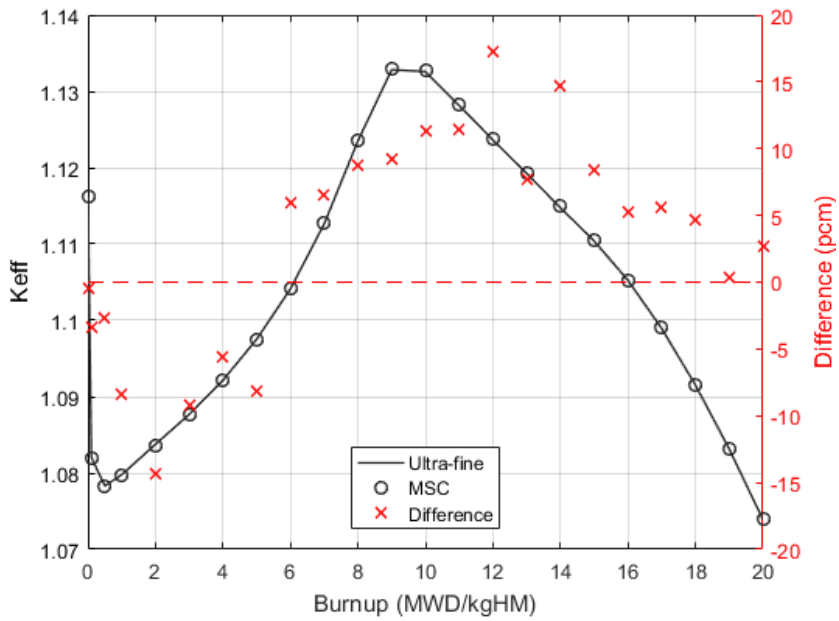


Figure 4.57 APR1400 3D C3 assembly depletion letdown curves of the ultra-fine-group reference and MSC.

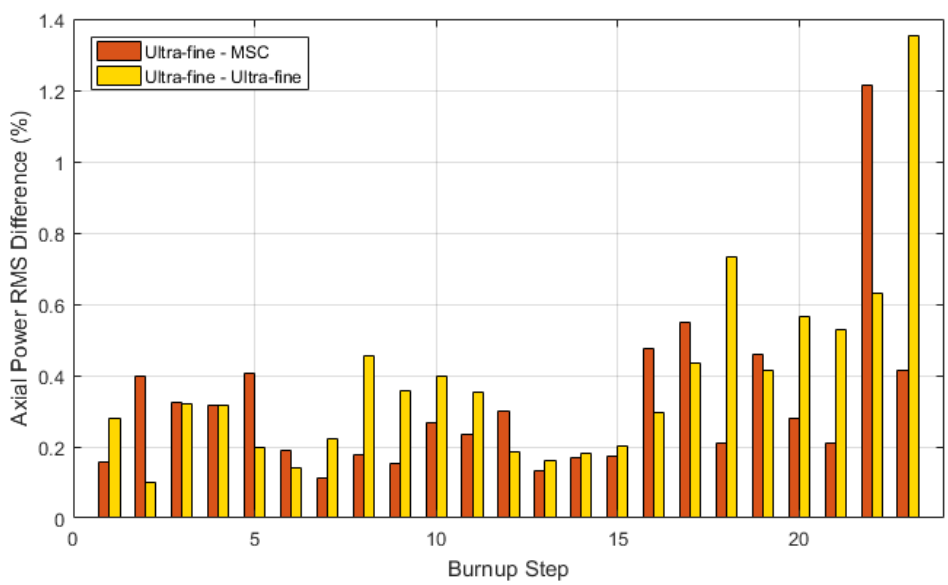


Figure 4.58 Axial power RMS differences in each burnup step of APR1400 3D C3 assembly depletion.

Lastly, the effect of xenon equilibrium treatment was examined. Figure 4.59 shows the evolution of axial power during the depletion of APR1400 3D B0 assembly. If the xenon equilibrium treatment is not employed, it can be seen that the axial power presents a cataclysmic behavior due to the numerical oscillation of xenon. With the xenon equilibrium treatment, however, the axial power evolution is well-stabilized. This verifies the soundness of the xenon equilibrium module in PRAGMA.

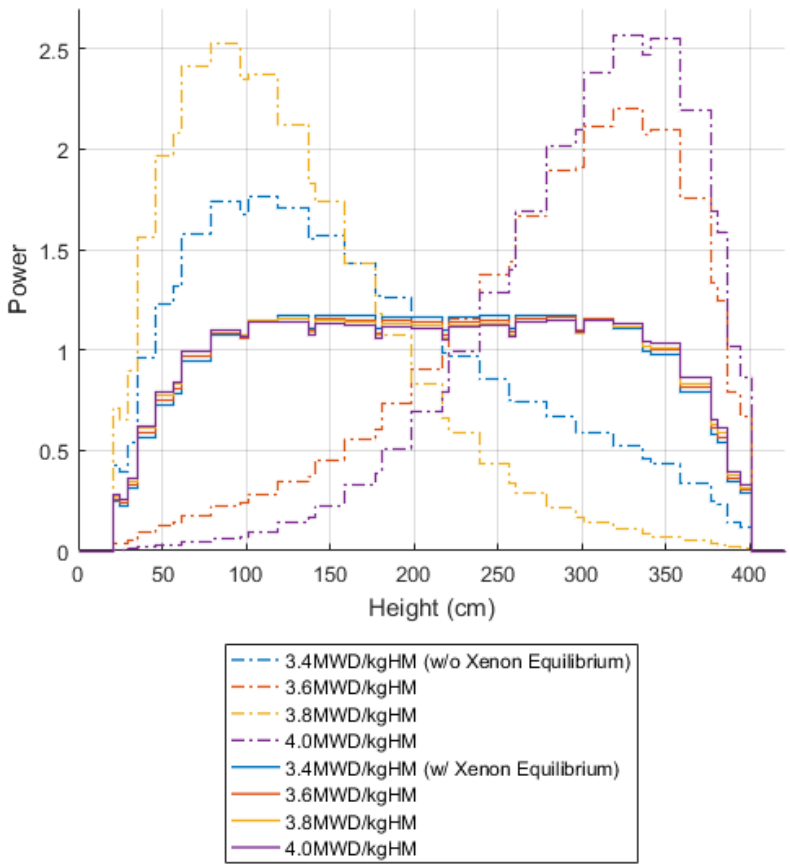


Figure 4.59 Effect of the xenon equilibrium treatment in APR1400 3D B0 assembly depletion.

4.8 Localized Delta-Tracking Scheme

Treatment of spatially varying quantities such as number densities or temperatures is cumbersome in the MC calculation. One of the major advantage of the MC method is that it can treat the geometries continuously. However, it is only valid either when there is no material variation in space or when a continuous representation of the material properties is available. But unfortunately, the material properties are not easily expressible as continuous functions. There have been efforts to continuously represent the material properties using functional expansion tally (FET) techniques, but none of the MC codes has been able to demonstrate their practical uses so far.

As the result, in the T/H coupled calculations and the depletion calculations, many regions should be used inside the fuel to capture the spatial variations of temperatures and number densities. This brings significant increases in the computational costs due to more frequent cross section calculations.

To circumvent this, PRAGMA employs a special tracking scheme called localized delta-tracking scheme. This scheme allows treating the temperature distributions in the fuel rods continuously, and prevents the degradation of the tracking performance coming from the discretization of the fuel pellets for the depletion calculations. This section introduces the localized delta-tracking scheme of PRAGMA.

4.8.1 Theory and Methodology

The delta-tracking scheme was developed by Woodcock et al [115]. In this scheme, the concept of virtual collision, which does not change the status of the neutron, is introduced. The DTC for the virtual collision is sampled as follows:

$$x' = -\frac{\ln(1-\xi)}{\Sigma_t^{\max}} \tag{4.86}$$

where $\Sigma_t^{\max}$ is the maximum total cross section in a domain. If a virtual collision had occurred, it is accepted as a physical collision with the probability determined as the ratio of the actual cross section of the colliding location to $\Sigma_t^{\max}$:

$$p_{col} = \frac{\Sigma_t}{\Sigma_t^{\max}}. \tag{4.87}$$

In this way, the number of physical collisions is preserved and the result is statically consistent with the ordinary tracking scheme.

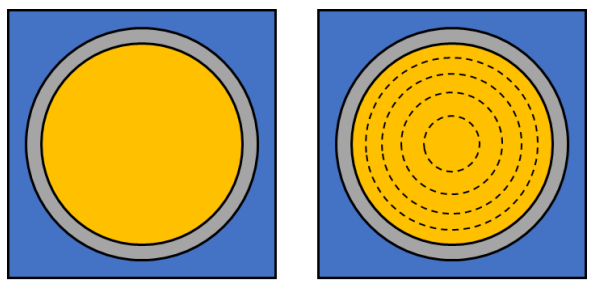
The significance of the delta-tracking scheme is that the DTS calculations can be omitted, since $\Sigma_t^{\max}$ is constant over the entire domain. That is, the performance of the delta-tracking scheme is virtually unaffected by the number of surfaces in the domain. Focusing on this advantage which implies that the material interfaces can be neglected in the tracking as far as $\Sigma_t^{\max}$ is known, Carter et al. [116] suggested an approach to treat functionalized cross sections over space with the delta-tracking scheme.

Motivated by the research of Carter, we had developed the localized delta-tracking scheme. The major drawback of the ordinary delta-tracking scheme which relies on the global maximum cross section is that it is difficult to find the maximum value itself. Therefore, it is typically circumvented by adding a fictitious term Σ^* to the cross section of the current position:

$$\Sigma_t^{\max} = \Sigma_t + \Sigma^* \tag{4.88}$$

The selection of Σ^* can be made arbitrarily, but it should be set sufficiently large so that the statistics is not biased. Therefore, the ordinary delta-tracking scheme often suffers from high rejection rates, especially when there are localized heavy absorbers in the domain.

The localized delta-tracking scheme localizes the range of the delta-tracking kernel and determine $\Sigma_t^{\max}$ in a more physically based way. To achieve this, the geometry model of PRAGMA adopts the concept of zones and regions. A region is the basic discretization unit in which the number densities are constant. On the other hand, a zone is where the constituent nuclides are identical, and it is determined by the initial mixture assignment. Usually, zones distinguish major core components such as fuel, gas gap, cladding, moderator, and structure. Figure 4.60 illustrates a pin cell example of zone and region definitions.



(a) **(b)**

Figure 4.60 Example of (a) zone and (b) region definitions.

The basic idea of the localized delta-tracking scheme is to neglect the interfaces of regions while considering the surfaces between zones in intersection calculations. That is, the range of the delta-tracking kernel is localized to a zone, and a standard geometry tracking algorithm is employed between the zones. Since it is ensured that the constituent nuclides are identical among the regions inside a zone, neither finding the global maximum cross section nor is adding a fictitious term to the cross section necessary. Instead, microscopic cross sections can be utilized.

Using the localized delta-tracking scheme, intra-fuel-rod temperature distributions are treated analytically. For each nuclide, temperature majorant microscopic total cross sections, denoted as σ_{maj}, are precomputed in the initialization phase. σ_{maj} is the maximum microscopic total cross section of a nuclide over the temperatures as

illustrated in Figure 4.61. Since the temperature-wise grids of each nuclide had been unified by the grid unification process, calculating σ_{maj} is a straightforward task.

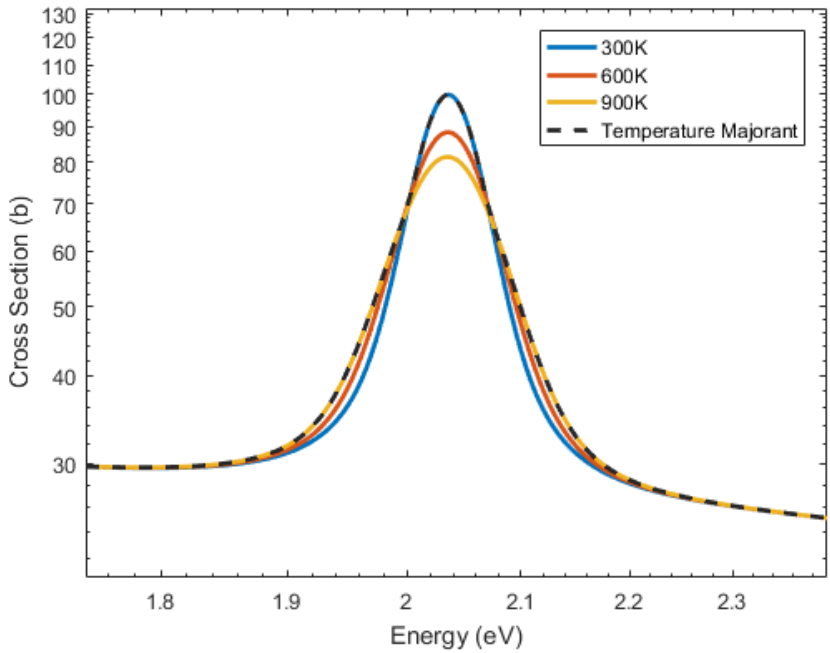


Figure 4.61 Temperature majorant microscopic total cross section of U-235.

Then, the maximum number density of each nuclide, denoted as $N_{\max}$, among the regions in the zone is set, and $\Sigma_t^{\max}$ of the zone is calculated as follows:

$$\Sigma_t^{\max} = \sum_{i=1}^{n} N_{\max}^{i} \sigma_{maj}^{i} \tag{4.89}$$

where n is the number of nuclides contained in the zone and i is the nuclide index.

The resulting $\Sigma_t^{\max}$ is consequently the maximum cross section reflecting both the temperature and the number density distributions in a zone. Figure 4.62 contrasts the actual geometry and the tracking geometry of the localized delta-tracking kernel. Regardless of the number of regions, the localized delta-tracking kernel will only see a single zone represented by σ_{maj}, $N_{\max}$, and the analytic temperature profile $T(r)$.

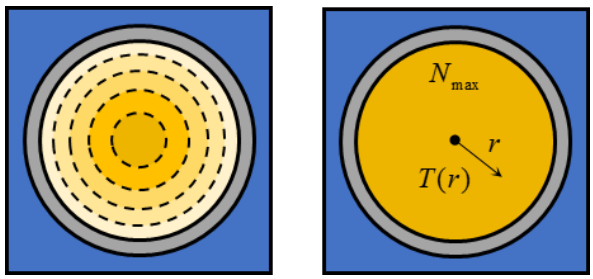


Figure 4.62 Illustration of the actual geometry (left) and the tracking geometry of the localized delta-tracking kernel (right).

To summarize, the procedure of the localized delta-tracking is given as follows:

1. Calculate Σ_t^{max} with Eq. (4.89) and sample DTC to the virtual collision with Eq. (4.86).
2. Calculate DTS to the zone boundary.
3. If DTS is smaller than DTC, move to the zone boundary.
4. Otherwise, retrieve actual number densities of the collided region and calculate the temperature of the collided spot using the temperature profile $T(r)$. Using the actual number densities and temperature, calculate the actual cross section Σ_t, and accept the virtual collision as the physical collision by the probability in Eq. (4.87).

4.8.2 Functionalization of Temperature Distributions

As mentioned, the localized delta-tracking scheme allows to analytically represent the temperature distributions in each zone. Thus, the T/H feedback module delivers the temperature distributions in a functionalized form. While there is no definite way of functionalizing the temperature distributions, it was empirically observed that fitting the discrete temperature points with a simple cubic polynomial is sufficient to express the temperature profile in the fuel, which is determined by the following four constraints:

1. Symmetry at the center: $T'(0)=0$.
2. Conservation of the center point: $T(0)=T_0$.
3. Conservation of the periphery point: $T(r_n)=T_n$.
4. Conservation of volume-averaged temperature: $\frac{1}{\pi r_n^2}\int_0^{r_n} 2\pi r T(r)dr=\bar{T}$.

Then, the following polynomial can be obtained:

$$T(r)=\frac{5T_n+5T_0-10\bar{T}}{r_n^3}r^3-\frac{4T_n+6T_0-10\bar{T}}{r_n^2}r^2+T_0 \quad (r\le r_n). \tag{4.90}$$

In case of gas gap, there are only two temperature points available, which gives a linear function:

$$T(r)=\frac{T_g-T_n}{r_{n+1}-r_n}(r-r_n)+T_n \quad (r_n\le r\le r_{n+1}). \tag{4.91}$$

In case of cladding, there are three temperature points – one at the center and two at the boundaries – and a quadratic polynomial is determined uniquely:

$$T(r)=ar^2+br+c \quad (r_{n+1}\le r\le r_{n+3}) \tag{4.92}$$

where the coefficients are as follows:

$$a=\frac{r_{n+1}(T_{n+2}-T_W)+T_g(r_{n+3}-r_{n+2})+r_{n+2}T_W-r_{n+3}T_{n+2}}{(r_{n+2}-r_{n+1})(r_{n+3}-r_{n+1})(r_{n+3}-r_{n+2})}, \tag{4.93}$$

$$b=\frac{T_W-T_g}{r_{n+3}-r_{n+1}}-a(r_{n+3}+r_{n+1}), \tag{4.94}$$

$$c=T_g-ar_{n+1}^2-br_{n+1}. \tag{4.95}$$

Note that all the notations are consistent with Figure 4.48.

Figure 4.63 shows the fitting result of the fuel temperature profiles of a fresh UO_2 pin using Eq. (4.90). Temperature profiles at several power levels were examined and it can be seen that the cubic polynomial can closely follow the numerical results.

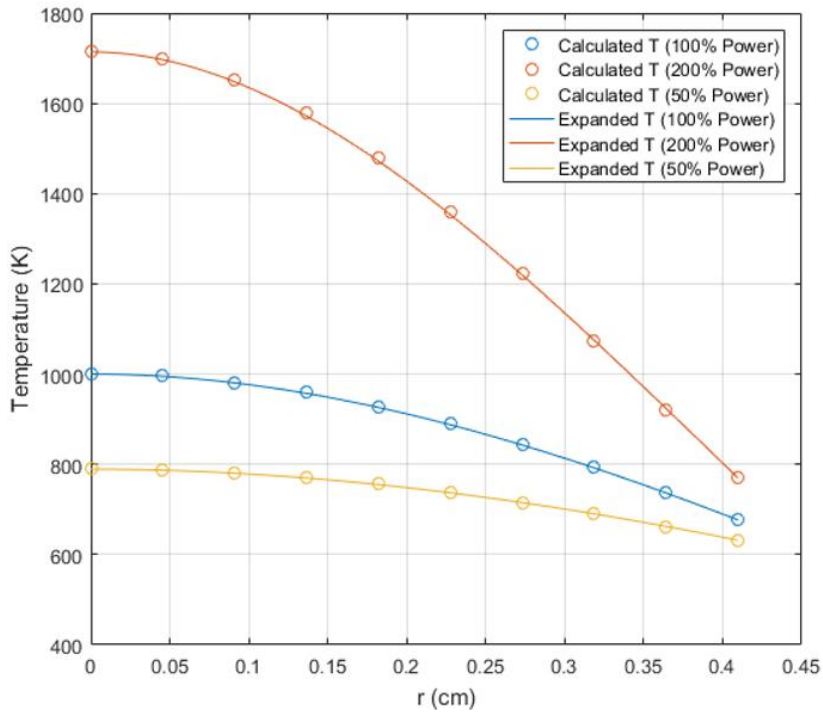


Figure 4.63 Comparison of calculated and functionalized fuel temperature profiles at various power levels.

4.8.3 Initial Verification

Treatment of Functionalized Temperature Distributions

The verification for treating the inter-fuel-rod temperature distributions was made with a 2.64% UO_2 pin having 17.5kW/m linear power rate. The coolant temperature was set to 600K and the fuel pellet was divided into 15 equi-volume regions and 100 meshes were used for the T/H fuel heat conduction solution. Three fuel temperature treatment options were compared: averaged fuel temperature (AFT), piecewise AFT (PAFT), and functionalized fuel temperature (FFT) paired with the localized delta-

tracking scheme. Figure 4.64 illustrates the fuel temperature profiles to be compared, from which it can be seen that the cubic polynomial follows the actual temperature distribution accurately.

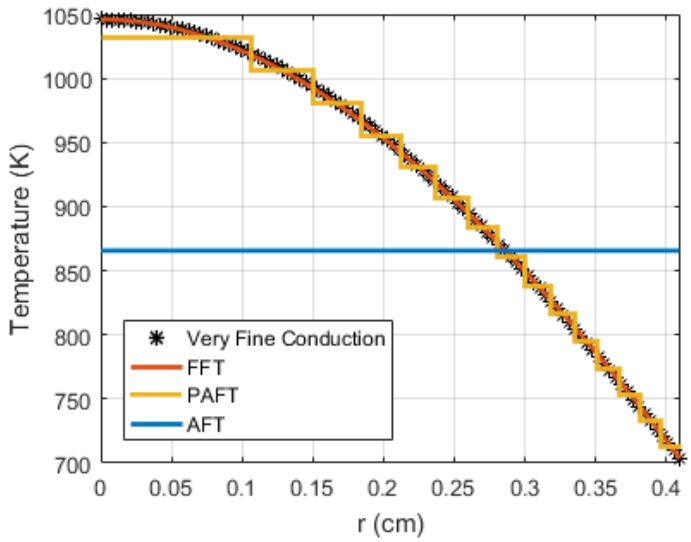


Figure 4.64 Comparison of different fuel temperature profiles.

Figure 4.65 and Figure 4.66 illustrate the intra-pellet macroscopic absorption and fission reaction rates calculated with different fuel temperature profiles, respectively, and Table 4.19 lists the eigenvalues. Since AFT gives higher peripheral temperature than PAFT, the absorption reaction rate at the periphery is overestimated by more than 1% due to the increased resonance self-shielding. For the fission reaction rate, however, not much difference was observed, as the thermal neutrons that majorly contribute to the fission reactions are not affected by the resonance self-shielding. These effects are reflected in the eigenvalues; due to the overestimated self-shielding absorption, AFT predicts a lower eigenvalue than PAFT. On the other hand, it can be seen that the results of PAFT and FFT show close agreement. It is verified that FFT can produce equivalent results to using a very fine discretization with significantly less computational burden. All these results show the importance of properly treating the intra-pellet temperature distributions and the strength of localized delta-tracking scheme.

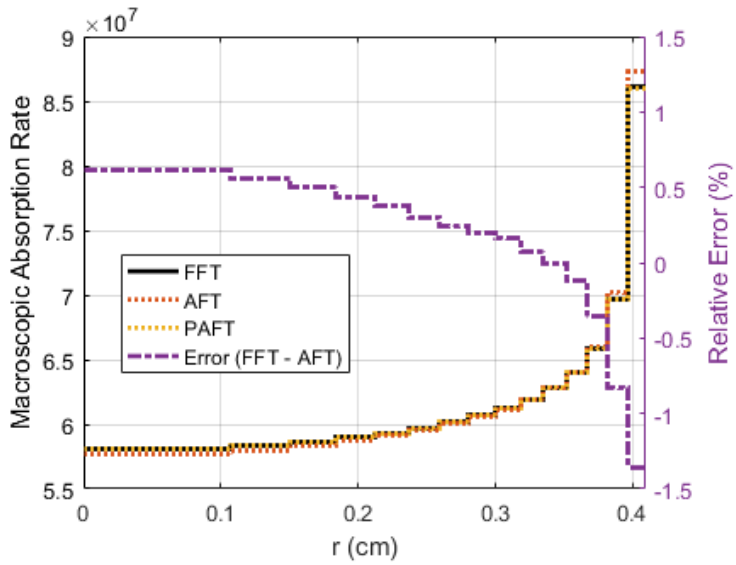


Figure 4.65 Comparison of region-wise absorption reaction rates of AFT, PAFT and FFT.

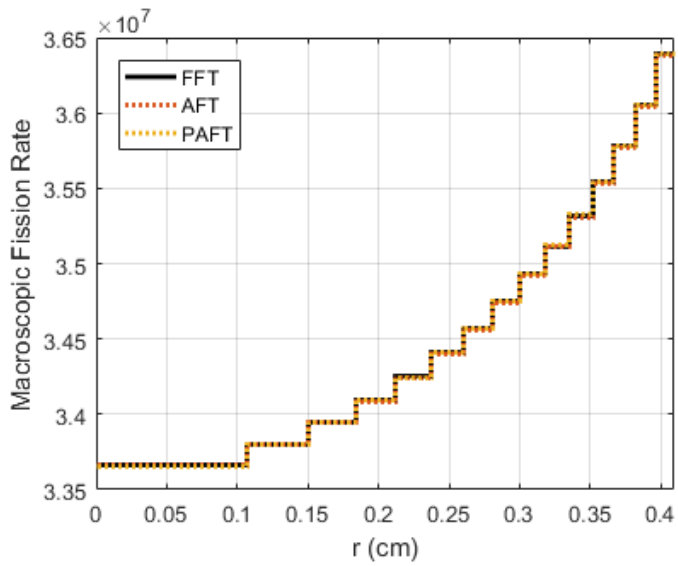


Figure 4.66 Comparison of region-wise fission reaction rates of AFT, PAFT, and FFT.

Table 4.19 Comparison of eigenvalues of AFT, PAFT, and FFT.

Case	AFT	PAFT	FFT
k_{eff}	1.28570 (2)	1.28604 (2)	1.28597 (2)

Treatment of Number Density Variations in Depletion

The validity of localized delta-tracking scheme under the variation of intra-pellet number densities was examined with a 1.72% UO_2 pin cell depletion problem. Figure 4.67 compares the letdown curves of the depletion calculations performed with the standard tracking scheme and the localized delta-tracking scheme. Five regions were used in the fuel pellet, and each step employed 10 inactive and 100 active cycles with 1M neutrons per cycle. The standard deviations of the eigenvalues are around 6 pcm at each step. In most steps the differences of the eigenvalues stay within 1σ range, which verifies that the localized delta-tracking scheme is consistent with the standard tracking scheme.

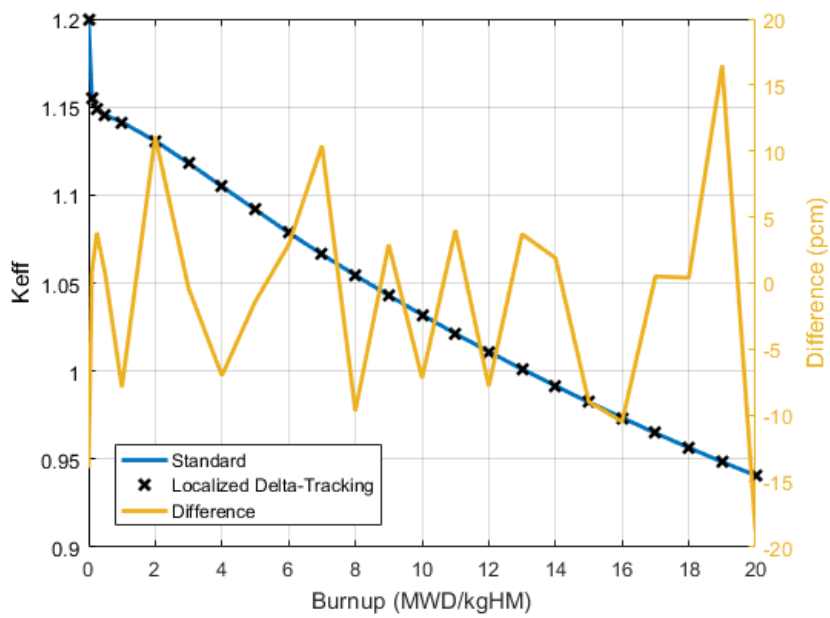


Figure 4.67 Comparison of pin cell depletion results of the standard tracking and the localized delta-tracking schemes.

However, the performance of localized delta-tracking scheme far outweighed the standard tracking scheme. Table 4.20 presents the total computing times of the two tracking schemes, in which a single GeForce RTX 2080 Ti GPU was used. Since the localized delta-tracking scheme neglects the intra-pellet meshes, the frequency of

surface crossing is much lower, which significantly reduces expensive macroscopic cross section updates on depleted fuels. As the result, the time for the macroscopic cross section calculation in the localized delta-tracking scheme was reduced to one-third of the standard tracking scheme.

Table 4.20 Comparison of pin cell depletion computing times (s) of the standard tracking and the localized delta-tracking schemes.

Scheme	Standard	Localized Delta-Tracking
Macroscopic	12,117	3,932
Tracing	2,031	791
Collision	474	396
Sorting	1,672	983
Total	16,294	6,102

4.9 Fission Source Convergence Acceleration

The characteristic iteration scheme of the MC method that the fission neutrons of a cycle becomes the source of the subsequent cycle induces inter-cycle correlations which is difficult to quantify. As the result, the apparent variance which is observed from a single run tends to underestimate the real variance, and quantifying the real variance from a single run has been a long research topic in this field.

In this regard, PRAGMA aims to deploy massive amount of particles at each cycle and reduce the number of cycles such that the intervention of inter-cycle correlations is minimized. However, the use of massive particles per cycle significantly increases the computing time of the inactive cycles, as the number of inactive cycles is only dependent to the characteristics of the problem, not to the number of particles used. Namely, although the number of active cycles can be reduced, the number of inactive cycles stays the same as far as the problem is unchanged.

Hence, accelerating the fission source convergence during the inactive cycles is crucial in performing massive particle simulations, and PRAGMA employs CMFD acceleration and ramp-up technique together to remarkably reduce the inactive cycle computing cost. This section introduces the two acceleration methods and examine their performances.

4.9.1 CMFD Acceleration

CMFD acceleration is one of the most successful numerical acceleration scheme for the deterministic calculations. Motivated by the remarkable performance of the CMFD acceleration in the deterministic calculations, Lee et al. [117] had suggested the use of CMFD acceleration in the MC method and has shown great success. By the CMFD acceleration, the number of inactive cycles required to reach convergence for a typical power reactor can be reduced from several hundreds to dozens.

Since the goal of CMFD acceleration in the MC calculation is merely to perform a global rebalance of neutrons, high numerical resolution is not necessary. Therefore, an assembly-wise one-group CMFD formulation is used in PRAGMA. In the CMFD calculation of PRAGMA, the diffusion coefficient is approximated by using the total cross section due to the difficulties in tallying transport cross sections:

$$D = \frac{1}{3\Sigma_t}. \tag{4.96}$$

In addition, the CMFD problem domain only includes the active core region, and the interface with the reflectors are considered as the vacuum boundary. By doing so, the possible hazard of divergence due to the existence of rare tally regions in the ex-core domain is prevented. Though these are not very accurate, equivalence is always retained by the correction factor $\hat{D}$.

For stabilizing the CMFD acceleration, the FIFO CMFD tally update scheme [118] is adopted, which is illustrated in Figure 4.68. At each cycle, a one-group tally packet is enqueued to the tally queue. The tally queue has an upper limit of its size, and if the queue is full, old packets are dequeued in FIFO manner. The group constants for the CMFD acceleration at each cycle are calculated by taking the average of the tally packets in the current queue.

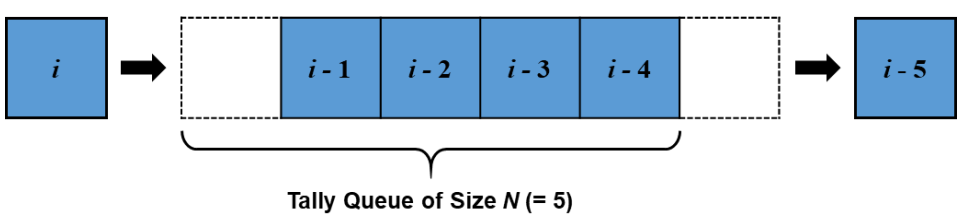


Figure 4.68 Schematic diagram of FIFO CMFD tally update scheme.

The FIFO CMFD tally update scheme can be interpreted as giving an inertia to the CMFD acceleration towards the global convergence direction by using the previous

cycle tallies. In addition, by setting the maximum queue size, the effect of previous cycles is gradually attenuated.

4.9.2 Ramp-up Technique

While the CMFD acceleration is a deterministic method, ramp-up technique is a pure stochastic acceleration method which has been used conventionally in the MC codes. Noting that the fission source distribution is inaccurate during the evolution phase and that the convergence is not affected by the number of particles, the ramp-up technique minimizes the waste of histories during the inactive cycles by gradually increasing the population instead of performing the calculation with the prescribed population from the very beginning.

Although the CMFD acceleration is applied and the number of inactive cycles is effectively reduced, still dozens of inactive cycles are required for the typical power reactors. As the number of active cycles is also reduced to dozens under the massive particle condition, the computing time portion of the inactive cycles is increased. In this regard, PRAGMA employs the CMFD acceleration and the ramp-up technique together to further drag down the computing cost of the inactive cycles.

The ramp-up technique is easy to implement. Once the ramp-up factor f, which is the ratio between the initial and the target populations, is determined, the weights of neutrons are adjusted at each cycle by the following factor:

$$W = \exp(\ln f / n) \tag{4.97}$$

where n is the number of inactive cycles. The population is then naturally increased exponentially by the artificially increased fission neutron yields.

The theoretical reduction ratio of the number of inactive cycle histories by ramp-up, denoted as R, can be estimated as follows:

$$R = 1 - \frac{W^n - 1}{nf\,(W - 1)}. \tag{4.98}$$

For a 20-times ramp-up with 25 inactive cycles, which is illustrated in Figure 4.69, the reduction is approximately 70%.

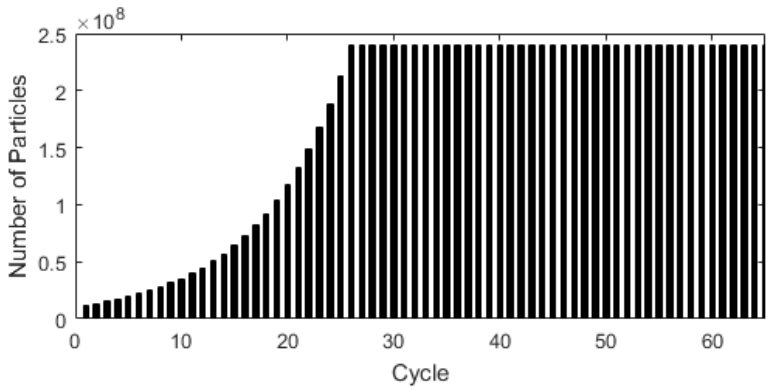


Figure 4.69 Example of population change by a 20-times ramp-up with 25 inactive cycles.

Technically, CMFD acceleration and ramp-up technique are fully independent and can be used together without any costs. However, CMFD acceleration can diverge when the number of tallies is not sufficient. Since the ramp-up technique involves a progressive increase of population during the inactive cycles, initial population may be too small for the CMFD calculation. However, PRAGMA employs hundreds of millions of particles for power reactor calculations and the population during ramp-up will be at least tens of millions, which is sufficient to stabilize the assembly-wise CMFD group constant tallies even for a full-core geometry.

4.10 Comprehensive Verification and Validation

This section presents the solution of whole-core problems using PRAGMA and demonstrates the practical whole-core massive particle simulations realized by the state-of-the-art GPU acceleration techniques. The core problems to be solved include APR1400 [68], VERA [90], and BEAVRS [91]. Unless specified, the calculations were performed with the point-wise cross section library based on ENDF/B-VII.1. The specification of the computing resources used for the calculations are given in Table 4.21. Note that the GPUs are consumer-grade GPUs used for gaming, and the computing cluster is a small-sized cluster which can be afforded by the industries.

Table 4.21 Specification of computing resources.

Number of Nodes	6
CPU / Node	2 × Intel Xeon E5-2630 v4
GPU / Node	4 × NVIDIA GeForce RTX 2080 Ti
Memory / Node	128GB DDR4
Interconnect	Mellanox Infiniband FDR
Compilers and Toolkits	GCC 7.3.0 CUDA 11.0 OpenMPI 4.0.4rc3

4.10.1 APR1400 Initial Core

The first problem to be presented is the Korean advanced power reactor APR1400 initial core model. The PRAGMA model for the APR1400 initial core is based on the nTRACER model [68] which was extended to the full-core configuration, and it consists of 3,650,832 tally cells and 10,578,812 material regions. The number of tally cells and material regions represent a typical complexity of a full-scale power reactor model which is directly related to the computational cost. Massive particle simulations deploying over 11 billion histories are carried out with the calculation conditions in Table 4.22.

Table 4.22 Calculation conditions for APR1400.

Number of Cycles	25 (Inactive), 40 (Active)
Number of Neutrons / Cycle	240,000,000 (Total 11.4 Billion)
Ramp-up Factor	20
Core Power	3983 MW
Core Flow Rate	20553.7 kg/s
Inlet Temperature	564.45 K
Pressure	15.514 MPa
Boron	1180 ppm (HZP), 1085 ppm (HFP)
Libraries (K)	550 / 600 / 700 / … / 1700 / 1800

First, base HZP and HFP calculations were performed to validate the power reactor analysis capability of PRAGMA. Table 4.23 shows the summary of the calculation results, and Figure 4.70 and Figure 4.71 illustrate the 3D power and the axial power distributions, respectively. Fuel and moderator temperature distributions of the HFP case is also presented in Figure 4.72.

It is observed that the T/H feedback effects are incorporated reasonably; physically expected phenomena such as flattening of the power distributions and skew of the axial power distribution are observed. Regarding the performance, less than 1% of RMS uncertainty in 3D power could be obtained within 15 minutes for an operational full-core power reactor problem, which is considered to be a milestone in the journey towards a feasible whole-core MC simulation. However, note that even these are not yet the optimal performance that can be achieved by PRAGMA.

Table 4.23 Summary of HZP and HFP calculation results for APR1400.

Condition	HZP	HFP
k_{eff}	1.00000 (1)	1.00006 (1)
Computing Time	12m 55s	14m 13s
RMS of Pin Power Uncertainty	0.168%	0.163%
Max. of Pin Power Uncertainty	0.424%	0.430%
RMS of 3D Power Uncertainty	0.872%	0.851%
Max. of 3D Power Uncertainty	11.636%	9.817%

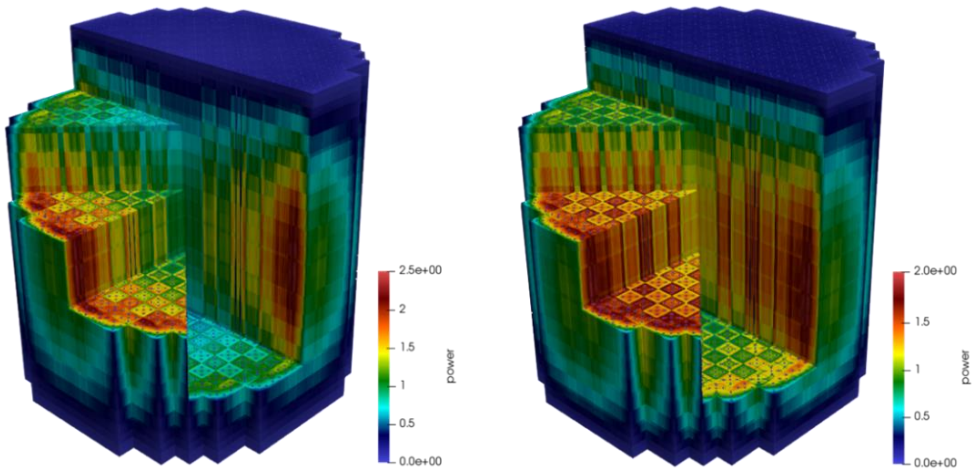


Figure 4.70 Illustration of HZP (left) and HFP (right) 3D power distributions of APR1400.

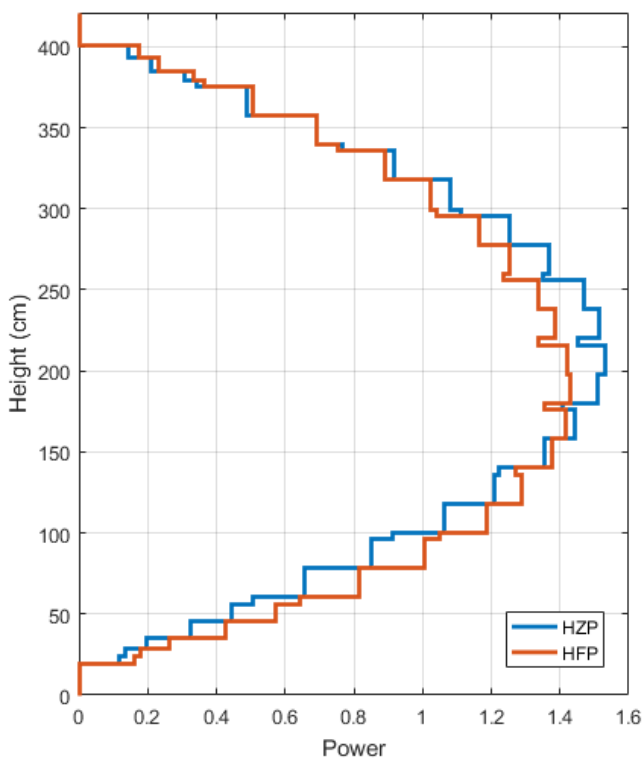


Figure 4.71 Comparison of HZP and HFP axial power distributions of APR1400.

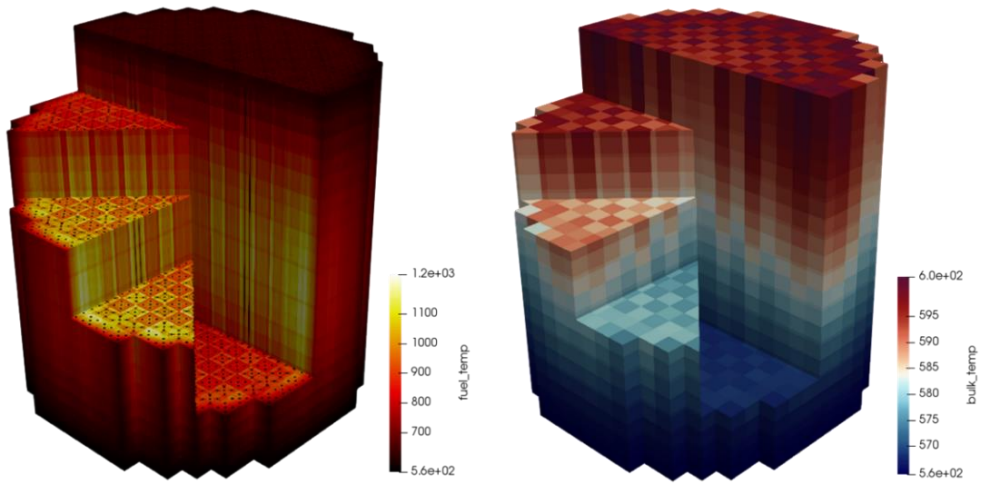


Figure 4.72 Illustration of average fuel temperature (left) and moderator temperature (right) distributions for the APR1400 HFP case.

From now on, the performance improvement by the advanced tracking schemes for whole-core analysis will be presented progressively. First, the effectiveness of domain decomposition scheme is demonstrated. Table 4.24 compares the computing time of HFP calculations with and without the domain decomposition scheme.

It can be seen that there is a significant reduction in the tracking and the T/H time by the domain decomposition, and they come from different reasons. The reduction in the tracking time comes from the increased spatial locality of data accesses, which results in a better utilization of cache memories. The domain decomposition scheme reduces the size of the geometry that each GPU should handle, which increases the chance of threads accessing nearby geometric data in each GPU.

Table 4.24 Comparison of computing times of the cases with and without the domain decomposition scheme for the APR1400 HFP case.

Domain Decomposition	Off	On
Tracking Time (s)	781.1	572.5
Comm. Time (s)	-	71.0
CMFD Time (s)	9.8	9.0
T/H Time (s)	42.8	13.4
Total Time	14m 13s	11m 21s

The reduction in the T/H time due to the removal of collective data communication. The *MPI_Scatter* operations which are required in the ordinary calculation to update the core-wide temperature and mass density information is not needed under domain decomposition, since the T/H domain and the tracking domain are consistent. Same is true for processing tallies. In the ordinary calculation, there must be a process that collects partially accumulated tallies from other processes by *MPI_Reduce*. It is not significant for the time being because only 3D power is being tallied, but when the amount of tallies increase with depletion, this advantage will become substantial.

Figure 4.73 illustrates the load balance of the domain decomposed case. Domain 7 turned out to take about 9.3% higher load than the average, while the load of other domains are very neatly balanced. It is because domain 7 takes an additional center assembly as can be seen from the domain map illustrated in Figure 4.74. Since the number of fuel assemblies of APR1400 is 241 which cannot be divided with 24, one process must take an additional assembly. Since the center assembly has the same distance with all the centroids in the wheel clustering scheme, it can belong to any domains and thus it becomes the additional assembly. As the result, the computing time is bound to domain 7.

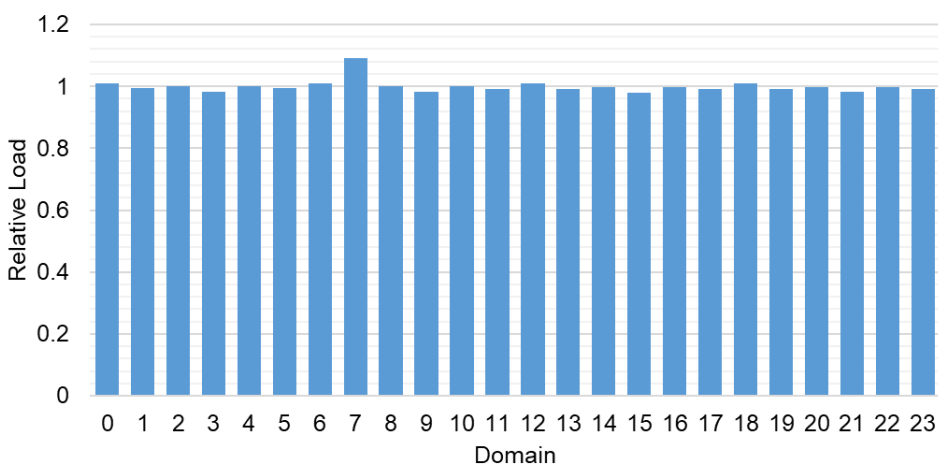


Figure 4.73 Load balance of the APR1400 HFP case.

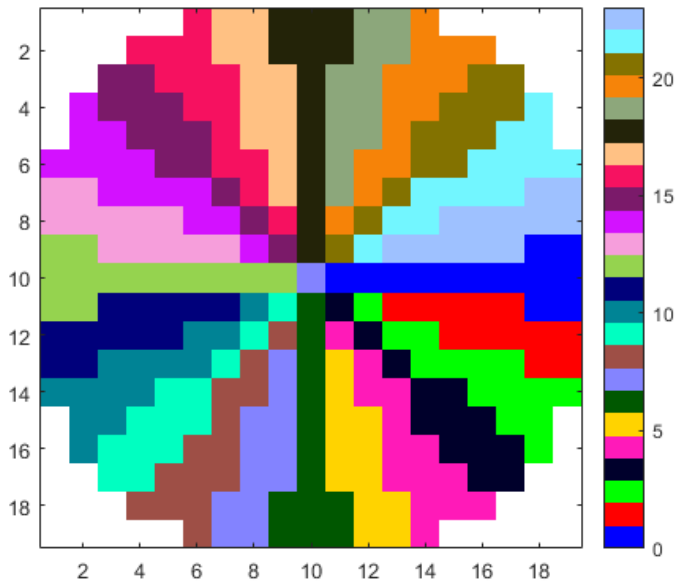


Figure 4.74 Domain map for the APR1400 calculation generated by the wheel clustering scheme.

Next, the effectiveness of localized delta-tracking scheme in the whole-core HFP calculation is demonstrated. Table 4.25 presents the eigenvalues and the computing time of AFT, PAFT with 5 subrings in the pellets, and FFT schemes, and Figure 4.75 illustrates the Shannon entropy behaviors of the schemes. The Shannon entropy is an indirect measure of the source distribution. Higher Shannon entropy indicates that the source distribution is more uniform.

It can be seen that while FFT is producing equivalent results with PAFT in both the eigenvalue and source distribution, the computing time is the shortest, which is even shorter than AFT. The reason why FFT is faster than AFT is the decrease of the cross section update in the localized delta-tracking scheme. Recall from Figure 4.26 that the mean-free-path of neutrons is longer than the dimension of pin cells in typical LWRs except for thermal energies, and as the result, surface crossing events are more frequent than the collision events. While the standard tracking scheme calculates the four types of principal cross sections at every migration, the localized delta-tracking scheme only calculates the majorant total cross section and the other cross sections are calculated only when a collision is sampled. Since the collision events are not as

frequent as the surface crossing events, the number of cross section updates differs between the standard tracking scheme and the localized delta-tracking scheme.

Table 4.25 Comparison of computing times and eigenvalues of AFT, PAFT, and FFT schemes for the APR1400 HFP case.

Case	AFT	PAFT (5 Rings)	FFT
k_{eff}	1.00004 (1)	1.00044 (1)	1.00045 (1)
Computing Time	11m 21s	16m 17s	9m 32s

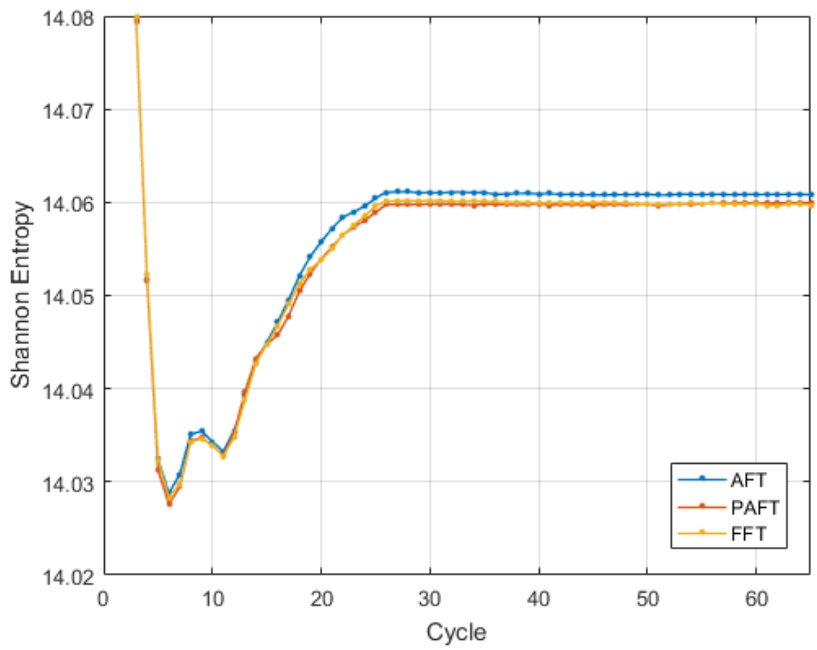


Figure 4.75 Shannon entropy trends of AFT, PAFT and FFT schemes for the APR1400 HFP case.

Figure 4.76 illustrates the axial power distribution of AFT and FFT. It can be seen that the axial power of AFT has lower power peaking than FFT, which is due to the overestimation of self-shielding absorption. It is also consistent with the observation through the Shannon entropy that the source distribution is more uniform in AFT. This result demonstrates that an imprecise treatment of the intra-pellet temperature distributions can produce up to 1% of relative error in the axial power.

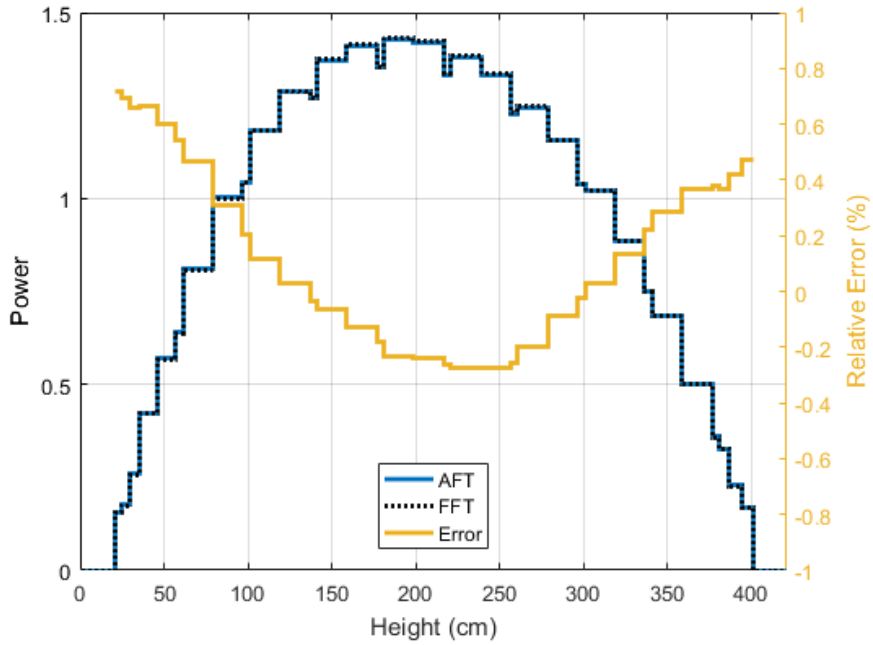


Figure 4.76 Comparison of axial power distributions of AFT and FFT schemes for the APR1400 HFP case.

However, the localized delta-tracking scheme shows higher uncertainty than the standard tracking scheme due to the preclusion of track-length estimators. Therefore, real variances of power were obtained for the AFT and FFT cases by executing each of the case by 20 times, and the figure of merits (FOM) defined as Eq. (4.99) were examined. The computing times were taken from Table 4.25.

$$\mathrm{FOM} = \frac{1}{\sigma^2 T}. \tag{4.99}$$

Table 4.26 shows the real standard deviations and FOMs of each case, and Figure 4.77 illustrates the distributions of the pin power standard deviations. It was assumed that the PAFT case will have similar uncertainties with the AFT case. As expected, the FFT case clearly shows higher uncertainty and therefore has lower FOM than the AFT case. However, it should be noted that AFT is an incorrect scheme as it uses the pellet-averaged fuel temperature, and that FFT shows higher FOM than PAFT. If the

core goes depleted or more rings are used in each fuel pellet, PAFT will become even slower, and the effectiveness of FFT will be increased.

Table 4.26 Comparison of real standard deviations and FOMs of AFT, PAFT, and FFT schemes for the APR1400 HFP case.

Case	AFT / PAFT	FFT
RMS of Pin Power Uncertainty	0.206%	0.246%
RMS of 3D Power Uncertainty	0.887%	1.117%
Pin Power FOM	346.03 / 241.20	288.89
3D Power FOM	18.66 / 13.01	14.01

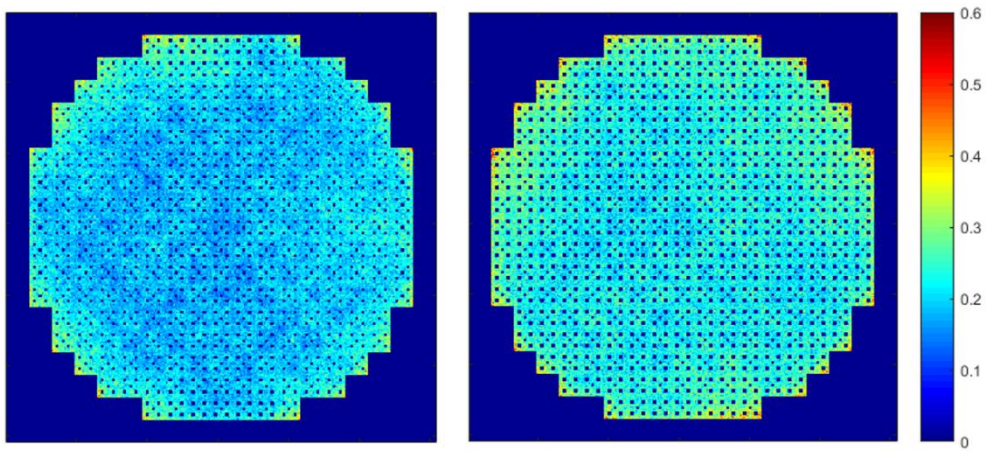


Figure 4.77 Pin power real standard deviation (%) of AFT (left) and FFT (right) for the APR1400 HFP case.

The statistical impact of ramp-up acceleration was also examined. With ramp-up, the neutrons will have more correlated distributions because the fission neutrons tend to be produced locally due to the increased weights. Since the ramp-up is performed drastically within tens of cycles, there might not be sufficient cycles to 'disperse' the clustered neutrons. As the result, the power tallies will have higher variance.

Table 4.27 shows the real standard deviations and the FOMs of FFT scheme with and without ramp-up; each case was run 20 times. As expected, the clustering effect of ramp-up is apparent in the assembly power tallies, which gives higher uncertainty

of assembly power tallies in the ramp-up case. However, it turned out that the effect diminishes in the shorter-range tallies; the uncertainty increase in the pin power tally is marginal, and almost no difference is seen in the 3D power tally uncertainties. As the result, using ramp-up becomes more beneficial.

Table 4.27 Comparison of real standard deviations and FOMs of FFT and FFT with ramp-up for the APR1400 HFP case.

Case	FFT	FFT with Ramp-up
Total Computing Time	12m 42s	9m 32s
Inactive Cycle Computing Time	5m 13s	1m 57s
RMS of Asy. Power Uncertainty	0.102%	0.138%
RMS of Pin Power Uncertainty	0.228%	0.246%
RMS of 3D Power Uncertainty	1.110%	1.117%
Asy. Power FOM	1261.4	918.0
Pin Power FOM	252.4	288.9
3D Power FOM	10.65	14.01

Figure 4.78 illustrates the effect of acceleration schemes. Compared to the standard MC calculation, using CMFD and ramp-up schemes can reduce 96% of the histories in the inactive cycles.

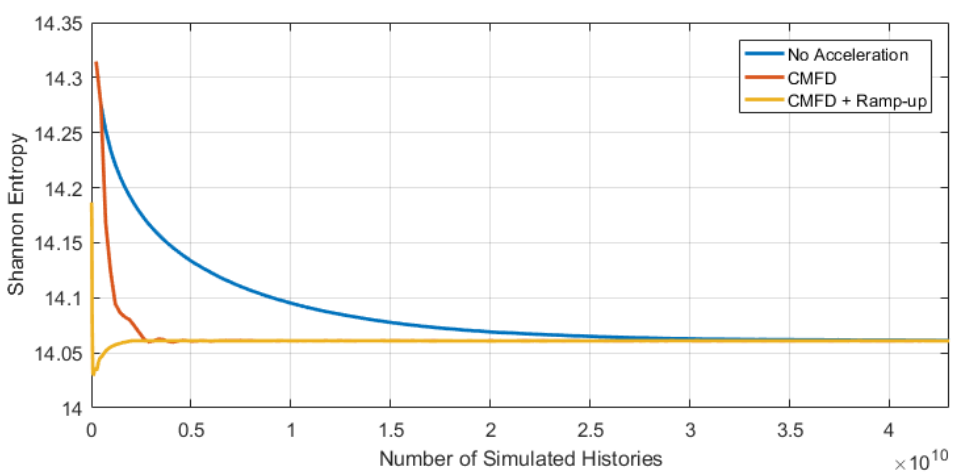


Figure 4.78 Shannon entropy versus the number of simulated histories of different acceleration schemes for the APR1400 HFP case.

Lastly, the CBC search and xenon equilibrium capabilities were examined. Table 4.28 presents the CBCs at different conditions using different TVS schemes: CXS and RST. The reference CBC values are from the Design Control Documents (DCD) of APR1400 [119] released by the U.S. Nuclear Regulatory Commission (NRC). In fact, the CBCs in DCD were calculated with low-order solution codes, namely the DIT/ROCS code system [120], so they are not the true reference. However, since the measurements are not available and as the model was designed based on the DCDs, it is reasonable to compare with the design values obtained with commercial codes. It can be seen that the CBCs calculated by PRAGMA match closely with the design values. For the HFP case, there exists more than 12 ppm difference in CBC between CXS and RST, which demonstrates the importance of using a proper TVS scheme. It is also verified that the xenon equilibrium capability is sound; the estimated xenon worth of PRAGMA is close to the design value.

Table 4.28 Comparison of CBCs under various conditions of APR1400.

Condition	DCD	CXS	RST
HZP	1187	1179.9 (0.1)	1176.2 (0.1)
HFP	1067	1089.0 (0.1)	1076.8 (0.1)
HFP with Eq. Xe	817	826.3 (0.1)	814.0 (0.1)

4.10.2 VERA Benchmark Problem 5

VERA Core Physics Benchmark Progression Problem [90] is a benchmark suite based on the Watts Bar Nuclear Plant Unit 1 (WBN1) in the United States, and this benchmark suite has been released by CASL. The benchmark suite contains total 10 problems, and here the problem 5 will be solved by PRAGMA, which describes the Zero Power Physics Test (ZPPT) results of WBN1. The core geometries are shown in Figure 4.79 and Figure 4.80. Following four cases in problem 5 will be examined, and the detailed descriptions for the cases can be found in the manual:

1. Comparison of control rod bank worths (CRBW).
2. Comparison of Bank D differential and integral worths.
3. Comparison of eigenvalues at critical rod conditions.
4. Comparison of power distributions.

In this problem, continuous-energy MC solutions of KENO-VI [121] are provided as the reference solutions. Hence, a rigorous code-to-code comparison of whole-core calculation results with a well-established production code can be made. In this case, PRAGMA employed the cross section library based on ENDF/B-VII.0 to retain the consistency with the KENO-VI calculation conditions.

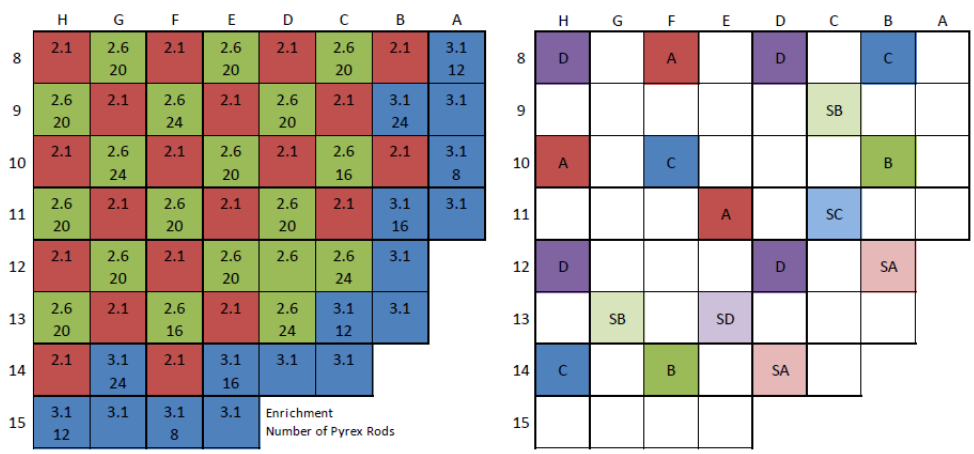


Figure 4.79 VERA problem 5 assembly, poison, and control rod layout.

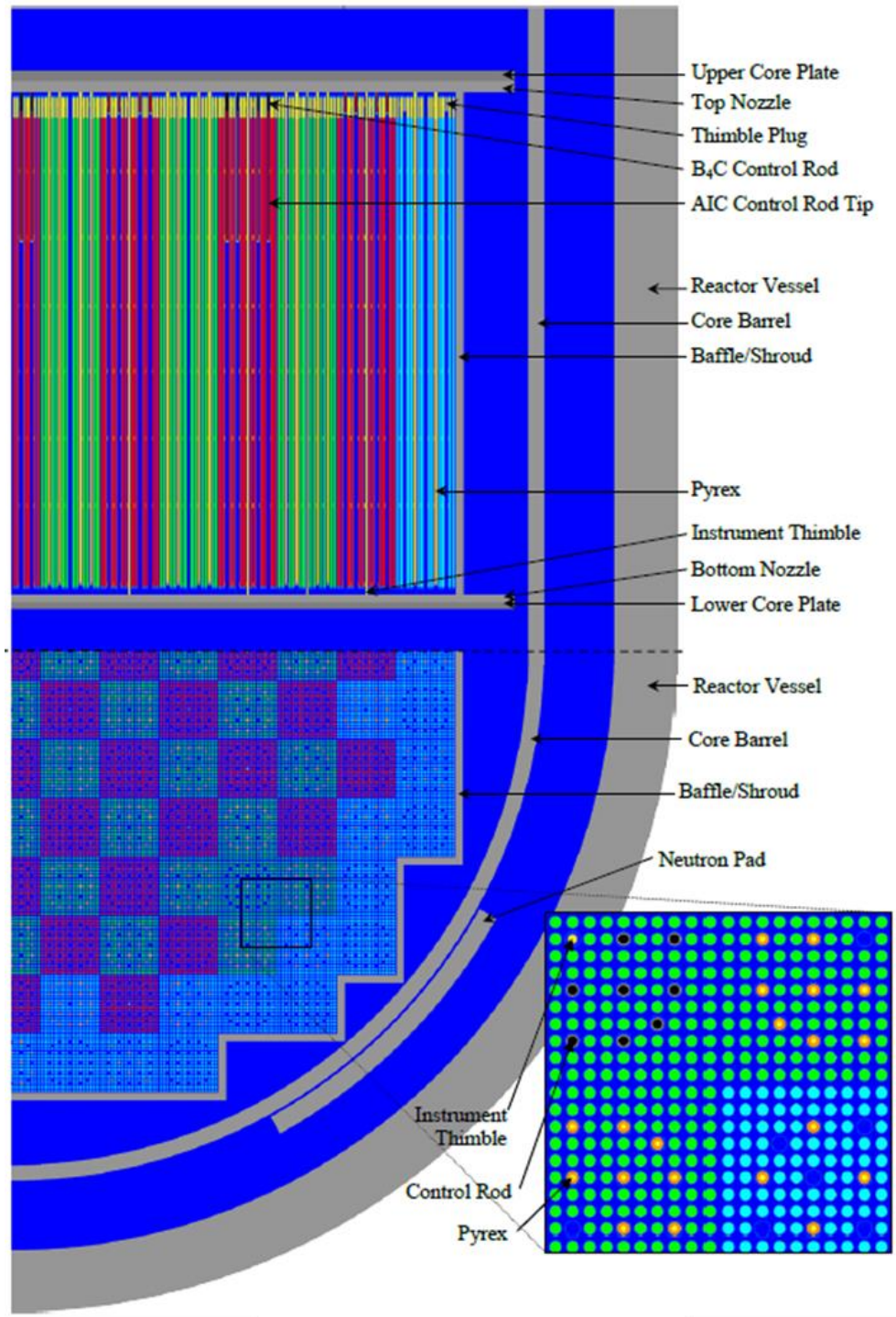


Figure 4.80 VERA problem 5 detailed geometry.

Note that PRAGMA explicitly modelled the instrument tube thimbles irregularly located as illustrated in Figure 4.81, which requires the full-core representation. On the other hand, the KENO-VI reference solutions were obtained by using the bottom right quadrant pattern; namely, the quarter symmetry was assumed. Still, the impact of the instrument tube thimbles to the reactivity is negligible.

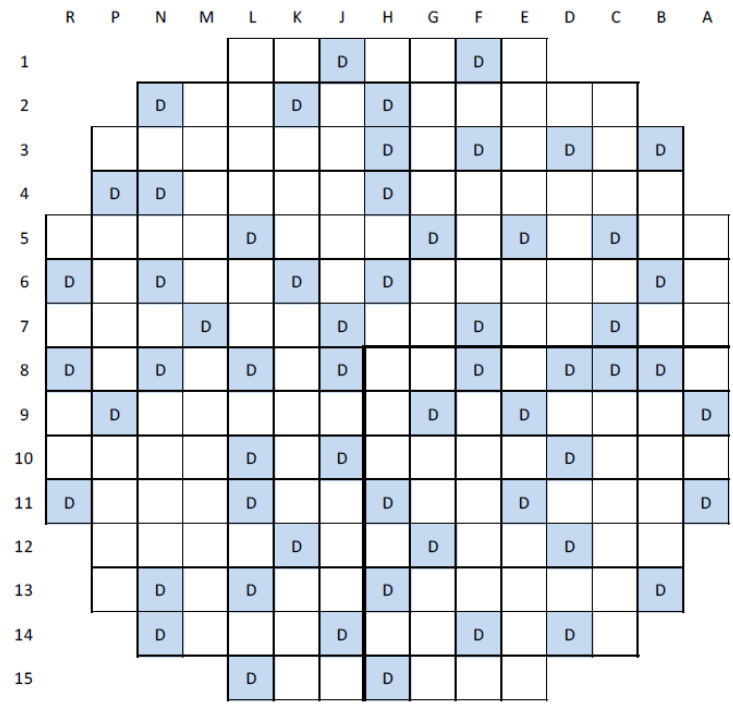


Figure 4.81 VERA problem 5 in-core instrumentation locations.

The temperature mixing of hydrogen S(α, β) tables was turned off and only 550K table was used in order to be consistent with KENO-VI. Accordingly, the eigenvalues of PRAGMA were adjusted by -44 pcm as was done for the KENO-VI results. For the TVS scheme, RST was not used for U-238 as the KENO-VI results did not apply DBRC. This applies to all the cases that will be presented below.

First, the comparison of CRBWs were made. Table 4.29 shows the eigenvalues of each bank insertion cases, and Table 4.30 presents the comparison of the CRBWs. PRAGMA is having a very close agreement in the eigenvalues and the CRBWs with KENO-VI. Considering that KENO-VI and PRAGMA are individual code systems

having countless differences in libraries and parameters, the results are unexpectedly well-matching.

Table 4.29 Comparison of the eigenvalues for CRBW examination of VERA problem 5.

Bank	KENO-VI	PRAGMA	Difference (pcm)
ARO	1.01284 (1)	1.01284 (1)	0
A	1.00372 (1)	1.00374 (1)	2
B	1.00394 (1)	1.00392 (1)	-2
C	1.00284 (1)	1.00286 (1)	2
D	0.99882 (1)	0.99883 (1)	1
SA	1.00828 (1)	1.00827 (1)	-1
SB	1.00202 (1)	1.00203 (1)	1
SC	1.00775 (1)	1.00775 (1)	0
SD	1.00775 (1)	1.00774 (1)	-1

Table 4.30 Comparison of CRBWs (pcm) of VERA problem 5.

Bank	Measured	KENO-VI	PRAGMA	Difference (Measured / KENO-VI)
A	843	898 (2)	895 (2)	52 / -3
B	879	875 (2)	877 (2)	-2 / 2
C	951	984 (2)	983 (2)	32 / -1
D	1342	1386 (2)	1385 (2)	43 / -1
SA	435	447 (2)	448 (2)	13 / 1
SB	1056	1066 (2)	1065 (2)	9 / -1
SC	480	499 (2)	499 (2)	19 / 0
SD	480	499 (2)	500 (2)	20 / 1
Total	6467	6654 (16)	6652 (16)	187 / -2

Second, the eigenvalues at the critical rod conditions were compared. Table 4.31 presents the eigenvalues at several critical rod conditions. Similarly to the previous result, the differences between the eigenvalues of PRAGMA and KENO-VI are very small.

Table 4.31 Comparison of eigenvalues for critical tests of VERA problem 5.

Test Bank	Bank D Step	KENO-VI	PRAGMA	Difference (pcm)
A	97	0.99880 (1)	0.99878 (1)	-2
B	113	0.99936 (1)	0.99933 (1)	-3
C	119	0.99904 (1)	0.99904 (1)	0
D	18	0.99908 (1)	0.99911 (1)	3
SA	69	0.99902 (1)	0.99891 (1)	-11
SB	134	0.99932 (1)	0.99935 (1)	3
SC	71	0.99898 (1)	0.99893 (1)	-5
SD	71	0.99898 (1)	0.99893 (1)	-5

As an extension of this test, control rod searches were performed to determine the actual critical Bank D position of each case. Table 4.32 shows the search results and the results of verification calculations using the searched positions, and Figure 4.82 illustrates an example of the control rod search history of the Test Bank C case. In the example, the number of skipped cycles is 15 and the size of search batch is 5, and the Bank D starts from fully inserted state. Note that the reference eigenvalues are the ones before adjusting -44 pcm. It can be seen that the critical rod positions can be properly found using the control rod search capability in PRAGMA.

Table 4.32 Bank D search results for critical tests of VERA Problem 5.

Test Bank	Reference Eigenvalue	Reference Bank D Step	Searched Bank D Step	Verification Run
A	0.99922	97	106.7 (107)	1.00006 (1)
B	0.99977	113	116.0 (116)	1.00003 (1)
C	0.99948	119	126.2 (126)	1.00000 (1)
D	0.99955	18	30.6 (31)	1.00001 (1)
SA	0.99935	69	73.9 (74)	1.00003 (1)
SB	0.99979	134	137.7 (138)	1.00005 (1)
SC	0.99937	71	76.1 (76)	1.00003 (1)
SD	0.99937	71	76.3 (76)	1.00002 (1)

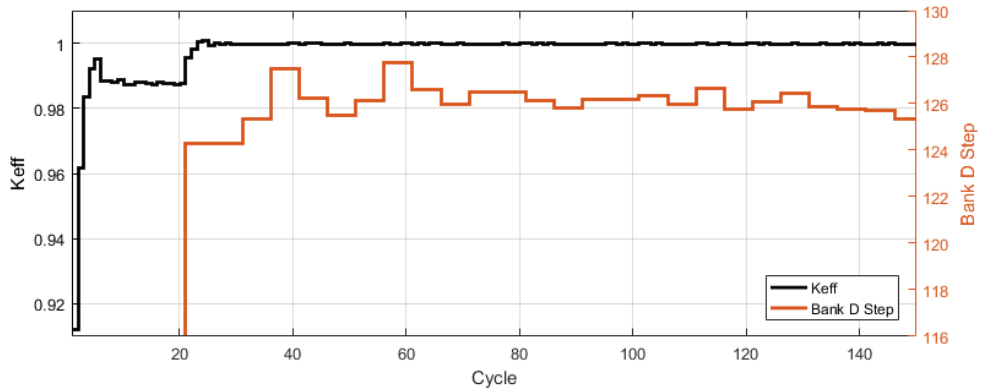


Figure 4.82 Control rod search history of Test Bank C.

Third, Bank D differential and integral rod worths were calculated and compared. Table 4.33 presents the eigenvalues at several Bank D positions, and Figure 4.83 and Figure 4.84 illustrate the differential worths and the integral worth curve of Bank D, respectively. Both the differential worths and the integral worth curves coincide; the error of the differential worths does not exceed 4 pcm, and the integral worth curves of KENO-VI and PRAGMA are almost overlapped. The maximum difference in the integral worth curve of Bank D is around 3 pcm.

Table 4.33 Comparison of eigenvalues for Bank D differential worth examination of VERA problem 5.

Bank D Step	KENO-VI	PRAGMA	Difference (pcm)
0	0.99276 (1)	0.99275 (1)	-1
23	0.99316 (1)	0.99314 (1)	-2
46	0.99456 (1)	0.99454 (1)	-2
69	0.99737 (1)	0.99734 (1)	-3
92	1.00028 (1)	1.00024 (1)	-4
115	1.00254 (1)	1.00253 (1)	-1
138	1.00416 (1)	1.00414 (1)	-2
161	1.00530 (1)	1.00530 (1)	0
184	1.00607 (1)	1.00606 (1)	-1
207	1.00647 (1)	1.00648 (1)	1
230	1.00658 (1)	1.00656 (1)	-2

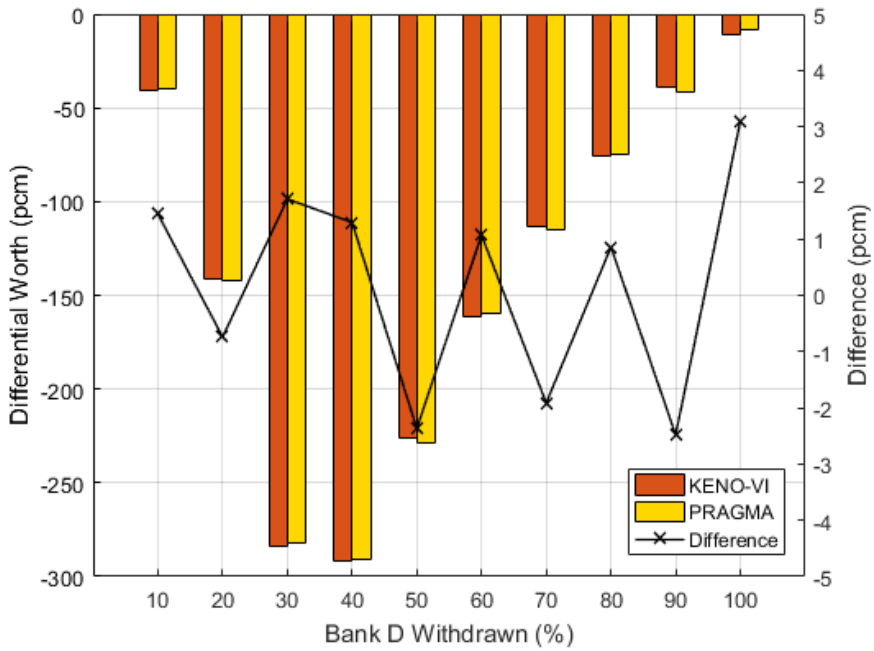


Figure 4.83 Comparison of Bank D differential worths for VERA problem 5.

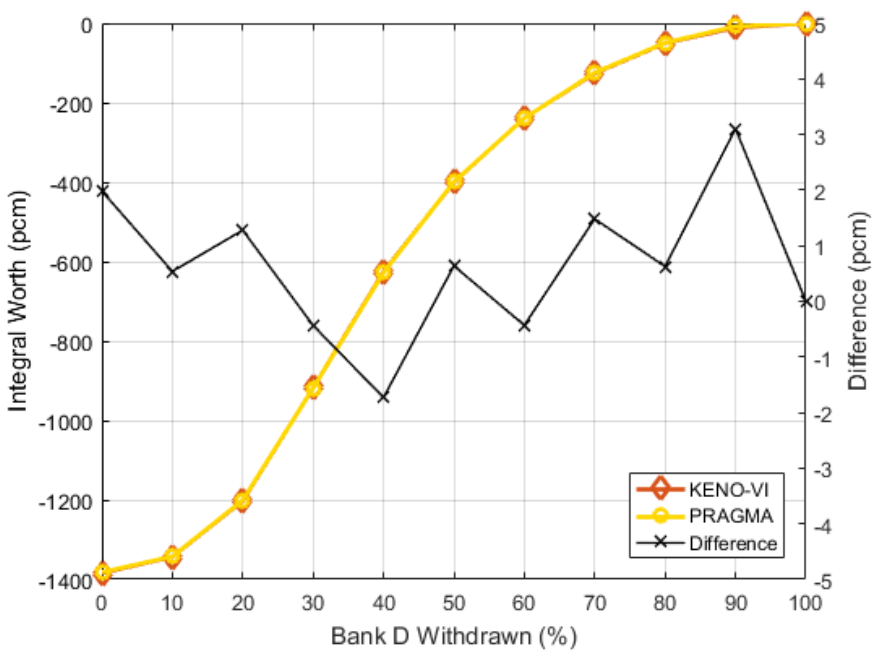


Figure 4.84 Comparison of Bank D integral worth curves for VERA problem 5.

Finally, the comparison of power distributions and performance were made. The reference model for the power tally consists of 5,659,397 tally cells and 19,521,059 material regions. The instrument tube thimbles were removed in accordance to the KENO-VI model. The KENO-VI model consists of 49 fuel planes, but two planes were added in the PRAGMA model to take into account the misaligned tips of the PYREX rods and the fuel rods.

This case is a computationally intense calculation deploying ~ 100 billion particles, and therefore it is appropriate for the demonstration of the PRAGMA's capability to perform massive particle whole-core calculations at substantially higher efficiency than the conventional MC codes. Table 4.34 presents the calculation conditions and the summary of KENO-VI and PRAGMA results. While PRAGMA takes only two hours for the completion of a 100 billion history simulation, KENO-VI had taken 29 days for the same calculation. Assuming that the performance of KENO-VI is fully scalable with the number of CPU cores used, it requires ~ 62,000 CPU cores to have equivalent performance with PRAGMA employing 24 GPUs, which is a significant speedup demonstrating the practicality of PRAGMA.

In case of the 3D power (pin power, according to the expression used by the manual) uncertainties, PRAGMA shows higher values than KENO-VI. However, it should be noted that PRAGMA solved a full core model while KENO-VI employed the quarter symmetry. Dividing the average uncertainty of PRAGMA by two (square root of the ratio of the geometry size) gives equivalence. In addition, the maximum uncertainty occurs at the thin planes which were added in the PRAGMA model.

Figure 4.85 and Figure 4.86 demonstrate the difference in the radial and the axial power distributions between KENO-VI and PRAGMA, respectively. The difference of the assembly-wise power distributions and the axial power distributions are only 0.071% and 0.037% in terms of RMS, respectively. However, there exists a clear tilt in the radial power errors although the magnitude is small, for which investigations will be required.

Table 4.34 Comparison of whole-core calculation conditions and results of KENO-VI and PRAGMA for VERA problem 5.

Code	KENO-VI	PRAGMA
Total Number of Particles	1.0000E+11	1.0154E+11
Number of Particles / Generation	1.0E+07	1.0E+08
Number of Generations	10,000	1,050
Number of Skipped Generations	500	50
Ramp-up Factor	-	20
Core Symmetry	Quarter	Full
Processors	180 CPU Cores	24 GPUs
Runtime	29 Days	2.08 Hours
k_{eff}	1.000072 (0.2)	1.000127 (0.2)
Average 3D Power Uncertainty	0.209%	0.425%
Maximum 3D Power Uncertainty (by Power)	1.630% (< 1.0) 0.414% (> 1.0)	8.442% (< 1.0) 0.635% (> 1.0)

0.9487 0.9486 **-0.01%**						
0.9193 0.9189 **-0.04%**	0.9973 0.9967 **-0.06%**					
1.0181 1.0175 **-0.06%**	0.9083 0.9074 **-0.10%**	1.0648 1.0636 **-0.11%**				
0.9850 0.9844 **-0.06%**	1.0819 1.0810 **-0.08%**	1.0412 1.0401 **-0.10%**	1.1615 1.1604 **-0.09%**			
1.0647 1.0645 **-0.02%**	1.0471 1.0466 **-0.05%**	1.1746 1.1739 **-0.06%**	1.0850 1.0843 **-0.06%**	1.2368 1.2363 **-0.04%**		
1.0480 1.0483 **0.03%**	1.1619 1.1621 **0.02%**	1.1520 1.1519 **-0.01%**	1.1508 1.1508 **0.00%**	0.8969 0.8969 **0.00%**	0.9126 0.9128 **0.02%**	
1.0841 1.0853 **0.11%**	1.0652 1.0662 **0.09%**	1.1039 1.1047 **0.07%**	1.0496 1.0501 **0.04%**	0.9452 0.9456 **0.04%**	0.6296 0.6299 **0.04%**	
0.7931 0.7945 **0.17%**	0.9071 0.9085 **0.15%**	0.8046 0.8054 **0.10%**	0.6590 0.6595 **0.07%**			KENO-VI PRAGMA **Rel. Err.**

Figure 4.85 Comparison of assembly-wise power distributions of KENO-VI and PRAGMA (octant folded) for VERA problem 5.

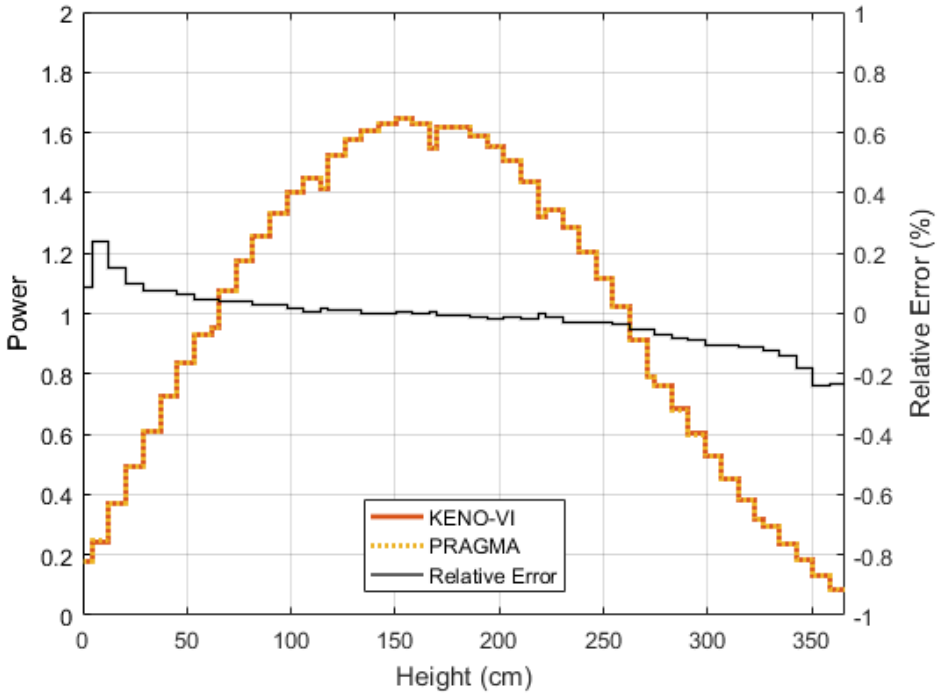


Figure 4.86 Comparison of axial power distributions of KENO-VI and PRAGMA for VERA problem 5.

To quantify how economical the PRAGMA calculation is, a rough comparison was made with the metric called performance per price (PPP), as presented in Table 4.35. The performance of an MC code is represented by the tracking rate, so the tracking rate per price (TRPP) can be considered as an equivalent metric to PPP. The result of MCS [122], which is a newer and more dedicated MC code designed for whole-core calculations, was included for the comparison. However, the paper does not provide the result of MCS for the 100 billion particle calculation, so a proportional adjustment was made.

Although the PRAGMA calculation has the highest processor cost, the computing time is significantly reduced compared to other MC codes, which leads to the highest TRPP; the TRPP of PRAGMA is 240 times higher than that of KENO-VI, and still 24 times higher than that of MCS. This comparison clearly shows how cost-effective PRAGMA is compared to the conventional CPU-based MC codes.

Table 4.35 Comparison of performance per price for different MC codes.

Code	KENO-VI	MCS	PRAGMA
Processors	180 CPU Cores (Opteron 6136)	180 CPU Cores (Xeon E5-2620 v4)	24 GPUs (RTX 2080 Ti)
Processor MSRP	$ 744 / Socket	$ 422 / Socket	$ 999 / GPU
Processor Total Cost	$ 17,112	$ 9,706	$ 23,976
Runtime	29 Days	5 Days	2.08 Hours
Tracking Rate (/s)	39,911	231,481	13,560,363
Tracking Rate per Price (/$·s)	2.33	23.85	565.58

4.10.3 BEAVRS Benchmark

BEAVRS (Benchmark for Evaluation And Validation of Reactor Simulations) [91] benchmark is a realistic two-cycle whole-core benchmark problem developed by the Computational Reactor Physics Group (CRPG) of MIT. The benchmark replicates a real Westinghouse 4-loop reactor operating in full detail and the operational data are provided. Therefore, the BEAVRS benchmark problem is appropriate for examining the validity of PRAGMA for actual applications.

The PRAGMA model for BEAVRS was made based on Version 3.0.1. Figure 4.87 and Figure 4.88 illustrate the axial and the radial configuration of the BEAVRS core, respectively. Two approximations were made from the exact geometry: 1. five planes between 'Bottom of Active Fuel' and 'Bottom of Active Absorber' were merged into two planes by extending Grid 1 upwards by 0.038cm and downwards by 0.4141cm, and 2. the grid sleeve of each assembly was smeared with the grids of the pin cells surrounding the assembly. It is expected that both approximations will have marginal impacts, but the impacts were not rigorously examined.

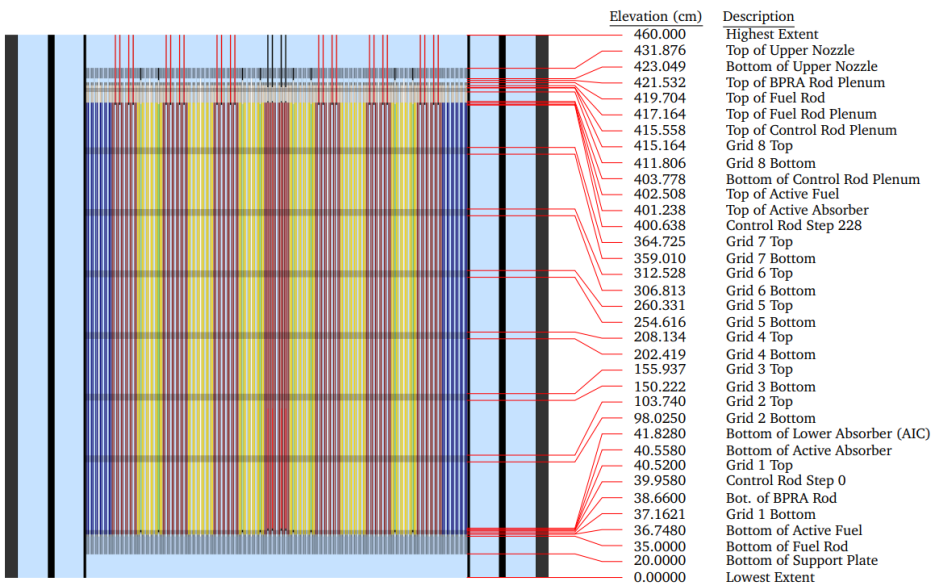


Figure 4.87 Axial configuration of the BEAVRS core.

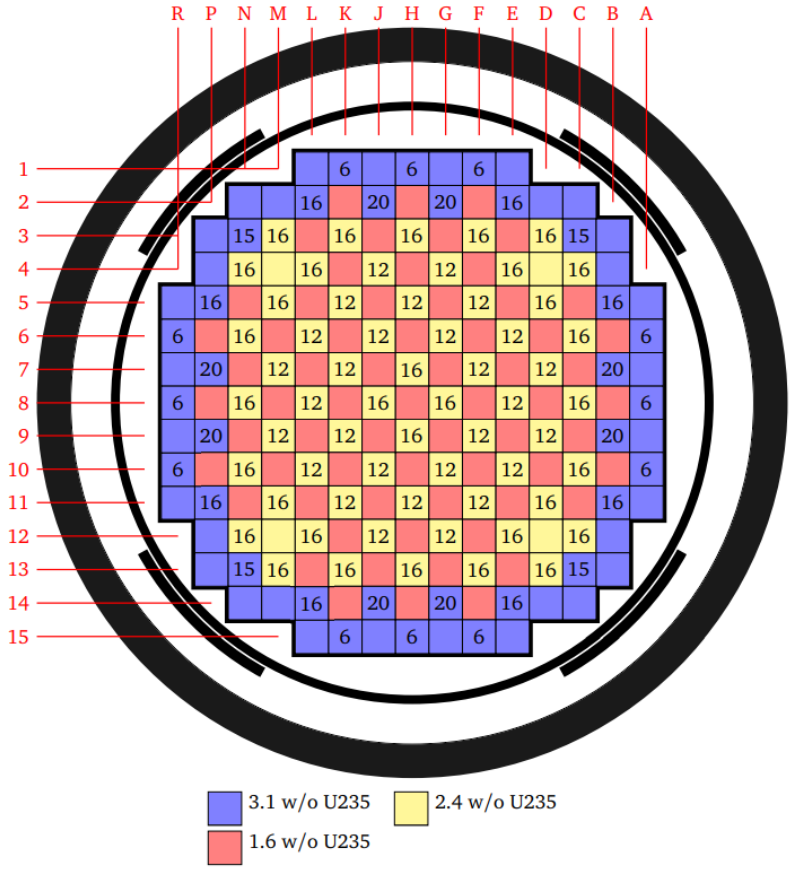


Figure 4.88 Radial configuration of the BEAVRS core (cycle 1).

First, cycle 1 ZPPTs were analyzed and the PRAGMA results were compared with the measurement data. The cycle 1 HZP initial criticality condition is given in Table 4.36; note that the rod bank D is slightly inserted and certain amount of power exists.

Table 4.36 Cycle 1 HZP initial criticality condition of BEAVRS.

Core Power	25 MWth
Core Flow Rate	61.5×10^6 kg/h
Inlet Coolant Temperature	560 °F
Rod Bank D Position	Step 213
Boron Concentration	975 ppm

Table 4.37 through Table 4.39 show the calculated and measured CBCs, CRBWs, and isothermal temperature coefficients (ITC). The ITCs were obtained by changing

the inlet temperature by 5°F. Uncertainties of PRAGMA results are negligible; each calculation employed 10 billion active histories. The design review criteria (DRC) [123] of a typical PWR is given as follows:

1. CRBW: individual bank worths within 15% or 100 pcm whichever is greater, and total bank worth within 8%.
2. CBC: ± 50 ppm or ± 500 pcm.
3. ITC: ± 2 pcm/°F.

It is confirmed that all the calculation results satisfy the DRC. However, the errors are not small, and further investigations will be required.

Table 4.37 Comparison of HZP CBCs (ppm) for various rod bank insertion cases of BEAVRS cycle 1 HZP.

Case	Measured	PRAGMA	Difference
ARO	975 (22)	978.0 (0.1)	3.0
D in	902 (22)	916.7 (0.1)	14.7
D, C in	810 (22)	817.4 (0.1)	7.4
D, C, B in	-	724.9 (0.1)	-
D, C, B, A in	686 (22)	678.9 (0.1)	-7.1
D, C, B, A, SE in	-	638.2 (0.1)	-
D, C, B, A, SE, SD in	-	574.5 (0.1)	-
D, C, B, A, SE, SD, SC in	508 (22)	486.4 (0.1)	-21.6

Table 4.38 Comparison of HZP CRBWs (pcm) of BEAVRS cycle 1 HZP.

Case	Measured	PRAGMA	Difference
D in	788 (29)	784 (2)	-4 (-0.5%)
C with D in	1203 (32)	1255 (2)	52 (4.3%)
B with D, C in	1171 (31)	1221 (2)	50 (4.3%)
A with D, C, B in	548 (26)	587 (2)	39 (7.1%)
SE with D, C, B, A in	461 (25)	502 (2)	41 (8.9%)
SD with D, C, B, A, SE in	772 (28)	784 (2)	12 (1.6%)
SC with D, C, B, A, SE, SD in	1099 (31)	1109 (2)	10 (0.9%)
Total	6042 (202)	6242 (14)	200 (3.3%)

Table 4.39 Comparison of HZP ITCs (pcm/°F) for various bank insertion cases of BEAVRS cycle 1 HZP.

Case	Measured	PRAGMA	Difference
ARO	-1.75 (0.54)	-3.12 (0.14)	-1.37
D in	-2.75 (0.54)	-4.08 (0.14)	-1.33
C, D in	-8.01 (0.54)	-9.58 (0.14)	-1.57

Next, the detector signals were compared with the calculated values. The detectors here are the fission chambers filled with U-235. Therefore, a trace amount of U-235 was filled in the instrument tubes and the local power tallies were compared with the detector signals. Figure 4.89 illustrates the relative error distribution between the tilt-corrected detector measurements and the PRAGMA power tallies. The RMS error was 1.98%, which is considered to be a reasonable value.

0.779 0.788 **1.11%**	1.011 1.040 **2.89%**					
1.065 1.084 **1.80%**	0.897 0.908 **1.24%**	1.138 1.161 **2.00%**				
0.940 0.937 **-0.31%**	1.143 1.166 **2.02%**	0.968 0.983 **1.50%**	1.249 1.266 **1.33%**			
1.147 1.163 **1.36%**	0.974 0.964 **-0.99%**	1.212 1.214 **0.18%**		1.343 1.311 **-2.40%**		
0.935 0.932 **-0.32%**	1.168 1.195 **2.34%**	0.984 0.970 **-1.45%**	1.301 1.357 **3.36%**	1.196 1.182 **-1.19%**	0.852 0.836 **-1.83%**	
1.264 1.255 **-0.68%**	0.873 0.856 **-1.92%**	1.242 1.253 **0.90%**		0.958 0.930 **-2.92%**	0.702 0.688 **-2.00%**	
0.778 0.758 **-2.60%**	0.815 0.790 **-3.10%**	0.728 0.707 **-2.87%**	0.584 0.578 **-1.11%**			Detector PRAGMA **Rel. Err.**

Figure 4.89 Comparison of the tilt corrected detector measurements and the power tallies of PRAGMA (octant folded) for BEAVRS cycle 1 HZP.

Finally, the cycle 1 depletion of BEAVRS using PRAGMA is demonstrated. The calculation conditions for whole-core cycle depletion is shown in Table 4.40 and the specification of the computing cluster used for the calculation is shown in Table 4.41. This is a super-scale calculation where 7.7 million depletable regions are defined and 360 billion histories are employed. Conventional MC codes will take at least several months to complete this calculation on an ordinary computing cluster. However, note that this calculation does not precisely model the power and rod bank histories of the BEAVRS benchmark; the core was depleted at ARO HFP condition. This calculation is to demonstrate the performance of PRAGMA for cycle depletion calculations, and rigorous analysis of the benchmark remains as the future work.

Table 4.40 Calculation conditions for BEAVRS cycle 1 depletion.

Number of Cycles	25 (Inactive), 50 (Active)
Number of Neutrons / Cycle	200,000,000
Number of Material Regions	20,225,259
Number of Depletion Regions	7,677,036
Ramp-up Factor	20
Core Power	3411 MW
Core Flow Rate	17083.3 kg/s
Inlet Temperature	566.5 K
Pressure	15.513 MPa
Libraries (K)	550 / 600 / 900 / 1200 / 1500
Burnup Steps (MWD/kgHM)	0.1 / 0.5 / 1 / 2 / … / 12 / 13
Time Marching Scheme	Full Predictor-Corrector
Number of Total Histories	3.565E+11

Table 4.41 Specification of computing resources for BEAVRS cycle 1 depletion.

Number of Nodes	6
CPU / Node	2 × Intel Xeon Gold 6230
GPU / Node	4 × NVIDIA Tesla V100
Memory / Node	768GB DDR4
Interconnect	Mellanox Infiniband EDR

Table 4.42 presents the computing time breakdown of the entire cycle depletion calculation and Table 4.43 shows the time breakdown of the MC calculation. Total computing time was around 30 hours, where only 12.5 hours were spent for the MC calculation. The depletion solver is currently un-accelerated; namely, it runs on CPU and thus takes longer time than the MC calculation. Especially, the pin-wise group collapse using ultra-fine-group spectra takes a dominant portion in the depletion time. The GPU acceleration of depletion solver remains as the future work, and with the GPU acceleration, the total computing time will be reduced significantly.

One of the significant result is that the increase of computing time in MC from the beginning-of-cycle (BOC) to the end-of-cycle (EOC) is only 1.8 times. Let alone the tallies, conventional MC codes suffer from drastic slowdowns due to the explosion of nuclides and skyrocketing of the cross section calculation time, as shown in Table 4.11. In PRAGMA, however, the increase of the cross section calculation time in the depleted fuel calculation is effectively overcome by the vectorization of cross section calculation with the fuel partitioning and energy sort schemes. As the result, the time portion of cross section calculation kernel in MC is kept under 50%.

However, the inter-domain communication time takes approximately a quarter of the total MC time, which must be improved. The communication itself is overlapped by the inner − outer iteration scheme, but the communication setup phase which is not overlapped, such as surface neutron sorting, appears to be inefficient.

Table 4.42 Computing time breakdown for BEAVRS cycle 1 depletion.

Calculation	Time
Monte Carlo Per Step Average BOC → EOC	12h 27m 24.1m 15.5m → 27.5m
Depletion Group Collapse CRAM	17h 15m 14h 31m 2h 42m
Other	39m
Total	30h 21m

Table 4.43 MC time breakdown for BEAVRS cycle 1 depletion.

Kernel	Time (Fraction)
Macroscopic	5.37h (43.26%)
Tracing	1.23h (9.87%)
Collision	0.69h (5.53%)
Sorting	1.34h (10.76%)
Domain Comm.	2.93h (23.55%)
Other	0.88h (7.03%)

Figure 4.90 shows the measured and calculated boron letdown curves, and Figure 4.91 demonstrates the 3D power distributions at BOC, middle-of-cycle (MOC), and EOC. The calculated letdown curve closely matches with the measurements, and it is expected that the accuracy will be further improved by following the operational histories more rigorously.

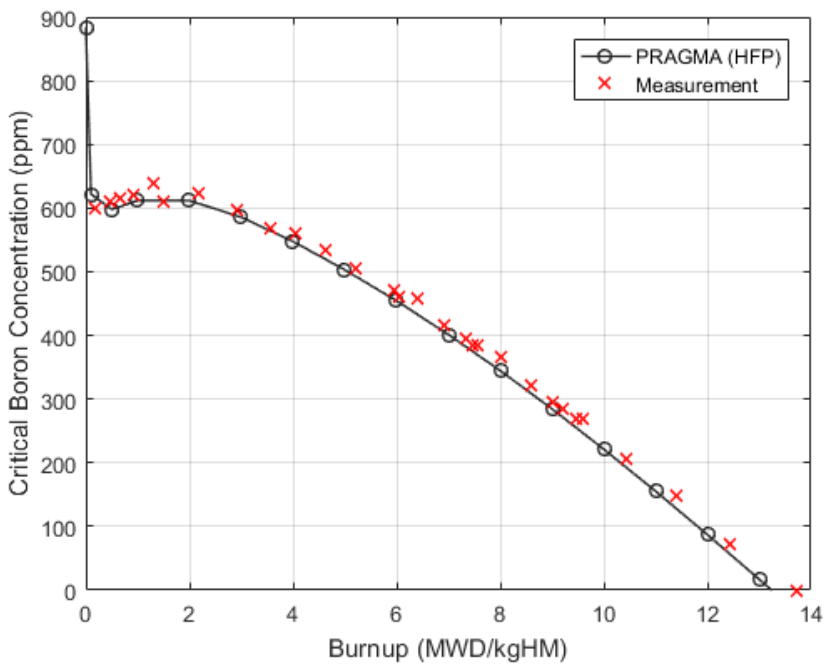


Figure 4.90 Calculated and measured boron letdown curves for BEAVRS cycle 1 depletion.

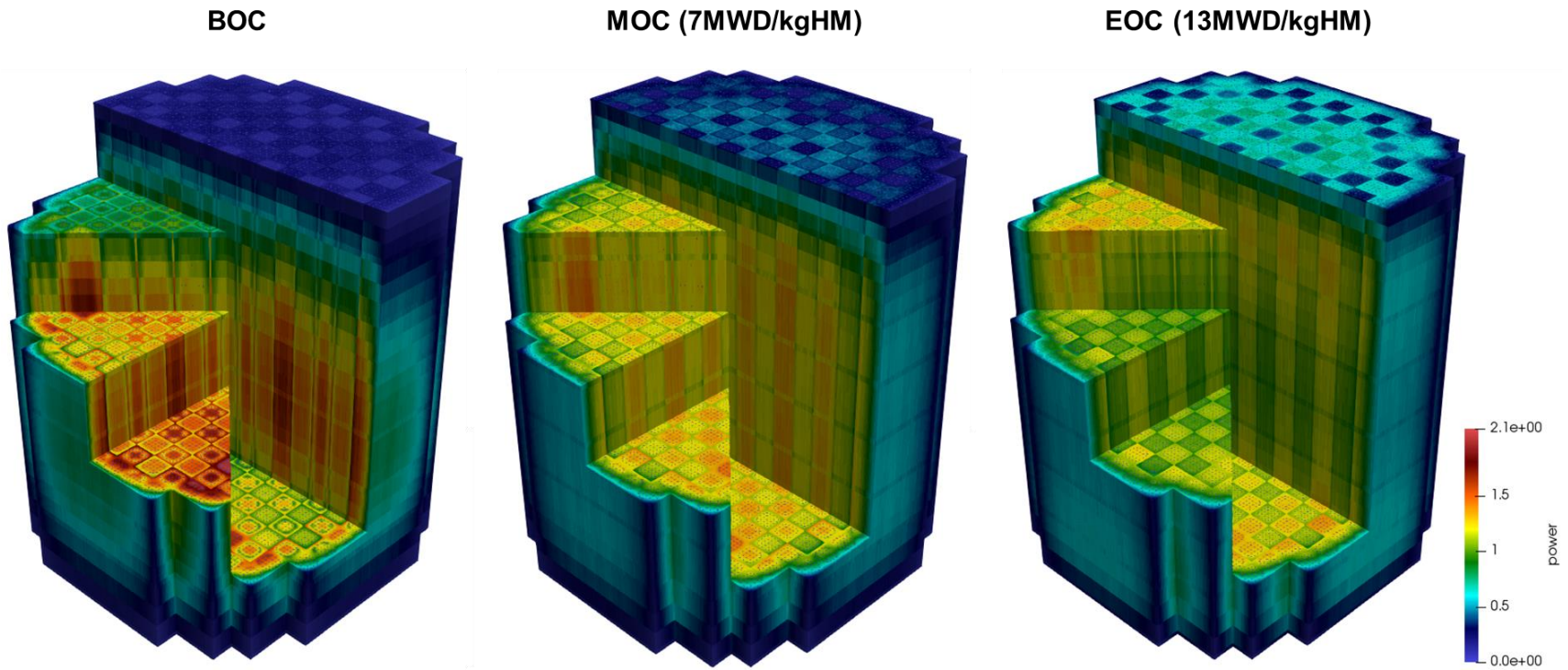


Figure 4.91 BEAVRS cycle 1 power distributions at BOC, MOC, and EOC.

Chapter 5. Conclusions

In this research, practical direct whole-core calculations employing the 2D/1D and continuous-energy MC methods were realized by the GPU acceleration. Numerous schemes and algorithms for the GPU acceleration were developed and integrated into complete frameworks which can be applied to real engineering problems. This is a meaningful progress in the development of future reactor physics codes employing GPUs which has not yet reached production level. Especially, consumer-grade GPUs were the key of retaining the practicality of direct whole-core calculations, and the results had demonstrated that the consumer-grade GPUs can be indeed powerful for scientific computing if mixed precision arithmetic is properly utilized.

For the GPU acceleration of the 2D/1D method, nTRACER served as the basis of this research. The GPU acceleration of the three primary solvers of 2D/1D method, namely the planar MOC, CMFD, and axial solvers, was conducted. The algorithms of legacy solvers had to be completely replaced to achieve efficient GPU acceleration. For the planar MOC calculation, the sweep algorithm was changed from the Gauss-Seidel scheme to the Jacobi scheme. For the CMFD calculation, the linear system structure was changed from the group major ordering to the node major ordering and the LU-type preconditioners were substituted by the DSPAI preconditioner. For the axial solver, the whole-node SENM axial solver was superseded by the augmented axial MOC solver. Algorithms specific to the GPU acceleration were also introduced, such as the CPU − GPU asynchronous execution model and the iterative refinement technique. However, the distributed parallelization strategy which assigns each GPU to an MOC plane was inherited from the conventional nTRACER approach, which we consider as the most efficient form in the computing clusters of practical size.

The resulting GPU-based steady-state solution module demonstrated significant speedups compared to the built-in CPU-based module. Ray tracing calculation which

was the dominant burden in the 2D/1D calculation showed the largest improvement, showing ~ 500 times speedup with a consumer-grade GPU compared to the single CPU core performance. A whole-core calculation could be finished in less than four minutes; with only dozens of consumer-grade GPUs, the performance equivalent to thousands of CPU cores was achieved. By embracing GPUs, nTRACER had reborn as a true practical numerical reactor realizing minute-level whole-core steady-state calculations on an affordable computing cluster.

For the GPU acceleration of the continuous-energy MC method, PRAGMA was newly developed with dedication to power reactor analysis. In contrast to nTRACER, PRAGMA has been developed to be executed on GPUs from the very beginning and elaborate optimizations for GPU acceleration were made. First of all, considerable efforts to efficiently cast the random cross section look-up and tracking into GPUs were made. The unionized grid method was improved by hashing and temperature-dependent grid collapse to shrink the double index table, which alleviates excessive memory requirement of the unionized grid method and reduces the strided accesses to the double index table. For the vectorization of tracking, an event-based tracking algorithm was developed, and the conventional history-based tracking algorithm was modified by limiting the number of transitions. Then, the neutron sorting algorithms such as region partitioning and energy sort were incorporated to the tracking loop to increase the chance of coalescing in the cross section look-up.

With these optimizations, PRAGMA achieved 4.5 times higher performance than the GPU version of Shift on the same GPU model for fresh SMR problems. However, the vectorized cross section look-up and tracking algorithms revealed real strengths in the treatment of depleted fuels that the conventional CPU-based MC calculations find cumbersome. The fuel partitioning and energy sort schemes were very effective and could reduce the cross section look-up time by 80%. As the result, PRAGMA rendered the performance equivalent to that of OpenMC using ~ 500 cores of a high-end CPU using only a single consumer-grade GPU.

Algorithms and schemes to enhance the applicability of the PRAGMA code were also developed. For accurate and efficient treatment of resonance scattering, the RST target velocity sampling scheme was devised as an alternative of DBRC and WCM. A domain decomposition scheme was implemented to realize large-scale whole-core calculations involving cycle depletion with limited GPU memories. For an efficient depletion, the MSC scheme was developed and the reaction rates are calculated from two-level flux spectra. Localized delta-tracking scheme enables fast yet accurate T/H feedback and depletion calculations by employing the delta-tracking method locally in the pellet along with the temperature majorant cross sections and local maximum number densities. The intra-fuel-rod temperature distributions are functionalized and treated analytically. Lastly, hybrid CMFD and ramp-up acceleration was introduced to effectively reduce the inactive cycle costs in massive particle simulations.

As the result, PRAGMA had brought down whole-core MC simulations employing billions of particles to a routine level. APR1400 full-core calculation with feedback employing more than 10 billion histories could be finished in 10 minutes, and VERA problem 5 full-core calculation using 100 billion histories was completed in around two hours with only 24 consumer-grade GPUs, for which KENO-VI spent 29 days with 180 CPU cores. Cycle 1 depletion of the BEAVRS core employing 360 billion histories was completed in a day and quarter, with only 12.5 hours of MC. This will be further improved by the GPU porting of the depletion solver.

Throughout all these accomplishments of this research, high potential of the direct whole-core calculation methods as the nuclear design tool was demonstrated. While the performance of CPU-based computing platforms has reached a saturation point, the performance of GPUs is still under an exponential growth and the GPUs in the next decade will be even much faster than the GPUs of today. Therefore, GPU-based direct whole-core calculations will become more and more practical and eventually replace the two-step methods in the near future.

References

[1] J. Turner et al., "The Virtual Environment for Reactor Applications (VERA): Design and Architecture," *Journal of Computational Physics*, vol. 326, pp. 544-568, 2016.

[2] "TITAN: Advancing the Era of Accelerated Supercomputing," Oak Ridge Leadership Computing Facility, [Online]. Available: https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/.

[3] K. Rupp, "42 Years of Microprocessor Trend Data," [Online]. Available: https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/.

[4] J. Hennessy and D. Patterson, Computer Architecture - A Quantitative Approach Fifth Edition, Morgan Kaufmann, 2012.

[5] "TOP500 The List," [Online]. Available: https://top500.org/.

[6] J. Lai, H. Li, Z. Tian and Y. Zhang, "A Multi-GPU Parallel Algorithm in Hypersonic Flow Computations," *Mathematical Problems in Engineering*, no. 2053156, 2019.

[7] N. Z. Cho, G. S. Lee and C. J. Park, "Fusion of Method of Characteristics and Nodal Method for 3-D Whole-core Transport Calculation," *Transactions of the American Nuclear Society*, vol. 86, pp. 322-324, 2002.

[8] J. Y. Cho et al., "Three-Dimensional Heterogeneous Whole Core Transport Calculation Employing Planar MOC Solutions," *Transactions of the American Nuclear Society*, vol. 87, pp. 234-236, 2002.

[9] H. G. Joo et al., "Methods and Performance of a Three-Dimensional Whole-Core Transport," in *International Conference on Physics of Reactors (PHYSOR2004)*, La Grange Park, IL, 2004.

[10] Y. S. Jung, C. B. Shim, C. H. Lim and H. G. Joo, "Practical Numerical Reactor Employing Direct Whole Core Neutron Transport and Subchannel Thermal/Hydraulic Solvers," *Annals of Nuclear Energy*, vol. 62, pp. 357-374, 2013.

[11] B. Collins et al., "Stability and Accuracy of 3D Neutron Transport Simulations Using the 2D/1D Method in MPACT," *Journal of Computational Physics*, vol. 326, pp. 612-628, 2016.

[12] J. Chen et al., "A New High-Fidelity Neutronics Code NECP-X," *Annals of Nuclear Energy*, vol. 116, pp. 417-428, 2018.

[13] J. Y. Cho et al., "Performance of a Whole Core Transport Code, nTER," in *The 6th International Conference on Nuclear and Renewable Energy Resources (NURER2018)*, Jeju, Korea, 2018.

[14] J. Briesmeister, "MCNP - A General Monte Carlo N-Particle Transport Transport Code," Los Alamos National Laboratory, LA-13709-M, 2000.

[15] H. J. Shim et al., "McCARD: Monte Carlo Code for Advanced Reactor Design and Analysis," *Nuclear Engineering and Technology*, vol. 44, no. 2, pp. 161-176, 2012.

[16] P. Romano and B. Forget, "The OpenMC Monte Carlo Particle Transport Code," *Annals of Nuclear Energy*, vol. 51, pp. 274-281, 2013.

[17] J. Leppänen, "Serpent − A Continuous-Energy Monte Carlo Reactor Physics Burnup Calculation Code," VTT Technical Research Center of Finland, 2015.

[18] T. Pandya et al., "Implementation, Capabilities, and Benchmarking of Shift, a Massively Parallel Monte Carlo Radiation Transport Code," *Journal of Computational Physics,* vol. 308, pp. 239-272, 2016.

[19] Y. Wu et al., "CAD-based Monte Carlo Program for Integrated Simulation of Nuclear System SuperMC," *Annals of Nuclear Energy,* vol. 82, pp. 161-168, 2015.

[20] H. Lee et al., "MCS - A Monte Carlo Particle Transport Code for Large-Scale Power Reactor Analysis," *Annals of Nuclear Energy,* vol. 139, no. 107276, 2020.

[21] S. Johnson et al., "Shift: A New HPC Monte Carlo Package," Oak Ridge National University, CASL-U-2015-0170-000-a, 2015.

[22] W. Boyd, K. Smith and B. Forget, "A Massively Parallel Method of Characteristics Neutral Particle Transport Code for GPUs," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C2013),* Sun Valley, ID, 2013.

[23] W. Boyd et al., "The OpenMOC Method of Characteristics Neutral Particle Transport Code," *Annals of Nuclear Energy,* vol. 68, pp. 43-52, 2014.

[24] Z. Zhang, K. Wang and Q. Li, "Accelerating a Three-Dimensional MOC Calculation Using GPU with CUDA and Two-Level GCMFD Method," *Annals of Nuclear Energy,* vol. 62, pp. 445-451, 2013.

[25] Y. Han, X. Jiang and D. Wang, "CMFD and GPU Acceleration on Method of Characteristics for Hexagonal Cores," *Nuclear Engineering and Design,* no. 280, pp. 210-222, 2014.

[26] P. Song et al., "Implementation and Performance Analysis of the Massively Parallel Method of Characteristics Based on GPU," *Annals of Nuclear Energy,* vol. 131, pp. 257-272, 2019.

[27] P. Song et al., "Implementation of the CPU/GPU Hybrid Parallel Method of Characteristics Neutron Transport Calculation Using the Heterogeneous Cluster with Dynamic Workload Assignment," *Annals of Nuclear Energy,* vol. 135, no. 106957, 2020.

[28] J. Tramm et al., "A Task-Based Parallelism and Vectorized Approach to 3D Method of Characteristics (MOC) Reactor Simulation for High Performance Computing Architectures," *Computer Physics Communications,* vol. 202, pp. 141-150, 2016.

[29] P. Song et al., "GPU Based Two-Level CMFD Accelerating Two-Dimensional MOC Neutron Transport Calculation," *Frontiers in Energy Research,* vol. 8, no. 124, 2020.

[30] A. Nelson, "Monte Carlo Methods for Neutron Transport on Graphics Processing Units Using CUDA," Pennsylvania State University, Master Thesis, 2009.

[31] F. Brown and W. Martin, "Monte Carlo Methods for Radiation Transport Analysis on Vector Computers," *Progress in Nuclear Energy,* vol. 14(3), pp. 269-299, 1984.

[32] T. Mori, M. Nakagawa and M. Sasaki, "Vectorization of Continuous Energy Monte Carlo Method for Neutron Transport Calculation," *Journal of Nuclear Science and Technology*, vol. 29, no. 4, pp. 325-336, 1992.

[33] A. Ding et al., "Evaluation of Speedup of Monte Carlo Calculations of Two Simple Reactor Physics Problems Coded for the GPU/CUDA Environment," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*, Rio de Janeiro, Brazil, 2011.

[34] Q. Xu, G. Yu, X. Wu and K. Wang, "A GPU-Based Local Acceleration Strategy for Monte Carlo Neutron Transport," *Transactions of the American Nuclear Society*, vol. 107, pp. 526-528, 2012.

[35] X. Du et al., "Evaluation of Vectorized Monte Carlo Algorithms on GPUs for a Neutron Eigenvalue Problem," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C2013)*, Sun Valley, ID, 2013.

[36] R. Bleile et al., "Investigation of Portable Event Based Monte Carlo Transport Using the NVIDIA Thrust Library," *Transactions of the American Nuclear Society*, vol. 114, pp. 941-944, 2016.

[37] T. Okubo, T. Endo and A. Yamamoto, "An Efficient Execution of Monte Carlo Simulation Based on Delta-Tracking Method Using GPUs," *Journal of Nuclear Science and Technology*, vol. 54, no. 1, pp. 30-38, 2017.

[38] S. Hamilton, S. Slattery and T. Evans, "Multigroup Monte Carlo on GPUs: Comparison of History- and Event-based Algorithms," *Annals of Nuclear Energy*, vol. 113, pp. 506-518, 2018.

[39] T. Liu et al., "Optimizing the Monte Carlo Neutron Cross-Section Construction Code XSBench for MIC and GPU Platforms," *Nuclear Science and Engineering*, vol. 185, no. 1, pp. 232-242, 2017.

[40] E. Brun, S. Chauveau and F. Malvagi, "PATMOS: A Prototype Monte Carlo Transport Code to Test High Performance Architectures," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2017)*, Jeju, Korea, 2017.

[41] J. Sweezy, "A Monte Carlo Volumetric-Ray-Casting Estimator for Global Fluence Tallies on GPUs," *Journal of Computational Physics*, vol. 372, pp. 426-445, 2018.

[42] R. Bergmann and J. Vujić, "Algorithmic Choices in WARP − A Framework for Continuous Energy Monte Carlo Neutron Transport in General 3D Geometries on GPUs," *Annals of Nuclear Energy*, vol. 77, pp. 176-193, 2015.

[43] S. Parker et al., "OptiX: A General Purpose Ray Tracing Engine," *ACM Transactions on Graphics*, vol. 29, no. 4, 2010.

[44] B. Molnar, G. T and D. Legrady, "A GPU-Based Direct Monte Carlo Simulation of Time Dependence in Nuclear Reactors," *Annals of Nuclear Energy*, vol. 132, pp. 46-63, 2019.

[45] S. Hamilton and T. Evans, "Continuous-Energy Monte Carlo Neutron Transport on GPUs in the Shift Code," *Annals of Nuclear Energy*, vol. 128, pp. 236-247, 2019.

[46] M. McKinley et al., "Status of LLNL Monte Carlo Transport Codes on Sierra GPUs," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2019)*, Portland, OR, 2019.

[47] "Sierra," Lawrence Livermore National Laboratory, [Online]. Available: https://computing.llnl.gov/computers/sierra/.

[48] N. Choi, J. Kang and H. G. Joo, "Massively Parallel Method of Characteristics Neutron Transport Calculation with Anisotropic Scattering Treatment on GPUs," in *International Conference on High Performance Computing in Asia-Pacific Region (HPCAsia 2018)*, Tokyo, Japan, 2018.

[49] J. Kang and H. G. Joo, "GPU-Based Parallel Krylov Linear System Solvers for CMFD Calculation in nTRACER," in *Korean Nuclear Society Spring Meeting*, Jeju, Korea, 2018.

[50] N. Choi, J. Kang and H. G. Joo, "Performance Comparison of Linear System Solvers for CMFD Acceleration," in *Korean Nuclear Society Spring Meeting*, Jeju, Korea, 2018.

[51] N. Choi and H. G. Joo, "Stability Enhancement of Planar Transport Solution Based Whole-core Calculation Employing Augmented Axial Method of Characteristics," *Annals of Nuclear Energy*, vol. 2020, no. 107440, 143.

[52] N. Choi, J. Kang and H. G. Joo, "Preliminary Performance Assessment of GPU Acceleration Module in nTRACER," in *Korean Nuclear Society Autumn Meeting*, Yeosu, Korea, 2018.

[53] N. Choi, K. M. Kim and H. G. Joo, "Initial Development of PRAGMA − A GPU-Based Continuous Energy Monte Carlo Code for Practical Applications," in *Korean Nuclear Society Autumn Meeting*, Goyang, Korea, 2019.

[54] N. Choi and H. G. Joo, "Domain Decomposition for GPU-Based Continuous Energy Monte Carlo Power Reactor Calculation," *Nuclear Engineering and Technology*, vol. 52, pp. 2667-2677, 2020.

[55] K. M. Kim, N. Choi, H. G. Lee and H. G. Joo, "Initial Development of Depletion Capability in the GPU-Based Monte Carlo Code PRAGMA," in *Korean Nuclear Society Autumn Meeting*, Changwon, Korea, 2020.

[56] N. Choi and H. G. Joo, "Analytic Treatment of Intra-Fuel-Rod Temperature Distributions in the GPU-Based Continuous Energy Monte Carlo Code PRAGMA," *Transactions of the American Nuclear Society*, vol. 122, pp. 721-724, 2020.

[57] J. Im, N. Choi and H. G. Joo, "A Study on the Optimal Use of Ramp-up Technique Under Massive Particle Condition with CMFD Acceleration," in *Korean Nuclear Society Spring Meeting*, Jeju, Korea, 2020.

[58] "CUDA Toolkit Documentation," NVIDIA Developer Documentation, [Online]. Available: https://docs.nvidia.com/cuda/.

[59] "OpenCL Reference Pages," Khronos Group, [Online]. Available: https://www.khronos.org/registry/OpenCL/sdk/2.2/docs/man/html/.

[60] "Whitepaper - NVIDIA's Next Generation CUDA Compute Architecture: Fermi," NVIDIA.

[61] M. Clark, P. La Plante and L. Greenhill, "Accelerating Radio Astronomy Cross-Correlation with Graphics Processing Units," *The International Journal of High Performance Computing Applications*, vol. 27, no. 2, pp. 178-192, 2013.

[62] V. Mehta and M. Milakov, "Optimizing CUDA Applications for the Volta/Turing Architecture," in *GPU Techonology Conference Israel (GTC Israel)*, Tel Aviv, Israel, 2018.

[63] "How AI Impacts Memory Systems," Semiconductor Engineering, [Online]. Available: https://semiengineering.com/how-ai-impacts-memory-systems/.

[64] N. Sakharnykh, "Maximizing Unified Memory Performance in CUDA," NVIDIA Developer Blog, [Online]. Available: https://developer.nvidia.com/blog/maximizing-unified-memory-performance-cuda/.

[65] J. Kraus, "An Introduction to CUDA-Aware MPI," NVIDIA Developer Blog, [Online]. Available: https://developer.nvidia.com/blog/introduction-cuda-aware-mpi/.

[66] M. Ryu, Y. S. Jung, H. H. Cho and H. G. Joo, "Solution of the BEAVRS Benchmark Using the nTRACER Direct Whole Core Calculation Code," *Journal of Nuclear Science and Technology*, vol. 52, no. 7-8, pp. 961-969, 2015.

[67] H. Hong et al., "Analyses of AP1000® First Core with Direct Whole Core Calculation Codes," in *International Conference on Physics of Reactors (PHYSOR 2016)*, Sun Valley, ID, 2016.

[68] H. Hong and H. G. Joo, "Analysis of the APR1400 PWR Initial Core with the nTRACER Direct Whole Core Calculation Code and the McCARD Monte Carlo Code," in *Korean Nuclear Society Spring Meeting*, Jeju, Korea, 2017.

[69] S. Jeon and H. G. Joo, "Analyses of the B&W-1810 and KRITZ-2 Critical Experiments with nTRACER," in *Korean Nuclear Society Spring Meeting*, Jeju, Korea, 2018.

[70] J. Kang, S. Jae and H. G. Joo, "Modelling and Preliminary Analysis of the SPERT III E-core with nTRACER," in *Korean Nuclear Society Spring Meeting*, Jeju, Korea, 2020.

[71] J. I. Yoon and H. G. Joo, "Two-Level Coarse Mesh Finite Difference Formulation with Multigroup Source Expansion Nodal Kernels," *Journal of Nuclear Science and Technology*, vol. 45, no. 7, pp. 668-682, 2008.

[72] M. Ryu and H. G. Joo, "Performance Enhancement of Anisotropic Scattering Treatment in MOC Calculation by Angular Flux Storage," in *Korean Nuclear Society Spring Meeting*, Jeju, Korea, 2016.

[73] A. Yamamoto et al., "Derivation of Optimum Polar Angle Quadrature Set for the Method of Characteristics Based on Approximation Error for the Bickley Function," *Journal of Nuclear Science and Technology*, vol. 44, no. 2, pp. 129-136, 2007.

[74] C. Moler, "Iterative Refinement in Floating Point," *Journal of the ACM*, vol. 14, no. 2, pp. 316-321, 1967.

[75] N. Z. Cho and G. S. Lee, "Comparison of CMFD and p-CMFD Acceleration

Methods for Neutron Transport Calculations," in *Korean Nuclear Society Spring Meeting*, Gyeongju, Korea, 2003.

[76] A. Zhu et al., "An Optimally Diffusive Coarse Mesh Finite Difference to Accelerate Neutron Transport Calculations," *Annals of Nuclear Energy,* vol. 95, pp. 116-124, 2016.

[77] M. Jarrett, B. Kochunas, A. Zhu and T. Downar, "Analysis of Stabilization Techniques for CMFD Acceleration of Neutron Transport Problems," *Nuclear Science and Engineering,* vol. 184, no. 2, pp. 208-227, 2016.

[78] J. Y. Cho et al., "Axial SPN and Radial MOC Coupled Whole Core Transport Calculation," *Journal of Nuclear Science and Technology,* vol. 44, no. 9, pp. 1156-1171, 2007.

[79] J. Y. Cho, J. S. Song and H. C. Lee, "Low Order Polynomial Expansion Nodal Method for a DeCART Axial Solution," in *International Nuclear Atlantic Conference*, Rio de Janeiro, Brazil, 2009.

[80] M. Hursin, "Full Core, Heterogeneous, Time Dependent Neutron Transport Calculations with the 3D Code DeCART," University of California, Berkeley, Ph.D. Thesis, 2010.

[81] R. Ferrer, J. Rhodes and K. Smith, "A Linear Source Approximation Scheme for the Method of Characteristics," *Nuclear Science and Engineering,* vol. 182, no. 2, pp. 151-165, 2017.

[82] M. Hursin, B. Collins, Y. Xu and T. Downar, "The Development and Implementation of a One-Dimensional Sn Method in the 2D-1D Integral Transport Solution," *Nuclear Science and Engineering,* vol. 176, no. 2, pp. 186-200, 2014.

[83] S. Stimpson, B. Collins and T. Downar, "Axial Transport Solvers for the 2D/1D Scheme in MPACT," in *International Conference on Physics of Reactors (PHYSOR2014)*, Kyoto, Japan, 2014.

[84] S. Stimpson, "An Azimuthal, Fourier Moment-Based Axial SN Solver for the 2D/1D Scheme," University of Michigan, Ph.D. Thesis, 2015.

[85] M. Smith, E. Lewis and B. C. Na, "Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation," NEA/NSC/DOC(2005)16, 2005.

[86] H. Park and H. G. Joo, "Effective Subgroup Method Employing Macro Level Grid Optimization for LWR Applications," *Annals of Nuclear Energy,* vol. 129, pp. 461-471, 2019.

[87] "cuSPARSE Library," NVIDIA, DU-06709-001_v11.1, 2020.

[88] "cuBLAS Library," NVIDIA, DU-06702-001_v11.1, 2020.

[89] "Intel Math Kernel Library Developer Reference - Fortran," Intel, 2020.

[90] A. Godfrey, "VERA Core Physics Benchmark Progression Problem Specifications, Revision 4," Oak Ridge National Laboratory, CASL-U-2012-0131-004, 2014.

[91] N. Horelik, B. Herman, B. Forget and K. Smith, "Benchmark for Evaluation and Validation of Reactor Simulations (BEAVRS)," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C2013)*, Sun Valley, ID, 2013.

[92] "OpenMC Documentation - Neutron Physics," [Online]. Available: https://docs.openmc.org/en/stable/methods/neutron_physics.html.

[93] R. Macfarlane et al., "The NJOY Nuclear Data Processing System, Version 2016," Los Alamos National Laboratory, LA-UR-17-20093, 2017.

[94] C. Everett and E. Cashwell, "A Third Monte Carlo Sampler," Los Alamos National Laboratory, LA-9721-MS, 1983.

[95] F. Brown, "MCNP6 Monte Carlo Code Optimization," Los Alamos National Laboratory, LA-UR-14-27694, 2014.

[96] P. Romano, "An Algorithm for Generating Random Variates from the Madland-Nix Fission Energy Spectrum," *Computer Physics Communications,* vol. 187, pp. 152-155, 2015.

[97] J. Leppänen, "Two Practical Methods for Unionized Energy Grid Construction in Continuous-Energy Monte Carlo Neutron Transport Calculation," *Annals of Nuclear Energy,* vol. 36, pp. 878-885, 2009.

[98] T. Scudiero, "Monte Carlo Neutron Transport: Simulating Nuclear Reactions One Neutron at a Time," in *GPU Technology Conference (GTC2014),* San Jose, CA, 2014.

[99] "CUB Documentation," NVIDIA Research, [Online]. Available: https://nvlabs.github.io/cub/.

[100] P. Romano and J. Walsh, "An Improved Target Velocity Sampling Algorithm for Free Gas Elastic Scattering," *Annals of Nuclear Energy,* vol. 114, pp. 318-324, 2018.

[101] "SUMMIT: Oak Ridge National Laboratory's 200 Petaflop Supercomputer," Oak Ridge Leadership Computing Facility, [Online]. Available: https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.

[102] K. Smith, "NuScale Small Modular Reactor (SMR) Progression Problems for the ExaSMR Project," Exascale Computing Project, Milestone Report WBS 1.2.1.08 ESCP-SE-08-43, 2017.

[103] L. Devroye, "Chapter 4 - Nonuniform Random Variate Generation," *Handbooks in Operations Research and Management Science,* vol. 13, pp. 83-121, 2006.

[104] W. Rothenstein, "Neutron Scattering Kernels in Pronounced Resonances for Stochastic Doppler Effect Calculations," *Annals of Nuclear Energy,* vol. 23, no. 4-5, pp. 441-458, 1996.

[105] B. Becker, R. Dagan and G. Lohnert, "Proof and Implementation of the Stochastic Formula for Ideal Gas, Energy Dependent Scattering Kernel," *Annals of Nuclear Energy,* vol. 36, pp. 470-474, 2009.

[106] T. Mori and Y. Nagaya, "Comparison of Resonance Elastic Scattering Models Newly Implemented in MVP Continuous-Energy Monte Carlo Code," *Journal of Nuclear Science and Technology,* vol. 46(8), pp. 793-798, 2009.

[107] A. Zoia, E. Brun, C. Jouanne and F. Malvagi, "Doppler Broadening of Neutron Elastic Scattering Kernel in TRIPOLI-4®," *Annals of Nuclear Energy,* vol. 54, pp. 218-226, 2013.

[108] R. Mosteller, "Computational Benchmarks for the Doppler Reactivity Defect," Los Alamos National Laboratory, LA-UR-06-2968, 2006.

[109] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory,* vol. 28, no. 2, pp. 129-137, 1982.

[110] "*k*-means Clustering," Wikipedia - The Free Encyclopedia, [Online]. Available: https://en.wikipedia.org/wiki/K-means_clustering/.

[111] A. Siegel, K. Smith, P. Fischer and V. Mahadevan, "Analysis of Communication Costs for Domain Decomposed Monte Carlo Methods in Nuclear Reactor Analysis," *Journal of Computational Physics,* vol. 231, pp. 3119-3125, 2012.

[112] W. Luscher, K. Geelhood and I. Porter, "Material Property Correlations: Comparison between FRAPCON-4.0, FRAPTRAN-2.0, and MATPRO," Pacific Northwest National Laboratory, PNNL-19147 Rev.2, 2015.

[113] C. Morris, "Can Reactors React? Is a Decarbonized Electricity System with a Mix of Fluctuating Renewables and Nuclear Reasonable?," *IASS Discussion Paper,* 2018.

[114] M. Pusa, "Higher-Order Chebyshev Rational Approximation Method and Application to Burnup Equations," *Nuclear Science and Engineering,* vol. 182, no. 3, pp. 297-318, 2016.

[115] E. Woodcock et al., "Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors an Other Systems of Complex Geometry," Argonne National Laboratory, ANL-7050, 1965.

[116] L. Carter et al., "Monte Carlo Sampling with Continuously Varying Cross Sections Along Flight Paths," *Nuclear Science and Engineering,* vol. 48, no. 4, pp. 403-411, 1972.

[117] M. J. Lee, H. G. Joo, D. Lee and K. Smith, "Coarse Mesh Finite Difference Formulation for Accelerated Monte Carlo Eigenvalue Calculation," *Annals of Nuclear Energy,* vol. 65, pp. 101-113, 2014.

[118] Y. G. Jo and N. Z. Cho, "Stablization of Monte Carlo Fission Source Distribution in p-CMFD Acceleration Method," in *Korean Nuclear Society Autumn Meeting*, Gyeongju, Korea, 2013.

[119] Korea Electric Power Corporation and Korea Hydro & Nuclear Power Co., Ltd, "APR1400 Design Control Document Tier 2, Revision 3," U.S. Nuclear Regulatory Commission, APR1400-K-X-FS-14002-NP, 2018.

[120] "The ROCS and DIT Computer Codes for Nuclear Design," Combustion Engineering Inc., CENPSD-266-P-A, 1983.

[121] D. Hollenbach, L. Petrie and N. Landers, "KENO-VI: A General Quadratic Version of the KENO Program," in *SCALE: A Modular Code System for Performing Standardized Computer Analysis for Licensing Evaluation*, ORNL/NUREG/CR/CSD-2R7, Oak Ridge National Laboratory, 2004.

[122] T. Nguyen et al., "Validation of UNIST Monte Carlo Code MCS Using VERA Progression Problems," *Nuclear Engineering and Technology,* vol. 52, no. 5, pp. 878-888, 2020.

[123] "Indian Point 3 Nuclear Power Plant Cycle 11 Startup Physics Test Report," New York Power Authority, IPN-00-004, 2000.

초 록

지난 수십 년간 이어진 CPU 계산 성능의 괄목할 발전은 전체 노심을 수송 해법을 이용하여 직접적으로 계산하는 직접 전노심 계산을 가능하게 했다. 직접 전노심 계산은 가동 원전 안전 해석 및 신형 원자로 설계에 있어서 고신뢰도의 정밀한 계산 결과를 제공해주기 때문에, 정확하고 빠른 직접 전노심 계산에 대한 지속적인 수요가 존재해 왔다. 그러나 CPU 계산 환경은 직접 전노심 계산을 일상적인 설계 해석에서 사용하기에는 여전히 충분히 빠르지 않다. 따라서 전통적인 이단계 기법이 여전히 산업계에서 주된 핵설계 도구로서의 역할을 하고 있다.

불행하게도, 직접 전노심 계산을 산업계에서 가용하게 해줄 CPU 계산 환경의 추가적인 발전은 전력 및 메모리 장벽에 의한 한계로 더 이상 기대하기 어렵다. 따라서 기존의 이단계 계산에서 직접 전노심 계산으로의 근본적인 핵설계 패러다임 전환을 이룩하기 위해 새로운 계산 환경으로의 전면적인 이행이 필요하다.

이런 점에서, 본 연구는 결정론적 및 확률론적 직접 전노심 계산의 대표 방법인 2D/1D 및 연속에너지 몬테칼로 기법의 GPU 가속 방법 및 체계를 개발하고 직접 전노심 계산의 실용화를 위한 초석을 다진다. 최근 대두된 인공지능 및 빅데이터 산업과 디스플레이 해상도의 향상은 과학 계산과 그래픽 처리의 양면에서 막대한 전산 수요를 유발하고 있으며, 이는 GPU 컴퓨팅 기술의 발전을 가속화하고 있다. 개별 소비자용 GPU는 이미 서버 CPU 수백 코어에 필적할 만큼 강력하며, 최첨단 슈퍼컴퓨터들은 GPU의 높은 전력 효율에 의존하여 목표한 성능을 달성하고 있다. 이러한 컴퓨팅 패러다임의 조류에 편승하여, 본 연구는 소비자용 GPU를 이용한 GPU 가속화로 직접 전노심 계산 기법의 시간 측면 현실성뿐만 아니라 비용 측면

현실성도 달성하고자 한다.

본 연구는 2D/1D 방법과 연속에너지 몬테칼로 방법의 효율적인 GPU 가속을 위한 알고리즘과 기법을 제시한다. 그러나 본 연구는 단편적인 알고리즘들의 모음에 그치지 않고 이들을 통합하여 완전한 해석 체계들을 구성한다. 이를 위해 본 연구는 간이 코드를 이용한 제한적인 구현에 그쳤던 기존의 연구들과 달리 상용 수준의 코드들을 이용하여 수행된다. 2D/1D 방법의 GPU 가속에 대한 연구는 nTRACER를 기반으로 수행되며 GPU 기반 연속에너지 몬테칼로 코드 PRAGMA 개발을 통해 연속에너지 몬테칼로 방법의 전 해석 과정을 GPU 가속화한다. 그리고 실제 공학 문제에 대한 적용을 시연하여 개발된 알고리즘의 효과성을 입증한다.

구체적으로, 2D/1D 방법에서는 층별 MOC, CMFD, 그리고 축방향 계산을 아우르는 nTRACER 정상상태 계산 모듈이 GPU 가속의 대상이 된다. MOC 선추적 기법 및 CMFD 선형계 해법이 대규모 병렬화에 최적화되고, MOC 계산에서는 이종 계산 환경의 이점을 살려 CPU – GPU 동시 계산이 활용된다. CMFD 계산에서는 대규모 병렬화가 가능한 DSPAI 선조건자가 LU 계열의 선조건자를 대체하며 혼합 정밀도 구현을 위한 반복적 보정 기법이 도입된다. 축방향 계산에서는 병렬 효율, 정확성, 그리고 안정성이 개선된 축방향 MOC 해법을 개발한다.

연속에너지 몬테칼로 방법에서는, 즉 PRAGMA 개발에서는 반응 단면적 검색 최적화와 무작위 행보의 벡터화가 GPU 가속의 핵심이 된다. 통합그리드 방법이 선형 해싱 기법과 핵종 별 온도 간 그리드 구조 병합을 통해 개선된다. 벡터화된 사건 기반 입자 추적 알고리즘이 도입되며, 지역 구분 및 에너지 정렬 기법이 사건 기반 알고리즘과 함께 사용되어 연소 핵연료 계산에서 반응 단면적 검색의 메모리 코얼레싱 확률을 획기적으로 향상시킨다.

GPU 기반 연속에너지 몬테칼로 계산의 실제 가동 원전 적용을 위한 다양한 기법들도 개발된다. 효과적인 공명 산란 처리를 위해 DBRC와 WCM의 단점을 해결한 RST 표적핵 속도 추출법을 개발하고, 제한된 GPU 메모리 용량으로 대규모 동력로 계산을 실현하기 위한 영역 분할 계산법이 도입된다. 더하여, 실용적인 몬테칼로 연소 계산을 위한 독창적인 MSC 기법이 사용되며, 국소 Delta-tracking 기법은 추가적인 비용 없이 연료 소자 내 온도 분포와 조성 변화를 정확하게 처리할 수 있어 효율적인 열 궤환 및 연소 계산을 가능하게 한다. 대량 입자 계산의 실용성을 높여주는 CMFD 및 Ramp-up 선원 수렴 가속 기법도 도입된다.

본 연구의 결과는 실용적인 핵설계 도구로서 직접 전노심 계산 기법의 높은 잠재력을 보여주었다. 2D/1D 방법을 이용한 전노심 정상상태 계산은 수 분내에 수행될 수 있었고, 수십억 개의 입자를 사용한 대량입자 전노심 몬테칼로 계산은 분 단위에 수행될 수 있는 일상적인 작업이 되었다. 모든 이러한 성과는 수십 장의 상용 GPU를 장착한 실용적인 계산 클러스터로 달성되었다. GPU의 성능은 여전히 지수적으로 성장하고 있으며, 따라서 개발된 해석 체계들의 실용성 또한 시간이 지남에 따라 더욱 향상될 것이다.

주요어: GPU 가속
직접 전노심 계산
2D/1D 방법
연속에너지 몬테칼로 방법
PRAGMA

학번: 2016-29920