Optimizations of the Energy Grid Search Algorithm in Continuous-Energy Monte Carlo Particle Transport Codes

Jonathan A. Walsh^{a,*}, Paul K. Romano^b, Benoit Forget^a, Kord S. Smith^a

^a *Department of Nuclear Science and Engineering
Massachusetts Institute of Technology
77 Massachusetts Avenue, 24-107
Cambridge, MA 02139, United States*

^b *Mathematics and Computer Science
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439, United States*

Abstract

In this work we propose, implement, and test various optimizations of the typical energy grid-cross section pair lookup algorithm in Monte Carlo particle transport codes. The key feature common to all of the optimizations is a reduction in the length of the vector of energies that must be searched when locating the index of a particle's current energy. Other factors held constant, a reduction in energy vector length yields a reduction in CPU time. The computational methods we present here are physics-informed. That is, they are designed to utilize the physical information embedded in a simulation in order to reduce the length of the vector to be searched. More specifically, the optimizations take advantage of information about scattering kinematics, neutron cross section structure and data representation, and also the expected characteristics of a system's spatial flux distribution and energy spectrum. The methods that we present are implemented in the OpenMC Monte Carlo neutron transport code as part of this work. The gains in computational efficiency, as measured by overall code speedup, associated with each of the optimizations are demonstrated in both serial and multithreaded simulations of realistic systems. Depending on the system, simulation parameters, and optimization method employed, overall code speedup factors of 1.2 – 1.5, relative to the typical single-nuclide binary search algorithm, are routinely observed.

Keywords: Monte Carlo, particle transport, search algorithm, optimization, multithreading

---

1. Introduction

In neutron scattering and reaction theory, the probability of a neutron interacting with a nuclide in a particular way is characterized by a cross section [1]. The values of these cross sections, which depend strongly on the energy of the incident neutron, are different for every nuclide and for every type of interaction. Monte Carlo codes utilize cross sections when simulating the behavior of neutrons in a system. The neutron cross section data utilized in modern Monte Carlo codes often come in ACE (A Compact ENDF¹) Format data libraries that are generated with the NJOY nuclear data processing code system [4]. These pre-processed libraries contain data in so-called pointwise, continuous-energy form which consists of energy-cross section

---

*corresponding author

Email address: walshjon@mit.edu (Jonathan A. Walsh)

¹Evaluated Nuclear Data File (ENDF) may refer to a file format or an actual file of evaluated nuclear data. ENDF-6 [2] is the internationally adopted format for the storage of evaluated nuclear data. ENDF/B-VII.1 [3], on the other hand, refers to the latest library of evaluated nuclear data files, in ENDF-6 format, distributed by the USA.

pairs that are piecewise linearly interpolable in both energy and cross section. So, for each nuclide in a simulation, there is a one-dimensional grid of energies, and for each type of interaction associated with that nuclide, there is a corresponding one-dimensional grid of cross section values.

In a Monte Carlo neutron transport simulation, due to the nuclide dependence of cross section values, every time that a neutron enters a material with a composition of nuclides different from the composition of the previous material, a new set of cross section values is needed to describe interaction probabilities in the new material region. Similarly, due to the neutron energy dependence of cross section values, each time that the laboratory system energy of a neutron changes — as in an interaction with a nuclide — a new set of cross section values is needed to describe interaction probabilities for the new neutron energy. As a result, every time that a neutron moves into a new material or changes energy, total interaction cross sections must be computed for the material at the current neutron energy so that the tracklength between the neutron's current location and next collision site can be sampled. Additionally, various cross section values are needed at collision sites and cell boundaries for the sampling of reaction types and the accumulation of tallies [5].

There are different methods by which Monte Carlo codes handle neutron energy grid and cross section data. The methods have evolved to balance the competition between memory and speed that is frequently encountered when dealing with the management of data in Monte Carlo simulations. While a detailed accounting of all the cross section treatment methods currently in use is beyond the aim of this introduction, considering a few common procedures will be useful.

First, there is the option of separately storing each individual nuclide's energy grid and cross sections. When a cross section value for some nuclide is needed, the value is computed by linear-linear interpolation between the cross section values corresponding to the energies that bound the neutron's current energy so that the microscopic cross section, σ , at laboratory coordinate system neutron energy, E , for some interaction, x , is given by

$$\sigma_x(E) = \sigma_x(E_i) + \frac{E - E_i}{E_{i+1} - E_i}(\sigma_x(E_{i+1}) - \sigma_x(E_i)), \quad (1)$$

where E_i and E_{i+1} are the nearest lower and upper bounding energy grid points, respectively, for the exact neutron energy. In order to carry out this interpolation, a search of the nuclide's energy grid must be performed so that the index of one of the bounding energy points can be determined. A binary search of the sorted energy grid is typically employed for the task. This algorithm achieves $O(\log(N_{\text{grid}}))$ scaling whereas a linear search displays scaling of $O(N_{\text{grid}})$ where N_{grid} is the length of the grid to be searched [6]. Historically, this treatment of energy grid-cross section data has been widely used. It is the default procedure in versions of the MCNP Monte Carlo particle transport code [7] prior to 6.1.1 [8]. Its main advantage, which still resonates for obvious reasons, is that it preserves, with complete accuracy, all of the physical information contained in an ACE cross section file with the minimum allowable memory requirement. Its chief shortcoming is that energy grid lookups and cross section computations must be performed on-the-fly, whenever they are needed, for each nuclide required by the simulation.

The unionized energy grid treatment [9] is one method which has been shown to significantly cut down simulation runtime relative to the individual nuclide energy grid method just described. This algorithm, which can be traced at least as far back as its implementation in RACER [10] and RCP01 [11] and was recently reintroduced in the Serpent code [12], calls for the construction of a grid containing the union of all of the energy points from the individual grids of every nuclide in the problem. Then, the individual reaction cross sections for each nuclide are stored on this same grid. A single energy grid for the entire problem can reduce the time spent in performing binary searches to locate the appropriate energy grid index. In fact, a single binary search determines the energy and cross section indices for each nuclide in the problem because they all utilize the same grid structure. This means that a single interpolation factor (the ratio of energies in Eq. 1) can be used for all nuclides, further accelerating any cross section computations. In addition, the unionized grid provides a natural method for storing material cross sections that are frequently accessed within a simulation such as those for the total, elastic scattering, absorption, and fission interactions.

While this method has been shown to yield impressive speedup over treatments with unique energy grids for each nuclide [9], the drawback of the unionized grid is that it can dramatically increase the memory needed for cross section data storage. Because each nuclide's cross section grids contain points corresponding

to the energy grid points from every other nuclide in the problem, there is storage of many unneeded data points. This is an important consideration, especially with respect to current and proposed high-performance computing (HPC) architectures. The on-node memory of these systems can become limiting for large, practical Monte Carlo simulations [13]. For example, we can consider a full core depletion problem from the field of reactor physics. In a simulation of this type, it may be required that the inventories of hundreds of nuclides be tracked. Each of these nuclides may have up to hundreds of thousands of energy and cross section grid points; the unionized energy grid will have millions. For each of these grid points, the cross section values for multiple reactions must be stored. If we allow for the existence of the same nuclide at multiple temperatures, as is encountered in physical systems, we may need to store cross section data on a fine temperature grid [14]. Intervals on the order of 10 K have been suggested as a possible requirement [13]. The cross section data burden associated with the problem described can exceed 100 GB of memory — a value that exceeds the on-node memory capacity of most available HPC systems. This memory footprint may approach 1 TB when a unionized grid is employed.

As a means of reducing the added memory burden introduced by the unionized grid method, a companion double indexing scheme was developed [9]. With this scheme, rather than storing each nuclide’s cross section data on the dense unionized grid for the entire problem, a code must only store one dense grid — that of the unionized energies. Then, for each nuclide, a mapping is constructed from the indices of the problem’s unionized energy grid to the indices of that nuclide’s individual energy grid and cross sections. In this way, a single binary search on the unionized grid is still sufficient to determine the indices on individual nuclide grids, and without the added memory requirements of storing each nuclide’s cross section data on the dense grid. Depending on the problem, this double indexing of grid values can reduce the memory burden of the repeated unionized grid method by roughly one half [9].

Recently, energy grid search algorithms making use of hashing functions have received renewed attention [15, 16]. In this type of algorithm, as applied to the problem at hand, a function will take the neutron energy and map it directly to bounding indices on an energy hash grid. These hash grid indices are then mapped to indices on individual nuclide energy grids. By quickly mapping a neutron energy to a narrowed range of energies from a full grid and then performing a binary search in that subset of energies,² the time spent looking up an energy grid index can be reduced.

The search of an energy grid to determine the index corresponding to an energy grid point adjacent to the current neutron energy and — after the interpolation step is complete — the cross section values at that energy is a process that must be repeated trillions of times in realistic reactor physics simulations. In aggregate, these searches are computationally expensive, often occupying an appreciable fraction of overall runtime. This is true whether a unionized or individual nuclide energy grid scheme is employed, though, as discussed, unionized schemes are typically faster. For practical, full core reactor simulations with hundreds of nuclides present, it has been suggested that 80-85% of total runtime may be spent looking up energy grid indices [17]. Development and demonstration of computational methods for reducing this expense are the aims of this work.

In Section 2 we detail the theory behind the proposed energy grid search optimizations and briefly sketch out their implementation in the OpenMC Monte Carlo neutron transport code [18]. Section 3 gives an overview of the test cases that we use to measure the effectiveness of the new optimization schemes as well as the numerical results of those tests. A concluding discussion is offered in Section 4.

2. Energy Grid Search Optimizations

In this section we propose optimizations to the energy grid search algorithm in continuous-energy Monte Carlo neutron transport codes. The theory behind each of the optimizations is presented along with brief descriptions of how they are implemented in the OpenMC code. Other considerations are addressed where appropriate. Also, it is important to note that none of the presented optimizations alter the physics of a simulation in any way. The optimizations do not change the energy grid indices that are identified by the energy grid lookup algorithm. All that is changed is the procedure by which those indices are determined.

---

²If the narrowed energy range has few enough points, a linear search may be faster.

2.1. Scattering Kinematics-constrained Searches

The models used to treat neutron-nucleus scattering kinematics in Monte Carlo codes make reasonable assumptions about collision physics as well as the behavior of the modeled system, in general, in order to simplify calculations without appreciably impacting the accuracy of the results. Often, these assumptions place restrictions on the outcome of a particular interaction. We can take advantage of these constrained outcomes in designing methods for accelerating energy grid lookups. Two such methods are presented below.

2.1.1. Stationary Target Nuclei

When the speed of an incident neutron is sufficiently high (i.e. much greater than the speeds associated with the thermal motion of target nuclei), the kinematics of a neutron-nucleus interaction can safely be treated as if the target nuclide is at rest in the laboratory frame of reference with a velocity of zero [19]. When the target-at-rest approximation is invoked, by conservation of energy and linear momentum, all elastic scattering events result in a reduction of the neutron's kinetic energy in the laboratory coordinate system. That is, upscattering is not possible.

Also, level inelastic scattering, which occurs when the combined kinetic energy of the neutron-nucleus system in the center-of-mass reference frame is sufficient to reach the first excited level of the target nuclide and is characterized by a final constellation with total kinetic energy less than that of the initial constellation,³ and other compound nucleus energy threshold reactions which do not terminate the simulation of the incident particle should be considered. With a resting target, these threshold reactions occur at incident neutron energies of a few MeV in light nuclides and at tens of keV in heavier nuclides [1]. These energies are much greater than the energies of thermally agitated nuclei that are encountered in reactor systems which are typically a fraction of an eV. Therefore, the kinematics of threshold reactions such as level inelastic scattering can be treated with the target-at-rest approximation, and, by conservation of energy, upscattering cannot occur.

So, we see that any of these events in which the kinematics are treated with the target-at-rest approximation will result in a loss of neutron kinetic energy in the laboratory system. This fact can be exploited to construct an energy grid search optimization. In Monte Carlo codes, when handling scattering kinematics, it is a standard procedure to neglect the thermal motion of target nuclei when the incident neutron energy is above some threshold. This threshold is often taken to be $400k_{\text{B}}T$ for all nuclides except ^1H — to which no threshold is applied [20] — where k_{B} is the Boltzmann constant and T is the temperature of the material being traversed by the neutron.⁴ When a neutron's incident energy is greater than this threshold, we know that, after a scattering event, its energy will be less than it was before the event. This means that we do not have to search an entire nuclide or unionized energy grid to determine a post-scatter energy grid index. We only have to search the section of the energy grid corresponding to energies less than the pre-scatter neutron energy.

This is accomplished by first storing the index of the energy grid point — on a unionized or individual nuclide grid — just above the current neutron energy, E_{MAX} . The current energy is the maximum energy that the neutron can possibly have after its next scatter from a stationary nucleus and E_{MAX} is its nearest upper bound on the energy grid in question. Then, if the next event is a scatter, we simply have to search a subset of the energy grid, $\{E_1, E_2, \dots, E_{\text{MAX}}\}$, to find the post-collision neutron energy grid index.

2.1.2. Thermal Motion of Target Nuclei

At incident neutron energies below the $400k_{\text{B}}T$ target-at-rest cutoff, the thermal motion of target nuclei can no longer be ignored in the handling of scattering kinematics with negligible consequences and the upscattering of neutrons to higher-than-incident energies is now allowed. However, it is still possible to constrain the portion of a nuclide or unionized energy grid that must be searched to find the index of a post-scatter energy.

---

³This type of inelastic scattering event is to be distinguished from the low energy inelastic scattering that arises as a result of the effects of chemical binding of target nuclei in molecules.

⁴As a point of reference, $400k_{\text{B}}T$ for a material at 900 K translates to an energy of 31.0 eV.

In generating effective, temperature-dependent (i.e. Doppler broadened) cross section data for use in transport simulations, nuclear data processing codes typically employ some variation of the SIGMA1 algorithm for numerical Doppler broadening [21]. In the Doppler broadening algorithm, the target nuclei are assumed to behave as a free gas with isotropic velocities described by the Maxwell-Boltzmann distribution. It is further assumed that the maximum energy of a target nuclide with which a neutron can interact is $16k_{\text{B}}T$. This is because the probability of a particle existing in a Maxwell-Boltzmann free gas at an energy greater than $16k_{\text{B}}T$ is negligible. Consistency requires that the same $16k_{\text{B}}T$ maximum target energy be utilized when treating scattering kinematics.

With the maximum target nuclide and incident neutron energies both known, the maximum energy to which a neutron can be upscattered in a given event, $416k_{\text{B}}T$, can be determined. This value, E_{upscat} , and the corresponding upper bound energy grid index, can be determined for a single nuclide, single material, or entire problem at simulation initialization. The energy grid search optimization based on this knowledge simply restricts the post-scatter searchable energy grid to $\{E_1, E_2, \dots, E_{\text{upscat}}\}$ whenever the incident neutron energy is less than $400k_{\text{B}}T$, or whatever thermal motion cutoff energy is used in a particular simulation.

2.2. Material Energy Grid Unionization

The optimization presented in this section follows closely the unionized energy grid method [9] with the distinguishing feature being that a unionized energy grid and cross section values are constructed for each material, individually, rather than globally for the entire problem.

When a neutron is being tracked as it streams across the geometry of a system in a Monte Carlo transport simulation, the only cross section information that is needed is the total cross section for a single material, which is simply the summation of nuclide total cross sections over all nuclides in that material. Therefore, at the time that this required total material cross section is computed, the cross section data for all other materials are irrelevant. This implies that we can, at least,⁵ neglect the nuclides which are not in the current material when computing a cross section. The individual nuclide energy grid method outlined earlier exploits this naturally but the global unionized grid scheme does not.

Constructing unionized energy-cross section grids on a material-by-material basis preserves the key advantage of the global unionization — that is, a single binary search will identify the desired energy grid indices of multiple nuclides — while reducing the size of the energy grid that must be searched. In addition to the potential speedup associated with the search of a grid with a reduced length, this scheme can also reduce memory because, for each nuclide, we must only store values on the unionized material grid instead of a global grid containing all energy points for all nuclides in the problem. A unionized material grid may also be used in conjunction with an appropriately-modified double indexing methodology which maps indices on the unionized material grid to indices on the individual grids of the nuclides in the material to further lessen memory demands.

There are a few caveats related to the use of unionized material grids worth noting here. With a global unionized grid, when a neutron streams into a new material, the grid indices for the nuclides in that material at the current energy are immediately known. They were determined in the last binary search that occurred, regardless of the material, because the global grid contains the energy points for all nuclides in the problem. That is, the computational expense for this immediate knowledge of indices is paid earlier in the simulation. With a unionized material grid, a binary search must occur whenever the neutron enters a new material, because, in general, knowledge of the previous material's energy indices is of no value in the current material.

Also, the benefits of a unionized material grid may be somewhat reduced in simulations where the same nuclide is contained in materials with different unionized grids. This is because a nuclide's energy index in one material's unionized grid will generally not be the same as its index in another material's unionized grid if the two grids are not identical. However, this is not as much of a handicap as it may, at first, appear to be. In the case of a depletion calculation, where there will be many different fuel material compositions (in terms of nuclide densities), the unionized material grid for those materials can actually be the same. When

---

⁵If the cross sections for only a single nuclide are needed, as in sampling the type of reaction, all other nuclides in the problem are extraneous.

dealing with microscopic cross sections, it does not matter how much of a nuclide is present, just whether or not it is present, when deciding if storage of its cross sections is required. Additionally, in the case of non-isothermal fuel conditions, with the same nuclide existing at different temperatures, we are no worse off by using the unionized material grid than by using the global grid. The nuclides' different energy grids for the different temperatures must be stored in either case.

2.3. Energy Hash Table Searches

The optimizations presented in this section rely on hashing functions that take the neutron energy at which a cross section must be calculated and return the indices, into either an individual nuclide or globally unionized energy grid, which correspond to energies that bound the current energy. The following subsections discuss two simple hashing schemes, as well as the problem of determining an optimal hashing treatment.

2.3.1. Logarithmic Energy Spacing

This energy grid lookup optimization closely follows a recently reintroduced method [16]. In it, the energy range for which cross section data exist is divided logarithmically. This can be done on either a nuclide-by-nuclide or global grid basis. That is, in the individual nuclide case, the j^{th} nuclide's energy range is divided into N_{hash}^j equal lethargy bins of lethargy width

$$\Delta u_{\text{hash}}^j = \frac{\log\left(\frac{E_{N_{\text{grid}}}^j}{E_1^j}\right)}{N_{\text{hash}}^j} \quad (2)$$

where E_1^j and $E_{N_{\text{grid}}}^j$ are the minimum and maximum values on the nuclide's energy grid, respectively. Then, as a pre-processing step, before the simulation begins, the indices on the nuclide's original energy grid that correspond to a set of coarse hash table energies, $\{E_{H,1}^j, E_{H,2}^j, \dots, E_{H,N_{\text{hash}}^j+1}^j\}$, where

$$E_{H,N_{\text{hash}}^j-i+1}^j = \frac{E_{N_{\text{grid}}}^j}{\exp\left(i\Delta u_{\text{hash}}^j\right)}, \text{ for } i = N_{\text{hash}}^j, N_{\text{hash}}^j - 1, \dots, 1, 0, \quad (3)$$

are stored. Now, in a simulation, whenever a cross section must be computed, the indices in the coarse hash table that bound the current neutron energy, E_n , can be determined directly from a rearranged form of Eq. 3 to be

$$k_{\text{hash}}^j(E_n) = N_{\text{hash}}^j - \left\lceil \frac{\log\left(\frac{E_{N_{\text{grid}}}^j}{E_n}\right)}{\Delta u_{\text{hash}}^j} \right\rceil \quad (4)$$

and $k_{\text{hash}}^j(E_n) + 1$. These hash indices are then mapped to the corresponding indices on the original nuclide energy grid. Once the subset of the original nuclide grid determined by these indices is isolated, a binary search is performed within the subset to find the exact index for the current energy. This scheme increases the speed of energy grid lookups by reducing the length of the searchable grid to the length of the grid subset bounded by the hash table energies. As implemented in a beta release of MCNP6.1.1 [8], this method is applied to individual nuclide energy grid lookups. In this work we implement the method in OpenMC and extend its use to global unionized grids.

2.3.2. Constant Energy Spacing

As will be discussed shortly, the choice of an effective hash table energy structure is not entirely straightforward. To first order, the number of cross section computations, and, thus, energy grid lookups, required by a simulation is proportional to the flux. In many systems, neutron slowing down results in a flux that approaches a constant value, in lethargy-space, over a wide range of energies. Without any information about the spacing of energy grid points, this suggests that the choice of logarithmically-spaced hash table

energies is a good one. However, in order to capture the widely varying cross section structure encountered in different energy regions, energy grid points are generally not evenly spaced. This suggests that hash table structures other than logarithmic may be effective. With this in mind, a simple constant energy spacing hash table is also employed. The j^{th} nuclide's energy range is divided into N_{hash}^j bins spaced equally in energy with energy bin widths of

$$\Delta E_{\text{hash}}^j = \frac{E_{N_{\text{grid}}}^j - E_1^j}{N_{\text{hash}}^j}. \quad (5)$$

In a fashion similar to the case of constant lethargy spacing, we proceed by constructing the coarse hash table energy grid, $\{E_{H,1}^j, E_{H,2}^j, \dots, E_{H,N_{\text{hash}}^j+1}^j\}$, where

$$E_{H,i+1}^j = E_1^j + i\Delta E_{\text{hash}}^j, \text{ for } i = 0, 1, \dots, N_{\text{hash}}^j. \quad (6)$$

The nuclide or unionized energy grid indices that correspond to the hash table energies are stored. Just as before, for a given neutron energy, the bounding hash table energies can be directly calculated and mapped to the stored nuclide or unionized grid indices. These indices are then used as the bounds for a binary search on a subset of the original energy grid to arrive at the desired energy grid index and corresponding cross sections.

2.3.3. Optimal Energy Spacing

In the general case, selection of an optimal energy hash table structure — that is, the selection of an ensemble of optimal hash table energy boundaries — is non-trivial. The optimized structure of table boundaries will be a complicated function of the energy spectrum of the neutron flux of the system as well as the density — in energy space — of the individual or unionized energy grids on which cross sections are represented for nuclides present in the system. To illustrate this point, we briefly outline the problem that must be solved in order to determine this optimal spacing of hash energies.

We start with the goal of minimizing the time spent in energy grid searches. That is, we wish to identify the set of energy hash table boundaries in a problem with J total nuclides, $\{E_{1/2}^j, E_{3/2}^j, \dots, E_{N_{\text{hash}}^j-1/2}^j, E_{N_{\text{hash}}^j+1/2}^j\}$, where $E_{i-1/2}^j$ and $E_{i+1/2}^j$ are the energies that bound the i^{th} hash table bin of the j^{th} nuclide, that minimizes the value of

$$T(\{\Delta E_i^j\}) = \sum_{j=1}^J \sum_{i=1}^{N_{\text{hash}}^j} L_i^j(\{\Delta E_i^j\}) \times \bar{T}_i^j(\{\Delta E_i^j\}), \quad (7)$$

the total time spent in a simulation performing energy grid lookups. Here, L_i^j and $\bar{T}_i^j$ stand, respectively, for the number of energy grid lookups that must be performed in energy hash bin i of nuclide j and the mean time spent in each of these lookups. Each of these quantities is dependent on the set of energy hash grid spacings, $\{\Delta E_i^j\}$. With the $O(\log(N_{\text{grid}}))$ scaling of the binary search algorithm and an expression for the number of required lookups given by

$$L_i^j(\{\Delta E_i^j\}) \propto \int_{\Delta E_i^j} dE' \int_0^\infty dE \phi(E) f(E \rightarrow E'); \quad (8)$$

$$E' \in [E_{i-1/2}^j, E_{i+1/2}^j],$$

we are left wanting to minimize

$$T(\{\Delta E_i^j\}) \propto \sum_{j=1}^J \sum_{i=1}^{N_{\text{hash}}^j} \int_{\Delta E_i^j} dE' \int_0^\infty dE \phi(E) f(E \rightarrow E') \times \log(N_{\text{grid},i}^j(\{\Delta E_i^j\})); \quad (9)$$

$$E' \in [E_{i-1/2}^j, E_{i+1/2}^j].$$

Solving this minimization problem is an exercise in non-linear optimization. In practice, it is not a problem we wish to solve as its solution depends in a complicated way on the physics of the system at hand. In

particular, the transfer function, $f(E \rightarrow E')$, varies with nuclide-dependent, energy-dependent cross section values and models of reaction physics. The number of energy grid points in a hash bin, $N_{\text{grid},i}^j$, will also depend in a complicated fashion on energy boundaries as the energy spacing between energy grid-cross section pairs can fluctuate dramatically due to the need to capture resonance structure of cross sections with pointwise data. Finally, the energy-dependent neutron flux, $\phi(E)$, is likely the most problematic term in Eq. 9. It also is a function with strong variations in energy. To further complicate the matter, we do not have a solution for the flux, a priori, as we do, in theory, for the transfer function and number of energy grid points in a given interval. This means that, to solve the generalized energy hash spacing optimization problem so that we may speed up our simulation, we must first run the simulation with an un-optimized spacing so that the flux is determined. There are, of course, simplifying assumptions that can be made about the energy dependence of cross sections and flux that would facilitate a more tractable problem formulation. This is, however, beyond the scope of the present work.

3. Results and Analysis

In this section we present the descriptions and results of simulations that are carried out in order to determine the effectiveness of the proposed optimizations in achieving their aim — overall code speedup. Speedup is defined here as the ratio of the particle simulation rate for a calculation in which an optimization scheme is applied to the simulation rate observed for the same calculation with no energy grid search optimizations employed. An attempt is made to select test problems that exhibit physical phenomena relevant to practical reactor physics and criticality safety applications. With this in mind, the models for three of the test problems shown here are derived from the Benchmark for Evaluation and Analysis of Reactor Simulations (BEAVRS) [22]. Pin cells with both fresh and depleted fuel are simulated along with a depleted full core. A fourth model consisting of a simple metal sphere with a fixed source is also simulated.

Continuous-energy pointwise cross section data from the ENDF71x nuclear data library [23] are utilized in all simulations. Thermal scattering from ^1H nuclei bound in light-water molecules is treated with the continuous $S(\alpha, \beta)$ data found in the ENDF71SaB library [24]. These libraries consist of ENDF/B-VII.1 nuclear data [3] processed with the NJOY code system, version 99.393 [4].

All simulations are run with 100 active batches. In the effective multiplication factor, k_{eff} , eigenvalue calculations, an additional 10 inactive batches are run.⁶ Each batch consists of 10^4 particles. A spatially uniform fission source over all fissionable material is used to initiate each run. For the problems considered, excepting the full core model, these simulation parameters are typically sufficient to converge on a k_{eff} value in the inactive cycles. It is not a primary concern, though, if they are not. Because we are only monitoring code speedup, and the reference and optimization cases are the exact same problems from a physical standpoint, runtime results would still be relevant and of interest. Results from a simulation with an unconverged eigenvalue would indicate the speedup to be expected in the period of the simulation in which the eigenvalue is converging. The only tallies present in these simulations are those for k_{eff} . Including additional tallies would increase overall runtimes and, therefore, reduce the overall speedup values resulting from the optimizations somewhat.

For the hashing optimization runs, the presented results are obtained with 10^4 bins.⁷ Verification that the introduction of each optimization does not alter the physics of the simulations — as it shouldn't⁸ — is carried out by simply confirming that the batch eigenvalues, batch Shannon entropy, running mean eigenvalues, running standard errors of the mean eigenvalues, and system leakages are all unchanged from the reference cases.

The reference results are obtained with release 0.6.1 of OpenMC [25]. Each of the optimization methods is implemented in its own branch of that same release. These branches are not modified in any way other

⁶Inactive batches are not required in calculations with a fixed source.

⁷Comparable results, and a reduction in the already-modest memory overhead, can be obtained with a reduction to 10^3 bins.

⁸As discussed earlier, the energy grid index found with each algorithm does not differ from the reference case — only the method by which that index is identified does.

than to implement one specific optimization method apiece; no combined optimizations are tested. All code is compiled with release 4.9.0 of the GNU Fortran (GFortran) compiler [26]. Both serial and multithreaded simulations are performed on dual Intel Xeon E5620 2.4 GHz quad-core processors. The observed particle simulation rates that are used in the computation of speedup results can vary by up to a few percent in identical simulations, when repeated.

3.1. Fresh Fuel Pin Cell

First, we isolate a 3.1 wt.% enriched fresh fuel pin cell at beginning-of-core, hot zero-power conditions, with fuel and all other materials at 600 K, from the full BEAVRS model. A summary of the materials present in this model including the number of nuclides and energy grid points per material can be found in Table 1. Reflective boundary conditions are applied on each of the six pin cell faces resulting in an infinite two-dimensional array of this pin cell type. Overall code speedup results for each of the optimizations relative to the OpenMC 0.6.1 nuclide energy grid reference case are given in Table 2. The additional memory required for each optimization is also given. As a point of reference, the baseline total memory requirement of the nuclear data needed for the simulation of this model, without any optimization, is 35.6 MB. Noticeable speedup is observed for each of the optimizations. Relative to the nuclide grid reference case, each optimization speeds up the code, on an overall basis, by at least a factor of 1.1, with the material grid run seeing a speedup of more than 1.3. Kinematics-constrained nuclide grid searches achieve a speedup of greater than 1.1 with negligible memory requirements. Logarithmic energy hashing of the nuclide grids gives a 1.27 speedup with a quite-acceptable memory addition of only 1.3 MB.

Table 1: Fresh Fuel Pin Cell Materials

Material	Nuclides	Unionized Energy Points
Fuel	5	219458
Gap	1	503
Clad	25	362674
Moderator	6	6132

Table 2: Speedup and Memory for Fresh Fuel Pin Cell Simulations

Unionization	Kinematics	Hashing	Speedup	Added Memory [MB]
nuclide	—	—	1.00	0.0
material	—	—	1.34	45.5
global	—	—	1.14	81.1
nuclide	true	—	1.13	$2.6 \cdot 10^{-4}$
global	true	—	1.28	81.1
nuclide	—	lethargy	1.27	1.3
global	—	lethargy	1.28	81.1
nuclide	—	energy	1.13	1.3
global	—	energy	1.23	81.1

3.2. Depleted Fuel Pin Cell

Next, we examine the speedup from applying each of the optimizations to a simulation of the same two-dimensional, infinite pin cell array as in the previous section, but with 900 K fuel at a burnup of

20.0 MWd/kg.⁹ The use of depleted fuel, which contains fission products and actinides that are not present in fresh fuel, results in the inclusion of many more nuclides in this model. This can be seen in the summary of materials contained in Table 3. The speedup values for the depleted fuel pin cell are given in Table 4. For this case, the optimizations provide an across-the-board improvement relative to the fresh fuel speedup results. The unionized material grid optimization gives a speedup value of more than 1.6. This higher value is likely due to the large discrepancy between the number of nuclides contained in the fuel and any of the other materials. When the neutron is traversing non-fuel materials, there are runtime savings to be had by not searching over the energy grids of the 100+ nuclides contained in the fuel. Kinematics-constrained searching provides a 1.11 speedup at a negligible memory cost. The memory values reported in Table 4 are in addition to a baseline total nuclear data memory requirement of 136.6 MB.

Table 3: Depleted Fuel Pin Cell Materials

Material	Nuclides	Unionized Energy Points
Fuel	107	1121483
Gap	1	503
Clad	25	362674
Moderator	6	6132

Table 4: Speedup and Memory for Depleted Pin Cell Simulations

Unionization	Kinematics	Hashing	Speedup	Added Memory [MB]
nuclide	—	—	1.00	0.0
material	—	—	1.61	528.3
global	—	—	1.23	810.5
nuclide	true	—	1.11	$1.1 \cdot 10^{-3}$
global	true	—	1.26	810.5
nuclide	—	lethargy	1.28	5.5
global	—	lethargy	1.29	810.6
nuclide	—	energy	1.14	5.5
global	—	energy	1.24	810.6

3.3. Depleted Full Core

Here, in order to see the effect of model complexity on the attainable speedup with each of the optimizations, we take a look at a three-dimensional, full core BEAVRS model that has been depleted out to 20.0 MWd/kg.⁹ The materials contained in this detailed model are described in Table 5. The baseline total nuclear data memory requirement is 170.3 MB. Speedup values are shown in Table 6. The same three trends, previously discussed, of the material union grid method performing well due to widely varying material composition, kinematics-constrained searches offering a bump in simulation rates for essentially no additional memory, and logarithmic hashing of nuclide energy grids providing appreciable speedup for a reasonable additional memory requirement are observed in the full core simulation. Speedup values are

⁹Nuclide densities for the fuel in this simulation are obtained by running a depletion calculation for the same model with the CASMO-5 lattice physics code [27].

slightly reduced, though, relative to the depleted pin cell case. This is because, in the more complex full core model, relatively more runtime is spent performing operations unrelated to energy grid lookups such as those related to the tracking of particles across geometric regions.

Table 5: Depleted Fuel Full Core Materials

Material	Nuclides	Unionized Energy Points
1.6% Enriched Fuel	108	1130727
2.4% Enriched Fuel	108	1130727
3.1% Enriched Fuel	107	1121483
Gap	1	503
Clad	25	362674
Moderator	4	3442
Stainless Steel	17	304501
Carbon Steel	21	291360
Inconel	17	304501
Control Rod	12	252553
Borosilicate	8	23663
Air	8	29061

Table 6: Speedup and Memory for Depleted Full Core Simulations

Unionization	Kinematics	Hashing	Speedup	Added Memory [MB]
nuclide	—	—	1.00	0.0
material	—	—	1.53	1612.6
global	—	—	1.20	1308.0
nuclide	true	—	1.10	$1.4 \cdot 10^{-3}$
global	true	—	1.22	1308.0
nuclide	—	lethargy	1.24	6.9
global	—	lethargy	1.20	1308.0
nuclide	—	energy	1.11	6.9
global	—	energy	1.20	1308.0

3.4. Fast Spectrum Metal Sphere

Finally, we examine a model consisting of a single sphere, with a uniform, fixed 14.1 MeV neutron source and vacuum boundary condition, composed of equal parts lead, gold, and mercury. This system is constructed to exhibit a fast neutron energy spectrum, in contrast to the thermal spectra of the pin cell and full core systems, so that the relative merits of the optimization schemes can be compared across different neutron spectra. The composition of the model’s single material is briefly summarized in Table 7 and the observed speedup values are given in Table 8. The baseline total nuclear data memory requirement is 10.9 MB for this model. No tallies are recorded in this fixed-source simulation.

Again, we see the unionized material grid speed up the simulation more than the other methods, except, in this case, the global union scheme. Because we are dealing with a single material for the entire model,

the material and global union grid methods are functionally identical. All methods result in a speedup by a factor of at least 1.3 relative to the individual nuclide grid reference case. Also, possibly owing to the harder spectrum of this model and a resultant flux that does not exhibit as definite of a $1/E$ form as a more moderated system might, the constant energy hashing scheme outperforms that with constant lethargy spacing.

Table 7: Pb-Au-Hg Sphere Materials

Material	Nuclides	Unionized Energy Points
Pb-Au-Hg	12	122075

Table 8: Speedup and Memory for Pb-Au-Hg Sphere Simulations

Unionization	Kinematics	Hashing	Speedup	Added Memory [MB]
nuclide	—	—	1.00	0.0
material	—	—	1.74	6.8
global	—	—	1.72	6.8
nuclide	true	—	1.33	$9.6 \cdot 10^{-5}$
global	true	—	1.74	6.8
nuclide	—	lethargy	1.38	0.5
global	—	lethargy	1.82	6.9
nuclide	—	energy	1.59	0.5
global	—	energy	1.82	6.9

3.5. Multithreading

The optimizations are implemented to be compatible with the shared memory parallelism capabilities of OpenMC via multithreading with the OpenMP application programming interface, release 3.1 [28], as implemented in GFortran 4.9.0. In order to gauge the performance of the optimizations in multithreaded applications, additional simulations of the depleted fuel pin cell and lead-gold-mercury sphere models are run utilizing both 4 and 8 threads on the same dual Intel Xeon E5620 2.4 GHz quad-core processors. The speedup values for each of the optimizations — relative to the corresponding serial runs already discussed and also the multithreaded nuclide energy grid runs — that are observed in pin cell and metal sphere simulations are found in Tables 9 and 10, respectively.

The threaded pin cell simulations show minimal overhead at 4 threads relative to the serial runs — no more than a few percent. Moving to 8 threads, overhead is still manageable, typically less than 10%. The performance in multithreaded fast spectrum sphere simulations is somewhat reduced with overhead at around 10% with 4 threads. For 8 threads, overhead values hover around 20%. It is worth noting, though, that, in each of the multithreaded cases, the overhead incurred when optimization methods are employed is comparable to the overhead in the default nuclide energy grid reference case. In other words, the optimizations do not significantly degrade multithreaded performance. This is illustrated further in the speedup values for each optimization relative to the multithreaded nuclide energy grid reference. These speedup results, given in the final column of both Tables 9 and 10, are comparable in magnitude to the corresponding serial speedup values. Concordantly, the multithreaded optimization speedup results, relative to one another, are largely unchanged from the serial cases.

Table 9: Multithreaded Speedup for Depleted Pin Cell Simulations

Unionization	Kinematics	Hashing	4 Threads ¹⁰	8 Threads ¹⁰	8 Threads ¹¹ (Nuclide Reference)
nuclide	—	—	3.84	7.41	1.00
material	—	—	3.79	7.24	1.58
global	—	—	4.16	7.48	1.24
nuclide	true	—	3.90	7.48	1.12
global	true	—	4.20	7.52	1.28
nuclide	—	lethargy	3.89	7.48	1.29
global	—	lethargy	4.04	7.42	1.29
nuclide	—	energy	3.93	7.57	1.16
global	—	energy	4.13	7.42	1.24

Table 10: Multithreaded Speedup for Pb-Au-Hg Sphere Simulations

Unionization	Kinematics	Hashing	4 Threads ¹⁰	8 Threads ¹⁰	8 Threads ¹¹ (Nuclide Reference)
nuclide	—	—	3.75	6.98	1.00
material	—	—	3.55	6.27	1.56
global	—	—	3.56	6.32	1.56
nuclide	true	—	3.61	6.60	1.26
global	true	—	3.49	6.17	1.54
nuclide	—	lethargy	3.63	6.64	1.31
global	—	lethargy	3.56	6.18	1.61
nuclide	—	energy	3.50	6.40	1.46
global	—	energy	3.54	6.17	1.61

4. Conclusions

The presented energy grid lookup algorithm optimizations are shown to result in non-negligible overall speedup of the OpenMC Monte Carlo neutron transport code in both serial and multithreaded simulations of practical systems. If the additional memory requirement is permissible, applying the unionized energy grid scheme on a material-by-material basis is a desirable alternative to the use of individual nuclide grids. Additionally, the material union grids offer performance that is superior to the global union grid for all of the systems considered here. The kinematic constraint on energy grid lookups offers non-negligible overall code speedup at negligible memory expense. Even for some of the most demanding depletion calculations with several hundred nuclides, this would amount to only a few kB. A trivial modification, the kinematic constraint is an attractive addition to the default energy grid lookup algorithm in pointwise continuous-energy Monte Carlo codes. The energy hash table methods also provide simple means of speeding up simulations that can be relatively cheap from a memory perspective. However, as the speedup with respect to the global unionized

¹⁰Speedup values are relative to the same optimization schemes run in serial.

¹¹Speedup values are relative to the single-nuclide energy grid binary search run with 8 threads.

grid runs is typically minimal, application of these methods should likely be restricted to simulations in which unique energy grids for each nuclide are used. With moderate levels of speedup observed with each of the proposed optimizations, individually, investigation of applying these techniques in concert with one another is warranted. Also, efforts to move away from pointwise energy grid data, altogether, as through direct cross section value calculations using the multipole resolved resonance formalism [29, 30], should receive continued attention.

5. Acknowledgements

This material is based upon work supported under the first author's appointment to a Department of Energy Nuclear Energy University Programs Graduate Fellowship. This research is partially supported by the Consortium for Advanced Simulation of Light Water Reactors (<http://www.cas1.gov>), an Energy Innovation Hub (<http://www.energy.gov/hubs>) for Modeling and Simulation of Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725.

References

- [1] A. Foderaro, *The Elements of Neutron Interaction Theory*, MIT Press, 1971.
- [2] A. Trkov, M. Herman, D. Brown, ENDF-6 Formats Manual, Technical Report BNL-90365-2009 Rev.2, National Nuclear Data Center, Brookhaven National Laboratory, 2012.
- [3] M. Chadwick, M. Herman, P. Obložinský, M. Dunn, Y. Danon, A. Kahler, D. Smith, B. Pritychenko, G. Arbanas, R. Arcilla, R. Brewer, D. Brown, R. Capote, A. Carlson, Y. Cho, H. Derrien, K. Guber, G. Hale, S. Hoblit, S. Holloway, T. Johnson, T. Kawano, B. Kiedrowski, H. Kim, S. Kunieda, N. Larson, L. Leal, J. Lestone, R. Little, E. McCutchan, R. MacFarlane, M. MacInnes, C. Mattoon, R. McKnight, S. Mughabghab, G. Nobre, G. Palmiotti, A. Palumbo, M. Pigni, V. Pronyaev, R. Sayer, A. Sonzogni, N. Summers, P. Talou, I. Thompson, A. Trkov, R. Vogt, S. van der Marck, A. Wallner, M. White, D. Wiarda, P. Young, ENDF/B-VII.1 nuclear data for science and technology: Cross sections, covariances, fission product yields and decay data, *Nucl. Data Sheets* 112 (2011) 2887–2996.
- [4] R. Macfarlane, D. Muir, *The NJOY Nuclear Data Processing System*, Version 91, Technical Report LA-12740-M, Los Alamos National Laboratory, 1994.
- [5] L. Carter, E. Cashwell, *Particle-Transport Simulation with the Monte Carlo Method*, TID-26607, ERDA Critical Review Series, U. S. Energy Research and Development Administration, Technical Information Center, Oak Ridge, TN, 1975.
- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, 2009.
- [7] T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. Cox, J. Durkee, J. Elson, M. Fensin, R. Forster, J. Hendricks, H. Hughes, R. Johns, B. Kiedrowski, R. Martz, S. Mashnik, G. McKinney, D. Pelowitz, R. Prael, J. Sweezy, L. Waters, T. Wilcox, T. Zukaitis, Initial MCNP6 release overview, *Nucl. Technol.* 180 (2012) 298–315.
- [8] T. Goorley, MCNP6.1.1 - Beta Release Notes, Technical Report LA-UR-14-24680, Los Alamos National Laboratory, 2014.
- [9] J. Leppanen, Two practical methods for unionized energy grid construction in continuous-energy Monte Carlo neutron transport calculation, *Ann. Nucl. Energy* 36.7 (2009) 878–885.
- [10] T. Sutton, F. Brown, F. Bischoff, D. MacMillan, C. Ellis, J. Ward, C. Ballinger, D. Kelly, L. Schindler, *The Physical Models and Statistical Procedures Used in the RACER Monte Carlo Code*, Technical Report KAPL-4840, Knolls Atomic Power Laboratory, 1999.
- [11] N. Candelore, R. Gast, L. Ondis, RCP01- A Monte Carlo Program for Solving Neutron and Photon Transport Problems in Three-Dimensional Geometry with Detailed Energy Description, Technical Report WAPD-TM-1267, Bettis Atomic Power Laboratory, 1978.
- [12] J. Leppanen, *Serpent – a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code*, Technical Report, VTT Technical Research Centre of Finland, 2013.
- [13] W. R. Martin, Challenges and prospects for whole-core Monte Carlo analysis, *Nucl. Eng. Technol.* 44 (2012) 151–160.
- [14] T. Trumbull, Treatment of nuclear data for transport problems containing detailed temperature distributions, *Nucl. Technol.* 156 (2006) 75–86.
- [15] Y. Liu, D. She, K. Wang, G. Yu, Optimization treatment of point-wise nuclear data in Monte Carlo criticality and burnup calculations, *Ann. Nucl. Energy* 38 (2011) 1489–1495.
- [16] F. B. Brown, New hash-based energy lookup algorithm for Monte Carlo codes, *Trans. Am. Nucl. Soc.* 111 (2014) 659–662.
- [17] A. R. Siegel, K. Smith, P. K. Romano, B. Forget, K. G. Felker, Multi-core performance studies of a Monte Carlo neutron transport code, *Int. J. High Perform. Comput. Appl.* 28 (2014) 87–96.
- [18] P. Romano, B. Forget, *The OpenMC Monte Carlo particle transport code*, *Ann. Nucl. Energy* 51 (2013) 274–281.
- [19] G. I. Bell, S. Glasstone, *Nuclear Reactor Theory*, Division of Technical Information, United States Atomic Energy Commission, 1970.
- [20] X-5 Monte Carlo Team, MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Technical Report LA-UR-03-1987, Los Alamos National Laboratory, 2008.
- [21] D. Cullen, C. Weisbin, Exact Doppler broadening of tabulated cross sections, *Nucl. Sci. Eng.* 60.3 (1976) 199–229.

- 1
2
3
4 [22] N. Horelik, B. Herman, B. Forget, K. Smith, Benchmark for evaluation and validation of reactor simulations (BEAVRS),
5 in: Proc. Int. Conf. on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C
6 2013).
7 [23] J. L. Conlin, D. K. Parsons, S. J. Gardiner, M. Gray, A. Kahler, M. B. Lee, M. C. White, Continuous Energy Neutron Cross
8 Section Data Tables Based upon ENDF/B-VII.1, Technical Report LA-UR-13-20137, Los Alamos National Laboratory,
9 2013.
10 [24] J. L. Conlin, D. K. Parsons, Release of Continuous Representation for $S(\alpha, \beta)$ ACE Data, Technical Report LA-UR-14-
11 21878, Los Alamos National Laboratory, 2014.
12 [25] OpenMC Monte Carlo particle transport code, 2014. <https://github.com/mit-crpg/openmc>.
13 [26] GNU Fortran, 2014. <https://gcc.gnu.org/fortran/>.
14 [27] J. Rhodes, K. Smith, D. Lee, CASMO-5 development and applications, in: PHYSOR-2006.
15 [28] OpenMP Architecture Review Board, OpenMP Application Program Interface Version 3.1, Technical Report, 2011.
16 [29] R. Hwang, A rigorous pole representation of multilevel cross sections and its practical applications, Nucl. Sci. Eng. 96
17 (1987) 192–209.
18 [30] B. Forget, S. Xu, K. Smith, Direct Doppler broadening in Monte Carlo simulations using the multipole representation,
19 Ann. Nucl. Energy 64 (2014) 78–85.
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65