LA-UR-14-24530
Approved for public release; distribution is unlimited.

Title: New Hash-based Energy Lookup Algorithm for Monte Carlo Codes

Author(s): Brown, Forrest B.

Intended for: American Nuclear Society, 2014 Winter Meeting, 2014-11-09/2014-11-13 (Anaheim, California, United States)

Issued: 2014-06-18

New Hash-based Energy Lookup Algorithm for Monte Carlo Codes

Forrest B. Brown

Los Alamos National Laboratory, Los Alamos, NM, fbrown@lanl.gov

INTRODUCTION

This note provides a new algorithm for energy lookups during the construction of material cross-sections in a continuous-energy Monte Carlo code. A new hash-based energy lookup algorithm provides speedups of 15-20x over conventional schemes and requires about 1,000x less memory than unified grid methods. The hashing scheme is based on a log-energy grid and provides search bounds for each isotope that greatly reduce the lengths of energy table searches. It should be useful to code developers for optimizing the performance of any Monte Carlo code for particle transport.

Problem Statement

In the Monte Carlo simulation of neutrons, the macroscopic total cross-section must be constructed for the material in the current spatial region containing the neutron. This must be done after every change in energy (from collisions) or material (from boundary crossings). Typically, these cross-sections are not precomputed due to limitations on memory space, but are computed on-the-fly from the constituent isotopic cross-section data. For problems with 100s of isotopes in each material, the construction of the total material cross-section has been observed to take about 33% or more of overall computing time for MCNP [1,2] and 85% of the time for OpenMC [3]. The first and most time-consuming part of this construction involves searching the energy table for each isotope's cross-section data to determine the energy bin. The energy table for an isotope typically has 10^2-10^6 bins, depending on the variation in the cross-sections, number of resonances, threshold reactions, etc. Each isotope has a different energy table. Because of the generality and size of the energy tables, binary searches are always used. (Binary searches scale as *O(logN)*, while linear searches scale as *O(N)*, where *N* is table size.). Typically, 100s of binary searches in tables of length 10^2-10^6 must be performed after every collision or material boundary crossing for every neutron, requiring $\sim 10^{14}$ binary searches for a billion-neutron simulation.

Background

Selecting algorithms for a code usually involves balancing space (memory storage) and time (fewer operations). The fastest algorithms usually require large amounts of data to be precomputed and stored; the slower ones usually compute or search repeatedly to save memory space.

MCNP has always used the repeated, time-consuming binary search of all isotopic energy tables to save memory and preserve full accuracy of the cross-section data tables. In the 1980s, some codes (e.g., RACER [4]) made use of a unified energy grid to attain speedups and aid vectorization. All isotopic cross-section data was interpolated and averaged onto a single energy table with 10^4-10^5 bins. That scheme required only 1 binary search for a material, rather than 1 for every isotope in the material. Speedups were impressive, 20-50x faster than MCNP, due to fewer searches and to vectorization, but the cross-section remapping in energy introduced approximations and greatly increased memory requirements. The unified energy grid could only be used on problems with small to moderate numbers of isotopes per material, or few materials. Later, as problem sizes grew and Monte Carlo depletion became routine, the unified energy scheme was replaced by on-the-fly computation. In the early 2000s, Serpent [5,6] used a unified energy grid, with all material cross-sections precomputed. That provided significant speedups, but limited problem size. To preserve accuracy (by not remapping the cross-sections to fewer energy bins), 5-10 M energy bins are required. To avoid the huge memory penalty for interpolating and storing all cross-section data onto such a large energy grid, the unified grid scheme was enhanced to a "unified grid with pointers" scheme, where only a single index for each isotope and unified energy bin was stored. After 1 energy lookup per material, the indexes for each isotope could be used to retrieve and interpolate isotopic cross-sections on-the-fly. While only slightly slower than the previous scheme, the memory saving was significant. Still, 5-10 M indexes must be stored for each isotope in the problem, requiring 10-20 GB of memory for depletion problems with 300 isotopes.

HASH-BASED ENERGY LOOKUP ALGORITHM

The new algorithm for energy lookups during material cross-section construction in MCNP is called a "hash-based energy lookup." It was suggested by G. Zimmerman [7], but turns out to be identical to an energy lookup scheme used in the 1980s [8], and probably in other codes before that.

The setup portion of the algorithm, performed prior to neutron random walks, involves the following steps:

1. Determine the minimum and maximum energy bounds for the problem, by checking each of the isotopic cross-section datasets in the problem. These will be denoted E_{min} and E_{max}.
2. Choose a value for M, the number of bins in a single energy grid with uniform spacing in $ln(E)$ between E_{min} and E_{max}. This energy grid will be called the *ugrid*.
3. For each bin in the *ugrid*, determine and store for each isotope the bounding indexes $k_1(u,i)$ and $k_2(u,i)$ in the energy table for that isotope, where i is the isotope index and u is the index in the *ugrid*.

The *ugrid* is completely defined by M, E_{min} , and E_{max}, and need not be explicitly stored. Also, because $k_2(u,i) = k_1(u+1,i)+1$, the $k_2(u,i)$ values need not be explicitly stored. The default value for M in MCNP6 was chosen after some experimentation to be 8,000. This requires total memory storage of $(M+1)I$ values, where I is the number of isotopes in the problem. For M=8000 and 250 isotopes, the total memory requirement is only 8 MB. Larger values for M give improved speed, but the dependence is weak for M greater than a few 1000. Smaller values of M reduce the speedup but also reduce the memory requirements. The value chosen for M does not in any way affect accuracy of the cross-section data. The k_1 and k_2 indexes for each isotope for each of the *ugrid* bins are simply the bounds for performing ordinary binary searches in the isotope datasets. These bounds narrow the range of the binary searches, so that only a small portion of the isotopic energy table need be searched.

During the neutron simulation, a simple hash function is used to determine the location of a neutron's energy in the *ugrid*. For that bin in the *ugrid*, the k_1 and k_2 indexes for each isotope are retrieved and binary searches are performed in the isotope energy tables in the normal manner for each isotope. The difference from the previous energy lookup scheme is simply that the hash function, *ugrid*, and k_1 and k_2 indexes greatly narrow the search regime to a few isotopic energy table entries, rather than a more costly search of the entire tables.

Defining $u_{min}=ln(E_{min})$, $u_{max}=ln(E_{max})$, and $du=M/(u_{max} - u_{min})$, the new algorithm for energy lookups for neutron energy E is:

1. $u = 1 + \lfloor du \cdot (ln(E)-u_{min}) \rfloor$, where $\lfloor\ \rfloor$ denotes truncation to the next lowest integer.
2. For each isotope i, search its energy table between entries $k_1(u,i)$ and $k_2(u,i)$.

It should be noted that frequently the search range in the isotopic energy tables is small, often less than 8. For such small ranges, a simple linear search scheme will be slightly faster than a binary search and may provide additional small speedups.

COMPUTATIONAL TIMING RESULTS

Two tests were carried out to demonstrate the speedups attained by the new hash-based energy lookup algorithm. The first involved stand-alone coding focused solely on measuring the timing for the energy lookups, while the second involved overall timing measurements for MCNP6 using different energy lookup schemes.

Timing results for energy lookup

The first test involved stand-alone coding to compare 3 methods: the conventional scheme used by MCNP6 with an external function for performing binary searches, the conventional scheme used by MCNP5 with explicit inline coding for the binary searches, and the new hash-based scheme with inline binary searches. The energy tables for 9 isotopes from the ENDF/B-VII.1 nuclear data libraries were used in the comparisons: 1001.80c, 8016.80c, 26056.80c, 92235.80c, 92238.80c, 94239.80c, 94240.80c, 94241.80c, 6000.80c. These isotopes had energy table sizes ranging from 590 to 157,744. For each energy lookup scheme, many millions of neutron energies were randomly sampled in the *ugrid* range, and then the energy lookups were performed for all 9 isotopes. Results are given in **Table 1**.

From the timing results in Table 1, it can be seen that ordinary coding optimization (inlining the binary searches) can provide a speedup of only 10-20%, while the use of the new hash-based algorithm gives speedups of 15-20x.

A number of variations on the testing were carried out. To see the effect of different choices of M, for example, values of 64k, 32k, 16k, 8k, 4k, and 1k were used resulting in MacBook timings of 2, 2, 2, 3, 3, and 5 ns/lookup, respectively. For MCNP6.1.1, M=8192 was chosen as the default. To see the effect of a mixed search scheme, after the hash-based lookup of bounds, a linear search was used when table intervals were 8 or fewer and a binary search when table intervals were greater than 8. For the stand-alone Mac timing tests, there was no improvement in speed, so this mixed search scheme was not implemented in MCNP6.1.1. In all of the stand-alone testing, all lookup results were checked to verify that they were exactly the same as those for the conventional lookup scheme.

Table 1. Timing results for stand-alone test of energy lookup methods. Results are the average time for each energy lookup.

	Mac Pro, 3 GHz Xeon, 667 MHz mem	MacBook, 3 GHz i7, 1600 MHz mem
MCNP6.1 energy lookup, with external binary search function	**97 ns**	**67 ns**
MCNP5 energy lookup, with explicit inline binary search coding	**81 ns**	**57 ns**
New hash-based energy lookup, with explicit inline binary search coding	**6 ns**	**3 ns**

It is also interesting to note the timings for 2 different systems: While the Mac Pro and MacBook used for testing had the same cpu speed, the speed of the memory chips gave significantly different timing results. This is due to the nature of the energy lookup scheme – it is primarily energy retrieval from memory and comparisons, with no arithmetic operations. In addition, the new algorithm is more cache-friendly than the conventional scheme: there are fewer memory references, and those references are not as widely dispersed as for the conventional scheme.

Timing results for MCNP6

It has been reported that MCNP6.1, the initial release of MCNP6 in 2013, runs significantly slower than MCNP5 [9]. A number of routine coding optimizations were made to MCNP6.1 and incorporated into the MCNP6.1.1 Beta release for 2014. The new hash-based energy lookup algorithm was then included into MCNP6.1.1 as the final step in the initial code optimization work. **Table 2** presents the speedups in MCNP6.1.1 due solely to the new energy lookup algorithm.

CONCLUSIONS

While the new hash-based energy lookup algorithm by itself is 15-20x faster than the conventional scheme, it is only a portion of the overall work and computations performed by MCNP. The overall speedups due to the new energy lookup algorithm will vary, depending on the particular physics and geometry of each problem. Problems with more collisions per history and more isotopes per material will show larger speedups; problems with more tallies, more complicated geometry, or fewer isotopes per material will show smaller or no speedups.

Table 2. MCNP6.1.1 speedups due to new hash-based energy lookup algorithm.

Problem	Overall Code Speedup
OECD Performance Benchmark, 3D PWR, 60 isotopes, no tallies	**1.2x**
BAW XI(2), ICSBEP Problem LEU-COMP-THERM-008, Case-2, 31 isotopes, no tallies	**1.2x**
Godiva problem, 2 isotopes, no tallies	**1.0x**
Godiva problem, with trace amounts of 421 other isotopes	**1.9x**
Reactor pin cell with 147 isotopes	**1.5x**
Porosity tool for well-logging, 5 isotopes	**1.0x**

A particular advantage of the new hash-based energy lookup algorithm over unified energy grid schemes is the very significant reduction in memory requirements – MBs instead of GBs. Such significant memory savings are important for future generations of computers that are expected to have very many more processors, but less memory per processor.

The new hash-based energy lookup algorithm has been incorporated into the MCNP6.1.1 Beta release in 2014. Currently, the new lookup algorithm is used only for energy lookups in neutron data tables. Future work will investigate whether the new algorithm can be effective for other problems (e.g., photons, light ions).

ACKNOWLEDGMENTS

Discussions of Monte Carlo algorithms and coding with George Zimmerman were, as always, stimulating and encouraging, and contributed significantly to the success of this work.

This work was supported by the US DOE/NNSA Nuclear Criticality Safety Program and the Advanced Simulation & Computing Program.

REFERENCES

1. Forrest Brown, Brian Kiedrowski, Jeffrey Bull, "MCNP5-1.60 Release Notes", Los Alamos National Laboratory report, LA-UR-10-06235 (2010).
2. J.T. Goorley, et al., "Initial MCNP6 Release Overview - MCNP6 version 1.0," Los Alamos National Laboratory report, LA-UR-13-22934 (2013).
3. J.R. Tramm & A.R. Siegel, "Memory Bottlenecks and Memory Contention in Multi-Core Monte Carlo Transport Codes," Proceedings of SNA+MC 2013, Paris, France Oct 27-31 (2013).
4. F.B. Brown, "Present Status of Vectorized Monte Carlo," *Trans. Am. Nucl. Soc*. **55**, 323 (1987).
5. J. Leppänen, "Two practical methods for unionized energy grid construction in continuous-energy Monte Carlo neutron transport calculation," *Ann. Nucl. Energy*, 36 (2009).
6. J. Leppänen, A. Isotalo, "Burnup calculation methodology in the Serpent 2 Monte Carlo code," Proceedings of PHYSOR-2012, Knoxville, TN, Apr. 15-20 (2012).
7. G. Zimmerman, private communication to F.B. Brown (2013).
8. D. Austin, KAPL, private communication to F.B. Brown (~1989).
9. F.B. Brown, B.C. Kiedrowski, J.S. Bull, "Verification of MCNP6.1 and MCNP6.1.1 for Criticality Safety Applications," Los Alamos National Laboratory report, LA-UR-14-22480 (2014).