A Method for Simulating Real-Time Neutron Populations, Materials and Geometries Using the GEANT4 Monte Carlo Toolkit

L. Russell[1], G. Jonkmans[2] and A. Buijs[1]
[1] McMaster University, Hamilton, Ontario, Canada
(russellf@mcmaster.ca)
[2] Atomic Energy of Canada Limited, Chalk River, Ontario, Canada

Summary

GEANT4 is a Monte Carlo particle physics toolkit that simulates elementary particles moving through matter [1]. GEANT4 allows a population of neutrons to be tracked in a multiplying medium as the population *and* the medium evolve. However, the population must be artificially stabilized so that it neither explodes nor vanishes. We present a stabilization method where the simulation is divided into short time intervals and the population is renormalized at the end of each interval. This method was used with a simple sphere of U235 to calculate the effective neutron multiplication factor (k_{eff}) from the continuous evolution of the neutron population.

1. Introduction

Predicting the short-time behaviour of a nuclear reactor core is crucial to the control of the fission reaction and to the prevention and mitigation of nuclear accidents. The effectiveness of safety systems that control nuclear reactors is governed by the time dependence of the neutron population in the reactor core. Current predictions are often governed by point kinetics models and quasi-static diffusion calculations. These models suffer from the many approximations required to solve the analytical functions. We are investigating how modern stochastic Monte Carlo calculations can improve our understanding of transients over intervals on the order of seconds. The Monte Carlo calculations we have devised follow each individual neutron as it propagates through a multiplying medium.

GEANT4 (GEometry ANd Tracking 4) is a multipurpose particle physics toolkit that uses Monte Carlo methods to simulate the passage of particles through matter [1]. In this respect, GEANT4 is similar to the more established Monte Carlo code, MCNP (Monte Carlo N-Particle), which was developed at Los Alamos National Laboratory and has been used extensively in the nuclear industry [2]. GEANT4 offers the ability to model electromagnetic field interactions, optical photon propagation, particle decay and hadronic processes [3]. It also offers a wide range of particles including leptons (electrons and neutrinos), baryons (protons and neutrons), bosons (photons), mesons and nuclei. Additionally, the user may add their own physics processes and/or particle definitions using the C++ object-oriented structure of GEANT4. This flexibility and expandability, while beneficial, shifts some of the burden of code development onto the user. For example, the user is required to define any detectors, and to develop code to extract and save the necessary quantities from particle interactions in these detectors. This code is then compiled with the toolkit to create executable programs.

GEANT4 was originally developed by the high energy particle physics community, but the toolkit is equally applicable to the low energy neutron interactions found in conventional nuclear reactors. One area of application is the simulation of real-time neutron populations, where an entire population of neutrons is continuously tracked temporally and spatially over a specified time period (e.g. on the order of seconds). Additionally, the geometry and materials of the simulation can change over time allowing effects such as heating and feedback to be modelled. However, a neutron population in a multiplying medium must be artificially stabilized in time to ensure that it neither explodes nor vanishes. If the simulation environment is supercritical ($k_{eff} > 1$), the neutron population will grow quickly and slow down the simulation. Conversely, if the environment is subcritical ($k_{eff} < 1$), the population will shrink exponentially. The population must be large enough to preserve the spatial distribution of neutrons over time; otherwise, global properties such as k_{eff}, which should only be affected by changes in the composition or geometry of the simulation environment, will change with the source distribution. Thus, stabilization is necessary to model neutron populations in time, and one such stabilization method is presented below.

2. Methodology

The code for this method of population stabilization and real-time simulation was developed in C++ using the methods and classes provided by the GEANT4 toolkit. The following terminology is used to describe the flow of a simulation in GEANT4

- *Hit* – An interaction between a tracked particle and a nuclei or geometric boundary.
- *History* – A record of the movements and interactions of a particle from its inception to its loss by absorption, escape, or ending condition such as maximum run duration.
- *Primary* – An initial particle used to start an event.
- *Event* – Starting with n primaries, a collection of the histories of all particles including the primaries and any secondary particles they create.
- *Run* – A collection of independent events. This is the basic calculation executed in GEANT4.
- *Simulation* – The entire modelling process from start to finish, which may include multiple runs.

The following method divides the simulation in time into discrete intervals, each of which consists of a single run. Thus, a run will refer to both a collection of independent events and a discrete time interval in the following discussion.

2.1 Neutron Population Stabilization

To stabilize the neutron population in time, the simulation must first be broken into a number of temporal intervals (i.e. discrete runs). The duration of these runs depends on the strength of the multiplying medium; that is, the rate at which the neutron population changes due to production and loss of neutrons in the medium. For a neutron population of size N, the run duration, T, should be

$$\{ T \mid N(t_0 + T) \, \epsilon \, [0.5N(t_0), 2.0N(t_0)] \} \quad (1)$$

where t_0 is the time at the start of the run. Thus, the neutron population should neither increase nor decrease by more than a factor of two over a single run; this factor was chosen based on trial and error. This keeps the population within a manageable range and preserves the spatial distribution of the particles over multiple runs. Calculating the value of T is complicated for all but very simple geometries since the strength of the multiplying medium depends on both geometry and material composition. Therefore, trial and error is the easiest way to determine an appropriate value for T. Additionally, the value of T does not need to be constant over the course of the simulation, so it can be adjusted to accommodate changes to either the geometry or the materials.

Once a run duration has been specified, the simulation executes as a series of consecutive runs that are related to one another by two factors: the multiplication constant of the run, k_{run}, and the survivors of the previous run. The multiplication constant of the run is defined as

$$k_{run} = N(t_0 + T)/N(t_0) \tag{2}$$

which is *not* the eigenvalue k_{eff}; k_{run} depends on the duration of the run. This metric simply tracks the overall population change for a given run, while the survivors provide continuity between runs. The survivors are those neutrons that are still "alive" in the simulation at the end of a run, and these neutrons become the primaries for the next run. Thus, while the runs are independent, the survivors of one run continue in the next run at the same position, momentum, and age that they had at time $t_0 + T$. This allows for a continuous evolution of the neutron population.

However, in GEANT4, particles move in a series of steps, and the cumulative sum of these steps is the track of the particle. At the end of each step, the position, momentum and time of the track is updated. If the particle underwent an interaction, then it may be "killed" or its momentum may be adjusted to reflect changes in both momentum direction and kinetic energy. To stop the neutrons at the end of each run, a new physics process was added to limit the step size of the neutrons to the maximum distance they could travel before the run ended. This maximum step size, d_{max}, is calculated using

$$d_{max} = \left((t_0 + T) - t_{i-1}\right) v(t_{i-1}) \tag{3}$$

where t_{i-1} is the current time at the start of the step i, $t_0 + T$ is the time at the end of the run, and $v(t_{i-1})$ is the velocity of the neutron at t_{i-1}.

Since the neutron population changes by the factor k_{run} over one run, it must be renormalized at the start of the next run to keep the population within the range set in Equation 1. This is done by either deleting or duplicating survivors until the survivor population equals the initial population. For either case, a list of targets is created; that is, a list of survivors that will be duplicated or deleted. To preserve the randomness of the spatial and energy distributions of the survivors, the targets must be spread evenly over the entire list of survivors and each target should point to a unique survivor. The targets are chosen using the following algorithm: for M survivors, the i^{th} survivor is picked if

$$r < \frac{m}{M-i}\ ,\ \ i \in [0, M-1] \tag{4}$$

where r is a uniform random number between zero and one, and m is the number of targets that still need to be picked (the analysis of this algorithm is not shown in this paper) [4]. Then the targeted survivors are either deleted or duplicated.

Finally, the delayed neutrons also need to be saved. The delayed neutrons result from the decay of fission products and are born some time after the initial fission event. If a delayed neutron is born outside of the current run, then the time of its birth, as well as its position and momentum at birth, are recorded in the permanent delayed neutron list, which persists for the entirety of the simulation. At the start of a run, any delayed neutrons that will be born in that run are added to the survivors and removed from the delayed neutron list. Thus, the delayed neutrons occur at the time of their birth in the simulation. However, if the total simulation time is short, then the delayed neutrons will not have a chance to appear. To avoid this limitation, the simulation can be initialized from the source distribution of a previous simulation (both the survivor and delayed neutron lists) that was long enough to reach equilibrium in terms of delayed neutrons.

2.2 Code Validation

The code is currently being benchmarked against MCNP because MCNP is the industry reference for Monte Carlo reactor criticality calculations. The benchmark is being performed to gauge the agreement between GEANT4 and MCNP for criticality calculations of simple geometries. In both codes, ENDF/B-VI cross sections at 293.6K are used, and the simulations are initialized from an isometric, mono-energetic neutron point source with an initial neutron energy of 0.5 MeV. The two geometries being tested are a solid sphere of U235, and a homogenous sphere comprised of 10.7% natural uranium and 89.3% heavy water by mass. In both cases, the radius of the sphere is varied to get sub- and supercritical assemblies. These calculations and samples of the results will be discussed in Section 3.1.

3. Criticality Calculations

The criticality of a specific geometry and material composition is determined by calculating the average of the neutron production rate over the neutron loss rate. Since k_{eff} does not change over a run (neither the material or geometry can change), the rates of production and loss can both be multiplied by the duration of the run such that

$$k_{eff} = \frac{rate\ of\ production}{rate\ of\ loss}\left(\frac{T}{T}\right) = \frac{total\ produced}{total\ lost} \tag{5}$$

where total produced is the total number of neutrons produced over the run of length T, and likewise for the total loss. The figure below shows the results of the criticality calculations in GEANT4 over 1035 runs. Note that the total run time of the simulations is on the order of 25 μs, so any delayed neutrons born through fission will not have a chance to occur during the simulation. This condition was mirrored in MCNP by using only prompt neutrons in the criticality calculation.

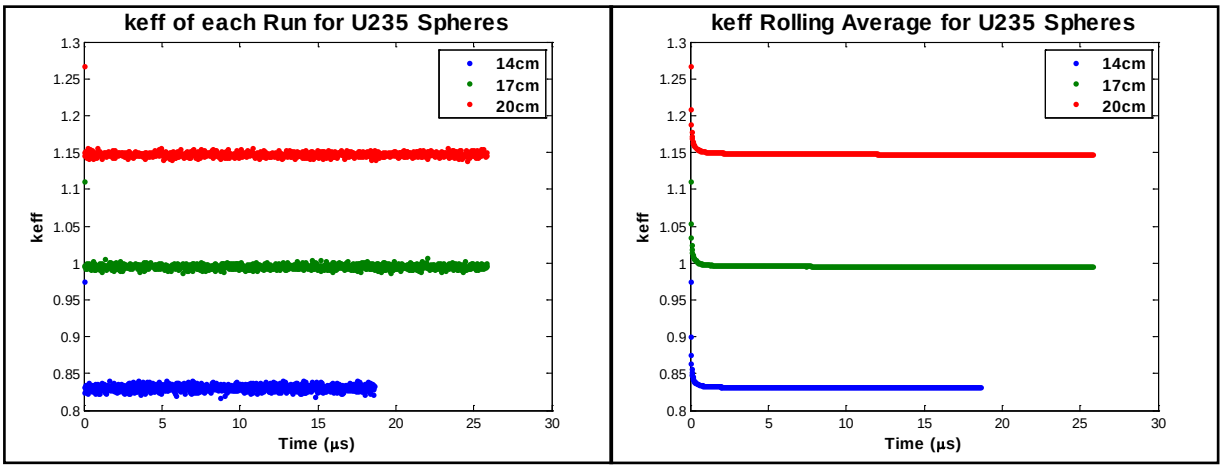

Figure 1 Estimates of k_{eff} for various diameters of U235 spheres at each run (left) and a rolling average for k_{eff} at each run (right).

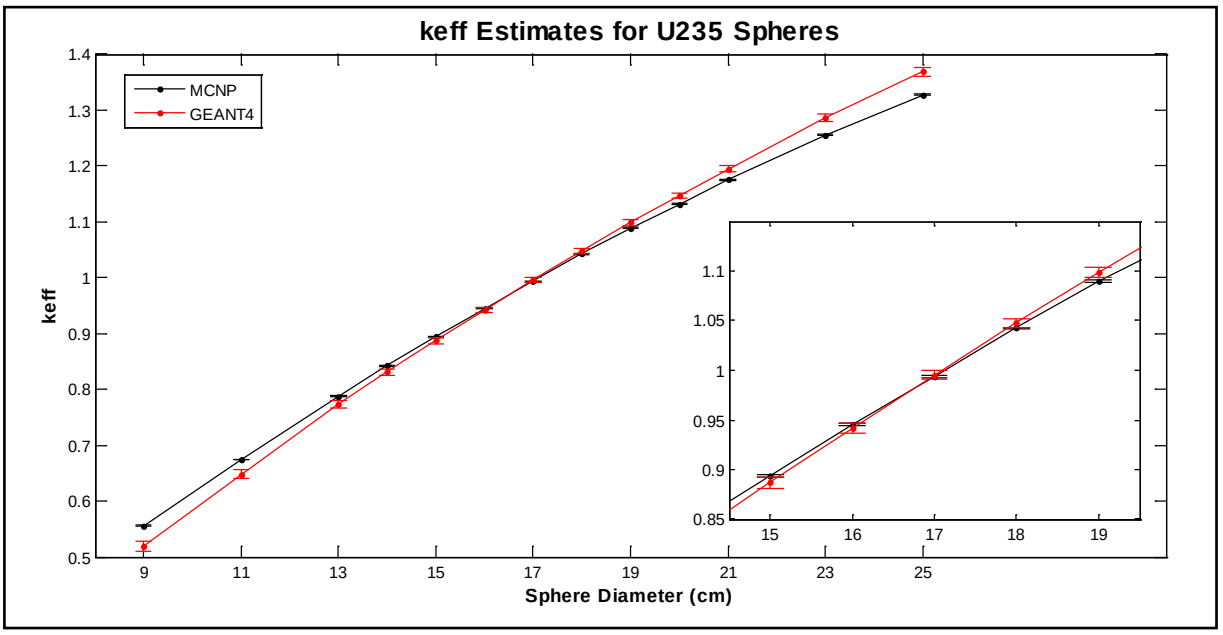

Figure 2 Estimates of k_{eff} for various diameters of U235 spheres in MCNP and GEANT4. The error bars show the 95% confidence interval of each estimate.

Figure 1 shows that the actual and average values of k_{eff} converge (within a region) after an initial period where the source distribution is converging. This indicates that the neutron population is stable in both time and space. Figure 2 shows that the GEANT4 estimates exhibit a bias relative to MCNP such that the bias increases as the system becomes increasingly more sub- or supercritical. Near $k_{eff} = 1$, the two codes agree within the 95% confidence intervals of each estimated k_{eff} value, which is shown in the inset of Figure 2.

Additionally, MCNP and GEANT4 have the closest agreement at approximately $k_{eff} = 1$ (~17cm), which indicates that both codes correctly recognize a critical system regardless of bias. However, outside of the range of $k_{eff} = 1 \pm 0.150$, the differences exceed the statistical uncertainty. The cause of this bias is unknown at this time, and work on an explanation is ongoing.

4. Conclusion

We have devised a Monte Carlo (stochastic) calculation that follows each individual neutron as it propagates through a geometry containing fissionable material. This method has the potential to predict how neutron populations in multiplying mediums evolve continuously with time. It was built using the GEANT4 toolkit, and benefits from the flexibility of GEANT4 in terms of access to the neutron information during the simulation. This makes it possible to track neutrons across multiple runs. This stabilization method can also be used for applications such as criticality calculations and real-time changes in either the geometry or material composition of the simulation. However, estimates for k_{eff} using GEANT4 show a bias relative to MCNP that increases with increasing sub- or supercriticality. Future applications of this code could include an investigation of the fuel temperature feedback mechanism by incrementally changing the fuel temperature between runs.

A day before submitting this paper, we found a technical report from Lawrence Livermore National Labs that described several of the algorithms and equations used in this paper [5]. While all of the work presented in this paper was performed without any knowledge of this technical report, it is important to note that some of fundamental concepts have been described previously in the literature.

5. References

[1] S. Agostinelli, et al, "GEANT4: A simulation toolkit," *Nucl. Instrum. Meth. A* 506, 2003, pp. 250.

[2] "MCNP – A General Monte Carlo N-Particle Transport Code – Version 5", Los Alamos National Laboratory, 2006. Retrieved online from http://mcnp-green.lanl.gov/.

[3] S. Agostinelli, et al, "Geant4 Physics Reference Manual", Conseil Européen pour la Recherche Nucléaire, 2010. Retrieved online from http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/PhysicsReferenceManual/fo/PhysicsReferenceManual.pdf

[4] K. Cronin, "Select a random N elements from List<T> in C#", Stack Overflow, 2008. Retrieved online from http://stackoverflow.com/questions/48087/select-a-random-n-elements-from-listt-in-c-sharp/48089#48089

[5] D. Cullen, et al, "Static and Dynamic Criticality: Are They Different?" Lawrence Livermore National Labs, 2003. Retrieved online from https://wci.llnl.gov/codes/tart/media/pdf/UCRL-TR-201506.pdf