LA-UR-12-25458

Approved for public release; distribution is unlimited.

Title: Integral Criticality Estimators in MCATK(U)

Author(s): Nolen, Steven D.
Adams, Terry R.
Sweezy, Jeremy Ed

Intended for: arXiv
Report
Web

Disclaimer:

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Integral Criticality Estimators in MCATK(U)

Steve Nolen, Terry Adams, and Jeremy Sweezy
Computational Physics Division, Monte Carlo Codes Group (XCP-3)
Los Alamos National Laboratory, Los Alamos, NM 87545
October 11, 2012

Abstract

The Monte Carlo Application Toolkit (MCATK) is a component-based software toolset for delivering customized particle transport solutions using the Monte Carlo method. Currently under development in the XCP-MCC group at Los Alamos National Laboratory, the toolkit has the ability to estimate the k_{eff} and α eigenvalues for static geometries. This paper presents a description of the estimators and variance reduction techniques available in the toolkit and includes a preview of those slated for future releases. Along with the description of the underlying algorithms is a description of the available user inputs for controlling the iterations. The paper concludes with a comparison of the MCATK results with those provided by analytic solutions. The results match within expected statistical uncertainties and demonstrate MCATK's usefulness in estimating these important quantities.

Introduction

The Monte Carlo Application Toolkit (or MCATK) is a set of parallel-aware libraries that provide a capability for simulating neutron particle behavior using the Monte Carlo technique. One common usage is the estimation of nuclear criticality for reactor design and safety assessments. The toolkit can provide estimates of the k -eigenvalue for simple spheres and for geometries modeled in the PARTISN defined lnk3dnt specification (Baker, Alcouffe, Dahl, Turner, & Ward, 2012). The purpose of this paper is to explain the methodology that MCATK employs including a description of relevant user required parameters, potential pitfalls, and future improvements under development.

In MCATK, the k -eigenvalue is estimated by comparing the magnitude of population change between one “generation” of neutrons and the next. As with other Monte Carlo codes, a “generation” is a computationally convenient grouping of all the neutrons that are sampled from the same fission source distribution. (Brown F. B., 2009) (Brown F. , 2005) This distinction is important since the fission source distribution evolves along with the eigenvalue estimate. As one generation of particles interacts with the system, their collisions define the fission distribution for the subsequent generation. When a particle undergoes a collision resulting in a fission event, MCATK samples isotope-specific distributions to get the number of neutrons

emerging from the fission and their initial energy and direction. Rather than being tracked immediately, these particles form the “next” generation, and are a sampling of the evolving fission source. The secondary neutrons from other reactions (i.e. $(n,2n)$, $(n,3n)$, etc.) are treated in same generation in which they are generated. In most cases, it is unnecessary to fully characterize the iterated fission source distribution because sampling from it happens in situ. Once the fission source is sufficiently converged, the relative size differences between the subsequent generations form an independent set of estimates for the system criticality. For the remainder of this paper, the term “iteration” refers to the tracking of a generation of neutrons that results in a new estimate for the k-eigenvalue and a new sampling of the fission source.

Settling

For an eigenvalue estimate to be a reasonable sampling of the *true* value requires that its generating population represents an unbiased sampling from the *true* fission source. If the population characteristics (or eigenvalue) indicate any systematic changes from generation to generation, it is symptomatic of sampling from a fission source that has not been adequately converged. In this context, the fission source can be considered “converged” if subsequent generations only exhibit statistical fluctuations from some population norm (i.e. the fission source). The associated eigenvalues produced by biased samplings are themselves biased and should not be expected to follow the central limit theorem. Unfortunately, it is impossible to determine *a priori* the number of iterations required to adequately converge the fission source. Several factors can impact the convergence rate including system attributes such as dimensions and composition, and the initial sampling used to initiate the iteration. For instance, the core of a power reactor is several meters across, and consequently, an initial guess of a spatially uniform fission source would converge faster than an initial source sampled from a point location at the geometric center. A problem’s material composition matters because each iteration continues until their last particle terminates (a.k.a. time-independent).

Rather than apply ad hoc heuristics for determining convergence, MCATK requires the user supply both the total number of iteration (N_{cycles}) and the number of “settle” iterations (N_{settle}). The user also supplies an initial guess for the fission source in the form of a source definition specifying the initial particles’ distribution in energy, space and angle. During the settling, no statistics are kept on the k_{eff} estimate as the fission source relaxes away from the bias induced by the initial guess. Once the iteration count exceeds the settle count, MCATK assumes any subsequent iteration provides an independent estimate of the eigenvalue and so collects and reports it, its average and its relative uncertainty. While trends in the k_{eff} average are obvious signs of convergence issues, recent work has shown that the absence of such structure is not sufficient evidence for assuming the convergence of the fission source (and confidence in the convergence of the k_{eff} estimate). (Cheatham & Brown, 2006)

Population Control

Once the user supplies the parameters for the iteration mechanics, they are then required to provide the desired number of Monte Carlo particles tracked during each iteration (N_{static}). As in other Monte Carlo transport packages, MCATK represents the true particle population with a limited set of pseudo-particles that are (ideally) a statistically valid sampling of the actual population. In the (common) event that the actual population is greater than the tracked population, reality and simulation are correlated through the assignment of a “weight” to the pseudo-particles such that the sum of their weights equals the expected population magnitude. Typical values for N_{static} are problem dependent, but usually are in the range of a few thousand to tens of thousands rather than millions. Regardless of the metric by which the user selects the value of N_{static} , the following section details how the toolkit maintains the population of tracked-particles at this level.

Due to the stochastic fluctuations in the number of sources generated within a cycle, one essential step in MCATK’s population control is to renormalize the weight of each generation prior to tracking. The initial cycle determines the normalization weight and assigns every starting particle a weight of unity. Before the tracking for the next cycle starts, the queued particles are assigned a common weight equal to the ratio of N_{static} and the last iteration population.

$$weight_i^{(n+1)} = \frac{N_{static}}{\sum_i^{N_{particles}^{(n)}} weight_i} \quad (1)$$

If the current particle count is greater than N_{static} , the smaller weights result in a proportionally smaller chance of creating fission sources. The renormalization also affects the weight based tallies like the scalar flux keeping those scores relative to the mean population, N_{static} . Naturally, the scaling preserves other aspects of the scalar flux estimate including the particle distributions in energy, space and angle.

One reason for the cycle fluctuations is that the number of neutrons released during an actual fission event is indeterminate and must be treated statistically. Rather than sampling the actual fission multiplicity which provides a discrete probability distribution for the number of emerging particles (Lestone, 2005), MCATK’s eigenvalue routines sample the *average* neutron multiplicity, $\bar{\nu}(E)$, which is tabulated in the data libraries read by MCATK as a floating point number. One straightforward approach for sampling this starts with computing the total weight emerging from fission as the multiple of the incident particle weight and $\bar{\nu}(E)$. This floating point result is converted to an integral value by adding a pseudo-random variate and then truncating the fractional part.

$$N_{fission} = \lfloor weight \cdot \bar{\nu} + \xi \rfloor \quad (2)$$

where

$\bar{\nu}$ is the average neutron multiplicity,

ξ is random variant selected on the closed interval $[0,1]$ and

$\lfloor x \rfloor$ denotes the floor function (or the integral part of x).

For most fissionable isotopes, the average neutron multiplicity may appear in the nuclear data files with and without the delayed neutrons. Since the presence of one or both of these is isotope and library dependent, MCATK requires the user specify which should be used for the eigenvalue search. The toolkit's default is to use the neutron multiplicity for the "prompt" neutrons only.

While using the average multiplicity mitigates the variance at the level of the collision, scaling the average multiplicity by the previous k_{eff} estimate reduces the variance across the entire cycle. MCATK regulates the outcome of a sampled fission event with a slight modification to Equation (2) that involves dividing the emerging count by the k_{eff} estimate.

$$N_{fission} = \left\lfloor \frac{weight \cdot \bar{\nu}}{\bar{k}_{eff}} + \xi \right\rfloor \quad (3)$$

where $\bar{k}_{eff}$ is usually the current averaged estimate for the eigenvalue. If the calculation is still in the settling phase (no statistics have been collected), MCATK uses the eigenvalue estimate from the previous cycle. In the case where the system is critical ($\bar{k}_{eff} = 1$), the effect is, of course, negligible as the neutron production rate via fission equals the loss rate through leakage and/or absorption. Away from unity, the accentuation (or attenuation) of the fission multiplicity acts to rebalance the source and sink terms leading to a relatively stable population of tracked particles. Although the mechanism does not exactly preserve N_{static} , the statistical fluctuation between cycles is normally safe to ignore.

k_{eff} Estimators

While it may seem intuitive to estimate k_{eff} by simply constructing the ratio between the number of neutrons created in one generation and the number generated in the previous generation, this ignores the obvious fact that a Monte Carlo simulation only tracks a finite number of particles and events. This is compounded by the reality that only a subset of particle events directly contribute to population changes and these are subject to their relative probability of occurrence. For a particle to contribute to the estimate, it must either collide with the material resulting in a fission (gain), collide resulting in a parasitic absorption (loss), or leak across an external boundary (loss). Even with a converged fission distribution from which to sample, the number and types of reactions sampled within an iteration are usually the dominate source of variability.

To illustrate this point, it is useful to look at how MCATK generates the track history for a particle. Using the particle's energy, MCATK first looks up the total microscopic cross section for each isotope in the surrounding material. It then combines the microscopic cross sections into a single macroscopic cross section weighting microscopic values by the relative atomic abundances.

$$\Sigma_t = \rho \cdot N_A \sum_j^{\text{\#isotopes}} \frac{f_j}{AW_j} \cdot \sigma_{t,j} \quad (4)$$

where

N_A is Avogadro's constant,

ρ is the material density in [gm/cc],

f_j is the mass fraction of the j th isotope,

AW_j is the atomic weight of the j th isotope,

$\sigma_{t,j}$ is the microscopic cross section for fission for the j th isotope.

This macroscopic quantity is the probability of colliding per unit path length. Based on this value, the code samples a distance to collision from an exponential distribution, and compares this to the distance along the particle's current trajectory to the nearest material or problem boundary.

$$dist = \frac{-\ln(\xi)}{\Sigma_t} \quad (5)$$

If this results in the particle moving to the boundary, it means the particle has streamed through the underlying material without affecting the population magnitude, and this scenario arises often especially when the mean free path is large compared to the scale lengths of the geometry. If the collision distance is the shortest, MCATK selects the target isotope and then selects the collision type. Since the goal is to populate the next generation from an adequately sampled fission source, the event handling is most productive if the final outcome is a sampling of a fission event. Naturally a non-fissile material will not produce the desired reaction, but even the material is fissile, the target isotope selected might still be non-fissile. The population may change because of absorption, but it precludes the opportunity to sample the fission source. Of course sampling a fissile target still may result in a non-fission event since the event selection is itself a random selection. All of these factors can make it difficult to sufficiently sample the fission source across multiple iterations, and nearly impossible to do so consistently.

Track Length Estimator

Rather than computing a contribution to an eigenvalue estimator only after sampling a fission, MCATK uses a "track length" estimator that scores each time a particle passes through a region with fissile material. (Urbatsch, Forster, Prael, & Beckman, 1995)

$$k_{tracklength} = \frac{1}{N^{(n)}} \cdot \sum_i^{\#Events} weight_i \cdot \bar{N} \cdot (dist_i \cdot \Sigma_f) \quad (6)$$

where

$N^{(n)}$ is the nominal source size of the n th generation,
 $dist_i$ is the distance the particle travels in the region,
 $\bar{N}$ is the *material* average neutron multiplicity¹, and
 Σ_f is the *macroscopic* cross section for fission² [cm^{-1}].

$$\bar{N}\Sigma_f = \rho \cdot N_A \sum_j^{\#isotopes} \frac{f_j}{AW_j} \cdot \bar{\nu}_j \sigma_{f,j} \quad (7)$$

where

N_A is Avogadro's constant,
 ρ is the material density in [gm/cc],
 f_j is the mass fraction of the j th isotope,
 AW_j is the atomic weight of the j th isotope,
 $\bar{\nu}_j$ is the average neutron multiplicity for the j th isotope, and
 $\sigma_{f,j}$ is the microscopic cross section for fission for the j th isotope.

The motivation for tallying in this fashion is exactly analogous to estimating any reaction rate using Monte Carlo. For any reaction, the *expected* reaction rate is the probability of occurring per unit path length (i.e. the macroscopic cross section) multiplied by total path length of the incident particle. The parenthesis in Equation (6) are provided to call out the estimation of the expected fission rate. For the purpose of estimating the size of the next generation, however, the fission reaction rate must be multiplied by the appropriate response which is the number of neutrons generated per fission. Although the track length estimator is the only k -eigenvalue tally currently available in MCATK, there are two other commonly used tallies that have been slated for inclusion in the future. They are described below for completeness.

¹ While unconventional, I have chosen $\bar{N}$ to represent the average neutron multiplicity for the material (i.e. composed of one or more isotopes) to avoid confusion with $\bar{\nu}$ which generally denotes the average neutron multiplicity of a *single* isotope.

² Unless otherwise noted, all cross sections and neutron multiplicity values are assumed to be energy dependent and are referenced at the energy of the incident particle.

Collision Estimator

Rather than computing a contribution to an eigenvalue estimator at *every* event, the collision estimator scores each time a particle collides with a fissile material.

$$k_{collision} = \frac{1}{N^{(n)}} \cdot \sum_i^{\#collisions} weight_i \cdot \bar{N} \frac{\Sigma_f}{\Sigma_t} \quad (8)$$

where $\bar{N}\Sigma_f$ and Σ_t are defined as above. Because the tallying occurs prior to sampling for the isotope involved in the collision, the presence of non-fissile isotopes within a material only affects the magnitude of the contribution. In fact, a non-fissile isotope with a large cross section will increase the frequency of the tally score while lowering expected contribution of each. As with the track-length based estimator, the probabilities are based on the composite *macroscopic* cross sections computed as in Equations (7) and (4).

Absorption Estimator

A third eigenvalue estimator usually termed the “absorption estimator” triggers when a particle collides resulting in fission.

$$k_{absorption} = \frac{1}{N^{(n)}} \sum_i^{\#fissions} weight_i \cdot \bar{v} \quad (9)$$

If all reactions are being treated explicitly, the estimator uses the entire particle weight but is consequently subject to infrequent scoring. If certain events are treated implicitly (see next section), then it triggers any time an event occurs that has been lumped with fission, but only a portion of the particle weight contributes. Unlike the previous estimators that use the macroscopic collision data for the material, the absorption estimator only scores after the target nuclide has been selected and only uses its nuclear properties.

Variance Reduction Techniques

In systems where the overall fraction of fission events to all others is vanishingly small, only the few fission events (if any) that are actually sampled determine the availability of fission sites from which to draw the next generation of particles. Not only does this result in large statistical fluctuations between generations, it also means the fission source is only weakly sampled particularly in space. To address these issues, MCATK contains a selection of variance reducing techniques with more coming as time allows.

Fission as Capture

One means for generating more samples from the fission source is to lump fission and capture events into an all-inclusive “absorption” event. In this technique, the fission cross section is added to the capture cross section increasing the probability of selection.

$$P_{absorption} = \frac{\sigma_f + \sigma_c}{\sigma_t} \quad (10)$$

where

σ_c is the microscopic capture cross section (i.e. (n, γ) , (n, p) , (n, α) , etc.).

To keep the outcome from the increased fission collisions un-biased, the entire weight of the particle is not used to sample the number of “next-generation” neutrons, but rather, the weight of the particle is first scaled by the ratio of the fission cross section to the lumped “absorption” cross section.

$$N_{fission} = \left\lfloor \frac{weight \cdot \bar{v}}{\bar{k}_{eff}} \left(\frac{\sigma_f}{\sigma_f + \sigma_c} \right) + \xi \right\rfloor \quad (11)$$

Lumping the fission and capture events increases the probability for sampling from the fission source while decreasing the number of neutrons generated from each event. The fission source is better sampled in the spatial domain without otherwise affecting its treatment.

Naturally this approach will also affect some of the previously described eigenvalue estimators. While employing fission as capture, the “absorption” estimator increments after each *absorption* event, and similar to the generation of the fission neutrons, the estimated contribution is reduced by the probability that the absorption event *could* have been fission.

$$k_{absorption} = \frac{1}{N^{(n)}} \sum_i^{\#absorptions} weight_i \cdot \bar{v} \left(\frac{\sigma_f}{\sigma_f + \sigma_c} \right) \quad (12)$$

The fission-as-capture approach is available in the MCATK library as an alternative formulation of the default collision kernel. Team intervention would be necessary to make this option generally available.

Implicit Capture

Although not currently available in MCATK, other variance reducing techniques, like implicit-capture, are slated for inclusion in future versions. Similar to fission-as-capture, the implicit-capture approach works by attempting to increase the number of fissions sampled within a generation. The basic premise is to continue tracking particles that would otherwise be lost due to parasitic capture. While parasitic captures are a critical component of the loss mechanism, the effect is sometimes better treated implicitly given the relatively high cost of allocating and deallocating storage associated with an explicit approach. To treat the capture implicitly, the technique reduces a particle’s weight rather than terminating it when it undergoes an otherwise terminal collision like fission or (n, γ) .

$$weight_{exit} = weight_{incident} \left(1 - \frac{\sigma_c}{\sigma_t} \right) \quad (13)$$

While preserving the correct absorption effect, the particle continues tracking retaining the potential for causing future fission events which result in a more thorough sampling of the fission source. Similar to the fission-as-capture method, the increase in fission events generated is offset by the corresponding decrease in the number of neutrons sampled from each. This effect is guaranteed by the shrinking particle weight's appearance in Equation (3).

Because implicit-capture creates the potential for tracking a number of low weight particles, it should be accompanied by a weight management scheme. Often a simple “weight window” scheme is sufficient. If the particle emerges from the collision with a weight below the “weigh cutoff”, a random sampling determines whether to terminate the particle or to increase its weight to a value within the window. Without the means to control the particle weights, the efficiency gained from sampling more fissions per cycle is quickly offset by the additional time required to follow low weight particles. Although they still undergo fission events, their decreased weight make it increasingly unlikely that such events will result in a sampling of the fission source.

If fission-as-capture is used jointly with implicit capture, the “absorption” estimator increments every time there is a collision, and the contribution is scaled by the ratio of the fission cross section to the total for the selected isotope.

$$k_{absorption} = \frac{1}{N^{(n)}} \sum_i^{\#collisions} weight_i \cdot \bar{v} \frac{\sigma_f}{\sigma_t} \quad (14)$$

The neutron weight emerging from the “absorption” event is then scaled by the combined cross sections for fission and absorption.

$$weight_{exit} = weight_{incident} \left(1 - \frac{\sigma_f + \sigma_c}{\sigma_t} \right) \quad (15)$$

Time Constant (α) and k_{eff}

Despite its ubiquitous usage as a measure of a system's criticality, k_{eff} is a quantity that can only be inferred from an experiment rather than actually measured. What is measurable is the reactor period, subcritical multiplication, die-away behavior, etc. and from these observable quantities one infers a system's proximity to criticality. Rather than delve into the semantics of k_{eff} 's interpretation and relevance near and away from unity, it is easier to discuss another related quantity known as the system's time constant. When introducing the time constant, it is customary to begin with the time-dependent, Boltzmann transport equation.

$$\begin{aligned}
\frac{1}{v} \frac{\partial \Psi}{\partial t} + \hat{\Omega} \cdot \nabla \Psi(\vec{r}, \hat{\Omega}, E, t) + \Sigma_t \Psi(\vec{r}, \hat{\Omega}, E, t) &= S(\vec{r}, \hat{\Omega}, E, t) + \\
\int_0^{4\pi} \int_0^\infty \Psi(\vec{r}, \hat{\Omega}', E', t) \Sigma_s(\vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}, E' \rightarrow E) dE' d\hat{\Omega}' &+ \\
\frac{\chi(E)}{4\pi} \int_0^{4\pi} \int_0^\infty \Psi(\vec{r}, \hat{\Omega}', E', t) \nu(E') \Sigma_f(\vec{r}, \hat{\Omega}', E') dE' d\hat{\Omega}' &
\end{aligned} \tag{16}$$

where

v is the neutron velocity,

$\Psi(\vec{r}, \hat{\Omega}, E, t)$ is the neutron angular flux density,

$S(\vec{r}, \hat{\Omega}, E, t)$ represents the external sources,

$\Sigma_s(\vec{r}, \hat{\Omega}' \rightarrow \hat{\Omega}, E' \rightarrow E)$ is the probability of a neutron at some direction, $\hat{\Omega}'$,

and energy, E' , scattering into the direction, $\hat{\Omega}$, at the energy, E , and

$\chi(E)$ is the probability a neutron emerges from fission with energy E .

Because this particular formulation does not account for the time varying behavior of the delayed neutron precursors, the average neutron multiplicity in (16) is understood to only account for the prompt neutrons. For this reason, MCATK will automatically switch to the “prompt” neutron multiplicity regardless of the user preference for the k -eigenvalue.

We next assume a separable form, $\Psi(\vec{r}, \hat{\Omega}, E, t) = \psi(\vec{r}, \hat{\Omega}, E) e^{\alpha t}$, where α is the aforementioned time constant, and focus on the first term yielding

$$\frac{1}{v} \left[\psi(\vec{r}, \hat{\Omega}, E) \frac{d}{dt} e^{\alpha t} + 0 \cdot e^{\alpha t} \right] = \frac{1}{v} \psi(\vec{r}, \hat{\Omega}, E) \alpha e^{\alpha t} . \tag{17}$$

By dividing through by $e^{\alpha t}$, combining like terms and introducing operator notation, we obtain

$$\mathbf{L} \psi + \left(\Sigma_t + \frac{\alpha}{v} \right) \psi = \mathbf{S} \psi + \mathbf{F} \psi , \tag{18}$$

where $\mathbf{L}$ is the leakage operator, $\mathbf{S}$ is the scattering operator, $\mathbf{F}$ is the fission operator, and we assume there are no external sources.

Recognizing Eq. (18) as a balance equation with the loss terms (i.e. leakage and collision) collected on the left-side and sources on the right leads to the interpretation of α/v as a “time loss” or “time collision”. The equation will become more familiar if we next combine the two collision terms into a pseudo-total interaction cross section.

$$\tilde{\Sigma}_t = \Sigma_t + \frac{\alpha}{v} \quad (19)$$

While mathematically obvious, the new term affects how MCATK computationally tracks the particles through a problem domain. In most situations, MCATK samples a distance to collision using Equation (5). If a particle moves through some region where $\Sigma_t = 0$ (i.e. void), MCATK simply moves the particle to the region's boundary and continues tracking. When substituting $\tilde{\Sigma}_t$ into Equation (5), the probability for a neutron colliding in a region becomes dependent on its velocity coupled with the current estimate for the time constant. Consequently, the sampling may legitimately yield a “time collision” event even in a void region. MCATK currently treats any “time collision” similarly to a nuclear capture terminating the particle track at the collision site although treating the capture implicitly is also reasonable. For this reason, the event is often referred to as “time absorption”.

By substituting (19) into (18) and multiplying the fission operator by $1/k'$, the resulting equation is finally recognizable as an eigenvalue equation in which k' is analogous to the k_{eff} discussed previously.

$$\mathbf{L}\psi + \tilde{\Sigma}_t\psi = \mathbf{S}\psi + \frac{1}{k'}\mathbf{F}\psi \quad (20)$$

As such, MCATK begins its search for α by first estimating the static k -eigenvalue. Starting with each new guess for α , MCATK cycles through a user specified number of generations arriving at an estimate for k' that includes the presence of the time absorption. If the value of k' is greater than unity, MCATK slightly increases the estimate for α , and repeats the process. Conversely, if the value of k' is less than unity, the estimate is tweaked downward reducing the loss to “time absorption”. Similar to the k_{eff} calculation, the user specifies the total number of iterations (“cycles”) with each iteration yielding a new estimate for the time constant. The notion of “settling” still applies to this algorithm because the initial guess for the fission source comes directly from a k -eigenvalue search, and as such, a settling phase allows the fission source to evolve away from its initial bias.

An important addition to the nominal k_{eff} calculation required by the α -eigenvalues is the specification of the number of “inner” iterations (N_{inners}). This number determines the number of individual generations to track while holding α constant before returning an estimate for k' . Because the first generation of an inner iteration is a sampling of the fission source from the last iteration of the previous inner, its associated estimate of k' will be biased and should be discarded. In practice, all but the last of the inner iterations are ignored with the last estimate of k' being reported rather than computing an average. In MCATK, the default number of inner iterations is 4, but the user can set the value higher which will increase the time to completion, but it will also result in a better estimate for k' .

Although this technique is analogous to a root-finding problem, it is not obvious what the correction to α should be for a given k' . MCATK's adjustment depends on the current k' estimate and on the *mean production time*(Λ).

$$\Lambda = \frac{1}{N^{(n)}} \sum_i^{\#collisions} \tau_i \cdot weight_i \cdot \bar{N} \frac{\sum_{f,i} \tau_{f,i}}{\sum_{t,i}} \quad (21)$$

where

τ is the time elapsed between a neutron's birth and the current event.

To start the iteration, MCATK computes the first time constant from the static k_{eff} calculation using (22). The production time contains a subscripted k_{eff} to differentiate it from those computed after applying the α/v term.

$$\alpha^{(0)} = a_{k_{eff}} = \frac{k_{eff} - 1}{\Lambda_{k_{eff}}} \quad (22)$$

The time constant adjustment is

$$\alpha^{(n+1)} = \alpha^{(n)} + (k' - 1) \left(\alpha^{(n)} + \frac{k'}{\Lambda^{(n)}} \right). \quad (23)$$

Because the process is analogous to root-finding, other methods from that field may be leveraged in the future such as the bisection or secant method.

Unfortunately, even a new method would face similar convergence difficulties as the one currently implemented. Ideally, a guess for a root would lead to a precise value that either indicates the guess was too high or too low. In this algorithm, the measure of any time constant's fitness is determined by the estimate of the intermediate k' -eigenvalue which is only known approximately and always includes an estimate of uncertainty. Once the code begins using estimates near the actual value, the sign of the correction (23) is expected to oscillate just as the value of k' oscillates above and below one. A user may lengthen the inner iterations to reduce the uncertainty surrounding k' , but some level of uncertainty will remain. This extra work will only be beneficial when the estimate for k' is oscillating about unity. As a failsafe, MCATK allows the user to specify an estimate for the time constant that will remain fixed allowing the code to focus on estimating the k' with a low uncertainty.

$\alpha < 0$ and Time Source

A serious issue with an algorithm based on Equation (20) arises when the system is sufficiently subcritical that the α/v term has larger magnitude than the total cross section. In that case, it can be argued that the collision term should be grouped with source terms. Depending on the spectra and the presence of the particle velocity in the denominator, the correction may only impact a

portion of the tracked particles. If the “time absorption” becomes “time creation”, MCATK is left with a quandary with respect to handling this source. Without knowledge of how the source should be handled in phase space, the most reasonable option is to increase the particle weight, but this means adding other features such as weight windows. Because of issues with negative time constants, MCATK does not attempt to estimate them. If MCATK determines the time constant is negative, it will “fixup” the current estimate setting it to 0.0. For now MCATK allows a user specified maximum number of alpha fixups (50 is the default) before saying the system is too near critical to predict an accurate alpha using this technique. The user determines whether such a message is fatal.

Results

Although MCATK routinely uses continuous energy cross section data, the verification process relies heavily on analytic sets, and the results from these appear first. (Sood, Forster, & Parsons, 2001) These sets collected by Sood, et al. are published with analytical solutions for k_{eff} for a set of simple geometries including spheres, infinite slabs and infinite cylinders. Keeping the data and the geometry simple provides a clearer picture of the algorithmic behavior. The data set labels are taken directly from the Sood paper with deviations being noted. Naturally, the data only notionally relates to actual nuclear data. The initial source for all configurations is a simple point source located at the origin emitting isotropically.

Analytic k_{∞}

The first set of test problems are for infinite medium geometries using the monoenergetic cross sections. These problems provide all the essential tracking elements for validating the eigenvalue routines without the complication of leakage terms. Some of the tracking components evident in the following results include parallel particle transport based on MPI and the use of reflective boundaries. In particular, the problem labels with “FC” denote runs where the “fission-as-capture” treatment was enabled. In all cases, the values for k_{eff} are well above one because the cross sections were developed for geometries where leakage is the dominant loss mechanism.

Because of the simplicity of the modeled systems, the iteration parameters in Table 1 are not generally recommended. In particular, the number of settles required for a realistic system is seldom this small. The geometry was modeled as a single radius sphere ($r=10cm$) with a reflective outer boundary.

Table 1- Iteration parameters for infinite medium eigenvalue runs

Parameter	Value
# Particles	10k
# cycles	5k
# settles	10

Table 2- MCATK k_∞ comparison in infinite media problems compared to analytic

Name	Analytic	MCATK	σ	$ k_\infty - k /\sigma$
PUa-1-0-IN	2.612903	2.613412	3.75e-4	1.36
PUB-1-0-IN	2.290323	2.289988	3.30e-4	1.01
PUB-1-0-IN(FC)	2.290323	2.290577	1.42e-4	1.79
Ua-1-0-IN	2.25	2.250356	3.19e-4	1.12
Ub-1-0-IN	2.330917	2.330583	3.33e-4	1.00
Uc-1-0-IN	2.256083	2.255968	3.20e-4	0.36
Ud-1-0-IN	2.232667	2.232201	3.17e-4	1.47

1-D Spherical, 2-D Cylindrical and 3-D Cartesian Geometry Eigenvalues

The next set of problems include a surface boundary through which particles are allowed to escape. Except for the spherical configuration, the other two geometries are *still* infinite medium problems having finite constraints on only one of their dimensions. The modeling was accomplished by creating meshes in the appropriate coordinate systems and then applying reflective boundary conditions as needed. An alternative approach available in MCATK is to create one dimensional versions of the mesh in each coordinate system.

Table 3 -- Iteration parameters for static k -eigenvalue runs.

Parameter	Value
# Particles	10k
# cycles	5k
# settles	100

Table 4 -- MCATK static k -eigenvalue results compared to analytic

Name	MCATK	σ	$ k_{exact} - k /\sigma$
PUa-1-0-SL	1.000078	1.35e-4	0.57
PUB-1-0-SL	0.999925	1.36e-4	0.55
PUB-1-0-CY	1.000094	1.30e-4	0.72
PUB-1-0-SP	0.999954	1.27e-4	0.36
PUB-1-0-SP(FC)	1.000036	1.19e-4	0.30

By tweaking the cross sections presented in Sood, et al., we can derive analytic time constants by reducing (or increasing) the provided capture cross section by a desired alpha. (Sood, 2008) We can then compare the resultant eigenvalue alpha against the analytic. As mentioned previously, negative time constants are currently unavailable in MCATK.

Table 5 -- Iteration parameters for time constant computations

Parameter	Value
# Particles	50k
# Outer cycles	250
# Outer settles	5
# Inner k' cycles	4
# Initial k settles	25
# Initial k cycles	50

Table 6 -- MCATK time constant comparison to analytic

Name	Analytic	MCATK	σ	$ \alpha_{exact} - \alpha /\sigma$
PUB-1-0-SP(A)	0.1	0.098968	1.481e-3	0.697
PUB-1-0-CY(A)	0.1	0.100470	1.520e-3	0.309
PUB-1-0-CY(A)	0.2	0.201130	1.566e-3	0.722

The **A** denotes that the cross sections associated with these Sood, et al. problems have been modified to generate positive time constants.

Conclusions

This paper describes the static eigenvalue solution techniques implemented in the Monte Carlo Application Toolkit (MCATK) and describes the relevant user parameters. The implementation of the algorithms has been validated with a suite of problems that have known analytic solutions. The MCATK results compare within statistical uncertainty both for k_{eff} and static time constant estimates.

Bibliography

- Baker, R. S., Alcouffe, R. E., Dahl, J. A., Turner, S. A., & Ward, R. C. (2012). PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System. *LA-UR-12-23848*.
- Betzler, B. R., Kiedrowski, B. C., Brown, F. B., & Martin, W. R. (2012). Calculating alpha Eigenvalues in a Continuous-Energy Infinite Medium with Monte Carlo. *LA-UR-12-24472*.
- Brockway, D., Soran, P., & Whalen, P. (1985). Monte-Carlo Eigenvalue Calculation. *Monte Carlo Methods and Applications in Neutronics, Photonics, and Statistical Physics Lecture Notes in Physics*, 240, 378-390.
- Brown, F. (2005). Fundamentals of Monte Carlo Particle Transport. *LA-UR-05-4983*.

- Brown, F. B. (2009). A Review of Best Practices for Monte Carlo Criticality Calculations. *LA-UR-09-03136*.
- Cheatham, J., & Brown, F. (2006). Investigation of Shannon Entropy for Characterizing Convergence of MCNP5 Eigenvalue Calculations. *LA-UR-06-5886*.
- Hill, T. R. (1983). Efficient Methods for Time Absorption (a) Eigenvalue Calculations. *LA-9602-MS*.
- Lestone, J. P. (2005). Energy and Isotope Dependence of Neutron Multiplicity Distributions. *LA-UR-05-0288*.
- Sood, A. (2008). *Private Communications*.
- Sood, A., Forster, R., & Parsons, D. (2001). Analytical Benchmark Test Set for Criticality Code Verification. *LA-UR-01-3082*.
- Urbatsch, T. J., Forster, R. A., Prael, R. E., & Beckman, R. (1995). Estimation and interpretation of k-eff confidence intervals in MCNP. *LA-12658*.

APPENDIX

Example k_{eff} Output

Cycle	k - eigenvalue			Radius of Gyration		
	Current	Average	Rel. Err.	Current	Average	Rel. Err.
1	1.356254			2.687144		
2	1.167966			3.622131		
3	1.064366			4.001714		
4	1.023844			4.158393		
5	0.990590			4.210557		
6	1.003634			4.263275		
7	1.008917			4.233720		
8	1.018805			4.213452		
9	1.003066			4.224177		
10	1.018615			4.256415		
11	0.999972			4.270043		
⋮				⋮		
44	1.002592			4.247583		
45	0.993565			4.216019		
46	1.023972			4.248474		
47	1.013760			4.257612		
48	0.997638			4.221598		
49	1.002875			4.245724		
50	1.009120			4.224563		
51	1.001836	1.001836	+/- 0.000000	4.224246	4.233404	+/- 0.002088
52	1.019767	1.010801	+/- 0.008806	4.247429	4.238079	+/- 0.001633
53	0.989310	1.007071	+/- 0.007081	4.290044	4.251070	+/- 0.003266
54	1.017373	1.005560	+/- 0.005695	4.258446	4.252545	+/- 0.002552
55	0.999513	1.007862	+/- 0.005171	4.266297	4.254837	+/- 0.002151
56	1.019372	1.003726	+/- 0.006019	4.264669	4.256242	+/- 0.001847
57	0.978915	1.004232	+/- 0.005234	4.198754	4.249056	+/- 0.002330
58	1.007769	1.003478	+/- 0.004680	4.254543	4.249666	+/- 0.002059
59	0.997448			⋮		
⋮				⋮		
493	1.010665	0.999516	+/- 0.000617	4.251376	4.264109	+/- 0.000347
494	1.005848	0.999530	+/- 0.000616	4.222168	4.264014	+/- 0.000347
495	1.006402	0.999546	+/- 0.000615	4.251478	4.263986	+/- 0.000346
496	1.022254	0.999597	+/- 0.000614	4.252508	4.263919	+/- 0.000344
497	1.015639	0.999633	+/- 0.000615	4.228688	4.263841	+/- 0.000344
498	0.980130	0.999589	+/- 0.000613	4.275204	4.263866	+/- 0.000343
499	1.003106					
500	0.996262					
Final Answer Keff: 0.999589 Relative Error: 0.000613						

Example Time Constant Output

k - eigenvalue				Radius of Gyration		
Cycle	Current	Average	Rel. Err.	Current	Average	Rel. Err.
1	1.344171			3.213161		
2	1.159585			3.806572		
3	1.080421			4.062678		
4	1.040986			4.168172		
⋮						
24	1.006850			4.271169		
25	1.014390			4.271169		
26	1.005609	1.005609	+/- 0.000000	4.251401	4.251401	+/- 0.000000
27	1.005553	1.005581	+/- 0.000028	4.263567	4.257484	+/- 0.001429
⋮						
72	1.008334	1.008487				
73	1.010252	1.008524				
74	1.003583	1.008423	+/- 0.000515	4.262759	4.263343	+/- 0.000313
75	1.005994	1.008374	+/- 0.000507	4.260715	4.263291	+/- 0.000307
Final Answer Keff: 1.008374 Relative Error: 0.000507						

Initial iteration values based on k-eigenvalue results

Initial k_{eff} to determine criticality

Starting Alpha : 0.097000 Starting Gen. Time : 0.086333

Cycle's α -eigenvalue

Cycle's k'

Cycle	Current	Alpha - Eigenvalue		Lifetime	Keff
		Average	Std Error		
1	0.1060447			0.0851081	1.0007629
2	0.0788198			0.0843681	0.9977183
⋮					
10	0.1365403			0.0851148	1.0024433
11	* 0.0961931	0.0961931	0.0000000	0.0850902	0.9988228
12	* 0.0798757	0.0880344	0.0081587	0.0846102	0.9986287
13	* 0.0937753	0.0899480	0.0050843	0.0850163	1.0011724
14	* 0.1164032	0.0965618	0.0075278	0.0851799	1.0019086
15	* 0.1499082	0.1072311	0.0121587	0.0856977	1.0028350
16	0.0800078	0.1026939			0.9941374
17	0.1050652	0.1030326			0.9901996
⋮					
259	0.1019834	0.0990292	0.0014854	0.0851741	1.0002505
260	0.0838201	0.0989683	0.0014807	0.0842308	0.9987279

Final α -eigenvalue with absolute uncertainty

Running average and uncertainty for α -eigenvalue

Cycle estimate of system lifetime

notes cycle value will be used in computing correction rather than using average.

***Final Alpha: 0.098968 +/- 0.001481 Keff: 0.998728 Lifetime: 0.084231

Cycle's α -eigenvalue

Denotes cycle value will be used in computing correction rather than using average.

Cycle estimate of system lifetime

Running average and uncertainty for α -eigenvalue

Final α -eigenvalue with absolute uncertainty