Investigation of Improved Methods for Assessing Convergence of Models in MCNP Using Shannon Entropy

Ruaridh Macdonald

May 14, 2012

SUBMITTED TO THE DEPARTMENT OF NUCLEAR SCIENCE AND ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN NUCLEAR SCIENCE AND ENGINEERING AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE, 2012

[illegible]

Signature of Author.______________________

Ruaridh Macdonald
Department of Nuclear Science and Engineering
11th May, 2012

[illegible]

Certified by:______________________

Benoit Forget
Assistant Professor of Nuclear Science and Engineering
Thesis Supervisor

[illegible]

Certified by:______________________

Kord Smith
KEPCO Professor of the Practice of Nucelar Science and Engineering
Thesis Reader

[illegible]

Accepted by:______________________

Dennis F. Whyte
Professor of Nuclear Science and Engineering
Chairmen, NSE Committee for Undergraduate Students

Abstract

Monte carlo computationals methods are widely used in academia to analyze nuclear systems design and operation because of their high accuracy and the relative ease of use in comparison to deterministic methods. However, current monte carlo codes require an extensive knowledge of the physics of a problem as well as the computational methods being used in order to ensure accuracy. This investigation aims to provide better on-the-fly diagnostics for convergence using Shannon entropy and statistical checks for tally undersampling in order to reduce the burden on the code user, hopfully increasing the use and accuracy of Monte Carlo codes. These methods were tested by simulating the OECD/NEA benchmark #1 problem in MCNP. It was found that Shannon entropy does accurately predict the number of batches required for a source distribution to converge, though only when when the Shannon entropy mesh was the size of the tally mesh. The investigation of undersampling showed evidence of methods to predict undersampling on-the-fly using Shannon entropy as well as laying out where future work should lead.

Acknowledgements

I would like to express my deepest appreciation to everyone who has helped me with this thesis. In particular I would like to extend a thank you to Professor Forget, my thesis advisor, and Paul Romano, the PhD student for who I worked; both of whom patiently waited for me to catch up with all the concepts as well as putting up with me when my variety of interests caused me to be distracted. Finally, I would like to thank Professor Smith for providing me with feedback and additional questions when writing this document.

Contents

List of Figures

List of Tables

1 Background

1.1 Motivation

Computational analysis forms the bedrock of most nuclear systems design due to the high costs inherent in building physical prototypes. As such, the development of tools which can handle a large variety of geometries, energy spectrums and other user inputs while providing fast and accurate results is a major goal of many national laboratories, universities and companies. Monte Carlo codes and methods are used extensively in the modeling of nuclear systems and provide some of the most accurate analysis currently available. However, use of codes such as Monte Carlo N-Particle (MCNP), developed at Los Alamos national lab, require the operator to have a high level of understanding of both the physics of a problem as well as the computational methods used to solve the problem correctly. In particular, knowing that the source distribution of a model has fully converged before beginning to collect results is critical to ensuring that those results are accurate. However, some of the indicators used to judge convergence can be misleading or simply wrong, telling the user that the source is converged when it hasn't. Even if the user is aware of this issue, their only solution is to make an estimate of when convergence will occur and then checking after the run has finished, necessitating repeated runs and wasted time if this guess proves wrong. In all cases,this problem can trip up users, either forcing them to waste time with extra runs or simply giving them incorrect results. Previous work, which will be described below, has shown how this problem may be solved, giving means for users to more accurately assess convergence and other aspects of their simulation. This investigation continues that work, focusing on methods that may be used to asses convergence on-the-fly as well as look for undersampling in tallied results.

This work has four deliverables:

1) Demonstration that, in the limit of high numbers of particles per batch, the Shannon entropy eventually becomes smooth, even for difficult problems.

2) Demonstration that as mesh size for the Shannon entropy becomes smaller, the observed convergence based on Shannon entropy is faster.

3) For varying tally sizes, determine how many particles per cycle are needed in order to eliminate bias in the tallies.

4) Show that if one uses a Shannon entropy mesh size proportional to the tally volume, convergence can be diagnosed with the Shannon entropy.

These will be explored in more detail in the methods section.

By working toward these goals, this investigation aims to reduce the difficulties faced by code users by identifying the possible relationships between model inputs such as tally size, mesh size and the number of particles per cycle in order to better inform them of appropriate inputs for different situations and in the future perhaps allow codes like MCNP to self-select some inputs based on the final results requested by the user. This will reduce the burden on the user and cut the computation time required for this type of analysis by increasing the chance that it will be done correctly the first time. In time other inputs could be handed over to the computer system, allowing the user to focus on the bigger picture of the simulation and reducing the barrier to entry of these analysis techniques. This in turn will hopefully increase the adoption of Monte Carlo methods in industry and elsewhere, improving the quality of nuclear analysis.

1.2 Monte Carlo Code, Overview

Monte Carlo N-Particle (MCNP) is a powerful general purpose, continuous-energy, generalized geometry, time-dependent (sometimes), coupled neutron/photon/electron Monte Carlo transport code [1]. MCNP functions by having the user build a geometry of their system, define all materials and sources within, as well as list the surfaces and volumes over which particle paths, effects and/or energies should be tallied. MCNP then uses a user-input initial distribution of source points and runs a number of particles, again decided by the user, through the model one at a time, randomly sampling from distribution of possible outcomes to determine the actions of the particles. These particle histories are tracked and recorded in batches,

Computational analysis of radiation systems generally falls into one of two categories: time-dependent and time-independent calculations. Both model the behavior of particles, materials and geometries in a user-built geometry either for one point in time, as is the case with time-independent calculations, or over time, as with time-dependtent calculations. Criticality are a type of (usually) time-independent calculations which assume that there is no transmutation of materials or other time dependent changes in the system and instead predicts the particles fluxes and tallies across all the volumes and surfaces for one particular time. Depletion calculations are a typical type of time-dependent calculation which combines the what is done in a usual criticality calculation with the ability to model time dependent changes such as the transmutation of materials. While this makes depletion calculations very useful it also makes them much slower, particularly when using Monte Carlo methods. As such, criticality calculations are far more commonly used. With the correct estimations of what a systems steady-state conditions will be, a handful of criticality calculations can predict particle densities, source profiles or the fractional change in the number of neutrons per neutron generation (Keff).

There are two means of solving both of these types of problems: deterministic codes and Monte Carlo codes. Deterministic codes set up and solve an explicit eigenvalue equation, which will be described in more detail later. The eigenvalue equation describes the particle populations in the system being simulated and deterministic codes solve this directly for the different regions of the system quite quickly, though at the cost of making certain assumptions which can make analysis very difficult or reduce it's accuracy. Monte Carlo codes, on the other hand, solve the problem using stochastic methods. This involves monitoring and recording the actions of many individual particles as they move through a system geometry and predicting average behaviors and the particle fluxes from the patterns that these particles form.

Criticality calculations, as described above, are large eigenvalue equations, where the source distribution of the particles, i.e. the initial positions of a generation of particles, is the eigenvector and Keff is the eigenvalue. In a Monte Carlo code, rather than ever writing out the matrix in the differential equation that is to be solved, the geometry of the system itself and how the particles interact with it when they are modeled make up the matrix of the equation. MCNP and other Monte Carlo codes use a power series method which involves using the results of previous batches of particles as inputs to the next batch. This causes the initial guesses of the positions of the source particles and Keff to gradually converge to their true values for the situation being modeled. This information can then be used to quickly predict reactor behavior and radiation flux across critical systems.

1.3 Monte Carlo Code, Details

As mentioned briefly above, codes like MCNP and OpenMC [12], an open source Monte Carlo code being developed at MIT, use a form of the power series method to solve for the source distribution and Keff of a system as part of solving a criticality calculation. The codes organize the particles it tracks through the geometry into equally sized batches (a.k.a. cycles), the size of which is chosen by the user. Each of these batches uses the final positions of the particles in the previous batch as initial locations and then the code models them, using their macroscopic cross section to calculate the expected distance each particle will travel between each new interaction. In a criticality calculation, MCNP and OpenMC model the creation of new particles as well as the destruction of existing ones and once each particle in the cycle has passed, record their final locations. The next batch then uses these positions as their initial source distribution. Because certain regions of the model will more favorably retain particles, i.e. areas of high particle fluxes, gradually more particles will be created in those regions and the source distribution will drift towards those locations. In order for each batch to be equivalent and to avoid exponential growth in the number of particles being modeled, the total number of particles is renormalized by randomly adding or removing particles. Figure 1 shows this process.

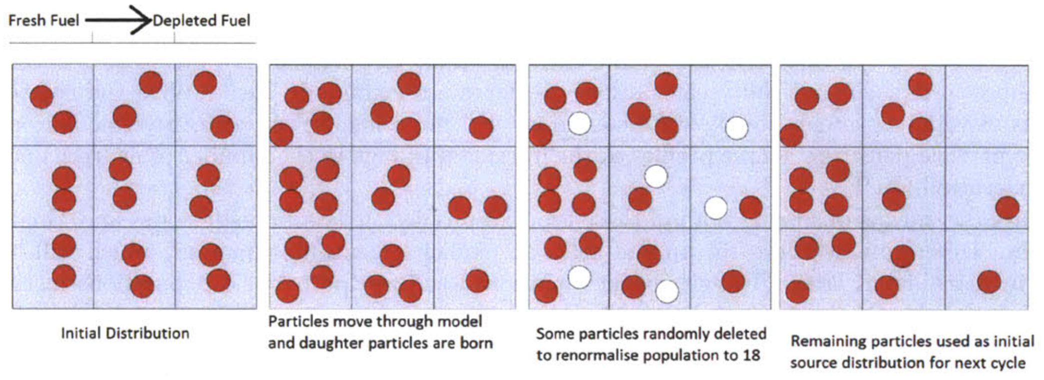
Fresh Fuel → Depleted Fuel
Initial Distribution
Particles move through model and daughter particles are born
Some particles randomly deleted to renormalise population to 18
Remaining particles used as initial source distribution for next cycle

Figure 1: Demonstration of source distribution converging. Each red dot represents a source site

The number of batches required for this process to complete depends on many different variables, both to do with the geometry and materials of the problem itself as well as the statistics of the numbers of particles per batch and the accuracy of the initial source distribution guess. One example of when this can go wrong is if too few particles are tracked through the model each cycle which can lead to one or two problems occurring. These particles may not be numerous enough to 'fill' the model and be drawn to the parts that will retain them best, i.e. the places in the geometry where the true source distribution will be high. This is especially the case in highly decoupled geometries, where the parts of the model with high particle and source densities are separated either by long distances, shielding or other particle-poor zones. This will slow down or completely inhibit the Monte Carlo method from converging to the true source distribution. This issue is often closely related to the dominance ratio of a problem. The dominance ratio describes the ratio of the first harmonic solution to the fundamental solution. Generally, the ratio is quite small and the fundamental solution, which applies to the first eigenvalue: Keff, has easily the greatest magnitude and dominates the other solutions. However, if the dominance ratio is close to one then it will take many more iterations for the fundamental solution to overcome and stand out from the first harmonic. A problem related to having too few particles, is when the adding or removal of a significant portion of the particles causes too much information to be lost or randomly added, inhibiting the speed at which the Monte Carlo system finds the true eigenvalue and eigenvector. These two particle population problems are described in Figure 2.

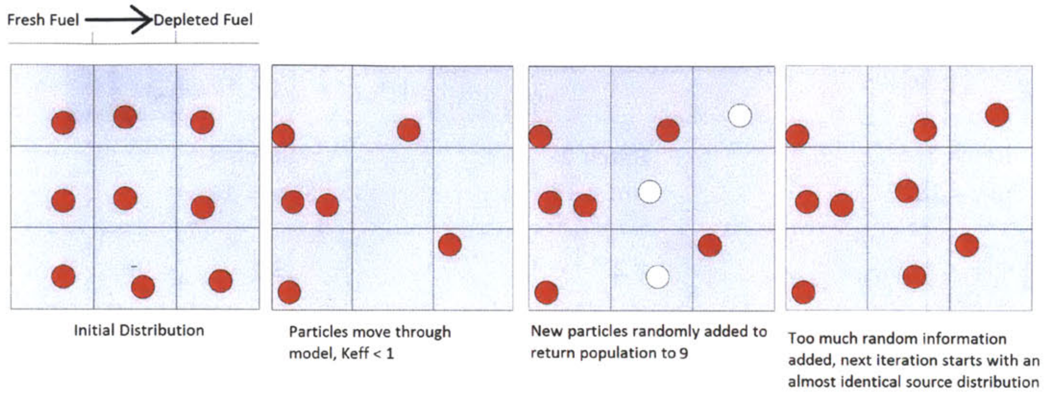
Fresh Fuel → Depleted Fuel
Initial Distribution
Particles move through model, Keff < 1
New particles randomly added to return population to 9
Too much random information added, next iteration starts with an almost identical source distribution

Figure 2: Demonstration of undersampling. Each red dot represents a source site

These early cycles, before the eigenvalue and eigenvector have converged, are referred to as inactive cycles because MCNP and OpenMC do not collect tally data during these cycles. Tallies are specific volumes of surfaces over which the user has asked for information, such as the number of particles crossing, their energy, flux, reactions rates, whether they have been scattered beforehand and many others. If this information is collected before the source distribution is converged, the information could be wrong. For example, if a tally was taken across a surface near the right hand edge of the core in Figure 1 and the code began tallying immediately at the first cycle, the flux tallied would be artificially high, distorting the data unless the model was allowed to run for enough cycles that the initial incorrect data becomes insignificant, though that would be a time-consuming and wasteful practice. Instead, tally data is only collected during active cycles, i.e. cycles after the source distribution has converged.

The above has hopefully demonstrated the need for Monte Carlo users to know when their source distribution has converged so that they do not accidentally start collecting tally data during what should be inactive cycles. Keff, the eigenvalue in the criticality problem, can be easily calculated and compared between cycles and many previous efforts have focused on using this to allow the user to test whether they should swap to active cycles after Keff (and hopefully the source distribution) converged to their true values. The calculation is easy because it involves simply dividing the final number of neutrons after a given cycle by the number that began the cycle and it is easy to compare single numbers for obvious reasons. However, Keff is a global value, i.e. it gives information about the entire model but not each point that makes it up. Because tallies are generally only concerned with local particle effects this means that while the entire model may appear converged, the source distribution near the tally may not be. This is demonstrated in Figure 3. Each of the bitmaps could be described as 'grey', i.e. the sum of their color maps is two. However, each of these is very different from the other. It is imaginable that this sequence of pictures describes the convergence of

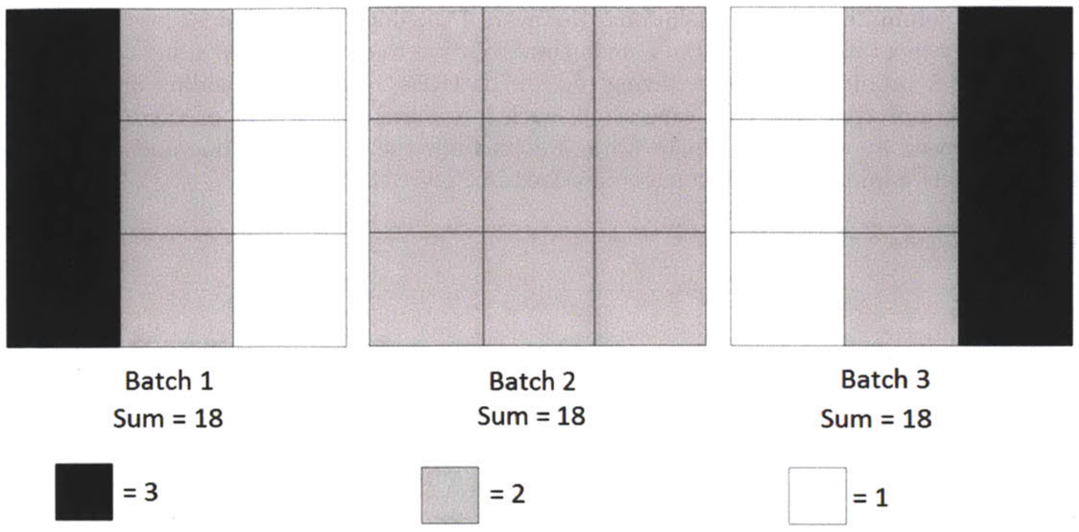

Figure 3: Demonstration of how local changes can be hidden when counted as part of a global variable

a source distribution where the initial source distribution was entirely wrong. At each point Keff is converged to two but the source distribution, given that it is made up of local information, is not.

The logical step from this is to develop methods to compare the convergence of the source distribution directly; however, there are problems in doing this. First and foremost is that the source distribution contains significant amounts of information, generally five double precision floats for each particle (x,y,z positions, energy and an ID number). Given that some larger or more complicated simulations may have millions or hundreds of millions of particles and thousands of batches. Storing this amount of information even once per hundred cycles is impractical. Secondly, even if this was a possibility, there is the problem of how to compare such large data sets, especially given that any given particle/source point in cycle 'n' may not be easily traceable to a particle/source point in cycle 'n+1' which makes it hard to compare how far they have traveled or how different each source distribution is from the previous one, though this has been attempted with some success[2]. One significant suggestion on how to solve this is the use of Shannon Entropy, a concept from information theory, to measure the randomness and amount of information in a system and describe it using a single number. This number can then be easily compared between cycles.

1.4 Shannon Entropy

Shannon entropy (a.k.a information entropy) is a concept from information theory, originally proposed by Claude Shannon as part of his masters thesis at MIT [11]. Shannon entropy is very analogous to thermal entropy and is a measure not only of the chaoticness in a system but also of the number of bits required to fully define a system. This is similar to how thermal entropy is both a measure of 'chaoticness' and the number of states required to fully define a system. Previous work, especially by T. Ueki, has shown a high correlation between the source distribution and the

Shannon entropy and it is thought that if one is converged the other will be as well, giving users a means of checking the convergence of the source distribution [3]. Some work by Bo Shi at Georgia Tech seemed to suggest that this was not true and that the Shannon entropy failed to sufficiently predict when the source distribution converged. This will be discussed in more detail in the methods section.

In MCNP, the Shannon entropy is calculated for a given cycle/batch by imposing a 3D mesh on the model and counting the number of source sites / particles in each. This mesh can either be user defined or MCNP will automatically generate a mesh it believes is appropriate based on the average number of particles per bin. The number of particles in a given bin is then divided by the total number of particles 'p' to give a probability that a particle will be found in any given bin. This is then input into Equation 1 to calculate the entropy for the whole system.

$$-\sum^{N} p.log_2(p) \tag{1}$$

Where N is the number of bins in the mesh.

To understand this concept better it is useful to examine an example from Shannon's thesis. If we imagine a case with just two bins, A and B, the Shannon entropy, H, of a system containing any number of particles is given by the graph shown in Figure 4, where p is the probability of a particle being in bin A. If p equals one or zero, all the particles will be in one bin; the system is very ordered and 'unchaotic' and very little information is needed to describe it. As p goes to 0.5, this is reversed and more information is required to describe the system till p = 0.5 exactly and you have a uniform distribution of particles between the two bins. In this case this entropy is equal to one but in general it can take any positive value. Depending on whether a user uses a uniform or point initial source distribution, the Shannon entropy should grow or fall to it's converged value. The magnitude of the converged value could be compared to that of other systems but this would likely be quite a difficult task to make worthwhile because the 'chaoticness' of the system will be highly dependent on the number of bins and source distribution of the system.

Shannon entropy presents a way to compare the source distribution between cycles easily without needing to save large amounts of data. MCNP and most other codes delete the source distribution of a model once the Shannon entropy has been calculated. Methods are also required to compare the Shannon distribution between cycles accurately. While these methods could be used after a calculation has finished running, these methods are most promising when they done on-the-fly calculation so that the computer running the code could decide for itself when to swap from inactive to active cycles, reducing the need for repeated runs. There have been many suggestions[4, 6, 7, 5] to this end but one promising suggestion, developed at MIT by Paul Romano, is the use of a stochastic oscillator to quantitatively show when the Shannon entropy is 'flat' enough to be considered converged.

1.5 Stochastic Oscillator and Alternative Convergence Analytics

The stochastic oscillator works by reading trends in a single variable time series and determining when it meets a level of stationarity decided by the user beforehand. It uses an indicator 'K' to determine whether the time series is increasing or decreasing in value and if K stays within a certain bound from 0.5 then it is assumed to be stationary [8]. Applied to Shannon entropy the formula looks like this:

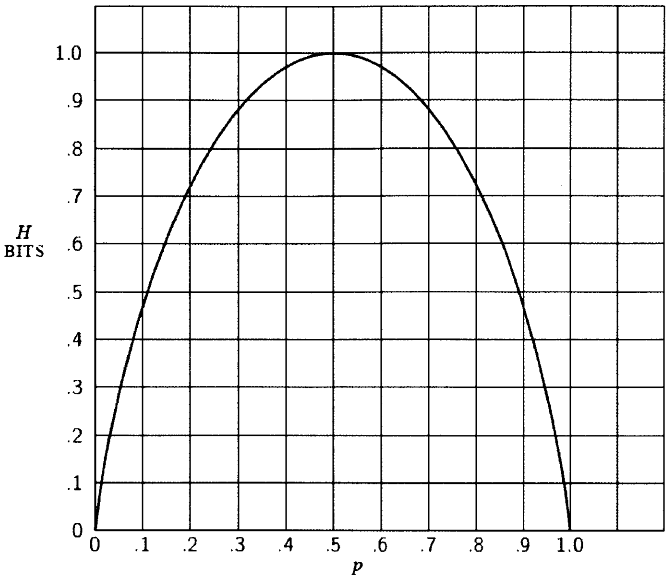

Figure 4: Shannon entropy for a system with two bins A and B where p = probability that a particle will be found in A[11]

$$K^{(n)} = \frac{H^{(n)} - H_{min}^{(n,p)}}{H_{max}^{(n,p)} - H_{min}^{(n,p)}} \tag{2}$$

Where H is the Shannon entropy, $H_{max}^{(n,p)}$ and $H_{min}^{(n,p)}$are the maximum and minimum Shannon entropies for the previous 'p' batches and K is the stochastic indicator. If the value is increasing, $H \to H_{max}$and K = 1, as the latest value is likely to have the greatest or close to the greatest value historically. Similarly, when the value is decreasing $H \to H_{min}$and K = 0. If the condition in Equations 3 and 4 holds true at any point then the Shannon entropy is considered to have achieved stationarity.

$$\left\| K^{(n)} - 0.5 \right\| < \epsilon \tag{3}$$

$$\left\| \frac{1}{m} \sum_{i=0}^{m-1} K^{(n+i)} - 0.5 \right\| < \varepsilon \tag{4}$$

Where m is another set of batches decided by the user. In other words, both $K^{(n)}$and it's average for the next 'm' batches must be withinε of 0.5 [8].

Applied to the OECD/NEA benchmark #1 problem, which will be described later in greater detail, Romano found the following results for various p,m and ε, shown in Table 1.

Histories per batch	**Batch converged via inspection**	*p*	*m*	*ε*	**Batch converged via Eq. (24)**
5000	Not converged	20	50	0.1	60
		500	500	0.1	Not converged
10000	2500	20	50	0.1	143
		500	500	0.1	2603
20000	1700	20	50	0.1	179
		500	500	0.1	2175
40000	1800	20	50	0.1	174
		500	500	0.1	1889
100000	1600	20	50	0.1	156
		500	500	0.1	3200

Table 1: Comparison of the number of batches for convergence in OECD/NEA benchmark problem #1 determined by visual inspection and by the stochastic oscillator diagnostic [8]

The second parameter set (p = 500, m = 500, ε = 0.1) worked fairly well, except in the last case, and this method definitely invites more work to see how the concept could be fleshed out and applied to these geometries which are traditionally more difficult to model accurately, such as this benchmark problem.

Other methods to assess source convergence include that built into MCNP as well as those developed by Bo Shi, Taro Ueki, Forrest Brown and others. MCNP checks Shannon entropy convergence by looking for the first cycle after which all Shannon entropy results are within one standard deviation of the Shannon entropy value for the second half of the cycles (active or inactive) [1]. While this method has some usefulness and is similar to some modifications made to the stochastic oscillator method that will be discussed late, on the whole it did not prove useful. As will be shown in the results section, this method completely fails for simulations using less than 10,000 particles per cycle, often evaluating convergence as being within the first hundred cycles and consistently gives convergence points that are significantly early even up to 1M particles per cycle. It is unclear whether this is a function of the difficulty of solving this problem and this method works well with other models but some of the literature shows use of this method and as such it was decided to use it as a comparison to other methods.

Bo Shi's masters thesis involved an investigation of the usefulness of Shannon entropy as a means of assessing convergence and the development of other methods to replace it [2]. As will be demonstrated later, this benchmark problem requires significantly more histories to be run in order to give accurate results than is suggested by the Expert Group on Source Convergence who made the model. Shi used around 10,000 particles per cycle for his investigations which this work has shown is about an order of magnitude too few to ensure timely convergence. As such, when Shi correctly determines that the source does not converge he concludes that Shannon entropy is not a valid method for judging source convergence. Instead, he suggests using the FMESH function in MCNP to produce a plot of the flux and perform statistical checks on it to compare multiple cases for varying numbers of completed batches and identify when convergence has occurred. Shi argues that the benefit of using a flux map as opposed to source site map, as is done in the investigation, is that the flux map will give information about non-fuel regions which allows any analysis to incorporate more of the model. One downside however is that the use of FMESH bins to tally the flux as opposed to finite source sites makes the data somewhat coarser. This method seemed effective for the simplified benchmarks in which it was applied but is not a method that can be used on-the-fly very easily and as such was not pursued in this investigation.

Finally, other methods have been developed by Ueki and Brown to assess source convergence using Shannon entropy and other techniques [6, 4, 5]. Most of these involved the creation of two or more data sets from one simulation and the comparison of these sets using statistical checks to see where they became similar, showing convergence. Some of these methods are only possible after a simulation has completed whereas others, such as Ueki's source ratio vector method can be performed on-the-fly[7]. Due to the complexity and long-winded explanations for most of these methods and the fact that the investigation largely made use of the stochastic oscillator, both for its ease of use and relative accuracy, these methods will not be described here but interested parties are recommended to read the references included above if interested further.

2 Method

2.1 Project Objectives

Before looking at the methods used in this investigation we will look at the objectives that were set out at the beginning of this paper in more detail.

1) Demonstration that, in the limit of high numbers of particles per batch, the Shannon entropy eventually becomes smooth, even for difficult problems.

2) Demonstration that as mesh size for the Shannon entropy becomes smaller, the observed convergence based on Shannon entropy is faster.

3) For varying tally sizes, determine how many particles per cycle are needed in order for there to be no bias in the tallies.

4) Show that if one uses a Shannon entropy mesh size proportional to the tally volume, convergence can be diagnosed with the Shannon entropy.

The ultimate aim of the project is to reduce errors in MCNP calculations by developing tools for users and eventually the system itself to setup and diagnose problems and how well they are running. These tasks build towards that, culminating in third and fourth objectives, which demonstrate undersampling and the use of Shannon entropy as a convergence check.

The first two objectives are concerned with demonstrating that when used correctly, Shannon entropy can diagnose source convergence in criticality calculations. As discussed in the background section, this method has been rejected by some investigators who have produced experimental results where the Shannon entropy in their simulation has converged before the flux in their system has stopped changing, i.e. before source convergence. However, these researchers did not discuss how they were managing the size of the Shannon entropy bins in their model, suggesting they used the MCNP defaults which allow for very large bins. If these bins were significantly larger than the areas on which the users were tallying flux then it is possible that while the macroscopic Shannon entropy is converged there would be still be local areas which are changing. This would cause premature Shannon entropy convergence for most of the same reasons as Keff does.

Completing objectives one and two involved modeling the OECD/NEA benchmark problem #1 in MCNP and comparing how the Shannon entropy curves with the flux convergence for runs with a variety of numbers of particles per cycle, number of active and inactive batches and differently sized Shannon entropy meshes (i.e. the mesh which divides the bins). This hinged on being able to read the source distribution of the simulations after intermediate batches as well as just the final batch, as is usually the case and then being able to take one number which summarizes the convergence of that distribution so that its convergence over multiple batches could be compared to that of the Shannon entropy. In order to this, the MCNP simulations were terminated at intermediate cycles so that the source distribution could be read before using a CONTINUE card to restart the simulation. This was automated using a python script, with the source distribution being extracted every fifty cycles. A second python script, written by Paul Romano, then converted the binary source file into a comma separated value (csv) format which could be read by most software packages, including matlab. Matlab was then used to organize the data from multiple runs together so it could be plotted or analyzed easily. After developing this method it was found that something similar had been suggested previously by H. Shim and C. Kim in 2007 [5].

The next challenge was to find one number to describe the state of the source distribution so it could be compared with the Shannon entropy. It was decided to take 3D tallies from the model by using matlab to plot the number of particles in the volume against the number of batches completed by the simulation. The curve produced could then be easily compared to the Shannon entropy and the model was considered to have converged when this curve became flat. This method could be scaled to assembly, pin or any other sized tallies which was convenient when trying to compare the same source distribution to multiple different Shannon entropy curves. As will be discussed early, because the OECD/NEA benchmark used has an almost linear decrease in Keff going from left to right in the geometry, it was possible to investigate whether the closeness of the relationship between the source and entropy convergences is dependent on Keff.

The third objective is concerned with the problem of undersampling, namely how it can be avoided and diagnosed. This involved measuring the bias in differently sized tallies in the system when using different numbers of particles per cycle. In each case, the total number of active histories was the same which should have meant that the quality of the statistics was the same in each case. However, because the histories are divided into differently sized batches and the manner in which information is carried by MCNP between batches, as discussed earlier, there will be very different results. By comparing tally results to those from a simulation which included many particles per cycles and active cycles, i.e. a case which has had enough particles that it should definitely be close to the true tallies, this bias effect from undersampling can be measured and compared for different numbers of particles per cycle.

The last objective then brings the previous objectives together, showing that if used appropriately, Shannon entropy is an appropriate means of measuring source convergence and that as long as the user avoids undersampling their tallied results will be meaningful.

2.2 OECD/NEA Benchmark Problem #1

Throughout this investigation the most commonly used model was the OECD/NEA benchmark problem #1. This problem was designed by the Expert Group on Source Convergence to greatly undersample the fissionable regions as well as being highly decoupled, with a very high dominance ratio and hence much slower to converge[9]. The model describes a spent fuel storage pool, with thirty six fuel assemblies arranged in a checkerboard pattern submerged in water and enclosed by concrete on three sides. This is shown in Figure 5, where the darkest squares are fuel and the lightest are water.

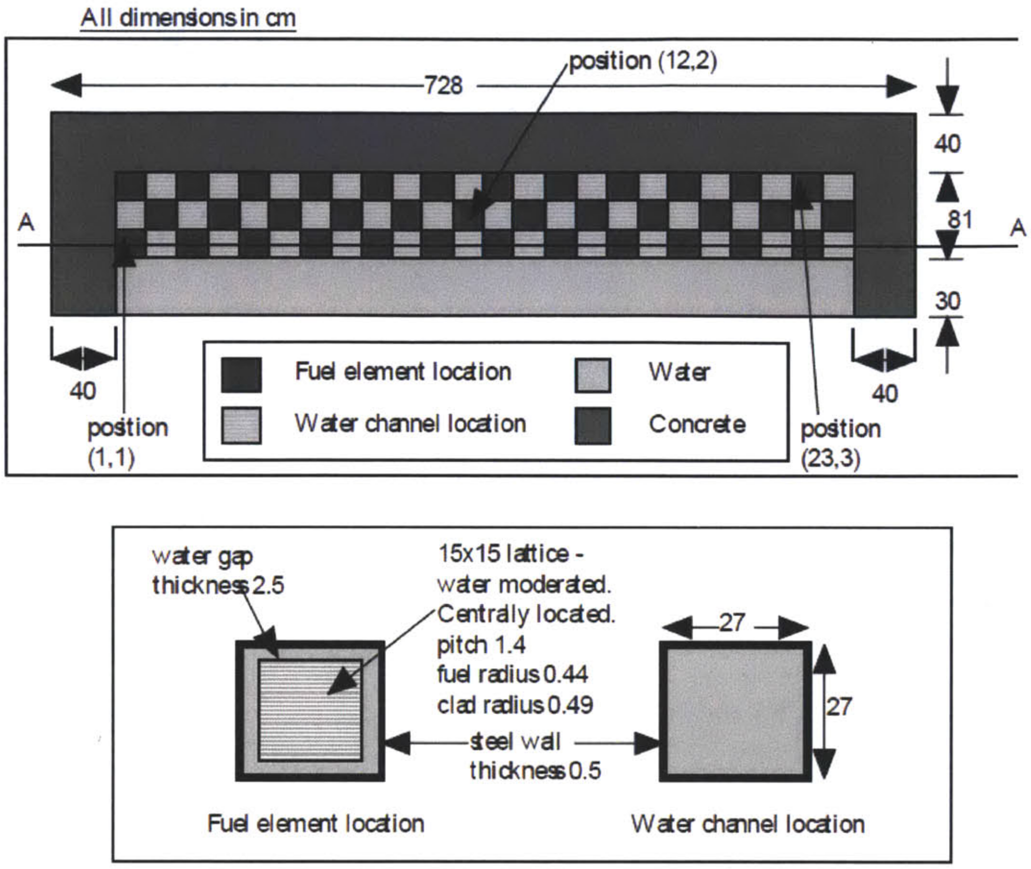

Figure 5: x-y view of OECD/NEA benchmark #1 [9]

MCNP simulations were always initiated with a uniform source distribution in the fuel regions. The final flux profile of the model is very unique, with almost all of the source points being concentrated in the three upper left assemblies. This is because of the reduced neutron leakage in that corner due to the concrete lining two of the three sides of the upper left assembly. When the model is fully converged there will be almost no source particles in the middle or to the right of the model. A picture of this final source distribution is given in Figure 6.

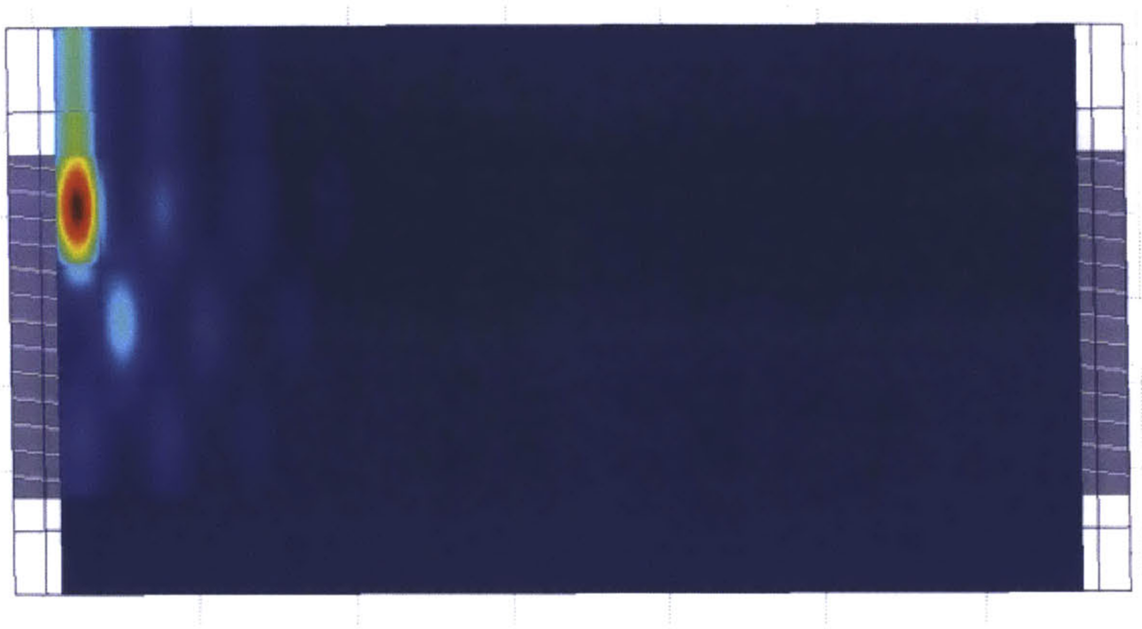

Figure 6: Flux map of OECD/NEA Benchmark Problem #1 MCNP made using FMESH command

Where red indicates high flux and blue low flux.

3 Results

3.1 Characterizing Model

In their report, the Expert Group on Source Convergence suggests using 5,000 particles per cycle when simulating the #1 benchmark problem [9]. Very quickly however, it became clear that, at least when using a uniform initial source distribution, that was completely inadequate. It was found that at least 50,000 particles per cycle were needed to get relatively speedy convergence of the Shannon entropy and that in most cases 3000 inactive cycles needed to be run to ensure convergence. Figure 7 shows the Shannon entropy curves for the simulation using various numbers of particles per cycle. Note that these have not been normalized by dividing each of them by their maximum possible Shannon entropy ($log(N)$, where N is the total number of mesh bins).

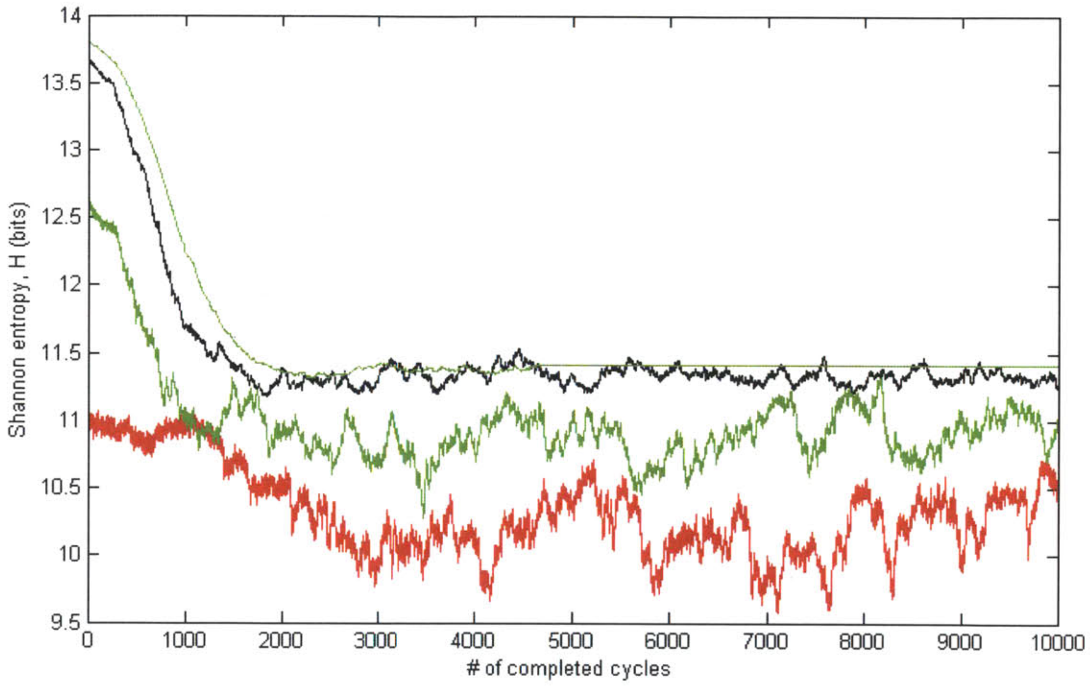

Figure 7: Shannon Entropy curves calculated over pin cell sized bins. In descending order at BOL, the runs have 1M, 100,000, 10,000 and 3000 particles per cycle. Note that the 1M particle per cycle simulation was only run for 5000 batches; all data points past that are extrapolations of the 5000th value.

Note that for cases with greater than 100,000 particles per cycle the runs were terminated after 5000 cycles as opposed to 10,000 cycles in order to reduce calculation times and the straight lines that are plotted are simply extrapolations of the final value.

From the figure, it is clear that when using 5000 particles per cycle, the curve barely flattens in comparison to the curves from cases where more particles per cycle were used, suggesting that even over 10,000 cycles there was relatively little movement of the source particles to the upper left hand region. The idea that the source distribution is still very much changing after this many cycles is also supported by the large fluctuations in the entropy value. In comparison, when using greater than 50,000 particles per cycle and in particular past 300,000 particles per cycle, the curve shows minimal fluctuations, suggesting a stable source distribution and the overall value drops significantly as the source distribution presumably concentrates in the top left of the geometry. It was found in general that using fewer than 100,000 particles per cycle and less than 3000 inactive batches was unwise if you wanted a relatively quick convergence to a stable Shannon entropy.

3.2 Source Distribution Analysis Method

Having shown that the Shannon entropy converges for cases with enough particles per cycle, these curves can now be compared to the source distribution curves. Figure 8 gives an example of what the plots of the source distribution generally looked like for a converged simulation. The checkerboard pattern of the benchmark can clearly be seen in the source particles and most of the particles have moved to the top left of the geometry. Each red dot represents one source site. The information on the particles energy is not used in this case but examples have been made where the color of the particles indicated their energy.

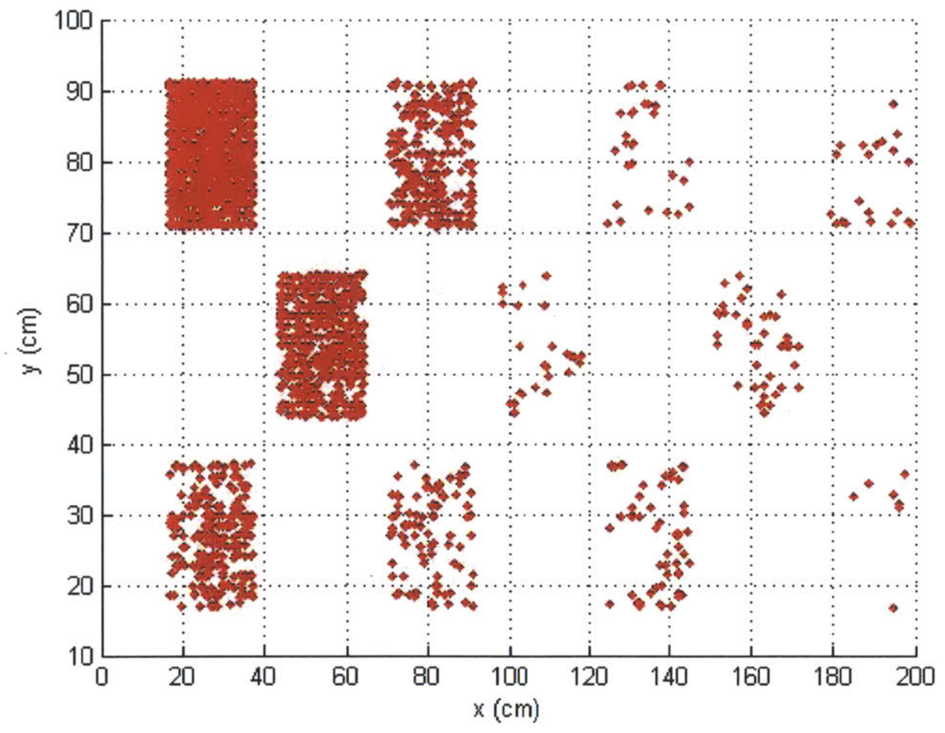

Figure 8: Example of OECD Benchmark Problem #1 source distribution. Run involved 100,000 particles per cycle for 5000 cycles.

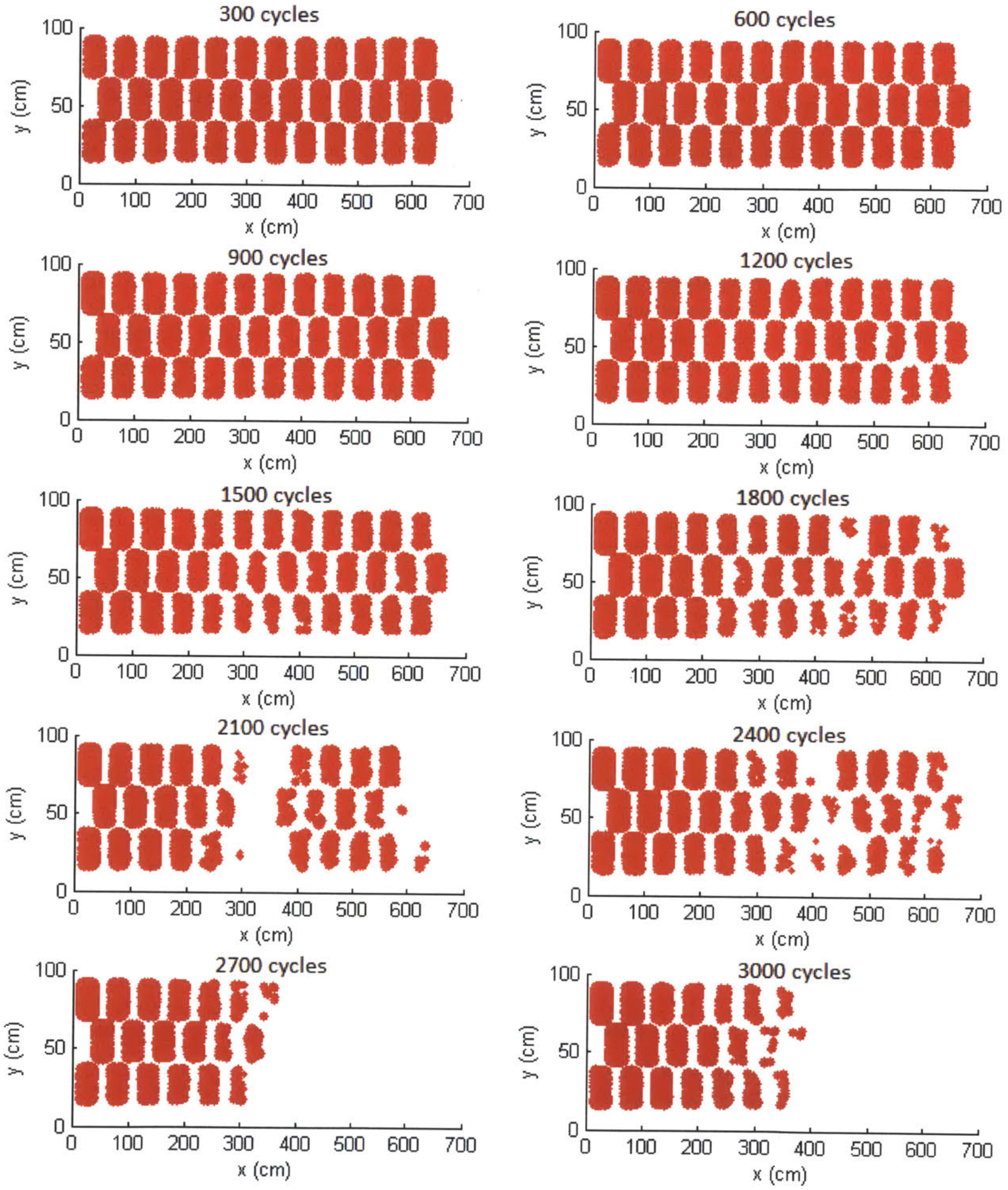

Figure 9: Small multiples plot of source distribution development from 300 to 3000 cycles using 10,000 particles per cycle

Figure 9 shows the source distribution of a simulation converging over multiple cycles, with each image being taken every 50 cycles. It is clear that this is a useful technique for looking at source convergence directly. The additional computation time due to using CONTINUE cards in MCNP and having to restart simulations multiple times is minimal if managed by a python script and careful cataloging of RUNTPE files (binary files saved periodically by MCNP during simulations which contain all the information about a run and can be used as starting points to continue or restart a simulation) can allow users to go back and make further investigations. One drawback to this method is that MCNP does not allow CONTINUE runs to add inactive cycles to a simulation, only active ones, which means there cannot be comparisons between simulations with differing number of inactive cycles. There was also evidence found that MCNP treats active and inactive cycles slightly differently because the Shannon entropy curves for the exact same situations and random number seeds produced different results depending on whether they were operating in active or inactive modes, as seen in Figure 10. The plot shows a four equivalent runs of a case, each using the same random number seed, initial source distribution and geometry. Each simulation ran for 800 total cycles but different numbers of inactive cycles, as described in the legend. Given that nothing changes in each of these runs except whether they are using active or inactive cycles there should be only one curve but at every point that a case changes to active cycles a new curve appears (the curves are layered so the ones with more inactive cycles sit on top). This suggests a change in the handling of active and inactive cycles. Given that it was unclear how much of an effect this would have or how to fix it, all cycles in the python-iterated CONTINUE runs described above were run as active cycles to avoid it completely. While this made any MCNP tally data meaningless it should not have effected the source distributions.

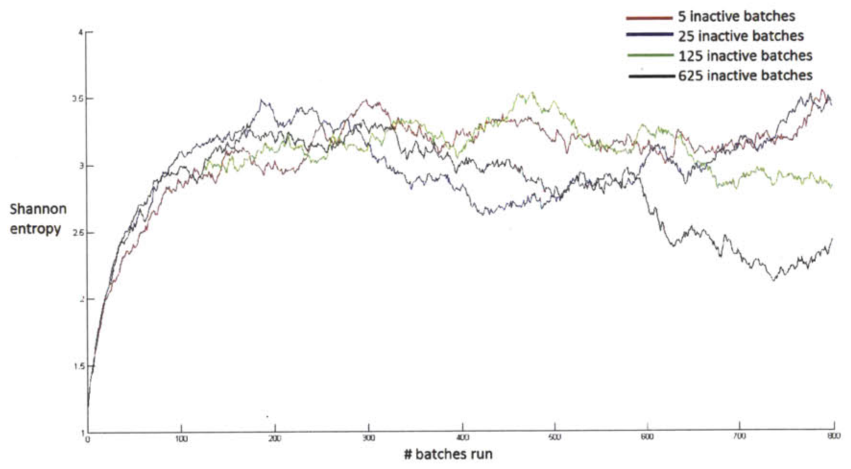

Figure 10: Shannon entropy curves from identical cases run on OECD benchmark problem #1 with 100,000 particles per cycle

As discussed above, in order to compare these source distribution plots to the Shannon entropy curves, the number of source points in a volume of the same size as the Shannon entropy bins used was summed, i.e. the number of particles in one pin or assembly if the Shannon entropy bins were each the size of a pin or assembly respectively. These sums were then plotted as a curve against the number of cycles that the simulation had run. One weakness of this method was that the curves would have a very large variance if there were relatively few particles in a given tally volume due to the random movement of source sites. To mitigate this, all cases were run with 1,000,000 particles per cycle. This produced very good results in most situations, the biggest exceptions being when summing over a pin cell in a part of the model with relatively low flux, and was considered a good compromise between running more particles per cycle (and increasing computation time) versus having relatively large variance on all the results.

In order to mitigate this effect further, most of the source distribution curves were taken by summing the number of particles in the regions with the highest particles fluxes. This also had the benefit of allowing the curves to be taken out to higher numbers of batches because the number of source particles didn't just rapidly go to zero. This assumed that the highest flux assembly would accurately represent the convergence rate of all of the assemblies in the geometry. Figure 11 tests this assumption by showing a plot of the number of particles in each fuel assembly along the top row of the model, and how quickly each of them converge. The flux largely decreases linearly as you move to the right of the model, with a slight uptick at the furthest right assemblies which have less leakage than the middles ones due to the concrete wall on the right hand side. The figure seems to support the assumption made for the most part, though in some cases, the number of particles in the volume descends to zero very quickly, making it difficult to be sure if the entire trend is shown. To be more certain of this, the example could be run with more particles per cycle but instead, as will be shown later, all of the experiments to see how source convergence related to Shannon entropy convergence were repeated with volume tallies from just right of the middle of the model, the assumption being that this region was representative of the lowest flux regions while not having a particle count that immediately dropped to zero, necessitating (slow) runs with very high numbers of particles per cycle.

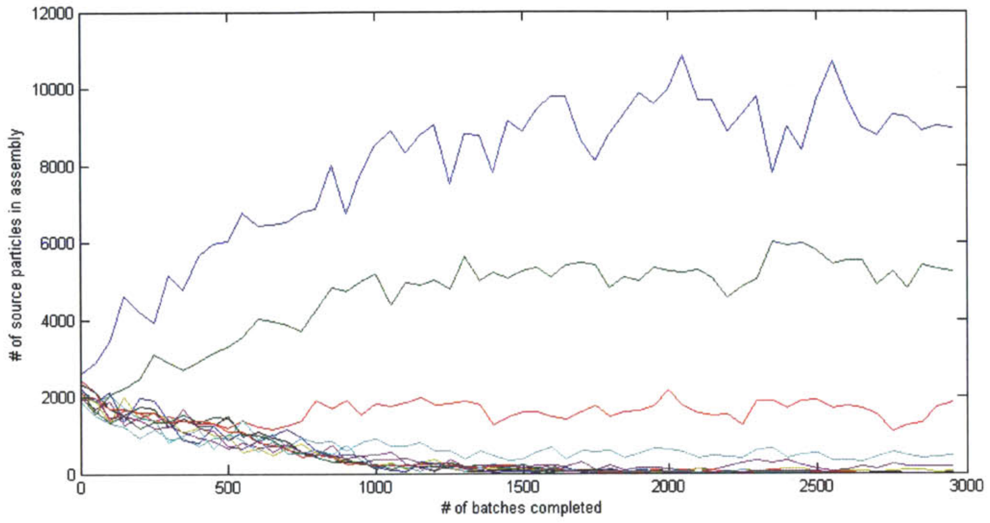

Figure 11: Number of particles in the twelve assemblies along the top row of the #1 benchmark problem benchmark when run with 100,000 particles per cycle. Each curve represents an individual assembly

3.3 Comparison of Shannon Entropy and Source Distribution Curves

Figures 12, 13 and 14 show a comparison of these curves to the Shannon entropy curves for equivalent cases. Some of the Shannon entropy curves have been raised to higher powers in order to more easily plot them against the source distribution curves. Note that the [1,3] co-ordinates refer to the assembly in question being in first column from the left, third row up from the bottom. The 3x3 mesh means that the Shannon entropy and source distributions are summed over a 3x3x1 grid. As well as the general result, of interest from these plots, though not something explored in great length in this investigation, is the periodic oscillations in the source distribution curves, particularly in Figure 14. The reason for this is unclear though it is not limited to just these cases and was seen in most examples where the variance was low enough to produce a significant curve.

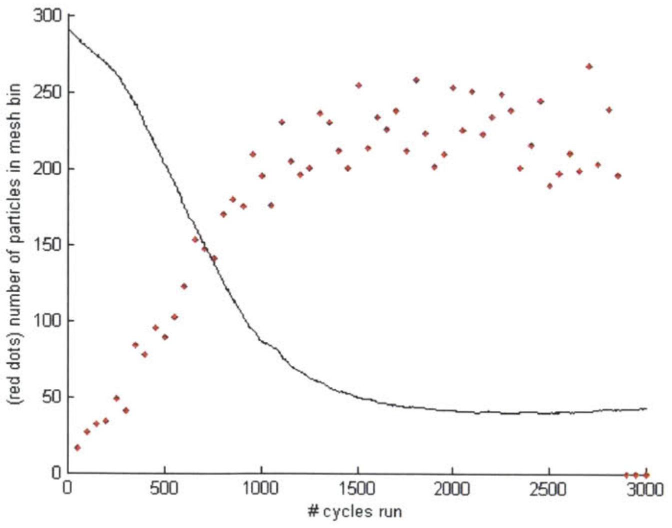

Figure 12: Pin Mesh, 1M particles per cycle, High flux (middle pin of [1,3] assembly) Shannon entropy curve raised to tenth power

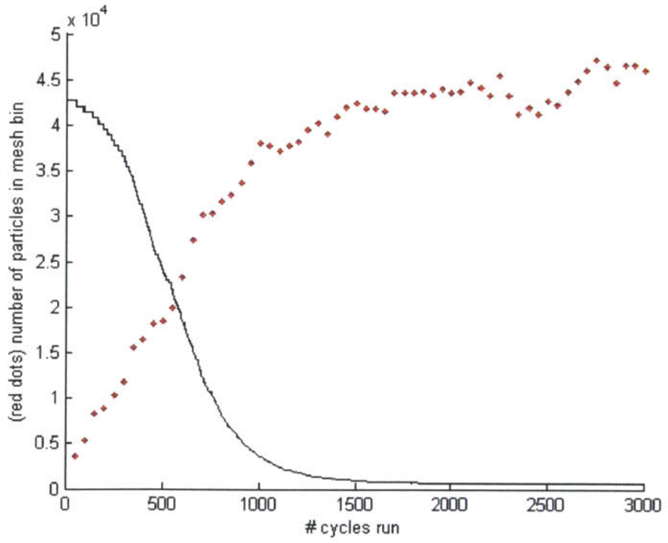

Figure 13: Assembly Mesh, 1M particles per cycle, High flux ([1,3] assembly) Shannon entropy curve raised to tenth power

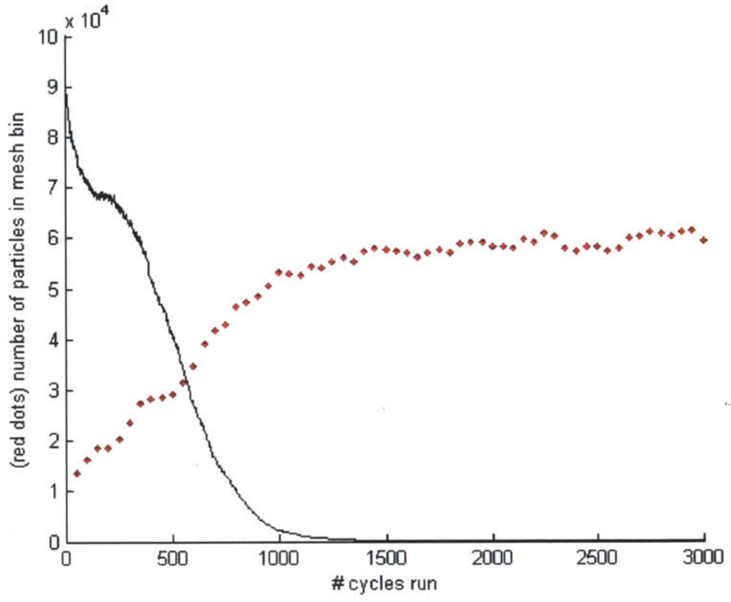

Figure 14: 3x3 Mesh, 1M particles per cycle, High flux ([1,3] mesh bin) Shannon entropy curve raised to tenth power

From visual inspection, the Shannon curves appear to converge at the same time or slightly after the source distribution curves but the margin is quite small, as in the cases shown in the figures. If true this would support the hypothesis put forward earlier in the paper that Shannon entropy will be a good indicator of convergence when calculated over mesh bins that are appropriately sized and hence deserves further investigation. A piece of evidence that seems to more clearly agree with what was put forward earlier, is that the point of convergence of both curves seems to be after a greater number of cycles if the mesh bins / tally volumes become smaller. This makes sense with what was discussed earlier about multiple local variations, which may each take a long time to converge, hiding one another when counted together, making larger areas converged seem converged too early. This effect is less pronounced between the figures showing pin cell and assembly sized bins due to the relative similarity in the bin sizes but is much clearer when comparing those two with the 3x3 mesh case.

3.4 Improved Source Distribution Analysis Methods

In order to make more accurate conclusions from the data, it was necessary to develop more accurate means of assessing it. To begin with, it was hoped to use the same methods as were used with the Shannon entropy curves but this didn't work, largely because in this case the data sets were relatively small, around sixty points, as opposed to thousands of points in the entropy curves. This meant comparisons of trends, as is used by the stochastic oscillator, could only realistically compare a handful of data points which made the method ineffective. Instead it was decided to make an approximation to this method by taking lines of best fit and doing analysis on that curve. Matlab was used to do this, using 6th order polynomial curves. These were chosen because, across all the different examples, they generally gave the best fit when assessed by summing the residuals. When assessing the point of convergence on these curves, it was then found that judging the point of convergence as being the batch at the midpoint between the first two peaks gave the most consistently sensible results. This is demonstrated in Figure 15.

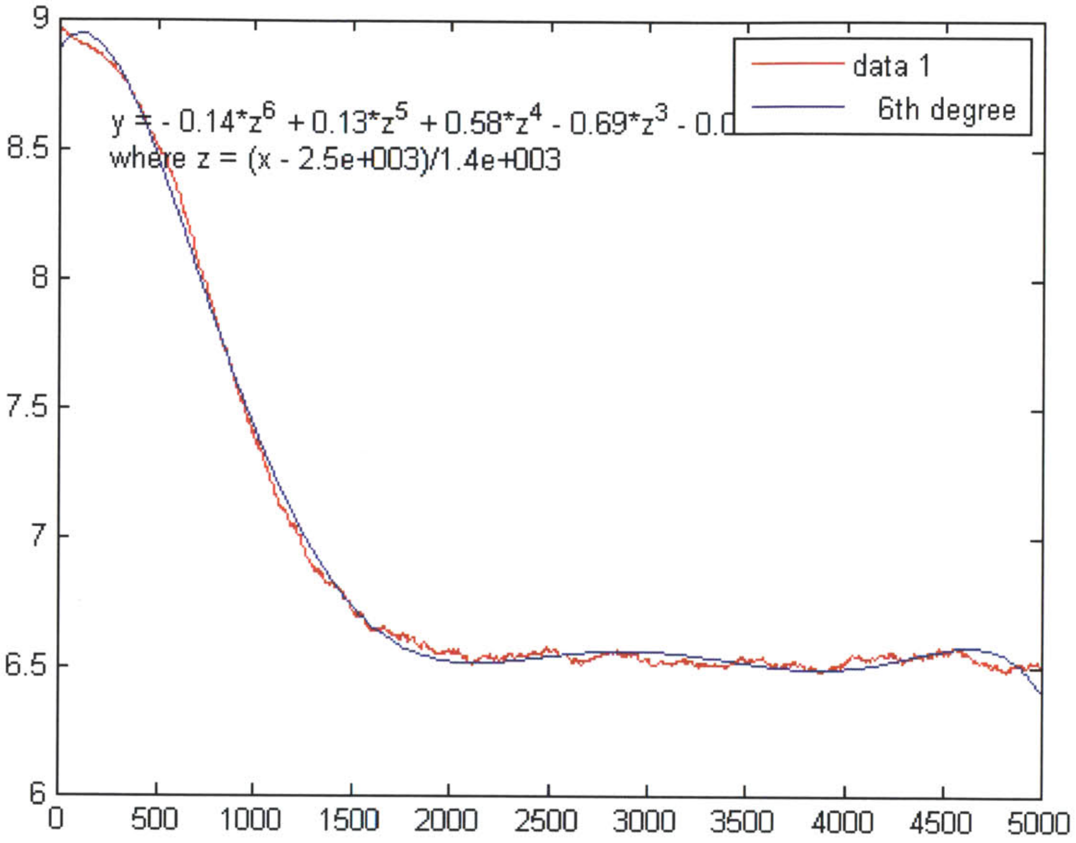

Figure 15: Example of source dist curve fitting. Red line shows Shannon entropy for case with 100,000 particles per cycles on a 108x22 mesh. Blue line shows the line of best fit for the number of source points in the upper left assembly for the same run

In this figure, the blue line is the line of best fit of the number of particles in a tally volume in the top left of the model during a simulation with 100,000 particles per cycle, run for 5000 cycles. The red line is the equivalent Shannon entropy curve. The blue line has had all of the values scaled by a multiplier to allow it to fit more easily on the plot. Note how well the two curves match. While this method may seem complicated, it was shown to work well across many cases, including different tally volume sizes, fluxes and initial source distributions. It is also easy to code into Matlab, allowing the processing to be done rapidly, though only at the end of the run; other methods would be needed for on-the-fly calculation.

3.5 Shannon Entropy Convergence Methods

In a similar manner to the source distribution, a rigorous method was needed to judge the convergence of Shannon entropy curves. To judge how well various methods worked, a combination of comparing the result to the by-eye result and seeing how well it fit the data from the equivalent source tally distributions was used. It does not escape the investigators that using the curve you are trying to approximate as your assessment of whether your method works might not always be rigorous. In order to avoid the human tendency to pick methods or data that were convenient, i.e. they estimated convergence to be near to the value predicted by the source distribution curves, the differentials of both curves were compared for the entire curve, not just at the inflection points. While again not perfectly rigorous it was believed to be the best means available.

# of particles per cycle	Stochastic Oscillator	MCNP checker	By eye
500	500	1	Does not converge
1000	500	1	Does not converge
2000	922	799	2393
3000	501	1677	3025
4000	561	1999	2861
5000	1388	873	3663
7000	1662	1030	3859
9000	1839	1156	3546
10000	1948	950	3238
12000	1401	935	3688
15000	2661	1739	2473
18000	1431	1343	2989
20000	2547	1642	3670
30000	3546	1522	3083
50000	1648	1339	2116
80000	2216	1720	2563
100000	1922	1459	1973
300000	3240	1624	2315
500000	3328	1871	2417
800000	2500	1846	2246

Table 2: Comparison of Shannon Entropy convergence points using various techniques

The stochastic oscillator came out very well compared to the built-in MCNP convergence checker, comparative Shannon entropy and other methods, both in terms of accuracy and computational cost and as such it was used in all the Shannon entropy analysis in this investigation. Table 2 summarizes results from the stochastic oscillator in comparison to those from the built-in MCNP checker and by-eye estimations of the convergence cycle. However, there were some occasions in which it did not work as effectively, all of which were simulations which had less than 100,000 particles per cycle. In these cases the stochastic oscillator would claim convergence well before it had actually occurred because large fluctuations in the Shannon entropy balanced out to produce a steady aggregate result. An example of a case where this was a problem is shown in Figure 16.

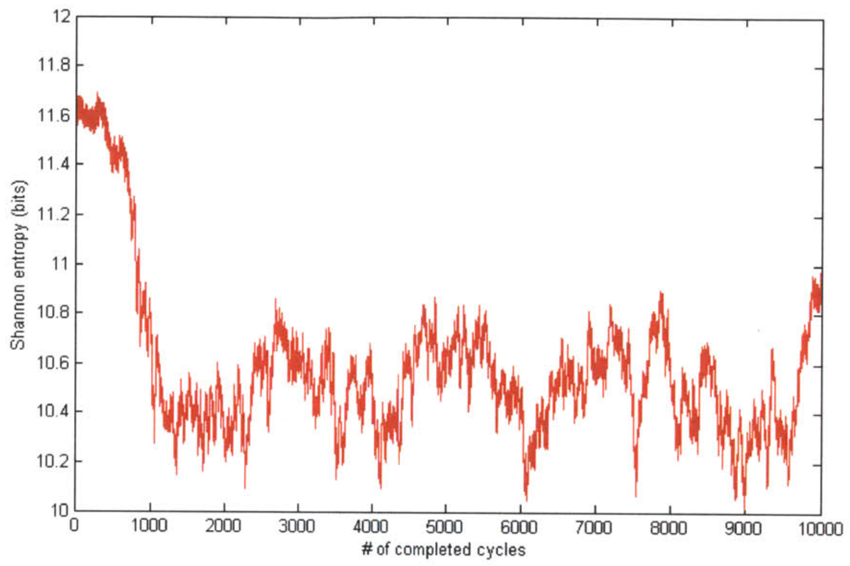

Figure 16: MCNP run from OECD Benchmark. 10,000 particles per cycle on a pin cell sized Shannon mesh

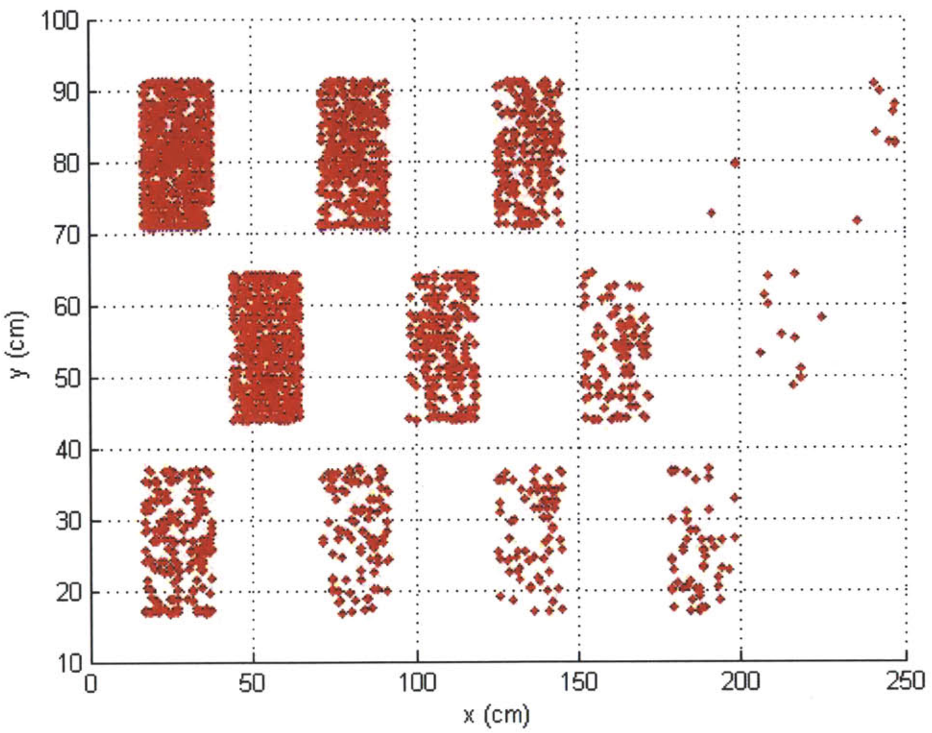

Figure 17: Source distribution after 3000 cycles for run described in Figure 16

Despite the large fluctuations in the Shannon entropy afterwards, the stochastic oscillator claims the curve flattens around 1900 cycles. This does not seem right and inspection of the source distribution plot in figure 17 supports that the the source distribution is not converged at this point. To address this issue, a third condition was set on the stochastic oscillator when it was checking for convergence: that past the point at which the first two conditions were met the entropy could not rise or fall by more than a fixed percentage of the maximum possible entropy. This was referred to as the limited stochastic oscillator. Initially, it was hoped to tie this percentage to the entropy bin size but it was found this failed when used with very small bins, in particular when calculating the entropy over pin cell sized bins. The method that came closest to working when the limit was set to be $\frac{1}{N}log_2(N)$. This made some logical sense as might be argued to describe the 'average entropy' per bin given a uniform distribution of particles which could be seen as a background variation in entropy, though this was never explored rigorously because the method was set too tight a limit when using small bins, even if using normalized entropy values. In general though, this issue was only a problem in cases with relatively few particles per cycle which should be avoided anyway and this third condition was immediately true if the other two conditions were true.

3.6 Shannon Entropy as a Measure of Source Convergence Conclusions

Equipped with consistent methods to measure the convergence points of both the source distribution and Shannon entropy curves it is possible to confirm the graphical result seen earlier. Table 3 shows a comparison of convergence points for the source distribution curves and Shannon entropy curves for simulations using 100,000 particles per cycles and a variety of mesh bin sizes, as described on the table. Where the table describes an NxM mesh it means that there are NxM mesh bins along the long and short sides of the geometry.

	Mesh size				
w/ 6th Order all	3_3	12_5	Assembly	108x22	pin
Shannon Ent.	1931	2160	2440	2895	3121
Source Dist.	1724	1816	1974	2242	2463

Table 3: Comparison of convergence points from Shannon entropy and source distribution curves for runs using 100,000 particles per cycle and variously sized calculation bins

Two important points can be seen in this data. Firstly that the Shannon entropy curve always converges after the source distribution curve. This means that users estimating convergence of their simulations using the Shannon entropy will rarely if ever swap to active cycles before the source is converged. This result has seen repeated for other examples with more particles per cycle but those are not plotted here to allow for space. Secondly, there is evidence from the table that, as expected, both the Shannon entropy and source distribution curves converge after fewer cycles when the volume they are being calculates over is larger. This is consistent with the idea that local information is lost or hidden when using larger volumes, giving the false impression of convergence if the user then attempts to draw data from an area/volume that is on a smaller scale. This relationship is shown graphically in Figure 18.

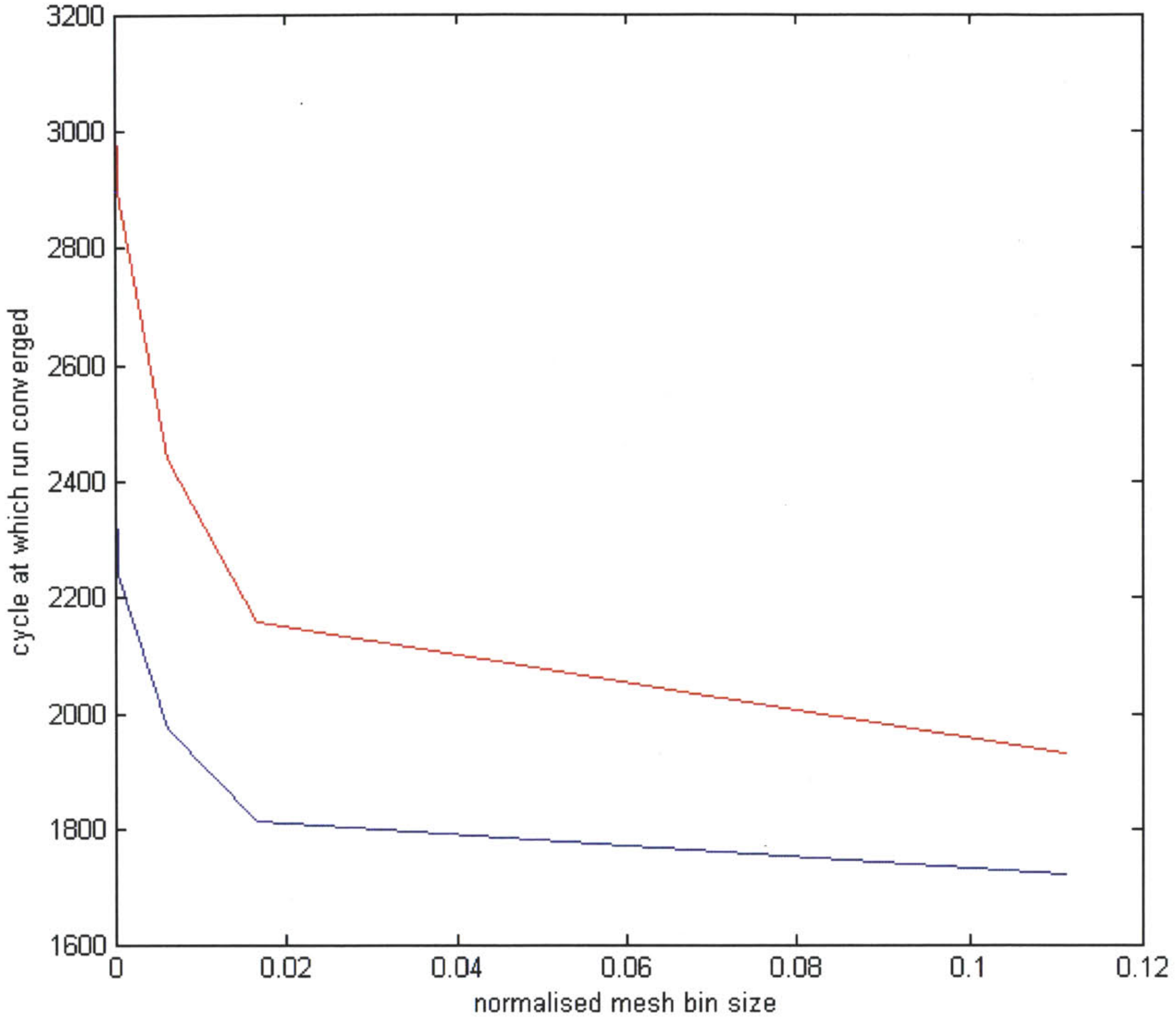

Figure 18: Comparison of points at which Shannon entropy (red) and source distribution (blue) curves claimed convergence had occurred for a variety of mesh sizes.

3.7 Undersampling

As well as investigating the link between source convergence and Shannon entropy, the objectives of this investigation included exploring undersampling in MCNP and other systems, something explored in work by others [10]but not characterized fully. This part of the project has not been carried to completion but the findings so far have raised interesting questions and possible avenues for future investigations and this section of the thesis will seek to go through those current results and questions.

As described earlier, undersampling was largely tested by comparing tally values for equivalent runs that consisted of the same total number of active particle histories but different numbers of inactive histories. Figure 19 shows the results from one attempt at this using MCNP and the OECD/NEA benchmark problem #1. The values plotted were calculated by taking average values of seven to ten results for the given tally using different random number seeds. This was done to get an impression of the true variance of the results. MCNP returns an estimate of the variance in the tallies but with so many particle histories run the result was always vanishingly small, with a relative error around 0.0002. It was expected that this curve would have a positive gradient and upwardly saturate and while this is arguably a general trend it is not what was seen specifically. The logic behind this expectation was that the tallies of neutron flux taken in any assembly with relatively high flux would have their value underestimated due to undersampling. This underestimating bias would be due to the fact that undersampling would reduce the ability of the source sites to move to the high flux region. creating an artificially low result. This effect should have been reduced as more particles per cycles were used.

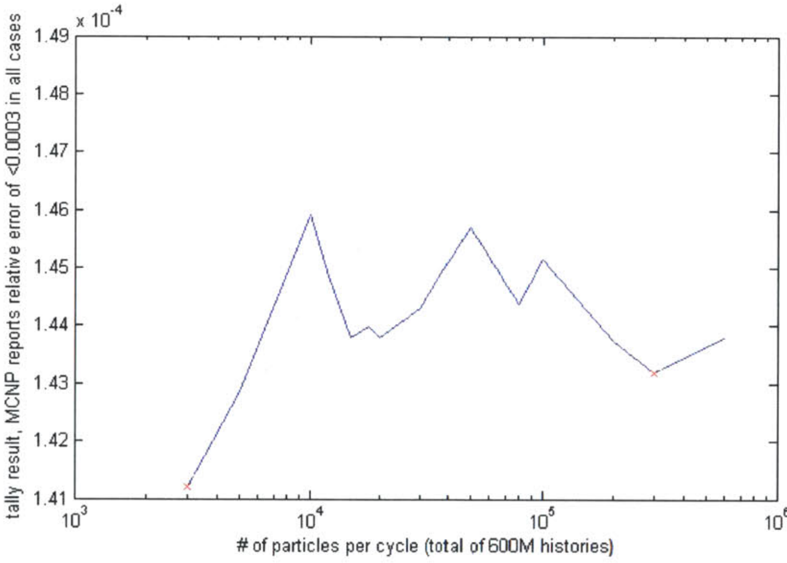

Figure 19: Undersampling measured by recording average tally values from the benchmark problem using increasing numbers of particles per cycle for a total of 600,000,000 active histories

It was not fully understood why this result was not as expected. After exploring multiple options, it was decided that a potential reason was the inherently high undersampling that is part of this benchmark problem making the problem very severe and hard to measure. The data for runs with fewer than 100,000 particles per cycle is potentially a bit shaky because the persistent large fluctuations in the Shannon entropy make it difficult to confirm source convergence and know that the number of inactive cycles run was sufficient. To mitigate this, the cases with fewer particles per cycle were run with a few thousand extra inactive cycles, however, in the 3000 and 5000 particles per cycle cases it is arguable that convergence is so slow that to all intents and purposes it never occurs.

To remedy these problems it was decided to use the OpenMC code, developed at MIT by Paul Romano, to test undersampling in relatively simpler geometries with much smaller dominance ratios. This part of the work is still ongoing, denying final results at this point though given the closer similarity to previous efforts (UEKI) it is thought to have a far higher chance of success.

In the meantime, efforts were made to draw more useful conclusions from the data that already existed. The above analysis was repeated using 3D tallies of different volumes in the source distributions of the runs performed. Figure 20 shows the results of that effort and while the plot is closer to the trend that was expected, it is still quite close to the results from Figure 19, suggesting that the problem is a real one and not simply created by not having enough data points to make accurate enough estimations of the tally values in Figure 19.

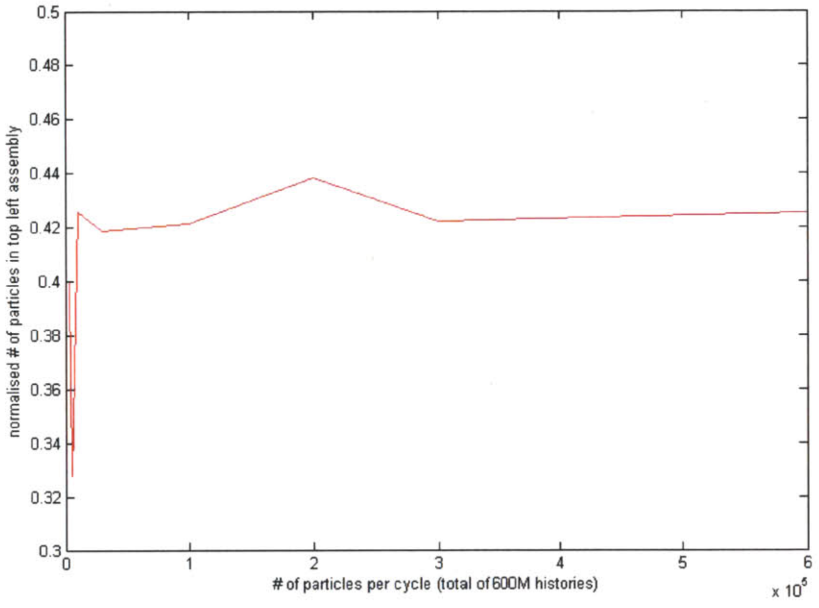

Figure 20: Undersampling measured by recording the number of particles in the upper left assembly in the benchmark problem using increasing numbers of particles per cycle for a total of 600,000,000 active histories

There was also a question of whether the Shannon entropy curve for a given simulation could predict in some way the presence of undersampling. Figure 21 shows the Shannon entropy curves for runs using 500 to 1M particles per cycle (in ascending order at the first batch) when calculated using assembly or pin sized bins. Given that the Shannon entropy describes the distribution of the source points it is arguable that the presence of undersampling would stunt the ability of the Shannon entropy to converge. This can be seen in Figure 21, where the 500 particles per cycle curve (the lowest one) hardly moves at all except due to random oscillation. This suggests that the source distribution hardly changed at all or that any movement towards the true flux profile was destroyed by random particle movement in the opposite direction or the manner in which MCNP renormalizes cases with Keff < 1 introduced too much random data compared to the amount being generated with each iteration.

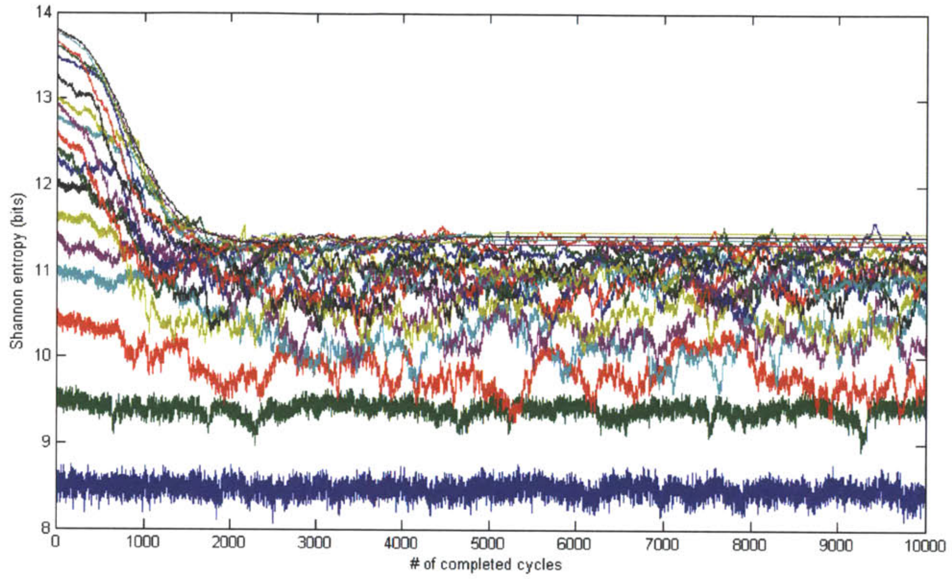

Figure 21: Shannon entropy curves for runs from benchmark #1 problem using 500 to 1M particles per cycle in ascending order. In ever case the Shannon entropy is calculated over pin cell sized bins

This explanation says that during each cycle a certain amount of information about the true source distribution is gained and eventually these small amounts are enough to make an accurate guess of the true distribution. However, say 100 of our 500 source particles move in the correct direction and produce some extra information but the Keff is 0.88192 +/- 0.00001 so 59 random particles will be put into the geometry $(500 * (1 - 0.88))$, significantly reducing the amount of useful information produced on aggregate. The same effect might also be seen if Keff was too high in comparison to the ratio of particles producing useful information because of the random destruction of too many of the useful particles as part of renormalization. This effect would likely be less prominent for runs with more particles per cycle as a greater proportion of particles would be producing useful information. This assessment could also be wrong if MCNP does not insert new particles using a uniform distribution. Information regarding this could not be found in the MCNP manual [1]or in other sources but it was thought that the uniform distribution was a good assumption as using any other typical distribution or one that reflects the current estimation of the source distribution would either insert too much incorrect information in the first case or reinforce existing trends too much in the latter.

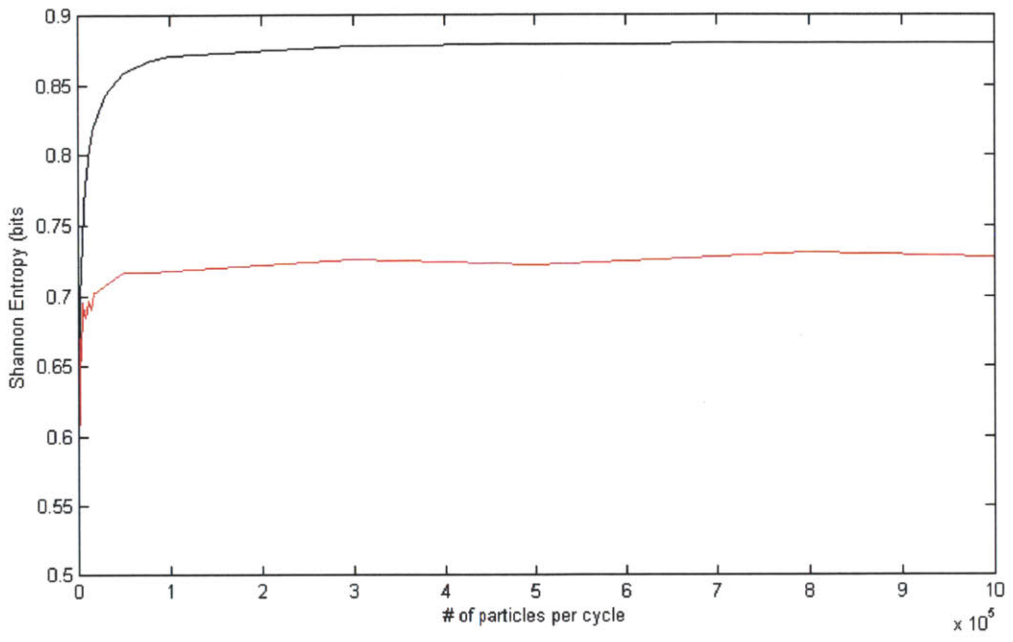

Figure 22: Comparison of normalized Shannon entropy values, calculated over pin cell-sized bins, at the beginning of a simulation (black) and after convergence (red) for an example using 100,000 particles per cycle

Returning to what was being said about Shannon entropy predicting the level of undersampling in a system, evidence was found of a link between the the number of particles per cycle and the Shannon entropy after one cycle and 5000 cycles (after most of the curves have converged). As can be seen from Figure 22, the Shannon entropy clearly grows along a saturating function and is roughly saturated for any run with more than about 200,000 particles per cycle. This concurs with some of what has been seen earlier that reliable results could only be found when using runs with more than 100,000 particles per cycle. Looking at the equivalent curve for the relative Shannon entropies after 5000 cycles on the same figure, we see that the Shannon entropy, a measure of the source distribution, is roughly equal for all values greater than again about 200,000 particles per cycle. This suggests at least tentatively that the beginning of simulation Shannon entropy can be used to predict undersampling. In this case the Shannon entropy was calculated over pin cell mesh bins. Figure 23 shows the equivalent curves calculated using a 7x3 Shannon mesh. While we see a similar curve after 5000 batches the curve at BOL is not as useful, saturating much earlier including some cases where the eventual Shannon entropy is not that similar to that of runs with greater than 300,000 particles per cycle.

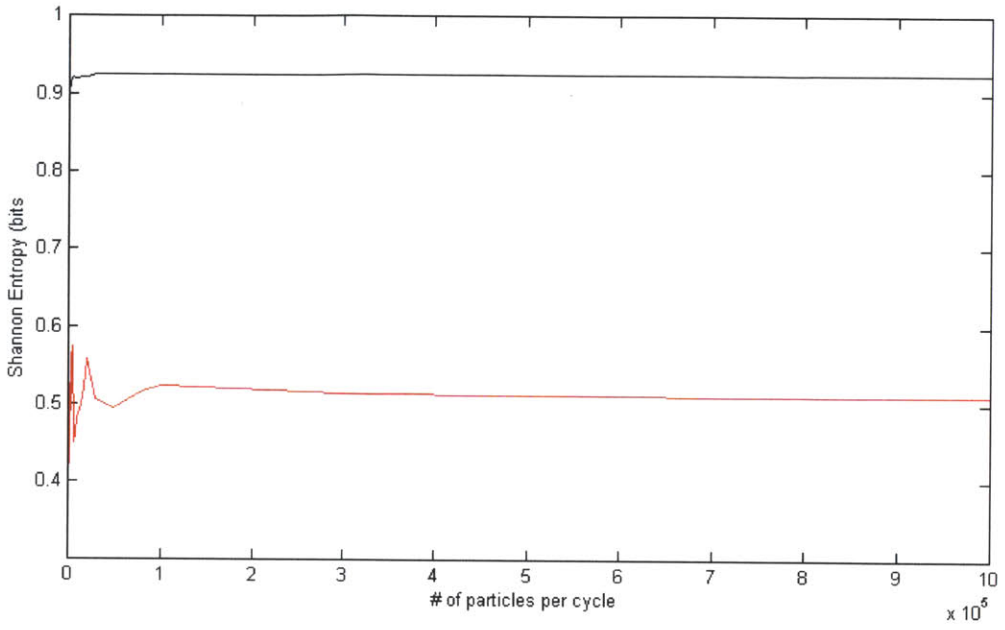

Figure 23: Comparison of normalized Shannon entropy values, calculated over a 7x3 grid, at the beginning of a simulation (black) and after convergence (red) for an example using 100,000 particles per cycle

Anytime a new simulation was run, a managing code like python could be used to run a handful of cycles with many millions of particles per cycle and if the early Shannon entropy results for the run the user was doing were too far from these it may indicate undersampling. There are no significant computational costs to running MCNP with smaller Shannon bins so using small bins to ensure this technique works should be fairly easy. To be sure of its usefulness, more work would have to be done to confirm the link between the Shannon entropies after convergence and determining at what threshold the method begins to work well with larger Shannon bins. As a first avenue of investigation, one reason that only smaller Shannon bins work might be that the effect is linked to the mean path length of the system. Similar analysis of fluid systems require that their mesh bins for analysis of various sort be sized on the order of the mean distance a fluid particle would travel in one time step. While this is not directly equivalent it's a possible explanation though there are immediate problems. Firstly that it would be quite difficult to calculate the mean path length for an entire model given how heavily that relies on the energy of the particles and secondly that these analyses are not entirely equivalent, though they are a good start.

3.8 Possible Algorithm for an on-the-fly Monte Carlo Code

Based on the work produced by this investigation, it is possible to imagine versions of MCNP or OpenMC with the ability to manage and make on-the-fly adjustments to many of its own inputs, reducing the burden on the code user and helping to ensure that results are accurate.

In this casd the user can make 'bigger picture' requests of the code and allow it to handle the details. For example, rather requiring the user to specifically ask for a simulation consisting of a certain number of active and inactive cycles, each using a certain number of particles per cycle, the user could simply determine how many active histories they would like to record information for. The code could then begin the simulation by comparing the magnitude of the Shannon entropy for a case with a modest number of particles per cycle with one many millions per cycle, both calculated over very fine meshes. By successively increasing the number of particles per cycle in the run till the Shannon entropy at the beginning of the simulation was within some range of that in the case with many millions of particles per cycle the code could determine how many particles are required per cycle to avoid undersampling. As long as this number was a few orders of magnitude smaller than the total number of histories asked for by the user, the code is now ready to begin the simulation.

The code would then run inactive cycles till convergence was noted based on the Shannon entropy curve and the limited stochastic oscillator. The Shannon entropy bins would be set by MCNP to be no bigger than the smallest tally defined by the user (MCNP already has a function to change the Shannon mesh fineness on-the-fly, though it is not accessible through the executable version). If convergence was not seen for a very long time a flag could be sent to the user and the code could increase the number of particles per cycle, eventually guaranteeing convergence as was shown in the section 3.1. After this point the code could switch to active cycles and collect the required number of active histories. Afterwards, the user or computer could use the final characteristics of the run, i.e. the number of particles per cycle, inactive cycles, etc. as the initial inputs into subsequent simulations using the same model, avoiding the need to repeat this potentially long process.

4 Conclusions

The purpose of this investigation was to make Monte Carlo simulations easier for users to operate as well as developing techniques to potentially offload some user responsibilities to the computer running the code so that it can make on-the fly adjustments to the run. The methods investigated here have been shown to work for a very difficult benchmark and suggests that they can be applied generally. The algorithm presented is not foolproof but lays the groundwork for more intelligent computational methods which reduce the burden on the user and may increase the use and accuracy of Monte Carlo analysis. While these methods may make an individual run longer, they should ensure that only one run is needed per task, reducing computational time overall.

Significant work is still to be done in developing these techniques and ensuring they are reliable and can be applied generally. In particular, further work is needed to explore methods to help users identify undersampling. However as an investigation intended to prove these techniques are possible, there has been evidence of each of the concepts and overall this investigation has met it's goals.

References

[1] Los Alamos National lab, MCNP5 Manual Vol 1

[2] Bo Shi, “Entropy-Based Diagnostics of Criticality Monte Carlo Simulation and Higher Eigenmode Acceleration Methodolgy”, Georgia Tech, August 2010

[3] T. Ueki and F. Brown, “Stationarity Diagnostics Using Shannon Entropy in Monte Carlo Criticality Calculation I: F Test”, Trans. Am. Nucl. Soc., 87, 156 (2002)

[4] T. Kitada and T. Takeda, “Effective Convergence of Fission Source Distribution in Monte Carlo Simulation,” J. Nucl. Sci. Eng., 38, 324 (2001)

[5] H. J. Shim and C. H. Kim, “Stopping Criteria of Inactive Cycle Monte Carlo Calculations,” Nucl. Sci. Eng., 157, 132 (2007).

[6] F. Brown et al., “Convergence Testing for MCNP5 Monte Carlo Eigenvalue Calculations,” Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007), Monterey, California, April 15-19, 2007, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2007).

[7] T. Ueki, “On-The-Fly Judgments of Monte Carlo Fission Source Convergence,” Trans. Am. Nucl. Sci., 98, 512 (2008)

[8] P. Romano, “Application of the Stochastic Oscillator to Assess Source Convergence in Monte Carlo Criticality Calculations, Intl. Conf. on Mathematics”, Computational Methods and Reactor Physics (M&C 2009)

[9] N. Smith et al. , “OECD/NEA Source Convergence Benchmark 1: Checkerboard storage of assemblies”, http://www.oecd-nea.org/science/wpncs/convergence/specifications/b1-checkerboard.pdf

[10] T. Ueki, “Information Theory and Undersampling Diagnostics for Monte Carlo Simulation of Nuclear Criticality”, Nuclear Science and Engineering, 151, 283-292 (2005).

[11] (C. E. Shannon, “A Mathematical Theory of Communication,” The Bell System Technical Journal, 27, pp. 379–423 (1948)

[12] P. Romano, B. Forget, "The OpenMC Monte Carlo Particle Transport Code," Submitted, Annals of Nuclear Energy (2012)