2nd International Serpent User Group Meeting
Madrid, Spain, September 19-21, 2012

Jaakko Leppänen
VTT Technical Research Centre of Finland
E-mail: Jaakko.Leppanen@vtt.fi

Summary and Notes

These notes summarize the observations, thoughts, and discussions carried out after the presentations, during the discussion session on Friday, as well as private communications with several participants. I will also try to provide explanations to some questions that remained open after the meeting. The topics are in no particular order.

Extending calculation methodologies from two- to three-dimensional applications

One of the main reasons for using Monte Carlo codes for neutron transport applications is their inherent capability to model complicated three-dimensional geometries, but due to the limitations in computational resources, these codes have not been used for routine tasks in reactor physics until fairly recently. Consequently, the methods and practices currently applied to reactor analysis have strong roots in two-dimensional transport theory, and extending tasks like homogenization and burnup calculation from two-dimensional assembly level geometries to three-dimensional systems has turned out to be far from straightforward. In a way, the capability to model complicated geometries has brought along a new set of complicated problems.

A good example of a problem encountered in three-dimensional geometries is the spatial instability of the burnup solution in a long LWR fuel assembly. These instabilities result from an interplay between several factors, but they can also be encountered in very simple models, and sometimes even using deterministic transport solutions. In the worst case the consequence of a spatial oscillation is that the results of the simulation have no value at all. The stability issues have been studied at Aalto University and KTH, and they were also reported in the presentation from BGU. There is not much that the Serpent user can do about the issue, but the studies continue.

Another application where moving from two- to three-dimensional systems changes the calculation scheme is group constant generation for fuel types with axial zoning. The traditional approach to homogenization is to assume that the neutronic properties in the reactor core change abruptly only in the radial directions. RBWR's and other novel reactor concepts, however, often involve assembly designs in which completely different fuel types are layered on top of each other, making the system as heterogeneous in the axial direction as a conventional UOX/MOX loaded PWR core is radially (see, for example, presentation from University of Michigan). The result of the axial heterogeneity is that the layers cannot be homogenized without accounting

for their surroundings above and below. Axial coupling between the nodes also necessitates the use of two additional discontinuity factors.

Instead of forcing an additional dimension in the methods and practices established for two-dimensional lattice transport codes, it might be worth the effort to revise the entire calculation scheme. Combining the continuous-energy Monte Carlo method to diffusion theory has not yet been fully accomplished, and the three-dimensional nature of the simulation should be kept in mind while developing new methods for calculating diffusion coefficients and performing leakage corrections.

Parallelization

The Monte Carlo method is generally known for its potential for almost linear scalability when run in parallel mode. In practice, however, there is a very simple theoretical limit, known as Amdahl's law, that determines the maximum speed-up S of a parallelized program when compared to a serial run:

$$S(N) = \frac{1}{(1-P) + P/N}\,, \tag{1}$$

where N is the number of parallel processes and P is the fraction of CPU time spent in the parallelized part of the program. Even though this formula is based on certain simplifications, it gives a good qualitative description of the problem. Figure 1 shows how attaining a good scalability for hundreds of processes requires a very large fraction of the program to be run in parallel, which is also what limits the scalability of the Monte Carlo simulation.

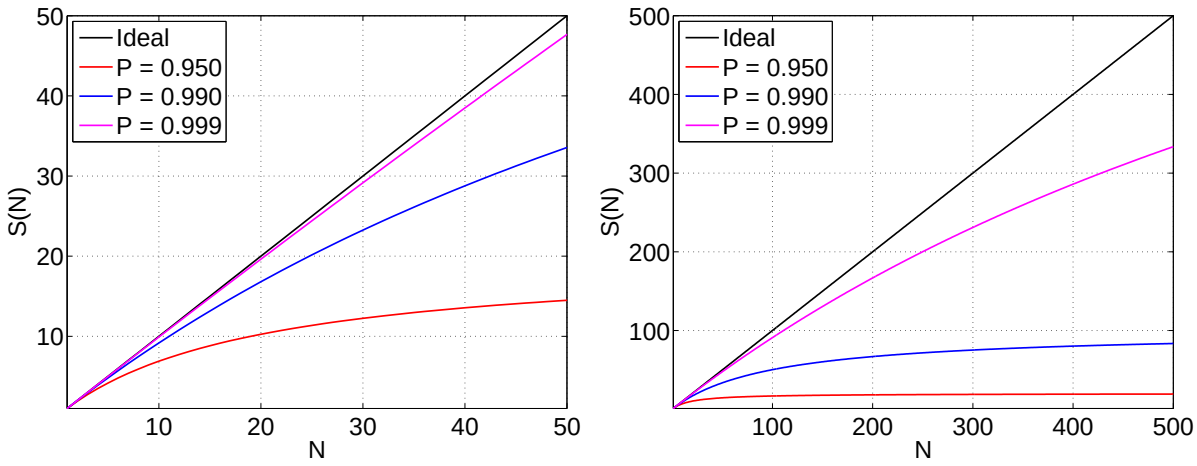

Figure 1: *Amdahl's speed-up factor for 95%, 99% and 99.9% parallelizations. Left: for up to 50 parallel processes. Right: for up to 500 parallel processes. A typical cluster node consists of 6 to 12 CPU cores, and parallelization beyond that is accomplished by coupling several nodes together. Techniques used for intra-node parallelization (OpenMP) differ from techniques used between the nodes (MPI), so in practice the scalability curves are not smooth functions of* N.

Parallelization in Serpent 1 is based on the Message Passing Interface (MPI), which means that each parallel task executes a separate copy of the program code, and communication is

handled by sending chunks of data between the tasks. The main drawback of this distributed approach is that the data arrays accessed by the code are not shared either, and memory usage is consequently multiplied by the number of tasks. For a code that already in serial mode requires several gigabytes of memory this means that the computational resources of multi-core CPU's cannot be fully utilized in practice.

The limitations due to excessive memory usage are overcome in Serpent 2 by using hybrid parallelization, based on the combination of OpenMP and MPI. Unlike MPI, OpenMP shares the same memory space between parallel threads, which means that all CPU cores within a single node can be assigned for the calculation without increasing the overall memory demand. Since data sharing cannot be extended beyond the computational unit, parallelization between separate nodes is handled using MPI.

The development and optimization of the parallel routines in Serpent 2 has so far mainly focused on OpenMP, while the MPI routines are taken almost directly from Serpent 1. The presentation by Professor Hoogenboom showed that the calculations scale reasonably well up to about 10 CPU cores, but beyond that there is considerable saturation in the performance. This observation shows that there is still some work to be done for the parallelization routines, especially when the calculation is divided between multiple nodes.

The scalability of OpenMP parallelization depends on several factors. The parallelization of the transport simulation is implemented by assigning each particle history with its own parallel thread. This means that P in Eq. (1) includes all operations performed on the particle history from the initial emission until the history is terminated by absorption, escape or cut-off. All CPU time that is spent between the simulated generations for collecting results, printing output, setting up the next criticality cycle, etc. decreases the value of P and the attained parallel speed-up. In general, calculation cases where the neutron histories are long tend to scale up better than cases where the particles are absorbed or leaked soon after they are born. This difference is seen, for example, when comparing LWR and HTGR calculations.

Another factor that may significantly affect the scalability of a criticality source simulation is the reproducibility of the random number sequence. The initialization of the random number generator in Serpent 2 is based on particle indexes, which are assigned to the neutron population at the beginning of each criticality cycle. The order in which new particles are stored in the fission bank is generally not the same in serial and parallel modes, and in order to preserve the random number sequences, the fission bank must be sorted before running the next cycle.[1] The CPU time required for sorting depends on population size, and also, on the number of threads, since the unsorted fission bank becomes more disordered as the number of threads is increased. OpenMP reproducibility can be switched off by input option:

```
set repro 0
```

Fission bank sorting is then omitted, which means better scalability, but the results will be slightly different each time the calculation is run.

[1] This problem is specific to the criticality source simulation mode. Neutron histories in an external source simulation are completely independent, and the random number sequence is automatically preserved.

Some improvement in scalability may also be attained by increasing the batching interval, i.e. the number of generations simulated before collecting the results. This interval is set to one by default, and the value can be changed by adjusting the fifth parameter in the "pop" card:

```
set pop <npop> <ncycles> <nskip> <keff0> <nbatch>
```

Scalability in burnup mode is also affected by the parallelization of depletion and processing routines. The speed-up is generally not as good as for the transport routine, so the overall performance depends on the number of burnable materials and the optimization mode used in the calculation.[2]

The presentation from RWTH Aachen University showed that, even though the implementation of MPI parallelization is made easy for the code developer by supplying a high-level programming interface with external libraries, hardware-related issues cannot be completely ignored. This is especially the case when MPI message size increases to tens of gigabytes, which is typical for Serpent burnup calculations. There is no computer scientist in the Serpent developer team, so feedback from users with the related expertise is considered extremely valuable.

All parallelization in Serpent is based on distributing the computational effort between several CPU's. Another interesting possibility is to use graphics processing units (GPU's) for the same purpose. This novel approach is making its way to scientific computing, including Monte Carlo particle transport simulation. Presentation form UC Berkeley showed examples of a simplified Monte Carlo program with very promising results. Implementing GPU parallelization in an existing code may require significant changes in the programming style and structure, but this possibility is certainly considered in the future development of Serpent as well.

Thermal energy groups in fast reactor cross section generation

Even though fast reactors operate well above the thermal energy region of neutrons, deterministic transport codes require multi-group cross sections that span the entire spectrum. Group structures used in fast reactor calculations usually reserve several groups for low-energy neutrons. When the group constants are produced using a Monte Carlo code, these groups receive a very low number of scores, and therefore suffer from poor statistics.

It was shown in the presentations from HZDR that this problem can be avoided by combining the lowest energy groups into a single group spanning the entire thermal region. A more universal solution, however, would be to use implicit methods to get more neutrons at low-energy. Serpent 2 does have an option for implicit capture, which should increase the number of neutrons slowing past the capture resonances of U-238. Most likely this method will turn out to be insufficient for this purpose, and the solution will require a weight-window type approach for the energy variable. This possibility will be studied in the future, when variance reduction techniques are implemented in Serpent 2.

[2] For the optimization modes, see: J. Leppänen and A. Isotalo, "*Burnup Calculation Methodology in the Serpent 2 Monte Carlo Code.*" In Proc. PHYSOR-2012.

Calculations of time constants

Serpent calculates various time constants, such as prompt neutron lifetimes (l_p) and neutron reproduction times (Λ) using implicit and analog estimators. There has been some confusion on how these parameters are calculated, and code versions prior to 1.1.18 and 2.1.8 handle vacuum boundary conditions in such way that the analog estimator of l_p can be significantly over-estimated. The methodology is not well documented, so I try to explain the basics here.[3]

The term "implicit estimator" refers here to parameters that are derived from other results, without direct relation to the transport simulation. The implicit prompt neutron lifetime is given by the text-book definition:

$$l_\mathrm{p} = \frac{1}{\overline{v}(\Sigma_\mathrm{f} + \Sigma_\mathrm{c} + L)} = \frac{k_\mathrm{eff}}{\overline{v}\nu\Sigma_\mathrm{f}}\,, \tag{2}$$

where $\overline{v}$ is the mean neutron speed, Σ_f and Σ_c are the fission and capture cross sections, respectively, and L is the leakage term (total leakage rate divided by total flux). All these parameters are calculated as the cross sections are homogenized over the geometry. Neutron reproduction time, which characterizes the time it takes for the neutron population to reproduce itself, is calculated similarly, as:[4]

$$\Lambda = \frac{1}{\overline{v}\nu\Sigma_\mathrm{f}} = \frac{l_\mathrm{p}}{k_\mathrm{eff}}\,. \tag{3}$$

The implicit prompt neutron lifetime given by Eq. (2) has been compared to MCNP results in a set of infinite lattice test cases, and the values are within the range of statistical accuracy from each other.[5] For some reason the results are generally not as good for fast critical benchmarks (Godiva, Jezebel, etc.), even though there is a perfect match for k_eff.

The analog estimators of prompt neutron lifetime and neutron reproduction time are calculated from the simulated neutron histories, using two simple definitions:

$$l_\mathrm{p} = \text{Average time from neutron emission to capture, fission or escape} \tag{4}$$

and

$$\Lambda = \text{Average time it takes for the neutron population to reproduce itself} \tag{5}$$

The calculation of (4) is extremely straightforward, but the earlier code versions had a problem defining the exact time of escape, due to the way the outer geometry boundary was handled. Also, while writing this, I now realize that the analog estimator used by Serpent for the neutron reproduction time is not that in (5). Instead, it is the average time between two consecutive fission events, which, using the terminology suggested by Lewins, should be called the "neutron generation time". I will try to fix this problem in the next updates.

[3] Meulekamp's method, which is used to calculate effective (adjoint-weighted) delayed neutron fractions, is well described in: R. K. Meulekamp and S. C. van der Marck, *"Calculating the Effective Delayed Neutron Fraction with Monte Carlo."* Nucl. Sci. Eng. **152** (2006) 142-148.

[4] The neutron reproduction time is sometimes called the neutron generation time, but there is a possibility for misconception, since the latter term is also used for the average time between two fissions. See: J. Lewins, *"Renaming the generation time the reproduction time."* Ann. Nucl. Energy, **33** (2006) 1071.

[5] See the validation reports at: http://virtual.vtt.fi/virtual/montecarlo/validation/lattice_calculations

It should also be noted that all parameters in Eqs. (2) – (5) are calculated as forward-, not adjoint-flux weighted averages.

Time-dependence and dynamic Monte Carlo

The difficulties related to the calculation of time constants were brought up in the presentation by Professor Wallenius from KTH, and there has been similar recent discussions with other users by e-mail. The different possibilities of simulating a time-dependent neutron transport process were also brought up. Since dynamic Monte Carlo simulation is also one of the work packages in the EU-HPMC project, from which Serpent development is partially funded, some considerable effort will be devoted to the topic during the next year.

As the first step towards the capability to simulate a time-dependent system, time binning will be added in detectors. This should be sufficient for running time-dependent external source simulations in sub-critical systems. Adding a time cut-off extends the capability to super-critical systems, but only for simulating relatively short transients. There are also other ways of coping with the exponentially growing neutron population, such as implicit treatment of fission reactions, which will be studied in the future.

In a criticality source simulation, time dependence (of the fundamental mode solution) can be handled using the α-eigenvalue method,[6] which is available, for example, in MCNP4C. Similar methodology was implemented in an earlier version of Serpent 1 as well, but the results were not completely satisfactory, and since there was no apparent use for the method at that time, the idea was dropped. The α-eigenvalue simulation mode will be revisited and, if found useful, implemented in Serpent 2.

One of the main difficulties in dynamic Monte Carlo is extending the simulation to such time scales that delayed neutron emission becomes significant. Methods have been developed for this type of problems at Delft University of Technology[7], and implementation of these methods to Serpent will be studied in the framework of the EU-HPMC project.

Systematic discrepancy in Pu-239 concentrations compared to other codes

A systematic over-prediction in the concentration of Pu-239 compared to CASMO-4E results in a PWR assembly burnup calculation was observed when the built-in depletion solver was first implemented in Serpent 1.[8] The cause of this discrepancy has not yet been resolved, but it has been suspected to result from either methodological differences or spectral effects from

[6] An excellent introduction to time dependence in transport theory and Monte Carlo simulation is found in: D. E. Cullen, et al., *"Static and Dynamic Criticality: Are They Different?"* UCRL-TR-201506, Lawrence Livermore National Laboratory, 2003. Available on-line at: http://home.comcast.net/ redcullen1/reports.htm

[7] See several publications related to the Ph.D. work of Bart Sjenitzer, for example: B. L. Sjenitzer and J. E. Hoogenboom, *"General Purpose Dynamic Monte Carlo with Continuous Energy for Transient Analysis."* In Proc. PHYSOR-2012.

[8] J. Leppänen and M. Pusa, *"Burnup Calculation Capability in the PSG2 / Serpent Monte Carlo Reactor Physics Code."* In Proc. M&C 2009.

fission product build-up, since some differences are also seen in the concentrations of Xe-135 and Sm-149.

The presentation from HZDR showed similar results for Pu-239 concentrations in SFR calculations, when Serpent was compared to HELIOS. In this particular case the differences can be attributed to the resonance parameters of Na-23 and O-16, as the cross sections were prepared using a generic LWR spectrum without resonance self-shielding treatment for the two nuclides. Even though the calculation case differs significantly from the earlier CASMO comparison, it shows how sensitive the build-up rate of Pu-239 is to spectral effects.

New cross section data type for burnup calculations

In a typical burnup calculation Serpent tracks the concentrations of 1200-1700 nuclides for each burnable material. Of these nuclides, less than 300 have ACE format cross sections available for the transport simulation, which for the burnup solution means that most of the nuclides are assumed to undergo only decay reactions. The list of available cross sections probably covers the vast majority of important reactions, but the truncation of transmutation paths due to missing data inevitably results in some loss of information.

There exists neutron data libraries, such as TENDL2011[9], which contain cross sections for over 2300 nuclides. This data could, in principle, be used in Serpent burnup calculations, but increasing the number of nuclides with cross sections by almost a factor of 10 would most likely hang the entire calculation, either because of excessive memory usage, or due to dramatically increased processing time.

An alternative solution, currently being planned, is to use these cross sections only for producing one-group transmutation constants for the burnup matrix, without actually including the data in the transport simulation. Similar approach is used in various coupled Monte Carlo burnup calculation codes, in which the external depletion solver reads pre-generated one-group cross sections from a secondary data library for nuclides that are not involved in the Monte Carlo simulation.

New cross section libraries for Serpent 2

The development of gamma and coupled neutron/gamma transport capability also brings the need to generate new cross section libraries for Serpent 2. These libraries will include gamma transport data and gamma production cross sections for neutrons, as well as heating and radiation damage cross sections required, for example, for certain multi-physics applications. In addition, zero-Kelvin neutron transport libraries will be included to be used with the on-the-fly temperature treatment routine and DBRC. Any other needs should be reported to the developer team (preferably sooner than later).

[9] The TENDL data libraries can be downloaded from: http://www.talys.eu

Multi-physics interface

The four-year NUMPS-project (Numerical Multi-Physics), recently started at VTT, will focus on developing calculation methods for the coupling of Monte Carlo neutronics and computational fluid dynamics (CFD). As a part of this project, a new interface is being developed for the external coupling of Serpent into thermal hydraulics and fuel performance codes.[10] This interface is not intended to be limited to the couplings studied in the framework of the NUMPS project, and all users working with multi-physics applications are encouraged to use it for passing data between the coupled codes.

The interface is based on the idea that density and temperature distributions can be passed into Serpent transport routines without any modifications in the geometry model. In addition to the simplified coupling scheme, the interface offers the capability to model continuous distributions for the physical state variables. Power distributions, which are needed for completing the coupling, are automatically calculated by Serpent, and written in a separate output file. The work has already begun, and the interface is available for use in the most recent Serpent 2 update. Near-future work involves developing a new interface type based on an unstructured mesh, specifically designed for coupling with CFD codes.

Fuel cycle analysis

There are several users who are performing complicated fuel cycle analyses that require capabilities beyond what is currently available in Serpent. The presentation from UC Berkeley introduced ADOPT, an automated calculation tool for optimizing the design parameters of fast reactor fuel assemblies. ADOPT uses Serpent as a neutronics and depletion solver. The calculation is currently based on external coupling, but work is on the way for implementing the automated equilibrium cycle search methodology (used in the BEAU code) as an integral part of the Serpent burnup routine. Similar work has been carried out at Politecnico di Milano, as a part of developing the capability to model the continuous reprocessing and refueling of molten salt reactors.

Even though the two applications presented at the meeting represent completely different reactor technologies, they share certain similarities at the code level. Other planned features related to fuel cycle analyses include branch calculations, fuel shuffling, and reactivity control by control rod movement and boron dilution. These new features do not involve difficult theoretical considerations, but they can complicate the burnup routine quite considerably, especially when combined with advanced time-integration methods and techniques developed for stabilizing spatial oscillations.

It is therefore recommended that some form of collaboration or information exchange is formed between the different groups working with these methods. This is not only to ensure the novelty of doctoral theses based on the work, but also to simplify the integration of the developed calculation routines with the other parts of Serpent source code.

[10] This interface will be presented at the upcoming ANS Winter Meeting in November in San Diego.

International co-operation

Serpent has been in public distribution for just over three years now, and feedback from users has proven extremely valuable for code development. During that time the user community has also reached a level of maturity, and there are several universities and research organizations where Serpent is not only used for calculations, but also as a platform for developing new methods (see, for example, the previous section on fuel cycle analysis). Collaboration within the user community is currently not coordinated as as well as it could be, and we are looking into different possibilities of improving the situation.

List of other actions

Below is a list of other actions and near-term modifications to be implemented in the source code (in no particular order).

1. Hexagonal and other mesh types for detectors (Serpent 2)
2. MPI message chunk size as a user variable (Serpent 1&2)
3. Radial zoning in the multi-physics interface output for types 1-3 (Serpent 2)
4. Calculation of analog neutron reproduction time will be fixed and all time constants revised (Serpent 1&2)
5. Time dependence in detector capabilities and time cut-off in external source simulation mode (Serpent 2)
6. Energy dependence in isomeric branching ratios (Serpent 2)
7. Capability to pass random number seed as an environment variable (Serpent 1&2)
8. Capability to read, write and combine statistics from multiple runs (Serpent 2)
9. Capability to use converged fission source distribution from previous burnup step as the initial guess for the next step (Serpent 2)
10. A repository will be set up for uploading documents, scripts, examples and test cases for code validation
11. The experimental fuel cycle routine “ucbburnupcycle.c” will be documented at the discussion forum (Serpent 2)
12. Temperature distributions will be included in the multi-physics interface (Serpent 2)
13. More examples will be provided for the multi-physics interface (Serpent 2)
14. Unnecessary memory allocation for reaction lists in materials with identical initial compositions will be fixed (Serpent 2)