EPJ Nuclear Sci. Technol. **10**, 19 (2024)

https://doi.org/10.1051/epjn/2024025

EPJ Nuclear Sciences & Technologies

Available online at:
https://www.epj-n.org

Status and advances of Monte Carlo codes for particle transport simulation, edited by Andrea Zoia, Cheikh Diop and Cyrille de Saint Jean

REGULAR ARTICLE

OPEN ACCESS

Status of Mercury and Imp: Two Monte Carlo Transport Codes Developed Using Shared Infrastructure at Lawrence Livermore National Laboratory

Michael Pozulp*, Bret Beck, Ryan Bleile, Patrick Brantley, Shawn Dawson, Nick Gentile, Evan Gonzalez, John Grondalski, Michael Lambert, Michael McKinley, Matthew O'Brien, Richard Procassini, David Richards, Alex Robinson, Scott Sepke, David Stevens, Richard Vega and Max Yang

Lawrence Livermore National Laboratory, 7000 East Ave., Livermore, CA 94550, USA

Received: 10 June 2024 / Received in final form: 28 June 2024 / Accepted: 3 October 2024

Abstract. The Monte Carlo Transport Project at Lawrence Livermore National Laboratory develops two Monte Carlo transport codes used in production by a sizable internal user community. Mercury is a Monte Carlo particle transport code used to model the interaction of neutrons, gammas, and light ions with a material. Imp is an implicit Monte Carlo thermal x-ray photon transport code used to model the interaction of x-ray photons with a material. This paper describes the two codes and highlights recent developments.

1 Introduction

What is the nuclear reaction energy deposited in a material irradiated by neutrons? What is the intensity of soft x-rays in a laser fusion experiment? These are examples of questions scientists at Lawrence Livermore National Laboratory (LLNL) answer by running Monte Carlo (MC) transport simulations. The former may be answered by running Mercury, which is an MC particle transport code used for modeling the interaction of neutrons, gammas, and light ions with a material [1]. The latter may be answered by running Imp, which is an implicit Monte Carlo (IMC) [2] thermal x-ray photon transport code [3] used for modeling the interaction of x-ray photons with a material.

Development of the Mercury Monte Carlo particle transport code was started by Richard "Spike" Procassini in the early 2000s [1]. A series of conference overview papers have documented the ongoing development of Mercury over the last approximately twenty years [4–10]. The Imp IMC thermal x-ray photon transport code [3] was developed starting in late 2015 using infrastructure shared with the Mercury code, a development decision that was largely motivated by advanced computing architecture considerations.

In this paper, we review the basic physics models in the Mercury and Imp codes, describe some of the important features of the two codes, and describe the programming models used to target modern diverse computing architectures in Section 2. We next review in Section 3 the recent history of code development in the two codes, with a focus on the use of new nuclear data formats in Mercury, motivations for the development of the Imp IMC code, and ongoing work to port the codes to graphics processing unit computing architectures. We conclude with a discussion and suggestions for future work in Section 4.

2 Code physics models, features, and programming models

The Mercury Monte Carlo particle transport code and the Imp thermal x-ray photon transport code are developed using shared infrastructure. The codes consist of about 240 000 lines of uncommented C++ source code (370 000 with comments), 80% of which is shared, 15% of which is Mercury-specific, and 5% of which is Imp-specific.

2.1 Mercury neutral particle physics models

Mercury can track two neutral particle types, neutrons and gamma photons. Mercury solves the neutral particle Boltzmann transport equation [11–14],

$$\begin{aligned}\frac{1}{v}\frac{\partial \Psi}{\partial t} = &-\Omega\cdot\nabla\Psi(r,\Omega,E,t) - \Sigma_{\mathrm{t}}(r,E)\Psi(r,\Omega,E,t)\\ &+\int \mathrm{d}E'\int \mathrm{d}\Omega'\ \Sigma_s(r,\Omega'\to\Omega,E'\to E)\Psi(r,\Omega',E',t)\\ &+\int \mathrm{d}E'\int \mathrm{d}\Omega'\ \chi(E'\to E)\nu(E')\Sigma_f(r,E')\Psi(r,\Omega',E',t)\\ &+S(r,\Omega,E,t)\ , \end{aligned} \tag{1}$$

where $r = (x, y, z)$ is the position of the particle, $\Omega = (\Omega_x, \Omega_y, \Omega_z)$ is the direction the particle travels, E is the

* e-mail: pozulp1@llnl.gov

particle's energy, and t is the time. The quantity Ψ, the angular flux, is the product of the phase space number density $n(r, \Omega, E, t)$ and the particle speed v, so $\Psi = vn$, and its dimensions may be written as: # /(cm^2 MeV sr s), which is convenient for computing reaction rates in the material. $\Sigma_t(r, E)$ is the macroscopic total nuclear reaction cross-section, $\Sigma_f(r, E)$ is the macroscopic fission cross-section, and $\nu(E)$ is the average number of neutrons released per fission. The quantity $\chi(E)$ is the fission spectrum. Finally, $S(r, \Omega, E, t)$ is an external source. Equation (1) requires appropriate initial and boundary conditions.

Equation (1) is a general form of the Boltzmann transport equation, and not all terms are present in all problems and for all particle types. For example, the temporal derivative is omitted for steady-state problems. In addition to the fully time-dependent form of the transport equation above, the transport equation can take on some special forms such as for k-eigenvalue calculations.

Mercury can run steady-state or time-dependent calculations. Mercury can additionally solve for the k-eigenvalue or α-eigenvalue of the system as well as the probability of initiation or extinction [15,16].

2.2 Mercury-charged particle physics models

Mercury can also track five charged particle types: protons, deuterons, tritons, alphas, and helium-3 particles. Unlike neutral particles, charged particles do not lose energy only in discrete collisions, because charged particles continuously interact with the electrons and ions of the medium in which they are traveling. We employ a continuous slowing-down approximation to calculate the effect of the interactions of the charged particles with the medium. The energy lost by charged particles due to this continuous interaction often exceeds the energy lost through discrete nuclear reactions. In order to calculate the slowing down rate and distance to thermalization, one needs to calculate the Coulomb logarithm and additional parameters required for the slowing down model. Mercury implements thirteen different slowing-down models.

When the energy of a charged particle has decreased until it is approximately that of the surrounding medium, the charged particle is considered "thermalized" into the background medium. Whereas neutral particles only lose energy in collisions, which are a tracked event, charged particles lose energy continuously when they move, and can therefore cross an energy group boundary without a collision event. Neutral particles already have an event that marks the occurrence of an energy boundary crossing – the collision that results in energy loss. Since the energy loss for charged particles is continuous, the energy boundary crossing itself needs to be an event so that it can be properly handled. The calculation of the distance to the crossing of the lower energy bound mirrors how the distance to thermalization is calculated, except with a different final energy under consideration, E_{low} instead of E_{thermal}, where E_{low} is the energy of the lower boundary of the particle's current energy group.

2.3 Imp photon physics models

The one particle type that Imp tracks, the IMC photon, is also a neutral particle type. Imp solves a neutral particle Boltzmann transport equation coupled to a material energy balance equation [17]:

$$\frac{1}{c}\frac{\partial I}{\partial t} = -\Omega \cdot \nabla I(r, \Omega, \nu, t) - \sigma_{\text{a}}(r, \nu, T)\, I(r, \Omega, \nu, t) + \sigma_{\text{a}}(r, \nu, T)\, B(\nu, T), \tag{2a}$$

$$\frac{\partial E_m(r,t)}{\partial t} = \rho(r,t) c_{\text{v}}(r,T) \frac{\partial T(r,t)}{\partial t} = \int \mathrm{d}\nu' \int \mathrm{d}\Omega'\, \sigma_{\text{a}}(r, \nu', T)\, (I(r, \Omega', \nu', t) - B(\nu', T)), \tag{2b}$$

where we have omitted scattering and external sources in equation (2a). The particle speed v and energy E in equation (1) have been replaced with the photon speed c (the speed of light in a vacuum) and photon frequency ν. Instead of solving for Ψ, we solve for I, which is the photon intensity. The photon intensity is the product of the phase space number density $n(r, \Omega, \nu, t)$ and the photon speed c and energy $h\nu$, so $I = ch\nu n$, and its dimensions may be written as energy/(cm^2 Hz sr s), which is convenient for computing energy deposition rates in the material. The quantity T is the material temperature, σ_a is the absorption opacity, and B is Planck's function,

$$B(\nu, T) = \frac{2h\nu^3}{c^2} \frac{1}{\exp\left(\frac{h\nu}{kT}\right) - 1}, \tag{3}$$

where h is Planck's constant and k is Boltzmann's constant. The quantity E_m in equation (2b) is the material energy density, ρ is the mass density of the material, and c_v is the specific heat capacity of the material. Equation (2a) requires appropriate initial and boundary conditions. Equation (2b) requires an appropriate initial condition.

The system of equations (2a) and (2b) is nonlinear due to the Planck emission term B. We refer to Imp as an IMC code because we use the standard implicit Monte Carlo multigroup algorithm for thermal x-ray photon transport to linearize and solve the system [2,18]. The linearization produces an "effective" scattering term. If an IMC photon undergoes an effective scatter, then the physical photon to which the IMC photon corresponds was absorbed and reemitted within the time step. IMC photons traversing optically thick, highly absorbing media are very expensive to track because they undergo a large number of effective scatters. Imp uses random walk acceleration [19] to ameliorate this problem as well as source tilt to ameliorate teleportation error [18].

2.4 Code features

Some of the important general features shared by both Mercury and Imp are a powerful input syntax, an embedded Python interpreter used for input parsing and code

steering, support for both combinatorial geometry (constructive solid geometry) and meshes, a flexible user-defined tally infrastructure, support for many types of particle sources and variance reduction techniques, checkpoint/restart, domain decomposition, dynamic load balancing, writing graphics for visualization with VisIt, ray trace geometry rendering, and the ability to run on central processing units (CPUs) as well as graphics processing units (GPUs). Mercury additionally supports the following nuclear reaction cross-section features: multigroup or continuous energy, heating, photonuclear, thermal neutron scatter law, and interprocess shared memory.

Most code features have a default behavior that matches the most common use case. A question often asked by new users exemplifies default behavior: what does Mercury do if a nuclear reaction produces a particle type that Mercury is not tracking (because the user does not want to track that particle type) or cannot track (because the particle type is not one of the seven trackable particle types)? Mercury deposits the particle's energy in the material at the location of the reaction[1]. The user can override this default behavior so that the energy is considered leaked. The user would create an `Escape` functional triggered on the `Creation_Collision` particle event, which can be achieved by adding the lines in listing 1 to the input file[2].

In addition to reading an input file, the two codes accept environment variables for certain behaviors that cannot be achieved without them. The codes also accept a few command line arguments. One of the most useful is `-def key=value` which uses the embedded Python interpreter to evaluate `value` and assign it to a `key`. Users can combine this with the `mc_ifundef` input syntax as in listing 2 to create defaults for user-defined variables in input files.

Listing 1. Leak untracked particles

```
tally
    tal Leak
        event
            Creation_Collision
        end_event
        functional
            type Escape
        end_functional
    end_tal
end_tally
```

Listing 2. Command line defines

```
$ head -1 foo.inp
mc_ifundef photons {1e6}
$ imp foo.inp
$ imp foo.inp -def photons=1e7
$ imp foo.inp -def photons=1e8
```

[1] Unless the particle is a neutrino, in which case the energy is considered leaked.

[2] Listing 1 paraphrases the actual input. We omitted the `response...end_response` block.

2.5 Programming models

Mercury and Imp run in production on x86 servers using distributed- and shared-memory parallelism provided by MPI+OpenMP. We also use the MPI+OpenMP programming model to target ARM, POWER, and other CPU architectures. We target the Nvidia V100 GPU architecture, which Nvidia launched in 2017 and LLNL purchased for the Sierra supercomputer [20], using MPI+CUDA, and we have observed speedups of 7.6x for Mercury neutronics and 5.8x for Imp photonics in node-to-node comparisons with x86 servers of similar vintage [21]. Finally, we target the AMD MI300 accelerated processing unit (APU) architecture, which AMD launched in 2024 and LLNL purchased for the El Capitan supercomputer [22], using MPI+HIP/ROCm. We write HIP/ROCm to indicate that we use ROCm underneath HIP.

3 Recent history of code development

Figure 1 shows a timeline of selected code development events from the past fifteen years. We wrote the first line of Mercury source code more than twenty years ago, whereas we began development of the Imp IMC code less than ten years ago in late 2015. The events in the timeline can be partitioned into three topics: (i) use of the Generalized Nuclear Database Structure (GNDS) nuclear data file format in Mercury, (ii) development of the Imp IMC code, and (iii) porting production MC code to GPUs. Sections 3.1–3.3 describe these three topics, to which we have dedicated much of our code development effort in the last ten years.

Due to space limitations, we have focused on the development activities described above. Omitted from the timeline are events related to transitioning our source code version control system from svn to git, transitioning our embedded Python interpreter from Python 2 to Python 3, and Mercury and Imp physics capability development activities related to transport in stochastic media [3,23–25], capabilities in the two codes that are consistent as a result of the shared infrastructure.

3.1 Using the Generalized Nuclear Database Structure (GNDS)

Mercury uses a nuclear data library, along with random numbers, to determine whether an MC particle will undergo a collision in a material, which isotope the particle collides with, which reaction occurs, and then the types, quantities, energies, and directions of outgoing particles. Mercury supports two nuclear data formats: the legacy Evaluated Nuclear Data Library (ENDL) format [26], which was developed at LLNL [27] before the more widely adopted Evaluated Nuclear Data File (ENDF) format [28], and the Generalized Nuclear Database Structure (GNDS) format [29]. The GNDS specification [30] is an open standard maintained and publicly distributed by the Nuclear Energy Agency (NEA) of the Organisation for Economic Co-operation and Development (OECD).

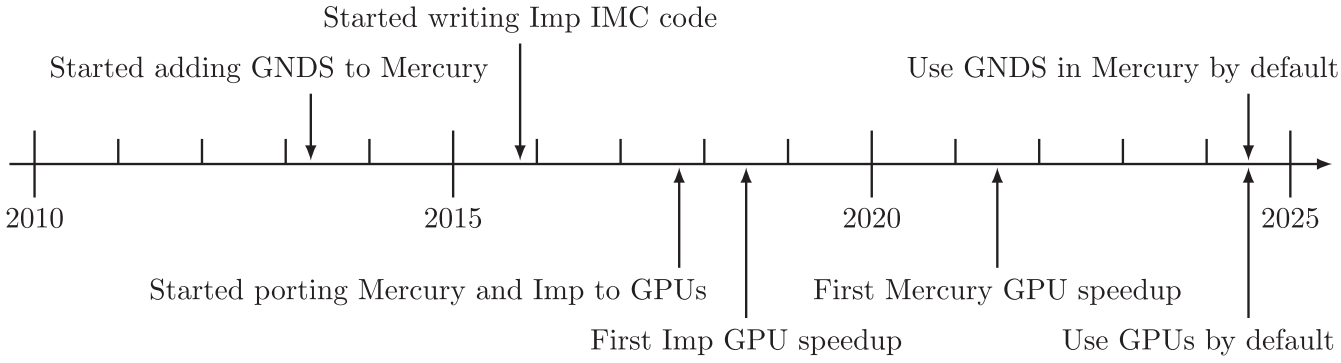

Fig. 1. Recent history of Mercury and Imp code development. All events below the axis involve graphics processing units.

Mercury accesses nuclear data through an abstraction layer that we call a collision library. A legacy collision library [31] allows Mercury to use nuclear data in the legacy ENDL format. For the GNDS format, we use the Monte Carlo General Interaction Data Interface (MCGIDI) collision library, which uses the GIDI library to read GNDS files into memory. The data flow is thus: Mercury ← MCGIDI ← GIDI ← GNDS. The OECD NEA publicly distributes the GNDS specification [30], and LLNL publicly distributes the source code for both GIDI and MCGIDI, along with supporting libraries, in a package called GIDI+ (pronounced "gidiplus") [32]. The GIDI/MCGIDI interface is also used by the Geant particle simulation toolkit which calls GIDI and MCGIDI to access GNDS-formatted nuclear data [33].

Issues that we encountered with the ENDL and ENDF formats [34] motivated us to undertake the effort to use GNDS in Mercury [35]. When we decided to use GNDS we also decided not to port our legacy collision library to advanced computing architectures, which means that Mercury calculations on GPUs always use GIDI+.

3.2 Development of the Imp IMC code

The development of Imp was strongly influenced by the algorithmic implementations demonstrated in the Kull IMC photon transport capability [36], which is itself a production IMC code. So why develop Imp? We realized that most Mercury code was infrastructure, not physics, and this is evidenced today by the fact that 80% of our Mercury-and-Imp codebase is shared by the two codes. Shared code enables:

1. GPU, APU, and other porting efforts to benefit both Mercury and Imp,
2. Mercury and Imp users to enjoy a consistent feature set (see Sect. 2.4), and
3. project staff to support users by learning one codebase instead of two.

A feature exemplifying the second point is our dynamic load balancer. We completed our dynamic load balancer implementation in Mercury over ten years ago [37–39]. A couple of years later, on the first day that we started writing Imp, we could already run domain-decomposed IMC calculations with a verified and robust dynamic load balancer.

We verified and tested Imp by running calculations and comparing them to analytic solutions, benchmark solutions, and Kull IMC package calculations [3]. One interesting project that we have recently undertaken in Imp involves modeling the Dante x-ray spectrometer [40] using next-event estimators [41]. Another project involves tracking IMC particles on curved high-order finite element meshes and tallying polynomial photon energy deposition which varies within a mesh element, instead of piecewise constant energy deposition which is constant within a mesh element. This capability will enable high-order radiation-hydrodynamics calculations in which we run Imp IMC photon transport on the same MFEM mesh [42] used by the Blast hydrodynamics package [43] from the MAPP project [44].

Finally, we are also researching approaches to accelerate calculations involving high amounts of effective scattering. Imp's random walk acceleration [19] can provide some speedup. Acceleration approaches such as Implicit Monte Carlo Diffusion (IMD) [45] and Discrete Diffusion Monte Carlo (DDMC) [46] have demonstrated significantly larger speedups. We are beginning development activities aimed at implementing such acceleration approaches.

3.3 Porting production MC code to GPUs

There is one GPU architecture and one APU architecture on which we run Mercury and Imp. They are the Nvidia V100 GPU and the AMD MI300 APU, which are the architectures in the LLNL Sierra and El Capitan supercomputers, respectively [20,22]. Before we started porting our production MC codes to the V100 GPU, which launched in 2017, we spent several years writing small MC programs in order to test out the programming models then available [10,47,48]. Even with the preparation provided by those experiences, we found porting the Mercury and Imp production codes to the V100 to be challenging. We invested significant effort from 2018 to 2023 to improve our Mercury neutronics speedup from 0.47x to 7.61x and our Imp photonics speedup from 1.45x to 5.81x on V100 GPUs [21,49].

We have also researched hybrid CPU + GPU load balancing approaches on heterogeneous computing architectures for both fully replicated and domain-decomposed meshes [50–52]. We have just begun porting to MI300 APUs, which is an architecture that AMD released this year.

The most significant obstacle we faced in our production code GPU porting effort was large kernels. Our largest kernel has about 100 000 lines of reachable code. When we use the nvcc compiler from the Nvidia toolchain to compile this kernel for the V100, the compiler emits the maximum 255 registers per thread permitted by the V100 instruction set, which limits the theoretical occupancy of the kernel to 12.5% and achieved occupancy to about 10%. Occupancy on Nvidia GPUs is the percentage of the maximum number of warps that can run simultaneously on an SM. Low occupancy prevents the latency hiding that we need for our calculations to become memory-bandwidth bound. We attempted to solve this problem by splitting our large kernel into smaller kernels by implementing event-based particle tracking, which runs faster as long as the kernel launch latencies are small. We recently implemented a "persistent threads" [53] variation of history-based tracking and observed higher throughput at smaller problem sizes, but that approach to date has not increased the maximum throughput.

We were surprised by the performance impact of some language features. One example is separate compilation of device code, the impact of which we were able to quantify by using device code link time optimization (LTO) in the Nvidia toolchain and observing a 36% speedup at the cost of increased link time [21]. Another surprise was the impact of language features which do not allow the maximum stack size of a kernel to be determined until runtime. Examples in our codebase include recursion, virtual functions, and function pointers. We are in the process of refactoring to remove these features from device code.

4 Conclusions and future work

The production of Mercury and Imp Monte Carlo transport codes have a significant development history: Mercury is early in its third decade of development, and Imp is approaching the start of its second. During this time, a sizable internal user community at LLNL has used the two codes in production to answer numerous questions. The physics models, features, and programming models that we have implemented in the codes and described in this paper could remain useful for yet more decades.

The three recent code development topics described in this paper continue to occupy a substantial amount of our code development effort today. The sustained effort that we have invested in these areas is leading to higher-fidelity physics, additional users, and faster calculations.

Funding

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes. This research was funded by the U.S. Department of Energy's Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Conflicts of interest

The authors declare that they have no competing interests to report.

Data availability statement

This article has no associated data generated and/or analyzed.

Author contribution statement

Project administration: P.B. and R.P.; Software: M.P., B.B., R.B., P.B., S.D., N.G., E.G., J.G., M.L., M.M., M.O., R.P., D.R., A.R., S.S., D.S., R.V., and M.Y.; Writing – original draft: M.P. and P.B.; Writing – review & editing: M.P., B.B., P.B., M.M., and M.O.

References

1. R. Procassini, J. Taylor, I. Corey, J. Rogers, Design, Implementation and testing of mercury: a parallel monte carlo transport code, in *Proceedings of M&C 2003* (Gatlinburg, TN, USA, 2003)
2. J.A. Fleck, J.D. Cummings, J. Comput. Phys. **8**, 313 (1971)
3. P.S. Brantley, N.A. Gentile, M.A. Lambert, M.S. McKinley, M.J. O'Brien, J.A. Walsh, A new implicit monte carlo thermal photon transport capability developed using shared monte carlo infrastructure, in *Proceedings of M&C 2019* (Portland, OR, USA, 2019)
4. R. Procassini, D. Cullen, G. Greenman, C. Hagmann, Verification and validation of mercury: a modern, monte carlo particle transport code, in *Proceedings of MC2005* (Chattanooga, TN, USA, 2005)
5. R. Procassini, J. Taylor, S. McKinley, G. Greenman, D. Cullen, M. O'Brien, B. Beck, C. Hagmann, Update on the development and validation of mercury: a modern, Monte Carlo particle transport code, in *Proceedings of M&C 2005* (Avignon, France, 2005)
6. R. Procassini, D. Cullen, G. Greenman, C. Hagmann, K. Kramer, S. McKinley, M. O'Brien, J. Taylor, New Capabilities in mercury: A modern Monte Carlo particle transport code, in *Proceedings of M&C + SNA 2007* (Monterey, California, USA, 2007)
7. R. Procassini, P. Brantley, S. Dawson, G. Greenman, M.S. McKinley, M. O'Brien, S. Sepke, D. Stevens, B. Beck,

C. Hagmann, New Features of the mercury Monte Carlo Particle transport code, in *Proceedings of M&C + SNA 2010* (Tokyo, Japan, 2010)
8. P.S. Brantley, S.A. Dawson, M.S. McKinley, M.J. O'Brien, D.E. Stevens, B.R. Beck, E.D. Jurgenson, C.A. Ebbers, J.M. Hall, Recent advances in the mercury Monte Carlo particle transport code, in *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)* (Sun Valley, Idaho, 2013)
9. P.S. Brantley, S.A. Dawson, M.S. McKinley, M.J. O'Brien, D.E. Stevens, B.R. Beck, E.D. Brooks III, Advanced Computing architecture challenges for the mercury Monte Carlo particle transport project, in *Proceedings of ANS MC2015 - Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method* (Nashville, Tennessee, 2015)
10. P.S. Brantley, R.C. Bleile, S.A. Dawson, N.A. Gentile, M.S. McKinley, M.J. O'Brien, M.M. Pozulp, D.F. Richards, D.E. Stevens, J.A. Walsh, LLNL Monte Carlo transport research efforts for advanced computing architectures, in *Proceedings of M&C 2017* (Jeju, Korea, 2017)
11. G.I. Bell, S. Glasstone, *Nuclear Reactor Theory* edited by Van Nostrand Reinhold Company (University of Michigan 1970)
12. J.J. Duderstadt, L.J. Hamilton, *Nuclear Reactor Analysis* 1st edn. (Wiley, 1976)
13. J.J. Duderstadt, W.R. Martin, *Transport Theory* (Wiley, 1979)
14. E.E. Lewis, W.F. Miller, *Computational Methods of Neutron Transport* (Wiley, 1984)
15. G.M. Greenman, R.J. Procassini, C.J. Clouse, A Monte Carlo method for calculating initiation probability, in *Proceedings of M&C + SNA 2007* (Monterey, CA, USA, 2007)
16. M. McKinley, P. Brantley, Probability of initiation and extinction in the mercury Monte Carlo code, in *Proceedings of M&C 2013* (Sun Valley, ID, USA, 2013)
17. G.C. Pomraning, *The Equations of Radiation Hydrodynamics* (Dover Publications, Inc., Mineola, New York, 1973)
18. A.B. Wollaber, J. Comput. Theor. Trans. **45**, 1 (2016)
19. J.A. Fleck, E.H. Canfield, J. Comput. Phys. **54**, 508 (1984)
20. S.S. Vazhkudai, B.R. de Supinski, A.S. Bland, A. Geist, J. Sexton, J. Kahle, C.J. Zimmer, S. Atchley, S. Oral, D.E. Maxwell et al., *The Design, Deployment, and Evaluation of the CORAL Pre-Exascale Systems* (2018), SC '18
21. M. Pozulp, R. Bleile, P. Brantley, S. Dawson, M. McKinley, M. O'Brien, A. Robinson, M. Yang, Progress porting LLNL Monte Carlo transport codes to Nvidia GPUs, in *Proceedings of M&C 2023* (Niagara Falls, Canada, 2023)
22. *El capitan: Preparing for nnsa's first exascale machine* (2024), https://asc.llnl.gov/exascale/el-capitan
23. P.S. Brantley, Trans. Am. Nucl. Soc. **111**, 655 (2014)
24. P.S. Brantley, P.F. O'Rourke, A.K. Prinja, Verification of a Monte Carlo levermore-pomraning algorithm for spatially-inhomogeneous binary stochastic media, in *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2019)* (Portland, Oregon, USA, 2019)
25. E.H. Vu, P.S. Brantley, A.J. Olson, B.C. Kiedrowski, Benchmark comparisons of Monte Carlo algorithms for one-dimensional n-ary stochastic media, in *Proceedings of ANS M&C 2021 – The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering* (Raleigh, North Carolina, USA, 2021)
26. R.J. Howerton, R.E. Dye, S.T. Perkins, *Evaluated nuclear data library* (Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States) 1981)
27. R.J. Howerton, *Review of nuclear reaction data evaluation in the us* (Lawrence Livermore National Lab., CA (USA) 1985)
28. A. Trkov, D.A. Brown, *Endf-6 formats manual: Data formats and procedures for the evaluated nuclear data files* (Brookhaven National Lab. (BNL), Upton, NY (United States) 2018)
29. C. Mattoon, B. Beck, N. Patel, N. Summers, G. Hedstrom, D. Brown, Nucl. Data Sheets **113**, 3145 (2012)
30. NEA, *Specifications for the Generalised Nuclear Database Structure Version 2.0*, OECD Publishing, Paris (2023)
31. P.S. Brantley, C.A. Hagmann, J.A. Rathkopf, *MCAPM-C generator and collision routine (Gen2000/Bang2000) documentation (revision 1.2)* (2003)
32. B. Beck, C.M. Mattoon, *Gidiplus* (2023), https://github.com/LLNL/gidiplus
33. *Geant4* (2023), https://github.com/Geant4/geant4
34. C.M. Mattoon, B.R. Beck, *Wpec sg38: Designing a new format for storing nuclear data* (Lawrence Livermore National Lab., CA (USA) 2014)
35. M.S. McKinley, B.R. Beck, Implementation of the generalized interaction data interface (GIDI) in the mercury Monte Carlo code, in *Proceedings of M&C + SNA + MC 2015* (Nashville, TN, USA, 2015)
36. N.A. Gentile, N. Keen, J. Rathkopf, *The kull imc package* (Lawrence Livermore National Lab., CA (USA) 1998)
37. R.J. Procassini, M.J. O'Brien, J.M. Taylor, Load balancing of parallel Monte Carlo transport applications, in *Proceedings of Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications* (Avignon, France, 2005)
38. M.J. O'Brien, P.S. Brantley, K.I. Joy, Scalable load balancing for massively parallel distributed Monte Carlo particle transport, in *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)* (Sun Valley, Idaho, 2013)
39. M.J. O'Brien, Ph.D. Thesis, University of California, Davis, 2013
40. J.L. Kline, K. Widmann, A. Warrick, R.E. Olson, C.A. Thomas, A.S. Moore, L.J. Suter, O. Landen, D. Callahan, S. Azevedo et al., Rev. Sci. Instrum. **81**, 10E321 (2010)
41. T.E. Grubbs, A.P. Robinson, P.S. Brantley, Trans. Am. Nucl. Soc. (to be published)
42. R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev et al., Comput. Math. Appl. **81**, 42 (2021)
43. V. Dobrev, T. Kolev, R. Rieben, SIAM J. Sci. Comput. **34**, B606 (2012)
44. R. Anderson, A. Black, L. Busby, B. Blakeley, R. Bleile, J.S. Camier, J. Ciurej, A. Cook, V. Dobrev, N. Elliott et al., *The Multiphysics on Advanced Platforms Project* (2020)
45. N. Gentile, J. Comput. Phys. **172**, 543 (2001)
46. J.D. Densmore, T.J. Urbatsch, T.M. Evans, M.W. Buksas, J. Comput. Phys. **222**, 485 (2007)

47. R. Bleile, P. Brantley, M. O'Brien, H. Childs, Trans. American Nucl. Soc. **115**, 535 (2016)
48. D.F. Richards, R.C. Bleile, P.S. Brantley, S.A. Dawson, M.S. McKinley, M.J. O'Brien, Quicksilver: A Proxy app for the Monte Carlo transport code mercury, in *2017 IEEE International Conference on Cluster Computing (CLUSTER)* (2017), pp. 866–873
49. M. McKinley, R. Bleile, P. Brantley, S. Dawson, M. O'Brien, M. Pozulp, D. Richards, Status of LLNL Monte Carlo transport codes on sierra GPUs, in *Proceedings of M&C 2019* (Portland, OR, USA, 2019), pp. 2160–2165
50. M. O'Brien, R. Bleile, P. Brantley, S. Dawson, S. McKinley, Hybrid CPU/GPU load balancing for Monte Carlo particle transport, in *Proceedings of 26th International Conference on Transport Theory (ICTT-26)* (Paris, France, 2019)
51. R.C. Bleile, Ph.D. Thesis, University of Oregon, 2021
52. R. Bleile, P. Brantley, M. O'Brien, H. Childs, A dynamic replication approach for Monte Carlo photon transport on heterogeneous architectures, in *Proceedings of the International Conference on Computational Science* (Krakow, Poland, 2021)
53. T. Aila, S. Laine, Understanding the efficiency of ray traversal on GPUs, in *Proceedings of the Conference on High Performance Graphics 2009* (Association for Computing Machinery, New York, NY, USA, 2009), HPG '09, p. 145–149

Cite this article as: Michael Pozulp, Bret Beck, Ryan Bleile, Patrick Brantley, Shawn Dawson, Nick Gentile, Evan Gonzalez, John Grondalski, Michael Lambert, Michael McKinley, Matthew O'Brien, Richard Procassini, David Richards, Alex Robinson, Scott Sepke, David Stevens, Richard Vega, Max Yang. Status of mercury and Imp: Two Monte Carlo transport codes developed using shared infrastructure at Lawrence Livermore National Laboratory, EPJ Nuclear Sci. Technol. **10**, 19 (2024)