EPJ Nuclear Sci. Technol. **11**, 49 (2025)

https://doi.org/10.1051/epjn/2025035

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

2024 MCATK Status

Jeremy Sweezy[1,*], Tim Burke[1,**], Terry Adams[1], Simon Bolding[1], Jesse Giron[1], Alex Long[1], Steven Nolen[1], Corey Skinner[1], Aaron Tumulak[1], Tony Zukaitis[1], Austin McCartney[1,2] and Travis Trahan[1,3]

[1] Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM, USA
[2] Currently Nvidia, Santa Clara, CA, USA
[3] Currently IAEA, Vienna, Austria

Received: 22 November 2024 / Received in final form: 15 April 2025 / Accepted: 17 April 2025

Abstract. The Monte Carlo Application Toolkit (MCATK) is a C++ component-based Monte Carlo particle transport capability that has been in development by Los Alamos National Laboratory (LANL) since 2008. This paper presents an update on MCATK's current capabilities, focusing on notable advancements made since the previous status reports (T. Adams, S. Nolen, J. Sweezy, A. Zukaitis, J. Campbell, T. Goorley, S. Greene, R. Aulwes, Ann. Nucl. Energy 82, 41 (2015), T.J. Trahan, T.R. Adams, R.T. Aulwes, S.D. Nolen, J.E. Sweezy, C.J. Werner, Monte Carlo Application ToolKit (MCATK): Advances for 2017, in International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering (Jeju, Korea, 2017). Key enhancements include a Python interface, photon physics, expanded geometry modeling options, improved tallies, and a broader selection of source definitions. Additionally, MCATK now offers stochastic system solvers, shared memory parallelism, and GPU acceleration of ray-tracing tallies. These enhancements have significantly expanded MCATK's functionality.

1 Introduction

The Monte Carlo Application Toolkit (MCATK) is a C++ component-based Monte Carlo particle transport capability developed by Los Alamos National Laboratory. The goals of MCATK are multifold, including (but not limited to): (1) provide components for general-purpose Monte Carlo codes, such as the MCNP® [3] software, (2) provide components for developing Monte Carlo tools for specific bespoke applications, (3) provide well-tested and reliable components, and (4) provide a modern code basis for use on advanced computing architectures. This paper provides an update on the recent additions to MCATK's capabilities, especially those that have been added since the prior update papers in 2015 and 2017 [1,2].

The development of MCATK began in 2008 is a part of the response to change the computing landscape. In 2008 the Roadrunner [4] supercomputer was installed at LANL, breaking the petaflop barrier. Roadrunner used AMD Opteron CPUs accelerated with the IBM Cell Broadband Engine (also used in the Sony PlayStation (3) [5]. The architecture and its challenges foretold the current state of high performance scientific computing that is now prevalent in the form of GPU accelerated HPC platforms. The architecture presented multiple programming challenges to software developers, including that no Fortran compiler was initially available. While Fortran continues to be well supported for CPU support, programming GPUs through Fortran remains a challenge.

Also at the same time, there came a need to develop bespoke Monte Carlo applications that provided significant performance, reduced time to delivery, with targeted verification and validation. It became clear that meeting all these demands with the MCNP® software, while also supporting the wide user base, would be difficult.

1.1 Development methodology

The early development of MCATK (2008–2013) [1] was performed using Extreme Programming (XP) [6] and Agile [7] software development methodologies. Hallmarks of this development included, paired programming, co-location of developers in the same room, test-driven development, unit testing, frequent commits to the head of the repository, continuous building and testing, a goal of testing and building in less than 10 minutes, and frequent delivery of products to users.

The introduction of the Git version control system [8], a growing development team, and multiple years with the team working from home due to the coronavirus pandemic – created an environment that changed the software development methods used by the MCATK team. Currently, the team uses a more long-term branching strategy,

* e-mail: jsweezy@lanl.gov
** e-mail: tpburke@lanl.gov

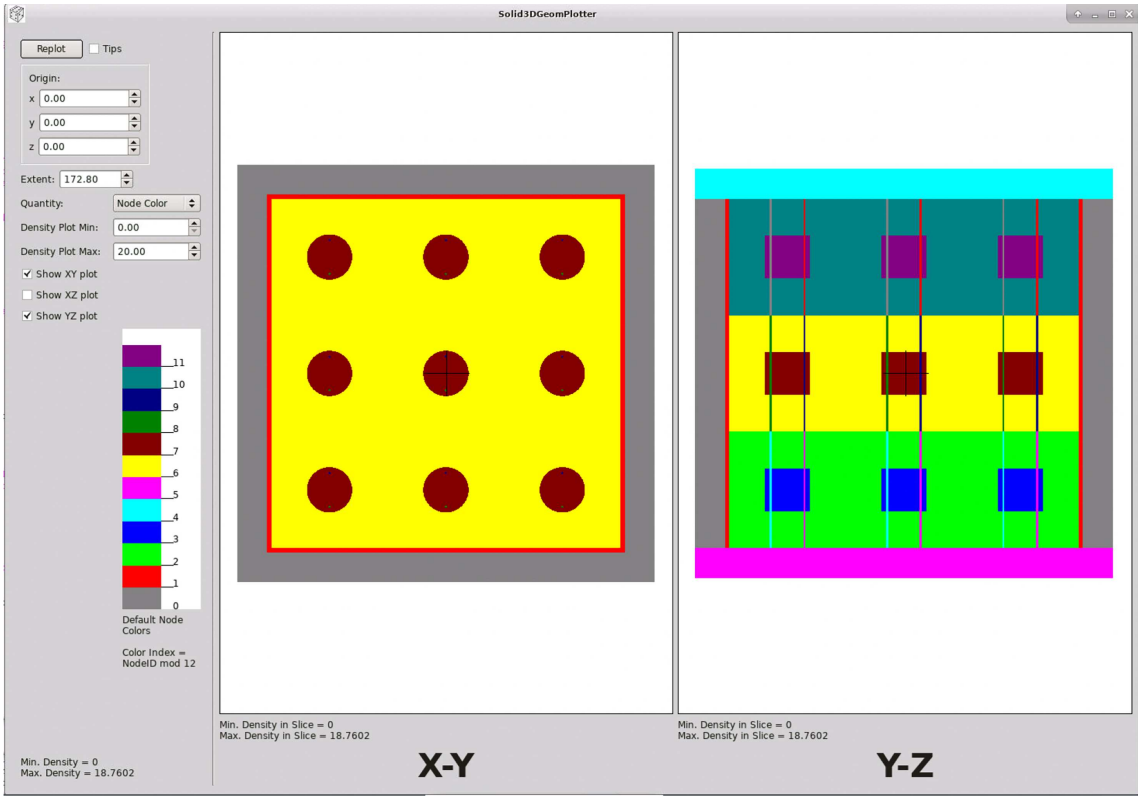

Fig. 1. MCATK geometry plotter, showing two views of the Tinkertoy 2 critical assembly.

thanks to easier branching management with Git. The use of paired programming has decreased with an increase in the reliance on code review that has been made easier with tools like GitLab.

1.2 Capabilities

The capabilities in the MCATK code base have been steadily growing to make MCATK a more capable Monte Carlo software library. Currently MCATK's major capabilities include:

- mesh-based and solid-body geometries
- multiple solution algorithms that include: static k_{eff} and α eigenvalue, time-dependent, fixed-source, radiography, and stochastic systems solution algorithms
- continuous-energy neutron, photon, and photonuclear transport physics
- track length and next-event estimators for tallies
- HDF5 output for tallies
- GPU acceleration support of ray-tracing tallies via the MonteRay library
- MPI parallelism, including: on-node shared memory and domain decomposition
- Python, C++, C and Fortran bindings

2 Features

The current major features of the MCATK library are listed in the following sections.

2.1 Geometry

MCATK supports both mesh geometry and solid-body geometry [9], allowing users flexibility in their geometry modeling. The mesh geometries available in MCATK include 3-D Cartesian, 2-D cylindrical, and 1-D spherical meshes. The solid-body geometry is based on a hierarchical scene graph structure, where each solid in the scene is assigned to a node in a tree structure. Each node is a child of, or contained by, the node above it, similar to the structure used in CERN's (Conseil Européen pour la Recherche Nucléaire) ROOT geometry [10]. Nodes in this geometry hierarchy have the properties:

- each node is assigned a geometric primitive.
- Each node can have other nodes registered as children.
- A node can have an optional transformation that will be inherited by its children.
- Children are allowed to overlap, but MCATK applies an order of precedence.
- Parent containment and child precedence can be used to replicate combinatorial operators.
- Additionally, a node can be assigned a mesh object as the geometric primitive, but these nodes cannot have any children.

MCATK can import its own geometry mesh format and the LNK3DNT mesh geometry format. The LNK3DNT mesh geometry is a mesh format supported by the PARTISN [11] deterministic transport software at Los Alamos, historically from the THREEDANT code [12]. The LNK3DNT format is a common format for transport

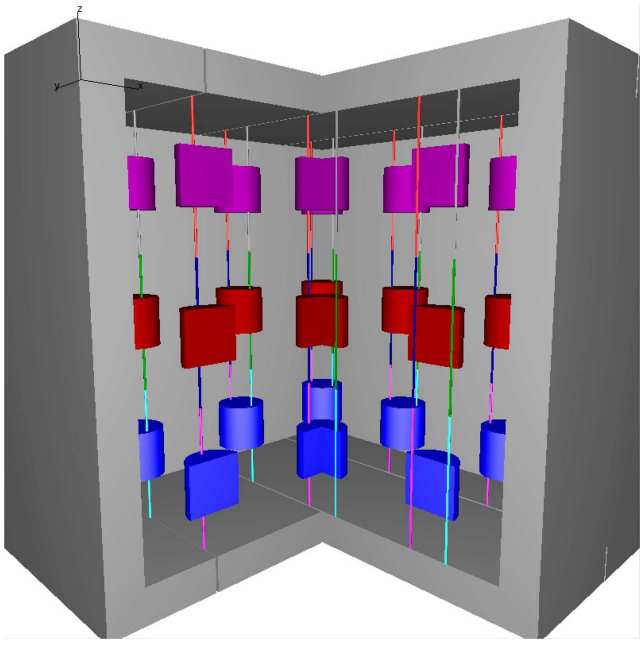

Fig. 2. Rendering of the Tinkertoy 2 critical assembly, created with the MCATK rendering tool.

codes at Los Alamos. It is a structured mesh format that supports both 3-D Cartesian, 2-D cylindrical, and 1-D spherical mesh representations. Both MCATK and MCNP can import LNK3DNT mesh geometry. MCNP can also create LNK3DNT mesh files, providing a mechanism to convert MCNP geometry to a format MCATK can read.

MCATK provides both a 2-D plotting tool and a 3-D rendering tool. The 2-D plotting tool is shown in Figure 1. A rendering from the 3-D rendering tool is shown in Figure 2.

2.2 Solution algorithms

MCATK has three main solution algorithms, or modes: a static k_{eff} eigenvalue mode, a static α eigenvalue mode, and a time-dependent mode [13]. MCATK contains a number of other solution types, which are subsets of these 3 solution types, including: fixed source, radiography, and stochastic system solution algorithms.

The k_{eff} eigenvalue mode is analogous to the `kcode` mode in the MCNP software [14]. k_{eff} is the effective multiplication factor, which is generally considered to be the number of neutrons in a given generation divided by the number of neutrons in the previous generation. Or in terms of the Boltzmann transport equation, k_{eff} can be thought of as a scaling factor on the multiplication term:

$$\mathbf{L}\psi + \Sigma_t\psi = \mathbf{S}\psi + \frac{1}{k_{\text{eff}}}\mathbf{F}\psi \tag{1}$$

where $\mathbf{L}\psi$ is the leakage term, $\Sigma_t\psi$ is the absorption term, $\mathbf{S}\psi$ is the scattering term, and $\mathbf{F}\psi$ is the fission term. MCATK has three estimators for k_{eff}, including a track length estimator, a collision estimator, and an absorption estimator. MCATK uses a radius of gyration estimate as a measure of stationarity [15].

MCATK's k_{eff} eigenvalue mode has been validated using 1-group analytic solutions developed by Sood [16]. And the k_{eff} eigenvalue mode has been compared with MCNP software for criticality benchmarks from the International Criticality Safety Benchmark Evaluation Project (ICSBEP) Handbook [17].

MCATK's k_{eff} algorithm mode has been validated against MCNP software, version 6.3.0, for criticality benchmarks from MCNP's original 31 benchmark model criticality validation suite [18,19], which model experiments from the ICSBEP. These 31 problems have been modeled with solid-body geometry in both MCNP software and MCATK. Some of the problems have also been modeled in both MCNP software and MCATK with the LNK3DNT mesh representation. And some spherical models have also been modeled directly in MCATK's spherical mesh representation. Thus the additional mesh representations increased the original 31 problems from the MCNP benchmark model criticality validation suite to 55 problems.

The k_{eff} and CPU timing results for 37 of the 55 problems are provided in Table 1 for the solid-body and LNK3DNT mesh geometries. The spherical mesh results are not provided as they are nearly identical to the results of the solid-body geometry – due to statistical variation. The timings for single CPU performance are provided in the last column relative to MCNP. The results show that MCATK is generally comparable to the performance of MCNP software for most problem types. MCATK is approximately 2–4 times faster for very simple geometries like Godiva. This indicates that MCATK's cross-section lookup may be faster than the MCNP code, however, the MCNP code is also tallying more information than MCATK which may explain the difference. For future work the performance difference will be investigated. But, MCATK is 7 times slower than MCNP software for the test problem SB5RN3, which is a closely packed hexagonal lattice. As MCATK does not have a lattice geometry capability it cannot reach the performance of MCNP software for closely packed lattices. But MCATK can approach the performance of MCNP software for lattices that can be arranged in non-overlapping rows, such as BAWXI2, ICT2C3, and PNL33, by nesting the cells in a lattice row into row nodes. This approach is not possible for closely packed lattices. MCATK is approximately 2–4 times faster than MCNP software for LNK3DNT mesh geometries. It is unclear why MCATK is faster than MCNP in some cases but is likely due to other simulation physics rather than geometry tracking algorithm differences.

The results of all 55 problems were analyzed statistically and found to be in agreement with MCNP software within expected normal statistics. To perform this analysis, we compared the number of tests agreeing within 1, 2, and 3 standard deviations to the expected values based on a normal distribution. The metric is the k_{eff} difference between the simulations divided by the combined uncertainty:

$$\frac{\Delta k_{\text{eff}}}{\sigma_\Delta} = \frac{k_{\text{eff,MCATK}} - k_{\text{eff,MCNP}}}{\sqrt{\sigma^2_{\text{MCATK}} + \sigma^2_{\text{MCNP}}}} \tag{2}$$

The results from this statistical analysis are presented in Table 2.

Table 1. MCATK and MCNP software results for 37 of the 55 problems in the MCATK criticality V&V suite for solid-body and LNK3DNT mesh geometries. Not shown are the 17 spherical mesh geometry benchmarks which closely match the results of solid-body geometry simulations.

	MCNP		MCATK			MCATK/MCNP
Run name	k_{eff}	Std. Dev.	k_{eff}	Std. Dev.	$\Delta k_{\text{eff}}/\sigma_{\Delta}$	CPU time
BAWXI2_Solid3D	1.00097	0.00033	1.00180	0.00052	1.34	1.59
BIGTEN_Solid3D	0.98419	0.00026	0.98267	0.00049	–2.77	0.54
BIGTEN_lnk3dnt	0.98382	0.00035	0.98450	0.00061	0.99	0.33
FLAT23_Solid3D	0.99529	0.00016	0.99531	0.00022	0.05	0.58
FLAT25_Solid3D	0.99632	0.00014	0.99600	0.00021	–1.28	0.57
FLATPU_Solid3D	0.99777	0.00037	0.99755	0.00049	–0.36	0.58
FLSTF1_Solid3D	0.98427	0.00023	0.98465	0.00040	0.83	1.01
GodivaR_Solid3D	1.01380	0.00057	1.01208	0.00087	–1.63	0.69
Godiva_Solid3D	0.99330	0.00013	0.99351	0.00021	0.87	0.38
HISHPG_Solid3D	1.00818	0.00040	1.00904	0.00064	1.12	0.71
ICT2C3_Solid3D	1.02088	0.00052	1.02203	0.00098	1.01	1.43
IMF03_Solid3D	0.99577	0.00013	0.99566	0.00022	–0.44	0.85
IMF04_Solid3D	1.00033	0.00014	1.00036	0.00023	0.08	0.73
JEZ233_Solid3D	0.99712	0.00012	0.99705	0.00018	–0.33	0.28
JEZ240_Solid3D	0.99833	0.00010	0.99834	0.00013	0.08	0.38
JEZPU_Solid3D	0.99807	0.00013	0.99811	0.00017	0.18	0.37
LST2C2_Solid3D	0.97176	0.00032	0.97238	0.00064	0.89	1.02
ORNL10_Solid3D	0.98569	0.00028	0.98565	0.00075	–0.06	0.86
ORNL11_Solid3D	0.99190	0.00026	0.98992	0.00081	–2.36	0.82
PNL2_Solid3D	1.00932	0.00021	1.00970	0.00035	0.90	1.06
PNL33_Solid3D	1.00873	0.00050	1.00926	0.00087	0.52	0.73
PUBTNS_Solid3D	0.99702	0.00045	0.99857	0.00066	1.95	0.95
PUSH2O_Solid3D	1.01190	0.00053	1.01243	0.00077	0.56	0.63
SB25_Solid3D	1.03477	0.00099	1.03776	0.00173	1.45	0.93
SB5RN3_Solid3D	0.97684	0.00090	0.97574	0.00165	–0.60	7.15
STACY36_Solid3D	0.97791	0.00029	0.97627	0.00067	–2.30	0.94
STACY36_lnk3dnt	0.97941	0.00042	0.97802	0.00095	–1.36	0.59
THOR_Solid3D	0.99571	0.00015	0.99577	0.00021	0.23	0.61
THOR_lnk3dnt	0.99560	0.00042	0.99476	0.00064	–1.10	0.25
TT2C11_Solid3D	1.00130	0.00051	1.00330	0.00092	1.89	0.66
UH3C6_Solid3D	0.98829	0.00054	0.98677	0.00091	–1.45	1.24
UH3C6_lnk3dnt	0.98869	0.00017	0.98887	0.00029	0.54	0.57
UMF5C2_Solid3D	0.99247	0.00014	0.99278	0.00023	1.17	0.47
ZEBR8H_Solid3D	1.00138	0.00027	1.00170	0.00057	0.50	1.08
ZEBR8H_lnk3dnt	1.00210	0.00030	1.00167	0.00062	–0.63	0.55
ZEUS2_Solid3D	0.98943	0.00039	0.98950	0.00064	0.09	1.18
ZEUS2_lnk3dnt	0.99271	0.00054	0.99339	0.00089	0.66	0.37

While the use of k_{eff} calculations is common for criticality safety applications, in the experimental world k_{eff} is inferred. What is actually measured is reactor period (λ) or subcritical multiplication. The system time constant, usually denoted as α where $\alpha = 1/\lambda$, describes the rate of change of neutrons in a static system. Using the expression for α in the Boltzmann transport equation from [14] results in the transport equation with α providing a loss term:

$$\mathbf{L}\psi + \left(\Sigma_t + \frac{\alpha}{\nu}\right)\psi = \mathbf{S}\psi + \frac{1}{k'}\mathbf{F}\psi \ , \tag{3}$$

Table 2. Comparison of k_{eff} differences ($\Delta k_{\text{eff}}/\sigma_{\Delta}$) between MCATK and MCNP software for 55 problems from the MCATK criticality V&V suite for solid-body, spherical mesh, and LNK3DNT mesh geometries.

$\Delta k_{\text{eff}}/\sigma_{\Delta}$ Interval	Actual Count	Expected Count	Expected Count Variance	Actual Percentage of Cases	Expected Percentage of Cases
$\lvert\Delta k_{\text{eff}}/\sigma_{\Delta}\rvert < 1$	34	36.72	11.75	62.96%	~68.0%
$\lvert\Delta k_{\text{eff}}/\sigma_{\Delta}\rvert < 2$	51	51.30	2.57	94.44%	~95.0%
$\lvert\Delta k_{\text{eff}}/\sigma_{\Delta}\rvert < 3$	54	53.84	0.16	100.00%	~99.7%
$\lvert\Delta k_{\text{eff}}/\sigma_{\Delta}\rvert > 3$	0	0.16	0.16	0.00%	<0.3%

where $\mathbf{L}\psi$ is the leakage term, $\mathbf{S}\psi$ is the scattering term, $(1/k')\mathbf{F}\psi$ is the fission term, $\Sigma_t\psi$ is the absorption term. $(\alpha/\nu)\,\psi$ is often referred to as the "time loss" term. MCATK calculates α through an iterative procedure until $k' = 1$.

2.3 Stochastic systems solution algorithms

MCATK contains several solution modes that are useful for describing stochastic systems, including multiplication, probability of initiation (POI), probability of extinction (POE) [20,21], probability of survival (POS) [22], and quasi-static [23] modes. These solution modes provide dynamic analysis for quantities of interest that cannot be obtained with static analysis. Such analyses include time-integrated or time-dependent quantities on a chain-by-chain basis such as total dose or instantaneous dose rate, burst wait time of pulsed reactor experiments, and the probability of survival (probability that a fission chain persists until some designated time). The POE and POS as a function of time for a simulated criticality excursion are shown in Figure 3. These solution modes are useful for modeling criticality accident scenarios, criticality experiments, transient reactor analysis, pulsed reactors, and subcritical systems.

The C5G7-TD benchmark has been used to verify MCATK's quasi-static solution modes. C5G7-TD is a light water reactor with 16 UO2 and MOX fuel assemblies [24]. The details of MCATK's modeling of this benchmark are provided in reference [23].

2.4 Sources

MCATK now supports a wide array of source options. MCATK allows sources to be specified in a variety of formats. The source can be provided on the geometry mesh or defined explicitly by the user. The available source definitions are listed in Table 3. MCATK also supports a surface source read/write capability in order to import or export sources to MCNP software.

2.5 Nuclear data and the Avalanche library

MCATK uses continuous-energy neutron and photon data from data processed through the NJOY nuclear data processing code. Traditionally MCATK directly read ACE (A Compact ENDF) [25] formatted Evaluated Nuclear Data Files (ENDF) [26]. Recently, the MCATK toolkit has been updated to utilize a new nuclear data interface library called *Avalanche*. *Avalanche* accesses nuclear data through the use of the ACEtk [27] library which reads ACE-formatted nuclear data files. The *Avalanche* C++ library provides a standardized interface for accessing continuous-energy photoatomic and nuclear data across various Monte Carlo particle transport codes. *Avalanche* offers several key benefits, including:

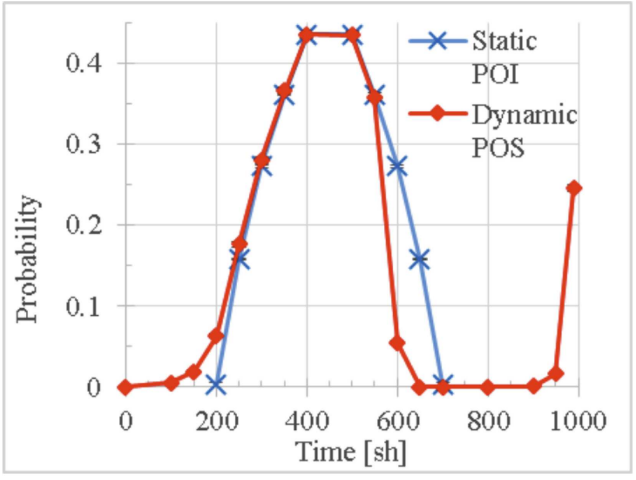

Fig. 3. Probability of initiation (POI) and probability of survival (POS) for a criticality excursion, from Trahan et al [22].

- a uniform interface for accessing both photoatomic and nuclear data
- support for both C++ and Python interfaces
- data format independence, ensuring that the API remains the same regardless of the underlying data storage format.

While *Avalanche* is agnostic to the data storage format, it still requires populating its data structures from a specific format. Currently, only ACE files are supported. Notably, *Avalanche* enables access to its data on Graphics Processing Units (GPUs). To optimize performance, *Avalanche* stores its data in GPU-friendly containers with minimal

Table 3. Available Source Definitions in MCATK.

Property	Options
Angle	Monodirectional
	Isotropic
	Histogram
	Linear
Energy	Monoenergetic
	Histogram
	Linear
	Discrete Lines
	Gaussian
	Maxwellian
	Watt
Space	Point
	Histogram
	Solid-body Node
	Line
	Parallelogram
	Right Parallelepiped
	Disc (Hollow/Solid)
	Annulus/Cylinder
	Spherical Shell
Time	Burst
	Histogram
	Uniform
Misc.	Spontaneous Fission
	Custom

pointer indirections, leveraging modern C++17 features like variants to achieve polymorphism without relying on virtual base classes and inheritance patterns. Additionally, users can opt to store data in single precision, reducing memory requirements and potentially improving performance on GPUs. Additionally, *Avalanche* allows for linearizing log-log interpolated photoatomic data that is provided by ACE files. Finally, *Avalanche* provides all necessary data structures and accessors used during transport in header files, allowing for maximum inlining and optimized performance on GPUs.

2.6 Hash-based cross-section lookup

MCATK uses a hash-based cross-section lookup to reduce the computational cost of cross-section access. The cross-section hash data table is based on the concept described by Brown [28]. By exploiting the binary representation of floating-point numbers according to the IEEE 754 standard, MCATK replaces the computationally expensive mathematical logarithm function with a bit manipulation operation. This approach significantly reduces the computational cost of cross-section access. The bit shifting hash function used in MCATK is listed in pseudocode in Figure 4. This capability has recently been moved into to the *Avalanche* nuclear reaction library.

2.7 Tallies

MCATK supports various types of tallies, including collision event tallies, surface tallies, track length tallies, next-event estimators (also known as point detectors) [29], and volumetric ray casting estimators [30].

Tally binning is facilitated through the use of tally filters. A filter is a function that maps a particle from anywhere in phase space to a specific tally bin, if applicable. Users can employ filters to control binning based on properties such as energy, time, particle type, direction, material, and cell location.

MCATK offers several score modifiers to customize tallies. Available options include:

- flux
- heating
- neutron production (fission-induced and net)
- photon production (neutron-induced and total)
- absorption
- cross-section multiplier
- energy multiplier
- inverse speed multiplier
- histogram and linear response functions.

MCATK's tallies have been verified against MCNP software for most tally types. The photon next-event estimator has been compared to MCNP software using a series of cylinder tests [31]. For these cylinder tests, shown in Figure 5, photons are incident on a cylinder, 400 cm long and 40 cm in diameter. A point detector is located 400 cm off axis and 175 cm from the face of the cylinder. Cylinders of differing material and density comprise multiple tests. The MCATK results compared with MCNP software are shown in Table 4. The MCATK results are within two standard deviations of the MCNP code results, except the 7.5×10^{-5} g/cc Tungsten case which has a large difference in the number of standard deviations but the result is calculated to the limits of numeric precision and the difference is on the order of 10^{-14}.

2.8 GPU acceleration with MonteRay

MCATK can use the MonteRay ray tracing engine to accelerate the calculation of ray tracing tallies, such as next-event estimators [30]. This is done by using a GPU to perform the ray tracing calculations while the random walk continues to be performed on the CPU. MonteRay can provide significant performance gains for ray tracing tallies on GPU hardware. A simulated radiograph of the Zubal head phantom created with MCATK utilizing MonteRay on GPU hardware is shown in Figure 6 [32,33]. For this simulation the collided component of the radiograph converges much slower than the source component, thus the performance of the calculation of the collided component dominates. The performance improvement of the collided component with MonteRay is 15×.

```
FUNCTION fastHashFunction(energy: 64-BIT REAL) -> INTEGER
    // 8 significand bits provides 5964 bins over the
    // energy range from 1e-11 to 1e3 for 64-bit float

    // Constants
    significandBits = 8
    exponentBits = 11
    shiftAmount = 64 - (significandBits + exponentBits)

    // Convert energy to a 64-bit integer
    i = bitwiseCopy(energy)

    // Shift the bits and return the result
    RETURN i >> shiftAmount
END FUNCTION
```

Fig. 4. Pseudocode for the bit-shift-based hash function used to convert energy to a logarithmic integer index used in place of the mathematical log function.

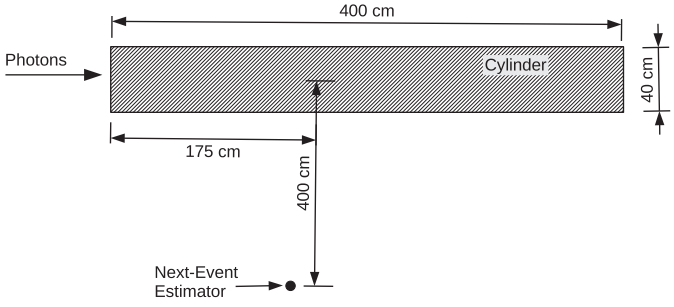

Fig. 5. Geometry of photon next-event estimator cylinder tests [31].

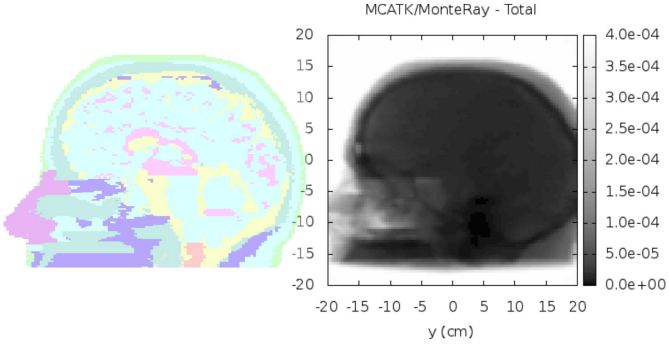

Fig. 6. Radiograph of a head phantom generated with MonteRay on GPU hardware providing a 15× performance increase for the collided component. The source component is more than 100× more intense than the collided component, but the calculation time of the collided component is 10.3× more than the source component.

Table 4. MCATK to MCNP software comparison of photon next-event estimator results for 8 tests using various cylinder materials and densities.

Material	Density (g/cc)	Photon Energy (MeV)	MCATK-MCNP Difference (# Std Devs)
W	7.5×10^{-5}	0.003	–6.9*
W	1.0×10^{-3}	0.015	0.4
W	5.6×10^{-2}	0.13	1.2
W	2.6	2.0	1.3
W	2.0	60.0	–1.8
Concrete	1.9	2.0	–0.1
Water	1.5	2.0	–0.2
Fe	2.3	2.0	–1.2

* Difference (6×10^{-14}), probably due to round-off

2.9 Parallel computing capabilities

MCATK has historically used MPI for parallel computing capabilities for message passing between distributed memory multiprocessors using the Boost C++ MPI interface [34,35]. MCATK uses MPI for distributing particle histories to all MPI processes. MCATK also has multiple options for distributing the geometry information to the processes. This includes replicating the entire geometry among the computing nodes, known as domain replication (see Fig. 7b). For mesh based geometries only, MCATK can also distribute the mesh among computing nodes in order to support modeling of very large mesh geometries, known as domain decomposition (see Fig. 7c). For domain decomposition, MCATK passes the particles among the nodes owning the local mesh geometry data. MCATK can also support mixing both domain decomposition and domain replication, referred to as domain hybrid, allowing some domains to be replicated as specified by the user (see Fig. 7d).

Since 2017, MCATK began to address the issue of threading on multi-processor nodes with shared memory. In contrast to the MCNP software, which uses a combined MPI+OpenMP approach for its hybrid programming model that requires developers to identify private data for each thread, MCATK has taken a different approach by

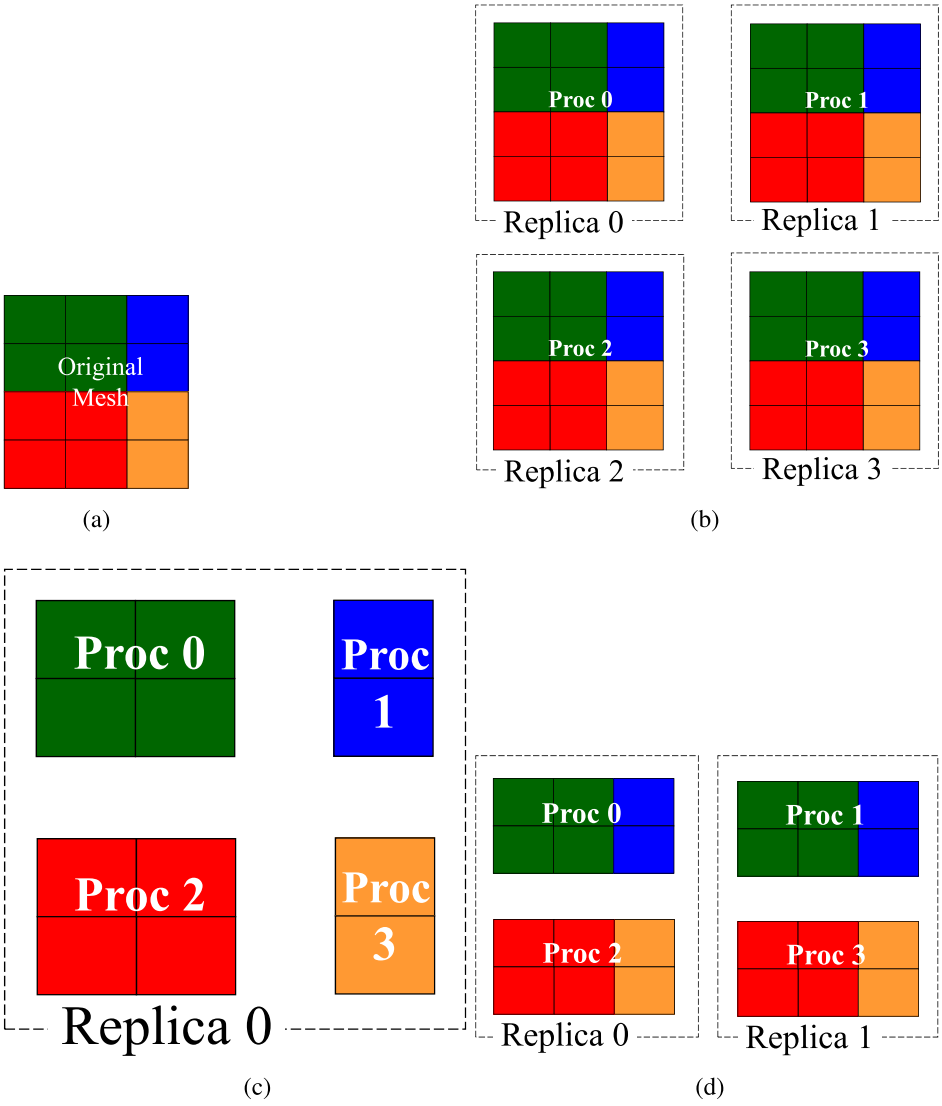

Fig. 7. A notional example of geometry mesh parallel decomposition options available in MCATK. (a) Original 3×4 rectangular mesh geometry. The original mesh is shown with no decomposition applied. (b) Domain Replication: The mesh is replicated on four processes, resulting in each process holding an identical copy of the entire geometry. (c) Domain Decomposition: The mesh is decomposed into four segments, each containing a unique portion of the geometry. Each process holds one segment. (d) Hybrid Decomposition: A combination of domain decomposition and replication, where the mesh is divided into two segments, with each process holding two replicas of each segment.

utilizing shared memory to designate which data is shared among all processors on a shared memory compute node. This method allows developers to concentrate on sharing large amounts of static data, like geometry and material information, that do not change during particle transport simulations. This approach to the hybrid "MPI+X" parallel programming model [36], which combines "distributed memory" (MPI) and "shared memory" (X) parallelism, is termed "MPI+MPI" [37]. By focusing on static data, this approach also reduces the risk of race conditions, making it easier to debug. Initially, MCATK employed the Boost interprocess library [35] to share mesh geometry data using shared memory; however, it is now transitioning to use MPI-3 shared memory for sharing both nuclear data and geometry data. In the future, this shared memory method will also be applied to optionally share read/write tally data, offering additional memory savings for large tallies. A detailed study of the performance impacts of storing nuclear data and geometry information in shared memory has not yet been conducted – the performance impacts will be investigated in the future.

3 Future work

MCATK continues to evolve. The near term features to be added to MCATK include:

- measurement of performance efficiency of the shared memory threading implementation.
- Supporting full particle transport on GPU hardware.

- Thick target bremsstrahlung photon creation.
- Electron transport.
- Tracking in boundary representation meshes, such as STL and STEP geometry files.

MCATK and Avalanche are not yet available for users outside of Los Alamos National Laboratory. It is expected that MCATK will be distributed through the Radiation Shielding Information Computational Center (RSICC) in the future, when MCATK's feature set and user documentation can support widespread use.

Acknowledgments

The authors would like to acknowledge the following people for their contribution to the development of MCATK: Rob Aulwes, George P. Burdell, Simon Greene, H. Grady Hughes, JP Morgan, Lori Pritchett-Sheats, and Chris Werner. AI acknowledgements: OpenAI's ChatGPT Enterprise (a large language model trained by OpenAI, based on the GPT-4 architecture) [38] and Meta's open source Llama 3.1 [39] were used to create BibTeX entries, format tables for LaTeX, and provide grammar corrections throughout the document.

Funding

This work is supported by the U.S. Department of Energy Advanced Scientific Computing Program. This work is supported by the U.S. Department of Energy through Los Alamos National Laboratory (LANL) operated by Triad National Security, LLC, for the National Nuclear Security Administration (NNSA) under Contract No. 89233218CNA000001.

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

Writing – original draft: J.S.; Writing – review and editing: T.B., T.A., J.G., S.B., A.T.; Visualization: T.A., J.G., S.N., J.S., T.T.; Formal analysis: T.B., T.A., A.T., T.T., J.S.; Validation: T.A., S.N., T.T., A.T., J.S.; Software: T.B., T.A., S.B., J.G., A.L., S.N., C.S., A.T., T.Z., A.M., T.T., J.S.; Project administration: T.T., A.L., T.A., S.N.

References

1. T. Adams, S. Nolen, J. Sweezy, A. Zukaitis, J. Campbell, T. Goorley, S. Greene, R. Aulwes, Ann. Nucl. Energy **82**, 41 (2015)
2. T.J. Trahan, T.R. Adams, R.T. Aulwes, S.D. Nolen, J.E. Sweezy, C.J. Werner, Monte Carlo Application ToolKit (MCATK): Advances for 2017, in *International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering* (Jeju, Korea, 2017)
3. T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. Cox, J. Durkee, J. Elson, M. Fensin, R. Forster et al., Nucl. Technol. **180**, 298 (2012)
4. K.J. Barker, K. Davis, A. Hoisie, D.J. Kerbyson, M. Lang, S. Pakin, J.C. Sancho, Entering the petaflop era: The architecture and performance of Roadrunner, in *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing* (2008), pp. 1–11
5. M.W. Riley, J.D. Warnock, D.F. Wendel, IBM J. Res. Develop. **51**, 545 (2007)
6. K. Beck, C. Andres, *Extreme Programming Explained: Embrace Change, 2nd Edition* (Addison-Wesley, 2004), ISBN 978-0321278654
7. J. Shores, S. Warden, *The Art of Agile Development, 2nd Edition* (O'Reilly Media, 2021), ISBN 978-1492080695
8. *Git* (2024), [Software], https://git-scm.com/
9. J. Sweezy, Tech. Rep. LA-UR-15-28544, Revision 3, Los Alamos National Laboratory, Los Alamos, New Mexico, US (2015)
10. R. Brun, F. Rademakers, Nucl. Inst. & Meth. in Phys. Res. A **389**, 81 (1997)
11. R.E. Alcouffe, R.S. Baker, J.A. Dahl, S. Turner, R. Ward, Tech. Rep. LA-UR-05-3925, Los Alamos National Laboratory (2005)
12. R.E. Alcouffe, R.S. Baker, F.W. Brinkley, D.R. Marr, R.D. O'Dell, W.F. Walters, Tech. rep., Los Alamos National Lab. (LANL), Los Alamos, NM (United States) (1995), https://www.osti.gov/biblio/212580
13. T. Trahan et al., *MCATK Code Version 23.0.0 User Manual* (Los Alamos National Laboratory, Los Alamos, NM, 2024)
14. S. Nolen, T. Adams, J. Sweezy, Tech. Rep. LA-UR-16-24139, Los Alamos National Laboratory, Los Alamos, New Mexico, US (2016), http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-16-24139
15. R. Gentle, P. Edwards, W. Bolton, in *Mechanical Engineering Systems* (Newnes, 2001), p. 264, ISBN 9780750652131, https://www.sciencedirect.com/book/9780750652131/mechanical-engineering-systems
16. A. Sood, R. Forster, D. Kent Parsons, Progress in Nuclear Energy **42**, 55 (2003)
17. Tech. Rep. ICSBEP-2015, OECD Nuclear Energy Agency, Paris, France (2015)
18. C.J. Josey, A.R. Clark, J.A. Kulesza, E.J. Pearson, M.E. Rising, Tech. Rep. LA-UR-22-32951, Rev. 1, Los Alamos National Laboratory, Los Alamos, NM, USA (2022), https://www.osti.gov/biblio/1907750
19. R.D. Mosteller, Tech. Rep. LA-UR-04-6489, Los Alamos National Laboratory, Los Alamos, NM, USA (2004), http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-04-6489
20. S. Nolen, Using fission chain analysis to inform probability of extinction/initiation calculations with MCATK, in *Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method* (American Nuclear Society, Nashville, TN, USA, 2015)
21. T.E. Booth, Nucl. Sci. Eng. **166**, 175 (2010)
22. T.J. Trahan, S. Nolen, T. Adams, J. Sweezy, A Monte Carlo algorithm for fission chain analysis of dynamic stochastic systems, in *Transactions of the American Nuclear Society* (Washington, D.C., 2015), Vol. 113
23. T.J. Trahan, Ann. Nucl. Energy **127**, 257 (2019)
24. V. Boyarinov, P. Fomichenko, J. Hou, K. Ivanov, A. Aures, W. Zwermann, K. Velkov, Nuclear Energy Agency Organisation for Economic Co-operation and Development (NEA-OECD), Paris, France (2016)
25. J. Conlin, P. Romano, Report LA-UR-19-29016, Los Alamos National Laboratory (2019)

26. A. Trkov, M. Herman, D.A. Brown, Tech. rep. BNL-203218-2018-INRE, National Nuclear Data Center, Brookhaven National Laboratory (2018)
27. *ACEtk: A toolkit for reading and interacting with ACE nuclear data files* (2024), [Software], https://github.com/njoy/ACEtk
28. F.B. Brown, Tech. Rep. LA-UR-14-27037, Los Alamos National Laboratory (2014), presented at the OECD-NEA-WPNCS Expert Group Meeting - Advanced Monte Carlo Techniques, September 8, 2014, https://www.osti.gov/biblio/1154979
29. M. Kalos, Nucl. Sci. Eng. **16**, 111 (1963)
30. J.E. Sweezy, J. Comput. Phys. **372**, 426 (2018)
31. J.E. Sweezy, Tech. Rep. LA-UR-16-28780, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (2016), https://www.osti.gov/biblio/1332213
32. I. Zubal, C. Harrell, E. Smith, A. Smith, Two dedicated software, voxel-based, anthropomorphic (torso and head) phantoms, in *Proceedings of the International Workshop, National Radiological Protection Board* (National Radiological Protection Board, Chilton, UK, 1996)
33. J. Evans, T. Blue, N. Gupta, Medical Phys. **28**, 780 (2001)
34. *Message passing interface (mpi) standard* (2015), https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf
35. *BOOST C++Libraries*, [Software], http://www.boost.org
36. R. Thakur, P. Balaji, D. Buntinas, D. Goodell, W. Gropp, T. Hoefler, S. Kumar, E. Lusk, J. Träff, Proc. SciDAC 2010 **2** (2010)
37. T. Hoefler, J. Dinan, D. Buntinas, P. Balaji, B. Barrett, R. Brightwell, W. Gropp, V. Kale, R. Thakur, Computing **95**, 1121 (2013)
38. OpenAI, *Chatgpt enterprise* (2024), Large language model trained by OpenAI, based on the GPT-4 architecture, https://www.openai.com
39. H. Touvron, R. Thoppilan, E. Cordonnier, A. Goyal, Y. Hua, A. Lachaux, S. Razavi, J. Hoffmann, J.B. Alayrac et al., *Llama: Open and efficient foundation model*, https://arxiv.org/abs/2302.13971 (2023)

Cite this article as: Jeremy Sweezy, Tim Burke, Terry Adams, Simon Bolding, Jesse Giron, Alex Long, Steven Nolen, Corey Skinner, Aaron Tumulak, Tony Zukaitis, Austin McCartney, Travis Trahan, 2024 MCATK Status, EPJ Nuclear Sci. Technol. **11**, 49 (2025). https://doi.org/10.1051/epjn/2025035.