LA-UR-22-33103

Approved for public release; distribution is unlimited.

Title: MCNP® Code Version 6.3.0 Release Notes

Author(s): Rising, Michael Evan; Armstrong, Jerawan Chudoung; Bolding, Simon R.; Brown, Forrest B; Bull, Jeffrey S.; Burke, Timothy Patrick; Clark, Alexander Rich; Dixon, David A (D. Dixon CCS-2); Forster, Robert Arthur Iii; Giron, Jesse Frank; Grieve, Tristan Sumner; Hughes, Henry G. (Grady) III; Josey, Colin James; Kulesza, Joel A.; Martz, Roger L.; McCartney, Austin P.; McKinney, Gregg W.; Mosher, Scott William; Pearson, Eric John; Solomon, Clell Jeffrey Jr.; Swaminarayan, Sriram; et al.

Intended for: Report

Issued: 2023-01-17 (rev.1)

Los Alamos NATIONAL LABORATORY

LA-UR-22-33103 Rev. 1

MCNP®

Code Version 6.3.0

Release Notes

Document compiled from `git` hash `f7c7fe3f9a` on January 10, 2023.

Los Alamos
NATIONAL LABORATORY

MCNP® Code Version 6.3.0 Release Notes

LA-UR-22-33103 Rev. 1

January 10, 2023

Los Alamos National Laboratory

Michael E. Rising	Robert A. (Art) Forster III	Gregg W. McKinney
Jerawan C. Armstrong	Jesse F. Giron	Scott W. Mosher
Simon R. Bolding	Avery S. Grieve	Eric J. Pearson
Forrest B. Brown	Henry G. (Grady) Hughes III	Clell J. (CJ) Solomon Jr.
Jeffrey S. Bull	Colin J. Josey	Sriram Swaminarayan
Timothy P. Burke	Joel A. Kulesza	Jeremy E. Sweezy
Alexander R. Clark	Roger L. Martz	Stephen C. Wilson
David A. Dixon	Austin P. McCartney	Anthony J. Zukaitis

Contents

The Monte Carlo N-Particle® (MCNP®) code is a general-purpose, continuous-energy, generalized-geometry, time-dependent, radiation transport code developed by the MCNP development team. The MCNP calculations provide predictive capabilities that can replace expensive or impossible-to-perform experiments. Specific application problems include simulations of experimental diagnostics, intrinsic radiation, radiation detection and measurement, criticality safety, nuclear threat reduction and response, radiation health protection, nuclear weapons effects, and nuclear forensics. This MCNP code, version 6.3.0, follows the MCNP6.2.0 version [1]. Since the release of MCNP6.2.0, many changes have been made to the MCNP code. These changes include new or improved features, a new build system, code enhancement and modernization, and bug fixes. The MCNP code, version 6.3.0, theory and user input information is documented in MCNP® Code Version 6.3.0 Theory & User Manual [2], the build guidance for various platforms is documented in MCNP® Code Version 6.3.0 Build Guide [3], and the verification and validation testing for various application benchmark test suites is documented in MCNP® Code Version 6.3.0 Verification & Validation Testing [4].

1 New Features and Notable Improvements

1.1 Deprecation and Removal of Old Features and Functionality

The MCNP code has undergone significant modernization since its last release. New capabilities have been added, the source code maintainability has been improved, and new output file formats are available that provide performance and usability benefits. By virtue of performing this modernization, some previous MCNP components have become obsolete. These obsolete components are kept in this release to permit users to perform calculations with the old and new components such that appropriate quality control and quality assurance practices (V&V testing, upgrading and testing post-processing utilities, etc.) can be followed.

Within these release notes, and in the MCNP Manual, obsolete components are identified as deprecated. As such, these obsolete components should NOT be relied upon because they will likely be removed from the code in the future. Removing these obsolete components from the code allows the MCNP development team to reduce their maintenance burden to then provide new capabilities (and code releases) more quickly. However, the MCNP development team is happy to receive feedback via the MCNP Forum (https://mcnp.lanl.gov) on the effect of deprecating features and will proactively work to identify solutions that still permit removing the deprecated components. For convenience, Table 1 contains a listing of deprecated features. More information for each of the deprecated issues can be found in the MCNP user manual by searching for the deprecation number listed.

Table 1: Deprecated Features

Deprecation Number	Category	Description
DEP-53292	File Formats	Mesh tally output formats other than `xdmf` (which is new) and `none` [§1.11, §1.2].
DEP-53294	File Formats	The legacy ASCII and binary **`eeout`** file format.
DEP-53361	UM LNK3DNT	Re-use of the embedded geometry background cell for any `matcell` pseudo-cell entries.

continued on next page...

Table 1 — continued from previous page

Deprecation Number	Category	Description
DEP-53382	File Formats	The legacy `PTRAC` ASCII and binary file formats [§1.6]. This includes the `FILE` keyword options `asc`, `aov`, `bin`, and `bov`. The `MAX`, `BUFFER`, and `WRITE` keywords are also deprecated, as they are unused by the new `hdf5` `FILE` option. The `FILTER` keyword options `icl` and `jsu` are deprecated, already replaced by the new `cel` and `sur` options, respectively.
DEP-53383	PTRAC	Use of the `COINC` keyword and the keyword `EVENT` = `cap` option is not supported in the new `PTRAC` `FILE` = `hdf5` format.
DEP-53421	UM Utilities	The **`um_convert`** utility.
DEP-53422	UM Utilities	The **`um_pre_op`** utility.
DEP-53423	UM Utilities	The **`um_post_op`** utility.
DEP-53424	File Formats	The **`mcnpum`** file format.
DEP-53482	Tallies	The `TIR` and `TIC` cards are deprecated. Use `FIR` and `FIC`, respectively.
DEP-53483	Materials	The `MPN` card is deprecated. Use `MX`.
DEP-53484	Tallies	The `PI` card is deprecated. Use `FIP`.
DEP-53519	File Formats	The **`GMV`** file output capability on the `EMBED` card.

While it is preferable to first mark features and capabilities as deprecated before they are fully removed from a future distribution, it is occasionally necessary to fully remove a feature without an immediate side-by-side modern replacement. Because of the desire to avoid this situation in as many circumstances as possible, only a few minor features fall into this category of being removed in this version of the MCNP code. For convenience, Table 2 contains a listing of removed features. If there are questions or comments regarding removed features, please contact the MCNP team at mcnp_help@lanl.gov.

Table 2: Removed Features

Tracking Number	Category	Description
MCNP-53210	Random Number Generator	For more than 15 years, the `RAND` card has been available for setting any options for the Random Number Generator (RNG), and warnings were issued if the obsolescent `DBCN` card entries were used to modify the default RNG settings. Use of the `DBCN` card to modify the RNG settings is now a fatal error. The `RAND` card must be used to modify any RNG settings. `DBCN` entries 1, 8, 13, and 14 should not be used.

continued on next page...

Table 2 — continued from previous page		
Tracking Number	**Category**	**Description**
MCNP-53532	Utilities	Removed the **HTAPE** utility from the distribution. This utility is unchanged from the previous release of the MCNP6.2.0 code and has not been maintained in recent years to remain consistent with the `HISTP` capability within the source code itself. The `HISTP` capability will be marked for deprecation in a future version of the MCNP6 code when a suitable, modern replacement is made available.
MCNP-53533	Utilities	Removed the **MCNP_RANDOM** utility from the distribution. This utility is unchanged from previous releases of the MCNP6 code and has not been maintained in recent years to remain consistent with the random number generator implemented within the source code itself.
MCNP-53524	Tallies	Due to copyright concerns, the built-in flux-to-dose conversion factors have been removed from the `DF` card and the `DOSE` keyword of the Type 1 `TMESH` tallies. They are available in the MCNP Manual formatted as MCNP input for `DE`/`DF` cards.

1.2 Added Mesh Tally `colsci` and `cfsci` Output Formats [MCNP-5367, MCNP-52997]

The `col` and `cf` output formats previously available for MCNP mesh tally outputs would express spatial mesh edge and center coordinates using fixed-format (**f10.3**) notation. This prevented the user from specifying either spatially small or large mesh because of inadequate precision. The `colsci` and `cfsci` output formats address this limitation by expressing *all* floating-point numbers in the **meshtal** file as **es13.5** with the degree of output consistent with the `col` and `cf` output formats, respectively.

These output formats are added to provide temporary relief from the limitations imposed by `col` and `cf` output and are immediately deprecated.

1.3 `KCODE` Convergence Acceleration and Convergence Detection [MCNP-28788]

Three new `KOPTS` keywords have been added to the code: `FMAT`, `FMATCONVRG`, and `FMATACCEL`. The first activates the computation of the fission matrix, which is a tally of probability of fission in mesh cell i caused by a fission source in mesh cell j. In the MCNP code, the mesh used is the same as the Shannon entropy's mesh. The fundamental eigenvalue and eigenvector of the fission matrix correspond to the k-eigenvalue and the steady state fission source, respectively. With `FMATCONVRG`, 11 statistical tests are used to determine when a k-eigenvalue simulation is ready for active cycles based on trends of estimators and comparisons of the fission matrix eigenvector to the Shannon entropy. Finally, the `FMATACCEL` option uses the current fission matrix eigenvector as an importance function for sampling fission points during `KCODE` convergence, which can often improve the convergence rate. Further reading can be found in [5].

1.4 CMake Build System [MCNP-48179, MCNP-50999, MCNP-52871]

The CMake build system, an industry standard build system, is implemented for the MCNP code, version 6.3.0, since the MCNP code base has been undergoing modernization and it would require non-negligible time and effort to rewrite (or update) the old MCNP make build system. For those that build, test, and install the code from source, a fully functional CMake 3 build system has replaced the legacy GNU **`make`** build system in previous versions of the MCNP code. The details on the current implementation and use of the CMake build system can be found in the MCNP® Code Version 6.3.0 Build Guide [3].

1.5 Implementation of Mixed-Material Treatment [MCNP-50112]

A mixed-material treatment needed for tracking particles on a structured mesh geometry is implemented and tested. The code implementation of the mixed-material treatment for embedded LNK3DNT mesh geometry was in a branch developed for LANL users, but this code implementation could not be integrated to the MCNP development branch because of design flaws in the preliminary implementation. In addition, this mixed-material treatment method could not be extended to other MCNP geometries such as the constructive solid geometry (CSG) or the unstructured mesh (UM) geometry. Thus a new mixed-material treatment design concept is developed for the MCNP code so that this design can be applied to all MCNP geometries. The mixed-material treatment feature is currently only for the use of mixed materials coming from an embedded LNK3DNT mesh geometry while maintaining the CSG, LNK3DNT mesh geometry, and UM geometry single material treatment already established in the code.

1.6 HDF5-based Parallel `ptrac` [MCNP-52880]

The previous `PTRAC` (particle track) output implementation has been replaced with one based on HDF5. The main benefits of the new version are efficiency, standardization and simplification of the output format, and the ability to run particle track problems with MPI and thread parallelism. This feature is not the default file format for this release; however, the legacy binary and ASCII formats will be removed in a future version. More details on using this capability can be found in the manual.

1.7 Doppler Broadening Resonance Correction (`DBRC`) Treatment and Utility Code [MCNP-52943]

A Doppler broadening resonance correction (DBRC) treatment is implemented to address known deficiencies in the free-gas scattering model [6, 7]. Modifications to the free-gas scattering treatment that account for non-constant scattering cross sections have been proposed and tested in previous versions of the MCNP code [8–10]. With availability of 0-K nuclear cross sections that are needed to apply the DBRC treatment, the previously tested treatments are available through the new `DBRC` data card.

To use the DBRC treatment, data tables with preprocessed energy and scattering cross section pairs at 0 K are prepared using the **`dbrc_make_lib`** code compiled and installed with the MCNP6.3.0 executables. Both the **`DBRC_endf71.txt`** and **`DBRC_endf80.txt`** files distributed within the MCNP6.3.0 code package are products of the **`dbrc_make_lib`** code based on the 0-K scattering data from the ENDF/B-VII.1 or ENDF/B-VIII.0 nuclear data libraries, respectively. Further information on the DBRC code, data files, implementation and testing is available in a separate report [11].

1.8 HDF5 Runtape [MCNP-52968]

The previous runtape implementation has been replaced with one based on HDF5. This format has numerous advantages. It is more portable, it is supported in a very broad variety of programming languages, and it becomes possible to have backwards compatibility between versions going forward. Implementation details for this capability can be found in [12].

1.9 Unstructured Mesh HDF5-based XDMF Output and HDF5 Input File Format [MCNP-52987, MCNP-53029]

The MCNP Unstructured Mesh (UM) output capabilities have been extended to optionally produce HDF5-based XDMF files [13, 14]. These files are not produced by default. The HDF5 files are binary and hold geometry, material, and edit information for each UM instance organized in a hierarchical tree structure. The XDMF is an ASCII XML-formatted file that acts as a road map into the HDF5 file and can be directly loaded in visualization applications such as ParaView [15] or VisIt [16]. Furthermore, if HDF5 output is enabled, restart information is also written (to the MCNP runtape) and used, as applicable. Providing HDF5-based output and restart capability permits deprecating the legacy **`eeout`** formats while still providing the ability to inter-compare results in the current version of the MCNP code.

In addition to the HDF5-based XDMF output files, a new HDF5-formatted model description is now available to be used as the mesh input file. The UM model input file formats available in previous MCNP6 releases have been based on either an ASCII-formatted Abaqus mesh file or an MCNP-generated **`mcnpum`** file format. Both the new UM HDF5 input and output formats are consistent such that the MCNP6 input processing execution mode can read in an Abaqus-formatted mesh model and produce the new HDF5-formatted mesh model that can be used in a subsequent calculation with or without existing elemental edit results.

More information on both files and how HDF5-based restarts are performed is available in the MCNP® Code Version 6.3.0 Theory & User Manual [2].

1.10 Options to Reduce Memory Usage in Burn-up Problems [MCNP-53048]

Two new input options are now available to reduce memory usage for certain classes of problems. A new `NOSTATS` option was added to the `BURN` card to disable burn-up tally statistics. Additionally, a new `DISABLE` card was added for disabling features that consume significant quantities of memory. The only current option is *`nuclide_activity_table`* that, when disabled, can free up large quantities of memory on problems with many nuclides and cells, typical for large burn-up problems. See §2.3 for further details on the impact of these new features.

1.11 Mesh Tally XDMF Output Format [MCNP-53092]

A new output format, `xdmf`, is added to the `FMESH` card. As a result of adding this new output format, all mesh tally output formats other than `xdmf` and `none` are immediately deprecated.

Mesh tallies specified as output type `xdmf` will have the array data and associated attributes written to the runtape in two places. The first is the same location and organization as the `none` results that can be used to restart the calculation and/or within the interactive plotter. The second is the runtape **`/results/mesh_tally`** group with meta data going to a separate XDMF [13, 14] file, named **`meshtal.xdmf`** by default. This permits direct and hierarchical access to the mesh tally results in the runtape with a variety of programming languages and also straightforward 3-D visualization with third-party software such as ParaView [15] and VisIt [16].

1.12 Special Tally Treatments for Reactor Analysis [MCNP-53102]

Four new tally special treatment options (`FT` card) have been added to assist with reactor analyses:

`SPM` Collision exit energy-angle scatter probability matrices

`MGC` Flux weighted multigroup cross sections

`FNS` Induced fission neutron spectra

`LCS` Legendre coefficients for scatter reactions

These new multigroup tally capabilities have been thoroughly described and verified via code-to-code comparisons [17].

1.13 Added Optional Unstructured Mesh Quality Metric Calculation [MCNP-53181]

During MCNP UM calculation input processing, a variety of quality metrics are calculated and reported for linear tetrahedral and hexahedral elements and the Jacobian matrix determinant (Jacobian) is calculated and reported for other types of elements. Each metric has a recommended range, and if any elements are found outside of the range, then a warning is issued. Furthermore, an additional warning is issued if a non-positive Jacobian is found. These warnings do not halt the calculation; however, the user should scrutinize his or her mesh to confirm that it is acceptable for use in a Monte Carlo particle-transport calculation. If the mesh is known to be good, this input-processing step can be skipped by opting out on the `EMBED` card.

More information on this capability is available in [2] and [18].

1.14 Stochastic Temperature Mixing of Thermal Neutron Scattering Data ($S(\alpha, \beta)$) [MCNP-53206]

One possible approach to estimate the impact of a specific temperature on a system response is to stochastically mix two different data sets at bounding temperatures around a given desired temperature. For the nuclear data assigned on the material `M` card, this "stochastic mixing" capability has been available in the code by including two instances of the same isotope at different temperatures within a single material specification. By assigning appropriate relative atom/mass fractions for these two instances of the same isotope, the Monte Carlo random sampling process will then stochastically select an isotope at a given temperature to use in the collision physics effectively mixing the physics at each temperature.

The `MT0` card was added to allow a user to assign an $S(\alpha, \beta)$ table to a specific ZAID, in case a user wishes to experiment with stochastic mixing of both nuclear data and thermal neutron scattering data at a desired temperature. Note that this approach can be helpful in understanding the potential impact of using data a specific desired temperature; however, it is an approximation and can not be considered interpolation. To correctly obtain temperature-specific nuclear and/or thermal neutron scattering data, it is recommended to use the NJOY processing code [19].

1.15 New `FMESH` Tallying Backends [MCNP-53269]

There are now four possible `FMESH` tallying backends, accessible through the `TALLY` keyword. The default value is `fast_hist`, which corresponds to a high performance history-statistics-based tally. This backend yields identical results to previous versions of the code with a moderate performance and memory utilization improvement. `hist` is a slower algorithm only suitable for very small tallies and is present for future work. `batch` is a batch-statistics-based tally that will often outperform `fast_hist` in memory and performance, at the cost of lower-quality statistics. As such, the batch-based tallies come with a number of important caveats discussed in the manual that should be noted to ensure quality of results. Finally, `rma_batch` yields identical results to `batch`, but does so through MPI 3 remote memory access. When run on a large cluster, the tally can be distributed over multiple machines to allow for even larger tally sizes. These tallies are also compatible with the new `PIO` card, which enables or disables parallel HDF5 IO, which is useful on parallel file systems as the tally size enters the TiB range. If the remote memory access or parallel IO capabilities are of interest, one needs to build the code in a specific way, which is discussed in the build guide [3].

1.16 New Qt-based Plotter in Technology Preview [MCNP-53270]

The legacy X11 interface to the MCNP plotter has been reimplemented in Qt in a special technology preview version of MCNP6.3.0. While the value of the built-in graphical user interface (GUI) plotter is clear because it allows users to view and explore the exact geometry of MCNP calculations before running transport, a more modern GUI for the plotter has been desired for many years. For this technology preview, we selected to maintain a familiar interface with self-explanatory elements to ease the transition for experienced users and make the on ramp gentler for new users so that they could be productive with basic functionality without having to read the manual. After considering a number of possibilities we decided on Qt (https://qt.io), a cross-platform framework that provides portability with per-platform optimizations. User interface and operating system events are handled by Qt, so code to interpret events can be removed from the MCNP code base in the future. Additionally, Qt is open source, which allows us to compile it for platforms where a prebuilt binary is not available. Additional information on the executables and dynamic libraries provided for the technology preview of the Qt-based plotter can be found in §5.3.

1.17 Code Enhancements

1.17.1 Code Modernization and Standards Compliance

Many changes have been made in the MCNP code, version 6.3.0, for code enhancement and modernization so that the code is more robust and easier to maintain.

Fortran 2018 Compliance and C/C++ Standards The Fortran 2018 [20] standard identified a variety of language features as obsolescent. These features have not been officially removed from the standard at the time of the release of MCNP6.3.0, but are candidates for future removal from Fortran compilers. To prepare the code for future compiler versions that remove support for obsolescent features, the following Fortran language features have been removed in the MCNP6.3.0 source code:

- **`COMMON`** statements
- **`EQUIVALENCE`** statements
- **`FORALL`** statements
- **`NUMBERED DO`** loops

With several additions of C and C++ code into the overall MCNP6.3.0 source-code base, the C 99 [21] and C++ 14 [22] standards are specified in the build system. The compilers tested and recommended in the MCNP® Code Version 6.3.0 Build Guide [3] support enough of the language features within the Fortran 2018, C 99, and C++ 14 standards to be used for configuring, building, and executing the code.

Modernization Efforts To be able to continue to develop an efficient MCNP code on modern computing architectures and platforms, several modernization efforts have been undertaken and continue to be ongoing. While some of these efforts could lead to some minor improvements in the building and/or execution times of the code, many of these efforts do not result in significantly measurable changes in performance and may be solely for developer maintainability.

- Reorganize directories and files. The **`mcnp_global.F90`** routine was completely rearranged to group array specifications together according to function, rather than alphabetically or randomly. This file provides specification for persistent arrays that require dynamic memory allocation. Readability is improved by grouping arrays together by function—cells, surfaces, tallies, etc.
- Remove unused subroutines and functions.
- Fix 8-byte integer versus 4-byte integer compatibility issues. The interfaces are modified to allow 8- and 4-byte integers as necessary. This fix is only for the Intel Fortran compiler.
- Remove unnecessary pre-processor directives. All compiler and system preprocessor directives in **`ttyint.F90`** are removed.
- Use an **`mcnp_alloc`** subroutine to allocate and initialize global dynamic arrays. Consistent use of the **`mcnp_alloc`** routine improves code readability and provides consistent error-checking and initialization for the persistent arrays from **`mcnp_global`** in the **`dyn_allocate`** routine used during problem setup.
- Improve the interface between C and Fortran components of the code. Only the MPI C interface is now used, removing the need for building Fortran components.

Code Development Tool and Infrastructure Improvements

- Migrate the code repository from the LANL TeamForge server to the Atlassian suite of tools, including Bitbucket for repository management and JIRA for issue and resolution tracking. The primary motivation for this migration was to move the tools that support more modern code review workflows since all development within the MCNP code and supporting libraries must undergo a code review process using the industry standard method of pull/merge requests.
- Migrate the MCNP nightly build, test, and reporting system to a combination of the open-source automation server Jenkins (https://www.jenkins.io) and the open-source test reporting server CDash (https://www.cdash.org) tools. This migration improves developer efficiency by providing build feedback on all code review submissions.

1.17.2 **`DATAPATH`** and **`XSDIR`** Updates [MCNP-52867, MCNP-53030, MCNP-53284]

The internally-stored **`DATAPATH`** character array length has been extended to 1024 characters to accommodate exceptionally long paths to the location of the MCNP data installed on the system.

The default **`XSDIR`** file is now **`xsdir_mcnp6.3`**. The logic to fall back to using the **`xsdir`** name for the **`XSDIR`** file when the default cannot be found has been removed. Now, if the default **`xsdir_mcnp6.3`** file cannot be found, the user must specify the name of the **`xsdir`** file through the execution-line or message-block mechanisms.

1.17.3 Preliminary Support for ENDF/B-VIII.1 and Corresponding ACE Format Changes [MCNP-53428]

ENDF/B-VIII.1 is expected to have two changes that require modifications to the MCNP code. The first is mixed thermal scattering, in which both incoherent and coherent elastic scattering tables can be used simultaneously in an $S(\alpha, \beta)$ evaluation. The second is that ACE LAW 61 can now be used in photonuclear data for secondary energy and angle distributions. Both of these changes have been incorporated into MCNP6.3.0 in anticipation of ENDF/B-VIII.1. One should expect the MCNP team to publish further information, including V&V reports, as ENDF/B-VIII.1 develops.

1.17.4 Upgrade to CGMF 1.1.1 Correlated Fission Event Generator [MCNP-53435, MCNP-53517]

A new and improved version of the CGMF correlated fission event generator code, version 1.1.1, is integrated into the MCNP code. The new CGMF library and standalone executable is also available in the open-source software community hosted on the GitHub website within the LANL organization (https://github.com/lanl/cgmf). Live CGMF-specific documentation can be found at https://cgmf.readthedocs.io/en/latest. Some of the most notable physics improvements in CGMF 1.1.1 include:

- New spontaneous fission isotopes ^{240}Pu, ^{244}Pu, and ^{254}Cf
- New neutron-induced fission systems n + ^{233}U, n + ^{234}U, n + ^{237}Np, and n + ^{241}Pu
- Late-time prompt fission gamma rays
- Fission fragment angular distributions
- Pre-equilibrium neutron emission

In addition to the code and documentation available online, more information on recent changes within the CGMF code can be found in a recent article [23].

2 Performance

2.1 Improved Unstructured Mesh Abaqus Input File Processing [MCNP-53123, MCNP-53143]

The MCNP unstructured mesh (UM) geometry can be provided through an Abaqus-formatted ASCII mesh input file. Significant improvements have been made to the data structures and algorithms used when processing this file. Both the memory and setup-time improvements described in §2.1.1 and §2.1.2, respectively, are provided automatically with no additional input required from the user.

2.1.1 UM Memory Improvements

The memory required during mesh processing has been reduced by eliminating some memory allocations based on the most-limiting part by element count. For calculations with parts that contain a consistent number of elements per part, minimal improvement will be observed. For other models, results will vary, but for parts with varying numbers of elements, substantial memory savings may be possible (e.g., $\sim 50\%$ memory reduction for the UM model described in [24]). Also, the MPI communication paradigm has been changed to pass more messages that are smaller in size than previously, which can result in better network utilization. There are additional error messages issued if memory allocations happen to fail. These are most likely to occur if the UM model is large enough to exceed to the system memory resources, and are meant for identifying and diagnosing this issue.

2.1.2 UM Setup Time Improvements

The algorithm for identifying UM element neighbors has been changed, where the scaling has gone from $\mathcal{O}(N^2)$, where N is the maximum number of elements in any part/instance, to $\mathcal{O}(N)$. This improvement is accomplished by replacing a linear search with a hash-table lookup within the inner loop of the algorithm. For an example one-million element, single-part mesh, the new algorithm reduces the time spent searching for element neighbors from 12 hours to one second. However, the overall UM initialization process in MCNP6 involves more than just identifying element neighbors. For large meshes, it will still take from several minutes to tens of minutes for initialization to complete.

This improvement eliminates the guidance traditionally given that parts should have no more than $\sim 50,000$ elements.

2.2 Cross Section Cache Array Improvement [MCNP-52816]

During particle transport, caches of nuclear data table interpolation indices and cross section values, the **`KTC`** and **`RTC`** arrays, respectively, are stored during particle transport. The purpose of the caches is to make it faster and more efficient to retrieve cross section values when a particle table value is needed and the particle has not changed energy. These caches are sized according to the number of nuclear data tables needed in the simulation such that all isotopes have their own cached data. See §2.3 for more information about other relevant work that resulted in improving the sizing of the **`RTC`** array.

To invalidate the cache, indicating that new values would need to be retrieved and computed based on the particle behavior, the previous code version zeroed out the entirety of the **`KTC`** and **`RTC`** arrays at the start of each history. For simulations with a very large quantity of materials, zeroing out a large array of this cached data would happen frequently creating a bottleneck in performance. To remedy this performance degradation, the logic was updated by zeroing out the nuclear data caches only on parts that were needed for the problem.

For an example problem with 10k+ materials, the MCNP6.3.0 code runs 5 times faster than MCNP6.2.0. By only zeroing out the energy component in the used portion of the **`RTC`** array, the performance losses were recovered in the MCNP6.3.0 code such that large problems with many materials (e.g., full-core with depleted fuel) no longer suffer from this performance issue.

2.3 Burn-up Memory Reduction [MCNP-53048]

A few modifications have been made to reduce memory utilization. When using some or all of the new options, a significant reduction in memory consumption can be realized in depletion calculation. This can be especially useful for large models that also have many burnable

First, the **`RTC`** array, used to cache previously calculated cross sections, is now allocated to the proper size. Second, a new option was added to the `BURN` card, *`nostats`*, which disables the calculation of the statistics associated with depletion tallies. Finally, a `DISABLE` card was added for disabling features. The only current option is *`nuclide_activity_table`*, which when disabled can free up large quantities of memory on problems with many nuclides and cells.

The need to disable depletion tally statistics and the nuclide activity table depends on the problem size and the compute hardware available. Memory issues in depletion calculations may start to occur when the problem has 10k–100k burnable tally regions. However, given that the problem size and compute hardware directly impact the memory use and availability, it is difficult to generalize any guidance on when it may be necessary to use these new options.

2.4 `ACT` card `DG` = `line` option [MCNP-42400]

The source code for the `ACT` card, the `DG` = `line` option has been modified to decrease the memory footprint and drastically increase performance. For gamma decay of plutonium, MCNP6.3.0 runs 10 times faster and uses a third of the memory compared to MCNP6.2.0.

2.5 Other Modest Improvements

Through the course of several separate code efforts, some modest performance improvements may be present under certain circumstances. Some of these performance improvements include:

- As noted in §1.15, the new default `FMESH` tally backend is faster for generally all use cases.
- Anything utilizing new HDF5 formats can see computational cost improvements. Generally applicable across all of the new HDF5 file formats in comparison to their ASCII equivalent, the read/write speeds can be 20–50 times faster while also providing better numerical precision of the file contents. For the specific features using new HDF5-formatted files, some other computational cost improvements may exist:
  - For the new HDF5/XDMF **meshtal** format, in comparison to the ASCII **meshtal** file formats, the writing of the mesh data is significantly faster. Additionally, for MPI-based calculations with a parallel HDF5 library, the HDF5/XDMF **meshtal** format can be written in parallel removing a significant I/O bottleneck at the conclusion of a calculation.
  - For the new `PTRAC` HDF5 format, both thread-based and MPI-based parallelism are enabled. Besides the parallelism now available in `PTRAC` that can now be used with all other parallel-compatible features, modest computational speed improvements can also be observed due to the internal buffering of the particle data written to the **PTRAC** file.
  - For the new UM HDF5 input and elemental edit output format, reading in and processing this binary format during input processing is generally significantly faster than reading in and processing the Abaqus ASCII-formatted file. However, when comparing to reading/writing the now deprecated **mcnpum** binary input file format, the UM HDF5 file reading/writing is generally slower. These observations are unrelated to the additional improvements to UM setup algorithm described in §2.1.2. Also note that the UM HDF5 elemental output data is written using chunked and compressed storage that can result in a $\sim 10\times$ reduction in file size versus ASCII **eeout** files.
- Improvements to parts of the code to reduce non-uniform memory access (NUMA) issues have been integrated. NUMA issues are present when the code is run in parallel and a particular processor needs to access non-local memory. As an example, this can occur when one processor is used to allocate memory, which would typically be allocated locally with respect to the physical location of that processor, that is then accessed by all other processors regardless of their proximity to the memory. With slower connection speeds between the non-local processors and the local memory, fetching the data for reading or writing can slow down the code significantly. While NUMA issues still remain within the code, some have been removed to improve the performance when using certain features generally related to tallying quantities of interest.

- The built-in X11 plotter performance is improved. The drawing of the plot window is now done in the background, rather than continuously drawing the image to the screen. While it may appear that the plotter is now briefly stalling when starting to render the image, it appears this way because all of the work is being done in the background. A partial plot will appear within the window when it is done processing.

3 Verification and Validation (V&V)

The verification and validation testing for various application benchmark test suites is documented in MCNP® Code Version 6.3.0 Verification & Validation Testing [4]. In this separate V&V document, the entirety of the MCNP6.3.0 simulation results are given alongside either the analytical or experimental results for the verification or validation test suites, respectively.

When comparing the calculated results of MCNP6.2.0 to those of MCNP6.3.0, the results are either the same or agree according to the expected statistical uncertainties, indicating that MCNP6.3.0 is as accurate as MCNP6.2.0 for the range of problems tested. Both the MCNP6.2.0 results and the MCNP6.3.0 results are saved for reference within the **`vnvstats`** framework described further in §5.5.5. Here, the few changes within the source code between MCNP6.2.0 and MCNP6.3.0 that caused changes to the simulated V&V test results are discussed.

3.1 Intel Floating-Point Model Options

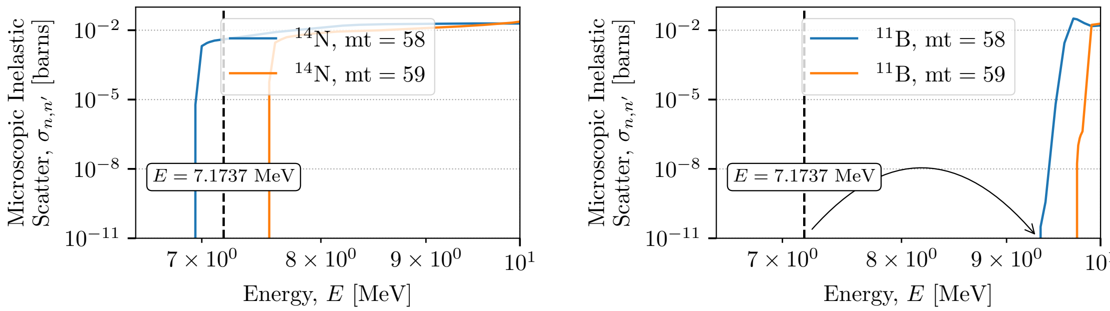
The change that impacted the most calculated results is the inclusion of the Intel compiler flag option that controls the semantics of floating-point calculations. By including the `-fp-model` = `consistent` on Linux / macOS and the `/fp` : `consistent` on Windows, the floating-point arithmetic within the Intel compiled executables now produce consistent, reproducible results across different optimization levels, for equivalent architectures. This option allowed the code optimization to be increased to **`O2`** with minimal differences, improving performance in some cases.

However, the change in arithmetic does lead to changes in random number usage, resulting in differences in calculated integral quantities in certain cases. As an example, $\sim 70\%$ of the validation results related to either criticality (**`criticality`** and **`crit_expanded`**) or reactor kinetics (**`rossi`**) are identical when using the Intel floating-point model flag. Contrast this with 100% identical results in the LLNL pulsed sphere validation calculations. Overall across all V&V test suites, less than half of the benchmark results calculated with MCNP6.3.0 are changed with respect to the results calculated with MCNP6.2.0. All these changes were found to be within the expected statistical uncertainties meaning that the MCNP6.3.0 performs comparably to MCNP6.2.0.

3.2 Thermal Neutron Scattering Data ($S(\alpha, \beta)$) Logic

Two independent changes within the thermal neutron scattering data ($S(\alpha, \beta)$) retrieval and caching routines resulted in changes to a few existing V&V test problems specifically within the **`criticality`** and **`crit_expanded`** test suites.

First, in an effort to prepare for upcoming changes to the ACE-formatted $S(\alpha, \beta)$ data to be included in ENDF/B-VIII.1 (see §1.17.3 and MCNP-53428 in Table 6), the data lookup and retrieval algorithm was slightly modified. Within these generally trivial modifications, the order of operations of a single math operation done during this step caused a change in the floating-point round-off which can then further potentially cause a change in the random number usage. Because the resulting differences are purely statistical, both old and new results are valid. The specific validation test



(a) Sampled Inelastic Scatter

(b) Attempted Inelastic Scatter

Figure 1: Inelastic Scattering Regimes for Boron and Nitrogen

problems that are impacted by this logic change include **TT2C11** in the **criticality** test suite and **heu-met-fast-026-case-c-11** and **u233-comp-therm-001-case-6** in the **crit_expanded** test suite.

Second, a minor $S(\alpha, \beta)$ data retrieval and caching bug was discovered and fixed (see MCNP-53470 in Table 7). This issue manifests itself when a model includes two neighboring cells with materials that have overlapping nuclides where only one of the materials includes the thermal neutron scattering treatment (`MT` card). In versions of the MCNP code, dating back to versions within the 4 series, the cached nuclide cross sections do not properly handle the addition and removal of the $S(\alpha, \beta)$ data contribution to the total cross section unless the cached nuclide cross sections are otherwise invalidated (i.e., the neutron energy changes). Therefore, by using this cache without having adjusted it properly to handle the addition or subtraction of the $S(\alpha, \beta)$ data from the nuclide cross section, the first flight of the neutron as it enters this neighboring cell can use an incorrect total cross section. Under extreme circumstances when trying to highlight the impact of this bug, the absolute bias in k-effective was on the order of ~ 0.00010. Because this bug has such limited impact and can only be present under rare geometry/material configuration/definition scenarios, the resulting impact is nearly undetectable in realistic problems. The validation test problems that are impacted by this fixed bug include **SB5RN3** in the **criticality** test suite and **u233-comp-therm-001-case-6** in the **crit_expanded** test suite.

4 Discussion of Significant Issues

4.1 Unstructured Mesh Pseudocell Surface-crossing Considerations

Late in the software-release cycle, a reproducible issue was provided to the MCNP development team that resulted in the bad-trouble message:

```
bad trouble in subroutine acecas of mcrun
emission energy was negative.
```

The issue arises from an internal code-state inconsistency regarding a computational particle's understanding of which cell/material it is in. As a result, (in this instance) the particle sampled an inelastic scattering collision with an incident energy of roughly 7.2 MeV believing it was in nitrogen but then attempted to perform the scatter within boron, which is energetically infeasible (see Fig. 1). The nuclear-data-processing check for this issued the aforementioned "bad trouble" message and the code aborted.

Table 3: ZAIDs Published by LANL with Missing **DLWH** for Recoil

Target	Proton	Deuteron	Triton	Helion
^{3}H	**1003.70h/71h/00h**			
^{3}He	**2003.70h/71h/00h**	**2003.70o/00o**	**2003.70r/00r**	
^{4}He	**2004.71h/00h**	**2004.70o/00o**	**2004.70r/00r**	**2004.70s/00s**

Following investigation, this issue is determined to have arisen from applying weight windows to an unstructured mesh (UM) calculation such that during a particle's surface crossing between two adjacent cells, an inconsistent particle state is introduced. What follows is a narrative of the events that lead to this behavior and a suggested workaround.

In this case, during surface crossing (in the **surfac** routine), the particle's cell attribute is correctly updated but then later surface-crossing-based weight windows are called that again update the cell to the background as the result of a UM cell lookup that identifies the particle as existing in the overlap (i.e., on the two co-planar element faces between the elements/pseudocells). Thus, the particle enters a pseudocell thinking it is in the background. This discrepancy leads to a desynchronization between the particle's cell attribute and the material-lookup data structure (**mdata**). Because of the cell-lookup operation occurring at the interface of two elements/pseudocells, the issue also depends on floating-point mathematics that can be subject to numerical precision issues such that the occurrence of this issue is relatively rare and simply reproducing this behavior is difficult.

A possible workaround to the root cause of the negative-emission-energy issue is to disable surface-crossing-based weight windows using the *mwhere* (fourth) option on the weight-window parameter (`WWP`) card. However, when this was attempted it was recognized that the logic within the code to control where weight-window processing occurs is incompatible with UM geometries. Thus, the logic was updated to be compatible, test cases covering these behaviors were added, and this is the recommended workaround to avoid surface-crossing-based weight-window misbehavior in the presence of UM geometries until such time as the desynchronization is fully addressed (see MCNP-53411 in Table 7).

As a result of this work, it is recommended that users scrutinize other calculations that contain surface-crossing-based banking operations with UM geometries (e.g., importance splitting). Further, users are encouraged to review historic calculations that used these features in combination to ensure that no unwanted behavior was silently present.

4.2 Charged Particle ACE Table Issues

Several issues were found when doing charged particle transport in the tabular energy regimes. Within the ACE file, there is a block of data that contains secondary particle production data (specifically the **ANDH**/**DLWH** blocks indexed by **IXS**). Some libraries placed the recoil particle in this block. However, within NJOY, there was not enough information to generate an outgoing energy spectra for the recoil particle. As a result, NJOY would leave that data missing. The entire list of known affected libraries are shown in Table 3. This technically could occur with neutron data, but no dataset was found that exhibited this behavior.

The MCNP code, however, relied on the presence of that data to function. As the data was missing, it would look in an arbitrary portion of the nuclear data and interpret it as if it were a secondary energy distribution. This results in incorrect outgoing energies. For specific data, it would read off the end of the nuclear data array and cause memory issues.

While investigating this issue, a second was found. When light-ion recoil kinematics (`PHYS:H` *recl*, for all charged particle projectiles, or `PHYS:N` *coilf*, for neutron projectiles) was enabled and the recoil particle production data was in these secondary production blocks, the number of recoil particles produced was doubled. The first was being produced via the secondary production table and the second was being produced with the light-ion recoil physics immediately after scattering.

In the MCNP code, version 6.3.0, both of these have been fixed with one change. If the secondary production block includes anything relating to elastic scatter, those segments of the secondary production block are ignored. Instead, the code will perform scatter (using the data within the **ESZ** block) as it has done prior. Then, recoil is only produced via the light-ion kinematics route. This produces the correct number of particles and the resulting particles exhibit conservation of energy, which was not true in MCNP6.2.0. A warning will be issued if the problematic data is detected noting that this will occur. This warning will be removed in the future once the nuclear data and the code are in harmony. This is issue MCNP-26747 in Table 7.

Finally, two minor changes were made after fixing this issue. First, the documentation on `PHYS:H` *`recl`* was updated to indicate that it enables light-ion recoil for all charged particle projectiles and not just protons. Second, the library **`3006.70h`** was found to be incorrectly processed and has an invalid structure. As such, it was removed from the MCNP testing suite and is not recommended for use.

5 Distribution and Installation

The MCNP6.3.0 code distribution package contains the source code, if requested (§5.1), the production and technology preview executables (see §5.2 and §5.3, respectively), various model physics data (§5.4), and several supplemental utilities and scripts (§5.5). Due to the significant differences when compared to the MCNP6.2.0 code distribution package, each section herein describing the code, data, and miscellaneous utilities should be read thoroughly. Additional details for each individual component of the package should also be read and are available in the multitude of **README** files included throughout the distribution.

Figure 2 shows the high-level layout of the MCNP6.3.0 code distribution. Many of the shown directories and files are referenced in the sections to follow.

Figure 2: General Layout of the MCNP6.3.0 Code Distribution

```
MCNP6.3.0
├── README.html ............................................ (file)
├── binaries ........................................... (directory)
│   ├── install_linux_mac.sh ............................... (file)
│   └── install_windows.bat ................................ (file)
├── data ............................................... (directory)
│   └── nd_manager ......................................... (directory)
├── docs ............................................... (directory)
├── licenses ........................................... (directory)
├── mcnp-src ............................... (optional) (directory)
└── utils .............................................. (directory)
    ├── isc ............................................ (directory)
    ├── um_pre_op ...................................... (directory)
    ├── um_pos_op ...................................... (directory)
    ├── vnvstats ....................................... (directory)
    └── whisper ........................................ (directory)
```

Table 4: Production MCNP6.3.0 Executable and Dependency Information

Executable Name	Operating System	Intel oneAPI Version	HDF5 Version	MPI Version
mcnp6	Linux	2021.5.0	1.10.8	*
mcnp6.ompi	Linux	2021.5.0	1.10.8	OpenMPI 4.1.4
mcnp6.mpi	Linux	2021.5.0	1.10.8	MPICH 4.0.2
mcnp6	macOS	2021.5.0	1.10.7	*
mcnp6.ompi	macOS	2021.5.0	1.10.7	OpenMPI 4.1.1
mcnp6.exe	Windows	2021.7.0	1.12.1	*
mcnp6.mpi.exe	Windows	2021.7.0	1.12.1	MS-MPI 10.1
Note: All executables configured with OpenMP enabled				
* Default configuration with MPI disabled				

5.1 MCNP6.3.0 Source Code

With the MCNP6.3.0 source code, it is possible to explore many build configurations with or without source code modifications. The most common reason to obtain the source code is to build an MPI-parallel version for use on multi-processor systems. While some production MPI executables are now distributed on each supported operating system, it may be necessary to obtain the source code to build an MPI-parallel version due to any unforeseen incompatibility issues with the distributed MPI versions. See the discussion on the production MPI executables in §5.2.

All details regarding building the code can be found in the MCNP® Code Version 6.3.0 Build Guide [3]. The compressed source code can be found in the **mcnp-src** directory of the distribution.

5.2 Production MCNP6.3.0

The executables bundled in the code distribution package are built for Linux, macOS, and Windows operating systems. In addition to the MCNP6.3.0 executables, the utilities described in Appendix E of the MCNP® Code Version 6.3.0 Theory & User Manual [2] are also built and packaged alongside the production executables. Table 4 includes all of the production executables by name for each operating system along with the versions of the Intel oneAPI Classic Compiler, HDF5 library, and MPI library used to build each distributed executable. The **install_linux_mac.sh** and **install_windows.bat** scripts in the binaries directory of the distribution shown in Fig. 2 are the installers for Linux/macOS and Windows, respectively.

The production MPI builds are considered "best-effort builds" because ensuring portability of MPI applications is more complex than serial applications. To use the MPI executables, a compatible MPI library must be installed on the system. The binary installers will attempt to detect if a compatible MPI library is available on the system before installing the distributed MPI executables. The **README**(s) alongside the binary installers in the binaries directory of the code distribution package includes more detailed system and compatibility information.

Each production executable is packaged with a variety of operating-system-dependent dynamic libraries (**.so** on Linux, **.dylib** on macOS, and **.dll** on Windows) needed by the MCNP6.3.0 code. The libraries are installed when the executables are installed. All of the associated licenses related to the distributed third-party dynamic libraries are included in the top-level licenses directory within the distribution.

5.3 Qt-based Plotter Technology Preview

Alongside the production MCNP6.3.0 executables provided within the code distribution package, a separate collection of technology preview MCNP6.3.0 executables are provided that contain the Qt-based plotter described in §1.16. Each of these executables, one each for Linux, macOS, and Windows operating systems, are **runtpe**-compatible with the production MCNP6.3.0 executables. That means that the results from calculations using the production executables can be visualized using either the production legacy plotter or the technology preview Qt plotter.

While the technology preview Qt plotter executables can also be used to do full MCNP simulations, including the Monte Carlo particle transport, these executables have not been rigorously verified and validated for performing full transport simulations. It is highly recommended to use the production MCNP6.3.0 executables for performing full transport simulations, and only using the Qt-based technology preview MCNP6.3.0 executables for geometry, tally, and cross section visualization purposes. Chapter 7 of the MCNP® Code Version 6.3.0 Theory & User Manual contains dedicated information on the new Qt-based plotter technology preview [2].

The Qt-based plotter technology preview executables are distributed with MPI disabled. One difference between the distributed Qt and production packages is the inclusion of a variety of dynamic Qt libraries specific to each operating system. These dynamic Qt libraries are distributed with the code such that Qt is not required to be installed by the user. However, if the user would prefer to install their own or use a system-installed version, the distributed Qt dynamic libraries can be removed in favor of a separately installed Qt library. Qt version 5.15.2 was used to build each of the technology preview executables, named **mcnp6.qt** and **mcnp6.qt.exe** for Linux / macOS and Windows, respectively.

Users are strongly encouraged to provide feedback on their experience with the Qt plotter to mcnp_help@lanl.gov as the MCNP Development Team continues to improve it and move toward it as the single plotting utility provided in a future release.

5.4 Nuclear, Atomic, and Model Physics Data

Unlike the MCNP6.2.0 and earlier distributions, the MCNP6.3.0 distribution is not bundled with all of the nuclear and atomic data needed to run the code for many applications. Instead of distributing the data only alongside MCNP6 distributions, all of the ACE-formatted data is now publicly available on the https://nucleardata.lanl.gov website including all of the same data that was distributed with the MCNP6.2.0 code. The non-ACE-formatted model physics data is included on the MCNP6.3.0 distribution.

With this change in the distribution contents, the installation process is now more complicated with the entirety of the data now split between the local distribution of the model physics data and the remotely-hosted nuclear and atomic data libraries. To help make the data installation process more manageable, the **nd_manager** utility discussed further in §5.5.3 is provided with the distribution to facilitate an easy and reliable installation.

5.4.1 Nuclear and Atomic ACE-formatted Data

The data on the https://nucleardata.lanl.gov website includes all of the ACE-formatted data that MCNP6.3.0 can use. All of this data is separated into individual libraries and archived as compressed tarballs or zip files along with the corresponding documentation and necessary **xsdir** file entries.

When new libraries are processed and made available on the website, they can be immediately downloaded and installed for use. It is highly recommended to use the **nd_manager** utility (§5.5.3) to setup, download, install, and update the local data installation needed for the code.

If the **nd_manager** utility cannot be used, it is possible to manually setup, download, install, and update the local data. However, this process may be cumbersome and potentially error prone if not done carefully. For tips on manually installing the data for MCNP6.3.0, see the latest release page https://mcnp.lanl.gov/release_630.html on the MCNP website.

5.4.2 Model Physics Data

The model physics data is included locally within the MCNP6.3.0 distribution. It can be found in the **data/nd_manager/builtin/MCNP_6.3_DATA** directory of the distribution.

The distributed model physics data is organized in a fashion such that the **nd_manager** utility can perform the installation without manual intervention by the user. Once again, it is highly recommended to use the **nd_manager** to handle the model physics data installation needed for the code.

If the **nd_manager** utility cannot be used, it is possible to manually install this local data. See the latest release page https://mcnp.lanl.gov/release_630.html on the MCNP website for tips on manually installing the data for MCNP6.3.0.

5.5 Miscellaneous Utilities and Scripts

In addition to the utilities described in Appendix E of the MCNP® Code Version 6.3.0 Theory & User Manual [2] several separate utilities are distributed as part of the overall MCNP6.3.0 code package. Each separate utility is briefly described herein including some information on the whereabouts of the utilities within the distributed package.

5.5.1 Intrinsic Source Constructor (ISC)

A new version of the Intrinsic Source Constructor, ISC 2.1.0 [25], is included within the MCNP6.3.0 installation. The distribution includes:

- The complete ISC 2.1.0 source code and data packaged into a compressed archive within the **utils/isc** directory of the distribution. Decompress the **isc-2.1.0.zip** archive and get started by reading the **README**.
- New **misc** and **mattool** binaries for Linux, macOS, and Windows. The MCNP code installation step of the installer handles installing these executables into the same path with the production MCNP6.3.0 executable.
- Python 3.9 and 3.10 wheels built for Linux, macOS (both x86-64 and ARM64 architectures), and Windows located in the **binaries/bin/isc** directory of the distribution. The installer does not install the Python wheels; see §2.4 of [25] for guidance on installing the wheels.
- Updated ISC data files. The data installation step of the installer handles installing the ISC data through the **nd_manager** utility (§5.5.3).

> **⚠ Caution**
>
> The **nd_manager**, invoked by the installer during the data installation step, does require the use of Python. If Python is not available, the ISC data archive **isc-data-2.1.tar.xz** can be decompressed in any preferred location on the filesystem. To use the data through the binaries or Python wheels, the **ISCDATA** environment variable must be set to the location of the installed data. On the distribution, the ISC data archive is located in the **data/nd_manager/builtin/MCNP_6.3_DATA/ISC_DATA-2.1** directory.

The main new features and improvements in the ISC 2.1.0 release are specific to the updated data and the **misc** utility. The improvements to the data (or use of the data) include:

- ENDF-VIII.0 data libraries are added.
- Proton decay is fixed. Previously, the atomic mass number (A) was not decremented for the daughter.
- ^{252}Cf neutron intensity is reduced by a factor of 31.4. Previously, the branching ratio was calculated incorrectly.

The new or enhanced features in the **misc** utility include:

- Additional options are available in the `biasing` keyword input.
- New `bias_limit` and `bias_order` keywords are available to support the new `biasing` options.
- A new `intensity_ratio` keyword is available to set a low-intensity threshold for accepted discrete lines.

Further information on improvements within **misc** are available in the updated user guide [26].

5.5.2 MCNPTools

The latest version of MCNPTools is released as open-source software on GitHub under the LANL organization at https://github.com/lanl/mcnptools. While documentation corresponding to the specific version of the source code can be found within the repository, the MCNPTools installation and use document published at the time of the MCNP6.3.0 release is also available [27].

5.5.3 nd_manager Nuclear Data Downloader

To support the nuclear data installation and updating process, the Python-based **nd_manager** utility is provided with the distribution in the **data/nd_manager** directory. The primary functions that the **nd_manager** serves includes:

- Local updates based on remote (or local) database changes.
- Listing available libraries to download and install.
- Downloading all or user-specified libraries.
- Decompress and install downloaded libraries.
- Create and/or update **xsdir** files when libraries are (un)installed.

For more information on the functionality and example usage of the **nd_manager**, see the **README** included with the **nd_manager** utility.

If the user elects to install the data through the MCNP6.3.0 installer, the **nd_manager** is configured on the system to handle both the remote https://nucleardata.lanl.gov database of ACE-formatted libraries and the local **data/nd_manager/builtin/MCNP_6.3_DATA** database of non-ACE-formatted model physics libraries. Additionally, the **nd_manager** is configured to handle MCNP-version-specific exceptions regarding known data files that are incompatible with various versions of the code. This information is maintained on the https://mcnp.lanl.gov website and is used when constructing the default **xsdir** files used by many versions of the MCNP code. The relevant local and remote files configured when the data installer is initiated includes:

- https://nucleardata.lanl.gov/libraries.json includes the LANL nuclear data team website database
- **data/nd_manager/builtin/MCNP_6.3_DATA/libraries.json** includes the MCNP6.3.0 distribution model physics database
- https://mcnp.lanl.gov/include/mcnp_support.json includes the LANL MCNP team website **xsdir** exceptions list

When the **nd_manager** is requested to perform an update, all of the configured libraries and exceptions listed above are queried for updates. If the library or exception files have been modified since the last time the **nd_manager** queried these files, it can update the local installation if requested. The format specification the **nd_manager** expects is described in [28].

The **nd_manager** utility is highly recommended to use to manage the nuclear data installation and updating process. If the full installation or data installation steps are selected and the appropriate version of Python is available on the system, the MCNP6.3.0 installer will configure and use the **nd_manager** to install the data.

⚠ Caution

When the **nd_manager** is used within the MCNP6.3.0 installer, the `-all production` flag is passed to the `download`, `install`, and `create-xsdir` steps. As a result, the data in the **xsdir** files is organized in reverse chronological order of the individual library release date. In comparison to previous releases of the MCNP6 code, this changes the ordering within the **xsdir** file to default to the EPRDATA14 photon data rather than the MCPLIB84 photon data.

5.5.4 Unstructured Mesh (UM) Scripts

With significant modifications to the UM file formats and input/output processing, both the legacy **um_pre_op** and **um_post_op** programs described in Appendix E of the MCNP® Code Version 6.3.0 Theory & User Manual [2] are deprecated (see §1.1). New Python-based utilities are now available to provide some new capabilities to support MCNP input file setup and calculated results extraction.

um_pre_op Given the complexity of correctly setting up UM geometry in an MCNP6 calculation, the **um_pre_op** scripts help to both ensure a mesh file meets the requirements of the MCNP6.3.0 code, and setup a skeleton MCNP input file for use with a properly formatted mesh file. The **utils/um_pre_op** directory in the distribution contains the Python scripts and further documentation describing and demonstrating use of the utility for various applications.

um_pos_op With new HDF5-formatted UM files potentially produced and the deprecation of the legacy file formats, the **um_pos_op** Python scripts help extract results from the elemental edit output (HDF5 and legacy **eeout**) files. The extracted results can be used for further post-processing, or they can be used to perform comparisons between both sets of results written to the different formats. The **utils/um_pos_op** directory in the distribution contains the Python scripts and further documentation describing and demonstrating use of the utility.

5.5.5 vnvstats Verification and Validation Testing Framework

The new Python-based **vnvstats** framework is used to completely setup, execute, post-process, and document several suites of benchmark problems used for verification and validation of the MCNP code.

For verification, where the code is tested against analytical or semi-analytical benchmarks using mock data, the goal is to ensure the algorithms and methods within the MCNP code are suitable for accurately solving particle transport problems. The verification suites established within the framework include analytic k-eigenvalue problems and analytic fixed-source duct-streaming problems.

For validation, where the code and real data are tested together against experiment benchmarks, the goal is to ensure the predictions of the MCNP code are consistent with reality for a variety of applications. The validation suites established within the framework include nuclear criticality safety and reactor kinetics applications, neutron time-of-flight fixed-source applications, electron energy-deposition applications, and high-energy neutron, proton, and deuteron physics applications.

The MCNP® Code Version 6.3.0 Verification & Validation Testing [4] document contains all of the comparisons between simulated results and the benchmark values within the **vnvstats** framework. A summary of the results is given in §3. To get started using the new framework, find the **README** in the **utils/vnvstats** directory within the distribution.

5.5.6 Whisper

The Whisper-1.1 [29] package is available to provide sensitivity-uncertainty capabilities that may be used to support nuclear criticality safety validation. The Whisper code, supporting scripts, criticality safety benchmark problems (with input files and sensitivity profiles), and covariance files for the nuclear data are the same as those distributed with the MCNP6.2.0 code. The **README** and installation script, found in the **utils/whisper** directory within the distribution, have been updated with recent information to help guide the installation process. From theory to usage to software quality assessments, the reference collection on the MCNP website (https://mcnp.lanl.gov/reference_collection.html) contains many documents related to Whisper.

6 Software Quality Assurance

For all MCNP6 development, including source code changes, testing, documentation, and code releases, the MCNP Development Team follows a software quality assurance (SQA) plan defined by LANL [30]. While the SQA plan that the MCNP6 code is developed under is in part derived from other quality control and nuclear safety standards, it cannot be claimed that the MCNP6 code strictly follows any of these other standards.

At LANL, the code is categorized as non-safety commercially controlled software for all general applications. Therefore, the code should not be used for safety significant applications unless qualified to do so by individual users of the code for their specific areas of application. As part of the qualification of the MCNP6 code for specific applications, it is recommended and may be required that a suite of qualification tests be developed to cover the application areas of interest beyond those applications presented in the MCNP® Code Version 6.3.0 Verification & Validation Testing report [4].

Acknowledgements

Research and development support for this version of the MCNP code was provided by a number of organizations including the US Department Of Energy's NNSA Advanced Strategic Computing program, the Nuclear Survivability program, the Nuclear Criticality Safety program, and Los Alamos National Laboratory Directed Research and Development (LDRD) program, the MCNP Site Support program, and a software code modernization project.

We thank these sponsors.

References

[Citing pages are listed after each reference.]

1. C. J. Werner, J. S. Bull, C. J. Solomon, F. B. Brown, G. W. McKinney, M. E. Rising, D. A. Dixon, R. L. Martz, H. G. Hughes, L. J. Cox, A. J. Zukaitis, J. C. Armstrong, R. A. Forster, and L. Casswell, "MCNP® Version 6.2 Release Notes," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-18-20808, 2018. URL: http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-18-20808 [Pages 3 and 46]

2. J. A. Kulesza, T. R. Adams, J. C. Armstrong, S. R. Bolding, F. B. Brown, J. S. Bull, T. P. Burke, A. R. Clark, R. A. Forster, III, J. F. Giron, A. S. Grieve, C. J. Josey, R. L. Martz, G. W. McKinney, E. J. Pearson, M. E. Rising, C. J. Solomon, Jr., S. Swaminarayan, T. J. Trahan, S. C. Wilson, and A. J. Zukaitis, "MCNP® Code Version 6.3.0 Theory & User Manual," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-30006, Rev. 1, Sep. 2022. DOI: 10.2172/1889957 [Pages 3, 7, 8, 18, 19, 20, and 22]

3. J. S. Bull, J. A. Kulesza, C. Josey, and M. E. Rising, "MCNP® Code Version 6.3.0 Build Guide," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-32851, Rev. 1, Dec. 2022. DOI: 10.2172/1906011 [Pages 3, 6, 9, and 18]

4. C. Josey, A. R. Clark, J. A. Kulesza, E. J. Pearson, and M. E. Rising, "MCNP® Code Version 6.3.0 Verification & Validation Testing," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-32951, Rev. 1, Dec. 2022. DOI: 10.2172/1907750 [Pages 3, 14, 23, and 24]

5. F. B. Brown, C. Josey, S. Henderson, and W. R. Martin, "Automated Acceleration and Convergence Testing for Monte Carlo Criticality Calculations," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-19-23887, Aug. 2019. URL: https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-19-23887 [Page 5]

6. R. R. Coveyou, R. R. Bate, and R. K. Osborn, "Effect of Moderator Temperature Upon Neutron Flux in Infinite, Capturing Medium," *Journal of Nuclear Energy (1954)*, vol. 2, no. 3-4, pp.

153–167, 1956. URL: https://www.sciencedirect.com/science/article/pii/0891391955900309 [Page 6]

7. M. Ouisloumen and R. Sanchez, “A Model for Neutron Scattering Off Heavy Isotopes That Accounts for Thermal Agitation Effects,” *Nuclear Science and Engineering*, vol. 107, no. 3, pp. 189–200, 1991. DOI: 10.13182/NSE89-186 [Page 6]

8. B. Becker, R. Dagan, and G. Lohnert, “Proof and Implementation of the Stochastic Formula for Ideal Gas, Energy Dependent Scattering Kernel,” *Annals of Nuclear Energy*, vol. 36, no. 4, pp. 470–474, 2009. DOI: 10.1016/j.anucene.2008.12.001 [Page 6]

9. E. E. Sunny, F. B. Brown, B. C. Kiedrowski, and W. R. Martin, “Temperature Effects of Resonance Scattering in Free Gas for Epithermal Neutrons,” Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-11-04886, Aug. 2011. URL: https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-11-04886

10. E. E. Sunny, F. B. Brown, B. C. Kiedrowski, and W. R. Martin, “Temperature Effects of Resonance Scattering for Epithermal Neutrons in MCNP,” in *Proceedings of PHYSOR—Advances in Reactor Physics—Linking Research, Industry, and Education*, Knoxville, TN, USA. April 15–20, 2012, pp. 15–20, Los Alamos National Laboratory Tech. Rep. LA-UR-11-06503. URL: https://www.osti.gov/biblio/1121992 [Page 6]

11. F. B. Brown, “Doppler Broadening Resonance Correction for Free-gas Scattering in MCNP6.2,” Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-19-24824, May 2019. DOI: 10.2172/1523218 [Page 6]

12. C. Josey, “Converting the MCNP Runtape to HDF5,” Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-19-29393, Sep. 2019. DOI: 10.2172/1566077 [Page 7]

13. J. A. Clarke and E. R. Mark, “Enhancements to the eXtensible Data Model and Format (XDMF),” in *HPCMP User’s Group Conference 2007. High Performance Computing Modernization Program: A Bridge to Future Defense*, Pittsburgh, PA, USA; June 18–21, 2007, pp. 322–327. [Page 7]

14. “XDMF Model and Format,” Website, Apr. 2020. URL: http://xdmf.org/index.php/XDMF_Model_and_Format [Page 7]

15. J. P. Ahrens, B. Geveci, and C. Law, *Visualization Handbook*. Burlington, MA, USA: Elsevier Inc., 2005, ch. ParaView: An End-user Tool for Large-data Visualization, pp. 717–731. URL: https://www.sciencedirect.com/book/9780123875822/visualization-handbook [Page 7]

16. “VisIt User’s Manual,” Lawrence Livermore National Laboratory, Livermore, CA, USA, Tech. Rep. UCRL-SM-220449, Oct. 2005. URL: https://wci.llnl.gov/simulation/computer-codes/visit/manuals [Page 7]

17. R. B. Wilkerson, G. W. McKinney, M. E. Rising, and J. A. Kulesza, “MCNP Reactor Multigroup Tally Options Verification,” Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-20-27819, Oct. 2020. URL: https://mcnp.lanl.gov/pdf_files/TechReport_2020_LANL_LA-UR-20-27819_WilkersonMcKinneyEtAl.pdf [Page 8]

18. J. A. Kulesza, “MCNP® Code Version 6.3.0 Unstructured-mesh Quality Metrics & Assessment,” Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-20-27150, Rev. 1, Sep. 2022. DOI: 10.2172/1888190 [Page 8]

19. R. E. MacFarlane, D. W. Muir, R. M. Boicourt, A. C. Kahler III, and J. L. Conlin, "The NJOY Nuclear Data Processing System, Version 2016," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-17-20093, Jan. 2017. DOI: 10.2172/1338791 [Page 8]

20. "Information technology — Programming languages — Fortran — Part 1: Base language," International Organization for Standardization, Geneva, Switzerland, ISO/IEC Standard 1539-1:2018, Nov. 2018. URL: https://www.iso.org/standard/72320.html [Page 9]

21. "Programming languages — C," International Organization for Standardization, Geneva, Switzerland, ISO/IEC Standard 9899:1999, Dec. 1999, withdrawn. URL: https://www.iso.org/standard/29237.html [Page 9]

22. "Information technology — Programming languages — C++," International Organization for Standardization, Geneva, Switzerland, ISO/IEC Standard 14882:2014, Dec. 2014, withdrawn. URL: https://www.iso.org/standard/64029.html [Page 9]

23. P. Talou, I. Stetcu, P. Jaffke, M. E. Rising, A. E. Lovell, and T. Kawano, "Fission Fragment Decay Simulations with the CGMF Code," *Computer Physics Communications*, vol. 269, p. 108087, 2021. DOI: 10.1016/j.cpc.2021.108087 [Page 11]

24. J. A. Kulesza and R. L. Martz, "Evaluation of the Pool Critical Assembly Benchmark with Explicitly Modeled Geometry Using MCNP6's Unstructured Mesh Capabilities," in *Proceedings of 16th International Symposium on Reactor Dosimetry (ISRD16)*. Santa Fe, NM, USA; May 7–12: ASTM International, 2017, accompanying presentation: LA-UR-17-23854. DOI: 10.1520/STP160820170041 [Page 11]

25. C. J. Solomon, Jr., C. R. Bates, J. A. Kulesza, and M. J. Marcath, "The Intrinsic Source Constructor Package: Installation and Use," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-31848, Nov. 2022. DOI: 10.2172/1897420 [Page 20]

26. C. J. Solomon, Jr., C. R. Bates, and M. J. Marcath, "MCNP Intrinsic Source Constructor (MISC): A User's Guide," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-32893, Dec. 2022. DOI: 10.2172/1903539 [Page 21]

27. C. R. Bates, S. R. Bolding, C. J. Josey, J. A. Kulesza, C. J. Solomon, Jr., and A. J. Zukaitis, "The MCNPTools Package: Installation and Use," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-28935, Aug. 2022. DOI: 10.2172/1884737 [Page 21]

28. C. Josey and J. L. Conlin, "LANL Nuclear Data Manager Format Specification v1.0," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-28306, Aug. 2022. DOI: 10.2172/1880470 [Page 22]

29. F. B. Brown, M. E. Rising, and J. L. Alwin, "User Manual for Whisper-1.1," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-17-20567, Jan. 2017. DOI: 10.2172/1341834 [Page 23]

30. M. E. Rising, F. B. Brown, J. R. Salazar, and J. E. Sweezy, "Overview of the MCNP6® SQA Plan and Requirements," Los Alamos National Laboratory, Los Alamos, NM, USA, Memorandum XCP-3:20-008/LA-UR-20-26666, Aug. 2020. DOI: 10.2172/1657095 [Page 23]

31. J. A. Kulesza and T. C. McClanahan, "A Python Script to Convert MCNP Unstructured Mesh Elemental Edit Output Files to XML-based VTK Files," Los Alamos National Laboratory, Los

Alamos, NM, USA, Tech. Rep. LA-UR-19-20291, rev. 2, Nov. 2019. DOI: 10.2172/1574726 [Page 36]

32. V. K. Mehta and J. C. Armstrong, “Processing MCNP Elemental Edit Outputs,” Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-20-24025, Jun. 2020. DOI: 10.2172/1631564 [Page 36]

33. J. A. Favorite and D. K. Parsons, “(U) Energy-Dependent Perturbations of Scattering Cross Sections Using the PERT Card in MCNP6.2,” Los Alamos National Laboratory, Los Alamos, NM, USA, Memorandum LA-UR-20-24082, Jun. 2020. URL: https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-20-24082 [Page 39]

34. ICRP, “1990 Recommendations of the International Commission on Radiological Protection,” *Annals of the ICRP*, vol. 21, no. 1–3, 1991. DOI: 10.1016/0146-6453(91)90065-O [Page 45]

A New Features, Code Enhancements, Closed Bugs, and Known Issues

This appendix is a summary of new features (Table 5), code enhancements (Table 6), and bug fixes that have been resolved since the last MCNP release (Table 7). Known issues that still need to be addressed are listed in Table 8. Users should familiarize themselves with these items to ensure that the problems they have and will run are not impacted.

Table 5: New Features

Tracking Number	Category	Description
MCNP-5367, MCNP-52997	Tallies	Added mesh tally `colsci` and `cfsci` output formats. See §1.2 for further details.
MCNP-28788	KCODE	Fission matrix-based k-eigenvalue convergence and acceleration added. See §1.3 for further details.
MCNP-48179, MCNP-50999, MCNP-52871	Build System	Fully functional CMake 3 build system. Includes build configuration, regression testing (CTest), and installation and packaging (CPack) support. Generic CMake system scripts and tools are included in the form of a subtree of the cmake repository from the shacl project located in the top-level **`shacl-cmake-subtree`** directory of the MCNP repository. MCNP-specific CMake system scripts and tools are included in the top-level **`cmake`** directory. See §1.4 for further details.
MCNP-50112	Materials	Added mixed-material treatment for structured LNK3DNT embedded mesh geometries. See §1.5 for further details.
MCNP-52880	File Formats	Added an optional HDF5 format for `PTRAC` output, enabling both threading and MPI parallelism. See §1.6 for further details.
MCNP-52943	Physics	Added Doppler broadening resonance correction (DBRC) treatment to enhance neutron scattering physics. Off by default, the `DBRC` input card is available to turn on and control this physics treatment. See §1.7 for further details.
MCNP-52968	File Formats	Replaced Fortran unformatted binary runtape with an HDF5-formatted runtape. See §1.8 for further details.
MCNP-52987, MCNP-53029	UM File Formats	The unstructured mesh model input can now be input in an HDF5 format. Additionally, the unstructured mesh elemental edit output file can now be output in an HDF5-based XDMF format. See §1.9 for further details.

continued on next page...

Table 5 — continued from previous page

Tracking Number	Category	Description
MCNP-53048	Coding	Several things were done to reduce memory usage in `BURN` problems with a very large quantity of materials. First, the reaction rate cache was altered to be allocated to the right size as opposed to the initial guess. Second, the ***nostats*** option was added to `BURN` to disable statistics and reduce memory usage. Third, a new `DISABLE` card was added with a ***nuclide_activity_table*** option to further reduce memory usage in problems with large numbers of isotopes and cells. See §1.10 and §2.3 for further details.
MCNP-53092	File Formats	Added new output format, `xdmf`, to the `FMESH` card. See §1.11 for further details.
MCNP-53102	Tallies	Added four new special tally treatment options to the `FT` card to compute energy-angle scattering matrices (`SPM`), multigroup cross sections (`MGC`), fission neutron spectra (`FNS`), and Legendre coefficients for scattering reactions (`LCS`). See §1.12 for further details.
MCNP-53181	UM	Added optional, on-by-default, unstructured mesh quality metric assessment. See §1.13 for further details.
MCNP-53206	Data Physics	The `MT0` card was added to allow a user to assign an $S(\alpha, \beta)$ table to a specific ZAID, in case a user wishes to experiment with stochastic mixing of nuclear data at different temperatures. See §1.14 for further details.
MCNP-53269	Tallies	Replaced the `FMESH` tally backend. See §1.15 for further details.
MCNP-53270	Plotter	A new Qt-based plotter replaces the legacy X11 plotter included in separate technology preview executables. See §1.16 and §5.3 for further details.

Table 6: Code Enhancements

Tracking Number	Category	Description
MCNP-42400	Performance	Reduce memory requirements and increase speed for the `ACT` card `DG` = `lines` option. See §2.4 for further information.
MCNP-43304	Testing	The maximum command line length is now 4096 characters.
MCNP-48485	Plotting	The cell shading in the interactive plotter can now be enabled for cell-based importances. If the importances differ by more than a factor of ten, then the legend is scaled logarithmically.
MCNP-50672	Testing	Some minor code changes and template clean-up changes for OS and OMP/MPI -agnostic regression testing

continued on next page...

Table 6 — continued from previous page		
Tracking Number	**Category**	**Description**
MCNP-50780, MCNP-50870	Coding	Improvements toward Fortran 2008 compliance by removing entry statements in legacy code. During the clean-up of LAQGSM, a memory bug was resolved.
MCNP-52594	Utilities	A mesh-mapping capability is added to the **um_pre_op** utility to show correspondence between an Abaqus mesh input file and a legacy **eeout** file. The objective is to provide a straightforward workflow to analyze **eeout** results when processed into Abaqus.
MCNP-52655	Coding	Entries on the `RAND` card can now be entered with exponential notation. Previously, entering the input values for the `RAND` keyword options as real numbers was not permitted; the values were required to be ordinary integers. This requirement was relaxed to conform with standard MCNP input handling conventions. Input values on the `RAND` card can now be entered as real or integer values. If a real number is entered, it will be converted to the nearest integer, and any fractional part of the value will be lost. Users are strongly encouraged to use integers, not real numbers, for input on the `RAND` card.
MCNP-52816	Coding	Performance in simulations with a very large quantity of materials was improved by zeroing out the reaction rate cache only on parts that were needed for the problem. In MCNP-53048, the cache was resized to not take as much memory. See §2.3 for further details.
MCNP-52817	Build System	Added the Intel compiler flag `-fp-model = consistent` (Linux / macOS) and `/fp : consistent` (Windows) to the build system. The impact of this change includes: a) a change in many integral responses, and b) more consistent floating-point arithmetic within the code. The latter change means that the code is less sensitive to compiler versions, optimization levels, and operating system differences. See §3.1 for further details.
MCNP-52867	Coding	The **DATAPATH** character array length has been extended to 1024 characters. See §1.17.2 for further details.
MCNP-52885	Physics	Removed a hardcoded limitation of 100 isotopes in calculating multiple Coulomb scattering for charged-particle transport.
MCNP-52926, MCNP-53280	UM	Added Abaqus SC8 element-type processing. Like the Abaqus C3D* element type, they are grouped as 1st order hex elements.
MCNP-52927	Coding	Fix white space differences in MCNP output files between MPI and non-MPI runs.

continued on next page...

Table 6 — continued from previous page

Tracking Number	Category	Description
MCNP-52932	UM	Update MCNP output file writing to include correct part and instance numbers of the UM model where parts are separated into multiple pseudo-cells and/or each part is instanced more than once.
MCNP-52934	UM	The maximum number of pseudo-cells is written to the output for a UM calculation.
MCNP-52965	Tallies	Relaxed k-eigenvalue covariance estimator tolerance to lower the likelihood of numerical round-off issues. See Known Issue MCNP-53139 for more information.
MCNP-53005, MCNP-53055	Coding	Changing calls to the **xss** array from **NINT** to **INT** provides minor performance improvements with certain Intel compilers.
MCNP-53030	Coding	The MCNP code will only search for the default **XSDIR** file, **xsdir_mcnp6.3**, unless otherwise requested by the user through the execution line or message block. This fixes an issue with GCC 9.2.0 on MSYS2 and prevents the user from loading unexpected data. See §1.17.2 for further details.
MCNP-53035	Coding	Minor improvements to stability with certain compilers by placing !$OMP PARALLEL directives around initialization of THREADPRIVATE variables for the random number generator.
MCNP-53039	Burnup	Large `BURN` problem initialization was optimized by precomputing several arrays instead of generating them on-the-fly.
MCNP-53144	UM	Improved error checking in the UM when mixing element types within a part. If mixed elements within a part are not supported, an error message will end the calculation.
MCNP-53180	Output	The "source version" is written to standard output and the MCNP output file. This information should be communicated to the MCNP team during requests for help, bug reports, etc.
MCNP-53123	UM	The memory footprint for unstructured mesh calculations has been reduced.
MCNP-53126	Coding	Refactor code to remove Common/Equivalence.
MCNP-53143	UM	The speed of UM input processing has been greatly improved. See §2.1 for more information.
MCNP-53284	Data	Updated default **XSDIR** file to **xsdir_mcnp6.3**. See §1.17.2 for further details.
MCNP-53305	Tallies	The `PTRAC` tally filter value default is now set to write any history that contributes a non-zero score to the tally. Previously, the default would be to write when a score was 10× the current tally fluctuation chart bin average.
MCNP-53347	Output	Fix the termination message when an MCNP run ends due to a **stopinp** file.

continued on next page...

Table 6 — continued from previous page

Tracking Number	Category	Description
MCNP-53381	KCODE	The Shannon entropy mesh is sized with a slightly altered algorithm, causing slight changes in entropy calculations. There is no impact on the k-eigenvalue or any other tally results.
MCNP-53404	Tallies	The number of point detectors allowed is increased from 1,000 to 10,000.
MCNP-53428	Physics	Added preliminary support for ENDF/B-VIII.1 ACE files. Mixed mode $S(\alpha, \beta)$ and ACE LAW 61 photonuclear support added. See §1.17.3 for further details.
MCNP-53435, MCNP-53517	Physics	Upgraded to CGMF version 1.1.1, including fission physics improvements and the availability of additional spontaneous and neutron-induced fission systems. See §1.17.4 for further details.
MCNP-53458	LNK3DNT	Improved `matcell` material and cell checking for embedded LNK3DNT geometries.

Table 7: Bug Fixes

Tracking Number	Category	Description
MCNP-4509, MCNP-52897, MCNP-52955	Comments	Fix trivial typo in comments and variable names.
MCNP-7529, MCNP-52935	Input	Previously, duplicate items on cell cards resulted in a fatal error. The nested Like/But feature on cell cards also triggers this fatal error. This fatal error has been removed to allow for legal Like/But cell card inputs to process.
MCNP-16278	Coding	Fixed the intent of a subroutine argument in the magnetic field tracking routines. When using the GCC compilers, version 9.1.0 and higher, the code would get caught in an infinite loop.
MCNP-26747	Coding	Charged particle ACE files with missing **`DLWH`** blocks would yield invalid secondary recoil energy spectra and might cause memory bugs. Additionally, if light-ion recoil physics was enabled and the recoil was present in the secondary production block, duplicate particles would be made. Scatter and recoil are now performed consistently. For more detail, see §4.2.
MCNP-26929, MCNP-52855	Tallies	If multiple tally segments (defined on the `FS` card) are used with complex cells (defined with unions and/or parentheses), memory corruption can occur during cell volume calculation that can lead to a crash. This has been fixed by fixing the memory allocation calculation.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-28193	Coding	Removed unused **ERF** functions from the CGM/CGMF library that conflicted with the MINGW math library **ERF** function.
MCNP-28495	LNK3DNT	The **cell_properties.F90** is fixed so that the material properties from LNK3DNT are set correctly.
MCNP-28495	LNK3DNT	When only structured mesh type is embedded in a problem, MCNP code checks if all material numbers listed on `matcell` of embedded LNK3DNT card are also in a LNK3DNT file. However, this checking is not called if both structured and unstructured mesh types are embedded in a problem. The fix was done by adding this checking for the structured mesh type when both mesh types are embedded in a problem.
MCNP-28495	LNK3DNT	The two LNK3DNT continue run test problems (52 and 53) do not work using the gfortran compiler because of a separate LNK3DNT continue-run issue. Not all of the LNK3DNT geometry data is initialized properly in a continue run. This bug is fixed in the **lnk3dnt_xport** subroutine.
MCNP-28495	LNK3DNT	Fix uninitialized variables in **lnk3dnt_mod.f90**.
MCNP-28495	LNK3DNT	Fix a bug in setting densities for LNK3DNT.
MCNP-30138	Coding	Fix an issue in which a distribution of transformations on an `SDEF` or `SSR` card would be ignored if only one transformation is in the distribution.
MCNP-34418	UM	Check `volumer` and source elements for consistency. If the `volumer` value is associated with the mesh or cell that does not have source elements, then a fatal error is thrown.
MCNP-39377, MCNP-53415	Coding	Previously, runtapes with file names (or file paths) longer than 36 characters caused the mesh tally (`FMESH`) plotter to crash. This has been fixed.
MCNP-42675	UM LNK3DNT	A fatal error will be thrown if the background cell is not unique with respect to the non-void pseudo-cell entries in the `matcell` keyword on an `EMBED` input card. A deprecation message will be thrown if the `background` cell is the same as the void cell on the `matcell` keyword on an `EMBED` input card.
MCNP-48481	Plotter	The default "color by" property is now always `MAT` unless the user selects a different parameter. Previously, it was `MAT` on the first coloring and then resets to `CEL` on subsequent colorings.
MCNP-49860	UM	Lost charged particles when crossing to the UM background cell. Some variables were not reset, causing charged particles to not track correctly to the next surface.
MCNP-49936	UM	A bug in **um_post_op** utility. The **um_post_op** routine that reads the **eeout** file header information was not correct. A few lines of codes were fixed. No testing is currently performed for UM Utilities.

continued on next page...

Table 7 — continued from previous page		
Tracking Number	**Category**	**Description**
MCNP-50041	Coding	Bounds array error with **colinp**. This bug occurs when the total was asked for in a column formatted tally option in an MCNP input file. This caused a bounds array error.
MCNP-50043, MCNP-50044	Coding	Several uninitialized variables are initialized to mitigate potentially different treatment by various compilers and optimization levels.
MCNP-50050	Coding	Memory bounds failure for long data file names. This bug caused by a material with a 6-digit ZAID prefix and the fix was done in **ixsdir.F90** file.
MCNP-50078	Coding	Fix for **DATAPATH** warning being printed by all MPI process. If **DATAPATH** is not set by the environment, a warning message is printed. This is a fix such that only the manager rank prints the message. Whatever value the manager process detects as the **DATAPATH**, it is broadcast to the worker processes when the cross sections are sent.
MCNP-50281	Coding	`KOPTS` + time/energy splitting and rouletting. In a `KCODE` calculation with either time or energy splitting/rouletting (`ESPLT` or `TSPLT`), the inclusion and exclusion of the `KOPTS` `KINETICS` = `yes` card yields significantly different k-effective results without a warning or error message. The coding in **ergimp.F90** modified to resemble the coding in **wtwndo.F90** where the k-adjoint banking of particles is handled differently when particle splitting occurs. Prior to this bug, either time or energy splitting with the k-adjoint weighting resulted in a changed k-effective result. Test problem 143 was added to the regression test suite to demonstrate this bug fix. No other test problems were changed.
MCNP-50369	Coding	The value of 'histories/hr' is incorrect if no particle is run for continue runs. The file **sumary.F90** is changed to skip the reporting of 'histories/hr' if no particle was run.
MCNP-50409	Coding	When running with threads, the event log prints out the wrong history number for a lost particle. The first line of an event log includes the particle number of the history. However, **eventp.F90** uses the variable **nps** instead of **npstc** to identify the history number. This causes the wrong history number to be written for the event log when a particle is lost when using threads. To fix this, the variable **nps** in **eventp.F90** was replaced with **npstc**.
MCNP-50839	Tracking	Fix a bug in setting the material number in the single event electron tracking routine.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-51394	LNK3DNT	For LNK3DNT, the incorrect **igeom** was being specified for spherical 1D meshes. The PARTISN manual indicates that it should be 3, whereas the MCNP code is using 4. Fixed to use the proper value of 3 in MCNP6.3.0.
MCNP-52062	LNK3DNT	Two elements of the **lnk3dnt** derived type, **dx** and **constant_spacing** were neither initialized nor read/written to the **runtpe** file. The subroutine **lnk3dnt_file_initialize** was modified to initialize these two variables. In addition, read/write statements were added to the subroutines **lnk3dnt_fdac_read** and **lnk3dnt_fdac_write**.
MCNP-52572	Coding	Using the `MAT` option on a `KPERT` card gives a fatal error unless the `KPERT` card appears after all of the material cards in an input file.
MCNP-52583, MCNP-52853	Coding	Column input that includes the jump (`j`) command to provide a default value previously failed. This has been fixed.
MCNP-52734	UM	Poor code performance with large UM edits. This fix makes the code faster for large UM models that have edits with multiple energy and/or time bins.
MCNP-52818, MCNP-53371	Tallies	In previous versions of the code, when a non-neutron/photon/electron particle was born or slowed down below the energy cutoff, the remaining kinetic energy was not added to any energy deposition tallies. This change causes all energy to deposit at the point in which the particle goes below cutoff. For energy deposition in very fine geometries, care should be taken to ensure that the resulting tally is insensitive to the cutoff energy. If a large number of particles go below cutoff in a region, but would have transported into another region with a lower cutoff, this will overestimate energy in the first region and underestimate it in the second.
MCNP-52829	Coding	Add missing common block in **rm48ut** routine and fix uninitialized variable in **laqmod31** subroutine
MCNP-52848	Coding	When the MCNP code is compiled with gfortran 7.3 using CMake, segmentation faults occur for these test problems: CGM (13 14) and CGMF (01 02 03 05 06 07 11 12 13 22 23). This bug reported that the combination of CMake and GCC 7.3.0 caused an error. However, it was in fact an out-of-bounds indexing into an array within CGMF that GCC 7.3.0 segmentation faults on compared to previous version of GCC. No templates change as a result of this fix.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-52858	UM	The **um_post_op** utility has an option to convert legacy **eeout** files to VTK files. Previously, the VTK files were written incorrectly because of an index-ordering difference between **eeout** and VTK. This deficiency has been fixed. However, because of the deprecation of the legacy **eeout** file format, alternative conversion utilities are available [31, 32].
MCNP-52859	UM	Fix in computing the bounding box of an element in an unstructured mesh model.
MCNP-52865	Coding	A bug for gcc segfault with `FS` card and complicated cell. This bug was fundamentally with the size of **lgc** array. With the `FS` tally segmenting card, the **lgc** array is used to hold the logical expression of inside a cell, inside a particular segment, and not inside all the other segments. This is used to calculate volume of segment. The **lgc** is now conservatively larger to handle this cases, with two extra spaces for parenthesis that are needed to prevent the crash.
MCNP-52866	Coding	If a material contains more than one isotope, the electron scattering angle distribution was calculated incorrectly. The effect of this bug was substantially apparent with materials with elements that have low atomic fractions. This bug has been present since MCNP5. The issue was caused by the Legendre polynomial expansion in the low energy regime being allocated incorrectly among isotopes. Therefore, causing the electron scattering angle distribution to be negative. The fix forces the Legendre polynomial expansion to be calculated to the recommended allocation of 240 entries in the low energy regime for materials with more than one isotope. With this fix, one notes that the 90$^{\text{th}}$ option on the `DBCN` card is only valid for electrons in the high-energy ($E_e > 0.256$ MeV). This bugfix can have a significant effect on electron results.
MCNP-52879	Coding	An out-of-bounds memory access was possible within the CGM code base, which has been fixed in the CGMF code base included with MCNP6.3.0.
MCNP-52892	Tallies	Fix for bug with an `F5` point-detector tally that produces a NaN if the source particle direction (`VEC`) is 90° from detector direction.
MCNP-52893	Sources Tallies	Fix for bug with an `F5` point-detector tally with `DIR` dependent on another variable. If the depended-upon variable was specified with the `SI A` option, the tally results were previously incorrect.

continued on next page...

Table 7 — continued from previous page		
Tracking Number	**Category**	**Description**
MCNP-52894	LNK3DNT	The **partinp** file generated using the `DAWWG` card incorrectly formatted the material-block ZAID entries. The additional white space within the quoted ZAID entries is now removed.
MCNP-52899	Sources	For dependent histogram and linear distributions on a `DS` card, an energy cutoff would lead to incorrect truncation and weight normalization of the upper bin of any distribution that was dependent on energy. The sampling and normalization are now correct.
MCNP-52933, MCNP-52938	UM	The number of instances and the part data for the UM geometry written to the output file were incorrect.
MCNP-52939	KSEN	Previously, if multi-group fission-chi distributions were processed by the `KSEN` card, the reaction name would be mangled when printed. This mangling has been fixed. In addition, a default name of "other reaction" has been added to avoid this behavior for other unspecified reaction MT identifiers.
MCNP-52944	Coding	Fix for an error in conversion of an arithmetic-if statement removed in the LCS model physics routines during Fortran 2003 compliance efforts for the MCNP6.2.0 code release.
MCNP-52949	UM	Fix for UM memory leak when pointers are deleted before arrays are deallocated.
MCNP-52954	UM	Fix segmentation fault for several UM test problems when running with both MPI and OpenMP parallelism.
MCNP-52960	PTRAC	Several issues in the legacy `PTRAC` capability are fixed in the new HDF5 `PTRAC` capability: • Photons were using a multiple of the node number for the ZA number. • The particle banking routine was resulting in an access of element the incorrect `PTRAC` array element. This is fixed by creating specific bank events for surface sources (`SSW`/`SSR`) and spontaneous fissions (`SDEF` `PAR = sf`). • A potential out-of-bounds access of `PTRAC` bank events is fixed. • Duplicate collisions were being written for elastic scatters during charged particle tracking.
MCNP-52964	Coding	A **runtpe** file written by the MPI parallel version of **mcnp6** (**mcnp6.mpi**) cannot be read by the regular version of **mcnp6** and vice versa.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-52991	Coding	Delayed neutron sampling was handled incorrectly when using the FREYA and CGMF fission options. The incorrect algorithm treated the delayed neutron sampling when using the FREYA and CGMF prompt-only models in the same manner as the LLNL Fission Library or standard built-in `FMULT` neutron multiplicity sampling algorithms. This resulted in $\approx \bar{\nu}_{\text{delayed}}$ too few prompt neutrons emitted per fission event.
MCNP-52951	Coding	Fix memory leak in tally array holding attenuator pointers setup during tally input parsing.
MCNP-53001	Coding	Fix for memory bugs in regression tests including k-adjoint progenitor list memory leak, k-eigenvalue sensitivity (`KSEN`) instance array leak, and unstructured mesh **`mcnpum`** file closure.
MCNP-53027	Coding	Fix for a misformatted print statement when there are too many detector tallies in a problem. Now prints the number "1000" rather than "***".
MCNP-53041	Burnup	The `OMIT` entry on the `BURN` card now properly checks that the listed number of entries matches the true number of entries.
MCNP-53054	UM	Fixes for unstructured mesh bugs in MCNP6.2.1. The fixed issues presented themselves as a collision taking place in a void and photon transport with all-zero photoatomic cross sections.
MCNP-53057	Coding	Simulations using the `KPERT` card and k-eigenvalue adjoint-weighted capabilities can be restarted with the "continue" option.
MCNP-53075	Data Physics	The $S(\alpha, \beta)$ physics in earlier versions did not account for fission, which resulted in invalid results when applied to nuclides with non-zero fission cross sections in the thermal region. This has been fixed.
MCNP-53086	Sources	The `DS Q` option is no longer allowed when the independent variable is cell, surface, or transform. Internally, the code did not preserve the order of these variables, which could lead to unexpected results.
MCNP-53119	Tallies	There was an indexing error in the combined total energy and total time bins for the `TYPE` = `source` `FMESH` tally which would cause wrong answers. This would occur only if both energy and time bins were specified on the `FMESH` card.
MCNP-53122	Tallies	If a surface tally exists with a particle type that is not listed on the mode card, then tally results were printed incorrectly for the surface tallies, for all particle types in some cases. The results are now printed correctly.

continued on next page...

Tracking Number	Category	Description
MCNP-53135	Output	A continue run now checks the requested dump to ensure that the simulation did not have an unexpected error that potentially left the code in an invalid state with potentially bad results. An error message is printed to the screen to indicate that a prior, valid dump should be used. See the continue-run portion of the manual for more details.
MCNP-53138	Coding	Energy-dependent perturbations of fixed-source tallies using the `PERT` card incorrectly bins secondary (banked) particle contributions to perturbation tallies [33]. The incident energy of the particle coming out of the secondary-particle bank was not being banked properly with the secondary-particle information, leading to an incorrect energy used in determining the proper perturbation tally bin. The incident energy of the particle prior to banking secondary particles is now a banked quantity.
MCNP-53187	Output	If an `FMESH` tally has energy bins, and the uppermost bin upper-energy boundary has a value equal to the maximum cutoff energy for the particle, and that particle's default lower-energy cutoff is 0 MeV, then the energy-bin-value column is not printed. Now, the energy-bin-value column is printed as long as more than one bin is requested. This bug only applies to neutrons and a few other high-energy particles.
MCNP-53216	Coding	Corrected misidentified card if more than one debug option is specified on the `EMBED` card. Previously, the error message incorrectly referred to the `DAWWG` card.
MCNP-53226	Sources	Previously, weight-cutoff (weight-rouletting) levels were not applied after a user-defined source is applied using the **`source.F90`** subroutine, so it was possible for particles to cascade down to effectively zero weight, which would cause undesirable behavior. Now, the same weight-cutoff approach is taken as with surface-sources to ensure all particles are treated appropriately with respect to weight cutoff and implicit capture following a user-defined source particle is generated.
MCNP-53232, MCNP-53233	UM	Fixed the point-of-intersection tracking algorithm to allow a particle that is exiting a cell to compute a backup distance. This distance is then used to search for all surface bounding boxes of the exiting surface.
MCNP-53237	UM	Prevent checking element faces with intersection distances that are in close proximity ($\approx 10^{-10}$ cm) for overlaps. This prevents an infinite loop of switching between close-together pseudo-cells that are misidentified as overlapping.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-53238	UM	Rare inappropriate contributions to a point detector as a result of prior-history behavior are possible through regions where such contributions should not occur (e.g., zero-importance void regions). This is because of stale values previously not being updated properly when computing the contribution to the point detector. These values are now updated and the appropriate, current, values are used to remove inter-history dependence.
MCNP-53246	Tallies	If the `KCLEAR` keyword was on an `FMESH` card, it would need to be at the end of the `FMESH` card or the code would issue a fatal error. The `KCLEAR` keyword can now be placed anywhere in the `FMESH` card parameter list.
MCNP-53247	UM	The response function defined on UM elemental edits using `EMBDE`/`EMBDF` cards would be correctly read from legacy **`eeout`** files but then incorrectly processed during continue calculations. This has been fixed.
MCNP-53260	UM	Since the UM library cannot handle a mixed-material cell and the elements in statistic elsets are used to build the pseudo-cells, a fatal error is thrown if the number of material elsets is greater than the number of statistic elsets in each part.
MCNP-53293, MCNP-53310, MCNP-53319	UM MPI	Fixed UM MPI issues related to broadcasting input from manager to worker processes, MPI communication termination after input processing, and worker process memory allocation.
MCNP-53295	Plotter	When attempting to plot with `ijk` labels, the plotter would produce a segmentation fault under certain circumstances. This fix corrects the `ijk` plotter labeling.
MCNP-53311	UM	Correct error in the **`um_pre_op`** utility where the element checker for 2nd-order tetrahedral elements incorrectly used the 2nd-order pentahedral element master-element and Gauss points to compute the Jacobian matrix determinant.
MCNP-53315	Tallies	An issue preventing creating more than 9 of some `FT` cards (`FT` `SCX` in particular) has been fixed.
MCNP-53326	UM	A fatal error will be thrown if all elements in each part are not assigned to material and statistic elsets.
MCNP-53327	UM	A fix for a segmentation fault that is thrown during UM input processing after a fatal error has already occurred in general input processing.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-53339, MCNP-53438	UM	The elemental edit results written into an **eeout** is wrong for some problems where type-6 `EMBEE` cards have multiple particles and the same particles appear on multiple `EMBEE` cards. These edit output results were written into the **eeout** file using both `EMBEE` numbers and particles listed on the `EMBEE` cards and the value of `EMBEE` and particle numbers are based how they appear in an MCNP input file. This code implementation produced wrong edit results or not writing some edit results when duplicated particle number values associated with the `EMBEE` cards that appear with particular ordering. These bugs are fixed. Only `EMBEE` numbers that are unique are now used to order edit output results written into an **eeout** file. The order of edits in a continue-run are also fixed.
MCNP-53349	Output	Continue runs from a simulation with a `TMESH`, and all other features compatible with a continue run, generated with an MPI executable can now be read by a non-MPI-enabled executable; the same is true for different numbers of ranks and tasks.
MCNP-53365	Coding	Fixed output format statements when **nps** is greater than or equal to 10^{12}.
MCNP-53366, MCNP-53370	Tracking	In the magnetic field tracking routines, particles can end up with a cell and surface number combination that breaks the particle tracking algorithm. This can cause 'bad trouble' errors or segmentation faults. Additional checks were added to the magnetic field tracking routines. The lost particle flag will be set if the new checks fail.
MCNP-53367	UM	The element face centroids of 2nd order pentahedra are computed incorrectly.
MCNP-53369	UM	The number of cell source elements was incorrect for the problem where the elements listed in source elsets are non-contiguous. See Problem **inp1057** as an example. **CellInfo(7,1)** of this problem was 12 instead of 96. **CellInfo** is internal to the code and only written into the **mcnpum** file and HDF5 **eeout** file.
MCNP-53373	Coding	In the delayed particle subroutines, several arrays were used before they were allocated. The arrays are now correctly allocated before usage and memory errors were removed.
MCNP-53397	Tallies	Previously, detectors could score negative flux if a source with `DIR = 1` with `VEC` towards the detector was used. This has been fixed by adjusting an if-statement inequality.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-53411	Variance Reduction	The ***mwhere*** parameter on the `WWP` card allows a user to select where to process weight windows during the transport process. However, the logic used internal to the code to apply the user's preference is incompatible with unstructured mesh geometries. This incompatibility has been fixed so the code will respect a user's choice to apply weight-window processing as a result of surface crossings (or for UM, entry and exit from a UM pseudocell), collisions, or both events. In addition, the surface-crossing-only treatment applied via ***mwhere*** would disable mean-free-path-of-travel weight-window processing, which is controlled using the ***nmfp*** parameter on the `WWP` card. This inadvertent disablement has been removed so now the ***nmfp*** parameter solely acts to control such processing. See §4.1 for additional discussion.
MCNP-53416	Comments	Fix typo in tally and cross section plotter for **mt** command.
MCNP-53418	Coding	Using the `CUT` card with multigroup adjoint now works correctly. Previously, using the `CUT` card in a problem with `MGOPT` A would terminate the transport with a bad trouble in **tallyd**. Neutrons or photons terminated by energy or weight cutoffs would be passed into the **tallyd** subroutine. This has been fixed along with a related oversight that prevented collided neutrons from scoring to next-event-estimator tallies in adjoint calculations.
MCNP-53426	Comments	Fix typos in comments distinguishing smooth cross sections and probability-table factors. Correct identification of absorption smooth cross section in a comment.
MCNP-53427	UM	An MPI MCNP UM calculation was not previously terminated after a fatal error was thrown in the input-setup step, which could lead to unproductive long-running jobs.
MCNP-53430	Utilities	When **mcnp_pstudy.pl** attempts to split lines longer than 75 characters, it doesn't consider the implications of splitting a title card that exceeds the 75 character limit. The fix includes logic to detect a message block, the subsequent blank line delimiter, and finally the title card. For the optional message block and required title card, the lines are kept as-is with no line splitting.
MCNP-53434	UM	Continue runs for multi-models using legacy **eeout** files would crash since the setup in the continue run is inconsistent with the allocation in an initial run.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-53440	TMESH Tallies	When using `TMESH` to tally charged particles in magnetic fields, tallies were incorrectly performed twice: once with the curved trajectory and once with the straight trajectory that would have occurred had no magnetic field been present. Now, only the correct curved trajectory is tallied. Further, the MCNP code will now warn users to not use `TMESH` energy deposition tallies in magnetic field regions, as this codepath is not yet tested.
MCNP-53445	LNK3DNT	Fix a bug in checking a void material. If there is a void material in a LNK3DNT file, this information must be declared in the `matcell` option with zero in the 1st entry. There was a bug in the code where a fatal error was thrown when the input setup for a void material had correct syntax.
MCNP-53446	Utilities	When **`mcnp_pstudy.pl`** tries to split an inline comment ($) that is longer than 75 characters, the code gets into an infinite loop. The fix includes keeping inline comments exactly as-is and printing them, regardless if they exceed the 75 character limit.
MCNP-53449	TMESH Tallies	Electrons in void did not contribute to `TMESH` tallies. This has been fixed.
MCNP-53457	Comments	Remove code-contributor attribution comment in favor of source-control-based attribution.
MCNP-53460	Coding	`SQ` surfaces with the same parameters but different X Y Z translations were considered the same surface. Translated, but otherwise identical `SQ` surfaces are now correctly parsed as unique surfaces.
MCNP-53470	Physics	In the MCNP code, there is a nuclear data cache for each nuclear data table that is used to improve the performance of the code. This cache is invalidated whenever the energy of the particle changes or the temperature of the material changes. In materials with $S(\alpha,\beta)$, the $S(\alpha,\beta)$ effects were added to the affected nuclide's total cross section. If two materials had the same nuclide, one modified with $S(\alpha,\beta)$ and one that was not (or had a lower $S(\alpha,\beta)$ energy cutoff), the $S(\alpha,\beta)$ contribution to total was not removed when the particle moved from the first material to the second. This is now fixed by caching a flag for each material which indicates if $S(\alpha,\beta)$ was previously applied to the total cross section. As this data is often invalidated and recomputed after the first collision in the new material due to a change in energy, the impact on simulations is typically small. This bug has been present since at least Version 4C.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-53473	Coding	An array bounds error that occurred when plotting tallies by cosine was fixed.
MCNP-53485	UM LNK3DNT	UM `EMBED` cards are required for a restart run (i.e., continue run). If a number of UM `EMBED` cards in an initial run and a restart run are mismatched, then a fatal error is thrown. LNK3DNT `EMBED` cards are not needed for a restart run. If LNK3DNT embed cards are present, then these cards are ignored.
MCNP-53474, MCNP-53500	Plotter	Negative values on an `FCL` card would show up as "0" when the Label 2 quantity in the geometry plotter was set to `FCL`. The initialization of the displayed variable was changed from 0 to -HUGE_FLOAT.
MCNP-53501	UM	Previously, the MCNP UM **eeout** and **gmv** files would contain the material numbers specified in the UM mesh input (i.e., Abaqus) file rather than the material numbers specified in the MCNP input file (which define the material IDs used during particle transport). This can lead to incorrect output interpretation if materials are defined inconsistently between the two input files. Now, the material numbers in the MCNP UM **eeout** file (either legacy and/or HDF5-formatted) match the values defined in the MCNP input file. This is also true when the MCNP code is run with just the i option to generate an HDF5-formatted UM input file. Similarly, the material "names" in the **gmv** file match the MCNP input file material numbers where the element-by-element values are the one-based indices into the material name list. Finally, a warning is now issued when the code identifies inconsistent material numbers used in the mesh input and MCNP input files.
MCNP-53504	Coding	The information following the column input designator (#) was restricted to 20 characters. The manual states the column input may be up to 128 characters. This discrepancy was fixed and now the line following the # may be 128 characters.
MCNP-53527	Coding	When using the `ZLEV` command with 3 (or more) numerical arguments, the code switches to display `FMESH` results at the value of the provided arguments with no gradient between the levels. This switch in color mode used to be irreversible. The code now reverts to coloring a gradient when the user enters a "`ZLEV log`" or "`ZLEV lin`" command.

continued on next page...

Table 7 — continued from previous page

Tracking Number	Category	Description
MCNP-53536	Coding	Previously, no guard against invalid input to the sample parameter on the `ACT` card existed. In turn, this would allow a user to specify an invalid string that the code would ignore with no message. Now, an error occurs when invalid input is given for this input.
MCNP-53539	Physics	Occasionally, if a proton or light-ion underwent a capture reaction, the particle would continue to be transported an additional step before it was finally terminated. This would only occur when the particle is using cross section data for transport. This has been fixed.
MCNP-53543	Coding	For neutrons, the built-in ICRP-60 [34] equivalent dose conversion function (`IC` = `99` on the `DF` card) was using a look up table to find the radiation weighting factors but not interpolating between the energy bins. The result was that the radiation weighting factors were applied as a step function. This step function was replaced with the continuous energy dose function for neutrons included in ICRP-60.
MCNP-53545	Utilities	Fixed memory issues in the **makxsf** utility by increasing the static allocation parameter for processing larger $S(\alpha, \beta)$ data files and working around a large stack allocation issue.

Table 8: Known Issues

Tracking Number	Category	Description
MCNP-24152	Data	ZAIDs with extensions with more than two digits are incompatible with the code. All known currently released data is compatible, but it is expected for this to change in the future.
MCNP-25475	Coding	When electrons are in a cell that is not at its lowest level of a problem with multiple universes, they become lost. Typically, these are knock-on electrons and they become lost once the particle is removed from the bank.
MCNP-26745	Coding	In rare cases, INCL will try to create an unphysical fission event. The MCNP code will print a message when this occurs and ignore the event.
MCNP-27742	Burnup	The burnup capability is incompatible with adjoint-weighted k-eigenvalue perturbations and sensitivities.

continued on next page...

Table 8 — continued from previous page

Tracking Number	Category	Description
MCNP-27942	Physics	Using `DXTRAN` spheres in photon-transport problems as a form of variance reduction with pulse-height tallies is incompatible with extended photon data such as that given in the eprdata12 library available via ID ".12p". If these features are combined, a fatal error results. No timeline is available for when this issue will be addressed.
MCNP-28168	Tallies	When an `FU` card is used with a radiography tally, the number of user bins is printed to the **`mctal`** file, but the values of those bins are not.
MCNP-28325	Physics	Proton and light-ion tables generally terminate at a minimum energy well above the lowest value allowed for the energy cutoffs. The mix-and-match logic deals well with the transition from models to tables at the table upper energy limits, but ignores the issue at the lower limits, leaving the code to rely on some inadequate approximation, such as extrapolation to low energies.
MCNP-32429	Tallies	When surface tallies are used in a lattice geometry, the tally scores in the wrong surface bin for a given index when tallying the surfaces between elements.
MCNP-32570	Geometry	Invalid geometry errors can occur with multiple `TR` surfaces. The user is urged to pay attention to warning messages to the console as this error doesn't necessarily show up as dotted lines on the geometry plotter.
MCNP-39260	Tallies	Incorrectly marked as fixed in the MCNP6.2.0 Release Notes [1]. If the `PHYS:E` keywords `BNUM` is >1, or `ENUM` <1, electron heating tallies results are incorrect.
MCNP-40520	Tallies	When a translation is applied to an `FMESH` tally, the **`meshtal`** file does not reflect the translation, though the tally plots correctly from the runtape file.
MCNP-40527	Coding	Some numeric output format statements are inadequate to properly display large numbers, such as those that may be encountered in calculations with extensive geometry, lots of histories, etc. No timeline is available for when this issue will be definitively addressed, but those who encounter it are encouraged to contact mcnp_help@lanl.gov and provide an example calculation that demonstrates the issue, so development effort can be best directed toward particular areas of interest.
MCNP-40875	KCODE	When `KPERT` has `rxn` = `-7` specified, doubling the nubar will result in `KPERT` reporting a negative reactivity.

continued on next page...

Table 8 — continued from previous page

Tracking Number	Category	Description
MCNP-40885	Physics	In some cases, such as test case **photons/inp011**, charged particles born above the cutoff energy level can lead to an imbalance in energy conservation, as shown in the particle creation and loss table.
MCNP-41000, MCNP-53467	Build System	The MCNP code cannot be built without the **-r8** or similar compiler flag. This is automatically added for GNU and Intel compilers. It may need to be added manually if one wishes to test with other compilers.
MCNP-42958	Physics	When transporting charged particles in magnetic fields, the charged particle's position is updated before calculating the probability of knock-on. Therefore, it is impossible to consistently create delta-rays along the charged-particle's trajectory. This should not be a possible bug for a user to experience without modifying the source code.
MCNP-44063	Coding	With an `SDEF` card used for an initial `KCODE` source distribution guess, using the `CEL` parameter with a top level universe will cause the code to not dive into the sub-universes. This can lead to a very small initial keff guess by the code followed by the code expiring due to the new source overrunning the old source.
MCNP-44344	Geometry	If a plane surface bounds a region that is filled with a transformed universe that contains the same surface as a component of that transformed region, particles which pass through the untransformed surface will miss the transformed surface.
MCNP-44535	Plotting	If the `DAWWG` and `MESH` cards are present, there is no way to plot the mesh voxel boundaries defined on the `MESH` card without also including the `WWG` card. No timeline is available for when this issue will be addressed.
MCNP-49242	UM	If an unstructured mesh problem has a `EMBEE` card with `mtype` = `isotopic`, the MCNP code will crash in a continue run.
MCNP-52888	TMESH Tallies	Energy deposition `TMESH` tallies in magnetic fields are not yet tested. A warning has been added to not overlap `TMESH` energy deposition tallies and magnetic fields.
MCNP-52907, MCNP-53229	UM	Only the "`EXIT`" overlap model has been widely tested for UM models. The other models may not work at all depending on the application. The user should ensure the unstructured mesh model is as overlap-free as possible and not rely on the "`ENTRY`" or "`AVERAGE`" overlap methods to produce correct results.
MCNP-52972	Coding	An unsaved warning message occurrence counter can cause an unreasonable number of duplicate "Bad JSU in ANGL" warnings to be printed to the terminal and output file.

continued on next page...

Table 8 — continued from previous page

Tracking Number	Category	Description
MCNP-53124	SDEF	For embedded source distributions, the -21 option cannot be used on the sp card option or undefined behavior will occur, producing invalid results. The code does not warn if this behavior occurs.
MCNP-53139	Tallies	The algorithm used for most tallies to compute the variance is less numerically stable than optimal. Further, the algorithm used to combine multiple estimators, as used in the k-eigenvalue combined estimator and the prompt removal lifetime, is also somewhat unstable. This can have two effects: 1. Serial runs may have different standard deviations than parallel runs due to a changed order of operations. This is more likely with combined estimators. 2. In rare circumstances (typically with neutrons per cycle < 1000), the combined estimator may diverge significantly from the original estimators. The latter case has only been observed in one test problem with a very small number of particles. See Enhancement MCNP-52965.
MCNP-53156	Tallies	If two `FMESH` cards share a number, but have different particle identifiers, the code will not throw a fatal error. Both `FMESH` tallies will be written to the ASCII **`meshtal`** file.
MCNP-53209, MCNP-53468	UM	A warning message is issued when mixing void and non-void pseudo-cells in an unstructured mesh embedded model.
MCNP-53230	Utilities	The **`fit_otf`** utility Doppler broadens to the minimum of the first threshold reaction or the start of the unresolved resonance range. This is inconsistent with the approach NJOY used for ENDF/B-VIII.0, in which broadening stops at the unresolved resonance range.
MCNP-53236	Coding Tallies	In rare cases with complex geometry, use of `F5` tallies can cause an overflow in the particle stack, which terminates the code with a message of “bad trouble in subroutine savpar of mcrun”. If the user encounters this issue, please consider sending the input that reproduces it to mcnp_help@lanl.gov.
MCNP-53240	UM	When a user attempts to mesh an instance with conflicting default meshing schemes in adjacent regions, Abaqus allows the user to use mesh tie constraints across the shared face to resolve the conflict. This produces a mesh with neighboring elements in an instance sharing no nodes and leads to bad tracking.

continued on next page...

Table 8 — continued from previous page

Tracking Number	Category	Description
MCNP-53241	UM	When tracking in an unstructured mesh, if there is a region of 0 importance between two parts, particles can sometimes track from part A to part B with no regard for the interim region. This occurs when a particle is within the machine precision distance from the exiting face of part A.
MCNP-53261	Coding	In rare cases, inadequate weight rouletting during "`MODE` p e" problems with photon forced collisions can lead to particles with vanishingly small weights, which will terminate the code with a bad trouble error.
MCNP-53303	Coding	To remove a compiler-identified array reference out-of-bounds warning, a block of code in LAQGSM was removed and replaced with a bad trouble message indicating a "logic error in storelaq". This block of code appears to be unreachable through all possible code paths. However, if this message appears please contact the development team at mcnp_help@lanl.gov for guidance.
MCNP-53338	Plotter	For very large mesh tallies, the plotter can hang when trying to iterate to find a point to plot within the mesh. A workaround to break the hanging iteration is used to break the loop after 100,000 cycles. A message stating "Warning: Loop2 in fmesh_plot_mod.F90 failed to converge, skipping:" is issued when the iteration hangs for too long.
MCNP-53390	Coding	For neutrons and photons, if any tracking places the particle within 10^{-12} cm of a surface, it is moved through anyways to improve the reliability of the code. When weight windows are enabled, the distance moved can be altered based on the mean free path. If this occurs, the next surface ID is cleared. If these two events occur simultaneously (distance altered by weight window to within 10^-12 cm of a surface), the code can lose track of what surface to cross and lose the particle.
MCNP-53436	UM	Forced collision variance reduction is incompatible within the UM. A fatal error is issued when attempting to use the `FCL` and the UM together.
MCNP-53448	Coding	Certain MCPLOT module "help" command text is out of date with the MCNP User Manual. Users should refer to the Manual for the correct syntax of the MCPLOT and PLOT commands.

continued on next page...

Tracking Number	Category	Description
MCNP-53456	UM	Additional warning and fatal error messages are issued where there are known deficiencies in the UM capabilities alongside other MCNP code capabilities. A few additional messages are now issued: • Checking for `volumer` option and its use for UM volume sources. • Warning about verification/validation of multiple structured and/or unstructured mesh geometries embedded in same model. • Warning about use of the noact option on `LCA` card within UM. • Fatal error when using charged-particle transport within UM geometry.
MCNP-53475	Coding	In rare cases, a user may define a region to be two cells at the same time with a `TRCL` card. This can deliver silent wrong answers. The user is urged to confirm geometry in the plotter. If the geometry looks wrong, it probably is.
MCNP-53495	UM	Using `LCA` *`7j -2`* or `LCA` *`7j -1`* with UM provided inconsistent results when compared to CSG.
MCNP-53502	Tallies	Lost particles may slightly alter tally results. The tally contributions of the lost particle will be added to the next non-lost particle. The weight of the lost particle will not be used for normalization.
MCNP-53507	Coding Geometry	In certain complex lattice geometries, particles may rarely get lost and terminate the code with a segmentation fault, thus failing to write a lost particle log to the output file.
MCNP-53510	UM	Add a warning message if weight windows are applied for unstructured mesh geometries since this feature is not fully tested.
MCNP-53511	UM	Add a warning message for electron and/or charged particle transport with unstructured mesh geometries since this feature is not fully tested.
MCNP-53547	Plotter	Viewing an `FMESH` tally in the geometry plotter and then setting the viewport to square with the “VIEWPORT SQUARE” command from the lower-left command box will not transfer control to the terminal window and effectively locks the plotter.
MCNP-53550	LNK3DNT	Currently, LNK3DNT embedded geometry information print table has a placeholder ID of “print table XX” and should be assigned a table number.

continued on next page...

Table 8 — continued from previous page

Tracking Number	Category	Description
MCNP-53552	Coding	The vertical input format (#) cannot accommodate (x, y, z) triplets for a distribution on POS on a single line.
MCNP-53553	Qt Plotter	Viewing an `FMESH` in the Qt plotter lists the wrong value in the "lines" dialog box. "FMESH+CSG" should be listed when "CSG Cell Boundary" is listed instead.
MCNP-53557	Coding	The input processor will not identify illegal data lines that start with Arabic numerals such as material cards compositions that are continued to a new line but insufficiently indented.
MCNP-53559	Build System	The MCNP build system cannot be statically linked to HDF5 1.10.9 or 1.12.2. This issue has been reported upstream. You can enable dynamic linkages by setting the CMake variable `HDF5_USE_STATIC_LIBRARIES` = `OFF` at configure.
MCNP-53566	Coding	In the most common `KSEN` calculations, the max number of progenitors is typically set to be 1.5× the size of the fission bank, and this is sufficient for most typical problems. However, when excessive splitting occurs in the simulation (which is unusual in most k-eigenvalue calculations), the max number of progenitors can greatly exceed this assumed maximum size. If the actual number of progenitors exceeds the maximum set, it will result in a segmentation fault. A workaround is to either disable the variance reduction causing excessive splitting or to increase the msrk value on the `KCODE` card which increases the memory allocated to the progenitor array.
MCNP-53567	Coding	With certain input files, the “additional error messages on file outp” message appears too many times.
MCNP-53569	Qt Plotter	The shift-key modifier for keyboard navigation of the Qt plotter view does not work on Windows builds.
MCNP-53571	Input Processing	The `FM` card processing does not check if the C/T entries are at the end of a bin/line in the input file.
MCNP-53575	ISC	The MCNP horizontal-output continuum-distribution writer incorrectly uses the `l` MCNP distribution identifier rather than the `h` identifier. This issue affects ISC version 2.1.0 and earlier. A workaround for those affected versions: the `h` should be changed to `l` after the data are written by ISC. This does not affect vertical-format output.
MCNP-53577	Plotter	The radial overlaid mesh lines in cylindrical `FMESH` plots are not displayed correctly when the origin of the view window is not aligned with the origin of the mesh. The location of plotted `FMESH` quantities are not affected.