List of changes in the program MCNelectron

Below is the list of changes for each version of MCNelectron that was available for download from http://web.vu.lt/ff/a.poskus/mcnelectron/, up to the current version (1.2.6).

1.0.1 (2014-06-29)**:**

1. In the current version, electron-positron pair production is simulated. However, a very simplistic model is used: both the electron and the positron continue in the direction of the incident photon, and positron energy distribution is assumed to be uniform.

2. In the current version, positron annihilation is simulated. Each annihilation event occurs when positron energy drops below the low-energy cutoff value.

3. In the current version, the source can emit positrons ("PART E+").

4. Fixed a bug in the procedure that is used to determine if a particle has escaped from the layer.

5. The current version implements a more realistic method of sampling of angular deflections of the two electrons existing after an electroionizatoin event. Their deflection angles are calculated from conservation of momentum.

6. The search for energy values in cross section tables has been made faster (this has been achieved by keeping in memory the current position in each table and using it as the starting position for the next search). As a result, the total simulation time has decreased by 20 % – 30 %.

1.0.2 (2014-07-01)**:**

1. In the current version, simulation of incoherent scattering at low energies (when atomic binding effects become important) is more realistic. The changes only affect the recoil electron. First, the atomic subshell is randomly selected on the basis of the number of electrons in each subshell. The energy of the ejected electron is obtained by subtracting the binding energy of the selected subshell from the energy loss of the photon. If the resulting energy is negative, then the electron is not ejected from the atom. Otherwise, the direction of its motion is the same as in the case of a free electron (only the energy is reduced as explained above). In previous versions, when tracking of particle position and direction was turned on ("TRACKPOS 1"), the effect of binding on the recoil electron energy was ignored (only the angular deflection included the effects of binding via the form factor).

2. In the output file, a new row "backscatter" has been added under "photon loss" and "electron loss". It contains the number and energy of particles that escaped from the layer through the surface that is exposed to the incident beam. The number and energy of photons or electrons that escaped from the other surface can be obtained by subtracting the values in the row "backscatter" from the corresponding values in the row "escape".

3. Fixed a bug that caused absence of angular deflection after coherent scattering of a photon.

1.0.3 (2014-07-04)**:**

1. The current version simulates Compton fluorescence. In the output file, the number and energy of the Compton fluorescence photons is shown separately from other fluorescence photons. However, the number and energy of electrons emitted during atomic relaxation after an incoherent scattering event is grouped with Auger electrons emitted after photoelectric absorption (similarly to MCNP6 output).

2. Fixed a bug that caused incorrect number of pair production events in the output file.

3. Added the keywords COH_THR, E_UNIT and E_CONTIGUOUS to the input file (see the updated README file for more information).

1.0.4 (2014-07-13)**:**

1. Added more information to the README file.

2. Added a subfolder "Test" with six MCNelectron input files and six MCNP6 input files corresponding to identical simulation conditions, together with respective output files. Additional information about those files and instructions on running the tests are in the file "Test_Test_info_.pdf".

1.0.5 (2014-07-19)**:**

1. Added the keyword NSTREAMS to the input file. It is used to specify the number of independent streams of random numbers. For optimum performance, NSTREAMS should be a multiple of the number of threads (see below). The maximum allowed number of streams is 256.

2. Added multithreading capability. The number of threads is specified on the command line in the same way as for MCNP, i.e., as "tasks <number_of_threads>". If there is no keyword "tasks" on the command line, then MCNelectron will run in single-thread mode. On multi-processor (or multi-core) systems, multithreading can reduce the simulation time significantly. The number of threads can be assigned any value that does not exceed the number of streams of random numbers (see above). However, there is no point in using a number of threads that is greater than the number of logical processors in the system (a further increase of the number of threads will not cause a decrease of the simulation time). The final results of the simulation do not depend on the number of threads that was used during the simulation.

3. Updated the README file.

2014-07-26**:**

4. Updated the batch file "W-value\Make_W-tables.bat", which is used to build the final tables of W-values. The previous version of that file worked correctly with MCNelectron v1.0.4 output files, but it did not work with MCNelectron v1.0.5 output files. The updated version of "Make_W-tables.bat" works correctly both with v1.0.4 and with v1.0.5 output files.

2014-08-13**:**

5. Fixed a bug that could in extremely rare cases cause a crash after a positron annihilation event (probability of such a crash was less than 0.01 %).

1.0.6 (2014-08-23)**:**

1. Log-log interpolation of photon cross sections and form factors has been made faster. This has been achieved by calculating their natural logarithms before the simulation and then using those logarithms for linear-linear interpolation with natural logarithm of photon energy (or of the change of photon momentum) as input. As a result, the simulation time in problems where most of the time is spent for sampling photon interactions (e.g., a beam of photons incident on a thin layer) has become less than a half of the simulation time with MCNelectron v1.0.5.

2. When sampling the distance to collision, expression "1 - pRand->random()" has been replaced by "pRand->random()". Although statistically the former expression is correct, too (because it also corresponds to a uniform distribution), it involves one extra operation (subtraction), which is not necessary. This replacement has been done in two places in the code (one for photons and one for electrons). Because of this change, simulation results when some of the particles can escape from the layer are not exactly identical to the results obtained with the previous versions, although they are identical statistically.

3. Added an optional command-line keyword "output_cs". When that keyword is used on the command line, files with values of excitation and electroionization cross sections are created before starting the

simulation. In such a case, MCNelectron creates a folder "Cross sections" in the current folder, with subfolders for each of the chemical elements that are present in the target material. Each of those subfolders contains a text file with values of the total excitation cross section, as well as text files with electroionization cross sections for each subshell. Each of those files contains two columns of numbers: energy values (eV) and corresponding cross section values (in barns).

1.0.7 (2014-08-26)**:**

1. Interpolation of cross sections has been made faster by calculating the slope coefficients for each energy sub-interval before the simulation and using their stored values for interpolation (in previous versions of MCNelectron, the slope coefficients were recalculated during each step of the simulation). In addition, sampling of energy transfer during electroionization and bremsstrahlung, as well as sampling of angular deflection during elastic scattering of electrons has been made faster using a similar approach. As a result of those changes, the simulation time has decreased by 10 % to 20 % in comparison with MCNelectron v1.0.6.

2014-08-28**:**

2. Fixed a bug that appeared after the most recent change (described in previous paragraph). That bug caused incorrect simulation results when alternative cross sections were used, because then the corresponding slope coefficients were not recalculated (instead, their values corresponding to the original cross sections were used). After fixing that bug, the slope coefficients are recalculated using the updated set of cross sections.

3. Merged source files "thread.cpp", "thread.h" and "MCNelectron.h" with "MCNelectron.cpp". Consequently, the total number of source files decreased from 6 to 3.

1.0.8 (2014-08-30)**:**

1. In the current version of MCNelectron, the values of cross sections, form factors and other quantities that only depend on electron or photon energy are only recalculated when the particle energy changes. As a result of this modification, the simulation time in problems where particle energy changes relatively infrequently (i.e., when most collisions are without energy change, or when the source emits monoenergetic particles and the average number of energy-loss collisions per source particle is much less than 1) has decreased by 10 % to 40 % in comparison with MCNelectron v1.0.7. In "W-value problems", i.e., when both elastic scattering of electrons and coherent scattering of photons are switched off and the target is infinite in all directions ("TRACKPOS 0"), the computational overhead associated with this modification causes an increase of the simulation time by up to 10 %.

2014-09-06**:**

2. The runtime components of Visual C++ libraries (i.e., the "Microsoft Visual C++ Redistributable") are no longer required to be installed on a computer in order to run MCNelectron (before this fix, absence of the mentioned files caused the error message "The program can't start because MSVCR100.dll is missing from your computer" after attempting to run MCNelectron).

2014-09-12**:**

3. Fixed a bug that caused an error message about incorrect format of the ENDF file when the target material contained elements with atomic number greater than 86 (i.e., the elements that are after radon in the periodic table).

2014-09-26**:**

4. Fixed a bug that caused too small energies of Compton recoil electrons when tracking of secondary photons was turned off ("TRACK_P 0").

2014-09-27**:**

5. Fixed a bug that caused the failure to start the file "W-value\MakeTable.exe", if the runtime components of Visual C++ libraries (i.e., the "Microsoft Visual C++ Redistributable") are not installed on the computer.

1.0.9 (2014-10-02)**:**

1. Simulation of several physical effects has been added:

   (a) non-isotropic angular distribution of bremsstrahlung photons,
   (b) partial suppression of positron bremsstrahlung in comparison with electron bremsstrahlung,
   (c) non-isotropic angular distribution of photoelectrons,
   (d) non-uniform distribution of positron and electron energies during pair production,
   (e) angular distribution of electrons and positrons created in pair production events,
   (f) Doppler broadening of energy distribution of incoherently scattered photons.

   For each of those effects, there is a keyword in the input file, which allows "turning on" or "turning off" simulation of the effect. By default, all those effects are simulated. If any of those effects is "turned off", then it will be replaced by a simplified model (e.g., isotropic bremsstrahlung) or neglected completely, as in the previous version of MCNelectron. For some of those effects, there are other keywords controlling some details of the simulation (see the updated README file for more information). Simulation of bremsstrahlung angular distribution and Doppler broadening requires additional data files, which are included in the MCNelectron distribution package: the file "Data\Brems_angular_prob_dens.dat" with angular probability densities of bremsstrahlung (calculated by integrating analytically over the triply differential cross-sections derived by Bethe and Heitler) and the file "Data\ComptonProfiles.dat" with Compton profiles. The data in the file "Brems_angular_prob_dens.dat" is in binary format (it is described in the updated README file). However, there is an executable "Data\ExtractBremsAngularDistr.exe", which extracts ASCII data from the binary data and creates a subfolder with human-readable files. Those text files can be used as input data for the simulation, too (instead of the mentioned binary file).

2. The current version of MCNelectron can calculate a pulse-height tally (similar to the MCNP "F8" tally).

3. Added the keyword TRACK_X to the input file (it controls tracking of X-ray photons emitted due to atomic relaxation).

4. The energy of the Compton recoil electron is calculated simply as the difference of the incident and scattered photon energies (i.e., the recoil electron energy is no longer reduced additionally by subtracting the binding energy of the subshell from which the electron was ejected).

5. When the keyword "output_cs" is specified on the command line, a file with values of the total ionization cross section is created (in addition to the files with ionization cross sections for individual subshells).

6. If the ENDF data folder name, which is specified in the input file after the keyword DIR, does not contain the full path (i.e., if it does not start with the backslash '\' and does not contain the colon ':'), then it is now assumed to be a subfolder of the folder where the MCNelectron executable is (rather than a subfolder of the current folder). The same applies to the keyword "BREMS_ANGULAR_DISTR_DIR", which is used to specify location of text files with bremsstrahlung angular distribution data, when the mentioned binary file "Data\Brems_angular_prob_dens.dat" is not used.

7. Updated the README file.

2014-10-03**:**

8. Multiple pulse-height tallies are now allowed.

9. If a value of absorbed energy exceeds the bin boundary of a pulse-height tally by 10^{-10} % of the bin width or less, then that energy value is treated as though it is equal to that boundary. In a special case when that boundary is the low-energy limit of the tally energy range, the source particle is not counted.

2014-10-05**:**

10. Added the keyword INCOH_IGNORE_BINDING_E to the input file (see the updated README file for information about that keyword).

11. Renamed the keyword “DOPPLER_BROADENING” to “INCOH_DOPPLER”, in order to be consistent with the naming convention of the other keywords that control simulation of physical effects (the first part of those keywords is derived from the name of an interaction type).

2014-10-30**:**

12. Added the capability to equalize the workload of different threads in multi-threading mode. When a thread finishes processing of all histories assigned to it, the program redistributes all unfinished random number streams among the threads, so that the number of active threads stays the same (in the previous versions, the threads were not restarted after termination). This is only possible when the number of unfinished random number streams is not less than the number of threads (otherwise, each stream of random numbers is assigned to a separate thread, so that the number of active threads is equal to the number of unfinished random number streams).

2014-11-14**:**

13. Fixed a bug that could in extremely rare cases cause an indeterminate energy value (“-1.#IND”) of a scattered photon when simulating Doppler energy broadening during incoherent scattering. Such an error was possible when the sampled value of the projection of the electron initial momentum to the direction of incident photon is very close to zero, resulting in calculation of the square root of a difference of two very close numbers. Although the exact value of that difference is always non-negative, the rounding errors can make it negative (however, since there are no known cases of this actually happening, it is possible that “granularity” of the random number generator implemented in MCNelectron would prevent this from occurring anyway).

2014-11-29**:**

14. Fixed a bug that prevented periodic redistribution of random number streams among the threads to equalize their workload (because of that bug, the mentioned redistribution was only done once per simulation).

15. Increased the maximum allowed number of random number streams from 256 to 2048.

2014-11-30**:**

16. Fixed a bug that sometimes caused incorrect value of the positron bremsstrahlung suppression factor $\Phi_{\mathrm{rad}}^{+}/\Phi_{\mathrm{rad}}^{-}$ (although inequality $0.02 \le \Phi_{\mathrm{rad}}^{+}/\Phi_{\mathrm{rad}}^{-} \le 1$ was still always satisfied, as it should). This bug only showed up when there was non-zero probability of pair production by photons and the target material consisted of more than one chemical element, or when the source emitted positrons.

2014-12-03**:**

17. Fixed a bug that could cause an infinite loop in the part of the code that simulates photon Doppler energy broadening during incoherent scattering. Such a situation occurred when the binding energy of the selected atomic subshell happened to be greater than the incident photon energy. This bug has been fixed by rejecting such subshells, i.e., by selecting only subshells with binding energy less than the incident photon energy.

2015-05-15**:**

18. Fixed a bug that prevented update of components of the electron momentum vector after a bremsstrahlung event, when energy of the emitted photon was below the photon cutoff energy (the absolute value of that vector and the electron energy were updated correctly). This bug could have a noticeable effect only when electron energy is comparable to the photon cutoff energy (1 eV by default) and when particle coordinates are tracked.

1.1.0 (2015-06-20)**:**

1. Plane-crossing tallies have been added, together with corresponding keywords ETALLY and PTALLY for electron and photon plane-crossing tallies, respectively.

2. Removed restrictions on the angle of incidence and on the position of the starting point of the incident beam (it may be inside the layer).

3. Added an option to use inner-shell electron impact ionization cross sections calculated using the distorted-wave Born approximation (DWBA). The latter cross sections are the same ones that are used by the PENELOPE code system. They are calculated using the code by D. Bote, F. Salvat, A. Jablonski, and C. J. Powell, which was published in 2009. That code has been translated from FORTRAN into C using f2c.exe and incorporated into MCNelectron (it is contained in the file "IonCS.cpp").

4. Added CUDA support, together with 18 new keywords ending with "_CUDA" (15 of them may be used both in the input file and on the command line; the remaining 3 keywords are command-line only). As a result, Nvidia graphics processing units (GPU) can now be used for simulations in parallel with the CPU. There are now two variants of MCNelectron: the CPU-only one (executable file name MCNelectron.exe) and the one with CUDA support (executable file name MCNelectron_CUDA.exe). The file MCNelectron_CUDA.exe can perform GPU-only, CPU-only or hybrid CPU/GPU computations (in CPU-only mode, MCNelectron_CUDA.exe produces the same results as MCNelectron.exe).

5. Changed the random number generator. The new random number generator is the original Fibonacci series generator published by George Marsaglia and Arif Zaman in 1987. In addition, simulations using CUDA devices may optionally use the XORWOW random number generator from the cuRAND library (the latter generator cannot be used on the CPU). The CPU random number generator was changed because the previous generator, which was a derivative of the one by G. Marsaglia and A. Zaman, was less suited for generation of a large number (≈30 000) of independent random number streams.

6. Added an option to apply a diffusion model of elastic scattering at extremely low electron energies, when the elastic scattering cross section is much larger than cross sections of other types of interactions. When the diffusion model is applied, the program first generates the distance to the next inelastic interaction event (i.e., the path traveled to that event, not the displacement), and samples the electron coordinates corresponding to that event from a Gaussian distribution with a root-mean-square displacement proportional to the square root of the generated path. This reduces the "electron trapping" effect (which could otherwise slow the simulation significantly) at the expense of fidelity of the simulated electron trajectory when its energy is only a few electronvolts above the ionization threshold of the material, or when it is below the ionization threshold. The transition to the diffusion model occurs at a user-specified value of the total-to-inelastic cross section ratio. The mentioned threshold value of that ratio is specified by the keyword "DIFFUSION_TOTAL_TO_INELASTIC_RATIO_THR". The allowed values of that threshold are from 10 to 10^{18}, and the default value is 100. The diffusion model of elastic scattering is never applied if the distance from the electron to the nearest surface of the layer is less than the mean free path times the square root of this parameter. In order to "turn off" application of the diffusion model, this parameter must be set to any sufficiently large value (e.g., to the maximum possible value 1e18).

7. The above-mentioned Gaussian distribution is only exact when the electron is at a depth d large enough to make the probability that the electron escapes from the layer before the next inelastic collision occurs practically zero. Consequently, the diffusion model of elastic scattering may introduce significant errors when estimating the moment and position of electron's escape from the layer if the electron is close to the surface. In order to decrease those errors, a single diffusion step mentioned above is replaced by a sufficiently large number of shorter diffusion steps, which are less than the root-mean-square displacement and also less than d (but still greater than the mean free path), if d is less than a user-specified number of r.m.s. displacements (the latter r.m.s. displacement is estimated as described above). This number is specified by the keyword "DIFFUSION_DIST_TO_RMSPATH_RATIO_THR". The allowed values of that number are from 2 to 10, and the default value is 5.

8. After an electron impact ionization event, the banked electron is the higher-energy ("primary") one (rather than the "knock-on" electron as in previous versions of MCNelectron). As a result, the number of banked electrons has been reduced significantly in comparison with previous versions (usually by an order of magnitude). An exception is a simulation on a CUDA device using the option "heap_CUDA 2": in this case the banked electron is the lower-energy one (i.e., the knock-on electron).

9. Replaced the keyword "INCOH_IGNORE_BINDING_E" by "INCOH_SUBTRACT_BINDING_E".

10. Updated the user manual.

2015-06-24**:**

11. Added the subfolder "X-rays" with 99 MCNelectron_CUDA input files and the corresponding output files. Each of them is used to simulate characteristic X-ray emission from a thick target composed of one of those elements: C, Al, Si, Ti, Fe, Cu, Zn, Ge, Zr, Sn, Sm, Ta, W, Pt, Au and Pb, bombarded by monoenergetic electrons with energy from 5 keV to 30 keV. Information about those files is provided in Section 8 of the user manual and in the file "_X-rays_info_.pdf", which is in the same subfolder.

2015-06-25**:**

12. Fixed a bug that caused the seed of the XORWOW random number generator from the cuRAND library to be always equal to the seed of the Fibonacci series random number generator, which is used on the CPU.

13. Fixed a bug that caused a CUDA error "invalid argument" after pressing any of the keys "+", "-", "[" or "]" during simulation using options "emFirst_CUDA 1" and "emLast_CUDA 1" on a computer with two or more CUDA devices (i.e., when the CUDA device No. 1 was being emulated by the CPU, while the CUDA device No. 2 was not being emulated).

2015-06-27**:**

14. The inner-shell electron impact ionization cross sections calculated using the DWBA approximation are now stored in the file "Data\IonDWBA.dat" in binary format, instead of being recalculated at the start of each simulation. Accordingly, the code for calculating those cross sections has been removed from MCNelectron.

15. Fixed a bug that caused incorrect values of inner-shell positron impact ionization cross sections when the option "ION_DWBA 1" was specified in the input file.

2015-07-03:

16. Added the file "randomlib.c" with the original code of the Fibonacci series random number generator to the distribution package; inserted a comment line with a reference to that file at the beginning of files "RandGen.cpp", "RandGen.cu" and "RandGen_CUDA.cu".

2015-07-15:

17. The error messages displayed after detecting an invalid command-line parameter have been made more informative.

2015-08-21:

18. Fixed a bug in the CUDA version of MCNelectron that sometimes caused overflow of the banked particles buffer when the option "heap_CUDA 2" was used. Now that overflow is prevented by controlling which of the two electrons existing after an impact ionization event is banked: when the banked particles buffer is less than 80 % full, the banked electron is the lower-energy one (i.e., the "secondary" or "knock-on" electron), and if the banked particles buffer is more than 80 % full, then the banked electron is the higher-energy one (i.e., the "primary" electron). In the case "heap_CUDA 0" or "heap_CUDA 1", the banked electron is always the primary one (as before).

1.1.1 (2015-10-02)**:**

1. Added an option to use elastic scattering cross sections obtained from relativistic (Dirac) partial-wave calculations. Those cross sections were calculated using the code ELSEPA by F. Salvat, A. Jablonski and C. J. Powell, which was published in 2005. These are the same cross sections that are used by the PENELOPE-2014 code system. Their values are stored in files "Data\ElasticDPW_totalCS.dat" and "Data\ElasticDPW_angularDistr.dat" in binary format. The latter file contains angular distributions, which were calculated for electron energies from 10 eV to 1 GeV on a logarithmic scale (the increment of the base-10 logarithm of energy is equal to 0.1). Usage of those cross sections is controlled by the switch "ELASTIC_DPW". By default, that switch is "on", i.e., those cross sections are used instead of the ENDF/B elastic scattering cross sections.

2. The default state of the switch “ION_DWBA” is “on”, i.e., 1 (previously, its default state was “off”, i.e., 0). Now, in order to avoid replacement of available ENDF/B data by third-party data, it is necessary to specify the following three options in the input file: “ION_DWBA 0”, “ELASTIC_DPW 0”, “BREMS_POSITRON_CORRECTION 0” (the latter option has an effect only when pair production is possible, or when the source emits positrons).

3. Added an option to reduce the size of the tables with values of the angular CDF of elastic scattering by removing some points from them (see the updated user manual for more details).

4. Added an option to speed up the process of searching for values in tabular data by “mapping” the tables to “index arrays” corresponding to sets of equidistant values (see the updated user manual for more details).

5. Added a range type “THETA”, corresponding to the angle between the particle’s momentum vector and the normal vector of a “tallying plane” (see the updated user manual for more details).

6. Increased the maximum number of random-number streams from 31329 to $2^{16} = 65536$. However, the maximum number of unique random number streams that can be generated by the Fibonacci series generator using a given seed (specified after the keyword “SEED”) is still equal to 31329. Consequently, the random number streams with sequence numbers from 31330 to $2 \cdot 31329 = 62658$ will be generated using the next seed (i.e., the seed that exceeds the specified seed by 1), and the streams with sequence numbers from 62659 to 65536 will be generated using the seed that exceeds the specified seed by 2 (if the seed value obtained by adding 1 or 2 is greater than 30081, then it is reduced by 30082).

7. Added four CUDA-specific keywords “use_CUDA”, “skip_CUDA”, “r_CUDA” and “em_CUDA”, which allow to use lists of CUDA devices, as well as to assign device workload ratios for individual CUDA devices (see the updated user manual for more details).

8. Added a CUDA-specific keyword “bankModeThr_CUDA”, which controls the “bank mode” when option “heap_CUDA 2” is used. The term “bank mode” refers to the choice of an electron that has to be banked after an impact ionization event: the banked electron may be either the lower-energy one (the so-called “secondary” or “knock-on” electron), or the higher-energy one (the “primary” electron). Normally, when option “heap_CUDA 2” is used, the knock-on electron is the one that is banked. However, this may cause an overflow of the banked particles buffer in the case of high energies of source particles and thick targets. Consequently, the bank mode is changed when the filling fraction of the banked particles buffer exceeds a certain value. That “threshold” value of the filling fraction is specified after this keyword. Its default value is 0.8, i.e., the bank mode will be changed when the banked particles buffer is more than 80 % full. In the case “heap_CUDA 0” or “heap_CUDA 1”, this keyword is ignored (in those cases, the primary electron is always the one that is banked).

9. Added four command-line keywords “NSTREAMS”, “SEED”, “N” and “E”. The first three ones are equivalent to corresponding keywords specified in the input file and allow overriding the number of random-number streams, the seed of the Fibonacci series random number generator and the number of source particles specified in the input file, respectively. The keyword “E” makes it possible to replace the definition of the source energy spectrum specified in the input file by a source of monoenergetic particles.

10. Added three command-line keywords “in”, “out” and “alt” that must precede the names of the input file, the output file and the file with alternative cross sections information, respectively (previously, the file names were entered without any keywords).

11. Moved subfolders “W-value” and “X-rays” into a new subfolder “Simulations”. Added a subfolder “Simulations\Backscattering” with results of simulations of electron backscattering from a thick layer of gold at incident electron energies from 200 eV to 15 MeV, obtained with MCNelectron_CUDA using the option “ELASTIC_DPW 1”.

12. MCNelectron and MCNelectron_CUDA are now distributed as WinRAR self-extracting archives. The names of those files are "MCNelectron_install.exe" and "MCNelectron_CUDA_install.exe", respectively. They are contained in WinZip archives "MCNelectron_install.zip" and "MCNelectron_CUDA_install.zip", respectively (they replace the files "MCNelectron.zip" and "MCNelectron_CUDA.zip", which were used previously).

13. Updated the user manual.

2015-10-10:

14. Fixed a bug that caused inconsistent sign of a bin count increment (±1) when a range type "MU" was used in the specification of a two-dimensional plane-crossing tally (before fixing this bug, the mentioned sign depended on the direction of crossing, whereas for this range type the sign should be always positive).

15. Fixed a bug in the CUDA version of MCNelectron that caused inconsistent sign of a bin count increment (±1) when a range type "THETA" was used in the specification of a two-dimensional plane-crossing tally (before fixing this bug, the mentioned sign depended on the direction of crossing, whereas for this range type the sign should be always positive). This bug affected only the 2D tallies calculated on CUDA devices (but not the tallies calculated on the CPU).

2015-10-14:

16. Fixed incorrect interpretation of a negative value of the input parameter ELASTIC_PDF_ERROR in the CPU-only version of MCNelectron. Before fixing this bug, a negative value of ELASTIC_PDF_ERROR was always interpreted by the CPU-only version of MCNelectron as a zero value (i.e., no values were removed from tables with values of elastic scattering angular PDF), whereas the correct way is to ignore the mentioned sign when the option "ELASTIC_DPW 1" is used (as explained in the user manual). ***Note***: This bug was absent in the CUDA version of MCNelectron.

1.1.2 (2015-11-10)**:**

1. Added an option to use interaction forcing (see Section 4.5 in the updated user manual).

2. The files with plane-crossing and pulse-height tally data now include relative standard deviations for each bin (in addition to the particle counts).

3. Added an option to terminate the simulation when the maximum, minimum or average relative standard deviation, calculated over all bins of one or more plane-crossing or pulse-height tallies ("control tallies"), becomes less than the "target error" specified by the user (see Section 4.4 in the updated user manual).

4. The simulations of electron-induced characteristic X-ray emission (folder "Simulations\X-rays" of the distribution packages) have been re-done using interaction forcing and "control tallies".

5. Updated the user manual.

2015-11-11:

6. Fixed a bug in MCNelectron_CUDA that could cause "loss" of one source particle during a GPU-only simulation using two or more CUDA devices with automatic workload balancing (this bug was caused by rounding errors when converting a floating-point number to an integer number).

2015-11-12:

7. Fixed a bug that caused exclusion of forced interactions from the sampled interaction types during analog tracking of particles with weights below the cutoff value. This bug could affect the simulation results only when at least one of parameters FORCING_WT_MIN or CUT_WT was not equal to zero (which is the default value of those parameters).

2015-12-02:

8. Fixed a bug in MCNelectron_CUDA that caused failure to terminate a CPU-only simulation after the minimum, maximum or average relative standard deviation of a "control tally" became less than the "target error".

9. Fixed a bug in MCNelectron_CUDA that caused incorrect value of the current minimum, maximum or average relative standard deviation when checking for a termination condition in a two-dimensional "control tally".

10. Corrected the language in several error messages.

2015-12-07:

11. Fixed a bug that could cause a crash or incorrect results when the specification of the "target error" of a "control tally" was not the last entry of the line with the tally definition in the input file.

2016-01-12:

12. Added subfolder "Test\Verification_for_K_and_L_X-rays" with the results of a χ^2 test for the difference between characteristic X-ray yields computed by MCNelectron (using ENDF/B data) and MCNP6.1 in single-event mode. The same subfolder contains all input and output files of both codes, which were used for this verification (those simulations were set to terminate when the number of incident electrons reached 10^9, or when the relative standard deviation of the number of photons counted in the bin containing the characteristic X-ray line energy became less than 1 %, whichever happened first).

13. Added Section 7.2 to the "MCNelectron User's Manual" with a detailed description of the above-mentioned χ^2 test and of the simulation data used for it.

14. Increased the precision of the simulations of characteristic X-ray emission using DWBA cross sections (folder "Simulations\X-rays"). All those simulations were re-done with the termination condition that the number of incident electrons must be 10^9, or the relative standard deviation of the number of photons counted in the bin containing the characteristic X-ray line energy must be less than 1 %.

15. Moved the files that had been previously in the folder "Test" into the subfolder "Test\Samples".

1.2.0 (2016-04-25)**:**

1. Added the capability to handle complex geometries. MCNelectron v1.2.0 implements constructive solid geometry on regions bounded by planes, spheres, circular cylinders and circular one-sheet cones (see Section 4.2 in the updated user manual). Specification of complex geometries may be simplified using the union and complement operators. In order to speed up tracking of particles when the simulation setup contains a large number of bounding surfaces, the entire volume of the simulated system is split into rectangular volume elements ("voxels") with associated data structures describing the cells and surfaces intersecting with each voxel, and only the surfaces corresponding to the current voxel (i.e., the voxel that contains the particle) have to be checked for intersection with the particle's track (see Sections 4.2 and 4.3 in the updated user manual).

2. Added the capability to calculate cell-entry tallies and pulse-height tallies for individual cells (see Section 4.4 in the updated user manual).

3. Improved calculation of directions of the two electrons existing after an impact ionization event (i.e., the primary electron and the knock-on electron) at low energies of the incident electron (of the order of the binding energy). In previous versions of MCNelectron, the angular distribution of the electrons was excessively peaked in the forward direction when the kinetic energy of the incident electron only slightly exceeded the binding energy of the subshell where the vacancy had been created. The formulas used for calculation of the angles are given in Section 5 of the updated user manual (the same formulas are implemented in MCNP6).

4. Single-line comments can now be used in input files. A comment is specified by the double slash "//".

5. All simulations described in the user manual have been re-done with MCNelectron v1.2.0.

6. Updated the user manual.

2016-04-26:

7. Fixed calculation of pulse-height tallies for unions of cells. Previously, only the energy absorbed in the first cell of the union was taken into account. Now, energy absorbed in all cells of the union is tallied.

2016-04-28:

8. Added the column "absorbed energy" to the table with information about electron activity in each cell. This column contains values of the total absorbed energy in each cell.

2016-04-29:

9. Added the keyword MAXDISTANCE_RATIO (see Section 4.2 of the updated user manual).

10. Fixed a bug in MCNelectron_CUDA.exe that caused a CUDA error when attempting to copy cell-entry tally data from a CUDA device to the host computer memory (this bug affected only the simulations that did not use interaction forcing).

2016-05-01:

11. Fixed a bug in MCNelectron_CUDA.exe that sometimes caused a failure to update the electron cross sections in complex geometry mode after an electron escape event.

2016-05-02:

12. Fixed a bug that caused a crash in complex geometry mode when the source of particles was inside an empty cell (i.e., a cell that is not filled with any material).

2016-05-03:

13. Fixed a bug in MCNelectron_CUDA.exe that sometimes caused incorrect behavior of the program after an electron escape event in complex geometry mode.

14. Interchanged rows "energy cutoff" and "coherent scatter" in the output table "photon events". Interchanged rows "energy cutoff" and "elastic scatter" in the output table "electron events".

2016-05-04:

15. Fixed a bug that could cause incorrect results of plane-crossing and cell-entry tallies in complex geometry mode when the source point was inside a cell.

2016-05-05:

16. Fixed a bug that caused an error when updating a cell-entry tally after an entry of a particle into a "host cell" from one of the cells that are contained in it (i.e., specified after the complement operator in the definition of the host cell), when the current voxel intersects only with the bounding surfaces of the contained cell (i.e., does not intersect with the bounding surfaces of the host cell).

17. Fixed a bug that could cause calculation errors in complex geometry mode, if some cells were defined using two or more complement operators.

2016-05-06:

18. Removed the limitation on the number of chemical elements in the simulated system.

2016-05-08:

19. The parts of the source code file MCNelectron_CUDA.cu, where the values returned by the CUDA runtime device and memory management routines are checked, have been rewritten in a shorter and more readable form (the functionality has not been modified).

20. Fixed the following bug: the voxels that are entirely contained in a "host" cell and intersect with some of the contained cells, i.e., the cells that are listed after the complement operator in the definition of the host cell, were assigned only the mentioned "first-level" contained cells, but were mistakenly not assigned the "deeper" contained cells (listed after the complement operator in the definitions of the "first-level" contained cells). This bug could cause calculation errors when some of the cells were defined using the complement operator.

2016-05-09:

21. Increased the maximum allowed value of identifiers of surfaces, materials, cells, tallies and ranges: now it is 2^{31}.

2016-05-10:

22. Added a column with line numbers to five tables with cell-by-cell statistics in the main output file.

2016-05-12:

23. Fixed a bug in MCNelectron_CUDA.exe that caused a crash when using CUDA devices with the option "TRACKPOS 0".

2016-05-13:

24. In MCNelectron_CUDA.exe, the warnings about a change of a status of an interaction from "forced" to "non-forced" are now displayed only once for each type of interaction in each used CUDA device (previously, they were displayed once for every CUDA task, which caused a very large number of identical warnings).

2016-05-15:

25. Fixed a bug in MCNelectron_CUDA.exe that sometimes caused incorrect behavior of the program after a photon escape event in complex geometry mode.

26. Fixed a bug that sometimes caused a failure to update a cell-entry tally after a particle has crossed the cell surface, if that cell is a part of a union of cells.

27. In the current version, all surfaces of a cell belonging to a union of cells are allowed to be excluded from a cell-entry tally, if some of the surfaces of the other parts of that union are included (previously, MCNelectron would quit with an error message in such a situation).

2016-05-29:

28. Fixed a bug that sometimes caused incorrect identification of the crossed bounding surface when a particle entered a cell. This bug affected only the results of the cell-entry tallies that exclude some of the bounding surfaces of a cell.

2016-06-06:

29. Fixed a bug that sometimes caused incorrect sampling of a source point when the source of radiation is uniformly distributed inside an arbitrary cell (directive "SOURCE_POSITION 3"). Because of that bug, a source point was occasionally generated outside of the source cell (near its surface).

30. Decreased granularity of the Fibonacci random number generator used for uniform generation of random source points inside an arbitrary cell (directive "SOURCE_POSITION 3") by a factor of 10^7. A side effect of this change is that the time spent for uniform generation of random source points inside a cell increased by factor of 2. ***Note***: This change does not affect the part of the simulation that is done on CUDA devices using the XORWOW generator from the cuRAND library (directive „rand_CUDA 2").

31. Added a table to the output file with the minimum and maximum coordinates of randomly generated source points inside an arbitrary cell (directive "SOURCE_POSITION 3").

2016-07-21:

32. Fixed a bug in MCNelectron_CUDA.exe that caused an error when using CUDA devices in complex geometry mode with a non-volume source ("SOURCE_POSITION 1" or "SOURCE_POSITION 2").

1.2.1 (2016-08-03)**:**

1. Added the capability to output particle track data (only the source particle tracks are output). Because of this modification, three new keywords have been added: “NTRACKS”, “NCOLLISIONS_TRACK”, and “SKIP_TRACKS_WITHOUT_COLLISIONS” (see the updated user manual for more information).

2. Two stages of initial geometry processing (definition of the voxel data and calculation of cell volumes) have been shortened by using multithreading. Because of this modification, two new keywords have been added: “RAND_BLOCK_SIZE” and “NTHREADS_GEOM” (see the updated user manual).

3. If the initial geometry processing takes more than several seconds, feedback is provided to the user in the form of messages informing about the progress in the current stage of geometry processing.

4. Now, MCNelectron checks if there are no identical materials in the simulation setup. If a material is found with the composition identical to the composition of one of the materials defined before, then a warning is displayed and all references to that material in definitions of cells and unions are replaced by the reference to the previously-defined material.

5. It is now possible to insert a material name after its identifier in the input file. The empty cells may be also optionally assigned names (in this case, the name must be specified after the zero or a negative number following “C<n>”). The names of the materials and the names of the empty cells are ignored by MCNelectron, but they are displayed in the main window of the “shell” program MCNScript, which is designed for running Monte Carlo simulations with MCNelectron, as well as for organizing the MCNelectron input and output files and creating graphics rendering scripts from MCNelectron input and track data files (MCNScript is described in a separate user manual).

6. Interchanged rows “total loss” and “total collisions” in the output tables “photon events” and “electron events”.

7. Updated the user manual.

2016-08-04:

8. If writing of track data to a file takes more than 1 s, feedback is provided to the user in the form of messages informing about the progress in the track data output.

2016-08-05:

9. Fixed a bug that prevented the program from detecting the occurrence of a source cell that is not “bracketed” by coordinate planes (the correct behavior in such a situation is informing the user about the input error and exiting).

2016-08-07:

10. Added an option to specify the ENDF data folder on the MCNelectron command line, using the keyword “DIR” (previously, this keyword could be used only in the input file).

2016-08-10:

11. Modified the track data file format by inserting a line with the track sequence number before the line with the sequence number of the source particle (this has been done in order to facilitate locating the data of a track with a known sequence number).

2016-08-16:

12. The number of random points per cell (for estimating the cell volume and the source point sampling efficiency) is now calculated by dividing the total number of random points (defined using a new keyword “N_RAND_TOTAL_VOLUME”) by the number of cells. The keywords “N_RAND_CELL_VOLUME” and “N_RAND_SOURCE_VOLUME”, which previously meant the number of random points for calculating the cell volume and the source sampling efficiency, respectively, now mean the *minimum* values of the respective numbers.

13. Fixed a bug that caused an error message about intersecting cells when one of the cells was inside a void, which was inside the other cell.

14. Fixed a bug that could cause incorrect results if the complement operator was applied to a cell whose definition included complements of other cells.

2016-08-19:

15. Fixed a bug that caused a memory error when the total number of voxels was greater than approximately $7.5 \cdot 10^6$.

2016-08-24:

16. Fixed a bug that could cause incorrect determination of the voxel containing the particle if the distance between the particle and the nearest voxel boundary was less than DISTANCE_PRECISION (its default value is 10^{-12} cm).

17. Fixed a bug that sometimes caused incorrect calculation of the intersection point of a particle track and a conical surface.

1.2.2 (2016-09-04)**:**

1. The 64-bit executables have been added to the distribution packages. The Visual Studio 2010 projects have been modified to generate both the 32-bit and the 64-bit executables (the names of the 64-bit executable files end with "_x64.exe").

2. A more efficient algorithm of interaction forcing has been implemented (see the updated user manual). Consequently, five of the keywords related to interaction forcing ("FORCING_FACTOR", "FORCING_PROB_MIN", "FORCING_PROB_MAX", "FORCING_WT_MIN" and "CUT_WT") have been removed and a new keyword "FORCE_WT" has been added.

3. The line-continuation character '&' can now be used in the input file.

4. Updated the user manual.

2016-09-05:

5. Fixed a bug that sometimes caused a crash when a particle exited a cell that coincides with a voxel.

6. Fixed a bug that sometimes caused an infinite loop upon a particle's exit from a cell if the round-off error of the particle's coordinates became greater that the "distance precision" (defined using the keyword DISTANCE_PRECISION).

2016-09-08:

7. Fixed a bug in MCNelectron_CUDA that caused a CUDA memory error in simple geometry mode when the number of source particles was less than the number of CUDA tasks (it is possible only when the option "heap_CUDA 2" is used).

8. Fixed a bug in MCNelectron_CUDA that caused incorrect simulation of incoherent scattering of photons when the photon energy was less than the ionization threshold.

2016-09-23:

9. Fixed a bug in MCNelectron_CUDA that caused a failure to create the track data file when the simulation setup did not require output of files with the overall cell-by-cell interaction statistics, tally data or second-by-second CUDA device statistics.

10. Removed a superfluous error message about a failure to calculate the tangent of a non-planar intersection line of two surfaces (this error does not usually affect the results, and in the rare cases when it does, another warning message is displayed).

2016-09-24:

11. Fixed a bug that sometimes caused incorrect determination of the next voxel number when tracking the particle, if the distance between the particle and the voxel boundary was less than DISTANCE_PRECISION (its default value is 10^{-12} cm). One of possible results of this bug was an infinite loop.

1.2.3 (2016-10-03)**:**

1. Improved computational efficiency of the random selection of electron interaction types. Because of this improvement, the time of the simulation of electron transport in materials consisting of a large number of chemical elements (such as biological tissues) has decreased down to 30 % – 40 % of the simulation time observed before this modification of the code.

2. Added an option to remove some of the energies for which the electron cross sections are stored in the cross section tables that are created before starting each simulation. This allows to reduce the time of the energy lookup and to reduce the amount of memory used for the simulation. The relative error introduced by this removal of data points is controlled using a new keyword "SIGMA_ERR" (see the updated user manual for more information). The default is "SIGMA_ERR 0.001". In order to minimize the differences between the results obtained with MCNelectron v1.2.3 and the previous versions of MCNelectron, the option "SIGMA_ERR 0" should be used.

3. Added an option to use the step-like ("bar") probability density functions (PDF), with the value of each bar equal to the average PDF in the corresponding interval of the argument values (similarly to the method implemented in MCNP6.1 for approximating the PDF of the cosine of the electron elastic scattering angle in single-event mode). In this case, the corresponding cumulative distribution function (CDF) is piecewise-linear. Alternatively, the PDF is continuous (piecewise-linear), whereas the CDF is piecewise-quadratic, i.e., "smooth" (as in all previous versions of MCNelectron). This option can be "switched on" or "switched off" using a new keyword "SMOOTH_CDF". By default, the smooth CDF is "on".

4. Fixed an error in calculation of the track length when the particle crossed an interface or a gap separating two cells consisting of the same material, but with different densities.

5. Fixed a bug that could cause an infinite loop if the particle is exactly on the line of intersection of two faces of a rectangular cell, which coincides with a voxel.

6. Calculation of the angle between the direction vectors of the knock-on electron and the incident electron has been slightly modified: in Equation (5.2) of the user manual, the incident electron momentum p_0 has been replaced by the "reduced" momentum p'_0, corresponding to the kinetic energy E_0 reduced by the binding energy, i.e., $E_0 - B$ (in MCNelectron v1.2.0 – 1.2.2, the calculation of the mentioned angle was done using the "full" momentum p_0 in the denominator and the reduced momentum in the numerator). Because of this change, the results obtained with MCNelectron v1.2.3 are not identical to the results obtained with v1.2.0 – 1.2.2 (this difference is usually not statistically significant).

7. Updated the user manual.

2016-10-05**:**

8. Fixed a bug that caused a crash when the simulation setup satisfied the following three conditions simultaneously: 1) positron creation was possible; 2) positron interaction cross sections were required to be different from the corresponding electron interaction cross sections; 3) the option "SIGMA_ERR" with a non-zero relative error was used. By default, such a situation occurs when the energy of the source particles exceeds the electron rest energy by a factor of 2 or greater, or when the source particles are positrons.

9. Fixed a bug that sometimes caused incorrect determination of the next voxel number when tracking the particle, if the distance between the particle and the voxel edge was less than DISTANCE_PRECISION (its default value is 10^{-12} cm), and the particle's momentum was directed outside of the voxel.

1.2.4 (2016-11-03)**:**

1. Added the capability to output tracks of secondary particles, with optional limitation of the secondary particle "generation number". For each track, the output file with track data contains the information about the type of the event where the primary particle of that track was created (a total of 12 categories of tracks are distinguished), as well as the sequence number of the secondary-particle "generation", to which that particle belongs (see the updated user manual for more information). The sequence number of the last generation of secondary particles to be included in the created track dataset is specified using a new keyword "MAXGEN_TRACKS".

2. Improved the computer memory usage when creating extremely large track datasets (for example, hundreds of millions of collisions, which would require several gigabytes of computer memory). Previously, the coordinates and energies of all collisions of a given history were stored in a single continuous array, which was dynamically re-allocated every time when its size became insufficient for inclusion of additional collisions. Since re-allocation of very large arrays is relatively slow, this method could cause a significant slowdown when the track dataset was very large. In the current version of MCNelectron, the track data are split into smaller arrays ("chunks"), which are not re-allocated: instead, a new "chunk" is allocated when needed. The number of vertices in a single chunk is specified using a new keyword "CHUNKSIZE_TRACKS".

3. In addition to the mentioned changes that are related to inclusion of secondary particles in the track data files, the method used for distinguishing the escape tracks and truncated tracks has been modified: now, the escape tracks are distinguished by the minus sign before the final energy, and the truncated tracks are distinguished by positive final energy (the final energy corresponding to the absorption tracks is zero, as before). The number of vertices is never output with the minus sign. Besides, the meaning of the coordinates specified in the last line for an escape track has been changed: now, that line contains the coordinates of the escape point (i.e., the point where the particle crossed the boundary of a cell for the last time on its way out of the system).

4. Simulation of elastic scattering using the ENDF/B data ("ELASTIC_DPW 0") has been made more accurate by taking into account the fact that the elastic scattering cross sections stored in the ENDF/B library correspond only to the values of the cosine of the scattering angle from −1 to 0.999999 (in the previous versions of MCNelectron, those cross sections were treated as though they correspond to the entire range of the cosine values, i.e., −1 to 1). This improvement consists in adding the small-angle scattering cross section (corresponding to the values of the cosine of the scattering angle from 0.999999 to 1, where the angular distribution is evaluated analytically) to the total cross section of elastic scattering. At electron energies above 1 MeV, the small-angle scattering becomes significant (~1 % or more in the total number of the elastic scattering events), and at energies above 20 MeV it becomes the dominant type of elastic scattering. The values of small-angle and large-angle scattering cross sections are calculated on the basis of the number specified after a new keyword "ELASTIC_SMALL_ANGLE" (see the updated user manual). If the number after "ELASTIC_SMALL_ANGLE" is 0, 1 or 2, then the small-angle elastic scattering cross section is calculated either by integrating the analytic part of the elastic scattering angular PDF, or by linear extrapolation of the tabular PDF from larger angles (see the updated user manual for more information). If the number after "ELASTIC_SMALL_ANGLE" is 3, then the elastic scattering cross sections from the original EEDL library (Ref. [5] of the updated user manual) will be used (see below).

5. Added an option to use the values of the elastic scattering cross sections from the original EEDL library (Ref. [5] of the updated user manual). Unlike the ENDF/B library, the EEDL cross section tables include the total elastic scattering cross sections (corresponding to the entire range of the cosine values, i.e., −1 to 1), hence the value of the small-angle elastic scattering cross section can be calculated simply as the difference of the EEDL and ENDF/B elastic scattering cross sections. This option is selected by using the directives "ELASTIC_DPW 0" and "ELASTIC_SMALL_ANGLE 3". The EEDL elastic scattering cross sections are stored in the ASCII file "Data\EEDL_elastic.dat" (see the updated user manual for more information).

6. If the options “ELASTIC_DPW 0” and “ELASTIC_SMALL_ANGLE <n>” with non-zero “<n>” are used, then the electron cross section data files created by MCNelectron (by using the command-line option “output_cs 1”) contain an additional column with the values of the large-angle elastic scattering cross section (corresponding to the values of the cosine of the scattering angle from −1 to 0.999999).

7. Added two new keywords “SMOOTH_CDF_ELASTIC” and “SMOOTH_CDF_INELASTIC”, which function similarly to “SMOOTH_CDF”, but affect only the PDF of electron elastic scattering or all other tabular PDFs, respectively (see the updated user manual for details).

8. The files with pulse-height tally data contain two additional columns with the values of the total absorbed energy and the average absorbed dose (in Gy) for each bin. The column with the absorbed dose is present only when the total mass of the system or of the given cell (for cell pulse-height tallies) is finite and known.

9. Added an option to specify the cell volume in the definition of a cell or a union of cells (keyword “VOL”). The user-specified volume replaces the volume estimated by MCNelectron when calculating the mass of the cell and the average absorbed dose (if those two volumes differ by more than 10 standard deviations, a warning is displayed).

10. Additional information has been included in the output file. For example, the densities and atomic concentrations are now provided for all cells (not only the cells with a known finite volume). The average absorbed doses for each cell have been added, too.

11. Computational efficiency of selection of the probability density tables has been slightly improved by skipping the lookup procedure when the current energy value is between the same two adjacent data points as the previous energy value.

12. Fixed a bug that caused computational errors when applying the diffusion model to low-energy elastic scattering (keyword “DIFFUSION_TOTAL_TO_INELASTIC_RATIO_THR“).

13. Fixed a bug that caused incorrect calculation of the particles’ coordinates in simple geometry mode when the direction of the normal vector of the front surface of the layer (defined using the keyword SURFACE) was the same as the direction of the *Z* axis and the *z* coordinate of the intersection point of the front surface with the *Z* axis was non-zero.

14. Fixed a bug in MCNelectron_CUDA that could cause a crash of the program when copying the chemical element data to a CUDA device, if the simulation setup includes an extremely large number of chemical elements (more than 50).

15. Updated the user manual.

2016-11-05**:**

16. Fixed a bug that caused a crash when creating a track dataset, if the number of tracks specified in the input file after the keyword “NTRACKS” was greater than 100.

2016-11-07**:**

17. Fixed a minor bug that occasionally caused some of the last tracks to be excluded from the track dataset, if the number specified after the keyword “NTRACKS” was equal to or slightly less than the number of source particles.

18. Fixed a bug in MCNelectron_CUDA that caused the number of track datasets created in a hybrid CPU/GPU simulation to be less than the required number (specified after the keyword “NTRACKS”), because some of the tracks were mistakenly assigned to the random number streams used by CUDA devices, which cannot create track data.

2016-11-19**:**

19. Fixed a bug in the function Roots() used to find intersection points of a curve and a surface by the simplex method (this bug occasionally caused some of the intersection points to be missed, if the number of the intersection points was greater than 2).

20. Decreased the errors of the intersection points determined by the simplex method by a factor of 10. As a result, those errors became less likely to cause an automatic increase of the "distance precision" parameter (this parameter is defined using the keyword DISTANCE_PRECISION).

21. Fixed a bug that caused incorrect determination of relative positions of a circle and a sphere when they did not intersect (this, in turn, sometimes caused a failure to determine the coordinate planes that "bracket" a cell with two or more spherical bounding surfaces, but did not cause any errors in tracking the particles).

2016-11-20**:**

22. Eliminated an error message about a failure to determine the position of a cell relative to a coordinate plane. This message was displayed when the standard method of testing for an intersection with a coordinate plane (based on analysis of the intersection lines) failed, but an interior point of the cell had been found, and it happened to be exactly on that coordinate plane. The corrected code flags such cells as intersecting with that coordinate plane.

2016-11-24**:**

23. Added the number of lost source particles (i.e., the particles that did not enter any cell) to the line of the MCNelectron_CUDA output file where the number of source particles is specified. This change concerns only MCNelectron_CUDA, because the output file of the CPU-only version of MCNelectron includes the number of lost source particles starting with MCNelectron v1.2.0.

1.2.5 (2016-12-11)**:**

1. Added a new type of geometric primitives – the so-called "macrobodies", i.e., the sets of two or more "regular" surfaces defining various 3D shapes, which can be used in definitions of cells and cell-entry tallies along with the regular surfaces. There are 8 "standard" macrobodies (right circular cylinder, truncated right-angle cone and 6 types of polyhedra) and a "general" macrobody (a shorthand notation for an arbitrary set of user-defined regular surfaces and their "senses"). It is also possible to reference individual facets (i.e., bounding surfaces) of each macrobody using the format "<macrobody ID>.<facet No>". 7 of the 8 standard macrobodies are defined using the same format as in MCNP (however, the standard macrobodies "BOX", "RPP" and "RHP" in MCNelectron must be finite in all dimensions, unlike in MCNP, where they can be infinite in a dimension). The numbering of their facets is also the same as in MCNP (see the updated user manual for more info).

2. In plane-crossing tallies and cell-entry tallies, added the option to take into account only the particles of specific origin (for example, only the source particles, or only the characteristic X-ray photons). This allows eliminating the "background" from the particles that are not important for the problem that is being solved (for example, eliminating the contribution of bremsstrahlung photons, if the goal is estimation of characteristic X-ray generation efficiency). The particle origin is specified by a set of comma-delimited numbers from 0 to 11 after the keyword "ORG" in the definition of the tally, where the zero corresponds to the source particles, and the meanings of the other numbers are the same as of the numbers used to categorize the secondary particles in the track data files (see the updated user manual for more information).

3. Added the "mixed" electron-photon tallies and the corresponding keywords "TALLY" (generalization of "ETALLY" and "PTALLY") and "CELL_ENTRY" (generalization of "ECELL_ENTRY" and "PCELL_ENTRY").

4. Added an option to apply electron interaction forcing only in particular cells of the geometry, or only in particular materials. This is controlled by the directive "FORCE 1" or "FORCE 0" in the definition of the material or the cell (or the union of cells). A new global switch "FORCE_MATERIALS" is used to

tell the program if interaction forcing should be applied by default. In any case, interaction forcing will be applied only after defining the set of forced electron interaction types (as in the previous versions of MCNelectron).

5. Added a new keyword “CHECK_FOR_INTERSECTIONS”, which controls checking for intersections of cells that are not parts of the same union of cells. This keyword must be followed by the number 0, 1, 2 or 3, where “0” corresponds to absence of any checking, and the other numbers correspond to varying degrees of thoroughness of the checking (see the updated user manual for more information).

6. Added an option to include the values of the energy transfer in the files with plane-crossing and cell-entry tally data. This is controlled by the directive “E_FLUX 1” or “E_FLUX 0” in the definition of the tally. A new global switch “TALLIES_E_FLUX” is used to tell the program if the energy transfer data should be output by default.

7. Inclusion of the energy deposition tally data (i.e., the values of absorbed energy and absorbed dose for each energy bin) in the files with the pulse-height tally data has been made optional, similarly to controlling the output of energy transfer data (see above). This is controlled by the directive “E_DEPOSITION 1” or “E_DEPOSITION 0” in the definition of the pulse-height tally. A new global switch “TALLIES_E_DEPOSITION” is used to tell the program if the energy deposition data should be output by default.

8. It is now possible to define a planar surface not only by specifying the coordinates of a point and the normal vector, but also by specifying the four coefficients of the plane equation or the coordinates of three points on the plane (see the updated user manual for more information).

9. Coinciding surfaces are no longer treated as a geometry error. Instead, a warning is displayed, but execution of the program is not interrupted.

10. The redundant regions in the definition of a cell (i.e., the regions that entirely contain at least one other region specified in the same definition) are no longer treated as a geometry error. If the cell volume is non-zero, then those regions are removed from the definition of the cell without interrupting execution of the program and without displaying any warning message.

11. Duplicate keywords are no longer allowed in the input file, except for keywords PULSE_HEIGHT, COORD_X, COORD_Y and COORD_Z.

12. The maximum numbers of surfaces and cells have been increased by a factor of 5. Now, the maximum total number of surfaces is 10000, the maximum number of cells is 5000, and the maximum number of surfaces per one cell is 5000. The limitation on the number of materials has been removed.

13. The 32-bit version of MCNelectron_CUDA is no longer compiled using CUDA Toolkit 6.5. Instead, it is compiled using CUDA Toolkit 7.5, which is also used for compiling the 64-bit version of MCNelectron_CUDA. Since the 32-bit version of the cuRAND library (which provides facilities for generation of random numbers) is not included in CUDA Toolkit 7.x and in its newer versions, the random-number generator type 2 (directive “rand_CUDA 2”) is no longer allowed when the 32-bit version of MCNelectron_CUDA is used.

14. Fixed the errors in the formatting of several error messages.

15. Updated the user manual.

2016-12-14**:**

16. The 32-bit version of MCNelectron_CUDA has been re-compiled using a smaller set of compute capabilities: 2.0, 3.5 and 5.0 (i.e., the compute capabilities 3.0 and 3.7 have been removed). This has been done in order to decrease the size of the executable file and to decrease the build time in future updates.

2016-12-16**:**

17. Fixed the errors in the construction of two types of standard macrobodies: right hexagonal prism (RHP or HEX) and four-sided polyhedron (ARB4).

2016-12-17**:**

18. A test simulation with all types of macrobodies supported by MCNelectron has been added. The MCNelectron and MCNP6.1.1 input and output files of that simulation have been placed into the subfolder "Test\Samples".

2016-12-18**:**

19. The sign convention for the identifiers of facets of standard macrobodies in the definitions of cells and general macrobodies has been changed. Now, if the identifier of a facet of a standard macrobody is specified in the definition of a cell (or a general macrobody) with the minus sign, then it means the same region that is used in the definition of the macrobody (i.e., the region intersecting with the inside of the macrobody). Otherwise, it means the complement of that region. [In the initial release of MCNelectron v1.2.5, the opposite convention was used.] This change has been made in order to improve compatibility with MCNP and to be consistent with the practice of assigning the negative sense to the inside of a closed surface.

2016-12-21**:**

20. The memory usage has been decreased by allocating the cell data array dynamically. Now, the cell data array is allocated in 20-element "chunks", so that the number of elements in this array does not exceed the number of cells by more than 19 (previously, the cell data array was static and contained 5000 elements, each more than 200 kB in size, resulting in more than 1 GB of used memory).

2016-12-23**:**

21. Fixed a bug that caused a crash if all vertices of a cell were the intersection points of four or more bounding surfaces.

2016-12-25**:**

22. Fixed a bug that could cause errors when checking if a facet of a polyhedron-type standard macrobody coincides with one of previously-defined planes.

2016-12-28**:**

23. Fixed a bug that caused incorrect concentration of atoms in the cells composing a union (excluding the first cell of the union), if the density of the material was specified in the definition of the union. [This bug first appeared in MCNelectron v1.2.4.]

2016-12-29**:**

24. Fixed the errors in calculation of a cell-entry tally for a union of cells when some of the bounding surfaces of that union are excluded from the tally.

2016-12-31**:**

25. Corrected the warnings that are displayed after a failure to find an interior point of a cell.

1.2.6 (2017-02-08)**:**

1. Added the capability to use coordinate transformations. The coordinate transformations can be used in MCNelectron in two ways: (a) as a method to create a new cell or surface by rotating and translating an existing cell or surface; (b) as a method to define an auxiliary right-hand Cartesian coordinate system for a subsequent definition of a new surface in this auxiliary coordinate system (this is useful if the definition of the surface in the auxiliary system is much simpler than its definition in the primary coordinate system). See the updated user manual for more details.

2. Added the capability of procedural generation of MCNelectron input directives. This is achieved by embedding user programs in the MCNelectron input file. The user programs must be written in MCNEcode – a scripting language, whose basic syntax elements are similar to the corresponding syntax

elements of the C language. MCNEcode has 70 built-in functions, including mathematical functions, data output functions and others. In the context of MCNelectron input directives, probably the most useful feature of MCNEcode is the loop operator "for", which can significantly shorten the specification of multiple directives having similar format (examples of using the "for" loop for defining complex geometries involving multiple transformations of the same object are given in Section 4.4.2 of the updated user manual). The overview of procedural generation of MCNelectron input directives is in Section 4.12 of the updated user manual, and the detailed specification of the MCNEcode syntax and built-in functions is in the Appendix.

3. Added the capability to save the tally data not only after ending the simulation, but also during the simulation, at regular intervals of time specified by the user. The corresponding "global" keywords are TALLIES_OUTPUT and TALLIES_OUTPUT_INTERVAL. If TALLIES_OUTPUT is 2, then the data of all tallies will be written to files at the specified intervals (this is the default behavior). If TALLIES_OUTPUT is 1, then only the data of the "control tallies" (i.e., the tallies with the specified "target error") will be written at the specified intervals. If TALLIES_OUTPUT is 0, then the tally data will be saved to files only after ending the simulation. The time period of saving the tally data is specified using the keyword TALLIES_OUTPUT_INTERVAL (it is 60 s by default). In the definition of each tally, the keywords OUTPUT and INTERVAL should be used to override the global settings controlled by TALLIES_OUTPUT and TALLIES_OUTPUT_INTERVAL.

4. It is now possible to stop the simulation manually at any time and to make MCNelectron create the final files with the current values of all required statistics. This is done by pressing the 'Esc' and 'Q' keys (in that order). If only 'Esc' was pressed, then it is possible to "cancel" the escape key by pressing any key except 'Esc' and 'Q'. After that, the 'Esc' key would have to be pressed again in order to stop the simulation.

5. In the current version of MCNelectron, the identifier of a previously-defined plane may be specified in the definition of a plane-crossing tally, and the identifier of a previously-defined energy range may be specified in the definition of pulse-height tally (see the updated user manual for additional details).

6. Several minor improvements of the user interface have been made, several minor bugs have been fixed.

7. Updated the user manual.

2017-02-09**:**

8. Added the option to define the *y* direction vector of the auxiliary coordinate system in terms of its azimuth and elevation angles. This is possible only after specifying the azimuth and elevation of the *x* direction vector. Thus, the "TR<n>" directive must have 7 numerical entries in this case: three components of the translation vector and two pairs of the azimuth and elevation angles.

9. Added the MCNEcode built-in function "dir", which should be used instead of "write" to create MCNelectron input directives after a call to "fopenw" (i.e., after opening a user-specified file for writing). If there is no file open for writing, the built-in functions "write" and "write2" are equivalent to "dir" and "dir2", i.e., their output is interpreted as MCNelectron input directives (which can be optionally written to the log file). However, after a call to "fopenw", the output of function "write" is written to the specified file instead of being added to the set of the MCNelectron input directives, whereas the output of the function "dir" continues to be interpreted as MCNelectron input directives (and to be written to the log file, if the command-line option "writeLog 1" has been specified).

10. Eliminated the error message "Some command-line arguments are missing" when only the input file name is specified on the command line.

2017-02-10**:**

11. Minor improvements have been made in the MCNelectron's built-in compiler/interpreter of MCNEcode.

2017-02-11**:**

12. The user manual has been updated and corrected.

2017-02-16**:**

13. Fixed a crash when the number of cell-entry tallies is greater than 100.

14. The tally data files that are updated periodically during the simulation are no longer kept open constantly until the end of the simulation. Instead, they are closed after each update and re-opened before the next update. Consequently, the number of tally data files that can be updated during the simulation is no longer limited to the maximum number of simultaneously open files allowed by the Windows operating system (that number is 500 by default).

15. The CUDA compute capability 5.0 has been removed from the set of compute capabilities of the 32-bit version of MCNelectron_CUDA (i.e., the executable file has been compiled using only the compute capabilities 2.0 and 3.5). This has been done in order to decrease the size of the executable file and to decrease the build time in future updates.

2017-02-17**:**

16. The estimates of the relative standard deviations of the pulse-height tallies have been made more accurate: now those standard deviations are calculated under the assumption of the binomial distribution (previously, the Poisson distribution was assumed, which is correct only in the limit of small probabilities).

2017-02-21**:**

17. Minor improvements have been made in the MCNelectron's built-in compiler/interpreter of MCNEcode.

2017-03-10**:**

18. The user manual has been corrected.