Geometric Representations in the Developmental Monte Carlo Transport Code MC21

T. Donovan[*1] and L. Tyburski[2]
[1] *KAPL, Inc. – A Lockheed Martin Company*
Schenectady, NY
[2] *Bechtel Bettis, Inc.*
West Mifflin, PA

Abstract

The geometry kernel of the developmental Monte Carlo transport code MC21 is designed as a combination of the geometry capabilities of several existing Monte Carlo codes. This combination of capabilities is intended to meet efficiently the general requirements associated with in-core design products and, at the same time, be flexible enough to support highly general geometric models. This paper provides a description of the different geometry representations of MC21 and outlines how the geometric data is stored internally through the use of Fortran-90 data structures.

Finally, two alternative geometric representations of a published BWR unit assembly model are discussed. Results for the two representations are contrasted, including k-effective results, relative memory footprints, and relative computational speeds. While total memory footprint is not noticeably reduced, results show significant speed advantages of one representation.

KEYWORDS: Monte Carlo, geometry, MC21

1. Introduction

MC21 [1] is the Monte Carlo neutron and photon transport code currently under joint development at Bettis Atomic Power Laboratory and Knolls Atomic Power Laboratory (KAPL). MC21 is the Monte Carlo transport kernel of the broader Common Monte Carlo Design Tool (CMCDT), which is also currently under development. The Vision of CMCDT is to provide an automated, computer-aided modeling and post-processing environment integrated with a Monte Carlo solver that is optimized for reactor analysis. CMCDT represents a strategy to push the Monte Carlo method beyond its traditional role as a benchmarking tool or "tool of last resort" and into a dominant design role. This strategy requires continued growth in computing power to bear the high cost of 3-D transport theory calculations, transport kernels that maximize performance and minimize memory for the largest models, and computer aided modeling and results processing to facilitate rapid engineering turn-around.

This paper presents an overview of the geometry representations of MC21. The geometry kernel in MC21 is one of the most important sub-systems of the solver and affects virtually every other aspect of MC21. In particular, because particle tracking is intricately woven into the Monte Carlo geometry, efficient tracking depends on a suitable geometry representation. The MC21

[*] Corresponding author, Tel. 518-395-6145, E-mail: donovat@kapl.gov

geometry kernel is designed as a combination of the geometry capabilities of RCP01 [2], RACER [3], and MCNP [4] optimized with respect to the general requirements associated with in-core design products. For example, like RACER and RCP01, MC21 possesses a dedicated two-dimensional (2-D) geometry kernel that ensures that 2-D geometries can be compactly represented and efficiently tracked. Similar to MCNP, MC21 possesses a flexible three-dimensional combinatorial geometry representation, including union/intersection operators and hierarchical relationships.

MC21's internal geometry representations are considered necessary to meet performance and memory requirements associated with 3-D reactor design applications. An individual user of MC21 is not expected to interact significantly with each of these representations, as this could be cumbersome and would not be consistent with the simplified model building platform that CMCDT is intended to provide. As the user interface features of CMCDT are in an early stage of development, this letter does not discuss how model building at the CMCDT level will be translated to the MC21 internal geometric representations.

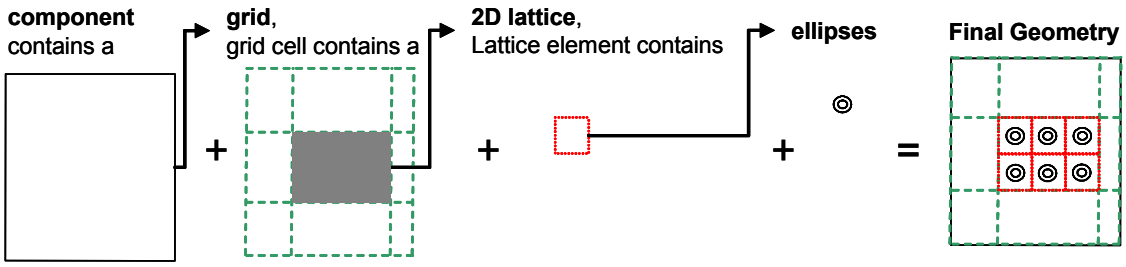
2. Geometry Description

2.1. Overview

The five main geometric constructs of MC21 are: surface, component, grid, 2Dlattice, and ellipse. General quadratic surfaces are the building blocks of components. Components, which are generally equivalent to cells in MCNP, are volumes defined by union and/or intersection sets of surface half-spaces. In addition to their half-space definitions, components can be further specified hierarchically with respect to other components. Through this hierarchical definition, components can be constrained to exist only within another component or may contain other components.

Any component can possess additional internal geometric detail in the form of a grid. The three available grid types are a spherical grid (*r*), a cylindrical grid (*r,z*) and a Cartesian grid (*x,y,z*). The Cartesian grid may have an arbitrary angle between the grid's x and y axes, and the axial grid separations are specified perpendicular to the grid's x-y plane. The grid cells of the Cartesian grid may possess an x-y 2Dlattice which repeats in the grid cell. The x-y 2Dlattice has the same angle between the x and y axes as the Cartesian grid it resides within. Finally, each 2Dlattice element may contain a series of ellipses. These ellipses may be nested, tangent, or disjoint, but for tracking efficiency may not intersect within the lattice element. Figure 1 shows a simple example of how these separate constructs can be combined to form a complex geometry.

Figure 1: The geometric structures in MC21 combined to form a single model.



For simplicity, the component view in Figure 1 is assumed to be an x-y slice and it is stated as a given that no geometric axial (z) variation exists in this component. This type of component is

referred to as 2D-extruded and is commonly encountered in nuclear reactors. The component is defined as the intersection of the four plane half-spaces shown in Figure 1 and the top and bottom planes, which are not shown.

The component is specified as containing a Cartesian grid. The grid is defined by parallel sets of x planes, y planes, and z planes. Essentially, the x and y planes define a non-uniform 2-D grid, which is then stacked axially based on the number of specified z planes. Each unique grid element is referred to as a grid cell. A grid must entirely fill the component it resides within but does not need to be contiguous with the component boundaries. Any part of the grid that extends beyond the component is effectively truncated. This means that a component that is not 2D-extruded in nature may still possess a Cartesian grid.

Each grid cell may be assigned additional geometric information in the form of a 2Dlattice. The 2Dlattice is a uniform mesh that repeats on the x-y grid frame and is the same height as the grid cell it occupies. The 2Dlattice repeats to the extent of the grid cell, and 2Dlattice elements that overlap the grid cell are effectively truncated by the grid cell. The repeating lattice element may be populated with a series of ellipses. An ellipse may be arbitrarily positioned and oriented in the 2Dlattice element and extends the height of the 2Dlattice element. Ellipse placement is subject to the restriction that ellipses may not intersect within the 2Dlattice element. The final result is a uniform array of elliptical tubes which fills the grid cell.

Materials are assigned at the innermost level of geometry. If a component contains no grid, it contains a material. If a grid cell contains no 2Dlattice, it contains a material. A 2Dlattice element always contains a base material, and each ellipse contains a material assignment.

The data associated with the MC21 geometry are stored in Fortran-90 data structures. This protocol permits associated data to be stored in a single location, which leads to improved maintainability and extensibility by simplifying the layout of data within the source code. The following sections detail the data structures for all the MC21 geometry constructs

2.2. Surface

All surfaces are defined in the absolute coordinate frame. MC21 holds a surface array of length equal to the number of surfaces in the model. Each array element in the surface array is of data type surface, which is a Fortran 90 derived data structure that holds all the data associated with a surface. The surface data structure possesses the geometric information shown in Table 1.

Table 1: Surface data structure attributes.

Surface Data	Data Type	Description
surface type	Integer	Defines the form of the surface equation
surface coefficients	Real Array	The equation coefficients. The number of coefficients is dependent on the surface type.
positive surface neighbors	Integer Array	A dynamic array listing the indices of all the components that are bounded by the positive half-space of the surface
negative surface neighbors	Integer Array	A dynamic array listing the indices of all the components that are bounded by the negative half-space of the surface
boundary condition	Integer	The boundary condition: normal, escape, or reflecting

Every surface has a surface type indicator and a set of coefficients that explicitly defines the surface. The number of coefficients is dependent on the surface type. At the current time, MC21

permits general quadratic surfaces as well as reduced forms of the general quadratic equation. Each surface contains two neighbor arrays: one for the positive half-space and one for the negative half-space. The positive and negative neighbor arrays for a given surface are lists of all of the components whose definitions include the positive and negative half-space of that surface, respectively. These neighbor lists are computed by MC21 from the component definitions and are used to accelerate tracking through components. Each surface also contains its boundary condition. The current options are normal (transmission), escape, and reflective.

2.3. Component

A component is a uniquely defined region in three-dimensional space. A component's two main features are its boundary representation and its hierarchical associations with other components. MC21 holds a component array of length equal to the number of components in the model. Each element of the component array is of data type component, which is a derived data structure that holds all the data associated with a component. The component data structure possesses the geometric information shown in Table 2.

Table 2: Component data structure attributes.

Component Data	Data Type	Description
number of items	Integer	The number of operators and surfaces used in the boundary list
boundary list	Integer Array	Array defining the Component as a string of intersections and/or unions of half-spaces
number of inner components	Integer	The number of Components contained within this Component
inner components	Integer Array	The list of Components that have been defined as existing only within this component
number of outer components	Integer	The number of Components that this component is contained within (0 or 1)
outer component	Integer Array	The Component that this Component exists within, if any
material	Integer	The index of the material assigned to the component
grid form	Integer	The type of grid contained within the component, if any
grid	Grid	The grid data structure

The boundary list is a set theory description of the unions and/or intersections of surface half-spaces that combine to form the boundaries of the component. The boundary list is held by MC21 as a string of integers. Surfaces are referenced by the position they occupy in the surface array, and the indices are signed according to their positive or negative half-space. The intersection operator is implicit in the boundary list. Unique integers are used to denote the union operator as well as right and left parentheses that are used to determine the order of operation. These special integers are identical to those used by MCNP for the same purpose.

The component contains hierarchical information that completes the geometric description of the component. This information describes the numbers and indices of other components at the next hierarchical level up (inner components) and the next hierarchical level down (outer components). The first, or outermost, hierarchical level is level 0. Only a single component

resides at level 0. All other components in a MC21 model exist either directly or indirectly within the outermost component. The MC21 definition of hierarchy is a generalization of the restricted hierarchy available in RCP01 and is similar to the Universe construct of MCNP.

A component may contain a grid, in which case the component data structure contains the attached grid data structure. The grid data structure is described in the next section.

2.4. Grid

MC21 permits three types of grids to be specified: Cartesian, spherical, and cylindrical. Spherical and cylindrical grids are very similar in structure to the Cartesian grid, except these two grid types do not permit the inclusion of embedded 2Dlattice information. For this reason, only the Cartesian grid will be described here. The component data structure possesses the geometric information shown in Table 3.

Table 3: Cartesian grid data structure attributes.

Grid Data	Data Type	Description
# of x planes	Integer	The # of x planes defining the grid
x list	Real Array	The x planes, from least to greatest
# of y plane	Integer	The # of y planes defining the grid
y list	Real Array	The y planes, from least to greatest
# of z planes	Integer	The # of z planes defining the grid
z list	Real Array	The z planes, from least to greatest
origin	Integer	Point in the absolute (component) frame that the grid origin is linked to
rotation	Real Array	component-to-grid rotation matrix
x-y angle data	Real	The sine, cosine, cosecant, and cotangent of the angle between the x and y planes
material	Integer Array	For each grid cell, the index of the material assigned
max Repeat	Integer Array	For each grid cell, the # of repeated 2Dlattice elements in the positive x and positive y directions relative to the 2Dlattice origin point
min Repeat	Integer Array	For each grid cell, the # of repeated 2Dlattice elements in the negative x and negative y directions relative to the 2Dlattice origin point

A Cartesian grid is defined by three sets of x coordinates, y coordinates, and z coordinates, respectively. The x and y planes may be orthogonal (by default) or may have an arbitrary included angle. The axial grid separations created by the z coordinates are specified perpendicular to the grid's x-y plane. A grid zone is the volume between two adjacent parallel grid boundaries. A grid cell is the volume between a complete set of adjacent grid boundaries, which is six adjacent planes for a Cartesian grid. For a Cartesian grid, the number of grid zones for each coordinate x, y, and z is the number of constant planes in each coordinate minus one. A grid cell is uniquely specified by the x, y, and z zones that it occupies. The number of grid cells is the product of the number of x zones, y zones, and z zones.

In addition to its own geometric structure, a Cartesian grid must also hold information describing its connection to the component coordinate frame and its internal 2Dlattice

information. The grid's origin and rotation connect the grid's coordinate frame to the absolute coordinate frame on which all components are defined. Each grid is uniquely defined. The grid data structure holds 2Dlattice information in two separate ways. First, maximum and minimum repeat indices are stored for every grid cell for both the x and y directions. If a grid does not contain a 2Dlattice, then the repeat indices are zeroes. The 2Dlattice data structure itself is not stored in the Grid data structure. Rather, a 2Dlattice is assigned to a grid cell through the use of special material indices. A grid cell containing a material index greater than the maximum number of defined materials is interpreted by MC21 as containing a 2Dlattice.

2.5. 2-D Lattice and embedded ellipses

Obviously not all of the internal detail of a component can be represented by the non-uniform x-y mesh of a Cartesian grid. Additional detail may be represented by arrays of elliptical cylinders within Cartesian grid cells. In MC21, this repeating array of elliptical cylinders is defined by a geometric structure referred to as a 2Dlattice. The x-y cross section of the repeating element, or sub-cell, of the 2Dlattice is a parallelogram whose x and y axes are parallel to the x and y axes of the grid cell. The sub-cell is considered to extend along the grid's z axis from the bottom to the top of the grid cell that contains it. The 2D lattice sub-cell automatically repeats in the positive and in the negative x and y directions, starting from an arbitrarily specified point in the grid cell, until the x and y boundaries of the grid cell are reached. Any 2Dlattice sub-cells that protrude through the x or y boundaries of the grid cell are effectively truncated. Table 4 describes both the 2Dlattice and ellipse data structures.

Table 4: 2Dlattice and ellipse data structures

2Dlattice Data	**Data Type**	**Description**
lattice origin	Real	Origin of the 2Dlattice on the grid coordinate frame
delta x	Real	2Dlattice element width in the x direction
delta y	Real	2Dlattice element width in the y direction
# of ellipses	Integer	The number of ellipses in the repeating 2Dlattice element
ellipses	Ellipse Array	An array of ellipse data types
material	Integer Array	The indices of the materials assigned to the sub-cell and each of the embedded ellipses
Ellipse data	**Data Type**	**Description**
center	Real	Origin of the ellipse center on the 2Dlattice coordinate frame
x-y angle data	Real	The sine and cosine of the angle between the ellipse semi-major axis and the grid x axis
ellipse coefficients	Real array	The inverse of the major and minor radii, respectively.

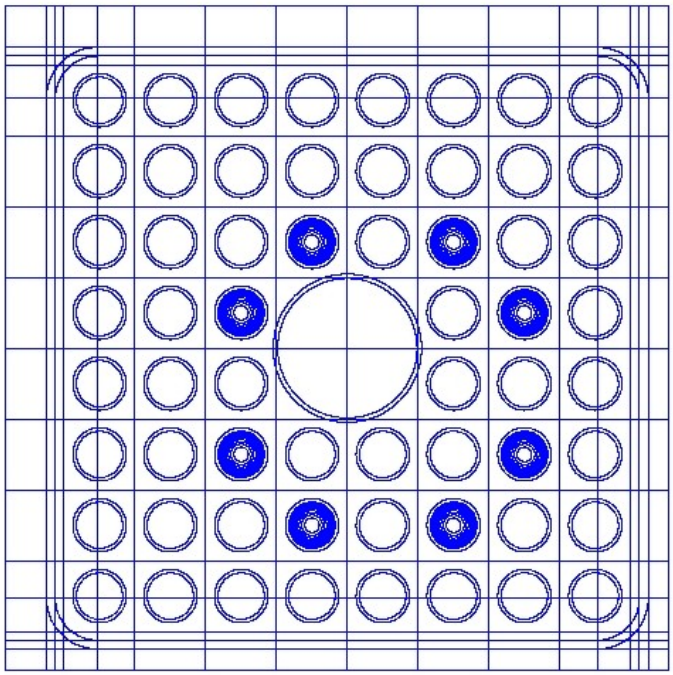
The interior of the sub-cell may be divided by an arbitrary number of ellipses. The ellipses may have arbitrary positions, rotations, and lengths along the semi-major and semi-minor axes. All ellipses are truncated by the x and y boundaries of the 2D lattice sub-cell. In order to reduce tracking complexity (i.e., cost), the areas of any two ellipses within the sub-cell must either be disjoint, tangent, or one completely contained within the other.

3. Application of MC21 to a Benchmark Problem

3.1. Model Description

A benchmark problem previously published by D.J. Kelly [5] to describe the RACER depletion capability was used to compare the relative performance of MC21 when calculating k_∞ for two different geometric representations of the benchmark. The benchmark problem is a fully reflected conventional BWR 8x8 fuel assembly design with a single large central water rod and eight urania-gadolinia rods. Figure 2 illustrates the Benchmark model as a single gridded component in MC21.

Figure 2: A BWR assembly as a single MC21 component containing a Cartesian grid and 2Dlattice data



The outer boundary of Figure 2, including the top and bottom z-planes (not shown), forms the component boundary. The assembly is 15.24cm by 15.24cm and is 1cm high. The component is at hierarchy level 1, meaning it resides within another bounding component at level 0. However, since the component shown has fully reflecting boundaries, neutrons never enter the bounding component. All of the inner detail shown in Figure 2 is represented as a single Cartesian grid with 2Dlattice and ellipse data to represent the numerous circles and circular arcs shown. Eight of the pins are further subdivided into ten equal volume regions using nested ellipses in the associated grid cells. Only one axial (z) zone is modeled.

In addition to the model shown in Figure 2, an MC21 model was created using only components. The majority of the physical detail shown in Figure 2 was modeled as 274 explicit components that were specified as simple intersections of surface half-spaces. These 274 components all reside at hierarchy level 1 and reside within a bounding component at level 0. A small amount of the physical space shown in Figure 2 was implicitly represented since it falls into the bounding component and was not accounted for by the components at level 1.

3.2. Results

The two different models were run on an AMD Opteron-based Linux cluster using four processors per job. The only tally computed with each model was the multiplication factor. Table 5 shows k_∞ results for each model and compares the relative speed and relative memory footprint of each model.

Table 5: Benchmark Results from 2 Alternative Representations.

MC21 Model	**k_∞**	Relative Speed [histories/time]	Relative Memory
Pure Component	1.0896(6)	1	1
Gridded Geometry	1.0897(6)	9.6	1

As expected, the multiplication factors are statistically equivalent for the two MC21 models. The memory footprints were also equivalent between the two representations. This is due to the fact that while the memory taken up by geometry information is less in the gridded model case than for the pure component case, the overall memory was dominated by the problem's cross section data. In general, for large problems where memory management is considered critical, the memory footprint is expected to be dominated by 3-D tally data. Therefore the use of grid representations is not likely to be justified on the basis of memory savings. The justification for modeling with grids lies in the expected speed benefit over an equivalent component representation. Table 5 shows that, for this model, the gridded model runs 9.6 times faster than the component model.

4. Conclusion

The geometric representations of MC21 are designed to provide a highly flexible 3-D combinatorial geometry capability coupled with a dedicated two-dimensional geometry kernel that allows 2-D extruded geometries to be compactly represented and efficiently tracked. MC21's combinatorial geometry and use of component hierarchy is similar to MCNP, while the 2-D geometries of grid, 2Dlattice, and ellipse are based on similar structures in RCP01 and RACER. Ultimately, when MC21 is fully integrated into CMCDT, user's will predominantly model using combinations of 3-D and 2-D primitive objects rather than dealing directly at the less intuitive level of surface half-spaces and grids. The CMCDT model will then be transferred via an automated process to the desired MC21 representation based on a minimal set of user requests.

The internal data structures are presented to illustrate how MC21 internally stores all geometric data in an intuitive way using Fortran 90. The BWR benchmark demonstrates the overall capability level of MC21, but is especially useful in contrasting the performance differences that can be observed due to the different geometric modeling options that MC21 provides. Both versions of the benchmark are considered to be tracking-dominated, meaning that the largest fraction of the overall run time is due to tracking. This benchmark therefore maximizes the performance differences between the two representations. For models with large numbers of nuclides and tally regions, collision handling and tallying also become dominant fractions of run-time. For such models, the performance difference between gridded and component-only models is not expected to be as great as shown in Table 5, but is clearly expected to be significant.

Acknowledgements

The authors wish to thank P. Dobreff and D. Kelly for their support in the benchmark model generation and results profiling.

References

1) T. M. Sutton, T. H. Trumbull, “A potential issue involving the application of the unit base transformation to the interpolation of secondary energy distributions,” Transactions of the American Nuclear Society, Washington, DC, **93**, 555 (2005).
2) L.A. Ondis II, L.J. Tyburski, and B.S. Moskowitz, "RCP01 - A Monte Carlo program for solving neutron and photon transport problems in three-dimensional geometry with detailed energy description and depletion capability," B-TM-1638 (2000).
3) T.M. Sutton, et al., “The physical models and statistical procedures used in the Racer Monte Carlo code,” KAPL-4840 (1999)
4) “MCNP – A general Monte Carlo n-particle transport code,” LA-13709-M (2000).
5) D. J. Kelly, "Depletion of a BWR lattice using the RACER continuous energy Monte Carlo code," Proceedings of the International Conference on Mathematics and Computations, Reactor Physics and Environmental Analysis, Portland, Oregon, **2**, 1011 (1995).