LA-UR-13-20997
Approved for public release; distribution is unlimited.

Title: MCNP6 Moving Objects. Part I: Theory

Author(s): Durkee, Joe W. Jr.
Johns, Russell C.
Waters, Laurie S.

Intended for: Progress in Nuclear Energy

# MCNP6 MOVING OBJECTS
# PART I: THEORY

**Joe W. Durkee, Jr., Russell C. Johns, and Laurie S. Waters**
**Los Alamos National Laboratory**
**jdurkee@lanl.gov**
**505-665-0530**
**Fax: 505-665-2897**
**PO Box 1663, MS K575**
**Los Alamos, NM 87545**

## ABSTRACT

Various radiation transport simulations pertaining to active and passive interrogation, medical physics, space applications, and other disciplines involve the motion of objects. In some instances radiation is emitted from within an object, while in other cases radiation is incident on an object. In our latest innovation for MCNP6, we introduce the capability to perform self-contained radiation-transport simulations involving moving objects. Object motion is treated using rigid-body kinematics as characterized by rectilinear translation, curvilinear translation, and curvilinear rotation along motion segments. Individual motion segments can be linked together to create complicated paths. Simulations can entail the simultaneous motion of up to 1000 moving objects. The moving-objects feature is designed for source-mode ("SDEF") simulations, and includes the capacity to model moving sources comprised of prompt particles as well as delayed neutrons and gammas. The motion of objects and sources can be simulated by constant velocity, acceleration or deceleration, or simple relocation. In addition, the MCNP6 plot utility has been upgraded to permit animated plots of geometry as it evolves in time. The moving-objects feature may be executed in either serial mode or parallel mode using

MPI. In this article we delineate the formulations and code modifications required in MCNP6 to implement the new moving-objects capability. In the companion article we illustrate the capability using several simulations. The moving-objects innovation will bring enhanced realism to simulations that exhibit motion.

KEYWORDS: MCNP6; moving objects; kinematics; rigid-body; rectilinear; curvilinear, DGNAA.

## 1. Introduction

Computational simulation methods offer a cost-effective avenue with which to examine complex phenomena in a virtual setting. These methods often serve as a complement to experimental and design endeavors. Simulations can help optimize designs via in silico changes to system parameters and characteristics as well as help to expeditiously conduct scenario investigations.

In the realm of radiation transport simulations, Monte Carlo techniques provide the analyst with a technique to study phenomena with a high degree of fidelity and sophistication. In the Monte Carlo paradigm, processes are chosen at random to depict the interaction of radiation with matter. It is not necessary to restrict particle energies, directions, spatial locations, or times to a finite number of values. Moreover, information can be generated regarding statistical behavior, including average values that can be compared with experimental results.

MCNP6 (Goorley et al., 2011) is the culmination of a multi-year effort to merge MCNP5 (Brown, 2003a and 2003b) and MCNPX (Pelowitz, 2011). MCNP6 is a Monte Carlo particle (radiation quanta) radiation-transport code that is developed at Los Alamos National Laboratory (LANL). With its roots extending back to the 1940's (vonNeumann, 1947), MCNP6 offers an assortment of features that provide detailed modeling and assessment capabilities. These features include three-dimensional (3-D) geometry modeling, continuous-energy transport, transport of 36 different types of fundamental particles as well as 2000+ types of heavy ions, delayed neutron and gamma radiation treatment, isotopic transmutation, the interaction of low- and high-energy radiation with matter, a variety of source and tally options, interactive graphics, and support for a variety of sequential and multiprocessing computer platforms. MCNP6 has several powerful variance-reduction techniques that enhance computational performance. This 250000-line Fortran 90 code has been parallelized, and works on platforms including single-processor personal computers, Sun workstations, Linux clusters, and supercomputers. MCNP6 has a world-wide user base consisting of approximately 10000 researchers, designers, and analysts whose applications encompass accelerator design, nuclear reactor physics, isotopic transmutation, cancer diagnostics and therapy, geophysics, active and passive interrogation involving prompt and delayed radiation, and space applications.

During MCNP6 calculations, the path, or "track," followed by each source particle to its termination is called a "history." A source particle history consists of its emission and tracking through the geometry, inclusive of its interaction with materials via cross-section sampling. Reactions resulting in the production of prompt and (if requested) delayed particles are treated. Information about each prompt and delayed particle is stored ("banked") until tracking of the source particle is complete. Once the source particle is terminated, each prompt and delayed particle that has been stored in the bank is retrieved and transported. The complete tracking of a source particle and all of the prompt and delayed particles created during its transport constitutes a history. Many source-particle histories must be executed to obtain representative samples for quantities of interest ("tallies"), such as flux, current, high-resolution delayed-gamma spectra, etc.

An MCNP6 model is constructed using geometric "cells" and "surfaces." The cells contain materials that are bounded by first- and second-degree surfaces and fourth-degree elliptical tori. The cells are defined by the intersection, union, and/or complement of the regions bounded by the surfaces. MCNP6 also provides a "macrobody" capability to specify closed volumes such as spheres, boxes, cylinders, etc.

In addition to basic cells and macrobodies, MCNP6 has a "repeated-structures" capability. The repeated-structures capability makes it possible to describe only once the cells and surfaces of any structure that appears more than once in a geometry. The user can specify that a cell is to be filled with a "universe." A universe consists of either a "lattice" or a user-defined collection of cells.[†] A single universe, described only once, can be designated to fill any number of cells in the geometry. In addition, the TRCL keyword makes it possible to define a cell once and then locate the cell at multiple places in the

[†] Lattices are not yet treated by the moving-object upgrade.

geometry. The TRCL keyword uses transformation rules listed on the TR card.[‡] In this paper we refer to cells and universes as "objects."

Until now, the treatment of objects in MCNP6 simulations was limited to static geometries. Consequently, the positions of objects defined in radiation-transport simulations were unchanged throughout a calculation. This limitation prohibited self-contained simulations that include dynamic objects. Previously, the effect of geometric motion had to be investigated using a series of independent calculations with different input files that contained stepwise adjustments of the geometric descriptions.

We have created a new MCNP6 capability that permits motion of one or more objects during a simulation. Rectilinear and circular curvilinear motion are treated. Objects can move with constant velocity, experience constant acceleration or deceleration, or can be relocated.

The MCNP6 moving-objects treatment is available for source-mode ("SDEF") execution. Historically MCNP6 has had appreciable capability to treat stationary radiation sources. The "SDEF" card has been the mechanism used to specify the energy, time, direction, location, and type of each source particle produced during an MCNP6 calculation. Included in the moving-objects feature are upgrades that permit SDEF source motion.

[‡] The TRCL keyword and TR card capabilities have been available in MCNP6 since about 1980 for models involving fixed geometry.

Secondary-particle emission plays an important role in a variety of applications involving fission and activation. For example, neutron-activation analysis (NAA) is a method for qualitative and quantitative determination of elements based on the measurement of characteristic radiation emitted by radionuclides created directly or indirectly by neutron irradiation of a material (Dodd, 2001). NAA is being used for active-interrogation techniques for explosives, chemical agents, and nuclear material is a very active area of research (Barzilov et al., 2009; Chichester and Seabury, 2009; Gozani, 2009; Maestas et al., 2009; Owen et al., 2009). NAA involving prompt (PGNAA) and delayed gammas (DGNAA) is employed in radioanalytical chemistry, environmental, health, medicine, and nutrition studies (Ayranov and Schumann, 2010; Fei et al., 2010; Frontasyeva et al., 2010; Shypailo and Ellis, 2004; Sengupta et al., 2011), and in geological and material science disciplines (Dasari et al., 2010), including nanomaterials (Barta et al., 2010). Earlier work described upgrades to MCNPX that permit high-resolution delayed-gamma analysis involving neutron (Durkee et al., 2009a) and photon (Durkee et al., 2009b) sources, including accelerator beams (Durkee et al., 2010). The moving-objects feature accommodates the treatment of secondary particles emitted by objects that are in motion.

To visualize object motion, the MCNP6 plot utility (Pelowitz, 2008) has been upgraded. The plot utility has been available since the early 1980’s as a means of plotting model geometry. Geometry visualization is a very useful aid for MCNP6 model (“inp” file) creation, and is virtually essently for the development of input files for models that involve complicated geometries. Model complexity is elevated with the advent of the moving-object capability, and much of the responsibility to ensure the integrity of a model lies with the user. The user must prepare models ensuring that moving objects do not collide with either static objects or other moving objects. This is also the case for conventional static-geometry models in which users must create models whose objects are properly distinct. We have added the capability to plot snapshots of the geometry as a function of time via the new plot “time” command. This animation feature enables the user to visually inspect moving-object models as they evolve in time and to ensure that the geometry is correct. Animation also produces illuminating graphics for documentation and presentations.

The moving-objects feature has several important limitations and constraints. The moving-objects feature is not available for kcode-mode (criticality) simulations. Object motion is characterized by and limited to isothermal rigid-body kinematics, i.e., the geometry of motion (Hibbeler, 1974). Kinetics is not treated, i.e., there is no allowance for the application of forces to objects (Hibbeler, 1974). There is no treatment involving mechanics of materials or thermomechanical phenomena. Thus, MCNP6 moving objects retain their initial shapes throughout a simulation. Object deformation involving surfaces that bend, twist, expand or contract, or otherwise alter their shape is not treated. Cross-sections and densities remain invariant during a simulation.

In the following section we review the geometry transformation capability in MCNP6 as a prelude to the discussion of transformation upgrades that have been added to MCNP6 to support moving objects. The geometry-transformation capability has been available since 1981 for static objects (Thompson et al., 1980; Thompson, 1981). The basic concepts that are used to develop the MCNP6 moving-objects geometry capabilities are discussed in Section 3. This discussion is followed in Sections 4–6 by derivational details for object motion. Sections 7 and 8 discuss moving sources and delayed particles, respectively. Additional sections review limitations, plot upgrades, and coding issues.

## 2. MCNP6 geometry transformation

MCNP6 uses quadratic and matrix representations of 3-D surfaces (Brown, 2003a, pp. 182–185). MCNP6 performs particle tracking and geometry plotting using surfaces that are represented in global coordinates ( $x^G, y^G, z^G$ ). Objects can be created using surfaces that are input using global coordinates. Alternatively, since MCNP 2B (Thompson, 1981) MCNP6 has offered the capability to input surfaces in local coordinates ( $x^L, y^L, z^L$ ) and then translate and/or rotate each surface to its global location and orientation. This transformation capability simplifies specification for objects whose orientations are not parallel to one of the global-coordinate axes.

The equation of a non-toroidal† surface in MCNP6 can be written as a general quadratic of the form (Spain, 2007; Tierney, 1974)

$$Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fzx + Gx + Hy + Jz + K = 0\,. \qquad (1)$$

In matrix form, Eq.(1) becomes

† Similar comments pertain to toroidal surfaces. Our focus here is on the transformation expressions. The moving objects upgrade treats the necessary transformation formulations, and applies to toroidal surfaces.

$$\begin{bmatrix}1 & x & y & x\end{bmatrix} AM \begin{bmatrix}1\\ x\\ y\\ z\end{bmatrix} = 0 \tag{2}$$

where the *AM* matrix is

$$AM = \begin{bmatrix} K & G/2 & H/2 & J/2 \\ G/2 & A & D/2 & F/2 \\ H/2 & D/2 & B & E/2 \\ J/2 & F/2 & E/2 & C \end{bmatrix}. \tag{3}$$

Equations (1)–(3) are valid for surfaces in local and global coordinates. Thus, the matrix representation for local coordinates is

$$\begin{bmatrix}1 & x^L & y^L & x^L\end{bmatrix} AM^L \begin{bmatrix}1\\ x^L\\ y^L\\ z^L\end{bmatrix} = 0, \tag{4}$$

while for global coordinates

$$\begin{bmatrix}1 & x^G & y^G & x^G\end{bmatrix} AM^G \begin{bmatrix}1\\ x^G\\ y^G\\ z^G\end{bmatrix} = 0. \tag{5}$$

The local- and global-coordinate representations of a surface are related. We may write this relationship via the transformation

$$\begin{bmatrix}1\\ x^L\\ y^L\\ z^L\end{bmatrix} = TRF \begin{bmatrix}1\\ x^G\\ y^G\\ z^G\end{bmatrix} \tag{6}$$

where *TRF* is the transformation matrix containing translation and rotation information. We will consider the contents of *TRF* shortly.

Taking the transpose of Eq.(6) gives

$$\begin{bmatrix} 1 & x^L & y^L & z^L \end{bmatrix} = \begin{bmatrix} 1 & x^G & y^G & z^G \end{bmatrix} TRF^T \, . \tag{7}$$

Then substituting Eqs.(6) and (7) in Eq.(4) leads to

$$\begin{bmatrix} 1 & x^G & y^G & x^G \end{bmatrix} TRF^T AM^L TRF \begin{bmatrix} 1 \\ x^G \\ y^G \\ z^G \end{bmatrix} = 0 \, , \tag{8}$$

which is the global-coordinate matrix form for a surface given its local-coordinate coefficients in the $AM^L$ matrix and the transformation matrix *TRF*. We may now express $AM^G$ as

$$AM^G = TRF^T AM^L TRF \, . \tag{9}$$

Now let us consider coordinate rotation involving the two rectangular coordinate systems shown in Fig.1. This configuration depicts coordinate rotation; coordinate translation is excluded because the origins of the two coordinates systems are identical.

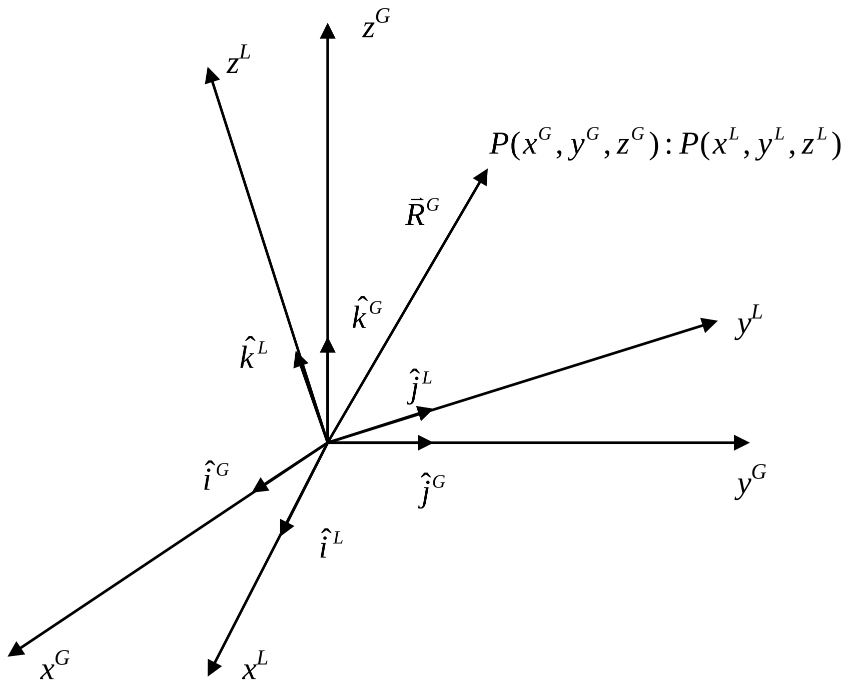

**Figure 1**. Coordinate-system rotation illustrated by global and local rectangular coordinate systems having the same origin.

The values of the local-coordinate variables $x^L, y^L, z^L$ can be obtained by projecting the position vector $\vec{R}^G$ onto the local-coordinate unit vectors $\hat{i}^L$, $\hat{j}^L$, and $\hat{k}^L$ (Wylie, 1975). Thus

$$\begin{aligned} x^L &= \Delta x^G + \hat{i}^G \cdot \hat{i}^L x^G + \hat{j}^G \cdot \hat{i}^L y^G + \hat{k}^G \cdot \hat{i}^L z^G = \Delta x^G + \vec{R}^G \cdot \hat{i}^L \\ y^L &= \Delta y^G + \hat{i}^G \cdot \hat{j}^L x^G + \hat{j}^G \cdot \hat{j}^L y^G + \hat{j}^G \cdot \hat{j}^L z^G = \Delta y^G + \vec{R}^G \cdot \hat{j}^L \, , \\ z^L &= \Delta z^G + \hat{i}^G \cdot \hat{k}^L x^G + \hat{j}^G \cdot \hat{k}^L y^G + \hat{k}^G \cdot \hat{k}^L z^G = \Delta z^G + \vec{R}^G \cdot \hat{k}^L \end{aligned} \qquad (10)$$

where

$$\vec{R}^G = x^G \hat{i}^G + y^G \hat{j}^G + z^G \hat{k}^G \, . \qquad (11)$$

If we define the dot products of the global- and local-coordinate unit vectors as[†]

$$\begin{aligned} B_1^{GL} &= \hat{i}^G \cdot \hat{i}^L, \quad B_2^{GL} = \hat{j}^G \cdot \hat{i}^L, \quad B_3^{GL} = \hat{k}^G \cdot \hat{i}^L \\ B_4^{GL} &= \hat{i}^G \cdot \hat{j}^L, \quad B_5^{GL} = \hat{j}^G \cdot \hat{j}^L, \quad B_6^{GL} = \hat{k}^G \cdot \hat{j}^L \\ B_7^{GL} &= \hat{i}^G \cdot \hat{k}^L, \quad B_8^{GL} = \hat{j}^G \cdot \hat{k}^L, \quad B_9^{GL} = \hat{k}^G \cdot \hat{k}^L \ , \end{aligned} \tag{12}$$

Eq.(10) becomes

$$\begin{aligned} x^L &= \Delta x^G + B_1^{GL} x^G + B_2^{GL} y^G + B_3^{GL} z^G \\ y^L &= \Delta y^G + B_4^{GL} x^G + B_5^{GL} y^G + B_6^{GL} z^G \\ z^L &= \Delta z^G + B_7^{GL} x^G + B_8^{GL} y^G + B_9^{GL} z^G \ . \end{aligned} \tag{13}$$

Consequently, the transformation matrix takes the form

$$TRF = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \Delta x^G & B_1^{GL} & B_2^{GL} & B_3^{GL} \\ \Delta y^G & B_4^{GL} & B_5^{GL} & B_6^{GL} \\ \Delta z^G & B_7^{GL} & B_8^{GL} & B_9^{GL} \end{bmatrix}. \tag{14}$$

In the following sections, we derive expressions for the translation and rotation components of Eq.(14) for motion involving rectilinear translation, curvilinear translation, and curvilinear rotation. As we shall see, the $TRF$ matrix will have time-dependent translation components. The $TRF$ matrix rotation components will be time-independent for rectilinear and curvilinear translation, but will be time-dependent for curvilinear rotation.

## 3. Moving objects dynamics model

The new MCNP6 moving-object capability can be applied to objects characterized by rigid-body kinematic motion. The dynamics concepts are familiar ones (Hibbeler,

[†] For MCNPX users, the "*B*'s" are the rotation-matrix values $xx'$, $yx'$, etc. that have been used since 1981 on the TR card to input an object's orientation in the global coordinate system.

1974). However, implementation in MCNP6 is complicated by the need to associate three coordinate systems and the transformations among these coordinate systems.

Our treatment considers three motion types: (1) rectilinear translation, (2) curvilinear translation, and (3) curvilinear rotation. For our purposes, translation means any straight line on an object remains fixed throughout the duration of motion. That is, an object's orientation is fixed as it moves. For rectilinear translation, the motion path of the object is along parallel straight lines as shown in Fig. 2. Note that the mid-bar in the object remains fixed in its orientation as the object moves along a straight path.

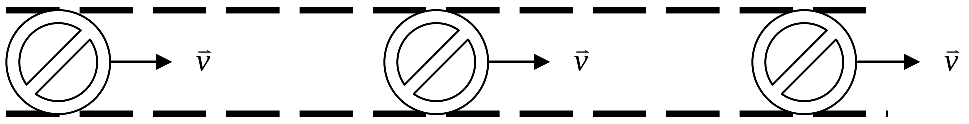

**Figure 2.** Rectilinear translation.

For curvilinear translation, illustrated in Fig. 3, the motion path of the object is along curved lines. As with rectilinear translation, the object does not vary in orientation as it moves.

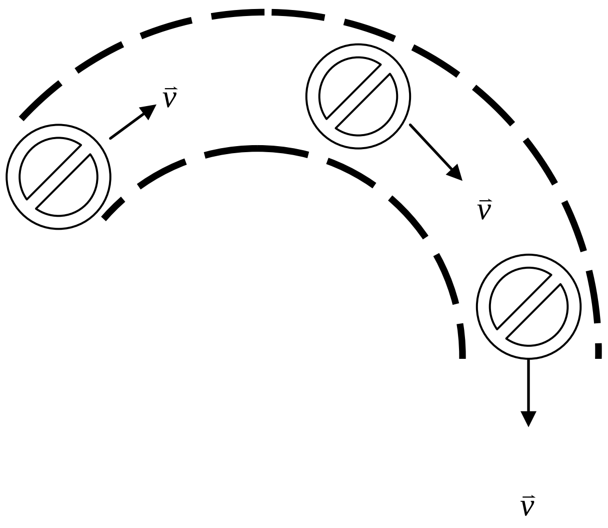

**Figure 3**. Curvilinear translation.

Curvilinear rotation is illustrated in Fig. 4. As with curvilinear translation, the motion path of the object is along curved lines. In addition, the object's orientation changes as the object moves along the curved path. This adjustment in orientation is indicated in the figure by the changing orientation of the object's mid-bar.

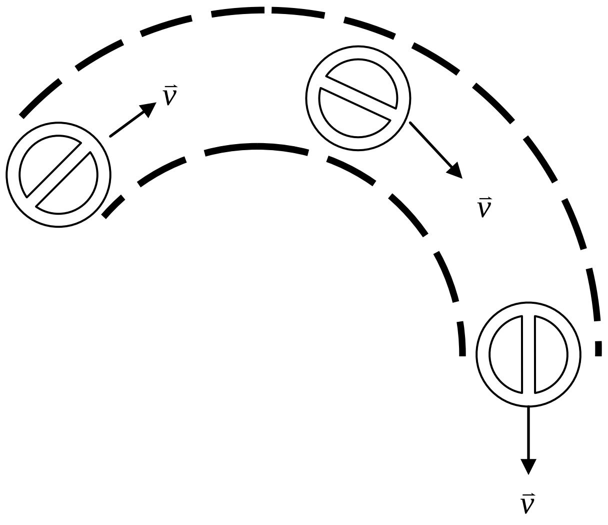

**Figure 4**. Curvilinear rotation.

Our goal is to derive the expressions that characterize an object's location and orientation as it moves for each of these three types of motion and to formulate the expressions in a format suitable for use by MCNP6. To proceed, we must make note of some overriding concepts and stipulations.

First, throughout MCNP6's development history, object shape and location, as well as particle tracking, have been characterized using 3-D Cartesian coordinates. We adhere to this convention for moving objects.

Second, we must consider three coordinate systems. The first coordinate system we call “MCNP6 global coordinates,” the second “object local coordinates,” and the third “object path coordinates.” MCNP6 global coordinates and object local coordinates have been used since 1981 in MCNP6 (Thompson, 1981). Object path coordinates are introduced with the moving-object feature.

MCNP6 uses a 3-D Cartesian global-coordinate system to locate objects and track particles. Using object local coordinates, an object can be conveniently created and then located and oriented. For example, a 2- by 3- by 4-cm rectangular box can be easily described using an object’s coordinate system. The box can then be moved to a desired location and oriented by prescribing the angles between the coordinate axes of the box and the MCNP6 global-coordinate axes. For complex models, this manner of prescribing an object’s size, location, and orientation is often much more convenient than doing a direct characterization in MCNP6 global coordinates.

Object path coordinates are introduced here as a way to characterize an object’s trajectory, or path, using dynamics expressions. Object path coordinates permit motion along a path whose orientation may differ from that of the object itself.

We need to develop relationships among the three coordinate systems. To add clarity, superscript notation “P”, “L”, and “G” will be used to explicitly designate “path”, “local”, and “global” coordinate systems, respectively.

Third, our implementation of the moving-objects feature requires that acceleration (or deceleration) be either zero or constant. The moving-object feature currently does not treat variable acceleration. This requirement renders our formulations analytic. In general, the formulations could be extended to permit variable acceleration at the expense of numerical integration.

Fourth, we assume object motion occurs on a timescale orders-of-magnitude slower than radiation transport. Thus, we do not account for motion effects during the transport of a particle. However, time-dependent effects involving delayed particles (e.g., delayed neutrons and photons) are treated  by moving the geometry to the time at which each delayed particle is emitted. The geometry is fixed during the transport of a source or delayed particle.

Fifth, the location and orientation of the object’s path is fixed (time invariant). That is, the object can move, but the path cannot.

Sixth, an object’s motion is described using one or more “motion segments.” Motion along a motion segment is either rectilinear translation, curvilinear translation, or curvilinear rotation. Individual motion segments can be connected to create a piecewise-continuous description of a more complicated motion sequence; e.g., rectilinear translation followed by curvilinear rotation followed by rectilinear translation.

Seventh, curvilinear motion is limited to circular paths. The axis of rotation is stationary for each circular path. A circular path is centered at the axis of rotation.

Eighth, curvilinear translation and rotation motion is limited to a plane. Thus, “corkscrew” motion is not permitted. Although the formulations readily permit non-planar motion, the coding needed to verify motion-segment linking has not been developed. The plane containing the curvilinear motion may be oriented arbitrarily in 3-D.

Nineth, the dynamics formulations are Newtonian.

These nine concepts and stipulations determine the form of the MCNP6 moving-objects feature. The next three sections contain the mathematical material that underpins the MCNP6 moving-object dynamics capability. The central objective of these sections is to develop the equations for an object’s position and orientation as a function of time in MCNP6 global coordinates. MCNP6 uses these equations to update the geometry to the time at which each prompt and delayed particle is emitted.

## 4. Rectilinear translation along a straight path

We begin by considering how an object’s location and orientation are specified at the start of a motion segment as shown in Fig. 5. An object’s location at the start of a motion segment is specified in global coordinates as $\vec{O}^G$, where

$$\vec{O}^G = O_x^G \hat{i}^G + O_y^G \hat{j}^G + O_z^G \hat{k}^G . \tag{15}$$

This expression is the translation vector for an object from object local coordinates to MCNP6 global coordinates.

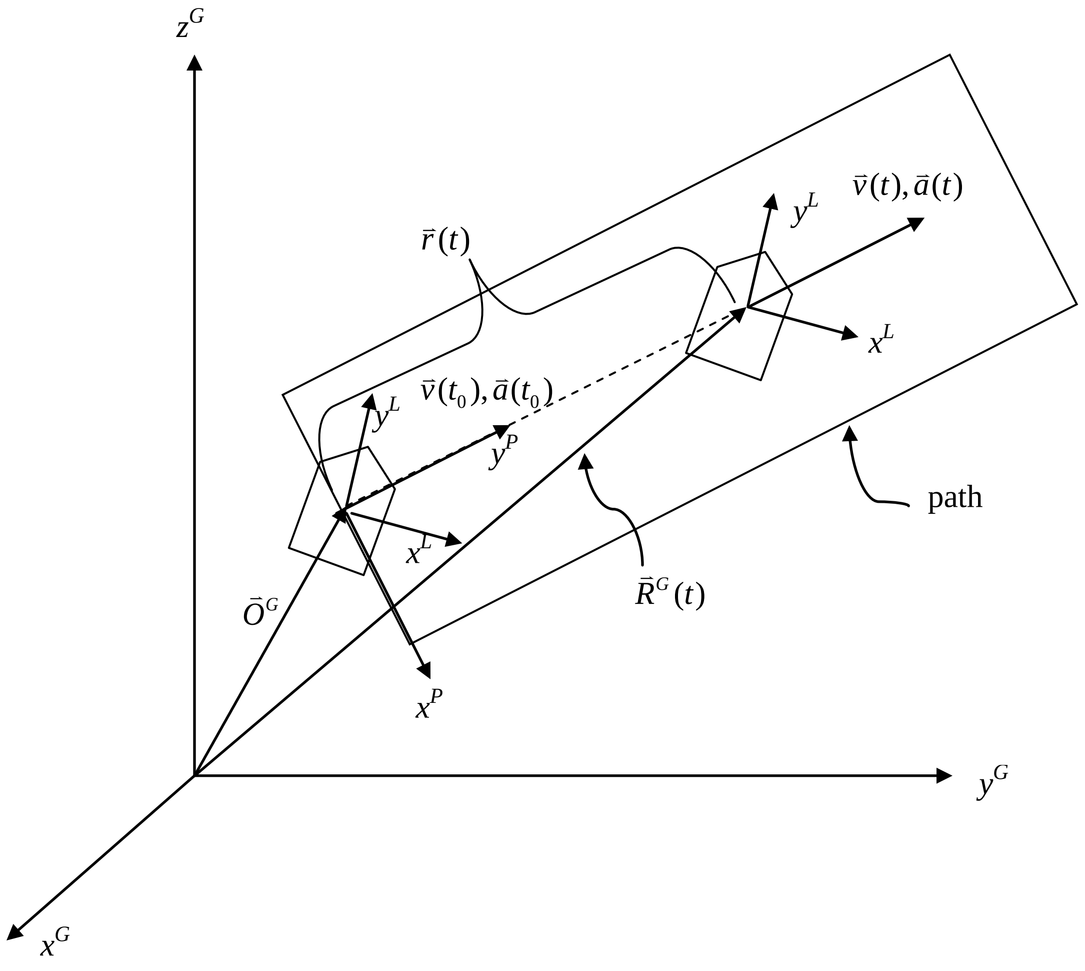

**Figure 5**. Locating and orienting an object and its path for rectilinear translation in MCNP6. At the start of the motion segment the object is located at $\vec{O}^G$ with its orientation specified in terms of the angles between its local-coordinate system and the MCNP6 global-coordinate axes. The object's path is prescribed using path coordinates ($x^P$, $y^P$), with motion in the $y^P$ direction for velocity $\vec{v}(t)$ and acceleration $\vec{a}(t)$. The path orientation is specified in terms of the angles between its path-coordinate system and the MCNP6 global-coordinate axes.

The object's orientation is invariant throughout the duration of motion as it traverses a rectilinear motion segment. Thus, the angles between its local-coordinate axes and the MCNP6 global-coordinate axes do not change. Consequently, the object's orientation parameters are given by Eq.(12), repeated here for convenience:

$$\begin{aligned} &B_1^{GL} = \hat{i}^G \cdot \hat{i}^L, \quad B_2^{GL} = \hat{j}^G \cdot \hat{i}^L, \quad B_3^{GL} = \hat{k}^G \cdot \hat{i}^L \\ &B_4^{GL} = \hat{i}^G \cdot \hat{j}^L, \quad B_5^{GL} = \hat{j}^G \cdot \hat{j}^L, \quad B_6^{GL} = \hat{k}^G \cdot \hat{j}^L \\ &B_7^{GL} = \hat{i}^G \cdot \hat{k}^L, \quad B_8^{GL} = \hat{j}^G \cdot \hat{k}^L, \quad B_9^{GL} = \hat{k}^G \cdot \hat{k}^L . \end{aligned} \tag{16}$$

As the object moves along a path, its location is given by the position vector $\vec{R}^G(t)$, where

$$\vec{R}^G(t) = \vec{O}^G + \vec{r}^{\,G}(t), \tag{17}$$

and $\vec{r}^{\,G}(t)$ is the global-coordinate representation of the object's position vector along the path from $\vec{O}^G$. Because MCNP6 uses the Cartesian coordinate representation of $\vec{R}^G(t)$ to locate objects, we rewrite the form of $\vec{R}^G(t)$ as

$$\vec{R}^G(t) = R_x^G(t)\hat{i}^G + R_y^G(t)\hat{j}^G + R_z^G(t)\hat{k}^G . \tag{18}$$

The rate-of-change of the position vector is the instantaneous velocity whose vector is

$$\vec{v}^{\,G}(t) = v_x^G(t)\hat{i}^G + v_y^G(t)\hat{j}^G + v_z^G(t)\hat{k}^G \tag{19}$$

for which the scalar components are

$$v_\xi^G = \frac{dr_\xi^G}{dt}, \xi = x, y, z . \tag{20}$$

Similarly, an object's acceleration vector is given by the rate of change of the velocity vector

$$\vec{a}^G(t) = a_x^G(t)\hat{i}^G + a_y^G(t)\hat{j}^G + a_z^G(t)\hat{k}^G \tag{21}$$

with scalar components

$$a_\xi^G = \frac{dv_\xi^G}{dt}, \xi = x, y, z\,. \tag{22}$$

For rectilinear motion, the object's velocity and acceleration vectors are tangent to its path.

As we have mentioned, it can be cumbersome to describe the surfaces of an arbitrarily positioned and oriented object. Consequently, in 1981 capability was added to MCNP that enables the user to characterize an object's surfaces in its local coordinate system. The object's surfaces then can be translated and rotated as desired. The same concept is implemented here for an object's path. Next we formulate the dynamics equations in path coordinates and then develop equations relating the path-coordinate equations to MCNP6 global coordinates.

The position vector of the object using path coordinates using the Cartesian form is given by

$$\vec{r}^P(t) = r_x^P(t)\hat{i}^P + r_y^P(t)\hat{j}^P + r_z^P(t)\hat{k}^P\,. \tag{23}$$

The object's velocity and acceleration are

$$\begin{aligned}\vec{v}^P(t) &= v_x^P(t)\hat{i}^P + v_y^P(t)\hat{j}^P + v_z^P(t)\hat{k}^P \\ \vec{a}^P(t) &= a_x^P(t)\hat{i}^P + a_y^P(t)\hat{j}^P + a_z^P(t)\hat{k}^P\end{aligned} \tag{24}$$

with the scalar components

$$\begin{aligned} v_\xi^P &= \frac{dr_\xi^P}{dt} \\ a_\xi^P &= \frac{dv_\xi^P}{dt} \quad . \end{aligned} \tag{25}$$

Stipulation 3 of Section 3 states that we treat only constant (or zero) acceleration. Consequently, integration of the expressions in Eq.(25) from time $t_0$ to $t$ results in the following scalar components of position and velocity for the path:

$$\begin{aligned} r_\xi^P(t) &= r_{\xi 0}^P + v_{\xi 0}^P\left(t-t_0\right) + \frac{1}{2}a_\xi^P\left(t-t_0\right)^2 \\ v_\xi^P(t) &= v_{\xi 0}^P + a_\xi^P\left(t-t_0\right), \end{aligned} \tag{26}$$

where $r_{\xi 0}$ (cm) and $v_{\xi 0}$ (cm/s) are the location and speed at time $t = t_0$ , $a_\xi^P$ are the constant Cartesian components of the acceleration, and $\xi = x, y, z$ .[†]

To relate the path-coordinate motion expressions in Eqs.(23)–(26) to MCNP6 global coordinates, we note that the object's location on the path relative to $\vec{O}^G$ , $\vec{r}(t)$, can be expressed in terms of MCNP6 global coordinates or path coordinates as

$$\begin{aligned} \vec{r}(t) &= r_x^G(t)\hat{i}^G + r_y^G(t)\hat{j}^G + r_z^G(t)\hat{k}^G \equiv \vec{r}^G(t) \\ &= r_x^P(t)\hat{i}^P + r_y^P(t)\hat{j}^P + r_z^P(t)\hat{k}^P \equiv \vec{r}^P(t) \end{aligned} \tag{27}$$

In order to express $\vec{r}(t)$ in MCNP6 global coordinates, we project the path-coordinate vector $\vec{r}^P(t)$ onto the MCNP6 global-coordinate axes to give (Wylie, 1975)

$$\begin{aligned} r_x^G(t) &= \vec{r}^P(t)\cdot\hat{i}^G = r_x^P(t)\hat{i}^P\cdot\hat{i}^G + r_y^P(t)\hat{j}^P\cdot\hat{i}^G + r_z^P(t)\hat{k}^P\cdot\hat{i}^G \\ r_y^G(t) &= \vec{r}^P(t)\cdot\hat{j}^G = r_x^P(t)\hat{i}^P\cdot\hat{j}^G + r_y^P(t)\hat{j}^P\cdot\hat{j}^G + r_z^P(t)\hat{k}^P\cdot\hat{j}^G \\ r_z^G(t) &= \vec{r}^P(t)\cdot\hat{k}^G = r_x^P(t)\hat{i}^P\cdot\hat{k}^G + r_y^P(t)\hat{j}^P\cdot\hat{k}^G + r_z^P(t)\hat{k}^P\cdot\hat{k}^G \quad . \end{aligned} \tag{28}$$

[†] For MCNP6 users, the unit of time for input values is shakes.

The dot product terms of the unit vectors in the path and global coordinates can be written as $B_1^{GP} - B_9^{GP}$, with

$$\begin{aligned}
B_1^{GP} &= \hat{i}^P \cdot \hat{i}^G, \quad B_2^{GP} = \hat{j}^P \cdot \hat{i}^G, \quad B_3^{GP} = \hat{k}^P \cdot \hat{i}^G \\
B_4^{GP} &= \hat{i}^P \cdot \hat{j}^G, \quad B_5^{GP} = \hat{j}^P \cdot \hat{j}^G, \quad B_6^{GP} = \hat{k}^P \cdot \hat{j}^G \\
B_7^{GP} &= \hat{i}^P \cdot \hat{k}^G, \quad B_8^{GP} = \hat{j}^P \cdot \hat{k}^G, \quad B_9^{GP} = \hat{k}^P \cdot \hat{k}^G \quad .
\end{aligned} \tag{29}$$

Because the orientation of the path is fixed (time-invariant) per Stipulation 5 of Section 3, the $B_1^{GP} - B_9^{GP}$ are constant. For MCNP6 simulations, the $B_1^{GP} - B_9^{GP}$ are input by the user; that is, the user specifies the orientation of the object's path.†

Using Eqs.(15), (17), (23), (28) and (29), an object's location can be determined in MCNP6 global coordinates. The global position variables in MCNP6 are given in terms of $\vec{R}^G(t)$ from Eq.(18)

$$\vec{R}^G(t) = xxx(t)\hat{i}^G + yyy(t)\hat{j}^G + zzz(t)\hat{k}^G , \tag{30}$$

where $xxx(t)$, $yyy(t)$, and $zzz(t)$ are the MCNP6 global-coordinate position variables

$$\begin{aligned}
xxx(t) &= R_x^G(t) = O_x^G + r_x^G(t), \\
yyy(t) &= R_y^G(t) = O_y^G + r_y^G(t), \\
zzz(t) &= R_z^G(t) = O_z^G + r_z^G(t).
\end{aligned} \tag{31}$$

## 5. Curvilinear translation along a circular path

An object undergoing curvilinear translation moves along a circular path that is centered about its axis-of-rotation. Here we consider object motion that is either constant

† This convention follows that used since 1981 for input of the object's orientation parameters $B_1^{GL} - B_9^{GL}$.

velocity or changing velocity as a result of constant acceleration or deceleration. As it moves, the object's orientation remains fixed. Its orientation can be different than the orientation of the path. In MCNP6 nomenclature this means that $B_1^{GL} - B_9^{GL}$, the object's orientation parameters, and $B_1^{GP} - B_9^{GP}$, the path's orientation parameters, are constant, and that the sets of values for the $B^{GL}$ and $B^{GP}$ can be different. This concept is illustrated in Fig. 6.

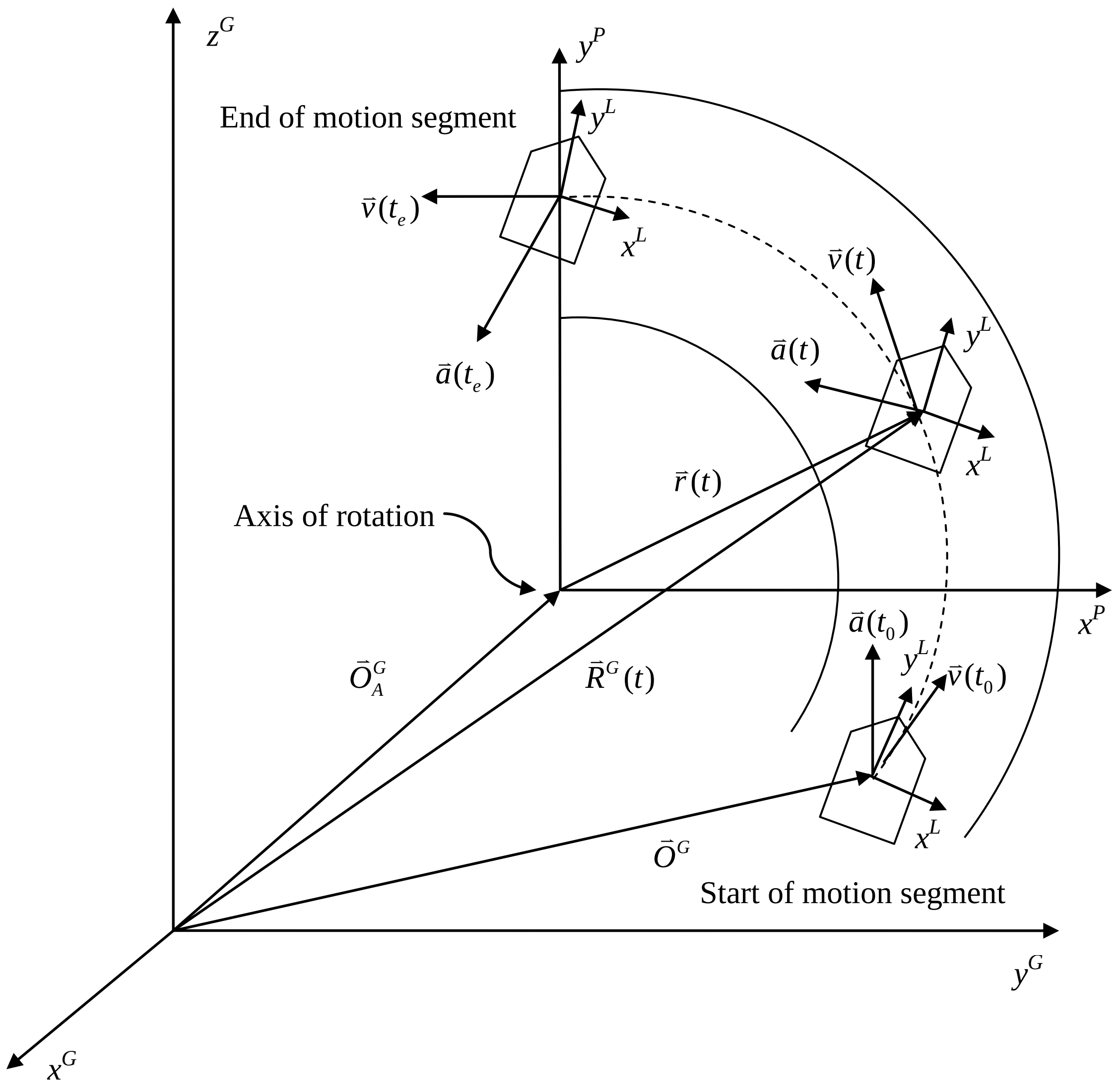

**Figure 6**. Curvilinear translation of an object about a fixed axis. The object's orientation remains fixed as it location changes from time $t_0$ to $t_e$. The orientation of the object and path are specified separately and can differ from each other, but both orientations are fixed in time.

We seek expressions relating the object's location as described by path coordinates to MCNP6 global coordinates. We begin by considering three position vectors for the 1) location of the object's axis-of-rotation, 2) the object's location on its path relative to its axis-of-rotation, and 3) its location relative to the MCNP6 global-coordinate origin.

As was the case for rectilinear translation, an object's location at the start of a motion segment is specified in global coordinates as $\vec{O}^G$, where from Fig. 6

$$\vec{O}^G = O_x^G \hat{i}^G + O_y^G \hat{j}^G + O_z^G \hat{k}^G . \tag{32}$$

For rectilinear translation, it was convenient to relate the object's motion to $\vec{O}^G$. For curvilinear translation, this convention could also be used. However, it is more intuitive to relate the object's motion to its axis of rotation. We introduce the vector $\vec{O}_A^G$ to connote the location of the object's axis of rotation as given in MCNP6 global coordinates, where

$$\vec{O}_A^G = O_{Ax}^G \hat{i}^G + O_{Ay}^G \hat{j}^G + O_{Az}^G \hat{k}^G . \tag{33}$$

Stipulation 7 of Section 3 states that the location of the axis-of-rotation $\vec{O}_A^G$ is fixed as the object moves.

The object's location relative to the MCNP6 global-coordinate origin is given by the vector sum of the vector locations of the object's axis of rotation and its location on the path. Thus,

$$\vec{R}^G(t) = \vec{O}_A^G + \vec{r}^G(t) , \tag{34}$$

where $\vec{r}^{\,G}(t)$ is the global-coordinate representation of the object's position vector along the path from $\vec{O}_A^G$. Despite the object moving with curvilinear motion, we express its location using Cartesian coordinates:

$$\vec{R}^G(t) = R_x^G(t)\hat{i}^G + R_y^G(t)\hat{j}^G + R_z^G(t)\hat{k}^G\,. \tag{35}$$

The object's instantaneous velocity and acceleration vectors are

$$\begin{aligned}\vec{v}^G(t) &= v_x^G(t)\hat{i}^G + v_y^G(t)\hat{j}^G + v_z^G(t)\hat{k}^G\\ \vec{a}^G(t) &= a_x^G(t)\hat{i}^G + a_y^G(t)\hat{j}^G + a_z^G(t)\hat{k}^G\end{aligned} \tag{36}$$

with the scalar components

$$\begin{aligned}v_\xi^G &= \frac{dr_\xi^G}{dt},\ \xi = x, y, z\\ a_\xi^G &= \frac{dv_\xi^G}{dt},\ \xi = x, y, z\,.\end{aligned} \tag{37}$$

The direction of the object's velocity vector is tangent to its path of motion. However, in general the acceleration vector is not directed along the tangent line to its path.

Next we develop the dynamics equations in path coordinates and then create equations relating the path-coordinate equations to MCNP6 global coordinates. We write the position vector of the object using path coordinates using the Cartesian form. In the path coordinate system, we stipulate that the $x^P$-axis coincides with the starting location for the object in the current motion segment. The object's motion along the circular path is given by the Cartesian coordinate vector

$$\vec{r}^{\,P}(t) = r_x^P(t)\hat{i}^P + r_y^P(t)\hat{j}^P + r_z^P(t)\hat{k}^P\,. \tag{38}$$

Because the object's motion in path coordinates is circular, it is convenient to write the scalar components of the object's position, velocity, and acceleration vectors using cylindrical coordinates. We can thereby simply prescribe a radius-of-curvature, a starting location, an angle through which the object moves on the path, and angular speed and acceleration values (Hibbeler, 1974, pp. 504–506). The magnitudes of the angular velocity and angular acceleration are given by the relationships

$$\begin{aligned} \omega &= \frac{d\theta}{dt} \\ \alpha &= \frac{d\omega}{dt} \quad . \end{aligned} \tag{39}$$

Stipulation 3 of Section 3 requires that the magnitude of the angular acceleration be constant. Thus, integration of the expressions in Eq.(39) gives the angular speed and location

$$\begin{aligned} \omega(t) &= \omega_0 + \alpha\left(t - t_0\right) \\ \theta(t) &= \theta_0 + \omega_0\left(t - t_0\right) + \frac{1}{2}\alpha\left(t - t_0\right)^2 , \end{aligned} \tag{40}$$

where $t_0$ (s) and $\omega_0$ (radians/s) are the time and angular velocity at the beginning of the motion segment and $\alpha$ (radians/s$^2$) is the constant angular acceleration.[†]

The scalar components of the path-coordinate position vector are given by

$$\begin{aligned} r_x^P(t) &= \rho\cos\theta \\ r_y^P(t) &= \rho\sin\theta \\ r_z^P(t) &= 0 . \end{aligned} \tag{41}$$

[†] For MCNP6 users, the unit of time for input values is shakes.

The condition for $r_z^P(t)$ in Eq.(41) connotes in-plane motion per Stipulation 8 of Section 3. The radius of curvature $\rho$ is given by

$$\rho = \left|\vec{r}^{\,P}(t)\right| \tag{42}$$

and it is constant per Stipulation 7 of Section 3.

The object's location on the path relative to its axis of rotation is given by the vector $\vec{r}(t)$. As with rectilinear translation, $\vec{r}(t)$ can be expressed in terms of MCNP6 global coordinates or path coordinates as

$$\begin{aligned}\vec{r}(t) &= r_x^G(t)\hat{i}^G + r_y^G(t)\hat{j}^G + r_z^G(t)\hat{k}^G \equiv \vec{r}^{\,G}(t) \\ &= r_x^P(t)\hat{i}^P + r_y^P(t)\hat{j}^P + r_z^P(t)\hat{k}^P \equiv \vec{r}^{\,P}(t)\,.\end{aligned} \tag{43}$$

In order to express $\vec{r}(t)$ in MCNP6 global coordinates, the path-coordinate form of this vector $\vec{r}^{\,P}(t)$ is projected onto the MCNP6 global-coordinate axes to give

$$\begin{aligned}r_x^G(t) &= \vec{r}^{\,P}(t)\cdot\hat{i}^G = r_x^P(t)B_1^{GP} + r_y^P(t)B_4^{GP} + r_z^P(t)B_7^{GP} \\ r_y^G(t) &= \vec{r}^{\,P}(t)\cdot\hat{j}^G = r_x^P(t)B_2^{GP} + r_y^P(t)B_5^{GP} + r_z^P(t)B_8^{GP} \\ r_z^G(t) &= \vec{r}^{\,P}(t)\cdot\hat{k}^G = r_x^P(t)B_3^{GP} + r_y^P(t)B_6^{GP} + r_z^P(t)B_9^{GP}\ ,\end{aligned} \tag{44}$$

where the time-invariant path orientation parameters $B_1^{GP} - B_9^{GP}$ are defined in Eq.(29).

Using Eqs.(33), (34), (38), (40), (41), and (44), the object's location can be determined in MCNP6 global coordinates as

$$\vec{R}^G(t) = xxx\hat{i}^G + yyy\hat{j}^G + zzz\hat{k}^G\,, \tag{45}$$

where $xxx$, $yyy$, and $zzz$ are the MCNP6 global-coordinate position variables

$$\begin{aligned} xxx(t) &= R_x^G(t) = O_{Ax}^G + r_x^G(t), \\ yyy(t) &= R_y^G(t) = O_{Ay}^G + r_y^G(t), \\ zzz(t) &= R_z^G(t) = O_{Az}^G + r_z^G(t). \end{aligned} \tag{46}$$

We thus see that the analysis of curvilinear translation involves only the determination of the object's time-dependent location. The object's orientation is invariant throughout the duration of its transit along the motion segment.

## 6. Curvilinear rotation along a circular path

An object undergoing curvilinear rotation will experience a change in both location and orientation as illustrated in Fig. 7. The object's location is determined using the same expressions used for curvilinear translation, Eqs.(33)–(46). As for rectilinear and curvilinear translation, the path's orientation is fixed so that $B_1^{GP} - B_9^{GP}$ are constant as given by Eq.(29).

Additional expressions are required to determine the object's orientation because it changes continuously with time as the object moves along the path of a curvilinear rotation motion segment. We next derive expressions for the object's time-dependent orientation parameters $B_1^{GL}(t) - B_9^{GL}(t)$.

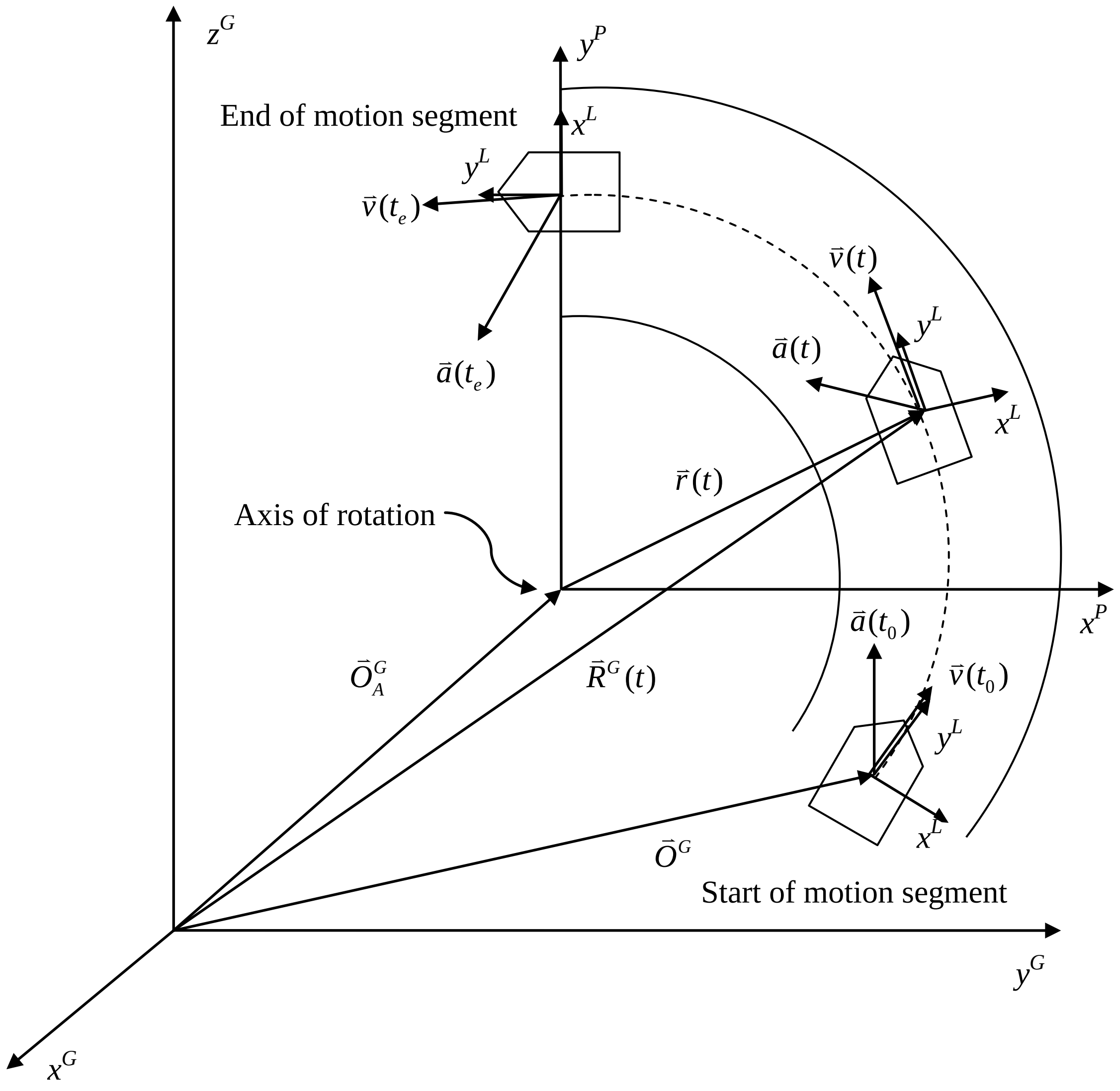

**Figure 7**. Object curvilinear rotation about a fixed axis. The object's location and orientation changes from time $t_0$ to $t_e$. The orientation of the object and path are specified separately and can differ from each other. The object's orientation changes as the object moves. The path orientation is fixed in time.

*6.1. Formulation of the object's orientation for curvilinear rotation.*

For curvilinear rotation, the object's orientation changes continuously as a function of time. To characterize this behavior, relationships between the MCNP6 global-coordinate axes and the object's local-coordinate axes must be formulated. That is, expressions for $B_1^{GL}(t) - B_9^{GL}(t)$ must be developed.

From Eq.(12), it is apparent that the $B_1^{GL}(t) - B_9^{GL}(t)$ contain dot products of the MCNP6 global-coordinate axes vectors and the object's local-coordinate vectors. We thus seek to form expressions that have the dot products consisting of the assortment of vectors in our problem onto the object's local-coordinate vectors. Viewing Fig. 7, we see the location vectors $\vec{R}^G(t)$, $\vec{r}^{\,G}(t)$, and $\vec{O}_{AOR}^G$ as well as the velocity and acceleration vectors $\vec{v}^{\,G}(t)$ and $\vec{a}^{\,G}(t)$. Let us consider the projection of these vectors onto the object's local-coordinate vectors.

We first project the object's position vector expressed in MCNP6 global coordinates onto the local-coordinate axes so that

$$\begin{aligned}
R_x^L(t) &= \vec{R}^G(t)\cdot\hat{i}^L = R_x^G(t)\hat{i}^G\cdot\hat{i}^L + R_y^G(t)\hat{j}^G\cdot\hat{i}^L + R_z^G(t)\hat{k}^G\cdot\hat{i}^L = \left|\vec{R}^G(t)\right|\mu_{R\hat{i}}(t)\\
R_y^L(t) &= \vec{R}^G(t)\cdot\hat{j}^L = R_x^G(t)\hat{i}^G\cdot\hat{j}^L + R_y^G(t)\hat{j}^G\cdot\hat{j}^L + R_z^G(t)\hat{k}^G\cdot\hat{j}^L = \left|\vec{R}^G(t)\right|\mu_{R\hat{j}}(t) \qquad (47)\\
R_z^L(t) &= \vec{R}^G(t)\cdot\hat{k}^L = R_x^G(t)\hat{i}^G\cdot\hat{k}^L + R_y^G(t)\hat{j}^G\cdot\hat{k}^L + R_z^G(t)\hat{k}^G\cdot\hat{k}^L = \left|\vec{R}^G(t)\right|\mu_{R\hat{k}}(t)\,,
\end{aligned}$$

where

$$\vec{R}^G(t) = R_x^G(t)\hat{i}^G + R_y^G(t)\hat{j}^G + R_z^G(t)\hat{k}^G\,, \qquad (48)$$

and the components $R_x^G(t)$, $R_y^G(t)$, and $R_z^G(t)$ are given by Eqs.(33), (34), and (44). The parameters $\mu_{R\hat{i}}(t)$, $\mu_{R\hat{j}}(t)$, and $\mu_{R\hat{k}}(t)$ are the cosines of the angles between $\vec{R}^G(t)$ and the local-coordinates axes. The dot products of the global- and local-coordinate unit vectors for curvilinear rotation are time dependent, with $B_1^{GL}(t) - B_9^{GL}(t)$ given by

$$\begin{aligned}
&B_1^{GL}(t) = \hat{i}^G \cdot \hat{i}^L, \quad B_2^{GL}(t) = \hat{j}^G \cdot \hat{i}^L, \quad B_3^{GL}(t) = \hat{k}^G \cdot \hat{i}^L \\
&B_4^{GL}(t) = \hat{i}^G \cdot \hat{j}^L, \quad B_5^{GL}(t) = \hat{j}^G \cdot \hat{j}^L, \quad B_6^{GL}(t) = \hat{k}^G \cdot \hat{j}^L \\
&B_7^{GL}(t) = \hat{i}^G \cdot \hat{k}^L, \quad B_8^{GL}(t) = \hat{j}^G \cdot \hat{k}^L, \quad B_9^{GL}(t) = \hat{k}^G \cdot \hat{k}^L \ .
\end{aligned} \tag{49}$$

Equation (47) contains three equations and 12 unknowns: $B_1^{GL}(t) - B_9^{GL}(t)$ and $\mu_{R\hat{i}}(t)$, $\mu_{R\hat{j}}(t)$, and $\mu_{R\hat{k}}(t)$. The latter three parameters are unknown at times other than the start of a motion segment. Because we have fewer equations than unknowns, additional relationships to that given in Eq. (47) must be formulated in order to obtain a unique solution for the unknowns.

Relationship two is developed by projecting $\vec{r}^G(t)$ onto the local-coordinate axes to give

$$\begin{aligned}
r_x^L(t) &= \vec{r}^G(t)\cdot\hat{i}^L = r_x^G(t)\hat{i}^G\cdot\hat{i}^L + r_y^G(t)\hat{j}^G\cdot\hat{i}^L + r_z^G(t)\hat{k}^G\cdot\hat{i}^L = \left|\vec{r}^G(t)\right|\mu_{r\hat{i}} \\
r_y^L(t) &= \vec{r}^G(t)\cdot\hat{j}^L = r_x^G(t)\hat{i}^G\cdot\hat{j}^L + r_y^G(t)\hat{j}^G\cdot\hat{j}^L + r_z^G(t)\hat{k}^G\cdot\hat{j}^L = \left|\vec{r}^G(t)\right|\mu_{r\hat{j}} \\
r_z^L(t) &= \vec{r}^G(t)\cdot\hat{k}^L = r_x^G(t)\hat{i}^G\cdot\hat{k}^L + r_y^G(t)\hat{j}^G\cdot\hat{k}^L + r_z^G(t)\hat{k}^G\cdot\hat{k}^L = \left|\vec{r}^G(t)\right|\mu_{r\hat{k}}
\end{aligned} \tag{50}$$

where we note that $\left|\vec{r}^G(t)\right|$ is constant because the path is circular and centered at the axis of rotation per Stipulation 7 of Section 3. On inspection of Fig. 7, we note that the angles between $\vec{r}^G(t)$ and the local-coordinate axes are fixed during rotation. This condition serves as a constraint on the orientation of the object. Thus, $\mu_{r\hat{i}}$, $\mu_{r\hat{j}}$, and $\mu_{r\hat{k}}$ can be determined at the beginning of the time segment and used at later times. Equations (47) and (50) thus give us 6 equations and 12 unknowns.

The third relationship is provided by projecting $\vec{O}_A^G$ onto the local-coordinate axes to give

$$\begin{aligned}
O_{Ax}^L(t) &= \vec{O}_A^G \cdot \hat{i}^L = O_{Ax}^G \hat{i}^G \cdot \hat{i}^L + O_{Ay}^G \hat{j}^G \cdot \hat{i}^L + O_{Az}^G \hat{k}^G \cdot \hat{i}^L = \left|\vec{O}_A^G\right| \mu_{O\hat{i}}(t) \\
O_{Ay}^L(t) &= \vec{O}_A^G \cdot \hat{j}^L = O_{Ax}^G \hat{i}^G \cdot \hat{j}^L + O_{Ay}^G \hat{j}^G \cdot \hat{j}^L + O_{Az}^G \hat{k}^G \cdot \hat{j}^L = \left|\vec{O}_A^G\right| \mu_{O\hat{j}}(t) \\
O_{Az}^L(t) &= \vec{O}_A^G \cdot \hat{k}^L = O_{Ax}^G \hat{i}^G \cdot \hat{k}^L + O_{Ay}^G \hat{j}^G \cdot \hat{k}^L + O_{Az}^G \hat{k}^G \cdot \hat{k}^L = \left|\vec{O}_A^G\right| \mu_{O\hat{k}}(t) \quad .
\end{aligned} \tag{51}$$

Although $\vec{O}_A^G$ is not time dependent, the angles between $\vec{O}_A^G$ and the object's coordinate axes vary with time. Thus, $\mu_{O\hat{i}}(t)$, $\mu_{O\hat{j}}(t)$, and $\mu_{O\hat{k}}(t)$ are not known as the object moves along its path. Equations (47), (50), and (51) contain 9 equations and 15 unknowns.

Two more sets of expressions are developed by making use of the object's velocity and acceleration vectors. First, the global-coordinate velocity and acceleration components are given by projecting the path-coordinate vectors onto the global-coordinate axes so that

$$\begin{aligned}
v_x^G(t) &= \vec{v}^P(t) \cdot \hat{i}^G = v_x^P(t)\hat{i}^P \cdot \hat{i}^G + v_y^P(t)\hat{j}^P \cdot \hat{i}^G + v_z^P(t)\hat{k}^P \cdot \hat{i}^G \\
v_y^G(t) &= \vec{v}^P(t) \cdot \hat{j}^G = v_x^P(t)\hat{i}^P \cdot \hat{j}^G + v_y^P(t)\hat{j}^P \cdot \hat{j}^G + v_z^P(t)\hat{k}^P \cdot \hat{j}^G \\
v_z^G(t) &= \vec{v}^P(t) \cdot \hat{k}^G = v_x^P(t)\hat{i}^P \cdot \hat{k}^G + v_y^P(t)\hat{j}^P \cdot \hat{k}^G + v_z^P(t)\hat{k}^P \cdot \hat{k}^G
\end{aligned} \tag{52}$$

and

$$\begin{aligned}
a_x^G(t) &= \vec{a}^P(t) \cdot \hat{i}^G = a_x^P(t)\hat{i}^P \cdot \hat{i}^G + a_y^P(t)\hat{j}^P \cdot \hat{i}^G + a_z^P(t)\hat{k}^P \cdot \hat{i}^G \\
a_y^G(t) &= \vec{a}^P(t) \cdot \hat{j}^G = a_x^P(t)\hat{i}^P \cdot \hat{j}^G + a_y^P(t)\hat{j}^P \cdot \hat{j}^G + a_z^P(t)\hat{k}^P \cdot \hat{j}^G \\
a_z^G(t) &= \vec{a}^P(t) \cdot \hat{k}^G = a_x^P(t)\hat{i}^P \cdot \hat{k}^G + a_y^P(t)\hat{j}^P \cdot \hat{k}^G + a_z^P(t)\hat{k}^P \cdot \hat{k}^G .
\end{aligned} \tag{53}$$

Next, the global-coordinate velocity and acceleration vectors are projected onto the local-coordinate axes to give

$$\begin{aligned} v_x^L(t) &= \vec{v}^G(t)\cdot\hat{i}^L = v_x^G(t)\hat{i}^G\cdot\hat{i}^L + v_y^G(t)\hat{j}^G\cdot\hat{i}^L + v_z^G(t)\hat{k}^G\cdot\hat{i}^L = \left|\vec{v}^G(t)\right|\mu_{v\hat{i}} \\ v_y^L(t) &= \vec{v}^G(t)\cdot\hat{j}^L = v_x^G(t)\hat{i}^G\cdot\hat{j}^L + v_y^G(t)\hat{j}^G\cdot\hat{j}^L + v_z^G(t)\hat{k}^G\cdot\hat{j}^L = \left|\vec{v}^G(t)\right|\mu_{v\hat{j}} \\ v_z^L(t) &= \vec{v}^G(t)\cdot\hat{k}^L = v_x^G(t)\hat{i}^G\cdot\hat{k}^L + v_y^G(t)\hat{j}^G\cdot\hat{k}^L + v_z^G(t)\hat{k}^G\cdot\hat{k}^L = \left|\vec{v}^G(t)\right|\mu_{v\hat{k}} \end{aligned} \tag{54}$$

and

$$\begin{aligned} a_x^L(t) &= \vec{a}^G(t)\cdot\hat{i}^L = a_x^G(t)\hat{i}^G\cdot\hat{i}^L + a_y^G(t)\hat{j}^G\cdot\hat{i}^L + a_z^G(t)\hat{k}^G\cdot\hat{i}^L = \left|\vec{a}^G(t)\right|\mu_{a\hat{i}}(t) \\ a_y^L(t) &= \vec{a}^G(t)\cdot\hat{j}^L = a_x^G(t)\hat{i}^G\cdot\hat{j}^L + a_y^G(t)\hat{j}^G\cdot\hat{j}^L + a_z^G(t)\hat{k}^G\cdot\hat{j}^L = \left|\vec{a}^G(t)\right|\mu_{a\hat{j}}(t) \\ a_z^L(t) &= \vec{a}^G(t)\cdot\hat{k}^L = a_x^G(t)\hat{i}^G\cdot\hat{k}^L + a_y^G(t)\hat{j}^G\cdot\hat{k}^L + a_z^G(t)\hat{k}^G\cdot\hat{k}^L = \left|\vec{a}^G(t)\right|\mu_{a\hat{k}}(t)\,. \end{aligned} \tag{55}$$

The object’s path-coordinate velocity is

$$\vec{v}^P(t) = v_x^P(t)\hat{i}^P + v_y^P(t)\hat{j}^P + v_z^P(t)\hat{k}^P\,. \tag{56}$$

By using Eqs.(38) and (41), we obtain

$$\begin{aligned} v_x^P(t) &= -\rho\sin\theta\frac{d\theta}{dt} \\ v_y^P(t) &= \rho\sin\theta\frac{d\theta}{dt} \\ v_z^P(t) &= 0 \quad . \end{aligned} \tag{57}$$

Similarly, the object’s path-coordinate acceleration is

$$\vec{a}^P(t) = a_x^P(t)\hat{i}^P + a_y^P(t)\hat{j}^P + a_z^P(t)\hat{k}^P \tag{58}$$

$$\begin{aligned} a_x^P(t) &= -\rho\left\{\cos\theta\left(\frac{d\theta}{dt}\right)^2 + \sin\theta\frac{d^2\theta}{dt^2}\right\} \\ a_y^P(t) &= \rho\left\{-\sin\theta\left(\frac{d\theta}{dt}\right)^2 + \cos\theta\frac{d^2\theta}{dt^2}\right\} \\ a_z^P(t) &= 0 \quad , \end{aligned} \tag{59}$$

where

$$\begin{aligned} \frac{d\theta}{dt} &= \omega_0 + \alpha\left(t - t_0\right) \\ \frac{d^2\theta}{dt^2} &= \alpha \quad . \end{aligned} \tag{60}$$

On inspection of Fig. 7, it is observed that the angles between the object's velocity vector and its local-coordinate axes remain invariant during motion. Thus, $\mu_{v\hat{i}}$, $\mu_{v\hat{j}}$, and $\mu_{v\hat{k}}$ can be determined at the beginning of the motion segment and are known throughout the object's transit of the path.

The angles between the object's acceleration vector and its local-coordinate axes may or may not change as the object moves. For the special case of constant velocity, these angles are fixed so that $\mu_{a\hat{i}}$, $\mu_{a\hat{j}}$, $\mu_{a\hat{k}}$ are known from the initial conditions at the start of the motion segment. For the general case of acceleration or deceleration, these angle change during motion so that $\mu_{a\hat{i}}(t)$, $\mu_{a\hat{j}}(t)$, $\mu_{a\hat{k}}(t)$ are unknowns.

For the general case of an accelerating or decelerating object, equations (47), (50), (51), (54), and (55) comprise 15 equations and the 18 unknowns $B_1^{GL}(t) - B_9^{GL}(t)$, $\mu_{R\hat{i}}(t)$, $\mu_{R\hat{j}}(t)$, $\mu_{R\hat{k}}(t)$, $\mu_{a\hat{i}}(t)$, $\mu_{a\hat{j}}(t)$, $\mu_{a\hat{k}}(t)$, $\mu_{O\hat{i}}(t)$, $\mu_{O\hat{j}}(t)$, and $\mu_{O\hat{j}}(t)$. To better visualize these equations, we collect them below and highlight the unknown quantities in red:

$$\begin{aligned}
R_x^G(t)B_1^{GL}(t)+R_y^G(t)B_2^{GL}(t)+R_z^G(t)B_3^{GL}(t)&=\left|\vec{R}^G(t)\right|\mu_{R\hat{i}}(t)\\
r_x^G(t)B_1^{GL}(t)+r_y^G(t)B_2^{GL}(t)+r_z^G(t)B_3^{GL}(t)&=\left|\vec{r}^{\,G}(t)\right|\mu_{r\hat{i}}\\
O_x^G B_1^{GL}(t)+O_y^G B_2^{GL}(t)+O_z^G B_3^{GL}(t)&=\left|\vec{O}^G\right|\mu_{O\hat{i}}(t)\\
v_x^G(t)B_1^{GL}(t)+v_y^G(t)B_2^{GL}(t)+v_z^G(t)B_3^{GL}(t)&=\left|\vec{v}^{\,G}(t)\right|\mu_{v\hat{i}}\\
a_x^G(t)B_1^{GL}(t)+a_y^G(t)B_2^{GL}(t)+a_z^G(t)B_3^{GL}(t)&=\left|\vec{a}^{\,G}(t)\right|\mu_{a\hat{i}}(t)\\
R_x^G(t)B_4^{GL}(t)+R_y^G(t)B_5^{GL}(t)+R_z^G(t)B_6^{GL}(t)&=\left|\vec{R}^G(t)\right|\mu_{R\hat{j}}(t)\\
r_x^G(t)B_4^{GL}(t)+r_y^G(t)B_5^{GL}(t)+r_z^G(t)B_6^{GL}(t)&=\left|\vec{r}^{\,G}(t)\right|\mu_{r\hat{j}}\\
O_x^G B_4^{GL}(t)+O_y^G B_5^{GL}(t)+O_z^G B_6^{GL}(t)&=\left|\vec{O}^G\right|\mu_{O\hat{j}}(t)\\
v_x^G(t)B_4^{GL}(t)+v_y^G(t)B_5^{GL}(t)+v_z^G(t)B_6^{GL}(t)&=\left|\vec{v}^{\,G}(t)\right|\mu_{v\hat{j}}\\
a_x^G(t)B_4^{GL}(t)+a_y^G(t)B_5^{GL}(t)+a_z^G(t)B_6^{GL}(t)&=\left|\vec{a}^{\,G}(t)\right|\mu_{a\hat{j}}(t)\\
R_x^G(t)B_7^{GL}(t)+R_y^G(t)B_8^{GL}(t)+R_z^G(t)B_9^{GL}(t)&=\left|\vec{R}^G(t)\right|\mu_{R\hat{j}}(t)\\
r_x^G(t)B_7^{GL}(t)+r_y^G(t)B_8^{GL}(t)+r_z^G(t)B_9^{GL}(t)&=\left|\vec{r}^{\,G}(t)\right|\mu_{r\hat{j}}\\
O_x^G B_7^{GL}(t)+O_y^G B_8^{GL}(t)+O_z^G B_9^{GL}(t)&=\left|\vec{O}^G\right|\mu_{O\hat{j}}(t)\\
v_x^G(t)B_7^{GL}(t)+v_y^G(t)B_8^{GL}(t)+v_z^G(t)B_9^{GL}(t)&=\left|\vec{v}^{\,G}(t)\right|\mu_{v\hat{j}}\\
a_x^G(t)B_7^{GL}(t)+a_y^G(t)B_8^{GL}(t)+a_z^G(t)B_9^{GL}(t)&=\left|\vec{a}^{\,G}(t)\right|\mu_{a\hat{j}}(t)
\end{aligned}\tag{61}$$

Before proceeding with the general formulation and its solution, let us consider formulations for motion in three simpler cases. The simpler cases not only provide insight into the solution scheme for the general case, but also are themselves necessary because of degeneracy of the general formulation that leads to these simpler cases.

## *6.2. Motion for a path parallel to* $x^G - y^G$ , $x^G - z^G$ , or $y^G - z^G$ *planes.*

It is anticipated that, for the sake of convenience, code users will often construct models wherein object motion occurs in the MCNP6 global-coordinate $x^G - y^G$ , $x^G - z^G$ , or $y^G - z^G$ planes. For motion in these planes, a subset of the expressions in Eq.(61) are needed because some of the orientation quantities are known. Consider, for example, the case of motion in the $x^G - y^G$ plane. Here the path and MCNP6 global-coordinate axes satisfy $B_9^{GP} = \hat{k}^G \cdot \hat{k}^P = 1$. To further simplify the analysis, also consider the situation in which the object's local-coordinate $z^L$ axis is oriented parallel to the MCNP6 $z^G$ axis. This means that $B_3^{GL}(t) = \hat{k}^G \cdot \hat{i}^L = 0$, $B_6^{GL}(t) = \hat{k}^G \cdot \hat{j}^L = 0$, $B_7^{GL}(t) = \hat{i}^G \cdot \hat{k}^L = 0$, $B_8^{GL}(t) = \hat{j}^G \cdot \hat{k}^L = 0$, and $B_9^{GL}(t) = \hat{k}^G \cdot \hat{k}^L = 1$. Consequently, the only unknown orientation coefficients in Eq.(61) are $B_1^{GL}(t), B_2^{GL}(t), B_4^{GL}(t)$, and $B_5^{GL}(t)$. We are free to select from Eq.(61) any of the equations involving $B_1^{GL}(t), B_2^{GL}(t), B_4^{GL}(t)$, and $B_5^{GL}(t)$. By selecting from Eq.(61) equations that involve $\vec{r}^G(t)$ and $\vec{v}^G(t)$, we get

$$\begin{bmatrix} r_x^G(t) & r_y^G(t) & 0 & 0 \\ v_x^G(t) & v_y^G(t) & 0 & 0 \\ 0 & 0 & r_x^G(t) & r_y^G(t) \\ 0 & 0 & v_x^G(t) & v_y^G(t) \end{bmatrix} \begin{bmatrix} B_1^{GL}(t) \\ B_2^{GL}(t) \\ B_4^{GL}(t) \\ B_5^{GL}(t) \end{bmatrix} = \begin{bmatrix} \left|\vec{r}^G(t)\right| \mu_{r\hat{i}} \\ \left|\vec{v}^G(t)\right| \mu_{v\hat{i}} \\ \left|\vec{r}^G(t)\right| \mu_{r\hat{j}} \\ \left|\vec{v}^G(t)\right| \mu_{v\hat{j}} \end{bmatrix}. \qquad (62)$$

This expression has several desired properties. First, it can be shown (Mathematica, 1991) that this system is nonsingular. Second, on careful inspection of Fig. 7 we observe that the direction cosines $\mu_{r\hat{i}}$ and $\mu_{r\hat{j}}$ between $\vec{r}^G(t)$ and the object's $\hat{i}^L$ and $\hat{j}^L$ local-coordinate axes do not change as the object moves along its circular path. Because the direction of the object's velocity vector is tangent to its path of motion, the direction cosines $\mu_{v\hat{i}}$ and $\mu_{v\hat{j}}$ between $\vec{v}^G(t)$ and $\hat{i}^L$ and $\hat{j}^L$ are also invariant as the object moves along the circular path. Thus, the right-hand side of Eq.(62) is known at all times.

Consequently, Eq.(62) contains four equations and four unknowns which can be solved for $B_1^{GL}(t)$, $B_2^{GL}(t)$, $B_4^{GL}(t)$, and $B_5^{GL}(t)$ to provide the orientation of the object at any time.[†] This result is valid for an object undergoing constant angular velocity or an object experiencing increasing (decreasing) angular velocity due to constant angular acceleration (deceleration). It is also valid for objects whose local-coordinate $\hat{k}^L$ axis is not oriented parallel to the MCNP6 $\hat{k}^G$ axis—in such cases $B_3^{GL}(t)$, $B_6^{GL}(t)$, $B_7^{GL}(t)$, $B_8^{GL}(t)$, and $B_9^{GL}(t)$ are known nonzero constants.[‡]

Similar expressions can be developed for motion in which the path is parallel to the $x^G - z^G$ or $y^G - z^G$ planes. Table 1 lists the associated $B^{GL}(t)$ and $\mu$ values for each case.

Table 1. Orientation location, velocity, and accelerator variables for motion path in the MCNP6 global-coordinate $x^G - y^G$, $x^G - z^G$, or $y^G - z^G$ planes.

| Motion plane | Cosines of angles between object local and MCNP6 global coordinate axes | Cosines of angles between object's position and velocity vectors and its local-coordinate axes |
|---|---|---|
| $x^G - y^G$ | $B_1^{GL}(t), B_2^{GL}(t), B_4^{GL}(t), B_5^{GL}(t)$ | $\mu_{r\hat{i}}, \mu_{r\hat{j}}, \mu_{v\hat{i}}, \mu_{v\hat{j}}$ |
| $x^G - z^G$ | $B_1^{GL}(t), B_3^{GL}(t), B_7^{GL}(t), B_9^{GL}(t)$ | $\mu_{r\hat{i}}, \mu_{r\hat{k}}, \mu_{v\hat{i}}, \mu_{v\hat{k}}$ |
| $y^G - z^G$ | $B_5^{GL}(t), B_6^{GL}(t), B_8^{GL}(t), B_9^{GL}(t)$ | $\mu_{r\hat{j}}, \mu_{r\hat{k}}, \mu_{v\hat{j}}, \mu_{v\hat{k}}$ |

So for the cases of object motion in the $x^G - y^G$, $x^G - z^G$, or $y^G - z^G$ planes, the time-dependent orientation involves the solution of Eq.(62) or its equivalent (Table 1) for four

[†] During execution, MCNP6 calculates the four direction cosines in the right-hand side of Eq.(62) using data at the start of the curvilinear motion segment.
[‡] These values are specified by the MCNP6 user in the input file.

orientation parameters. We now return to the issue of motion on a path having arbitrary orientation.

## *6.3. Motion for a path plane of arbitrary orientation.*

For motion along a path plane of arbitrary orientation, we have at our disposal the vectors $\vec{R}^G(t)$, $\vec{v}^G(t)$, $\vec{a}^G(t)$, $\vec{r}^G(t)$, and $\vec{O}_A^G$. Equation (61) comprises 15 equations and 18 unknowns involving the object's orientation parameters $B_1^{GL}(t) - B_9^{GL}(t)$ as well as the direction cosines between each vector $\vec{R}^G(t)$, $\vec{v}^G(t)$, $\vec{a}^G(t)$, $\vec{r}^G(t)$, and $\vec{O}_A^G$ and the object's local-coordinate axes given by the $\mu$ parameters. We must discover a way to close the system of equations.

One attempt to close the system can be made using the identities involving the sum of the squares of the direction cosines involving $\vec{R}^G(t)$, $\vec{a}^G(t)$, and $\vec{O}_A^G$ equaling 1:

$$\begin{aligned}\mu_{R\hat{i}}^2 + \mu_{R\hat{j}}^2 + \mu_{R\hat{k}}^2 &= 1\\ \mu_{a\hat{i}}^2 + \mu_{a\hat{j}}^2 + \mu_{a\hat{k}}^2 &= 1\\ \mu_{O\hat{i}}^2 + \mu_{O\hat{j}}^2 + \mu_{O\hat{k}}^2 &= 1.\end{aligned} \tag{63}$$

However, it is not clear that this system is well posed in terms of information uniqueness. For instance, the relationship among $\vec{R}^G(t)$, $\vec{r}^G(t)$, and $\vec{O}_A^G$ in Eq.(34) could mean that redundant information is used, thus jeopardizing the solution. Moreover, use of Eq.(61) and Eq.(63) would give a set of 18 coupled nonlinear algebraic equations. The numerical evaluation of such a set of equations is typically a difficult issue. So we seek an alternate formulation.

If Eq. (34) is used directly, then the dot products of $\vec{R}^G(t)$, $\vec{r}^G(t)$, and $\vec{O}_A^G$ with the object's local-coordinate axes can be evaluated and, subsequently, the results substituted in the right-hand side of the 15 equations involving $\vec{R}^G(t)$, $\vec{v}^G(t)$, $\vec{a}^G(t)$, $\vec{r}^G(t)$, and $\vec{O}_A^G$. This procedure results in 15 coupled linear algebraic equations in 15 unknowns. However, it can be shown (Mathematica, 1991) that this system is singular, which implies that the information content, i.e., the vector relationships, is not unique and that the system is not formulated correctly.

Let us consider the case in which an object experiences no angular acceleration (i.e., the angular velocity is constant). This means that the tangential component of the acceleration in path coordinates is zero. All acceleration is oriented towards the axis of rotation via the normal component. For the $x^G - y^G$ formulation, we used the equations in Eq.(61) involving $\vec{r}^G(t)$ and $\vec{v}^G(t)$ and exploited the fact that the direction cosines between $\vec{r}^G(t)$ and the object's local-coordinate axes are invariant in time. The same was true for $\vec{v}^G(t)$. These invariance properties also hold true for this 3-D case. However, using the equations in Eq.(61) involving $\vec{r}^G(t)$ and $\vec{v}^G(t)$ for this 3-D case leaves us with six equations with all nine $B^{GL}(t)$ values unknown. Consequently, at least three more equations must be stipulated.

Once again considering Eq.(61), we have available the equation involving the acceleration $\vec{a}^G(t)$. Using it along with the equations for $\vec{r}^G(t)$ and $\vec{v}^G(t)$ provides nine equations. However, in general three additional unknowns $\mu_{a\hat{i}}(t)$, $\mu_{a\hat{j}}(t)$, and $\mu_{a\hat{k}}(t)$ are introduced, leaving us with 12 unknowns. For the special case of no angular acceleration, the acceleration vector has only a normal component. Thus, the angle between the acceleration vector and the object's local-coordinate axes is constant so that we have nine equations and the nine unknowns $B_1^{GL}(t) - B_9^{GL}(t)$. However, it is readily shown (Mathematica, 1991) that this system is singular. The singular behavior arises because $\vec{r}^{\,G}(t)$ and $\vec{a}^{\,G}(t)$ vectors are collinear, but are oriented in opposite directions. The determinant of the coefficient matrix contains sums and differences of products of the direction cosines $\mu_{r\hat{i}}(t)$, $\mu_{r\hat{j}}(t)$, and $\mu_{r\hat{k}}(t)$ and $\mu_{a\hat{i}}(t)$, $\mu_{a\hat{j}}(t)$, and $\mu_{a\hat{k}}(t)$ that equate to zero.

Use of the equations involving either $\vec{R}^{G}(t)$ or $\vec{O}_A^{G}$ is problematic. The difficulty occurs because the direction cosines between these vectors and the object's local-coordinate axes, $\mu_{R\hat{i}}(t)$, $\mu_{R\hat{j}}(t)$, and $\mu_{R\hat{k}}(t)$ or $\mu_{O\hat{i}}(t)$, $\mu_{O\hat{j}}(t)$, and $\mu_{O\hat{k}}(t)$ are time dependent regardless of constant or varying motion. Thus, the resulting systems contain 9 equations and 12 unknowns.

A resolution to the dilemma requires a different approach. We consider a new vector $\vec{N}^{G}(t)$ that is orthogonal to the location and velocity vectors. $\vec{N}^{G}(t)$ is created by taking the cross-product of $\vec{r}^{\,G}(t)$ and $\vec{v}^{\,G}(t)$ to give

$$\begin{aligned}\vec{N}^G(t) &= \left[ r_y^G(t)v_z^G(t) - r_z^G(t)v_y^G(t) \right]\hat{i}^G + \left[ r_z^G(t)v_x^G(t) - r_x^G(t)v_z^G(t) \right]\hat{j}^G \\ &+ \left[ r_x^G(t)v_y^G(t) - r_y^G(t)v_x^G(t) \right]\hat{k}^G\end{aligned} . \quad (64)$$

This vector is used with $\vec{r}^G(t)$ and $\vec{v}^G(t)$ to form the following system of equations:

$$\begin{bmatrix} r_x^G(t) & r_y^G(t) & r_z^G(t) & 0 & 0 & 0 & 0 & 0 & 0 \\ v_x^G(t) & v_y^G(t) & v_z^G(t) & 0 & 0 & 0 & 0 & 0 & 0 \\ N_x^G(t) & N_y^G(t) & N_z^G(t) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_x^G(t) & r_y^G(t) & r_z^G(t) & 0 & 0 & 0 & 0 \\ 0 & 0 & v_x^G(t) & v_y^G(t) & v_z^G(t) & 0 & 0 & 0 & 0 \\ 0 & 0 & N_x^G(t) & N_y^G(t) & N_z^G(t) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r_x^G(t) & r_y^G(t) & r_z^G(t) \\ 0 & 0 & 0 & 0 & 0 & v_x^G(t) & v_y^G(t) & v_z^G(t) \\ 0 & 0 & 0 & 0 & 0 & N_x^G(t) & N_y^G(t) & N_z^G(t) \end{bmatrix} \begin{bmatrix} B_1^{GL}(t) \\ B_2^{GL}(t) \\ B_3^{GL}(t) \\ B_4^{GL}(t) \\ B_5^{GL}(t) \\ B_6^{GL}(t) \\ B_7^{GL}(t) \\ B_8^{GL}(t) \\ B_9^{GL}(t) \end{bmatrix} = \begin{bmatrix} \left|\vec{r}^G(t)\right|\mu_{r\hat{i}} \\ \left|\vec{v}^G(t)\right|\mu_{v\hat{i}} \\ \left|\vec{N}^G(t)\right|\mu_{N\hat{i}} \\ \left|\vec{r}^G(t)\right|\mu_{r\hat{j}} \\ \left|\vec{v}^G(t)\right|\mu_{v\hat{j}} \\ \left|\vec{N}^G(t)\right|\mu_{N\hat{j}} \\ \left|\vec{r}^G(t)\right|\mu_{r\hat{k}} \\ \left|\vec{v}^G(t)\right|\mu_{v\hat{k}} \\ \left|\vec{N}^G(t)\right|\mu_{N\hat{k}} \end{bmatrix} . \quad (65)$$

It can be shown (Mathematica, 1991) that Eq.(65) is nonsingular. Each of the direction cosines between $\vec{r}^G(t)$ and $\vec{v}^G(t)$ and the object's local-coordinate axes are known and time invariant. Consequently, the direction cosines between $\vec{N}^G(t)$ and the object's local-coordinate axes, $\mu_{N\hat{i}}$, $\mu_{N\hat{j}}$, and $\mu_{N\hat{k}}$, are also known and time invariant. Therefore, Eq.(65) constitutes a well posed formulation that can be solved for the object's orientation parameters $B_1^{GL}(t) - B_9^{GL}(t)$. This formulation is valid for an object experiencing either constant or varying velocity on a tilted plane.

In summary, for curvilinear rotation two sets of calculations are required. The first set determines the object's location, and is done using Eqs.(33)–(46). The second set calculates the object's orientation. For motion involving paths whose planes are parallel to the MCNP6 global-coordinate $x^G - y^G$, $x^G - z^G$, or $y^G - z^G$ planes, Eq.(62) or its equivalent (Table 1) is solved for four orientation parameters. For motion in a plane of arbitrary orientation, Eq.(65) is solved for all $B_1^{GL}(t) - B_9^{GL}(t)$. In MCNP6, LU decomposition (Press, 1992) is used to solve the systems of equations in Eq.(62) and Eq.(65).

## 7. Moving sources

MCNP6 contains a powerful "general" source capability that encompasses source particle type, geometry, energy, direction, orientation, and time dependence. Recent modifications to MCNP6 now enable simulations with general source motion.

The dynamics treatment of moving sources is identical to that for moving objects. Moving sources can experience rectilinear or curvilinear translation or curvilinear rotation. The equations of motion are given in the preceding sections.

Source motion in MCNP6 is accomplished by assigning a source to a moving object. The source moves according to the dynamics parameters of the moving object. The source characteristics, i.e., energy, spatial configuration, direction, time-dependence, etc. are then used to produce particles as the source moves.

## 8. Delayed particles

MCNP6 simulations can be configured to treat either "prompt" particles or prompt particles accompanied by delayed neutrons and/or gammas (DN and DG). MCNP6 treats

DN and DG emission, inclusive of fission and activation reactions (Durkee et al., 2009). The moving-object feature includes the treatment of DN and/or DG (delayed-particle “DP”) emission. Execution with moving objects is permitted in “source mode” only (kcode-mode execution is not allowed).

The DP emission treatment for moving objects applies the object’s motion characteristics. First, the location, direction, time, and energy data for the delayed particle at the instant of fission or activation are stored (the DP emission location occurs at some location within the object), as is motion data for the object. Second, at the time of DP emission (typically tens seconds to hours later), the object’s location and orientation are calculated. These two sets of object-motion data are used to calculate and provide the correct location and orientation of the delayed-particle emission from within the object at the time of emission. Figure 8 illustrates the concept.

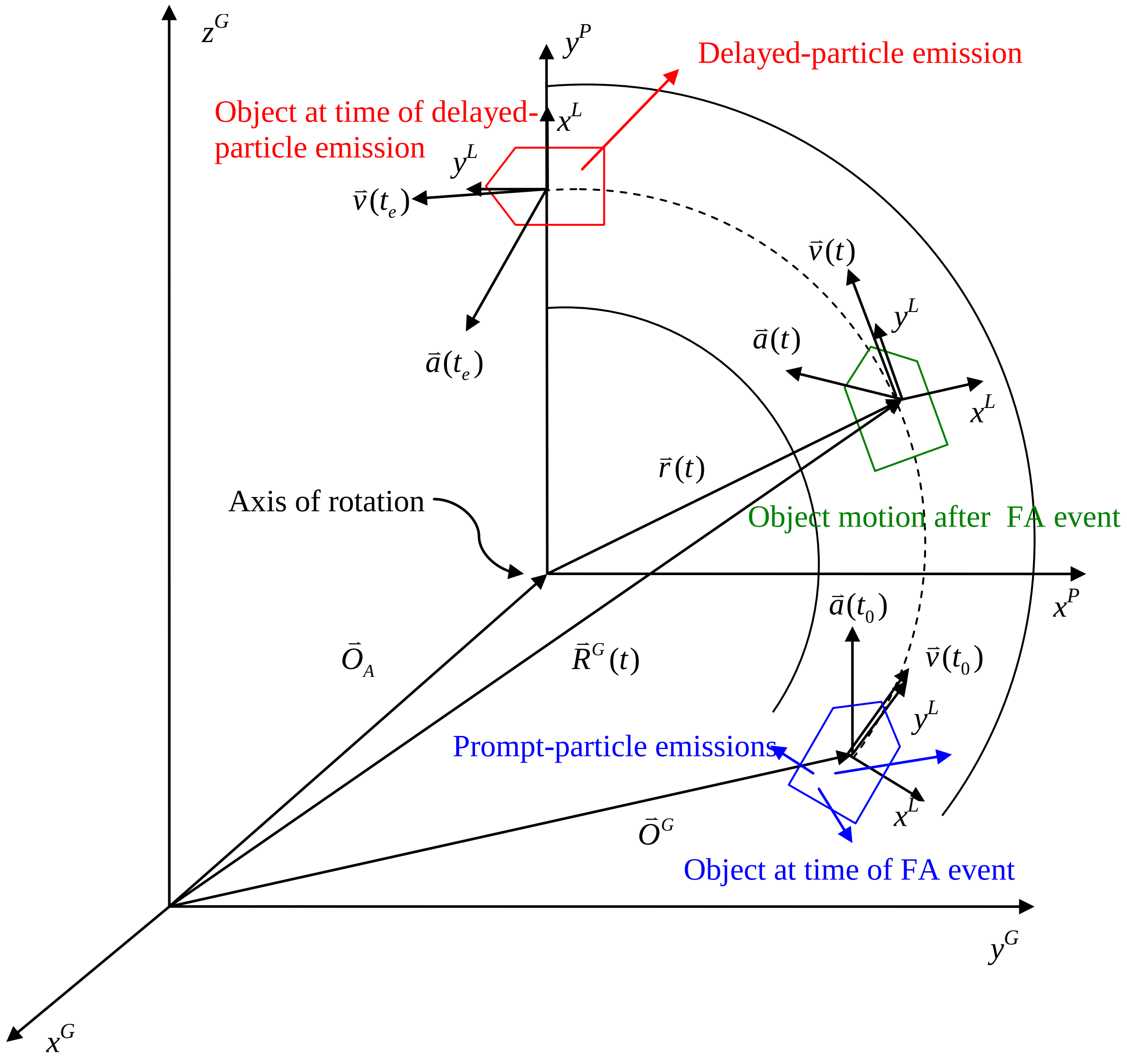

**Figure 8**. Delayed-particle emission. A fission or activation (FA) event occurs at time $t_0$ from within the object. The object's location and orientation changes from time $t_0$ to the time of delayed-particle emission, $t_e$.

Delayed-particle emission is performed only if the calculated emission time occurs prior to the maximum time that an object is tracked. For example, if an object is in motion for 1000 seconds, delayed particles that are emitted up to 1000 seconds are treated. Delayed particles emitted after 1000 seconds are not tracked. This treatment helps to alleviate geometry and emission conflicts and errors that would arise if the user desired to track an object for a given duration but emission occurred at a later time.

## 9. Moving object particle history treatment

Following emission, MCNP6 tracks a particle throughout its lifetime. When a particle experiences a reaction that results in the production of additional particles, information about the additional particles is stored ("banked"). The initiating particle is followed to its termination (i.e., absorption, movement out the the problem's geometry, low-energy cutoff termination, etc.). Each particle that was created during the initiating particle's life is then removed from the bank and followed.

Particle transport involving one or more moving objects is treated in the following manner. The entire geometry for the simulation model is updated to the time at which a source particle is created. The geometry is fixed (static) during the transport of the source particle. The geometry remains fixed for the transport of any prompt particles that are emitted as a result one or more interactions of a source particle with matter.

If one or more delayed particles are created, then the geometry is updated to the time at which each delayed particle is created. The geometry-update procedure is detailed in Section 8.

## 10. Moving objects feature limitations

In Section 3 we listed nine stipulations that limit the moving-objects capability. In addition to those stipulations, the treatment of motion involving particle transport and moving objects is approximate because the geometry is fixed during a particle's transport. Particle geometry is updated only to the time at which a source or delayed particle is created. Consequently, calculated results are the most accurate for simulations that have high-energy sources and slow moving objects in which the sources and objects are in close proximity. Deterioration of results become increasingly apparent as particle energy decreases, object speed increases, and the spacing between sources and objects increases. We have no mechanism with which to quantify this behavior.

## 11. Geometry plotting and checking

MCNP6 has long had the capability to detect geometry errors and lost particles (Pelowitz, 2008). Geometry fidelity can be inspected using the MCNP6 plot utility, which displays geometry errors using red dashed "cookie cutter" lines. During particle transport, MCNP6 checks for continuity in the cells and surfaces. Lost particles arise because of geometry errors, and a detailed "event log" of the lost particle's transport history is provided in the output (“outp”) file to help the user decipher geometry issues. In the end, it is the user’s responsibility to ensure geometry integrity.

For moving objects, the same geometry-checking and lost-particle capabilities are utilized by MCNP6. As is the case for static-geometry models, the user must develop models that function properly. For models involving moving objects, the process is more demanding that that for static models. The user must develop models carefully so that moving objects do not collide with either static objects or other moving objects. This task is accomplished in part by using plot, and by making use of the lost-particle event log.

To assist the user with this responsibility, some checking capabilities have been implemented in MCNP6.  These checks pertain to motion of an individual object when multiple motion segments are used to depict its motion. Checking is done to help ensure continuity of position and time at motion-segment boundaries.

The moving-object dynamics parameters and the geometry must include consideration of the motion (and extent thereof) for each object during the entirety of motion. Not only must objects avoid collision, but objects must also not move beyond the extent of the outer limits of the problem geometry.

Models that have delayed-particle emission must be given additional consideration. Delayed neutrons and gammas may be emitted seconds to hours after a fission or activation event. A restraint has been installed in MCNP6 to prohibit DP transport when the DP emission occurs after the maximum time at which moving-object motion is treated. This restraint precludes accidental geometry and lost-particle errors associated with failure to consider object location at late delayed-particle emission times. The user must ensure that the model's geometric extent is great enough (particularly when rectilinear translation is involved) that all desired DPs are treated (e.g., transported, tallied, etc). The problem setup must consider the time and location of the object when fission occurs and the motion of the object between the time of fission and delayed-particle emission.

In addition, the MCNP6 plot utility has been upgraded to provide snapshot images of the geometry as a function of time. To accomplish this, the new "time" command has been added. The user can use the "time" command to specify the range of time for which the geometry is to be plotted, the number of snapshot images to be plotted, and the duration for which each image is to appear on the screen.

## 12. Moving-object coding implementations

The moving-object development has resulted in modifications to 33 existing subroutines and the creation of 10 new subroutines in the form of approximately 4000 lines of commented Fortran. The required upgrades encompass the treatment of geometry, source, delayed particles, input processing, graphics, and MPI execution. The use of Fortran 90 derived data types (Ellis et al, 1998) allowed pertinent variables to be encapsulated for dynamic objects, thus improving the coding appearance and reducing clutter. Additional details are presented in the Appendix.

## 13. Summary and conclusions

Previously, radiation-transport simulations executed using MCNP6 were limited to stationary geometry. Our newest innovation establishes a new capability in MCNP6 that permits simulations involving object motion. This capability uses rigid-body kinematics to characterize object motion. Three motion types are treated: rectilinear translation, curvilinear translation, and curvilinear rotation. Rectilinear translation treats objects moving in straight lines without a change in the object’s orientation. Curvilinear motion treats motion along circular paths, either with (rotation) or without (translation) a change in the object’s orientation. Objects can be moved 1) by assigning the object a constant velocity, 2) by variable velocity subject to the constraints of constant acceleration or deceleration, or 3) simple relocation.

This moving-objects capability has been designed to provide the user with the flexibility to independently specify the location and orientation of each object and its path. This development has been achieved by the appropriate formulation and implementation of dynamics equations for the object, its path, and their relationship to MCNP6 global coordinates.

Motion involving orientation change (curvilinear rotation) of an object necessitates the calculation of the object’s orientation parameters as a function of time. Based on the examination of pertinent dynamics vector quantities in object, path, and MCNP6 global coordinates, we have formulated and implemented in MCNP6 matrix expressions whose solution gives the time-dependent object-orientation parameters.

Specification of  moving objects location and dynamics parameters is done using the new dynamic coordinate transformation (TRD) and motion (MOT) cards. The conventional coordinate transformation TRCL keyword is used to associate each moving object with its dynamics parameters.

The new capability also accommodates moving sources of prompt particles, including spontaneous-fission sources. The prompt particle source specifications are specified using the conventional MCNP6 SDEF card with the TR keyword.  The TR number is used to relate the moving source input data to TRD and MOT card data. The new MOT card contains a “srctype” keyword which is set to 0 to specify a fixed source and 1 to specify a moving source. This upgrade thus extends the MCNP6 SDEF source to include motion.

Delayed-particle (neutrons and photons) treatment is also included in the upgrade. This treatment includes delayed-particle emission due to fission, photofission, and activation events. Objects are moved from the time of these initiating events to their locations at the time of delayed-particle emission.

An animation feature has been added to the MCNP6 plot geometry plotter. This feature plots a series of images of a moving-object geometry to the user’s computer screen as the geometry evolves in time. This feature helps to examine and ensure geometry fidelity during model construction and provide illuminating graphics for documentation and presentations.

The moving objects feature has been implemented to execute in either serial mode or parallel mode using MPI, giving users the flexibility to develop models and execute complex simulations.

The MCNP6 moving-objects feature has been developed for simulations with radiation sources, i.e., source-mode (SDEF)  simulations. Future work will be required to enable criticality (kcode-mode) simulations with moving objects.

Simulation studies involving the new capability are presented in Part II. There we present several calculations that illustrate the motion capabilities. We also provide calculations with delayed-gamma emission from moving targets.

The moving-object feature is of  interest for a wide range of applications involving radiation transport and objects that experience motion. Such applications include active and passive homeland security studies, biomedicine, environmental studies, radiolabelled quantities, and others.

**Acknowledgements**

We appreciate the support provided by the Department of Homeland Security Domestic Nuclear Detection Office.

## References

Ayranov M. and Schumann D., 2010. “Preparation of $^{26}$Al, $^{59}$Ni, $^{44}$Ti, $^{53}$Mn and $^{60}$Fe From a Proton Irradiated Copper Beam,” J. Radioanalytical Nuclear Chemistry, 286, 649–654.

Barta J., Pospisil M., and Cuba V., 2010. “Photo- and radiation-induced preparation of nanocrystalline copper and cuprous oxide catalysts,” J. Radioanalytical Nuclear Chemistry, 286, 611–618.

Barzilov A.P., Novikov I.S., and Cooper B., 2009. “Computational Study of Pulsed Neutron Induced Activation Analysis of Cargo,” J. Radioanalytical Nuclear Chemistry, 282, 177–181.

Brown F.B., ed., April 2003a, “MCNP–A General Monte Carlo N–Particle Transport Code, Version 5, Volume I: Overview and Theory,” Los Alamos National Laboratory report LA-CP-03-0245.

Brown F.B., ed., April 2003b, “MCNP–A General Monte Carlo N–Particle Transport Code, Version 5, Volume II: User’ Guide,” Los Alamos National Laboratory report LA-CP-03-0245.

Chichester D.L. and Seabury E.H., 2009. “Using Electronic Neutron Generators in Active Interrogation to Detect Shielded Fissonable Material,” IEEE Transactions on Nuclear Science, 56(2), 441–447.

Dasari K.B., Acharya R., Swain K.K., Lakshmana Das N., and Reddy A.V.R., 2010. “Analysis of Large and Non-Standard Geometry Samples of Ancient Potteries by Internal Monostandard Neutron Activation Analysis Using in Situ Detection Efficiency,” J. Radioanalytical Nuclear Chemistry, 286 (2), 525–531.

Dodd B., ed., April 2001. “Use of Research Reactors for Neutron Activation Analysis,” International Atomic Energy Agency report IAEA-TECDOC-1215.

Durkee J. W., Jr., James M.R., McKinney G.W., Trellue H.R., Waters L.S., and Wilson W.B., 2009a. “Delayed-Gamma Signature Calculation for Neutron-Induced Fission and Activation Using MCNPX, Part I: Theory,” Progress in Nuclear Energy, 51, 813–827.

Durkee J.W., Jr., McKinney G.W., Trellue H.R., Waters L.S., and Wilson W.B., 2009b. “Delayed-Gamma Simulation Using MCNPX,” J. Nuclear Technology, 168, 761–764.

Durkee J.W., Jr., Elson J.S., Jason A.J., Johns R.C., and Waters L.S., 2010. “An MCNPX Accelerator Beam Source,” J. Nuclear Technology, 168, 761–764.

Ellis T.M.R., Philips I.R., and Lahey T.M., 1998. *Fortran 90 Programming,* Addison-Wesley, pp. 67–70.

Fei T., Dehong L., Fengqun Z., Junhua L., Hua T., and Xiangzhong K., 2010. “Determination of Trace Elements in Chinese Medicinal Plants by Instrumental Neutron Activation Analysis,” J. Radioanalytical Nuclear Chemistry, 284, 507–511.

Frontasyeva M.V., Pavlov S.S., and Shvetsov V.N., 2010. “NAA for Applied Investigations at FLNP JINR: Present and Future,” J. Radioanalytical Nuclear Chemistry, 286, 519–524.

Goorley T., James M., Booth T., Brown F., Bull J., Cox L.J., Durkee J., Elson J., Fensin M., Forster R.A., Hendricks J., Hughes H.G., Johns R., Kiedrowski B., Martz R., Mashnik S., McKinney G., Pelowitz D., Prael R., Sweezy J., Waters L., Wilcox T., and Zukaitis T., “Initial MCNP6 Release Overview,” *J. Nuclear Technology,* submitted December 2011.

Gozani T., 2009. “Fission Signatures for Nuclear Material Detection,” IEEE Transactions on Nuclear Science, 56(3), 736–741.

Hibbeler R.C., 1974. *Engineering Mechanics: Statics and Dynamics*, Macmillan Publishing Co., Inc., New York, pp. 435–514.

Maestas B.A., Clark S.D., Flaska M., Pozzi S.A., Poitrasson-Riviere A., Pausch G, Claus-Michael H., Guergueiev A., Ohmes M., and Stein J., 2009. “Monte Carlo Investigation of a High-Sensitivity Two-Plane Compton Camera for Long-Range Detection of SNM,” IEEE Nuclear Science Symposium conference record, N13-244, 964–967.

Owen M., Weston G., and O’Malley J., 2009. “AWE Development of Active Interrogation Techniques for the Detection of SNM,” Proceedings of SPIE, 7304, p. 73041K-1 (9 pp.).

Pelowitz D.B., ed., April 2011. “MCNPX User’s Manual Version 2.7.0,” Los Alamos National Laboratory report LA-CP-11-00438.

Press W.H., Teukolsy S.A., Vetterline W.T., and Flannery B.P., 1992. *Numerical Recipes in FORTRAN*, Cambridge University Press, pp. 376–381.

Sengupta A., Adya V.C., Acharya R., Mohapatra P.K., and Manchanda V.K., 2010. “Characterization of Purified $^{241}$Am for Common Impurities by Instrumental Neutron Activation Analysis,” J. Radioanalytical Nuclear Chemistry, 287, 281–285.

Shypailo R.J. and Ellis K.J., 2004. “Design Considerations for a Neutron Generator-Based Total-Body Irradiator,” J. Radioanalytical Nuclear Chemistry, 263, 759–765.

Spain B., 2007. *Analytical Conics*, Dover Publications, Inc., Mineola, NY, pp. 66–70.

Thompson W.L., Cashwell E.D., Godfrey T.N.K., Schrandt R.G., Deutsch O.L., and Booth T.E., May 1980. “The Status of Monte Carlo at Los Alamos,” Los Alamos National Laboratory report LA-8353-MS, 17–22.

Thompson W.L., ed., November 1981. “MCNP–A General Monte Carlo Code for Neutron and Photon Transport, Version 2B” Los Alamos National Laboratory report LA-7396-M, Revised, p.1.

Tierney J.A., 1974, *Calculus and Analytic Geometry*, Allyn and Bacon, Inc., Boston, p. 226.

vonNeumann J., 1947. “Statistical Methods in Neutron Diffusion” Los Alamos National Laboratory report LAMS-551.

Wolfram S. (1991), *Mathematica*, Addison-Wesley, New York, Second Edition, p.123.

Wylie C.R., 1975, *Advanced Engineering Mathematics*, McGraw-Hill, New York, pp. 636–637.

## APPENDIX

Implementation of the moving-objects feature has resulted in modifications to 33 existing subroutines and the creation of 10 new subroutines in the form of approximately 4000 lines of commented Fortran. Here we provide additional information about code-development matters involving the MCNP6 moving-objects feature. This information includes an outline of the names and purposes of several new and some existing (modified) subroutines.

Subroutine **applyDynTrcl.F** performs the transformation at time t, resets the surface-coefficient array to contain local-coordinate values, and calls **trfsrf.F** to apply the surface transformation.

Delayed particle treatment for moving objects is done as follows. On entry to the delayed-neutron subroutine **colidn.F** (used for delayed-neutron emission via ACE library data) or delayed neutron/gamma subroutine **dng_model.F** (used for delayed neutron and gamma emission using models), the time (tgeomsp = te) and location of the fission or activation event are known. The time at which each delayed particle is emitted is calculated (tme = te + tdelay). This information is sent to subroutine **bankit.F**. In **bankit.F**, this delayed-particle timing data are used in conjunction with the dynamics data for the moving object (from which the delayed particle is emitted) to calculate the object's position and orientation at the times of a fission/activation event and the delayed particle's emission via a call to **applyDynTrclDP.F**. The change in the object's position and orientation is then applied to the delayed particle's position and orientation. The delayed particle's position and emission orientation are then saved to the bank. Another call is then made to **applyDynTrclDP.F** to reset the global position and direction data for the initiating particle (tgeomte) prior to exiting **bankit.F** and returning to the calling delayed-particle creation subroutine (**colidn.F** or **dng_model.F**).

When delayed particles emitted from moving objects are removed from the bank, subroutine **hstory.F** calls **bankit.F** to recall the delayed particle's information from the bank. The problem's geometry is then updated to the time at which the delayed particle is emitted via a call to **applyDynTrclDP.F**. Control from **bankit.F** is returned to the **hstory.F** and the delayed particle is transported.

Subroutine **applyDynTrclDP.F** ensures that (1) a delayed particle is emitted from a moving object and (2) the emission time does not exceed the maximum permissible tracking time for the problem. It also manipulates the location and orientation information for delayed particles emitted from moving objects so that the correct data are available for **bankit.F**. Calls are made from **applyDynTrclDP.F** to **applyDynTrcl.F** wherein the dynamic transformation operations are performed. The operations are illustrated in Fig.10a.

Subroutine **trfsrf.F** applies the surface transformation at time t to create the global-coordinate surface coefficients. These surface coefficients are loaded into the scf array. The scf coefficients are loaded into the "AM" matrix.

The $B_1^{GP} - B_9^{GP}$ in Eq.(29) are the terms involving the cosines of the angles between the rotation-path and MCNP6 global-coordinate axes. These quantities must be input by the user, and supplied using the new TRD card.

The moving objects feature has been implemented to execute in either serial mode or using MPI. The MPI upgrades have required modifications to the **GLOBAL3_st.F** and **GLOBAL5_cm.F** modules as well as subroutines **imcn.F**, **IMCN_jc.F**, **tpefil.F**, **msgcon.F**, **msgtsk.F**. Modifications to **GLOBAL5_cm.F** included subroutines **dynamic_allocate.F**, **fdac_write.F**, and **fdac_read.F**.

Figures 9–12 contain general schematics of portions of the overall code flow involving moving objects. The illustrations are intended to provide some idea of the relationship between many of the subroutines.

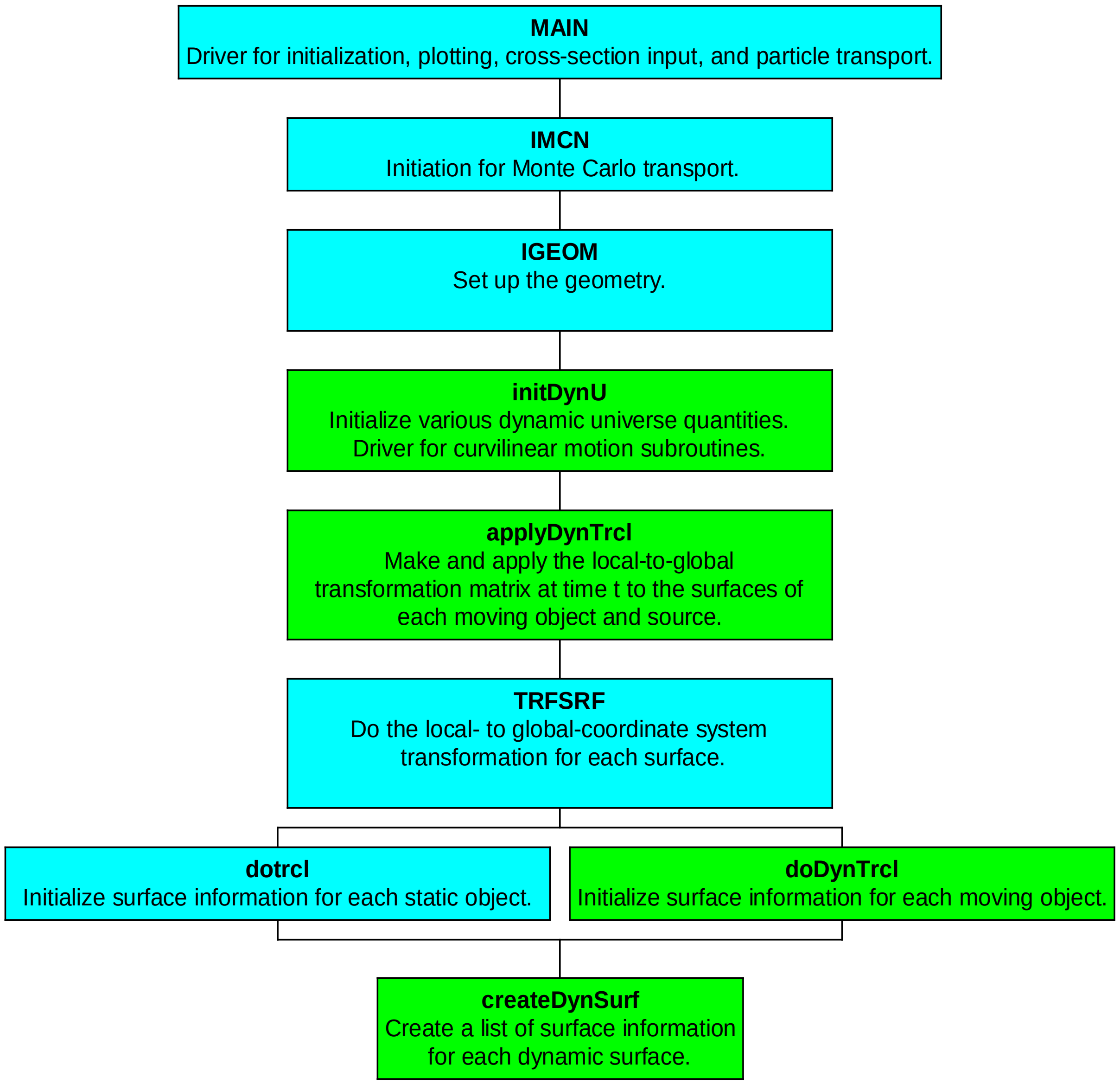
MAIN
Driver for initialization, plotting, cross-section input, and particle transport.
IMCN
Initiation for Monte Carlo transport.
IGEOM
Set up the geometry.
initDynU
Initialize various dynamic universe quantities. Driver for curvilinear motion subroutines.
applyDynTrcl
Make and apply the local-to-global transformation matrix at time t to the surfaces of each moving object and source.
TRFSRF
Do the local- to global-coordinate system transformation for each surface.
dotrcl
Initialize surface information for each static object.
doDynTrcl
Initialize surface information for each moving object.
createDynSurf
Create a list of surface information for each dynamic surface.

**Figure 9**. General MCNP6 initialization code flow for moving objects. Boxes shown in blue are historical subroutines. Green boxes connote new subroutines for moving objects.

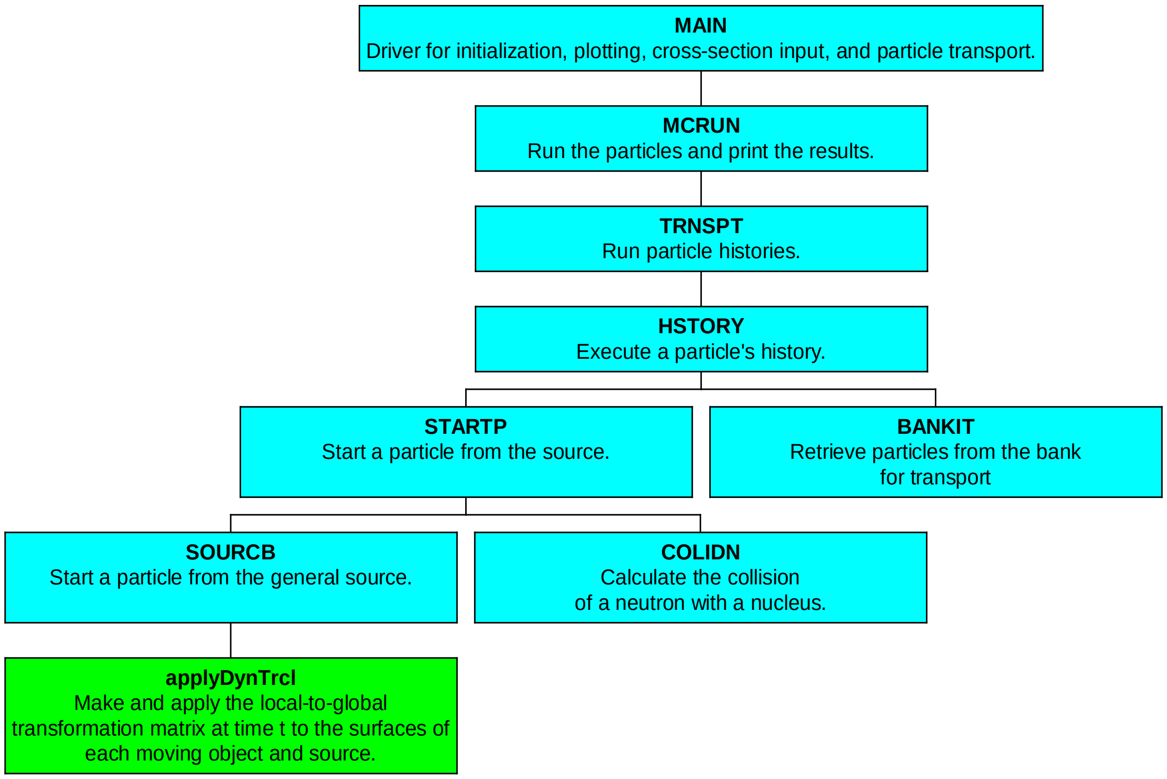
MAIN
Driver for initialization, plotting, cross-section input, and particle transport.
MCRUN
Run the particles and print the results.
TRNSPT
Run particle histories.
HSTORY
Execute a particle's history.
STARTP
Start a particle from the source.
BANKIT
Retrieve particles from the bank for transport
SOURCB
Start a particle from the general source.
COLIDN
Calculate the collision of a neutron with a nucleus.
applyDynTrcl
Make and apply the local-to-global transformation matrix at time t to the surfaces of each moving object and source.

**Figure 10**. General MCNP6 code flow for moving objects involving source-particle creation, particle retrieval from bank, or **colidn.F** for neutron interactions. Boxes shown in blue are historical subroutines. Green boxes connote new subroutines for moving objects.

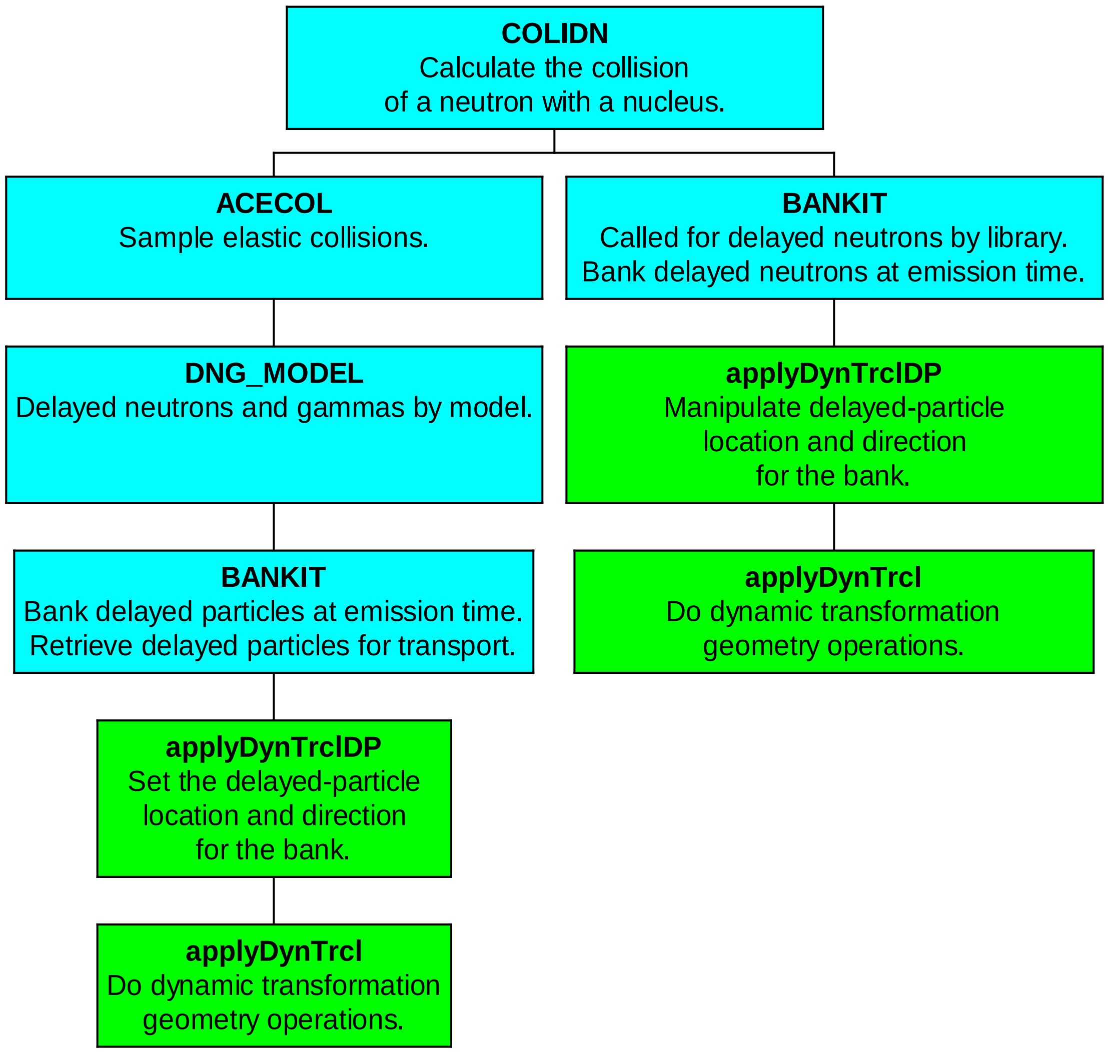
COLIDN
Calculate the collision of a neutron with a nucleus.
ACECOL
Sample elastic collisions.
BANKIT
Called for delayed neutrons by library. Bank delayed neutrons at emission time.
DNG_MODEL
Delayed neutrons and gammas by model.
applyDynTrclDP
Manipulate delayed-particle location and direction for the bank.
BANKIT
Bank delayed particles at emission time. Retrieve delayed particles for transport.
applyDynTrcl
Do dynamic transformation geometry operations.
applyDynTrclDP
Set the delayed-particle location and direction for the bank.
applyDynTrcl
Do dynamic transformation geometry operations.

**Figure 11**. MCNP6 code flow for moving objects involving neutron interactions driven through subroutine **colidn.F**. Boxes shown in blue are historical subroutines. Green boxes connote new subroutines for moving objects.

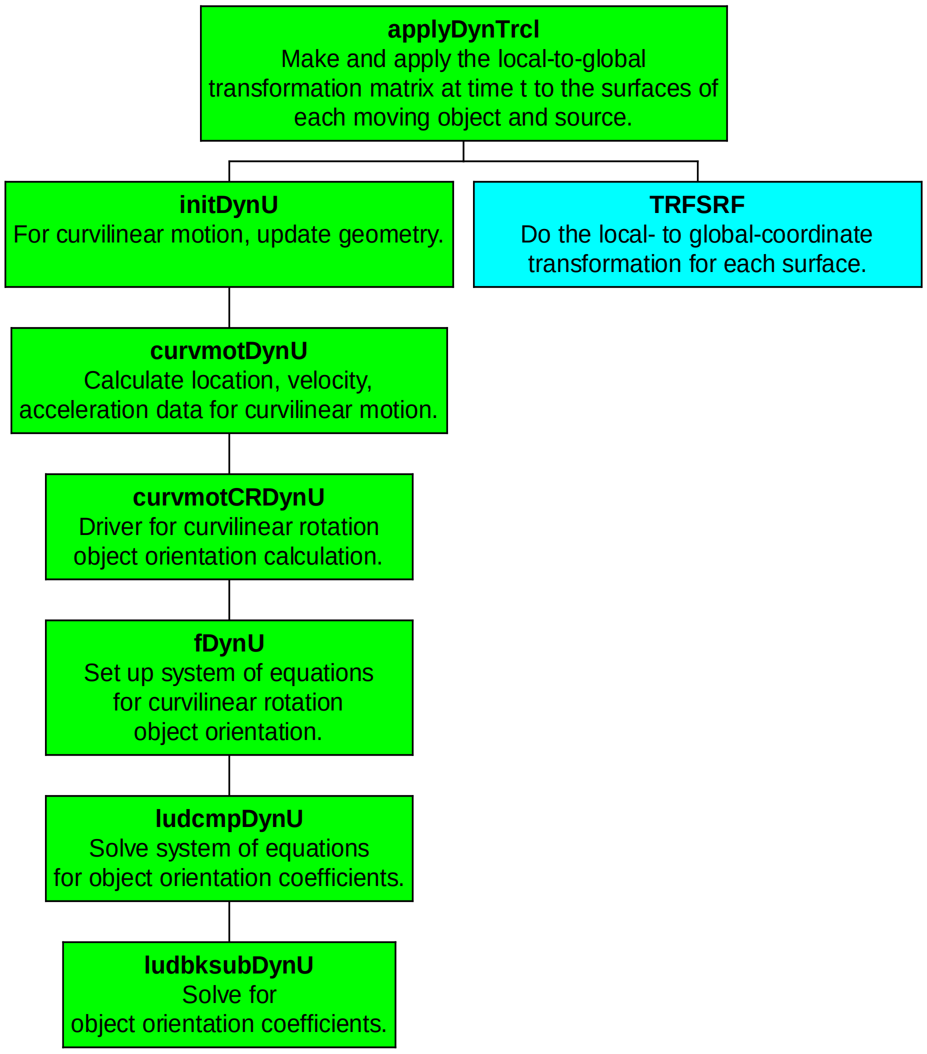
applyDynTrcl
Make and apply the local-to-global transformation matrix at time t to the surfaces of each moving object and source.
initDynU
For curvilinear motion, update geometry.
TRFSRF
Do the local- to global-coordinate transformation for each surface.
curvmotDynU
Calculate location, velocity, acceleration data for curvilinear motion.
curvmotCRDynU
Driver for curvilinear rotation object orientation calculation.
fDynU
Set up system of equations for curvilinear rotation object orientation.
ludcmpDynU
Solve system of equations for object orientation coefficients.
ludbksubDynU
Solve for object orientation coefficients.

**Figure 12**. MCNP6 subroutines involved in geometry updates, including curvilinear motion. Boxes shown in blue are historical subroutines. Green boxes connote new subroutines for moving objects.

**LIST OF TABLE CAPTIONS**

**LIST OF FIGURE CAPTIONS**

**Figure 1**. Coordinate-system rotation illustrated by global and local rectangular coordinate systems having the same origin.

**Figure 2.** Rectilinear translation.

**Figure 3**. Curvilinear translation.

**Figure 4**. Curvilinear rotation.

**Figure 5**. Locating and orienting an object and its path for rectilinear translation in MCNP6. At the start of the motion segment the object is located at $\vec{O}^G$ with its orientation specified in terms of the angles between its local-coordinate system and the MCNP6 global-coordinate axes. The object's path is prescribed using path coordinates ($x^P$, $y^P$), with motion in the $y^P$ direction for velocity $\vec{v}(t)$ and acceleration $\vec{a}(t)$. The path orientation is specified in terms of the angles between its path-coordinate system and the MCNP6 global-coordinate axes.

**Figure 6**. Curvilinear translation of an object about a fixed axis. The object's orientation remains fixed as it location changes from time $t_0$ to $t_e$. The orientation of the object and path are specified separately and can differ from each other, but both orientations are fixed in time.

**Figure 7**. Object curvilinear rotation about a fixed axis. The object's location and orientation changes from time $t_0$ to $t_e$. The orientation of the object and path are specified separately and can differ from each other. The object's orientation changes as the object moves. The path orientation is fixed in time.

**Figure 8**. Delayed-particle emission. A fission or activation (FA) event occurs time $t_0$ from within the object. The object's location and orientation changes from time $t_0$ to the time of delayed-particle emission, $t_e$.

**Figure 9**. General MCNP6 initialization code flow for moving objects. Boxes shown in blue are historical subroutines. Green boxes connote new subroutines for moving objects.

**Figure 10**. General MCNP6 code flow for moving objects involving source-particle creation, particle retrieval from bank, or **colidn.F** for neutron interactions. Boxes shown in blue are historical subroutines. Green boxes connote new subroutines for moving objects.

**Figure 11**. MCNP6 code flow for moving objects involving neutron interactions driven through subroutine **colidn.F**. Boxes shown in blue are historical subroutines. Green boxes connote new subroutines for moving objects.

**Figure 12**. MCNP6 subroutines involved in geometry updates, including curvilinear motion. Boxes shown in blue are historical subroutines. Green boxes connote new subroutines for moving objects.