JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, March 2004

Reference Frame Management and its Application to Reducing Numerical Error in Simulation

Satchidanand A. Kalaver[*] and Amy R. Pritchett[†]
Georgia Institute of Technology, Atlanta, Georgia 30332

This paper examines novel methods for the definition and management of reference frames in simulation software architectures. Reference frames are critical to aerospace dynamic models: they define the motion parameters of the vehicles and can affect model fidelity and numerical error within the simulation. This paper documents how reference frames can be defined as unique entities within the simulation software architecture, and how they can be centrally managed to create a shared simulation environment. The reference frame manager provides a central mechanism for transformations between reference frames and for selection of reference frames during runtime to meet fidelity requirements. The reference frame manager can also introduce intermediate frames that reduce numerical error by several orders of magnitude at very small time steps. This result can ensure a large degree of numerical accuracy in long-duration simulation runs, expanding the potential use of computer-based simulation in the aerospace industry.

Nomenclature

$\underline{X}$, $\underline{\dot{X}}$	=	state vector and time derivative of state vector
$\underline{X}_m$	=	vector of motion states
$\underline{P}$	=	position vector
$\underline{V}$	=	linear velocity vector
$\underline{\omega}$	=	angular velocity vector
Δt	=	time step
$\underline{Tr}_{A\rightarrow B}$	=	translation matrix from Frame A to Frame B
$\underline{R}_{A\rightarrow B}$	=	rotation matrix from Frame A to Frame B
$\underline{T}_{A\rightarrow B}$	=	homogenous transformation matrix from Frame A to Frame B
$\underline{F}$	=	forces on the vehicle
$\underline{M}$	=	moments on the vehicle
m	=	mass of body
$\underline{I}$	=	inertia tensor
$\underline{H}$	=	angular momentum
$\delta\underline{X}_{Tr}$	=	truncation error
$\delta\underline{X}_{Rnd}$	=	round off error
ε_m	=	machine accuracy

Presented as Paper 2002-4604 at the AIAA Modeling and Simulation Technologies Conference and Exhibit, Monterey, California, 5 August 2002. Received 6 June 2003; revision received 29 December 2003; accepted for publication 12 January 2004. Copyright © 2004 by Satchidanand A. Kalaver and Amy R. Pritchett. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/04 $10.00 in correspondence with the CCC.

[*]Graduate Student, School of Aerospace Engineering, 325171 Georgia Tech Station. AIAA Student Member.
[†]Associate Professor, School of Industrial and Systems Engineering, 765 Ferst Drive. AIAA Senior Member.

$\underline{Cr}$	=	critical levels used by intermediate frames
$\Delta\underline{X}_{Cr}$	=	round off error per time step due to propagation of position in intermediate frames
$\Delta\underline{X}_{U}$	=	round off error due to update of intermediate frames
$\Delta\underline{X}_{P}$	=	round off error due to propagation of intermediate frames
${}^{Pos}_{j}\Delta X_{Rnd}$	=	total round off error in the jth element of position
${}^{Vel}_{j}\Delta X_{Rnd}$	=	total round off error in the jth element of velocity
$k_{\Delta t}$	=	number of time steps
$k_{P,j}$, $k_{V,j}$	=	number of updates for jth element of position and velocity

Superscripts and Subscripts

${}^{A}_{j}X^{B}_{C}$	=	jth element of A with respect to B, measured in C
$b.f.$	=	body fixed frame
$b.c.$	=	body carried frame
n	=	navigation frame
i	=	inertial frame
IF	=	intermediate frame
V	=	vehicle

Introduction

Widely used in the aerospace industry, computer-based simulation can be tailored to specific applications, including design and evaluation of aerospace vehicles, pilot training, and the modeling of large systems such as air-traffic control. Dynamic models form the conceptual basis of aerospace simulations, propagating motion states such as position, orientation and velocity. These states are represented with respect to clearly defined reference frames, which historically have been fixed within the dynamic models.

The choice of reference frames is dependent upon the assumptions made in the dynamic model and the fidelity required from the simulation, where fidelity is an expression of how closely a dynamic model matches the vehicle behavior. In addition, the choice of reference frames affects the numerical error present in the simulation. Reference frames therefore need to be selected to meet fidelity and error requirements.

The ability to transform motion states between reference frames is critical because different components of a simulation, such as dynamic models and displays, often use different reference frames. Thus, a mechanism that handles the transformations between arbitrary reference frames will allow the various components to interact while eliminating the need for each component to store a large number of transformations. The software engineering benefits of such a mechanism can include rapid reconfiguration of a set of components for a large-scale simulation without needing to develop within each component the ability to perform transformations for all pairs of reference frames used by all the components.

Such a mechanism can be applicable to all types of simulation, but would especially benefit large-scale simulations such as air traffic simulations or distributed simulation in which many components need to share data. The importance of reference frames in distributed simulation is underscored by the research that has gone into the selection of common reference frames for large scale distributed simulations such as SIMNET [1, 2] and DIS. [3, 4]

This paper describes a mechanism, the reference frame manager or RFM, which allows reference frames to be de-coupled from the dynamic model within a simulator's software architecture. The RFM is responsible for all the reference frames in the simulation and creates a shared simulation environment, enabling reference frames to be selected by the components based on the fidelity and error requirements of the simulation. The RFM is able to manage an arbitrary number of reference frames, treating reference frames as individual entities that provide a standard set of motion parameters and transformations. Components of the simulation can then easily call the appropriate reference frames with access to the reference frames' parameters isolated to the RFM.

The RFM can also be used to generate 'intermediate frames' to reduce numerical error in the dynamic model. An intermediate frame can be assigned to follow each vehicle to act as its reference frame for navigation and can be positioned relative to the vehicle in such a way as to reduce numerical error. This will be especially beneficial to simulations that require a large number of iterations over long duration simulation run. While several methods already exist to reduce numerical error, the method described here does not require special hardware or operating systems, and does not require developers of simulation components to incorporate sophisticated error-reduction algorithms within their components.

This paper consists of four major sections. The first section provides a background review of key concepts: reference frames and their transformation, dynamic models and numerical errors. The second section is a conceptual treatment of reference frames in an extendable network and their use in reducing round off error. The third section deals with the implementation and validation of these concepts; the first portion of this section deals with the instantiation of reference frame management while the rest of the section describes testing, its results and the observations made. The final section concludes this paper and discusses the potential applications of reference frame management as well as areas of future research.

Background

Reference Frames in Simulation

A reference frame determines the origin and direction of measurement of the motion states of a dynamic model. The origin is the point from which the position is measured. The axes of the reference frame define the directions of measurement.

Common reference frames in aerospace simulations include body fixed frames, body carried frames, navigation frames and inertial frames.[5] The body fixed frame has its position and orientation fixed to the vehicle body. The body carried frame has its position fixed to the vehicle body and its orientation fixed to the navigation frame; this frame is typically used to determine the orientation, angular velocity and angular acceleration of the body fixed frame. The navigation frame is generally used to measure position and velocity of the vehicle. The inertial frame is the non-accelerating, non-rotating reference frame used for calculating the Newtonian equations of motion. The navigation frame may be fixed, rotating, accelerating or moving with respect to the inertial frame.

The *motion parameters* that define a reference frame commonly consist of position, orientation, linear velocity and angular velocity.[6] These properties are given with respect to another reference frame, known as the definition frame, and are measured in a measurement frame, which may be the reference frame in question, the definition frame, or a third reference frame. This paper uses a left superscript to identify the entity whose properties are being measured, the right superscript to identify the definition frame and the right subscript to denote the measurement frame. A left subscript can be used to identify a particular element of the motion parameter. Thus, ${}^{A}_{j}X^{B}_{C}$ represents the jth element of the motion parameters of A defined with respect to Frame B and measured in Frame C.

Transformation of Motion Parameters

Transformations of motion parameters are required to express them in a different reference frame; they may also be required when changing to a different definition frame. The types of transformations that are commonly used for converting motion parameters are translations, rotations and scaling. A translation involves the addition or subtraction of scalar values from the components of motion parameters without changing their direction. Scaling uses matrix multiplication and can result in either uniform or non-uniform scaling for the different components of the motion parameter. Rotation changes the direction of measurement of the motion parameters. These transformations can be represented by 3 dimensional matrix multiplication and addition or by 4 dimensional homogenous matrix multiplication for 3 dimensional motion parameters. [6, 7, 9]

Translations are typically represented as the vector addition of a motion parameter and a translation vector.[6,9] Both the motion parameters and translation vectors are represented as 3x1 column vectors for 3 dimensional motion parameters. For example, if the position expressed in Frame A is $(x_A, y_A, z_A)^T$ and requires a translation of $(\Delta x, \Delta y, \Delta z)^T$ to be expressed in Frame B $(x_B, y_B, z_B)^T$, then its transformation is

$$\underline{X}_B = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \underline{X}_A + \underline{Tr}_{A \to B} \tag{1}$$

Scaling and rotation are typically expressed as the matrix multiplication of a 3x3 scaling or rotation matrix and a 3x1 column vector representing the 3-dimensional motion parameter.[6,9] For example, if both frames share the same origin, the motion states expressed in Frame B can also be expressed in Frame C by using the rotation matrix shown here:

$$\underline{X}_C = \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \underline{R}_{B \to C} \underline{X}_B \tag{2}$$

The simplest rotation matrices represent the rotation of the frame about one of the X, Y or Z-axes and can be termed elementary rotations. Complex rotation matrices can be formed from multiple elementary rotations; however, the order of operations must be maintained, as the transformations are not commutative. If the axis of rotation does not pass through the origin, the origin must be translated to a point on the rotation axis. The rotation matrix used for reference frame transformations is typically generated using the relative orientation of the reference frames, which can be expressed as Euler angles or quaternions.

Transformation operations can also be expressed as homogenous 4x4 matrices.[7] Homogenous matrices have been used for transformations in kinematics of rigid bodies[8] as well as computer graphics.[9] In using homogenous matrices for the following rigid body transformations, the motion parameters are expressed as 4x1 column vectors where the first 3 elements are from the motion states and the 4th element is 1. Homogenous matrices allow all types of transformations to be expressed as matrix multiplications. For example, the translation shown in Eq. (1) can be expressed as:

$$\begin{bmatrix} \underline{X}_B \\ 1 \end{bmatrix} = \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} = \underline{T}_{A \to B} \begin{bmatrix} \underline{X}_A \\ 1 \end{bmatrix} \tag{3}$$

Likewise, the rotation shown in Eq. (2) can be expressed as:

$$\begin{bmatrix} \underline{X}_C \\ 1 \end{bmatrix} = \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} = \underline{T}_{B \to C} \begin{bmatrix} \underline{X}_B \\ 1 \end{bmatrix} \tag{4}$$

Thus, if the motion states need to be transformed from Frame A to Frame C using the transformations in Eqs. (3) and (4), the entire transformation can be expressed as a single transformation matrix as follows:

$$\begin{bmatrix} \underline{X}_C \\ 1 \end{bmatrix} = \underline{T}_{B \to C} \underline{T}_{A \to B} \begin{bmatrix} \underline{X}_A \\ 1 \end{bmatrix} = \underline{T}_{A \to C} \begin{bmatrix} \underline{X}_A \\ 1 \end{bmatrix} \tag{5}$$

Dynamic Models of Vehicles

Dynamic models capture the time-varying physical properties of vehicles in a state vector. The particular elements of the state vector, $\underline{X}$, that represent the motion properties of the vehicle such as vehicle position, orientation and velocity will be referred to as the vector of motion states $\underline{X}_m$.

The motion states of aerospace vehicles are commonly expressed using the navigation frame, the body carried frame and the body fixed frame. The vehicle position is commonly expressed as the position of the body carried frame with respect to the navigation frame and the vehicle velocity is commonly expressed as the velocity of the body fixed frame with respect to the navigation frame. The body fixed frame maintains a constant orientation relative to the body and provides a measure of orientation about the vehicle center of mass when compared to the body carried frame. It is a convenient reference frame in which to express many forces and moments generated on the body. It can further be transformed to reference frames common in the calculation of aerodynamic effects, such as stability and wind axes. Thus, $\underline{X}_m$ can often be abstracted as the motion properties of these reference frames.

Dynamic models are typically represented by differential equations that generate the time derivatives for the state vector as functions of the state $\underline{X}$, outside influences and controls $\underline{u}$, and time t in the form of

$$\underline{\dot{X}} = \underline{f}\left\{\underline{X}, \underline{u}, t\right\} \tag{6}$$

While it is common for these differential equations to be linearized or simplified in methods seeking analytical solutions, their complete non-linear form typically comprises the vehicle dynamic model in computer-based simulation.

A subset of the dynamic model provides the time derivatives for $\underline{X}_m$ through the application of Newton's Second Law, which relate forces and moments to the time derivatives of linear and angular velocity.[6,10] Likewise, kinematics relations are used to relate linear and angular velocity to the time derivatives of position and orientation. The exact form of these equations depends on the reference frames in which the derivatives are taken as well as the measurement and definition frames of the motion states.

One case where the dynamics are impacted by the choice of reference frames comes in applying Newton's Second Law.[6,10] This can, in theory, be represented by defining a purely inertial frame. In practice, it is difficult to define a reference frame that is not accelerating with respect to inertial space. For example, an Earth-fixed reference frame may be suitable for some low fidelity situations; conversely, in high fidelity situations the rotation and translation of the Earth needs to be accounted for. Therefore, different simulations (or phases of a single simulation) may require different inertial reference frames for the fidelity requirements at hand, even when they are using the same dynamic model.

Numerical Integration and Truncation Error

Time, which is typically considered a continuous entity, is discretized into discrete time steps in numerical simulation. At each time step of duration Δt, numerical integration is used to propagate $\underline{X}$ forward by using the time derivative provided by the dynamic model. In general, numerical integration routines calculate the value of $\underline{X}$ at the (n+1)th time step from the values of $\underline{X}$ at the nth time step as follows:

$$\underline{X}\left\{n+1\right\} = \underline{X}\left\{n\right\} + \Delta\underline{X}\left\{n, n+1\right\} \tag{7}$$

The incremental term $\Delta\underline{X}\left\{n, n+1\right\}$ represents the change in value estimated over the interval $\left\{n, n+1\right\}$. Using simple lower order numerical integration methods, such as First Order Forward Euler, this incremental term may be an approximation such as the product of $\underline{\dot{X}}\left\{n\right\}$ and Δt. Higher order methods, such as higher-order Runge-Kutta (RK) series,[11] express the derivative function as a power series for a more accurate estimate of the incremental term.

These numerical integration methods, by approximating continuously evolving dynamics to discrete increments of time, can incur a form of numerical error termed truncation error. The magnitude of this truncation error $\Delta\underline{X}_{Tr}$ per time step is impacted by the order of the method, which is commonly the number of terms from the power series used in estimating the incremental term: [11]

$$\left|\Delta\underline{X}_{Tr}\right| \approx O\left(\Delta t\right)^{order+1} \tag{8}$$

There are two standard methods for reducing truncation error. The first method is to use higher order numerical integration routines.[11] The other method is to reduce Δt so that the derivatives are propagated over smaller intervals; however, this requires more iterations for a given duration of simulated time, and thus more opportunities to accumulate truncation error.

Floating Point Variables and Round Off Error

Round off error is generated by the finite precision of floating point variables used in most types of computer systems. Floating-point numbers are represented by a positive integer called a mantissa *M*, a positive integer *e*, and a sign bit.[11] The sign bit determines if the number is positive or negative. The mantissa is stored in a binary form that consists of *N* bits. A bias *E*, predefined in the machine architecture, is subtracted from *e* so as to create an exponent capable of holding positive and negative values. This eliminates the need for an additional sign bit to determine the sign of the exponent. The base value *B*, typically 2 or 16, is raised to power of the exponent (*e-E*). This value is then multiplied by *M* to generate a positive floating-point number as follows:

$$M \cdot B^{e-E} \tag{9}$$

The sign represented by the sign bit is then assigned to the value obtained by Eq. (9). The memory allocation for the mantissa and exponent can vary with the computer architecture. A 32-bit floating-point number typically contains a sign bit, 23 bits for the mantissa and 8 bits for the exponent.[11]

The machine round off error is determined by the machine accuracy, ε_m, often called the precision of the variable.[11] The machine accuracy is defined as the smallest value that can be added to 1.0 without being lost. This value is 1.19×10^{-7} for 32 bit single precision and 2.22×10^{-16} for 64 bit double precision floating point numbers for IEEE Standard 754 compliant machines.

There are 2 ways of calculating the magnitude of the maximum round off error $\delta\underline{X}_{Rnd}$. The first method provides an easy approximation:

$$\delta\underline{X}_{Rnd} \approx \underline{X} \cdot \varepsilon_m \tag{10}$$

The second, exact method requires bit-wise analysis. The Most Significant Bit (*MSB*) represents the first bit in the number's mantissa and has the largest exponent while the Least Significant Bit (*LSB*) represents the last bit in the mantissa and has the smallest exponent. The maximum value represented by the *LSB* gives the maximum possible round off error. The difference in exponents of the *MSB* and *LSB* is (*N-1*). The exact value for $\delta\underline{X}_{Rnd}$ is

$$\delta\underline{X}_{Rnd} = 2^{(int)\,log_2(\underline{X}) - N + 1} \tag{11}$$

In addition to the error generated by the bit-wise representation of individual floating point numbers, arithmetic operations on values with different exponents also generate round off error. When two floating point numbers with different exponents are added or subtracted, some or all of the data contained in the number with the smaller exponent will be lost. If A has a larger exponent than B, addition or subtraction operations between A and B will set the value of B's exponent to the value of A's exponent. This is achieved by 'right shifting' the bits in the mantissa of B such that B's new LSB and A's LSB have the same exponent. Thus, data originally contained in the mantissa of the B whose exponents are less than the exponent of the new LSB are lost. Therefore, $\delta\underline{X}_{Rnd}$ of A forms the upper bound for the round off error generated during each addition or subtraction of A and B.

To illustrate the effect of round off errors, assume that an 8-bit floating-point number has 4 bits in the mantissa and 3 bits in the exponent with a bias of four and a base of two. In expressing the number 115 in binary [1110011], the value stored in the mantissa is [1110] and the last 3 bits are lost. Thus, the LSB has an exponent of 3 and the maximum round off error is 8. The actual value stored is 112 and the actual round off error is 3, corresponding to the 3 least significant bits, [011], that were dropped off.

Extending this example to arithmetic operations, let an arbitrary state X = 44, and its incremental term for a given time step $\dot{X} \bullet \Delta t_1$ = 7, be represented using the same 8-bit floating-point number. Figure 1 illustrates the loss of bits in the incremental term whose exponents are smaller than the LSB of the state. For X, the mantissa is 11 or [1011] in binary and the exponent of the LSB is 2. For $\dot{X} \bullet \Delta t_1$, the mantissa is 14 or [1110] and the exponent of the LSB is –1. When X and $\dot{X} \bullet \Delta t_1$, are added, the bits of $\dot{X} \bullet \Delta t_1$ that have an exponents less than the LSB of X are dropped. Thus, the new $\dot{X} \bullet \Delta t_1$ has a mantissa of 1 or [0001] and an exponent of 2. The result of the addition using the floating-point arithmetic is 48 compared to the correct result of 51. The $\Delta\underline{X}_{Rnd}$ due to X is the value of its LSB: 2^2 or 4. However, the round off error is 3, demonstrating that $\Delta\underline{X}_{Rnd}$ forms the upper bound on round off error.

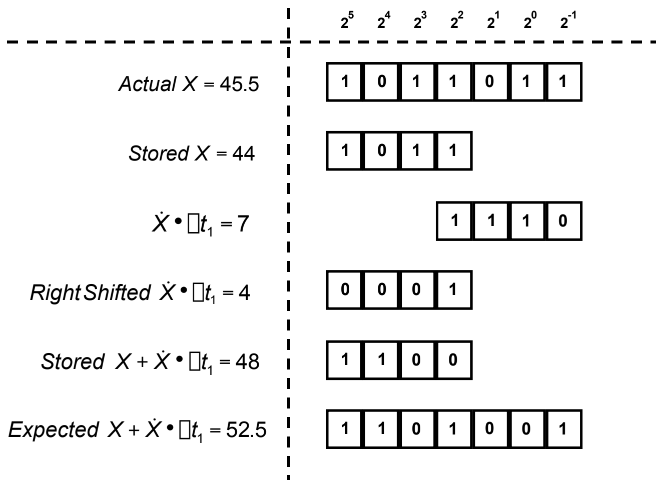
2^5 2^4 2^3 2^2 2^1 2^0 2^{-1}
Actual X = 45.5
1 0 1 1 0 1 1
Stored X = 44
1 0 1 1
$\dot{X} \bullet \Delta t_1 = 7$
1 1 1 0
Right Shifted $\dot{X} \bullet \Delta t_1 = 4$
0 0 0 1
Stored $X + \dot{X} \bullet \Delta t_1 = 48$
1 1 0 0
Expected $X + \dot{X} \bullet \Delta t_1 = 52.5$
1 1 0 1 0 0 1

Fig. 1 Bit wise representation of addition for floating point variables.

Numerical Error in Simulation

The truncation and round off errors added to the motion states at each time step will accumulate and compound in subsequent time steps. In many dynamic models, the derivative $\dot{\underline{X}}$ is a function of the state vector $\underline{X}$. Thus, any error in $\underline{X}$ will generate an error in $\dot{\underline{X}}$, which in turn will introduce additional errors when used to propagate $\underline{X}$. This allows the error to grow rapidly and, since states are typically coupled, to propagate into all aspects of the vehicle dynamics.

Numerical error can have two practical impacts of simulation use. First, they may reduce a simulation's accuracy in even short runs; while historically an issue, problems with accuracy in short duration runs are now limited to simulations of the most detailed dynamics. Second, given how accumulation of numerical error is propagated back into vehicle dynamics, it can limit the duration of simulation runs. For example, it is currently problematic to simulate spacecraft in interplanetary trajectories that end in a docking operation. Reducing numerical error, then, may enable longer duration simulation runs than commonly used to date.

The standard method for reducing truncation error for a given integration routine is to reduce the time step. However, reducing the time step creates a larger number of iterations over a given duration of simulated time where both truncation and round off error may be accumulated.

In addition, smaller time steps have the opposite effect on round off error: reducing the size of the incremental term implies that a larger fraction of the incremental term can be lost per iteration. Figure 2 illustrates this concept with a schematic representation of truncation and round off error as a function of time step and highlights how total numerical error at every time step (the sum of truncation and round off error) can be very large if the time step is made too small or too large. Therefore, it is not possible to reduce both types of error by changing only the time step. Instead, another method needs to be developed that will allow the integration method to control both the truncation error and round off error.

While it is possible to reduce round off error by using algorithms that allow arithmetic to arbitrary precision, such as using higher bases and Fast Fourier Transforms, these often require specialized expertise on the part of the developer, and can create software specific to specific types of problems. Likewise, the use of wider registers (or full use of registers) can reduce round off error, but creates software that is specific to certain hardware architectures and operating systems. Ideally, a method that is transparent to the developer of simulation components would facilitate control of numerical error in a manner that facilitates use of those components in a wide variety of simulator configurations and on a range of computer architectures.

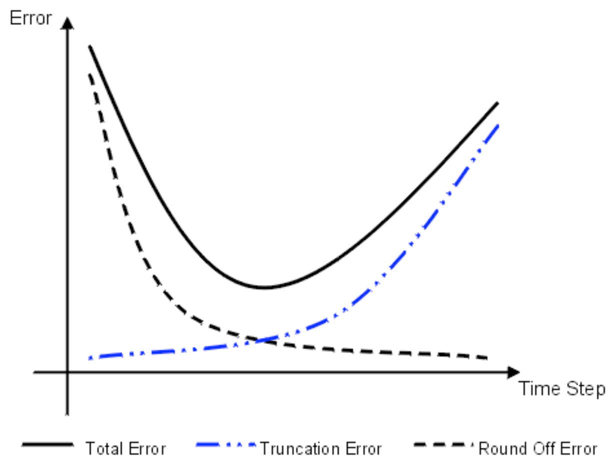

Fig. 2 Schematic of effect of time step on truncation, round off, and total error.

Managing Reference Styles

Representation of Reference Frames

Rather than use the current method of keeping reference frames implicit within the dynamic model's calculations, this paper proposes using a standard explicit representation of reference frames. This representation enables the formation of a network of reference frames within a simulator's software architecture and their systematic selection and transformation. As discussed in the next section, this representation of reference frames will also allow intermediate frames to be created to reduce numerical error in simulation. A reference frame can be treated as a unique entity in a simulation whose motion parameters are measured with respect to its definition frame. Each reference frame should be able to transform motion parameters to and from its definition frame or provide the corresponding transformation matrices, thus releasing these functions from the simulation components themselves. In cases where the motion parameters and their respective transformations are time varying, some of the motion parameters will need to be mathematical expressions or equations instead of constant values, or will need to be propagated using numerical integration.

Formation of a Network of Reference Frames

A network of reference frames can be created by viewing each reference frame as a node and the transformation between reference frames as the links between the nodes. Once such a network is established, motion states and parameters in one reference frame can be transformed into any other reference frame. The simplest form of a network is an extensible tree, where the motion parameters and transformations of each reference frame are given with respect to their definition frame. Each definition frame can be considered to be higher up in the tree than the reference frames using it. Several frames branch from each node when they use the same definition frame. A node from which the other reference frames branch is called the shared node and the reference frame represented by the node is the shared frame. In theory, there is no limit on the size or the number of branches in the tree.

To ensure that all the reference frames loaded into the tree can be linked, a 'global' definition frame should be selected for any simulation run. It serves as the highest node in the tree and all the reference frames used in the simulation will ultimately lead to it. This is similar to the use of a global frame in DIS that ensures all components are able to communicate with one another. However, the global frame in DIS is fixed and each reference frame is responsible for transforming its parameters to and from the global frame. In contrast, the global frame in the network of reference frames is set at runtime and can be changed for different simulation scenarios. Furthermore, the transformations are handled by the network of frames rather than the other components in the simulation such as dynamic models, visual displays, etc.

There is directionality associated with the links between the reference frame and its definition frame, as each reference frame is only responsible for the transformations between itself and its definition frame. This means that any given frame is not 'aware' of the frames that may be using it for their definition frames. An external Reference

Frame Manager (RFM) maintains the tree and provides transformations between the reference frames on request from simulation components.

Transformations in a Network of Reference Frames

With this representation of a network of reference frames, a transformation path is the sequence of reference frames that join two specified reference frames. If the transformation path is not available, a search algorithm creates it as follows. Since the link between each reference frame and its definition frame is only in one direction, the search algorithm creates two search paths. The *forward path* starts from the current reference frame, while the *reverse path* starts from the desired reference frame. Both paths follow the links from the reference frames to their definition frames until a common definition frame is found. Then, the forward and reverse paths are merged. For example, the forward and reverse paths for the transformation path in Fig. 3 can be expressed as:

Forward Path:	A→ B→ C→ F
Reverse Path:	D→ E→ F
Transformation Path:	A→ B→ C→ F→ E→ D

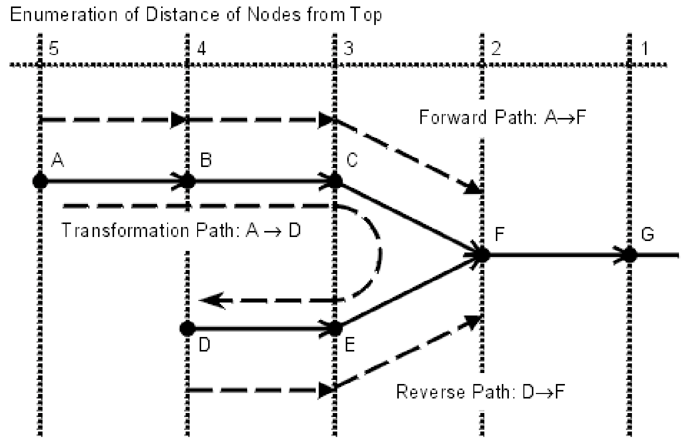
Enumeration of Distance of Nodes from Top
5
4
3
2
1
Forward Path: A→F
A
B
C
Transformation Path: A → D
F
G
D
E
Reverse Path: D→F

Fig. 3 Forward and reverse paths on a network of enumerated nodes.

One method that can be used to check for a shared node enumerates the distance of the nodes from the top of the tree. The search algorithm starts with the start and end points and compares their enumeration. The path containing the node with the higher enumeration is traversed towards the top of the tree until both paths are examining nodes with the same enumeration. If these nodes are the same, a shared node has been reached. If the nodes are different, both paths jointly step higher until a shared node is reached. In the worst case, the shared node will be the 'global' definition frame at the top of the tree.

Once the transformation path is obtained, the transformations between each reference frame are assembled into the single transformation between the starting and ending reference frames. Because of the directionality of the tree, inverse transformations are used for the links on the reverse path. For example, in Fig. 3, the complete transformation can be expressed in a homogenous transformation matrix as

$$T_{A\to D} = \left[T_{D\to E}\right]^{-1} \left[T_{E\to F}\right]^{-1} T_{C\to F} T_{B\to C} T_{A\to B} \tag{12}$$

Effect of Reference Frame Management on Dynamic Modeling

Reference frame management shifts the capability to transform motion parameters out of simulation components and onto reference frames acting as independent entities in the simulation. Thus, the instantiation of the dynamic model is not fixed to one set of reference frames; instead, it can be easily adapted to the immediate situation. For example, a common expression of the equations of motion assumes that the forces and moments on the vehicle, $\underline{F}$ and $\underline{M}$, are expressed in the body fixed frame, derivatives of linear and angular velocity, ${}^{b.f.}\dot{\underline{V}}^{n}_{b.f.}$ and ${}^{b.f.}\dot{\underline{\omega}}^{n}_{b.f.}$, are also taken in the body fixed frame, and the navigation frame is not inertial but instead has an angular velocity with respect to the inertial frame ${}^{n}\underline{\omega}^{i}_{n}$.[5,6,10] The equations of motion in this condition are

$$ {}^{b.f.}\dot{\underline{V}}^{n}_{b.f.} = \frac{\sum \underline{F}_{b.f.}}{m} - \underline{R}_{n \to b.f.} \bullet {}^{apparent}\underline{a}_{n} - {}^{apparent}\underline{a}_{b.f.} \tag{13} $$

$$ {}^{b.f.}\dot{\underline{\omega}}^{n}_{b.f} = \underline{I}^{-1} \bullet \left[\sum \underline{M}_{b.f.} - \left({}^{b.f.}\underline{\omega}^{i}_{b.f.} \times {}^{body}\underline{H}^{i}_{b.f.} \right) \right] - {}^{n}\dot{\underline{\omega}}^{i}_{b.f.} \tag{14} $$

where

$$ {}^{apparent}\underline{a}_{n} = {}^{n}\ddot{\underline{P}}^{i}_{n} + \left({}^{n}\dot{\underline{\omega}}^{i}_{n} \times {}^{b.c.}\underline{P}^{n}_{n} \right) + \left[{}^{n}\underline{\omega}^{i}_{n} \times \left({}^{n}\underline{\omega}^{i}_{n} \times {}^{b.c.}\underline{P}^{n}_{n} \right) \right] + 2\left({}^{n}\underline{\omega}^{i}_{n} \times {}^{b.c.}\underline{V}^{n}_{n} \right) \tag{15} $$

$$ {}^{apparent}\underline{a}_{b.c.} = {}^{b.f.}\underline{\omega}^{n}_{b.f.} \times {}^{b.f.}\underline{V}^{n}_{b.f.} \tag{16} $$

$$ {}^{b.f.}\underline{V}^{n}_{n} = \underline{R}_{b.f. \to n} \bullet {}^{b.f.}\underline{V}^{n}_{b.f.} \tag{17} $$

$$ {}^{V}\underline{H}^{i}_{b.f} = {}^{V}\underline{I}_{b.f.} \bullet {}^{b.f.}\underline{\omega}^{i}_{b.f.} \tag{18} $$

$$ {}^{b.f.}\underline{\omega}^{i}_{b.f.} = {}^{b.f.}\underline{\omega}^{n}_{b.f.} + {}^{n}\underline{\omega}^{i}_{b.f.} \tag{19} $$

$$ {}^{n}\underline{\omega}^{i}_{b.f.} = \underline{R}_{n \to b.f.} \bullet {}^{n}\underline{\omega}^{i}_{n} \tag{20} $$

$$ {}^{n}\dot{\underline{\omega}}^{i}_{b.f.} = \underline{R}_{n \to b.f.} \bullet {}^{n}\dot{\underline{\omega}}^{i}_{n} \tag{21} $$

Using the RFM, the dynamic model only needs to internally calculate external forces $\underline{F}_{b.f.}$ and moments $\underline{M}_{b.f.}$ acting on the vehicle. The RFM can supply the motion parameters of the navigation frame with respect to the inertial frame, ${}^{n}\ddot{\underline{P}}^{i}_{n}$, ${}^{n}\underline{\omega}^{i}_{n}$ and ${}^{n}\dot{\underline{\omega}}^{i}_{n}$, as well as the transformations $\underline{R}_{n \to b.f.}$ and $\underline{R}_{b.f. \to n}$. This enables the dynamic model to compensate for the motion of the body and navigation frames for any set of navigation and inertial frames, thereby allowing the model to change its navigation and inertial frames as dictated by the fidelity requirements of the simulation without changing its equations of motion. The RFM could also be supplied with the inertial properties of the vehicle and calculate the kinematics due to the motion of navigation frame with respect to the inertial frame.

Managing Round Off Error

Definition of Intermediate Frames

Numerical error can be reduced by using the RFM to introduce intermediate frames that act as surrogates to navigation frames. The motion parameters of the intermediate frame are measured with respect to the original navigation frame, allowing motion states expressed in the intermediate frame to be transformed to the original navigation frame and vice versa.

The intermediate frame is defined such that the difference in exponents between motion states of the vehicle expressed in the intermediate frame and their incremental terms during numerical integration will be bounded, thus bounding round off error. The intermediate frame is updated when any element j of the vehicle's motion states reaches its critical level ${}_{j}\underline{Cr}$; at that point, the corresponding element in the intermediate frame's motion parameters is shifted towards the vehicle by the value of that critical level.

To best reduce round off error in position and linear velocity, the intermediate frames should have the same orientation as the navigation frame. If they have different orientations, the transformation between them may require floating point operations across all axes. Furthermore, an angular velocity between these frames will lead to dynamic transformations. Both these operations may generate additional round off error.

Role of Critical Levels in Intermediate Frames

As a motion state is propagated, it may grow to the point that its magnitude will be much larger than the incremental term. If the ratio of the incremental term to the motion state approaches machine accuracy, significant round off errors will occur. The critical value, therefore, is set to bound the magnitude of a subset of the motion states relative to their incremental terms at each time step. Specifically, if the element j in the vehicle's motion states expressed and measured relative to the intermediate frame, ${}^{V}_{j}\underline{X}^{IF}_{IF}$, exceeds ${}_{j}\underline{Cr}$, its critical level, the corresponding element in the motion parameters of the intermediate frame, ${}^{IF}_{j}\underline{X}^{n}_{n}$, must be updated in a discrete 'jump'. This updates both the motion state of the vehicle as well as the intermediate frame as follows:

If $\left|{}^{V}_{j}\underline{X}^{IF}_{IF}\right| \geq {}_{j}\underline{Cr},$

$$ {}^{IF}_{j}\underline{X}^{n}_{n} = {}^{IF}_{j}\underline{X}^{n}_{n} + {}_{j}\underline{Cr} \;\square \operatorname{sgn}\left({}^{V}_{j}\underline{X}^{IF}_{IF}\right) \tag{22} $$

$$ {}^{V}_{j}\underline{X}^{IF}_{IF} = {}^{V}_{j}\underline{X}^{IF}_{IF} \square {}_{j}\underline{Cr} \;\square \operatorname{sgn}\left({}^{V}_{j}\underline{X}^{IF}_{IF}\right) \tag{23} $$

The critical level is not a constant value; instead, it can be selected to fit the requirements of the simulation. The upper limit of $\underline{Cr}$ is a function of the maximum allowable round off error, $\square\underline{X}_{Rnd}$, per time step:

$$ \underline{Cr} < \frac{\square\underline{X}_{Rnd}}{\square_{m}} \tag{24} $$

The lower limit of the critical level depends upon 2 distinct factors. The first is the round off error due to the motion parameters of the intermediate frame, $\left|{}^{IF}_{j}\underline{X}^{n}_{n} \,\square\square_{m}\right|$, and represents the smallest value by which the intermediate frame can be updated without its update being lost to round off error. The other factor is the magnitude of the incremental term, $\left|{}^{V}_{j}\dot{\underline{X}}^{IF}_{IF} \,\square\square t\right|$; if the incremental term is larger than the critical level, the intermediate frame will be updated at every time step and incur even larger round off errors than the vehicle. The larger of these values is used for the lower limit:

$$ \underline{Cr} > \max \square \left|{}^{V}_{j}\dot{\underline{X}}^{IF}_{IF} \,\square\square t\right|, \left|{}^{IF}_{j}\underline{X}^{n}_{n} \,\square\square_{m}\right| \square \tag{25} $$

Estimation of Round Off Error for Intermediate Frames

In a conventional simulation, there is a single source of round off error, which can be bounded as given by Eqs. (10) and (11). The use of intermediate frames reduces the magnitude of this error per time step while introducing two additional sources of error: the round off error per update of the intermediate frame upon reaching a critical level, and the round off error due to the propagation of the intermediate frame if it is moving with respect to the navigation frame. Thus, for the use of intermediate frames to be successful, the parameters that govern all these three sources of error must be selected carefully.

The first source of error when using intermediate frames is identical to conventional round off error due to the propagation of motion states. However, the round off error per time step is created by the vehicle's motion states expressed in the intermediate frame. Because the critical level bounds the maximum value of the motion states, their round off error per time step is bounded; their maximum round off error per time step, $\square\underline{X}_{Cr}$, is a function of $\underline{Cr}$:

$$\square \underline{X}_{Cr} = 2^{(\text{int})\log_2(\underline{Cr}) \square N+1} \square \underline{Cr} \square \square_m \tag{26}$$

The second source of error is incurred in updating the intermediate frames. Because the 'jump' consists of adding the critical value to the motion parameter of the intermediate frame, the round off error per update, $\square \underline{X}_U$, depends upon the magnitude of the motion parameter of the intermediate frame, ${}^{IF}\underline{X}_n^n$:

$$\square \underline{X}_U = 2^{(\text{int})\log_2\left(\left|{}^{IF}\underline{X}^n\right|\right) \square N+1} \square {}^{IF}\underline{X}_n^n \square \square_m \tag{27}$$

The third source of error occurs when the intermediate frame translates with respect to the navigation frame, requiring its position to be propagated and thus incurring round off errors. The accumulation of this error depends upon the specific implementation of the intermediate frame. If the intermediate frame only tracks the position and linear velocity of a vehicle, the intermediate frame's velocity only changes in discrete 'jumps' and is not directly affected by this error term, while the position of the intermediate frame accumulates error at each time step if propagated through numerical integration. On the other hand, if the position of the intermediate frame is determined analytically from velocity, round off error is induced only when its velocity is updated. This error term, $\square \underline{X}_P$, depends upon ${}^{IF}\underline{X}_n^n$:

$$\square \underline{X}_P = 2^{(\text{int})\log_2\left(\left|{}^{IF}\underline{X}_n^n\right|\right) \square N+1} \square {}^{IF}\underline{X}_n^n \square \square_m \tag{28}$$

The maximum upper bound for round off error in a given simulation run can be estimated by taking the sum of these three error terms. Their computation requires knowledge of the number of time steps and updates. If the intermediate frame's position is expressed analytically as a function of velocity, the maximum round off error for the *j*th elements of position and velocity can be calculated as follows. For $k_{\square t}$ time steps and $\underline{k}_{P,\,j}$ and $\underline{k}_{V,\,j}$ updates for the *j*th elements of position and velocity of the intermediate frame respectively, the maximum upper bound of the round off error for the *j*th element of position, ${}^{Pos}_{j}\square \underline{X}_{Rnd}$, and velocity, ${}^{Vel}_{j}\square \underline{X}_{Rnd}$, can be expressed as:

$${}^{Pos}_{j}\square \underline{X}_{Rnd} = {}^{Pos}_{j}\square \underline{X}_{Cr} \square k_{\square t} + {}^{Pos}_{j}\square \underline{X}_U \square \underline{k}_{P,j} + {}^{Pos}_{j}\square \underline{X}_P \square \underline{k}_{V,j} \tag{29}$$

$${}^{Vel}_{j}\square \underline{X}_{Rnd} = {}^{Vel}_{j}\square \underline{X}_{Cr} \square k_{\square t} + {}^{Vel}_{j}\square \underline{X}_U \square \underline{k}_{V,j} \tag{30}$$

Observing Eqs. (29) and (30), a small critical level corresponding to a vehicle motion state with a large time derivative will cause the intermediate frame to jump frequently, increasing the round off error due to updates, $\square \underline{X}_U$, and propagation, $\square \underline{X}_P$, but reducing the error per time steps, $\square \underline{X}_{Cr}$. Conversely, large critical levels will reduce the number of updates at the expense of increasing round off error per time step.

Selection of Critical Levels to Reduce Errors

The critical levels should be chosen so as to control all three error terms contributing to ${}^{Pos}_{j}\square \underline{X}_{Rnd}$ and ${}^{Vel}_{j}\square \underline{X}_{Rnd}$ in Eqs. (29) and (30). The first, $\square \underline{X}_{Cr}$, corresponds to the magnitude of the critical levels as seen in Eq. (26). To minimize the second, $\square \underline{X}_U$, a bit-wise analysis of the critical level is required. If the critical level, expressed in binary, is set to have a single bit in the mantissa with an exponent equal to or larger than the exponent of the LSB in the motion parameter, no bits are lost during the update of the intermediate frame's motion parameter, eliminating the $\square \underline{X}_U$ term. The third term, $\square \underline{X}_P$, can be reduced by ensuring an analytical implementation for the intermediate frame's velocity and by reducing the number of updates of the intermediate frame. However, the number of updates is approximately inversely proportional to the magnitude of the critical levels. Thus, a trade off between $\square\underline{X}_P$ and $\square\underline{X}_{Cr}$ is necessary, and minimizing the bounded error may depend on specific attributes of the vehicle's dynamics.

As a general rule, if the intermediate frame does not frequently change its velocity with respect to the navigation frame, $\square\underline{X}_P$ is eliminated and the best critical level is the smallest bit in the mantissa of the motion parameter of the intermediate state. If the intermediate frame's velocity with respect to the navigation frame changes frequently, $\square\underline{X}_P$ cannot be ignored and larger critical levels will have to be chosen for position, either heuristically or analytically.

Implementation of the Reference Frame Manager

A reference frame management system has been implemented using object oriented analysis and design principles within the simulation architecture provided by the Reconfigurable Flight Simulator (RFS).[12] For this implementation, each reference frame was represented as a Reference Frame Object. Intermediate frames were represented by Intermediate Frame Objects, which were derived from Reference Frame Objects and were capable of monitoring their critical levels and updating themselves accordingly. The reference frame management mechanism was implemented as a Reference Frame Manager (RFM). An Intermediate Frame Manager (IFM) was created for the sole purpose of creating and destroying Intermediate Frame Objects as commanded by the RFM.

The RFM provides three major operations. The first is loading the appropriate Reference Frame Objects into RFS and registering them with the RFM. To form a single network, a standard set of linked Reference Frame Objects needs to be provided to the RFM, or the user has to ensure that all the Reference Frame Objects loaded into RFS are linked (either directly or indirectly) to the highest node in the tree, i.e., some global frame.

The second operation is the creation of transformation paths during runtime. When any simulation component requires a transformation of motion states, it passes a request structure to the RFM. This structure includes the names of the current and requested frames, as well as the motion states and pointers to the return values. The RFM evaluates the current and requested frames and checks if the path connecting these frames exists. If the path does not exist, it is created and stored in the RFM for future use. The Reference Frames Objects in the transformation path calculate the individual transformations to or from their respective definition frames. The RFM collects these individual transformations to generate the required transformation, which is then applied to the motion states to create the return value. This is similar to the matrix stack used by OpenGL to handle the sequence of transformations used in the model matrix and projection matrix: the transformation matrix is 'pushed' when the RFM collects a transformation from a reference frame and adds it to the transformation matrix. However, the simulation components do not themselves need to specify, store or access the specific translation, rotation and scaling parameters needed for the transformation, nor do they need to each contain the logic creating the appropriate sequence of simple operations required for a complex transformation.

The third and final operation is the creation of intermediate frames. If a vehicle requests an intermediate frame to reduce round off error, the RFM calls the Intermediate Frame Manager (IFM). The IFM creates and initializes a new intermediate frame, assigns a name to it and returns its pointer to the RFM. The RFM adds the pointer of the intermediate frame to its list of registered frames and returns the name of the intermediate frame to the vehicle. The motion states of the vehicle are also transformed to the intermediate frame. Once a dynamic model component is compliant with the interface standards for using intermediate frames, a wide variety of intermediate frames can be chosen in run-time to suit the immediate needs of any simulation run without requiring changes to the dynamic model's software.

Testing of the Reference Frame Manager and Intermediate Frames

A series of simulation runs were carried out to test the RFM and the use of intermediate frames to reduce round off errors. The simulation scenario involved the use of several reference frame objects including an intermediate frame object for the reduction of round off errors. Since the intermediate frame object relies on the RFM to generate the transformations between reference frames, successful operation of the intermediate frame also demonstrated the RFM. The effectiveness of the intermediate frames was evaluated by comparing the simulated results with analytical expressions of motion states to find the error in the simulation with and without intermediate frames.

Test Scenario

The test scenario consisted of a spacecraft in an elliptical orbit around the Sun. A major assumption was that the only force on the spacecraft was the gravitational force of the Sun, allowing for an analytical solution of the position and velocity of the spacecraft as a benchmark against which numerical errors could be ascertained. Furthermore, the parameters of the orbit were taken so that the ratio of the motion states to their derivatives would be large, magnifying the round off error and allowing the effects of the intermediate frame to be noted. To this end, the spacecraft's orbit was given a semi major axis of 40 Astronomical Units (AU) or 5.9×10^{12} meters and an eccentricity of 0.2583, approximately the parameters of Pluto's orbit.[13] The orbit was assumed to lie along the X-Y plane of the Heliocentric Frame, allowing the motion to be treated in two dimensions. The single precision floating point representation was selected to amplify the development of round off errors.

The motion states of the spacecraft and simulation time were expressed in canonical units.[14] Canonical units, consisting primarily of distance units (DU) and time units (TU), are commonly used in spacecraft simulation to normalize the very large motion states that are typical for spacecraft. These units are essentially scaling operators chosen such that the magnitudes of radius, velocity and acceleration for a reference circular orbit are 1 and the period of the orbit is 2π. The parameters of the earth's orbit were chosen as the baseline for the canonical units used here: The distance unit was set to the semi major axis of the earth's orbit around the Sun (1 AU or 1.4×10^{12} meters) and the time unit was set to 1.4×10^{12} seconds so that 1 year would correspond to 2π time units.

The scenario was run using the 5th order Runge Kutta integration[11] method with a fixed time step per run. The scenario was run at five different time steps to evaluate their effect on error. Each simulation run had its duration fixed to the period of the orbit (7.81668×10^{9} seconds or 1589.534 TU), with the number of iterations in a run thus determined by the magnitude of the time step.

Two frame settings were used: the first frame setting did not use intermediate frames while the second frame setting used intermediate frames. A total of 40 runs were carried out for each time step and frame setting. Initial position and velocity in each run were randomly generated as the round off error depends greatly on the actual values and may thus span the entire range from zero to the maximum round off error based on the initial conditions.

The absolute errors in position and velocity were measured by taking the magnitude of their respective error vectors at the end of the simulation run. If there were no errors, the starting position would be the same as the final position after one orbit. Thus, the absolute position error was a measure of the distance of the final position of the spacecraft from its starting position.

Results of Simulation Runs

The maximum, minimum and average absolute errors were measured across all runs in each frame setting and time step. In addition to the measured error, the estimated maximum round off error $\square\underline{X}_{Rnd}$ estimated using Eqs. (11), (26), (28), and (29) were also calculated to validate those equations and to compare the errors with and without intermediate frames. Truncation error $\square\underline{X}_{Tr}$ was estimated from a 6th order term of the Runge Kutta integration routine.

Intermediate frames significantly reduced the error. The major observations are

1) For all sizes of time step in these runs, the use of intermediate frames significantly reduced the minimum, maximum and average absolute measured errors. In fact, the maximum absolute measured error with intermediate frames is lower than the average absolute measured error without intermediate frames, as illustrated in Fig. 4.

2) The intermediate frames were especially effective at small time steps (equivalently, large numbers of iterations), where round off error is especially significant in conventional simulations. In Fig. 4, the measured error without intermediate frames increases sharply at very small time steps. The ratio of the average measured errors for the case without intermediate frames to the case using intermediate frames exceeds two orders of magnitude for the smallest time step used in this study.

3) The estimated maximum error, calculated from Eqs. (11), (26), (28), and (29), is shown as a function of time step in Fig. 5. It is observed that this estimated maximum error was significantly lower when intermediate frames were used.

4) Examining estimated truncation error as a function of time step, as shown in Fig. 6, the use of intermediate frames also reduces estimated truncation error. Truncation error is typically proportional to the magnitude of the state and the values of a vehicle's motion states are bounded through the use of intermediate frames.

5) The simulation was also run with different orbital eccentricities, ranging from 0.2583 to 0.90, while maintaining a constant orbital period. The effectiveness of the intermediate frames was evaluated using the ratio of average measured error between the runs with intermediate frames over the runs without intermediate frames, where a small error ratio indicated a significant reduction in numerical error. Figure 7 shows the error ratio for a range of eccentricities. Intermediate frames significantly reduced round off error at small time steps for the entire range of eccentricities. However, the use of large time steps with large eccentricities seemed to diminish the benefit of using intermediate frames, perhaps due to the large truncation errors generated at the periapsis due to a large fixed time step with highly eccentric orbits, suggesting the need for variable time steps and the ability to update critical levels accordingly.

6) Representing values in other units (e.g., metric instead of canonical) did not significantly change the relative numerical errors or the effect of using intermediate frames. This is due to the fact that, although the use of different units changed the scaling on position, velocity and time, the ratio between the incremental term and the state did not change.

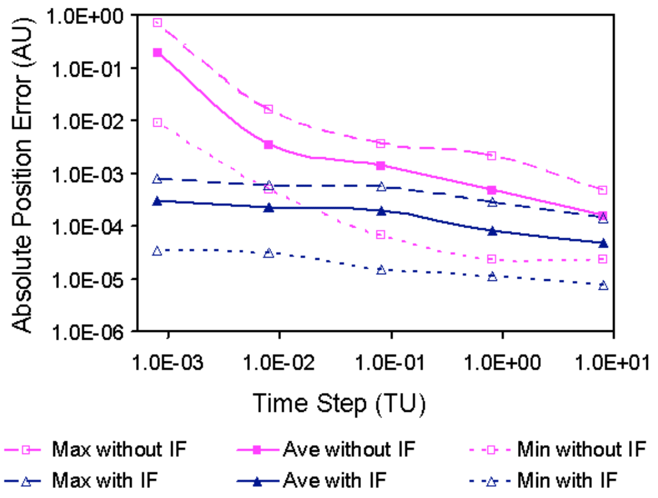

Fig. 4 Absolute measured error in position as a function of time step.

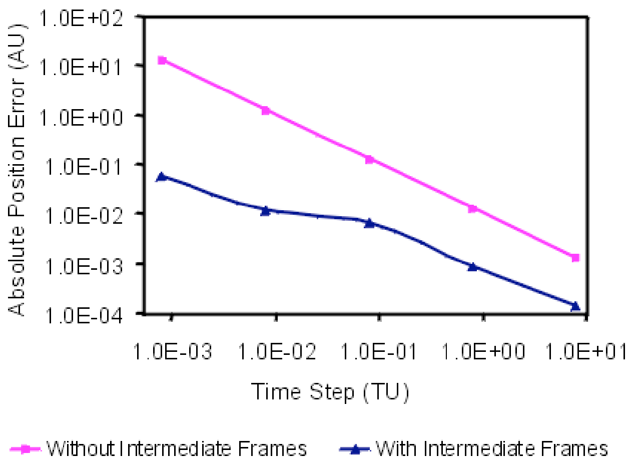

Fig. 5 Estimated maximum error in position as a function of time step.

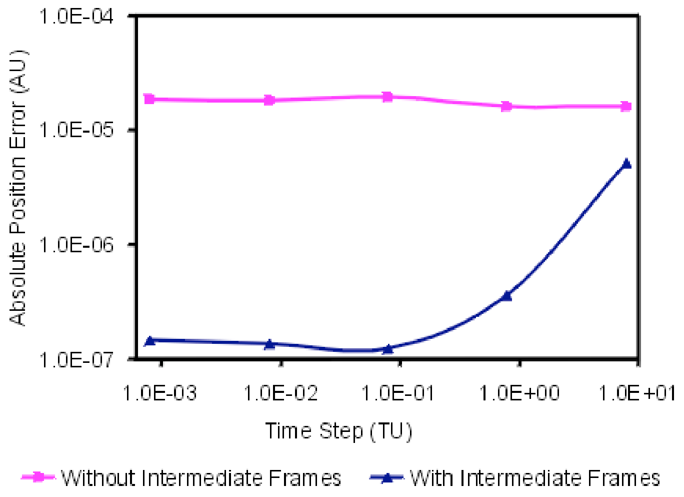

Fig. 6 Estimated truncation error in position as a function of time step.

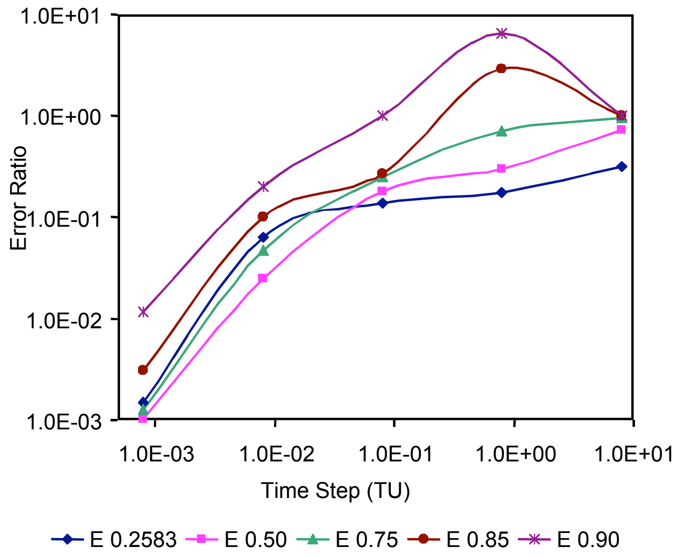

Fig. 7 Ratio of measured position error with intermediate frame over measured position error without intermediate frames as a function of time step for different eccentricities.

Conclusion

Reference frames are typically intrinsic to dynamic models when instantiated in computer simulations. This paper has instead treated reference frames as unique entities within a simulator's software architecture that can be manipulated for a variety of purposes. The focus of this work was to develop a mechanism that would facilitate the management of reference frames and to reduce round off error through the introduction of intermediate reference frames. To accomplish these purposes, a reference frame management system was developed and instantiated into the Reconfigurable Flight Simulator.

Reference frame management and intermediate frames were tested using elliptical orbits to examine numerical error. The RFM built a network of reference frames and handled the transformations between them. Furthermore, the experiment found that the use of intermediate frames can significantly reduce round off error. With small time steps, the improvement was of several orders of magnitude. In addition to reducing round off error, intermediate frames also reduced truncation error. Thus, intermediate frames make it possible to manage total error rather than trading off round off error against truncation error, and did so without requiring specialized computer architectures or software.

These results suggest several possible extensions. For example, given a method of automatically evaluating the error caused by the choice of reference frames (such as the selection of an inertial frame), a dynamic model can use the RFM to change its reference frames to meet fidelity requirements at any point in a simulation. Likewise, additional terms in the forces and moments equations due to the motion of the reference frames can be calculated automatically by the RFM and provided to the dynamic model. This could be used to develop a single generic 6-degree of freedom dynamic model for simulating a wide range of vehicles.

The RFM could also be used in distributed simulation; a large number of simulators using different reference frames could interact over a network without needing advance knowledge of the reference frames used in every other simulator and without being fixed to only one global definition frame for all applications.

Further developments of intermediate frames can examine the impact of changing their orientation and angular velocity for reducing round off error. Likewise, this study detailed the upper and lower bounds for the selection of critical levels as well as the trade off required between several error terms. Optimum settings may exist for reducing error while balancing computational load. These developments will enable total error management through the analysis of the relation between the time step, critical levels, round off error and truncation error.

Acknowledgments

This work was partially supported by the NASA grant NAG2-1291 with Irv Statler as technical monitor, and by a seed grant from Georgia Tech's Modeling and Simulation Research and Education Center (MSREC) with Eric Johnson as coinvestigator.

References

[1]Miller, D. C., and Thorpe, J. A., "SIMNET: The Advent of Simulator Networking," *Proceedings of the IEEE*, Vol. 83, No. 8, IEEE, New York, 1995, pp. 1114–1123.

[2]Burchfiel, J., and Smythe, S., "Use of Global Coordinates in the SIMNET Protocol," White Paper ASD-90-10, *Second Workshop on Standards for Interoperability of Defense Simulations*, Vol. 3, Inst. for Simulation and Training, Orlando, FL, 1990.

[3]Hofer, R. C., and Loper, M. L., "DIS Today," *Proceedings of the IEEE*, Vol. 83, No. 8, IEEE, New York, 1995, pp. 1124–1137.

[4]Lin, K. C., and Ng, H., "Coordinate Transformations in Distributed Interactive Simulation (DIS)," *Simulation*, Vol. 61, No. 5, Society for Computer Simulation, San Diego, 1993, pp. 326–331.

[5]Rolfe, J. M., and Staples, K. J., "Equations of Motion", *Flight Simulation*, Cambridge University Press, Cambridge, England, 1997, pp. 42-51.

[6]Ginsberg, J. H., *Advanced Engineering Dynamics,* 2nd ed., Cambridge University Press, New York, 1998.

[7]Maxwell, E. A., *General Homogenous Coordinates in Space of Three Dimensions,* 1st ed., University Press, Cambridge, MA, 1959.

[8]Denavit, J., and Hartenberg R. S., "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices," *Transactions of the ASME, Journal of Applied Mechanics*, Vol. 22, ASME, New York, 1955, pp. 215–221.

[9]Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F., "Geometrical Transformations," *Computer Graphics, Principles and Practice*, *Second Edition in C*, Addison-Wesley Publishing Company, 1999, pp. 201–226.

[10]Beer, F. P., and Johnston Jr., E. R., *Vector Mechanics for Engineers: Dynamics,* 3rd SI Metric ed., McGraw-Hill Ryerson Limited, Toronto, 1999.

[11]Press, W. H., Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, New York, 1997.

[12]Ippolito, C. A., and Pritchett, A. R., "Software Architecture for a Reconfigurable Flight Simulator," AIAA Paper 2000-4501, Aug. 2000.

[13]Hale, F. J., *Introduction to Space Flight*, Prentice-Hall, Upper Saddle River, NJ, 1994, p. 343.

[14]Bate, R. R., Mueller, D. D., and White, R. R., "Canonical Units", *Fundamentals of Astrodynamics*, Dover Publications, New York, 1971, pp. 40–43.