MANAGEMENT OF REFERENCE FRAMES IN SIMULATION AND ITS APPLICATIONS

A Dissertation
Presented to
The Academic Faculty

by

Satchidanand A. Kalaver

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
May 2006

MANAGEMENT OF REFERENCE FRAMES IN SIMULATION AND ITS APPLICATIONS

Approved by:

Dr. Amy R. Pritchett, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Eric N. Johnson
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Dewey H. Hodges
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Angus L. McLean
College of Computing
Georgia Institute of Technology

Dr. John R. Olds
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: February 17, 2006

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support and guidance of my advisor, Dr. Amy R Pritchett. Her guidance, knowledge and patience were instrumental in the completion of this dissertation.

I would like to extend my gratitude to Dr. Dewey Hodges, Dr. John Olds, Dr. Eric Johnson, and Dr. Thom McLean, for taking the time and effort to serve on my dissertation committee. Their suggestions and insight proved invaluable to the completion of this dissertation.

I would also like to thank my parents, Anil and Nivedita, for their constant support in all my endeavors.

I would especially like to thank Anuj Shah and Karen Feigh for their insights and suggestions on object oriented software and statistics respectively as well as their camaraderie. I would also like to thank Luis Nicolas Gonzalez Castro and Alexander Quinn for their lively banter and camaraderie that kept life interesting.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

$\underset{j}{{}^{A}\underline{X}_{C}^{B}}$	j^{th} element of motion parameter of A with respect to B expressed in C
${}^{A}\underline{P}_{B}^{B}$	Position vector of A with respect to B expressed in B
${}^{A}\underline{Q}^{B}$	Orientation of A with respect to B
${}^{A}\underline{V}_{A}^{B}$	Velocity of A with respect to B expressed in A
${}^{A}\underline{w}_{A}^{B}$	Angular velocity of A with respect to B expressed in A
${}^{A}\dot{\underline{V}}_{A}^{B}$	Acceleration of A with respect to B expressed in A
${}^{A}\dot{\underline{w}}_{A}^{B}$	Angular acceleration of A with respect to B expressed in A
$\left[{}^{A}\tilde{\underline{w}}_{A}^{B}\right]\underline{X}$	Matrix representation of cross product: ${}^{A}\underline{w}_{A}^{B}\times\underline{X}$
$[C_X(f)]$	Matrix Rotation about X-axis by angle f
$[C_Y(q)]$	Matrix Rotation about Y-axis by angle q
$[C_Z(y)]$	Matrix Rotation about Z-axis by angle y
$[Tr(\underline{X})]$	Homogenous matrix representing a translation by $\underline{X}$
$[Tr(C)]$	Homogenous matrix representing a rotation by matrix $[C]$
$\frac{{}^{B}d}{dt}\left({}^{A}\underline{X}_{A}^{B}\right)$	Time derivative of ${}^{A}\underline{X}_{A}^{B}$ to an observer in B
bf	Body Fixed Frame
bc	Body Carried Frame
n	Navigation Frame
i	Inertial Frame

$\underline{X}_m$	Motion states of a dynamic model
${}^{bf}F_{bf}$	Force on the body fixed frame expressed in the body fixed frame
${}^{bf}M_{bf}$	Moments about the body fixed frame expressed in the body fixed frame
m	Mass of body
${}^{bf}I_{bf}$	Inertia tensor of the body about the body fixed frame
$f,\ q,\ y$	Components of orientation in Euler angles
q_0, q_1, q_2, q_3	Components of orientation in Euler parameters /quaternions
$\dot{f},\ \dot{q},\ \dot{y}$	Components of orientation rates expressed in rates for Euler angles
$\dot{q}_0, \dot{q}_1, \dot{q}_2, \dot{q}_3$	Components of orientation rates expressed in rates for a quaternion
$p,\ q,\ r$	Components of angular velocity in an orthogonal frame
$x\ , y\ , z$	Components of position in an orthogonal frame
$u\ , v\ , w$	Components of velocity in an orthogonal frame
$\underline{X}\{n\}$	State vector at time *n*
$\underline{\dot{X}}\{n\}$	Derivative vector at time *n*
Δt	Time step for integration
$\underline{\dot{X}}\{n\} \bullet \Delta t$	Incremental Term at time *n*
$\Delta\underline{X}$	Total error per time step
$\Delta\underline{X}_{LTE}$	Local truncation error (truncation error per time step)
$\Delta\underline{X}_{Rnd}$	Local roundoff error (roundoff error per time step)
s	Sign bit in the floating-point representation of a variable
M	Value of bits in mantissa for the floating-point representation of a variable
B	Base used for exponent in the floating-point representation of a variable

e	Value of bits in exponent for the floating-point representation of a variable
E	Bias of exponent in the floating-point representation of a variable
e_m	Machine accuracy for the floating-point representation of a variable
MSB	Most significant bit in the mantissa
LSB	Least significant bit in the mantissa
N	Number of bits in the mantissa
PDS	Parallel and Distributed Simulation
RFS	Reconfigurable Flight Simulator
ECAD	Environment Controller and Database
RFM	Reference Frame Manager
${}_j\underline{Cr}$	j^{th} element in the vector of Critical Levels
$\Delta\underline{X}_{Cr}$	Roundoff error due to critical levels
$\Delta\underline{X}_{U}$	Roundoff error due to update of intermediate frame
$k_{\Delta t}$	Number of time steps in the simulation run
$\underline{k}_u$	Vector of updates for the intermediate frame
e	Eccentricity of elliptical orbit

SUMMARY

The choice of reference frames used in simulations is typically fixed in dynamic models based on modeling decisions made early during their development, restricting model fidelity, numerical accuracy and integration into large-scale simulations. Individual simulation components typically need to model the transformations between multiple reference frames in order to interact with other components, resulting in additional development effort, time and cost.

This dissertation describes the methods for defining and managing different reference frames in a simulation, thereby creating a shared simulation environment that can provide reference frame transformations, comprising of kinematics and rotations, to all simulation components through a Reference Frame Manager. Simulation components can use this Reference Frame Manager to handle all kinematics and rotations when interacting with components using different reference frames, improving the interoperability of simulation components, especially in parallel and distributed simulation, while reducing their development time, effort and cost. The Reference Frame Manager also facilitates the development of Generic Dynamic Models that encapsulate the core service of dynamic model, enabling the rapid development of dynamic models that can be reused and reconfigured for different simulation scenarios and requirements. The Reference Frame Manager can also be used to introduce Intermediate Frames that bound the magnitudes of vehicle states, reducing roundoff error and improving numerical accuracy.

CHAPTER 1

INTRODUCTION

Simulations are widely used in the aerospace industry and can be tailored to specific applications, including design and evaluation of aerospace vehicles, pilot training, mission planning and the modeling of large systems such as air-traffic control or command and control networks. Dynamic models form the conceptual basis of aerospace simulations, propagating motion states such as position, orientation and velocity, represented with respect to clearly defined reference frames.

The utility of simulations is limited by the accuracy of their dynamic models and the cost-effectiveness of both developing them and reconfiguring them to different scenarios. The accuracy of dynamic models refers to the fidelity of their sub-systems and kinetics models as well as the numerical accuracy of propagating the models through time with numerical integration. For the purpose of this research, fidelity expresses how closely a dynamic model matches the behavior of the real vehicle [1]. The fidelity required of a dynamic model may vary based on the requirements of different simulation scenarios or even during the course of a simulation run, necessitating its reconfiguration. Furthermore, different scenarios may require the use of different simulation components. Therefore, reconfiguring dynamic models may require their motion to be expressed with respect to different reference frames, introduce interactions with different simulation components, or even necessitate changes to the subsystems and kinetics model. Consequently, improving the cost-effectiveness of developing and reconfiguring dynamic

models facilitates their reuse in subsequent simulations, reducing the cost, time and effort needed to develop a large variety of simulations and scenarios [2][3].

Reference frame definitions are currently implicit within the simulation software and can vary with the individual simulation components. For example, dynamic models and displays implicitly use reference frames when representing motion. The choice of reference frames used is based on modeling decisions made early during simulation software development. Different dynamic models and displays may use different reference frames based on the scenarios they are designed to support. This implicit representation of reference frames within individual simulation components creates the following problems with accuracy of the model as well as its cost-effectiveness, especially with regards to software development, reconfiguration and reuse:

- Software needs to be developed within each simulation component to enable its interaction with other simulation components using different reference frames, increasing development time and cost of the simulation. This software is often implemented in multiple simulation components, leading to a duplication of development effort. Furthermore, it is unique to each pair of reference frames, and can lead to the N^2 problem if each component uses a different reference frame and interacts with all the other components, limiting the scalability of the simulation. This is especially true for large-scale simulation, and for parallel and distributed simulation, where knowledge of all the reference frames may not be feasible. While the selection of a common reference frame for component interaction during the software development phase has been used in distributed simulation to address this

issue, forcing all simulations components to use a pre-defined common reference frame can introduce separate problems, as it may not be appropriate for all scenarios.

- The choice of reference frames can affect the accuracy of the simulation. The kinetics fidelity of the dynamic model depends upon the choice of inertial frame used to evaluate the model's acceleration. Different scenarios or different stages of a simulation run may require different kinetics fidelity due to the motion of the dynamic model, necessitating a change of the inertial frame used, requiring a change in the software implementation of the model.
- The numerical error incurred by the integration routine consists of truncation error and roundoff error. The truncation error depends upon the choice of integration routine and size of the time step while the roundoff error depends upon the relative magnitudes of the motion states and their time derivatives as scaled by time step [4]. Therefore, choosing a reference frame that may be ideal for kinetics fidelity may in fact be detrimental to roundoff error.
- Reconfiguring scenarios and reusing simulation components to modify or expand existing simulations may introduce additional reference frames that a simulation component needs to interact with. Existing simulation components may need to be further modified to support interactions with new simulation components or may be required to express their motion with respect to different reference frames, further increasing development time and cost.

1.1 Proposed Solution

The proposed solution is to view reference frames as unique entities within the simulation environment that can be used by all the simulation components, enabling

individual components to interact with one another, irrespective of the reference frames used to express their motion. This eliminates the need for specialized software to enable interactions between components, improving scalability, reconfiguration and reuse of existing components. Dynamic models are able to use reference frames suitable for the kinetics fidelity required for each stage of the simulation and can also select reference frames that bound the roundoff error for their motion states.

Conceptually, reference frames can be viewed as entities in motion with respect to other reference frames. Consequently, the motion of vehicles can be treated as the motion of their body frames with respect to their navigation frames. Therefore, the kinematics of the vehicle can be viewed as the kinematics between reference frames. Even the kinetics model of the vehicle can be viewed as a relation utilizing the unique inertial properties of the vehicle and the motion of reference frames.

In a simulation environment, reference frames can be treated as a common resource, available to all simulation components through a centralized mechanism. This centralized mechanism calculates and provides the simulation components with the kinematics and rotations between reference frames. This mechanism, called the Reference Frame Manager or RFM, is also responsible for maintaining the reference frames in the simulation environment and forms the central part of the proposed solution.

The Reference Frame Manager assembles a modifiable network of reference frames in the simulation environment at runtime. The RFM is able to add or remove reference frames from this network during the course of the simulation, allowing the network to provide the reference frames required for different scenarios and simulation configurations. The individual reference frames are viewed as nodes within this network

and the RFM is able to traverse the links in the network to calculate the kinematics and rotations between any pair of reference frames in the network.

Simulation components are therefore able to request the kinematics and rotations between any pair of reference frames in the simulation environment through the RFM, allowing them to express the motion of other simulation components with respect to their preferred reference frames. Simulation scenarios can therefore be reconfigured rapidly without requiring modifications to existing components in order to interact with new components using new reference frames. Similarly, dynamic models are also able to use reference frames required by the scenario or kinetics fidelity and change them at runtime if necessary, improving the ease with which these models can be reused and reconfigured for different simulation scenarios. The RFM also enables the following applications to be developed:

- Intermediate frames that can be requested by dynamic models from the RFM to control roundoff error. These intermediate frames bound the magnitude of the model's motion states, allowing the roundoff error to be controlled independently of time step and truncation error.
- Software component representing a generic dynamic model that encapsulates the core services and functionality common to 6DOF dynamic models, especially kinetics, kinematics, transformation of motion parameters and integration routines. A generic model would encourage code reuse and reduce the cost, time and effort of developing new dynamic models. The software developer only needs to develop the components unique to the specific vehicle being modeled, primarily the external forces and moments on the vehicle, its internal sub-systems and its inertia properties. The ability

to choose its inertial frame as required and request an intermediate frame to control roundoff error would also be built into the generic model.

- The data passing protocols in parallel and distributed simulation can be modified so that the choice of a common reference frame, if desired, does not have to be made during the development phase. Each component within the simulation can express motion with respect to its preferred reference frame. The RFM on each processor handles all the kinematics and rotations to ensure that components on different processors can interact using their preferred reference frames. If a common reference frame is desired, it is selected at runtime to suit the simulation scenario.

1.2 Thesis Objectives

1. Represent reference frames as entities whose motion can be defined with respect to other reference frames.
2. Represent reference frames in the simulation environment as unique objects such that they can form nodes in a network of reference frames.
3. Develop a Reference Frame Manager that is able to assemble a network of reference frames as well as add or remove reference frames from the network at runtime.
4. Develop algorithms to calculate kinematics and rotations between arbitrary pairs of reference frames in the network
5. Develop intermediate frames that can be used to reduce the roundoff error incurred by vehicles during numerical integration.
6. Develop algorithms to adaptively control the motion of intermediate frames based on the vehicle dynamics.

7. Develop a generic dynamic model that encapsulates the core services of 6DOF rigid body dynamic models.
8. Identify the unique elements of dynamic models that can be used by the generic dynamic model to assemble models of different vehicles.
9. Develop data passing protocols to utilize RFM in parallel and distributed simulation, enabling components on each processor to use their preferred reference frames to express motion. Correspondingly, eliminate the need to fix a common reference frame during the development phase of the simulation components.

1.3 Thesis Outline

This thesis is divided into 3 sections. The first section, Chapter 2, deals with the background information highlighting key concepts in managing reference frames. The primary topics covered are reference frame, their kinematics and rotations, dynamic models and their components, numerical error and their associated errors, parallel and distributed simulation and software reuse.

The second section consists of 4 chapters, where each chapter deals with a major aspect of the research effort. Each chapter is further divided so as to include a conceptual treatment of the research and its implementation into software. Chapter 3 deals with the definition of reference frames and its management within the simulation environment to create an extendable network of reference frames. The instantiation of the Reference Frame Manager using the Reconfigurable Flight Simulator is briefly described. Chapter 4 deals with the use of reference frames to reduce roundoff error. The parameters for defining these intermediate frames are discussed with regards to their effect on roundoff error. This chapter also describes the development of an algorithm to adaptively select

these parameters based on the dynamics of the model. Chapter 5 deals with the modeling of dynamics using reference frames. This leads to the development of a generic dynamic model as well as the interface requirements for creating dynamics components that model the elements unique to each vehicle. Chapter 6 deals with the use of the RFM in PDS. This chapter looks at several design parameters that need to be considered and develops the data passing protocols to use RFM in PDS.

The final section describes the demonstration effort and discusses the contributions of this work and areas of future research. Chapter 7 demonstrates the capabilities and benefits of RFM and its applications. The costs of using RFM and generic models are also discussed. Chapter 8 provides a summary of this thesis and discusses the contributions of this work as well as areas of future research.

CHAPTER 2

BACKGROUND

This chapter describes the fundamental concepts required for the development of a reference frame management mechanism and its applications to dynamic modeling, error reduction and distributed simulation. Reference frames and their properties with regards to coordinate systems, kinematics equations and rotations are introduced and the reference frames commonly used in simulation are described. Dynamic modeling of vehicles, especially with regards to motion using Newton's 2nd Law, is then discussed. Numerical integration routines and the sources of numerical error in simulation are also described. Common software representations of dynamic models in simulation are illustrated. The effects of reference frames on distributed simulation and on the assembly and modification of simulation environments are discussed.

2.1 Reference Frames

A reference frame determines the origin and directions used for expressing the motion between different bodies. The origin is the point from which position with respect to the reference frame is determined while the axes define the direction vectors used to express the motion vectors as sets of scalar quantities. The directions of the axes can be defined using a left-hand system or a right-hand system. The following subsections deal with the representation of motion with respect to reference frames, the use of coordinate systems, and the kinematics and rotations needed to express motion with respect to different reference frames.

2.1.1 Representing Motion with Reference Frames

A reference frame can be viewed as a rigid entity as the orientation of its axes does not change with respect to one another. Consequently, a reference frame can be attached to a rigid body and the motion of the reference frame can be used to represent the motion of the rigid body. Since reference frames are used to express the motion of rigid bodies, representing the motion of rigid bodies with motion of reference frames implies that the motion of reference frames can be expressed with respect to other reference frames. Furthermore, the kinematics of these vehicles can be treated as the kinematics of reference frames.

The parameters used to define the motion of rigid bodies can be applied to reference frames. These parameters, called *motion parameters* in this thesis, commonly consist of position, orientation, velocity, acceleration, angular velocity and angular acceleration [5] and define the relative motion between two reference frames. Expressing the motion of a reference frame with respect to another reference frame, called the *definition frame*, determines the direction of the vectors representing these parameters. In this thesis, the reference frame whose motion is being expressed will be called the *object frame*. In Figure 2.1, the vector $\underline{X}$ represents the relative motion between the reference frames A and B and its direction depends upon the identity of the definition frame. This thesis uses a left superscript to identify the reference frame whose properties are being expressed while the right superscript identifies the definition frame. Therefore, the motion of A with respect to B is expressed as ${}^{A}\underline{X}^{B}$ while the motion of B with respect to A is expressed as ${}^{B}\underline{X}^{A}$; these corresponding vectors are illustrated in Figure 2.1.

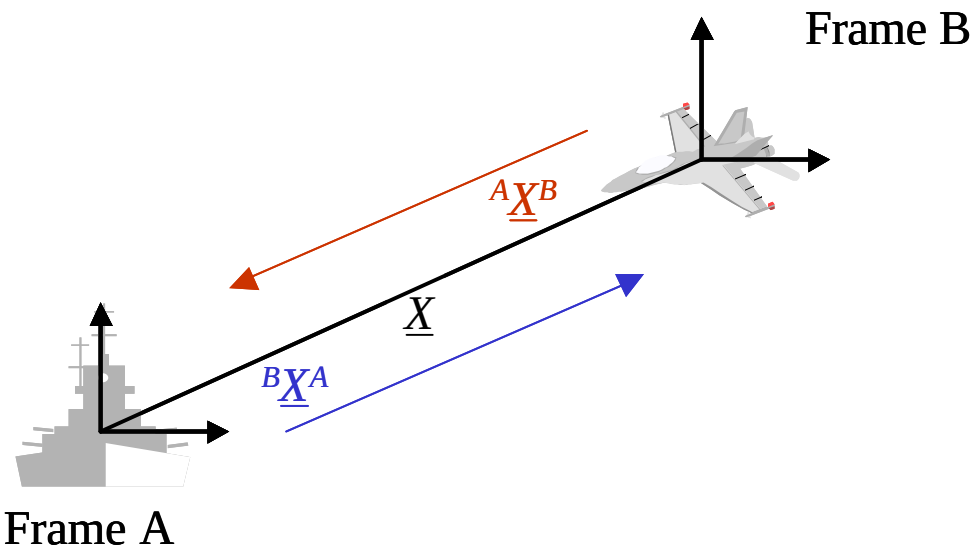


Figure 2.1: Relative Motion Between Reference Frames

While the identity of the definition frame determines the direction of the vector representing the motion parameters, the direction vectors used to express the motion parameters as sets of scalar quantities is determined by a *measurement frame*. The vector for each motion parameter is projected on the direction vectors defined by the axes of the measurement frame. The exception is the reference frame's orientation, which represents the rotation from the definition frame's axes to the object frame's axes. The measurement frame may be the reference frame in question, the definition frame or a third frame. This thesis uses a right subscript to denote the identity of the measurement frame. A left subscript may be used to identify a particular element of the motion parameter. Thus, $^{A}_{j}\underline{X}^{B}_{C}$ represent the j^{th} element of the motion parameters of object frame A with respect to definition frame B in measurement frame C.

The measurement frame used during the propagation of motion parameters of reference frames and dynamic models in simulation determines the implementation of its

kinematics equations. This thesis uses the object and definition frames of motion parameters as their measurement frames to facilitate a consistent representation of the kinematics equations throughout this document as listed in Table 2.1. However, these motion parameters can also be expressed in arbitrary measurement frames.

Table 2.1: Motion Parameters and Their Notation

Motion Parameter	Notation
Position	${}^{A}\underline{P}_{B}^{B}$
Orientation	${}^{A}\underline{Q}^{B}$
Velocity	${}^{A}\underline{V}_{A}^{B}$ or ${}^{A}\underline{V}_{B}^{B}$
Acceleration	${}^{A}\underline{\dot{V}}_{A}^{B}$ or ${}^{A}\underline{\dot{V}}_{B}^{B}$
Angular Velocity	${}^{A}\underline{w}_{A}^{B}$
Angular Acceleration	${}^{A}\underline{\dot{w}}_{A}^{B}$

The motion parameter for position uses the definition frame as its measurement frame. As discussed above, the orientation of the reference frame represents the rotation required to change the measurement frame from the definition frame to itself. Therefore, specifying a measurement frame is not applicable for orientation. The default measurement frame for the velocity and acceleration parameters is the object frame, although the definition frame can also be used as the measurement frame for velocity and acceleration if necessary.

2.1.2 Coordinate Systems in Simulation

Coordinate systems are used to express the motion of reference frames and vehicles as scalar values. The vectors of the motion parameters are projected onto the

axes of the measurement frame. The exact scalar values depend upon the coordinate system used [6], determining the manner in which the projections on the measurement frame's axes are interpreted. For example, the position of an object using the Earth Centered Reference Frame as its definition and measurement frames, as depicted by Figure 2.2, can be expressed in two different coordinate systems. The choice of coordinate system depends upon the application and determines the scalar representation of the motion parameter although the vector representing the motion parameter is unaffected.

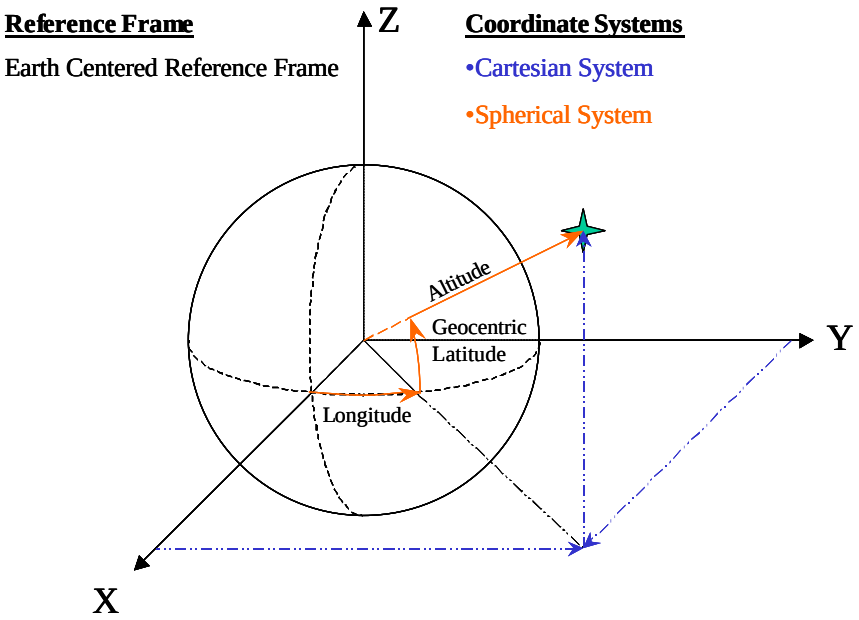


Figure 2.2: Reference Frame With Multiple Coordinate Systems

2.1.3 Kinematics Equations and Rotations of Motion Parameters

Motion parameters are defined and maintained by reference frames with respect to their definition frames. The default measurement frame used by reference frames for

maintaining their motion parameters during the course of the simulation was listed in Table 2.1. However, the motion parameters may need to be expressed with respect to different definition frames or have their scalar components expressed in different measurement frames during the course of the simulation. This may arise during the interaction of vehicles and components using different reference frames or if the calculation of vector derivatives in the dynamic model requires the vehicle's motion parameters with respect to different reference frames. Kinematics equations are used when the definition frame needs to be changed. Similarly, rotations are used when the measurement frame needs to be changed. These operations transform the motion parameters so that they are expressed in the appropriate definition and measurement frames.

The numerical representation of motion parameters as scalar values depends upon the choice of measurement frame and coordinate system. When a measurement frame is changed, the rotation or direction cosine matrix representing the orientation of the new measurement frame's axes with respect to the current measurement frame's axes is generated and applied to the motion parameters through matrix multiplication. If the type of coordinate system is changed, the exact coordinate transformation depends upon the specific combination of coordinate systems. While this thesis will only use the Cartesian coordinate system and only develop the rotations due to the change of measurement frames, the results of this thesis are not limited to this coordinate system.

When the definition frame is changed to a different reference frame, the object frame's motion parameters are updated to reflect the object frame's motion with respect to its new definition frame. Therefore, the object frame's motion parameters require the

use of kinematics equations to account for the motion between the old and new definition frames. These equations typically require translations, cross products, multiplication of vectors by scalars and rotations.

In three-dimensional space, these operations can be represented by 3×3 matrix multiplications and 3×1 addition or by 4×4 homogenous matrix multiplications [5][7][8]. When these operations are represented by a sequence of matrix multiplications on a vector, the transformation can be assembled and used repeatedly as long as the data used to assemble the matrix does not change, improving computational efficiency.

2.1.3.1 Operations Using 3×3 and 3×1 Matrices

Translations are typically represented by the addition of a 3×1 column matrices representing the motion parameter and a translation vector [5][8]. For example, if the position of an object P 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 $(x_B, y_B, z_B)^T$, assuming both frames have the same orientation, the transformation is expressed by equation (2.1).

$$ {}^{P}\underline{X}_{B}^{B} = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = {}^{P}\underline{X}_{B}^{A} + {}^{A}\underline{X}_{B}^{B} \qquad (2.1) $$

Multiplication of a vector by another vector or by a scalar can be represented by matrix multiplication. In a cross product, the first vector can be expressed as a skew symmetric matrix [6], and is multiplied to the 3×1 column matrix representing the second vector. For example, if the position of an object P with respect to Frame A is constant at

$(x_A, y_A, z_A)^T$ and Frame A has an angular velocity of $(p, q, r)^T$, the velocity of P can be calculated by a cross product of the angular velocity and position as expressed by equation (2.2). Similarly, when multiplying a vector by a scalar, the scalar can be expressed as a matrix by multiplying it to an identity matrix. Equation (2.3) illustrates the multiplication of a vector, $(x, y, z)^T$, by a scalar, a. Instead of using matrix representation, the equations in this thesis will represent vector multiplication with vector notation unless otherwise noted.

$$ {}^{P}\underline{V}_{A}^{A} = \begin{bmatrix} u_A \\ v_A \\ w_A \end{bmatrix} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = \left[{}^{A}\tilde{w}_A \right] {}^{P}\underline{X}_{A}^{A} = {}^{A}\underline{w}_A \times {}^{P}\underline{X}_{A}^{A} \tag{2.2} $$

$$ a\underline{X} = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az \end{bmatrix} \tag{2.3} $$

Rotation is typically expressed as the matrix multiplication of a 3×3 rotation matrix and a 3×1 column matrix representing the motion parameter [5][8] in three dimensional space. For example, the position of object P expressed in Frame A can be measured in Frame B by using the rotation matrix $[{}^{B}C^{A}]$ as shown in equation (2.4). This thesis will use matrix multiplication to represent rotation.

$$ {}^{P}\underline{X}_{B}^{A} = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = \left[{}^{B}C^{A} \right] {}^{P}\underline{X}_{A}^{A} \tag{2.4} $$

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. Equations (2.5), (2.6) and (2.7) illustrate the elementary rotations about the X, Y and Z axes respectively. Complex rotation matrices can be formed from multiple elementary rotations; however, the order of operations must be maintained, as the rotations are not commutative [5][8]. 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 changing the measurement frame represents the orientation of the new measurement frame with respect to the old measurement frame and can be generated using either Euler angles or Euler parameters.

$$[C_X(f)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos f & \sin f \\ 0 & -\sin f & \cos f \end{bmatrix} \tag{2.5}$$

$$[C_Y(q)] = \begin{bmatrix} \cos q & 0 & -\sin q \\ 0 & 1 & 0 \\ \sin q & 0 & \cos q \end{bmatrix} \tag{2.6}$$

$$[C_Z(y)] = \begin{bmatrix} \cos y & \sin y & 0 \\ -\sin y & \cos y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

If the orientation of Frame A with respect to Frame B is represented by Euler angles and $[C_X(f)]$, $[C_Y(q)]$ and $[C_Z(y)]$ represent the elementary rotations about the X, Y and Z axes respectively, the rotation matrix $[{}^{A}C^{B}]$, representing the direction cosine matrix of Frame A with respect to Frame B, can be formed as depicted in equation (2.8).

$$\left[{}^{A}C^{B}\right] = \left[C_X(f)\right]\left[C_Y(q)\right]\left[C_Z(y)\right] = \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \tag{2.8}$$

If the orientation of Frame A with respect to Frame B is represented by Euler parameters, commonly called quaternions, the elements of the direction cosine matrix in equation (2.8) can be calculated [9] as tabulated in Table 2.2. The quaternions can be calculated using Euler angles [9] as depicted by equations (2.9) to (2.12) while the Euler angles can be obtained from elements of the rotation matrix [9] as depicted by equations (2.13), (2.14) and (2.15).

Table 2.2: Generation of Rotation Matrix from Euler Angles and Quaternions

Matrix Element	From Euler Angles	From Quaternions
l_1	$\cos(q)\cos(y)$	$q_0^2 + q_1^2 - q_2^2 - q_3^2$
l_2	$\cos(q)\sin(y)$	$2(q_1q_2 + q_0q_3)$
l_3	$-\sin(q)$	$2(q_1q_3 - q_0q_2)$
m_1	$\sin(f)\sin(q)\cos(y) - \cos(f)\sin(y)$	$2(q_1q_2 - q_0q_3)$
m_2	$\sin(f)\sin(q)\sin(y) + \cos(f)\cos(y)$	$q_0^2 - q_1^2 + q_2^2 - q_3^2$
m_3	$\sin(f)\cos(q)$	$2(q_2q_3 + q_0q_1)$
n_1	$\cos(f)\sin(q)\cos(y) + \sin(f)\sin(y)$	$2(q_0q_2 + q_1q_3)$
n_2	$\cos(f)\sin(q)\sin(y) - \sin(f)\cos(y)$	$2(q_2q_3 - q_0q_1)$
n_3	$\cos(f)\cos(q)$	$q_0^2 - q_1^2 - q_2^2 + q_3^2$

$$q_0 = \cos\left(\frac{y}{2}\right)\cos\left(\frac{q}{2}\right)\cos\left(\frac{f}{2}\right) + \sin\left(\frac{y}{2}\right)\sin\left(\frac{q}{2}\right)\sin\left(\frac{f}{2}\right) \tag{2.9}$$

$$q_1 = \cos\left(\frac{y}{2}\right)\cos\left(\frac{q}{2}\right)\sin\left(\frac{f}{2}\right) - \sin\left(\frac{y}{2}\right)\sin\left(\frac{q}{2}\right)\cos\left(\frac{f}{2}\right) \tag{2.10}$$

$$q_2 = \cos\left(\frac{y}{2}\right)\sin\left(\frac{q}{2}\right)\cos\left(\frac{f}{2}\right) + \sin\left(\frac{y}{2}\right)\cos\left(\frac{q}{2}\right)\sin\left(\frac{f}{2}\right) \tag{2.11}$$

$$q_3 = -\cos\left(\frac{y}{2}\right)\sin\left(\frac{q}{2}\right)\sin\left(\frac{f}{2}\right) + \sin\left(\frac{y}{2}\right)\cos\left(\frac{q}{2}\right)\cos\left(\frac{f}{2}\right) \tag{2.12}$$

$$q = \sin^{-1}(-l_3) = \sin^{-1}(2(q_0 q_2 - q_1 q_3)) \tag{2.13}$$

$$y = \cos^{-1}\left(-l_1 \big/ \cos(q)\right) \times \operatorname{sgn}[l_2] \tag{2.14}$$

$$f = \cos^{-1}\left(n_3 \big/ \cos(q)\right) \times \operatorname{sgn}[m_3] \tag{2.15}$$

2.1.3.2 Operations Using Homogenous Matrices

Expressing the operations for kinematics equations and rotations in 3 dimensional space using 3×3 matrices and 3×1 column matrices requires the application of matrix addition and matrix multiplication. If these operations are expressed using homogenous 4×4 matrices [7] and 4×1 column matrices, they can be executed using only matrix multiplications, allowing a sequence of operations on a vector to be expressed as a single transformation matrix. Homogenous matrices have been used for transformations in kinematics of rigid bodies [10] as well as computer graphics [8].

In using homogenous matrices for the following rigid body transformations, the motion parameters are expressed as 4×1 column vectors where the first 3 elements are from the motion states and the 4th element is 1. The additional dimension allows the matrix addition of 3×1 column matrices to be executed using matrix multiplication, as

depicted in equation (2.16), which executes the translation shown in equation (2.1) using homogenous matrices.

$$\begin{bmatrix} {}^{P}\underline{X}_{B}^{B} \\ 1 \end{bmatrix} = \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} = \left[Tr\left({}^{A}\underline{X}_{B}^{B} \right) \right] \begin{bmatrix} {}^{P}\underline{X}_{B}^{A} \\ 1 \end{bmatrix} \tag{2.16}$$

Similarly, matrix multiplication of 3×3 matrices and 3×1 column matrices, used in the operations depicted by equations (2.2), (2.3) and (2.4), can also be executed by homogenous matrices and 4×1 column matrices. Equation (2.17) executes the rotation described in equation (2.4) using homogenous matrices.

$$\begin{bmatrix} {}^{P}\underline{X}_{B}^{A} \\ 1 \end{bmatrix} = \begin{bmatrix} x_B \\ y_B \\ z_B \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} = \left[R\left({}^{B}C^{A} \right) \right] \begin{bmatrix} {}^{P}\underline{X}_{A}^{A} \\ 1 \end{bmatrix} \tag{2.17}$$

Homogenous matrices allow a sequence of vector additions and matrix multiplications to be represented by a single homogenous matrix that can be multiplied to the 4×1 column matrix representing the motion parameter. Thus, if the position of an object needs to be transformed from Frame A to Frame C using the rotation expressed by equation (2.4) followed by the translation depicted by equation (2.1), the resulting transformation, expressed by equation (2.18), can be expressed as a single homogenous matrix as expressed by equation (2.19).

$$ {}^{P}\underline{X}_{B}^{B} = \left[{}^{B}C^{A}\right] {}^{P}\underline{X}_{A}^{A} + {}^{A}\underline{X}_{B}^{B} \tag{2.18} $$

$$ \begin{bmatrix} {}^{P}\underline{X}_{B}^{B} \\ 1 \end{bmatrix} = \left[Tr\left({}^{A}\underline{X}_{B}^{B}\right)\right]\left[R\left({}^{B}C^{A}\right)\right]\begin{bmatrix} {}^{P}\underline{X}_{A}^{A} \\ 1 \end{bmatrix} = \left[{}^{B}T^{A}\right]\begin{bmatrix} {}^{P}\underline{X}_{A}^{A} \\ 1 \end{bmatrix} \tag{2.19} $$

2.1.3.3 Applying Kinematics Equations and Rotations to Motion Parameters

The operations described above can be used to change the definition and measurement frames of motion parameters. Kinematics equations will be defined in this thesis as the equations used to change the definition frame of motion parameters. When the definition frame is changed, the measurement frame may also be changed. Equations (2.20) to (2.25) represent the kinematics equations used to change the definition frame for the motion parameters of Frame P from Frame A to Frame B. To keep the measurement frame for position as the definition frame, it is also changed to the new definition frame.

$$ \left[{}^{P}C^{B}\right] = \left[{}^{P}C^{A}\right]\left[{}^{A}C^{B}\right] \tag{2.20} $$

$$ {}^{P}\underline{w}_{P}^{B} = {}^{P}\underline{w}_{P}^{A} + \left[{}^{P}C^{A}\right] {}^{A}\underline{w}_{A}^{B} \tag{2.21} $$

$$ \frac{{}^{B}d}{dt}\left({}^{P}\underline{w}_{P}^{B}\right) = {}^{P}\dot{\underline{w}}_{P}^{A} + \left(\left[{}^{P}C^{A}\right] {}^{A}\underline{w}_{A}^{B}\right) \times {}^{P}\underline{w}_{P}^{A} + \left[{}^{P}C^{A}\right] \frac{{}^{B}d}{dt}\left({}^{A}\underline{w}_{A}^{B}\right) \tag{2.22} $$

$$ {}^{P}\underline{P}_{B}^{B} = \left[{}^{B}C^{A}\right] {}^{P}\underline{P}_{A}^{A} + {}^{A}\underline{P}_{B}^{B} \tag{2.23} $$

$$ {}^{P}\underline{V}_{P}^{B} = {}^{P}\underline{V}_{P}^{A} + \left[{}^{P}C^{A}\right]\left({}^{A}\underline{w}_{A}^{B} \times {}^{P}\underline{P}_{A}^{A} + {}^{A}\underline{V}_{A}^{B}\right) \tag{2.24} $$

$$\frac{{}^{B}d}{dt}\left({}^{P}\underline{V}_{P}^{B}\right)={}^{P}\dot{\underline{V}}_{P}^{A}+\left({}^{P}\underline{w}_{P}^{A}+2\left(\left[{}^{P}C^{A}\right]{}^{A}\underline{w}_{A}^{B}\right)\right)\times{}^{P}\underline{V}_{P}^{A}$$
$$+\left[{}^{P}C^{A}\right]\left(\frac{{}^{B}d}{dt}\left({}^{A}\underline{w}_{A}^{B}\right)\times{}^{P}\underline{P}_{A}^{A}+{}^{A}\underline{w}_{A}^{B}\times\left({}^{A}\underline{w}_{A}^{B}\times{}^{P}\underline{P}_{A}^{A}\right)+\frac{{}^{B}d}{dt}\left({}^{A}\underline{V}_{A}^{B}\right)\right) \quad (2.25)$$

If only the measurement frame is changed, the rotation matrices representing the orientation of the new measurement frame with respect to the measurement frames used by the motion parameters are applied to the appropriate motion parameters. A change of measurement frame is meaningless with regards to representing orientation since the orientation can be expressed as the rotation matrix for changing measurement frames from the definition frame to its own reference frame. Equations (2.26) to (2.30) represent the rotations used to change the measurement frame used by Frame P, which is defined with respect to Frame A, to Frame B. While these rotations may be viewed as a subset of kinematics, this thesis will treat the change of measurement frames as unique rotation operations.

$${}^{P}\underline{w}_{B}^{A}=\left[{}^{B}C^{P}\right]{}^{P}\underline{w}_{P}^{A} \quad (2.26)$$

$${}^{P}\dot{\underline{w}}_{B}^{A}=\left[{}^{B}C^{P}\right]{}^{P}\dot{\underline{w}}_{P}^{A} \quad (2.27)$$

$${}^{P}\underline{P}_{B}^{A}=\left[{}^{B}C^{A}\right]{}^{P}\underline{P}_{A}^{A} \quad (2.28)$$

$${}^{P}\underline{V}_{B}^{A}=\left[{}^{B}C^{P}\right]{}^{P}\underline{V}_{P}^{A}=\left[{}^{B}C^{A}\right]{}^{P}\underline{V}_{A}^{A} \quad (2.29)$$

$${}^{P}\dot{\underline{V}}_{B}^{A}=\left[{}^{B}C^{P}\right]{}^{P}\dot{\underline{V}}_{P}^{A}=\left[{}^{B}C^{A}\right]{}^{P}\dot{\underline{V}}_{A}^{A} \quad (2.30)$$

2.2 Dynamic Modeling of Aerospace Vehicles

Dynamic models are mathematical representations that describe and predict vehicle behavior, including both the motion of the vehicle and its subsystems. The time varying properties of the model that are governed by differential equations are known as states. Dynamics generate the time derivatives for the state vector as functions of the state vector $\underline{X}$, controls $\underline{u}$, and time t as expressed in equation (2.31).

$$\dot{\underline{X}} = f\left(\underline{X}, \underline{u}, t\right) \tag{2.31}$$

The particular subset of the state vector, comprising position, orientation, velocity and angular velocity, will be referred to as the vector of motion states, $\underline{X}_m$, and depicts the motion of the vehicle with respect to reference frames that represent the spatial environment and other entities in the simulation. The following subsections will describe the relation between the vehicle's motion states with the reference frames commonly used in aerospace simulation, the contribution of reference frame to the kinetics and kinematics of dynamic models and a typical software implementation of a six degree of freedom (6DOF) dynamic model.

2.2.1 Common Reference Frames in 6DOF Dynamic Models

Common reference frames in aerospace simulations include the body fixed frame, the body carried frame, the navigation frame and the inertial frame [9]. The body fixed frame has its position and orientation fixed to the vehicle body. It is typically defined with its origin fixed to a reference point on the vehicle body and is oriented such that the *x*-axis points to the nose of the aircraft, the *y*-axis points to the right or starboard wing

and the *z*-axis points down. The body carried frame has its position fixed to the vehicle body, typically to the reference point used by the body fixed frame, but its orientation is fixed to the navigation frame. In this thesis, the body fixed frame and body carried frame will be collectively referred to as the body frames. The motion parameters of a dynamic model's body frames are defined with respect to a navigation frame. The choice of navigation frame is arbitrary and depends upon modeling decisions made during the development of the dynamic model. 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 translating with respect to the inertial frame. Figure 2.3 illustrates the relation between the body frames and the navigation frame.

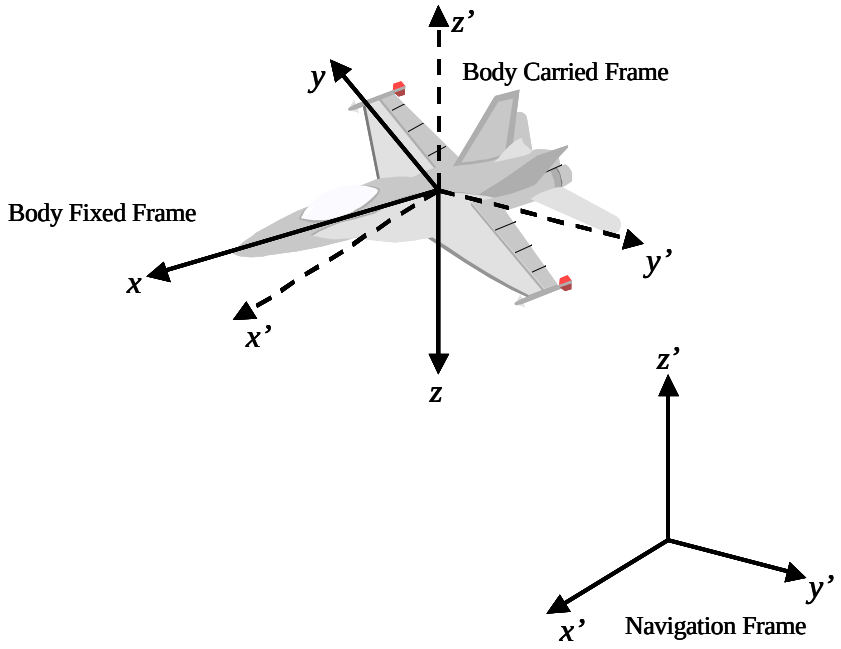


Figure 2.3: Body and Navigation Frames of 6DOF Dynamic Models

The motion states of the dynamic model may be treated as the respective motion parameters of the body frames with respect to the navigation frame. The position of the vehicle is expressed by the position of the body carried frame with respect to the navigation frame. Since the body frames share the same origin, the velocity of the vehicle can be expressed as the velocity of the body fixed frame or the body carried frame with respect to the navigation frame, depending on the desired measurement frame. The orientation and angular velocity of the vehicle are expressed by the respective parameters of the body fixed frame with respect to the body carried frame [9]. Thus, the body fixed frame uses the body carried frame as its definition frame, which in turn uses the navigation frame as its definition frame.

Several measurement frames can be used for position, velocity and angular velocity. Position is typically expressed in the navigation frame while angular velocity is expressed in the body fixed frame. Velocity can be measured in the body fixed frame [11], which is useful in kinetics, or in the navigation frame, which is useful in kinematics. Since the orientation and angular velocity of the body carried frame is identical to the navigation frame, the orientation and angular velocity of the body fixed frame can also be expressed with respect to the navigation frame. Common expressions for motion states and their components in a Cartesian coordinate system are included in Table 2.3.

Table 2.3: Motion States of Dynamic Models as Motion Parameters of Body Frames

Motion State	Notation	Components
Position	${}^{bf}\underline{P}_n^n$ or ${}^{bc}\underline{P}_n^n$	$x\ ,y\ ,z$
Orientation	${}^{bf}\underline{Q}^{bc}$ or ${}^{bf}\underline{Q}^n$	$f,\ q,\ y$ or q_0, q_1, q_2, q_3 or $\underline{C}$
Velocity	${}^{bf}\underline{V}_{bf}^n$ or ${}^{bc}\underline{V}_n^n$	$u\ ,v\ ,w$
Angular Velocity	${}^{bf}\underline{w}_{bf}^{bc}$ or ${}^{bf}\underline{w}_{bf}^n$	$p,\ q,\ r$

2.2.2 Kinetics and Kinematics in 6DOF Dynamic Models

A typical dynamic model calculates the forces and moments generated by the physical properties of the vehicle and its environment such as gravity, thrust and aerodynamic forces. The dynamic model then calculates the time derivatives of motion states using kinetics and kinematics equations. Kinetics equations relate these forces and moments to acceleration and angular acceleration of the body frame with respect to the inertial frame through the application of Newton's Second Law [5][12]. Equations (2.32) and (2.33) represent the kinetics equations for dynamic models with a fixed mass and inertia tensor. Additional terms representing the change in mass and inertia tensor can be added to these equations as required.

$$\sum\left({}^{bf}\underline{F}_{bf}\right)= m\left({}^{bf}\underline{\dot{V}}_{bf}^{i}+{}^{bf}\underline{w}_{bf}^{i}\times{}^{bf}\underline{V}_{bf}^{i}\right) \tag{2.32}$$

$$\sum\left({}^{bf}\underline{M}_{bf}\right)=\left[{}^{bf}I_{bf}\right]{}^{bf}\underline{\dot{w}}_{bf}^{i}+{}^{bf}\underline{w}_{bf}^{i}\times\left(\left[{}^{bf}I_{bf}\right]{}^{bf}\underline{w}_{bf}^{i}\right) \tag{2.33}$$

Kinematics equations in dynamic models relate the velocity, acceleration, angular velocity and angular acceleration of the body frames to the time derivatives of their

motion parameters, corresponding to the motion states of the vehicle. The implementation of the kinematics equations in dynamic models, as expressed by equations (2.34) to (2.37), differs from the kinematics equations used to change the definition frame of reference frames as expressed by equations (2.20) to (2.25), although both sets of equations utilize the motion between three reference frames.

$$\left[{}^{bf}\dot{Q}^{n}\right] = f\left(\left[{}^{bf}Q^{n}\right], {}^{bf}\underline{w}_{bf}^{n}\right) \tag{2.34}$$

$${}^{bf}\underline{\dot{P}}_{n}^{n} = \left[{}^{bf}C^{n}\right]^{T}\,{}^{bf}\underline{V}_{bf}^{n} \tag{2.35}$$

$${}^{bf}\underline{\dot{w}}_{bf}^{n} = {}^{bf}\underline{\dot{w}}_{bf}^{i} - \left(\left[{}^{bf}C^{n}\right]\,{}^{n}\underline{w}_{n}^{i}\right)\times{}^{bf}\underline{w}_{bf}^{n} - \left[{}^{bf}C^{n}\right]\,\frac{{}^{i}d}{dt}\left({}^{n}\underline{w}_{n}^{i}\right) \tag{2.36}$$

$$\begin{aligned}{}^{bf}\underline{\dot{V}}_{bf}^{n} = {}^{bf}\underline{\dot{V}}_{bf}^{i} + {}^{bf}\underline{w}_{bf}^{i}\times\left(\left[{}^{bf}C^{n}\right]\,{}^{n}\underline{V}_{n}^{i}\right) + {}^{bf}\underline{w}_{bf}^{n}\times\left(\left[{}^{bf}C^{n}\right]\left({}^{n}\underline{w}_{n}^{i}\times{}^{bc}\underline{P}_{n}^{n}\right)\right)\\ -\left(\left[{}^{bf}C^{n}\right]\,{}^{n}\underline{w}_{n}^{i}\right)\times{}^{bf}\underline{V}_{bf}^{n} - \left[{}^{bf}C^{n}\right]\left(\frac{{}^{i}d}{dt}\left({}^{n}\underline{w}_{n}^{i}\right)\times{}^{bc}\underline{P}_{n}^{n} + \frac{{}^{i}d}{dt}\left({}^{n}\underline{V}_{n}^{i}\right)\right)\end{aligned} \tag{2.37}$$

The exact form of equation (2.34), relating the time derivative of orientation to the angular velocity, depends upon the representation of orientation. The time derivatives for Euler angles and quaternions representations are expressed by equations (2.38) and (2.39) respectively.

$$\begin{bmatrix}\dot{f}\\ \dot{q}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}1 & \tan q\sin f & \tan q\cos f\\ 0 & \cos f & -\sin f\\ 0 & \sec q\sin f & \sec q\cos f\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{2.38}$$

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix} \bullet \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{2.39}$$

The fidelity of the kinetics equations may be impacted by the choice of reference frames when applying Newton's Second Law [5][12]. While a purely inertial may be defined in theory, 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 frames for the fidelity requirements at hand, even when they are using the same dynamic model.

2.2.3 Typical Software Implementation of 6DOF Dynamic Models

The software implementation of 6DOF dynamic models combines elements that generate time derivatives of the state vector with a numerical integration routine and handle interactions with other simulation components and the environment. Reference frame transformations, including the kinematics equations and rotation matrices for changing the definition and measurement frames of motion parameters to or from reference frames that the model may interact with, are also implemented in the model. The following process can represent the calculation of the derivative vector and its subsequent integration, as illustrated in Figure 2.4.

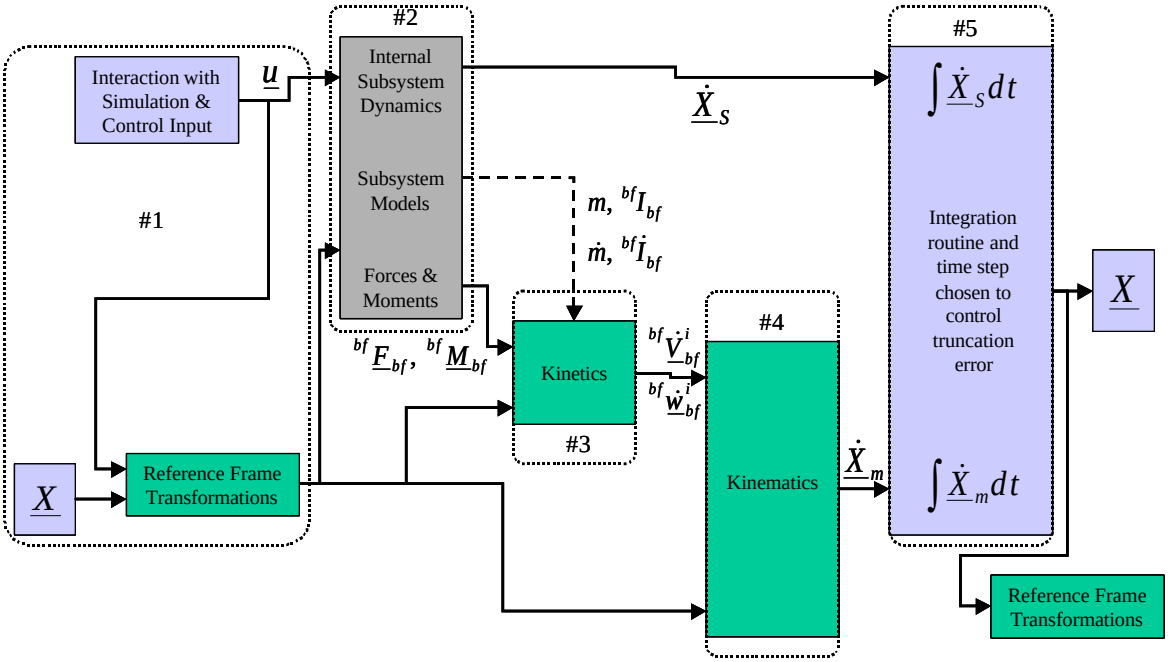


Figure 2.4: Typical Software Representation of 6DOF Dynamic Models

1. The model collects inputs from the simulation environment and other simulation components of interest, such as other dynamic models. This input may require reference frame transformations involving kinematics and rotations to be expressed in and defined relative to the same reference frames used by the dynamic model.
2. Subsystem models use the states and information from the simulation to implement the internal subsystem dynamics. These dynamic calculations provide the forces and moments and the derivatives of the internal subsystem states.
3. The forces and moments generated by the subsystem models are then used by the kinetics equations, as expressed in equations (2.32) and (2.33), to calculate the acceleration and angular acceleration of the body with respect to the inertial frame. These calculations also require the inertial properties of the body, including mass,

inertia tensor and their rates, as well as the motion parameters of the body frame with respect to the inertial frame.

4. These inertial accelerations are used by the kinematics equations, along with the motion parameters of the body and navigation frames, to calculate the derivatives for the motion states as expressed in equations (2.34) to (2.37).
5. Once the derivatives are calculated, a numerical integration routine is applied to calculate the state vector at the end of the time step. Depending upon the integration routine used, the derivatives may need to be recalculated several times at different stages in the routine. If the dynamic model requires numerical error to be controlled, it is often achieved through the use of an integration routine utilizing adaptive time steps. Common adaptive time step routines include Runge-Kutta-Fehlberg (RKF) [13] and Runge-Kutta-Cash-Karp (RKCK) [4] methods that modify the time step required by the dynamic model to control the local truncation error [13], which is the error per time step due to the use of discrete time steps during numerical integration.

2.3 Numerical Integration and Numerical Error in Simulation

Dynamic models use the numerical integration of time derivatives to propagate their states during the course of the simulation. Different numerical integration methods are available and the choice of method affects the numerical accuracy of the dynamic model. The use of numerical integration in computer-based simulation introduces two types of numerical error: truncation error and roundoff error. The following subsections briefly describe the common integration methods in aerospace simulations, the types of error introduced and some of the methods used to reduce these errors.

2.3.1 Numerical Integration Methods

Numerical integration is used to propagate the states of a dynamic model in simulation given their time derivatives. In these routines, time is segmented into discrete time steps, which may or may not be uniform. In general, these routines calculate the value of $\underline{X}$ at the $(n+1)^{th}$ time step from the value of the $\underline{X}$ at the n^{th} time step as follows:

$$\underline{X}\{n+1\} = \underline{X}\{n\} + \Delta\underline{X}\{n,\ n+1\} \tag{2.40}$$

The incremental term $\Delta\underline{X}\{n,\ n+1\}$ represents the change in state estimated over the interval $\{n,\ n+1\}$. 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 $\dot{\underline{X}}\{n\}$ and Δt. The order of an integration routine is a measure of how closely the method estimates the behavior of the incremental term over the time step. Higher order methods, such as higher order Runge-Kutta (RK) integration routines [4][13], obtain a more accurate estimate of the incremental term, improving the accuracy of the dynamic model for a given time step. Adaptive time step variants, such as the Runge-Kutta-Cash-Karp method, use additional terms of higher order to evaluate the error in integration and adjust the time step to keep the error within specified bounds. A typical simulation loop using numerical integration is illustrated in Figure 2.5.

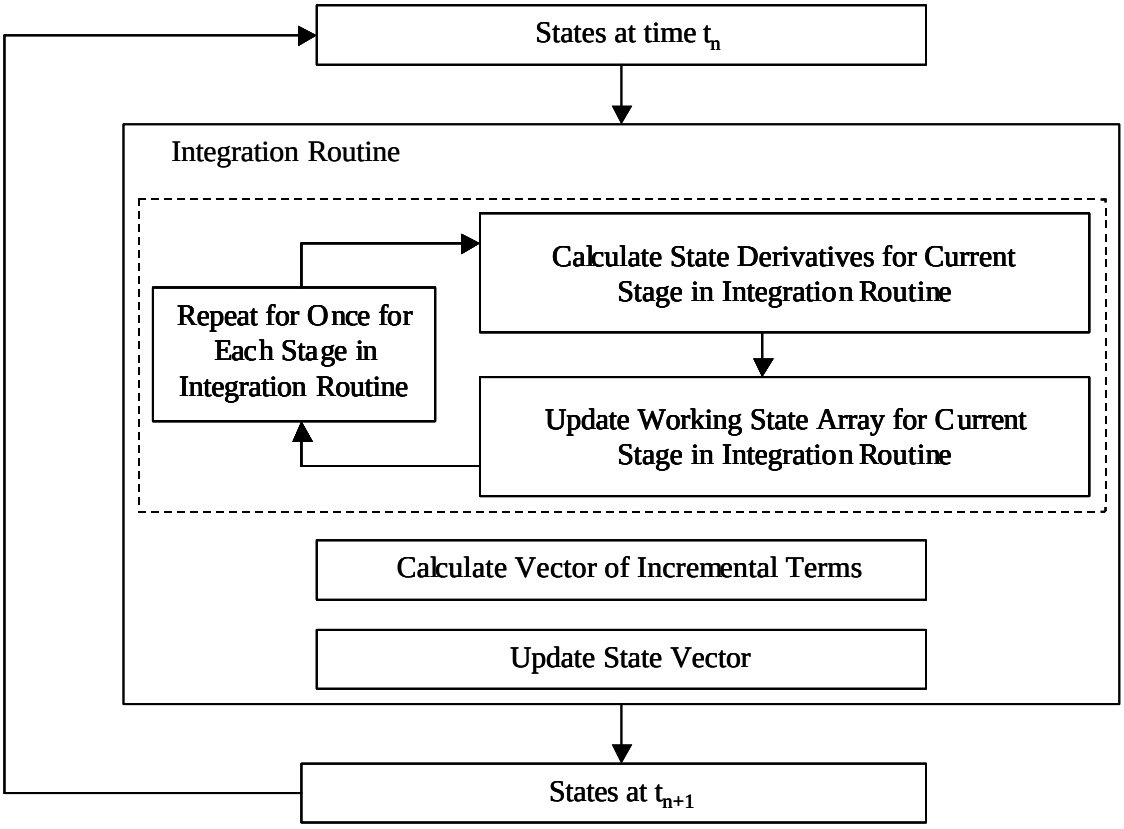


Figure 2.5: Typical Simulation Loop

2.3.2 Truncation Error in Numerical Integration

Numerical integration methods, by approximating continuously evolving dynamics to discrete increments of time, can incur a form of numerical error termed truncation error in this thesis. The magnitude of this truncation error per time step, also known as local truncation error [13], $\Delta \underline{X}_{LTE}$, scales with the order of the method and time step, as seen in equation (2.41).

$$\left|\Delta \underline{X}_{LTE}\right| \approx O\left(\Delta t\right)^{order+1} \tag{2.41}$$

There are two standard methods for reducing local truncation error. The first method is to use higher order numerical integration methods [4]. The other method is to reduce Δt so that the derivatives are propagated over smaller intervals; however, this requires more simulation steps for a given duration of simulated time, and thus provides more opportunities to accumulate truncation error. Thus, while the second method will reduce local truncation error, it may not reduce 'global' error over the entire simulation run [13].

2.3.3 Floating-Point Variables and Roundoff Error in Simulation

The binary representation of floating-point numbers by a discrete number of bits approximates a continuous space with a discrete space, introducing another type of numerical error termed as roundoff error in this thesis [4]. Floating-point numbers are represented by a positive integer called a mantissa *M*, a positive integer *e*, and a sign bit. 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 [4], as depicted in Figure 2.6.

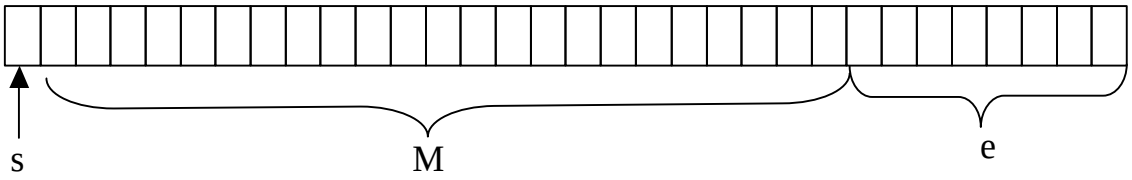


Figure 2.6: Representation of Bits in a 32 Bit Floating-Point Number

The sign bit, *s*, 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, eliminating the need for an additional bit to determine the sign of the exponent. The base value, B, typically 2 or 16, is raised to the power of the exponent. This value is then multiplied by M to generate a positive floating-point number as shown in equation (2.42).

$$X = s \times M \times B^{e-E} \tag{2.42}$$

The machine roundoff error is determined by the machine accuracy, e_m. This value is essentially the value of the least significant bit that is stored in the mantissa and is also called the precision of the variable. The machine accuracy is defined as the smallest value that can be added to 1.0 without being lost [4]. Table 2.4 lists the number of bits used to represent single precision and double precision floating-point numbers by IEEE Standard 754 compliant machines and their machine accuracies.

Table 2.4: Number of Bits and Machine Accuracy for Floating-Point Numbers

Property	Single Precision	Double Precision
Total Bits in Variable	32	64
Bits in Mantissa	23	52
Bits in Exponent	8	11
Machine Accuracy	1.19×10^{-7}	2.22×10^{-16}

The relative roundoff error, representing the ratio of the roundoff error to the actual value, is bounded by machine accuracy. The maximum roundoff error $\Delta \underline{X}_{Rnd}$ can

be estimated by multiplying the number and machine accuracy, as depicted by equation (2.43). To accurately determine the maximum roundoff error, a bit-wise analysis of the number is required. The Most Significant Bit (*MSB*) represents the first bit in the 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 roundoff error. The difference in the exponents of the *MSB* and *LSB* in an *N*-bit mantissa is *N*-1 and the maximum value for $\Delta \underline{X}_{Rnd}$ can be calculated using equation (2.44).

$$\Delta \underline{X}_{Rnd} \approx \underline{X} \times e_m \tag{2.43}$$

$$\Delta \underline{X}_{Rnd} = 2^{(\text{int})\log_2(\underline{X})-N+1} \tag{2.44}$$

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 roundoff error. When two floating-point numbers with different exponents are added and subtracted, some of the data contained in the number with the smaller exponent may be lost. The exponent of the smaller number is set to the value of the larger number's exponent, resulting in a 'right shifting' of the bits in the smaller number's mantissa. As a result, the LSB of the smaller number has the same exponent as the larger number's LSB. Since the size of the mantissa is fixed, data originally contained in the smaller number's mantissa whose exponents are less than the exponent of the new LSB are lost. Therefore, $\Delta \underline{X}_{Rnd}$ of the larger number forms the upper bound for the roundoff error generated during each addition or subtraction operation.

To illustrate the effect of roundoff errors, assume that an 8-bit floating-point number has 4 bits in the mantissa and 3 bits in the exponent, a bias of 4 and a base of 2. Expressing the number 45.5 in binary as [1011011] requires 7 bits with the LSB having an exponent of -1. However, the value stored in the 4-bit mantissa is [1011] and the last 3 bits are lost. The LSB has an exponent of 2 and the maximum roundoff error is 4. The actual value stored is 44 and the actual roundoff error is 1.5, corresponding to the 3 least significant bits, [011], that were lost. Extending this example to arithmetic operations, if this number represent a state that is to be propagated and its incremental term for a given time step is $\dot{X} \times \Delta t_1 = 7$, the roundoff error can be calculated by evaluating the values of the bits lost when the incremental term is 'right shifted', as illustrated in Figure 2.7.

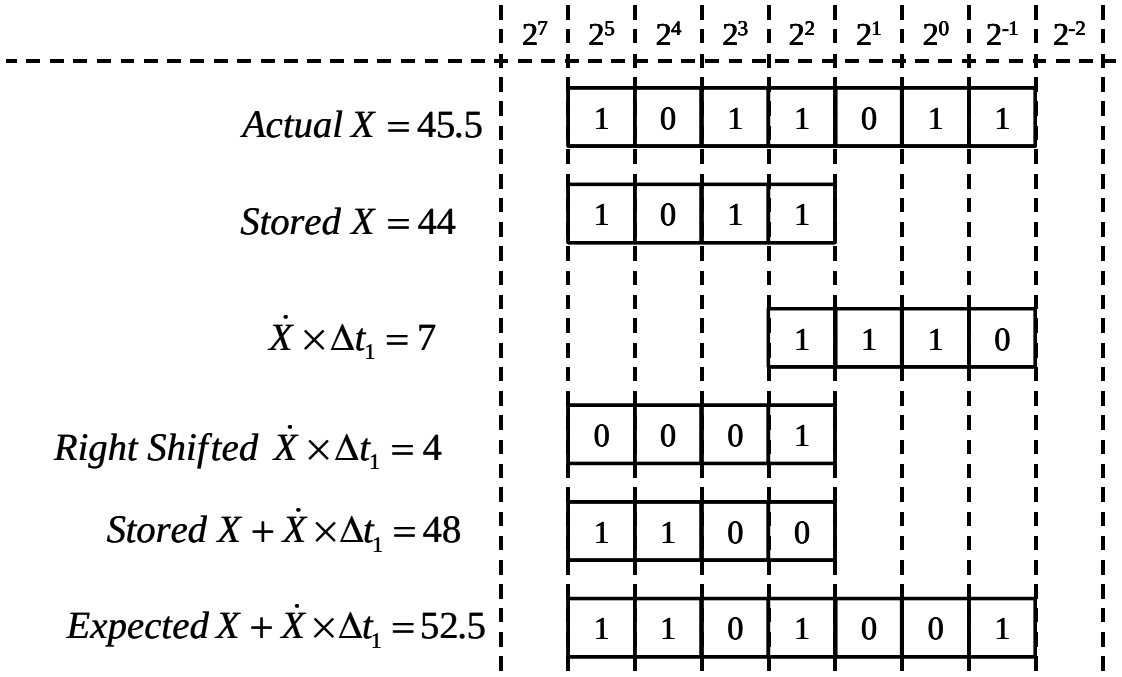


Figure 2.7: Roundoff Error in a 4 Bit Mantissa

The result of the addition using a 4-bit mantissa is 48 compared to the correct result of 52.5. The roundoff error incurred during the addition is 3 while $\Delta \underline{X}_{Rnd}$ due to X is the value of its LSB: 2^2 or 4. The remaining error is due to the loss of the last 3 bits when representing a number requiring a 7-bit mantissa with a 4-bit mantissa.

Since the limit for relative roundoff error depends upon the machine accuracy, improving machine accuracy by increasing the number of bits used to represent the floating-pointer number, especially the mantissa enables greater precision, thereby reducing relative roundoff error per operation. This is seen in Table 2.4 through the comparison of 32 bit single precision and 64 bit double precision numbers. Another method involves the use of higher bases and Fast Fourier Transforms to obtain arithmetic of arbitrary precision [4]. However, these methods often require specialized expertise on the part of the developer as new data types need to be developed along with their standard arithmetic operations, and can create software specific to specific types of problems or specific computational hardware.

2.3.4 Reduction of Total Numerical Error in Simulation

The local truncation and local roundoff 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 the states are typically coupled, to propagate into all aspects of the vehicle dynamics.

Numerical error can have two practical impacts on 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 simulation of very 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 may be problematic to simulate spacecraft in years-long interplanetary trajectories that end in a precise docking operation. Reducing numerical error, then, may enable longer duration simulation runs.

As mentioned earlier, the standard method for reducing truncation error for a given integration routine is to reduce time step. While this does reduce local truncation error, reducing the time step requires a larger number of simulation steps for a given duration of simulated time where both truncation and roundoff error may be accumulated, potentially increasing global error. In addition to increasing the occurrence of roundoff error, smaller time steps may have a detrimental effect on roundoff error as they reduce the size of the incremental term, implying that a larger fraction of the incremental term can be lost per time step. This fractional error may be expressed as the ratio of the LSB of the state to the MSB of the incremental term. If this value exceeds 1, the entire incremental term is lost and the actual dynamics are lost to roundoff error. Therefore it is not possible to reduce both types of error by changing only the time step. Figure 2.8 illustrates this concept and highlights how total numerical error (the sum of truncation and roundoff error) can be very large if the time step is made too large or too small. Instead, another method needs to be developed that will allow the integration method to control both the truncation error and roundoff error.

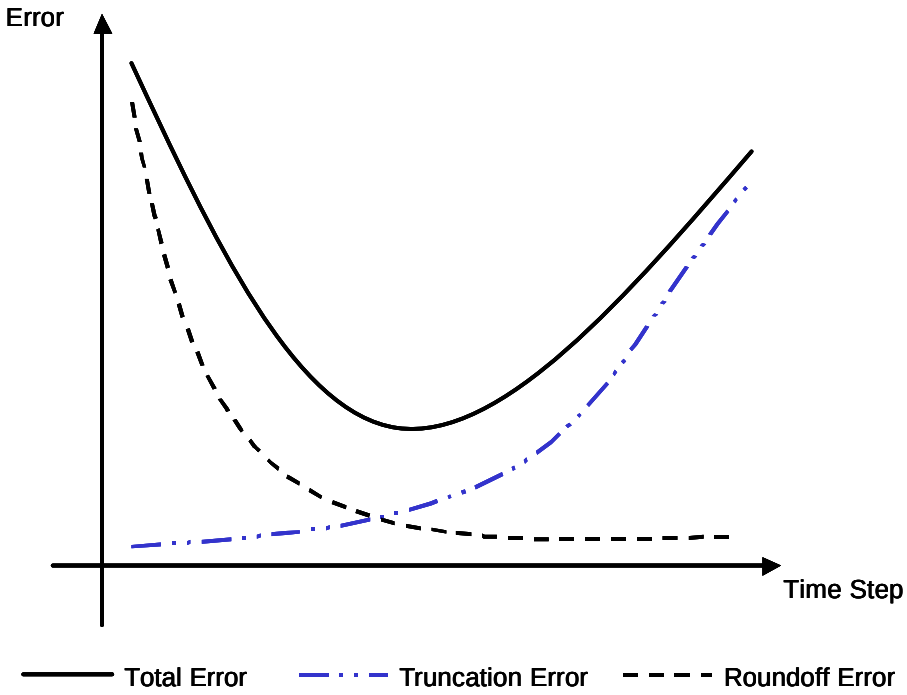


Figure 2.8: Schematic of Truncation, Roundoff and Total Error with Time Step

While algorithms using higher bases, Fast Fourier Transforms and larger mantissas may be used to reduce roundoff error, these methods often require specialized expertise and may create software specific to certain types of problems and computer architectures. Ideally, a method that is transparent to the developer of simulation components would facilitate control of numerical error in a manner that facilitates the use of those components in a wide variety of simulator configurations and on a range of computer architectures.

2.4 Parallel and Distributed Simulation

Parallel and distributed simulations (PDS) refer to the execution of simulations using a computational system of multiple processors connected by a communications network. In parallel simulation, the processors are often homogenous and have good

communications. Distributed simulations, on the other hand, often involve heterogeneous systems that may be geographically distant with communications that may involve significant lag [14]. The benefits of PDS include reduction in execution time by distributing computational load over several processors, the ability to run a simulation over geographically distant processors, the ability to integrate different simulators developed by different manufacturers, and improved fault tolerance [14]. The integration of different simulators that may be geographically distant is especially useful as it enables large-scale simulations with multiple human operators, such as military simulations of large exercises involving tanks and aircraft with their crews, and air traffic control simulation.

2.4.1 Evolution of Parallel and Distributed Simulations

Early parallel and distributed simulations were developed concurrently by several communities for different applications. The application of PDS to analytical simulations was developed for high performance computing at the same time that PDS was also used to develop Distributed Virtual Environments (DVE) by the military and computer gaming communities [14]. The development of PDS for military applications will be described, as it is most relevant to aerospace simulations.

SIMNET, developed for the Department of Defense in the 1980s, was the first successful implementation of large-scale, human in the loop simulation network for team training and mission rehearsal in military operations [15]. In SIMNET, each processor was treated as an autonomous node and data was broadcast to all nodes in the network. Each node used ‘dead reckoning’ models, described in the Section 2.4.3, to represent vehicles of interest on other nodes. Each vehicle broadcast its position, orientation and velocity to

update the other nodes' dead reckoning models. While the standard update rate was set to 15 updates per second, the average update was 1 update per second for ground vehicles and 3 updates per second for air vehicles, although the rate could increase to 15 updates per second during periods of rapid maneuvering [15]. SIMNET architecture and protocols evolved into Distributed Interactive Simulation (DIS) Standard Protocols (IEEE 1278-1993). Another development originating from SIMNET was the Aggregate Level Simulation Protocol (ALSP), which treated war games as analytical simulations resulting in the development of synchronization protocols [14].

The Distributed Interactive Simulation (DIS) evolved from SIMNET in the 1990s and was used to develop the infrastructure to link simulations of various types to create realistic, complex virtual worlds for simulating highly interactive activities [16]. The DIS protocols were used to connect independent computational nodes to create a coherent synthetic world that had consistent time and space representations. These protocols included network communications services, data exchange protocols called Protocol Data Units or PDUs, and common databases and algorithms [16].

The current standard for PDS is the High Level Architecture (HLA). Development of HLA began in 1995 and by September 1996 HLA was designated as the standard architecture for the Department of Defense with all its simulations being HLA compliant by 1999 [14]. HLA represents the integration of the analytical simulation architecture, represented by ALSP, and the DVE architecture, represented by DIS. Until the development of HLA, development of analytical simulations and DVE had proceeded independently of each other [14]. HLA is implemented through the Run Time Infrastructure (RTI) software and will be described in further detail in the next section.

While the development from SIMNET to HLA dealt primarily with the connection and data passing protocols between individual processors on a network, the development of a shared simulation environment was handled by the Synthetic Environment Data Representation and Interchange Specification (SEDRIS) project. The need to develop a representation of the shared simulation environment to achieve interoperability between heterogeneous simulations was identified by 1995 [17]. Without a shared environment, different representations of environment data may be used in the network. Interaction between processors involving the different representations would require expensive and time-consuming conversion operations utilizing software unique to each set of representations. Each conversion risks data loss or corruption and the number of these operations increased geometrically with the number of representations involved. The cost for development and maintenance of the conversion software would also be prohibitive [17]. Therefore, a common representation of the physical environment is needed as the level of interoperability depends upon the availability of consistent, complete and unambiguous definition of the environment data [17].

SEDRIS captures and provides a complete data model representing the physical environment that can be used by all simulations in the DVE. This data model is available to individual simulations through a pre-runtime distribution of source data, 3D models and integrated databases. The interfaces to this model are provided by an Application Programmer's Interface (API) software. The reuse of environment databases by different simulations is encouraged. These databases include terrain databases such as surface and volumetric data, feature data and 3D models. The simulation environment provided by

SEDRIS includes these databases, 12 standard coordinate systems and standard interactions to access and interact with the environment [17].

2.4.2 High Level Architecture (HLA) and its Implementation

The High Level Architecture (HLA) is the current standard for PDS and can be used to develop both analytical simulations as well as DVE simulations. In the HLA paradigm, each node in the network is referred to as a federate and the entire PDS is called their federation. While federates are typically distributed simulations, they can also represent hardware or even actual vehicles in the field [14]. Thus, the synthetic environment created by HLA enables military exercises that combine computer generated forces and manned simulators with actual vehicles such as aircraft and tanks. The ability to integrate such disparate systems into a single environment requires the architecture to clearly separate the semantics of each federate from the runtime interfaces and infrastructure of the environment. Therefore, HLA is defined by three concepts that ensure the separation of semantics and runtime interfaces. The three defining concepts are the HLA Compliance Rules, the Object Model, including the Object Model Template, and the Runtime Infrastructure (RTI) [14][18].

The Object Model is the non-runtime component of HLA and describes the objects used in the federation [14]. This description includes the attributes of each object, which represent data that each object is willing to share with other federates [18]. The Object Model is created using the Object Model Template. Each federate has a Simulation Object Model (SOM) that describes all the objects and their properties that are present in the federate. A Federation Object Model (FOM), describing all the objects and interactions in the federation, is then constructed from the SOMs within a particular

federation. The events that can occur in the simulation, called interactions, are also included in the models [18].

The Runtime Infrastructure (RTI) is the runtime component of HLA and is the software that provides the common services to each federate [14]. The emphasis in the implementation of RTI is interoperability between heterogeneous simulations and portability between platforms and operating systems. Therefore, RTI is independent of the semantics and provides only the following management services [18]. Federation management includes the creation or destruction of federations as well as the addition or removal of federates. Declaration management is used to declare desired objects and services by publishing and subscribing to attributes and interactions. Object management includes the creation and destruction of objects as well as the updating of attributes and delivery of interactions. Ownership management handles the ownership of attributes by objects in the federation. Time management is used to regulate the execution of the federation and the synchronization of federates. Time management applies the synchronization protocols developed in analytical simulations to control the execution of each federate. Interactions can be delivered in Time Stamp Order (TSO) to enforce causality, or in Receive Order (RO) where interactions are delivered in order of receipt by RTI [14][18].

The HLA Compliance Rules define the requirements of federates and federations to properly use the RTI. In particular, all object representations occur in federates, allowing the RTI to remain free of semantics [18]. Also, each federate is required to have a SOM while the federation is required to have a FOM. Additional rules cover the management of attributes and interactions [14].

As specified by the HLA Compliance Rules, each federate contains a SOM, built by the Object Model Template and connects to the RTI via the RTI API [19]. This API defines the interfaces to RTI and is written using the Interface Definition Language (IDL) of the Common Object Request Broker Architecture (CORBA). This API is compiled and included with each federate [19].

Objects in one federate are able to exchange data with objects in other federates using attributes and interactions. When a federate joins a federation, the attributes of all objects within the federate are published, informing RTI that the data can be accessed by other federates is necessary. If objects in other federates need to access any of these attributes, they needs to subscribe to the attributes of interest. These attributes are updated as required by the objects that publish them. Interactions, on the other hand, are not tied to specific objects. Interactions comprise of character strings and can be used to send a large variety of information, ranging from simple data sets to complex commands. Due to their nature, they often need to be interpreted by their recipient. Interactions can be sent to a specific federate, multicast to a group of federates or even broadcast to the entire federation. Typically, attributes are used to represent parameters within objects that are regularly updated, such as the states of dynamic models, while interactions generally represent events that may impact objects across several federates. Since the motion states of dynamic models are expressed with reference frames, their associated reference frames need to be identified or a single reference frame needs to be enforced on all motion states that are published. If federates publish the identities of the reference frames, a mechanism that handles the kinematics and rotations between them is required.

2.4.3 Dynamic Models and Dead Reckoning in PDS

The development of a DVE through a systematic definition of network and data passing protocols and a common representation of a shared environment enables distributed dynamic models to interact with one another. While some of these interactions may be for specific events, such as firing at a target vehicle in a military simulation, others may involve the sharing of data between dynamic models at every time step for a significant period of simulation time, such as continuously tracking the motion of an aircraft. While transmitting the motion states of dynamic models at every time step is possible, the resulting volume of network traffic can be prohibitive [14]. Furthermore, the simulations on each processor may be using different time steps, resulting in the interaction of models with different time stamps. Therefore, an efficient mechanism is needed that minimizes network traffic required while allowing synchronous interactions between models that may possess different time stamps. Since the motion states of dynamic models are governed by the equations of motion, dead reckoning models can be used to mirror the motion of a dynamic model on other processors. Therefore, frequent communication between federates is replaced by local computation [14].

Dead reckoning models use kinematics equations to model the motion of vehicles on other processors. Since a vehicle's dynamic model determines the variation of its acceleration with time, dead reckoning models need to assume constant accelerations. When the motion of the dead reckoning model deviates from the motion of the dynamic model by a specified error limit, the motion states and acceleration of the dead reckoning model are updated.

The motion of dead reckoning models can be updated using different methods [14]. The first method simply updates the motion states and accelerations with the new values. Consequently, the motion states can jump drastically to correct an error. Furthermore, the dynamic model and dead reckoning model may possess different time stamps, introducing errors in the dead reckoning model. In the second method, the motion states of the dynamic model are extrapolated to correct for any difference in time stamps and network latency. This method is more accurate but requires additional computation. The final method uses a smoothing function to generate a smooth transition from the trajectory using the old motion states and accelerations to the trajectory using the updated motion states and accelerations. It also accounts for differences in time stamp and network latency. While this method removes any abrupt changes in motion during an update, it requires significantly more computation than the other two methods.

Dead reckoning models are often implemented as a combination of Remote Vehicle Approximations (RVA) and Remote Vehicle Monitors (RVM). If a dead reckoning model represents the motion of a vehicle on another processor, it is called an RVA. In contrast, the RVM refers to the dead reckoning model of a vehicle that is being simulated on the same processor. While the use of an RVM implies that a processor simulating a vehicle needs to maintain a second model of its dynamics, it also enables the error in its dead reckoning model to be observed, facilitating the update and broadcast of the dead reckoning model to keep its motions within specified error limits. In SIMNET, local state updates and error evaluations were carried out at a rate of 15 Hz while the dead reckoning models were updated at an average of 1 Hz for ground vehicles and 3 Hz for aircraft [14][15].

2.4.4 Reference Frames and Coordinate Systems in PDS

The choice of reference frames and coordinate systems is critical in PDS as data pertaining to the motion of objects depends upon the reference frame and coordinate system used to describe their motion. Furthermore, a standardized Spatial Reference Model (SRM) is required for a consistent portrayal of the geophysical environment [20]. A standardized SRM includes reference frames and coordinate systems for expressing the motion of objects and environmental factors such as terrain. In addition to affecting the modeling of the geophysical environment, the choice of reference frames and coordinates systems also impact the kinetics and kinematics of dynamic models, inter-visibility of objects and the computational cost of integrating simulations and databases using different reference frames and coordinate systems [20]. The effect on kinetics and kinematics has already been discussed in section 2.2. The inter-visibility of objects is often used to filter data from vehicles that are too far away to be of interest, especially in large-scale simulations where line of sight and range may be important factors [21]. The ability to filter vehicles based on distance requires their position to be transformed to a common reference frame and coordinate system. The computational cost of these transformations also depends upon the choice of reference frames and coordinate systems used by the vehicles. Therefore, reference frames and coordinate systems need to be chosen judiciously to ensure interoperability between heterogeneous simulations without incurring prohibitive computational and development costs [17].

In early SIMNET simulations, reference frames were located on the surface of the earth in the vicinity of the exercise areas using Cartesian coordinate systems similar to the Military Grid Reference System (MGRS) or Universal Transverse Mercator (UTM)

projection [21][22]. Unfortunately, the use of Earth Surface Fixed (ESF) frames, while enabling efficient implementation of vehicle dynamics, assumed a flat earth and restricted the effective simulation area to a diameter of approximately 100 kilometers due to the Earth's curvature [20][21]. Also, integrating simulations using different ESF frames and coordinate systems posed significant technical challenges [23]. Later implementations of SIMNET used an Earth Centered Earth Fixed (ECEF) reference frame to define a Worldwide Coordinate System that could be used to share motion parameters across the network [21]. While a geodetic coordinate system, such as WGS84, models the earth's surface as an ellipsoid, its use of geodetic latitude, longitude and altitude required computationally expensive transformations if dynamic models and displays use different ESF frames. A geocentric Cartesian coordinate system, on the other hand, facilitated the filtering of distant objects without incurring significant computational costs during transformations [21].

In DIS, motion parameters of objects were expressed using an ECEF frame when shared with other simulations [22]. However, individual dynamic models and simulation components used different reference frames and coordinate systems for their internal calculations. Common reference frames included body frames for evaluating equations of motion, and earth fixed (ESF and ECEF) frames for evaluating the kinematics of dynamic models [22]. Cartesian coordinates, such as UTM, were used with the ESF, while geocentric Cartesian coordinates or geodetic spherical coordinates, consisting of latitude, longitude and altitude, were used with the ECEF frame. Due to the frequent transformations required between these reference frames and coordinate systems when sharing data across the network, a significant amount of research has been devoted to the

development of efficient transformation algorithms between these reference frames and coordinate systems [22][24].

The development of SEDRIS standardized the reference frames and coordinate systems available to individual simulations [17][25]. SEDRIS also provides efficient transformations services between these reference frames and coordinate systems [20][25]. However, these transformations are based on the Geospatial Reference Model within SEDRIS, restricting the available reference frames and coordinate systems to those provided by the SEDRIS API. While these may be sufficient for the simulation of Earth-bound vehicles or spacecraft orbiting the Earth, some simulation scenarios may require different sets of reference frames and coordinate systems. Also, all the simulations in the network still need to agree on a predetermined reference frame and coordinate system to publish motion parameters. However, this predetermined reference frame may not be suitable to all the scenarios and could introduce additional numerical errors.

2.5 Development and Reusability of Simulation Software

The development of software for a simulation often involves the integration of various simulation components, including dynamic models, control inputs and displays. Additional components such as agents may be used to model human elements such as pilots and air traffic controllers in large-scale agent based simulations. While some of these components may be unique to specific simulations, others may be applicable to a wide range of simulations, including distributed simulations. Reusing these components, or components from other simulations, can greatly reduce their development time and cost [26][27]. The following subsections will describe the benefits of software reuse along

with its costs and discuss its implications for developing dynamic models and simulation software.

2.5.1 Benefits, Costs and Metrics for Software Reuse

Software reuse has a significant effect on software development [2] and is one of the most promising methods for increasing productivity [3][26]. While software reuse also includes the reuse of specifications and designs used in software development [3][26], code reuse is often the most significant aspect and is the focus of much research. Software reuse can eliminate the need to re-implement common software elements [26]. Since some applications may contain only 15% application specific code, software reuse can provide significant savings in time and cost of software development [3][26]. In addition to reducing development time and cost, software reuse can also avoid the downstream cost of maintaining additional code [26]. The benefit of software reuse can be expressed as the ratio of reduction in development cost to baseline development cost [26], expressed in equation (2.45).

$$\text{Reuse Benefit} = \frac{\text{Cost without Reuse} - \text{Cost with Reuse}}{\text{Cost without Reuse}} \tag{2.45}$$

However, the development of reusable software incurs additional development cost that is amortized through its reuse in multiple software applications [3]. The development cost of reusable components can be 110% to 480% of the development cost of non-reusable components [2], depending on the type of application. This extra cost accounts for the higher quality of reusable assets [26]. Furthermore, the reuse of these

components also incurs additional costs, ranging from 10% to 63% of the development cost of non-reusable components [2], accounting for the identification, retrieval and integration of reusable software in different applications [3]. Also, documenting the capabilities of reusable software is essential as developers may be unaware of the full capabilities of the software and redevelop functionality, especially in large projects, reducing the effectiveness of software reuse [26].

The extent of reuse in software and its economic effect is often measured through the use of metrics. These metrics allow the reuse of software to be monitored over time and its effect on the productivity and quality of software [28]. Metrics can also provide insight into the development of reusable software and the effect of particular actions of the amount of reuse [28]. Different types of metrics can provide insights into the various aspects of software reuse. Metrics dealing with the reuse of code provide insights into the amount and type of code reused as well as the frequency of reuse [26][27][28], while other metrics based on economic models measure the effect of reuse on development cost [28]. Since this section primarily deals with the reusability of simulation components, code reuse is discussed in greater detail.

Both empirical and qualitative methods can be employed to study code reuse in software. Empirical methods involve the collection of objective data while qualitative methods often use subjective methods to evaluate how closely software adheres to specific standards [29]. Metrics for empirical methods studying code reuse typically involve the lines of code (SLOC) and components reused in the software [26][27]. While the number of components reused and the frequency of reuse provide a measure of code reuse in object-oriented programming [27], component based measures do not reflect the

size of the components reused. SLOC based measures, on the other hand, provide a measure of how much code is reused. This can be used to express the reduction in development effort through code reuse. A common metric for number of SLOC used is 'Amount of Reuse' or AOR [27] and is expressed by equation (2.46).

$$\mathrm{AOR} = \frac{\#\,\mathrm{SLOC}_{\mathrm{Reused}}}{\#\,\mathrm{SLOC}_{\mathrm{New}} + \#\,\mathrm{SLOC}_{\mathrm{Reused}}} \tag{2.46}$$

In this metric, the number of lines of reused code and the number of lines of new code. Since reusable assets are often more complex and contain more SLOC than non-reusable assets, this may not always provide an accurate measure of how much code the developer was saved from writing through code reuse. Instead, comparing the number of SLOC that need to be written for a software component with and without code reuse provides a better estimate of the reduction in development effort. However, this requires the development of non-reusable software for comparison. Therefore, the AOR metric is a more practical metric for monitoring reuse over the lifecycle of a reusable asset while the second method provides a more accurate metric of reduction in development effort at the cost of developing additional non-reusable software.

2.5.2 Software Reuse in Dynamic Models

Dynamic models are used to represent vehicles in a large variety of simulations. Since each simulation may use dynamic models to represent specific vehicles, a large number of dynamic models need to be developed to model the various vehicles in different simulations. Furthermore, some simulations may require dynamic models of the

same vehicle but with different fidelity requirements or different spatial environments, requiring the use of different reference frames. Hence, several dynamic models may be needed to represent the same vehicle in different scenarios. Therefore, the development of new simulations or the modification of existing simulations often requires significant time and effort for the development of dynamic models. Since software reuse can significantly reduce the time and effort required for software development, the identification and reuse of common functionality in dynamic models could facilitate the rapid development of dynamic models for different simulations. Models for different vehicles could be developed using a generic dynamic model that encapsulates the functionality common to all vehicle models.

By observing the functionality required by dynamic models in Figure 2.4, it can be seen that the numerical integration routine is independent of the dynamics of specific vehicles. Furthermore, inspection of equations (2.34) to (2.37) reveals that the kinematics equations depend solely upon the motion parameters of reference frames and the acceleration of the body frame with respect to the inertial frame. Since this acceleration is calculated by kinetics equations, the rest of the kinematics equations are based on the motion parameters of the navigation frame with respect to the inertial frame and the motion states of the body frame with respect to the navigation frame, as maintained in the dynamic model's state vector. If the reference frames can be treated as part of the simulation environment rather than being implicit to the model, the kinematics equations can be generalized to 6DOF dynamic models of all vehicles. Similarly, if the mass and inertia properties specific to the vehicle model and the force and moments generated by

its various subsystems, e.g. aerodynamic surfaces, engines, etc, can be calculated by the subsystem dynamics, the kinetics equations (2.32) and (2.33) can also be generalized.

While the process described above identifies the functionality common to all dynamic models, the dependency of this common functionality on reference frames, with the exception of numerical integration, complicates the development of a generic dynamic model. Since reference frames, specifically the navigation and inertial frames, are often implicit to the kinetics and kinematics of the model, rather than being explicitly defined in the simulation environment, separating kinetics and kinematics from the unique dynamics of the model is not feasible. Although existing software, such as AUTOLEV, enable the calculation of kinetics and kinematics to be automated [30], the reference frames need to be fixed during the development of the model. Furthermore, the kinematics and rotations between a model's reference frames and those used by other components need to be defined during its development, restricting the model's ability to interact with components using reference frames not specified during its development. These operations are duplicated in other dynamic models and simulation components that interact with these reference frames, causing duplication between components and incurring unnecessary development costs.

Similarly, while existing simulation software allow numerical integration to be automated and linked to other modules representing different elements in a dynamic model, once the model is constructed and compiled into an executable, the state and derivative vectors are fixed in the software implementation. This is further complicated by the fact that the dynamics of different subsystems may be coupled, resulting in states being shared by other subsystems.

The development of a generic dynamic model requires a paradigm that defines reference frames as part of the simulation environment. The motion parameters of the reference frames and the rotations and kinematics equations between reference frames need to be independent of dynamic models, enabling dynamic models to choose and access the relevant reference frames at runtime. Furthermore, the assembly of the state vector should be done in a manner that allows the contents of the vector to be identified and shared with the elements contributing to it.

2.5.3 Reference Frames and Interaction of Simulation Components

The interaction between different components in a simulation is typically carried out using interfaces. These interfaces may be specific to certain components or can be standardized to allow interaction with a broad range of components. While these interfaces enable data to be exchanged between components, the interpretation of data depends upon the data passing protocols between the components. This is especially true for standardized interfaces that may be accessed by a large number of components. In particular, the reference frames used to express motion parameters need to be specified for interfaces describing motion. If components express their motion parameters in different reference frames, rotations and kinematics equations are required to transform motion parameters to and from the reference frames set in the data passing protocols.

In a generalized case, each component may use a unique set of reference frames. Without an effective mechanism to coordinate these rotations and kinematics, point to point conversions will be required between each pair of components [17]. Each component will need to implement rotations and kinematics between its reference frame and the reference frames used by other components. The resulting development and component

integration effort would be extremely expensive and time consuming [17]. Figure 2.9 illustrates the point-to-point interactions required if all the simulation components use different reference frames and interact with one another. Therefore, a systematic and coordinated mechanism that handles the kinematics and rotations between all reference frames in the simulation is needed.

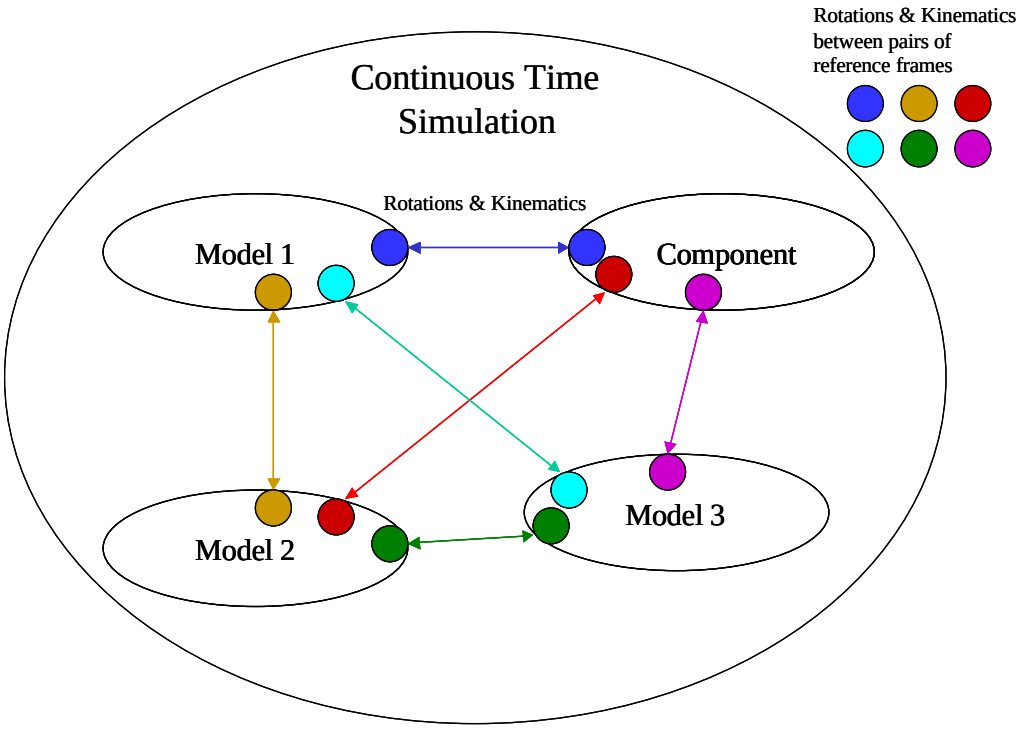


Figure 2.9: Point to Point Interactions Between Components

2.6 Summary of Issues with the Representation of Reference Frames in Simulation

Reference frames are essential for the expression of motion in simulation. However, these reference frames are often represented implicitly in a fixed manner by the simulation components and dynamic models that utilize them. Interactions between components may require expensive point-to-point data conversions, as illustrated by

Figure 2.9. Reconfiguring the simulation to include additional components using different reference frames may require the modification and redevelopment of existing simulation components. While Figure 2.9 depicts point-to-point conversions within a single simulation, it is even more relevant in PDS, where the simulation components on each federate may utilize different reference frames. While the reference frames used for exchanging motion parameters could be standardized, each component would then be expected to handle the appropriate kinematics and rotations between their preferred reference frames and the standardized reference frames. Also, the standardized reference frames may not be appropriate for all components or applications and may introduce additional numerical errors to the simulation.

Dynamic models also use reference frames to represent their navigation and inertial frames. Currently, these reference frames are implicit to and fixed within each dynamic model, limiting the ease with which these models may be reused or adapted to other applications using different reference frames or requiring different kinematic fidelity. Furthermore, using a single fixed navigation frame can lead to additional roundoff errors.

Therefore, a systematic representation of reference frames in the simulation environment should be developed that encapsulates the kinematics and rotations between all reference frames within the simulation environment and allows the addition of new reference frames without requiring existing components to be modified. Such a representation will enable simulations and their components to be rapidly reconfigured to different scenarios while facilitating the development of simulation components and dynamic models that can be reused and reconfigured for different simulations. Dynamic

models will be able to select reference frames based on the fidelity requirements of the simulation, reducing modeling error. Reference frames can also be selected to control total numerical error. Furthermore, allowing the individual components of various federates in a distributed simulation to use their preferred reference frames with an external mechanism handling all the kinematics and rotations within the federation would assist in the interoperability of distributed simulation.

CHAPTER 3

MANAGEMENT OF REFERENCE FRAMES

This chapter describes the development of a mechanism that assembles an extensible network of reference frames within the simulation environment. Treating reference frames, as well as their kinematics and rotations, as part of the simulation environment allows each simulation component to express motion using their preferred reference frame while allowing dynamic models to select their inertial and navigation frames as dictated by scenario and fidelity requirements. This mechanism also handles all the kinematics and rotations between reference frames, allowing simulation components to express motion parameters with respect to any reference frame in the network. A standardized representation of reference frames enables this network to be easily reconfigured to new models and scenarios, and to allow reference frames to be dynamically added or removed from the network during the simulation.

3.1 Network of Reference Frames

Reference frames in a simulation can be represented through an extensible network. The desired attributes for this network are:

1. The ability to rapidly expand and reconfigure the network
2. The ability to obtain consistent kinematics and rotations between arbitrary reference frames

The first attribute allows the network to be configured to meet the needs of different scenarios. The network is designed such that the addition or removal of reference frame from the network should not require further modification of other

reference frames in the network. The second attribute ensures that the same rotation matrix and kinematics equations are generated independent of the network path used to connect the corresponding pair of reference frames.

In a network, the entities represented by nodes and links depend upon the particular application. For example, a node may represent a data structure, a physical entity such as a computer [31], or simply a connection for network elements in the case of electrical circuits [32]. Similarly, a link in the network may represent mathematical relationships between nodes, data passing transmission lines, or even the components of electrical circuits. In a network of reference frames, the nodes represent the reference frames in the network and the links represent the modeling of relative motion between the reference frames. These models of relative motion, expressed through motion parameters, are maintained within the respective reference frames. Table 3.1 tabulates the representation of network components in a network of reference frames. The following subsections describe the selection of the network topology, the linking of nodes in the network, the standard operations required to create an extensible network, and the standard representation of reference frames within this network.

Table 3.1: Components Within a Network of Reference Frames

Components in a Network	Representation in a Network of Reference Frames
Node *('Vertex' in graph theory)*	Model of a reference frame: • Encapsulates its motion with respect to a definition frame through motion parameters • Propagates its own motion parameters • Identifies its definition frame • Updates the identity of child nodes during network assembly
Link *('Edge' in graph theory)*	Representation of relative motion between object and definition frame propagated using motion parameters • Stored in the node representing the object frame
Traversing a path connecting a pair of nodes	Change in definition frame or measurement frame used by a set of motion parameters • Kinematics equations are used to change the definition frame • A rotation matrix is used to change the measurement frame • The kinematics equations or rotation matrix is recursively constructed using the motion parameters of the nodes in the path

3.1.1 Selecting a Network Topology for Reference Frames

The geometry or topology of the network [31] that possesses the desired network attributes can be identified by observing the connections between nodes in a network. Following the general definitions used in graph theory [32], an arbitrary pair of vertices in a graph may be connected by either a single edge or a path defined by a sequence of edges, where the end vertex of each edge forms the initial vertex of the next edge [33]. Vertices connected by a single edge are called adjacent vertices [33]. In a connected graph, all vertices are connected through either edges or paths. Common topologies include star,

ring, tree and complete graph topologies [31][33][34], as depicted in Figure 3.1. The choice of topologies affects the number of edges and paths in a connected graph.

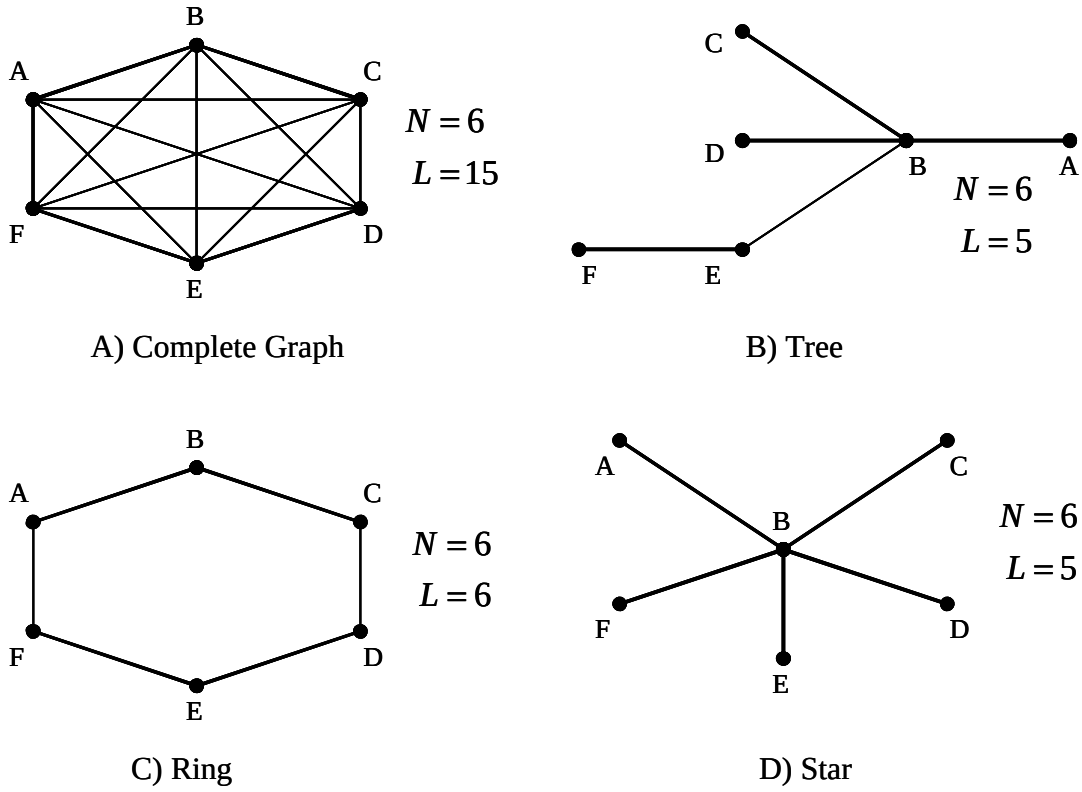


Figure 3.1: Number of Links in Common Network Topologies

The number of edges *L* in a connected graph of *N* vertices depends upon the topology of the graph. The complete graph topology requires the maximum number of edges, ${}^{N}C^{2}$ or *(N/2)´(N-1)*, in a graph of *N* vertices as each vertex is adjacent to every other vertex in the graph. In contrast, a tree topology only requires *N-1* edges [32][33]. Similarly, in a star topology, where *N-1* vertices are connected to a 'hub' vertex, *N-1* edges are required while a ring topology requires *N* edges, as depicted in Figure 3.1.

The number of paths connecting any pair of vertices also depends upon the choice of topology. Paths are defined such that edges are not repeated in the sequence [33]. The definition of a path shall be further restricted in this thesis such that vertices are not repeated, eliminating loops or circuits, in order to ensure consistency of kinematics and rotations between reference frames, which will be described in this section. By definition, the star and tree topologies have only one path between each pair of vertices while the ring topology has two paths. The complete graph topology has multiple paths connecting each pair of vertices.

Representing the propagation of relative motion between reference frames as links allows the edges and paths to represent the different ways in which the motion parameters between an arbitrary pair of reference frames can be calculated. If the vertices representing the reference frames are adjacent, their relative motion is represented by a single edge and the reference frames propagate the corresponding motion parameters, as described in Table 3.1. In contrast, if the vertices are connected through a sequence of edges, the motion parameters describing their relative motion is constructed through kinematics equations using the motion parameters represented by each edge in the sequence. Thus, a single edge implies that the reference frames encapsulate their relative motion whereas a path implies that the relative motion needs to be calculated through kinematics equations.

Reducing the number of edges not only reduces the number of motion parameters maintained for a given number of reference frames, but also reduces the need for the motion of each reference frame to be modeled with respect to multiple reference frames. In a complete graph, the motion of each reference frame needs to be modeled with

respect to all other reference frames in the network. Such a network would be difficult to expand, as a new reference frame would need its motion to be modeled with respect to all the reference frames in the network, requiring the models of all existing reference frames to be updated. Consequently, selecting a topology with the least number of edges for a given number of vertices improves the network's ability to be reconfigured rapidly for different scenarios, reducing the need to modify reference frames whenever the network is reconfigured, thus satisfying the first desired attribute.

The second desired attribute, requiring kinematics and rotations to be path independent, required the elimination of multiple paths between reference frames. Ensuring that paths between vertices are unique ensures consistent kinematics and rotations between reference frames. If a connected graph contains *N* vertices, a topology that only contains *N-1* edges, the minimum number of edges for connectedness, automatically ensures that all paths are unique while enabling the network to be reconfigured with minimal modification of reference frames. Based on these requirements, the complete graph and ring topologies can be eliminated due to the presence of multiple paths. Both the star and tree topologies only require the minimum number of edges and guarantee unique paths between pairs of vertices.

Comparing the star and tree topologies, it can be observed that the star topology uses one vertex as a hub and all other vertices are adjacent only to the hub. The implication for a network of reference frames is that the motion of all reference frames would need to be modeled with respect to the 'hub' reference frame. Furthermore, if a new reference frame is added, its motion can only be modeled with respect to the hub. While this may be acceptable for certain scenarios, in other scenarios the motion of

certain reference frames may need to be modeled with respect to other reference frames. In contrast, reference frames in a tree are not constrained to have their motions modeled with respect to a specified reference frame. If a new reference frame is added to a network using the tree topology, its motion can be modeled with respect to any reference frame in the network, allowing greater modeling flexibility. Therefore, the tree topology is the most suitable topology for developing a network of reference frames.

3.1.2 Linking Nodes in the Network

In a network of reference frames, the nodes represent the reference frames while the links represent the models of relative motion between the reference frames. While the tree topology provides an extensible network that reduces the dependencies between nodes, the links between the nodes must be handled judiciously to minimize modeling dependencies between each pair of reference frames.

While determining the network topology, the links between nodes were treated as edges between vertices, which did not possess any direction. A closer examination of these links requires the network to be represented as a directed graph or digraph, introducing direction to the edges [33]. Thus, the edges in a graph can be represented by two directed edges in a digraph where each directed edge represents the modeling of the relative motion with respect to one of the reference frame. If an edge between two vertices is represented by a single directed edge, the nodes are connected by a unidirectional link and, in this application, the relative motion between a pair of reference frames is only modeled with respect to one of them. If the edge is represented by two directed edges, the nodes are connected by a bi-directional link, and in this application, the motions of both reference frames are modeled with respect to each other. The link

between nodes A and B in Figure 3.2 is represented by two directed edges, signifying a bi-directional link whereas the link between nodes C and D is represented by a single directed edge, signifying a unidirectional link.

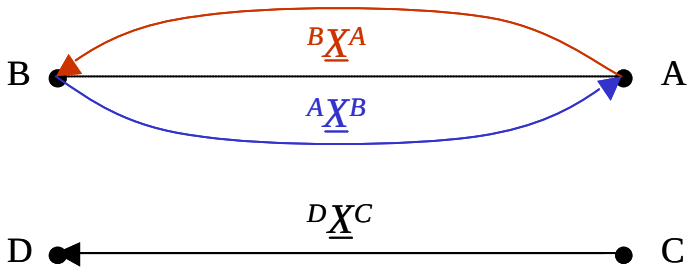


Figure 3.2: Unidirectional and Bi-directional Links Between Nodes

In the Figure 3.2, the bi-directional link shared by nodes A and B implies that two models describe their relative motion and both models must describe the same motion, albeit from opposite viewpoints. Thus, a modification in one of the models must be accompanied by an appropriate modification in the other model to ensure that the motion parameters of both models reflect the same behavior. Equations (3.1) and (3.2) applies the kinematics equations to the motion parameters of an object P, when its definition frame is switched from A to B and vice versa. The function $f({}^{P}\underline{X}^{A}, {}^{A}\underline{X}^{B})$ represents equations (2.20) to (2.25) and is generalized to all reference frames. Discrepancies between the relative motion of these reference frames, either due to modeling differences between ${}^{A}\underline{X}^{B}$ and ${}^{B}\underline{X}^{A}$, or numerical error in maintaining the motion parameters, produce differences between the initial and final motion parameters of P when its definition frame is switched from A to B and then back to A.

$$ {}^{P}\underline{X}^{B} = f\left({}^{P}\underline{X}^{A}, {}^{A}\underline{X}^{B}\right) \tag{3.1} $$

$$ {}^{P}\underline{X}^{A} = f\left({}^{P}\underline{X}^{B}, {}^{B}\underline{X}^{A}\right) \tag{3.2} $$

In contrast, nodes C and D in Figure 3.2 are connected by a unidirectional link, implying that only one model describes the relative motion between the reference frames, modeling the motion parameters of D with respect to C. If the motion parameters of C are required with respect to D, the same set of motion parameters are used, although from the opposite point of view. Since only one set of motion parameters is used to describe the motion, the kinematics and rotations between the reference frames are always consistent. Switching the definition frame of an object's motion parameters from D to C and back to D, as expressed by equations (3.3) and (3.4), will yield the object's initial motion parameters as the same set of motion parameters is used in both operations.

$$ {}^{P}\underline{X}^{C} = f\left({}^{P}\underline{X}^{D}, {}^{D}\underline{X}^{C}\right) \tag{3.3} $$

$$ {}^{P}\underline{X}^{D} = f\left({}^{P}\underline{X}^{C}, -{}^{D}\underline{X}^{C}\right) \tag{3.4} $$

Comparing the unidirectional and bi-directional links, the use of unidirectional links eliminates modeling dependency since a single model describes the relative motion between each pair of reference frames, ensuring consistency of kinematics and rotations between them. Likewise, the use of unidirectional links leads to the allocation of responsibility of maintaining the links within the network. The ability to treat trees as partially ordered structures with hierarchical levels [33] enables the systematic assembly of

the network and allocation of responsibility for maintaining its unidirectional links. The 'root' node forms the first level of the tree. New nodes branch out from existing nodes and are added to the next level. Consequently, a tree can be built up from its root node through the addition of nodes and links. Since a network using a tree topology has *N–1* links, adding a new node also adds a link, connecting the new node to the network. If the new node maintains this link, the network can be expanded without requiring any of its existing nodes to be updated. Furthermore, each node only needs to maintain the link that connects it to the network. Thus, each new reference frame only needs to model its motion with respect to one reference frame in the network: the definition frame of its motion parameters. Each new node is considered the child node of the node representing its definition frame. In Figure 3.3, frame A is the root node of the tree and the parent node for frame B, which is in turn the parent node for C, D and E in level 2.

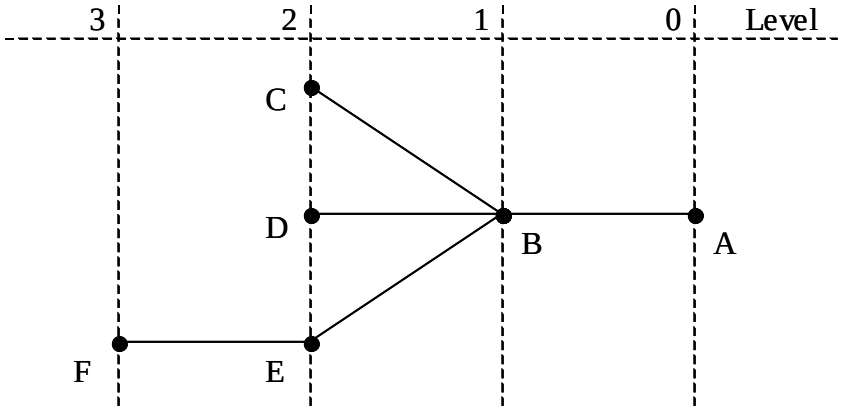


Figure 3.3: Levels in a Partially Ordered Tree

Each unidirectional link represents the relative motion of the child node with respect to its parent node. While these links can be viewed as vectors representing motion

parameters of the child node with respect to the parent node as depicted in Figure 3.2, each link is maintained by the child node and can be treated as an attribute of the child node. Therefore, the link can represent a ‘defined with respect to’ relationship and can originate from the child node, upon its addition to the network, as depicted in Figure 3.4.

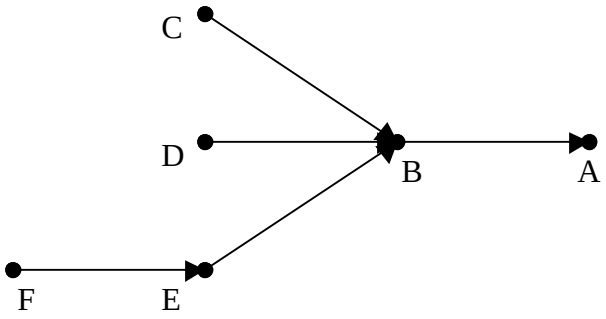


Figure 3.4: Links Add Child Nodes to Network

Assembling a tree with unidirectional links maintained by child nodes using hierarchical levels enables the formation of an extensible network that enforces consistency of kinematics and rotations. New reference frames can be introduced that use existing reference frames as their definition frames. If this definition frame is not available, the reference frame becomes the root node of a new network, implying that root nodes also have models of their motion parameters with respect to specific definition frames. If the node representing the definition frame is added to the network, the root node is linked to the new node. This new node may be added to another tree, allowing trees to be grafted to form larger networks. Conversely, removing a node that has several child nodes breaks a tree into several smaller trees. The ability to add and remove nodes

from a tree, as well as graft or prune trees, are standard network operations available to assemble an extensible network, described in the next section.

3.1.3 Standard Operations in an Extensible Network

An extensible network needs to support standard operations that enable the network to be modified through the addition and removal of nodes. Additional operations allow nodes to change their parent nodes, enabling the network to be reconfigured during the course of the simulation. These network operations include adding and removing nodes, grafting and pruning trees as well as changing parent nodes:

1. Trees are grafted when the root node of one tree uses a node located within another tree as its definition frame. The root node is linked to its definition frame and ceases to be a root node, merging the two trees. This is illustrated in Figure 3.5. If the root node of tree 2, E, uses D as its definition frame, D becomes the parent node for E and tree 2 is grafted to tree 1, forming tree 3.
2. When a link is broken, the child node is pruned from its parent's tree. The child node becomes a root node and its sub-tree becomes a new tree. This operation is typically carried out when the parent node is removed from the network or the definition frame is changed. In Figure 3.5, if the link between nodes D and E in tree 3 is broken, E becomes a root node of tree 2, and tree 3 is divided into 2 sub-trees, trees 1 and 2.
3. When a new node is added to the network, it may join an existing tree or form the root node of a new tree. If the node corresponding to the definition frame is available within the network, it becomes the parent node to the new node and the new node establishes the link to its parent node. If the node representing the definition frame is not available, the new node becomes a new root node, which could potentially form a

new tree. If the root node of another tree uses the new node as its definition frame, the grafting operation is used to graft this root node and its tree to the tree containing the new node. In Figure 3.6, node H uses D as its definition frame and is added to tree 1. The root node of tree 2, E, uses H as its definition frame and is grafted to tree 1.

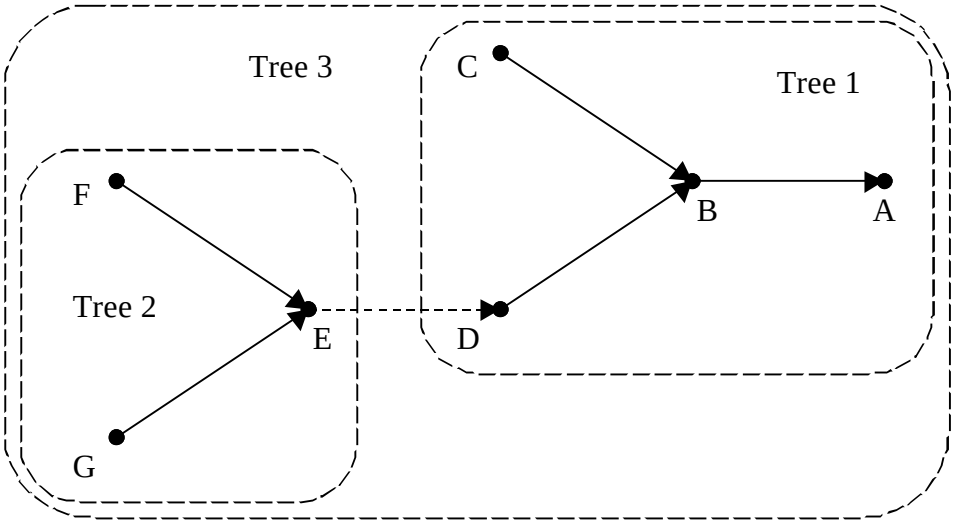


Figure 3.5: Grafting and Pruning Trees in a Network

4. When a node is removed, the network breaks the link between the node being removed and its parent node. If the node being removed is linked to child nodes, these links are also broken and the child nodes are pruned from the tree, becoming root nodes for their respective trees. In Figure 3.6, node H is removed from tree 3, and its child node, E, is pruned to form the root node of tree 2.
5. When the definition frame of a node is changed, kinematics and rotations reflecting the change in definition frames are typically applied to motion parameters. The node is then pruned from the tree, becoming the root node of a new tree. If the node

representing the new definition frame is available, the pruned node is grafted to its new definition frame.

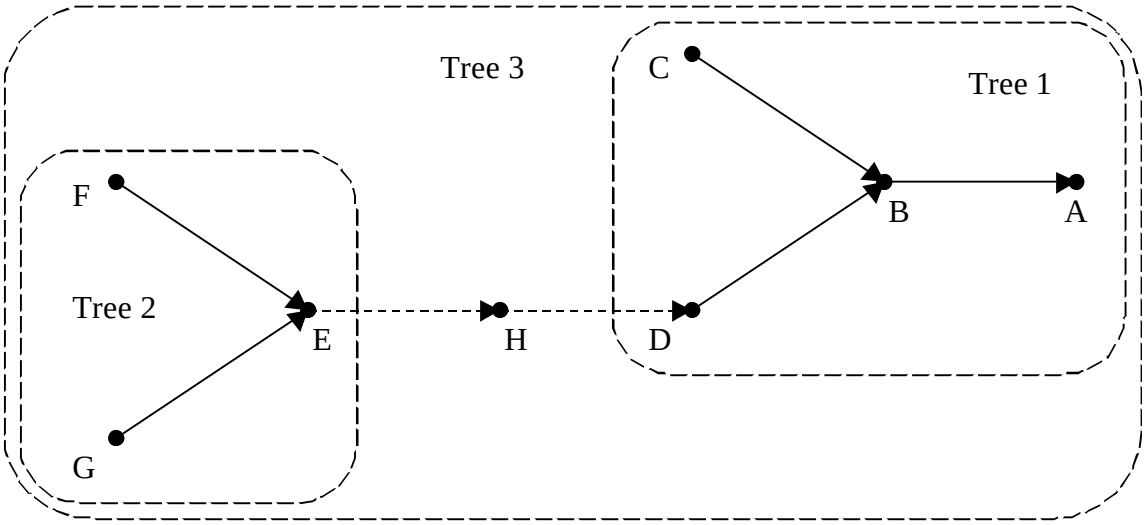


Figure 3.6: Adding and Removing Nodes in a Network

As a result of these standard operations, multiple root nodes can exist and may support multiple trees. While a path can link arbitrary nodes within a tree, nodes located in different trees cannot be linked. Thus, rotation and kinematics operations can only be executed between reference frames represented by nodes within the same tree. Ideally, there should be only one root node and tree in the network, although multiple trees could still be used if all simulation components in one tree do not need to access the motion parameters of any simulation components in another tree and vice versa.

Since the network uses a hierarchical tree configuration, it is important to ensure that closed loops are not formed, as these would create a paradox when determining levels within the tree. A closed loop will also introduce multiple paths between pairs of nodes, leading to the consistency problems discussed earlier. A closed loop can be

formed if the definition frame of a root node is located within its own tree as illustrated in Figure 3.7 where A chooses C as its definition frame.

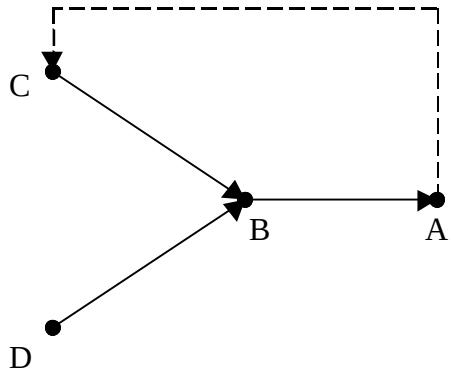


Figure 3.7: Closed Loop Within a Tree Network

The issue of closed loops within a network can be handled by either including mechanisms to enforce consistent kinematics in a closed loop or by preventing the formation of closed loops through judicious modeling of reference frames in the network. Mechanisms to handle closed loops would typically check whenever a new reference frame is added to the tree to see if a root node is using a node within its tree as a definition frame; if so, it would prevent the corresponding link from being established. While this method addresses the problem of consistent kinematics, it limits the extensibility of the network as the root node's ability to link to its definition frame is effectively removed, preventing it from being grafted to other networks. The alternate solution would be to model reference frames in a hierarchical manner mirroring the levels within the tree. One possible hierarchy could use the relationship between physical entities represented by the reference frames with inertial space and the scale of the environment typically expressed using the reference frames. For example, a reference

frame fixed on the Earth's surface could use the Earth Centered Earth Fixed (ECEF) frame as its definition frame. The ECEF frame, a rotating frame, in turn would use the Earth Centered Inertial (ECI) frame as its definition frame. Since the Earth orbits the Sun, the ECI frame could use the Heliocentric Inertial Frame as its definition frame.

3.1.4 Standard Representation for Reference Frames in a Network

In order for reference frames to be treated as nodes in the network described above, a standard representation needs to be developed that describes the unique properties of the reference frames while satisfying network requirements described above. While a reference frame's motion may be expressed with respect to any other reference frame, the network constraint for consistency stipulates that it may model its motion parameters with respect to only one definition frame, represented by its parent node. Furthermore, the reference frame should encapsulate the model of its motion parameters with respect to the definition frame. Thus, a reference frame can be treated as a unique entity whose motion parameters are modeled with respect to a specific definition frame. The identity of this definition frame is crucial as it forms the access point through which each reference frame joins the network. If the definition frame is chosen based on a physical hierarchy, closed loops within the network can be avoided, ensuring uniqueness of paths and consistency of kinematics and rotations.

The motion parameters of a reference frame are typically time variant, requiring propagation in time. The model of the motion parameters with respect to its definition frame can be generalized as a six degree of freedom (6DOF) model. While the acceleration and angular acceleration depends upon the specific reference frame's model with respect to its definition frame, the time derivatives of its position, orientation,

velocity and angular velocity can be generalized through the kinematics equations (2.34) to (2.37). Numerical integration is needed to propagate these motion parameters in time.

While the motion parameters of reference frames can be generalized as 6DOF models, the models of specific reference frames may be simplified, especially if they are not accelerating with respect to their definition frames. The motion parameters of these reference frames may be evaluated as algebraic functions of time rather than being propagated through numerical integration, reducing the roundoff errors introduced to the motion parameters and enhancing the accuracy of the reference frame's model.

3.2 Kinematics and Rotations in a Network of Reference Frames

Once a network of reference frames is created, the motion parameters of any object or reference frame can be expressed in or with respect to any other reference frame in the network. Since each node in the network is only linked to its parent and child nodes, an external mechanism is needed to create a path between the nodes. The mechanism that assembles and maintains the network, called the reference frame management mechanism, is used to identify the path between an arbitrary pair of the nodes. When the motion parameters of a reference frame are required with respect to a different definition frame or need to be expressed in a different measurement frame, a search algorithm is used to identify the path between the nodes representing these reference frames. Once the path is identified, the kinematics equations and rotation matrix is assembled by the reference frame management mechanism and used to change the motion parameters' definition frame or measurement frame. The following subsections describe the identification and assembly of a path using a search algorithm and the calculation of the kinematics and rotations along a path.

3.2.1 Assembling a Path Using a Search Algorithm

The search algorithm identifies the path connecting the nodes as a sequence of links from the initial node to the final node in the path. The initial node is the node corresponding to the reference frame currently used to express the motion parameters. The final node is the node corresponding to the desired reference frame. For example, if the motion parameters of reference frame A need to be expressed with respect to reference frame F, a path from node A to node F has to be identified, as depicted in Figure 3.8. Node A is the initial node in the path and node F is its final node. Once the sequence of links is identified, kinematics equations would be used to calculate the motion parameters of A with respect to F.

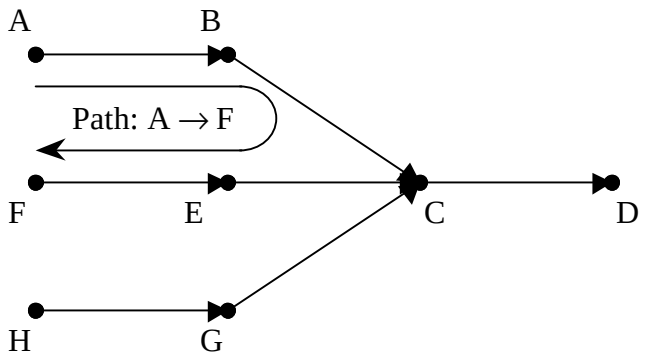


Figure 3.8: Transformation Path in a Network

The search algorithm utilizes the geometry of the tree topology and creates two search paths to identify the path linking the nodes. One path starts from the initial node and is called the forward path, while the other starts from the final node and is called the reverse path. Both paths traverse the tree towards the root node until a shared node is found. Once the shared node that links both paths is found, the forward and reverse paths

are merged. In Figure 3.9, the forward path links A to C, while the reverse path links F to C, the shared node.

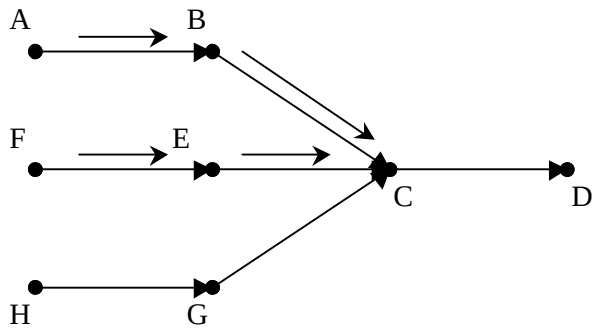


Figure 3.9: Forward and Reverse Paths Merge at Shared Node

While it is possible to use a search algorithm starting from the initial node to explore the tree until the final node is found, using forward and reverse paths is the most efficient algorithm as only the minimum number of nodes is traversed to identify the path. This is demonstrated in Figure 3.10, where the search algorithm starts from the initial node and explores all the branches in the tree, including the paths C to D and C to H, which are not in the path from A to F.

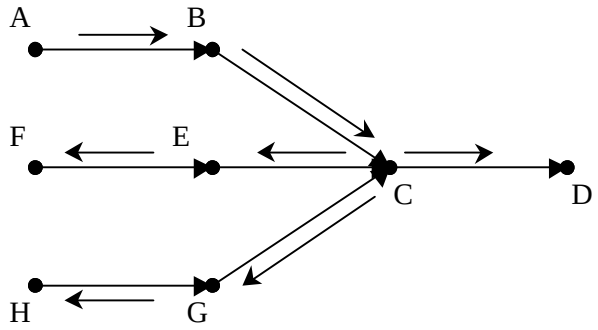


Figure 3.10: Exploring Branches in the Tree

Identifying the shared node when using forward and reverse paths is essential for merging the paths. The levels within a tree can be used to regulate the search algorithm since a parent node always has a lower level than its child node. When the search algorithm propagates the forward and reverse path, each path is assigned the level of its current node. The forward path is initialized with the level of the initial node and the reverse path is initialized with the level of the final node. The path with the higher level is propagated until both paths have the same level. If the nodes are the same, the shared node has been found. If not, both paths are propagated together. This method approaches the shared node from the initial and final nodes and traverses the minimum number of links between the nodes.

Once the shared node is identified, the forward and reverse paths are merged to form the path from the initial node to the final node. The forward, reverse and merged paths linking node A to node F in the network depicted by Figure 3.8 are as follows:

Forward path: $A \rightarrow B \rightarrow C$

Reverse path: $F \rightarrow E \rightarrow C$

Merged path: $A \rightarrow B \rightarrow C \rightarrow E \rightarrow F$

3.2.2 Evaluating Kinematics and Rotations along a Path

Once the path connecting the nodes is identified, kinematics equations and rotation matrices for changing the definition and measurement frames can be developed. Since the measurement frame defines the direction vectors used to express a vector as a set of scalars, changing the measurement frame involves the application of a rotation matrix, relating the orientation of the new measurement frame with respect to the current measurement frame, to the vector representing each motion parameter. Equation (3.5)

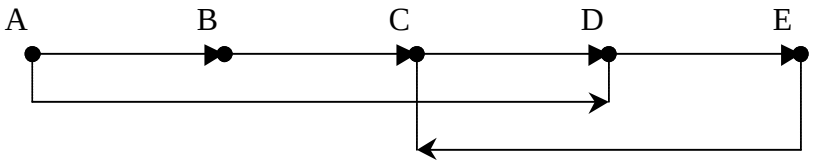
changes the measurement frame of a motion parameter from frame A to frame F using the direction cosine matrix (DCM) of F with respect to A as its rotation matrix. If nodes A and F in the Figure 3.8 represent frames A and F, the DCM of F with respect to A can be assembled by a sequence of rotations using the DCM corresponding to the orientation of each node along the path, as expressed by equation (3.6):

$$ {}^{P}\underline{X}_{F}^{A} = \left[{}^{F}C^{A} \right] {}^{P}\underline{X}_{A}^{A} \tag{3.5} $$

$$ \left[{}^{F}C^{A} \right] = \left[{}^{F}C^{E} \right] \left[{}^{E}C^{C} \right] \left[{}^{C}C^{B} \right] \left[{}^{B}C^{A} \right] \tag{3.6} $$

$$ \left[{}^{F}C^{A} \right] = \left[{}^{F}C^{E} \right] \left[{}^{E}C^{C} \right] \left[{}^{B}C^{C} \right]^{T} \left[{}^{A}C^{B} \right]^{T} \tag{3.7} $$

Since the standard notation used here dictates that the DCM formed through the orientation of a reference frame transforms a vector's scalar components from the definition frame to itself, the transpose of each DCM formed by nodes in the forward path is used, as expressed in equation (3.7). The network in Figure 3.8 shows a general case when both the forward and reverse paths are used. Linear networks or nodes along a chain require only one of the paths. Figure 3.11 illustrates these simpler paths. The rotation matrices for these paths are expressed by equations (3.8) and (3.9).



For Path: A →D

Forward Path: A→B → C → D

Reverse Path: <None>

For Path: E→C

Forward Path: <None>

Reverse Path: C →D →E

Figure 3.11: Forward and Reverse Paths Along a Chain

$$\left[{}^{D}C^{A}\right] = \left[{}^{C}C^{D}\right]^{T}\left[{}^{B}C^{C}\right]^{T}\left[{}^{A}C^{B}\right]^{T} \tag{3.8}$$

$$\left[{}^{C}C^{E}\right] = \left[{}^{C}C^{D}\right]\left[{}^{D}C^{E}\right] \tag{3.9}$$

Changing the measurement frame requires a transformation of direction vectors and is achieved through a sequence of rotations using the orientation of nodes in the path. Changing the definition frame, on the other hand, uses kinematics equations to express the motion parameters with respect to a new definition frame.

The position, orientation and angular velocity of a reference frame with respect to a new definition frame can be calculated using vector addition and rotations. The calculation of velocity, acceleration and angular acceleration with respect to a new definition frame, on the other hand, require the expressions for position, velocity and angular velocity to be differentiated with respect to time and the new definition frame. The resulting expressions use the motion parameters of the current definition frame with respect to the new definition frame to evaluate the motion parameters of the reference frame with respect to the new definition frame. The kinematics equations used when

changing the definition frame of object A from frame B to frame C are expressed by the following equations:

$$\left[{}^{A}C^{C}\right]=\left[{}^{A}C^{B}\right]\left[{}^{B}C^{C}\right] \tag{3.10}$$

$${}^{A}\underline{w}_{A}^{C}={}^{A}\underline{w}_{A}^{B}+\left[{}^{A}C^{B}\right]\,{}^{B}\underline{w}_{B}^{C} \tag{3.11}$$

$$\frac{{}^{C}d}{dt}\left({}^{A}\underline{w}_{A}^{C}\right)={}^{A}\dot{\underline{w}}_{A}^{B}+\left(\left[{}^{A}C^{B}\right]\,{}^{B}\underline{w}_{B}^{C}\right)\times{}^{A}\underline{w}_{A}^{B}+\left[{}^{A}C^{B}\right]\,\frac{{}^{C}d}{dt}\left({}^{B}\underline{w}_{B}^{C}\right) \tag{3.12}$$

$${}^{A}\underline{P}_{C}^{C}=\left[{}^{B}C^{C}\right]^{T}\,{}^{A}\underline{P}_{B}^{B}+{}^{B}\underline{P}_{C}^{C} \tag{3.13}$$

$${}^{A}\underline{V}_{A}^{C}={}^{A}\underline{V}_{A}^{B}+\left[{}^{A}C^{B}\right]\left({}^{B}\underline{w}_{B}^{C}\times{}^{A}\underline{P}_{B}^{B}+{}^{B}\underline{V}_{B}^{C}\right) \tag{3.14}$$

$$\begin{aligned}\frac{{}^{C}d}{dt}\left({}^{A}\underline{V}_{A}^{C}\right)&={}^{A}\dot{\underline{V}}_{A}^{B}+\left({}^{A}\underline{w}_{A}^{B}+2\left(\left[{}^{A}C^{B}\right]\,{}^{B}\underline{w}_{B}^{C}\right)\right)\times{}^{A}\underline{V}_{A}^{B}\\&\quad+\left[{}^{A}C^{B}\right]\left(\frac{{}^{C}d}{dt}\left({}^{B}\underline{w}_{B}^{C}\right)\times{}^{A}\underline{P}_{B}^{B}+{}^{B}\underline{w}_{B}^{C}\times\left({}^{B}\underline{w}_{B}^{C}\times{}^{A}\underline{P}_{B}^{B}\right)+\frac{{}^{C}d}{dt}\left({}^{B}\underline{V}_{B}^{C}\right)\right)\end{aligned} \tag{3.15}$$

If the frame B is not linked to frame C, the motion parameters of B with respect to C can be obtained recursively along the path connecting them using equation (3.10) to (3.15). While these equations express the relation between any pair of definition frames, these equations can only be used recursively within the forward path since the motion parameters of each node is expressed with respect to its parent node. The parent of a node within the forward path lies within the path in the direction of the final node, allowing the equations above to be generalized to any node within the forward path as depicted in

Figure 3.12. However, the parent of the shared frame is not in the path while parent nodes in the reverse path lead away from the final node.

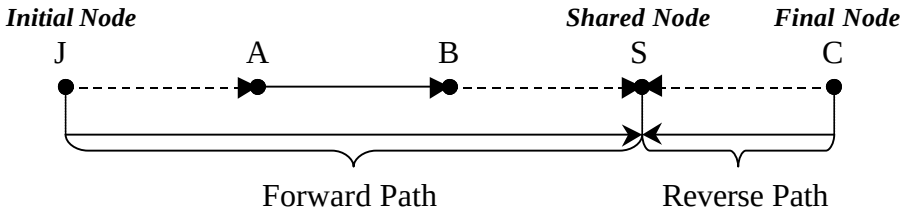


Figure 3.12: Generalizing Kinematics for Nodes in the Forward Path

The nodes in the reverse path can be generalized as depicted in Figure 3.13, where the parent node R, of node Q leads away from the final node. However, Q's child node, P, is the next node along the path to C, the final node.

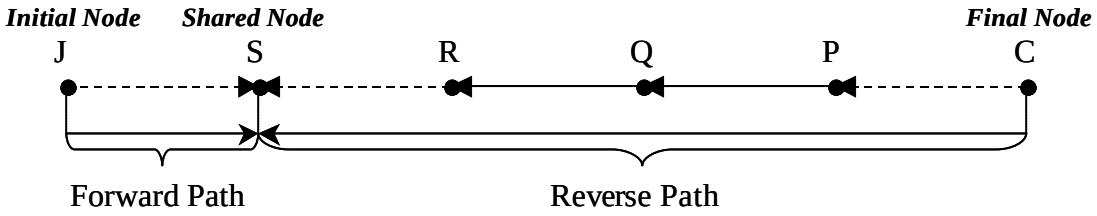


Figure 3.13: Generalizing Kinematics for Nodes in the Reverse Path

Since the motion parameters being differentiated are expressed in the form maintained and propagated by their respective reference frames, the equations above can be modified to calculate the motion parameters of node along the reverse path with respect to the final node. The motion parameters of node Q in Figure 3.13, representing

an arbitrary node in the reverse path can be expressed with respect to C using the motion parameters of P with respect to Q as expressed by the following equations:

$$\left[{}^{Q}C^{C}\right]=\left[{}^{P}C^{Q}\right]^{T}\left[{}^{P}C^{C}\right] \tag{3.16}$$

$${}^{Q}\underline{w}_{Q}^{C}=\left[{}^{P}C^{Q}\right]^{T}\left({}^{P}\underline{w}_{P}^{C}-{}^{P}\underline{w}_{P}^{Q}\right) \tag{3.17}$$

$$\frac{{}^{C}d}{dt}\left({}^{Q}\underline{w}_{Q}^{C}\right)=\left[{}^{P}C^{Q}\right]^{T}\left(\frac{{}^{C}d}{dt}\left({}^{P}\underline{w}_{P}^{C}\right)-\left({}^{P}\underline{\dot{w}}_{P}^{Q}+{}^{P}\underline{w}_{P}^{C}\times{}^{P}\underline{w}_{P}^{Q}\right)\right) \tag{3.18}$$

$${}^{Q}\underline{P}_{C}^{C}={}^{P}\underline{P}_{C}^{C}-\left[{}^{Q}C^{C}\right]^{T}{}^{P}\underline{P}_{Q}^{Q} \tag{3.19}$$

$${}^{Q}\underline{V}_{Q}^{C}=\left[{}^{P}C^{Q}\right]^{T}\left({}^{P}\underline{V}_{P}^{C}-{}^{P}\underline{V}_{P}^{Q}\right)-\left({}^{Q}\underline{w}_{Q}^{C}\times{}^{P}\underline{P}_{Q}^{Q}\right) \tag{3.20}$$

$$\begin{aligned}\frac{{}^{C}d}{dt}\left({}^{Q}\underline{V}_{Q}^{C}\right)=&\left[{}^{P}C^{Q}\right]^{T}\left(\frac{{}^{C}d}{dt}\left({}^{P}\underline{V}_{P}^{C}\right)-\left({}^{P}\underline{\dot{V}}_{P}^{Q}+\left(2{}^{P}\underline{w}_{P}^{C}-{}^{P}\underline{w}_{P}^{Q}\right)\times{}^{P}\underline{V}_{P}^{Q}\right)\right)\\&-\left(\frac{{}^{C}d}{dt}\left({}^{Q}\underline{w}_{Q}^{C}\right)\times{}^{P}\underline{P}_{Q}^{Q}+{}^{Q}\underline{w}_{Q}^{C}\times\left({}^{Q}\underline{w}_{Q}^{C}\times{}^{P}\underline{P}_{Q}^{Q}\right)\right)\end{aligned} \tag{3.21}$$

The kinematics equations for the forward and reverse paths allow the motion parameters of a reference frame to be expressed with respect to any other reference frame in the network. If the motion parameters of a reference frame need to be expressed with respect to a different definition frame, a path connecting the reference frame to its new definition frame is assembled using forward and reverse paths. Once the shared node linking the forward and reverse paths is identified, the first set of kinematics equations, (3.10 – 3.15), is applied recursively along the forward path until the motion parameters of the shared node with respect to the final node are required. These are then obtained by

recursively applying the second set of kinematics equations, (3.16 – 3.21), along the reverse path.

3.3 Effect on Dynamic Modeling

The ability to get motion parameters with respect to any reference frame enables the dynamic model to choose its navigation and inertial frames as required during runtime. However, to be able to effectively use the reference frame management mechanism the dynamic models need to adhere to certain standards.

First of all, each dynamic model has to be 'aware' of the reference frames it uses as its navigation and inertial frames. The choice of the navigation frame is often determined by the scenario requirements. The choice of the inertial frame is based upon the fidelity required in calculating inertial acceleration using Newton’s Second Law and can vary with the simulation scenario or during different stages of the simulation. Thus, the dynamic model must be aware of the navigation and inertial frames required, which may change at different stages of the simulation.

The dynamic model must also be aware of the effect of its navigation and inertial frames on its dynamics. Specifically, if the dynamic model requires a change in its navigation or inertial frames, it must be able to update its motion states and account for the change in reference frames in its kinetics and kinematics. For example, if the navigation frame needs to be changed, the dynamic model must be able to request a change in definition frame for its body frame and update its motion states. The force and moment equations may also need to be updated. Finally, the dynamic model should also account for the new navigation frame in its kinetics and kinematics equations. These equations can be generalized to any pair of navigation and inertial frames by obtaining

the motion parameters and time derivatives of the navigation frame with respect to the inertial frame from the reference frame management mechanism. While the reference frame management mechanism can calculate the motion parameters of the navigation frame with respect to the new inertial frame, the dynamic model should nevertheless ensure that the choice of inertial frames is appropriate for the scenario. If the model needs to adaptively select an inertial frame, it should evaluate the effect of different inertial frames on the kinematics and select the appropriate reference frame.

Since many elements of the kinetics and kinematics are independent of the attributes unique to specific dynamic models and can be automated in conjunction with a reference frame management mechanism, a generic dynamic model that handles the common functionality of dynamic models can be developed. The development and implementation of a generic dynamic model and the encapsulation of attributes unique to specific vehicles is discussed in the next chapter.

The dynamic model must also be able to access the interfaces provided by the reference frame management mechanism. Therefore, a standardized interface needs to be developed for the reference frame management system.

3.4 Interfaces and Implementation of a Reference Frame Management Mechanism

The object oriented analysis and design approach was used to create a reference frame management system that defines and adds reference frames to the simulation environment provided by the Reconfigurable Flight Simulator [35] (RFS), which is briefly described in Appendix A. This system consists primarily of base classes that are used to define the basic attributes and interface standards for reference frames, and a reference frame manger that assembles a network of reference frames and calculates the kinematics

equations and rotations between reference frames. These basic interface standards were compiled into a library that can be accessed by dynamic models and other simulation components, allowing them to access the network of reference frames and the kinematics equations and rotations, as depicted in Figure 3.14.

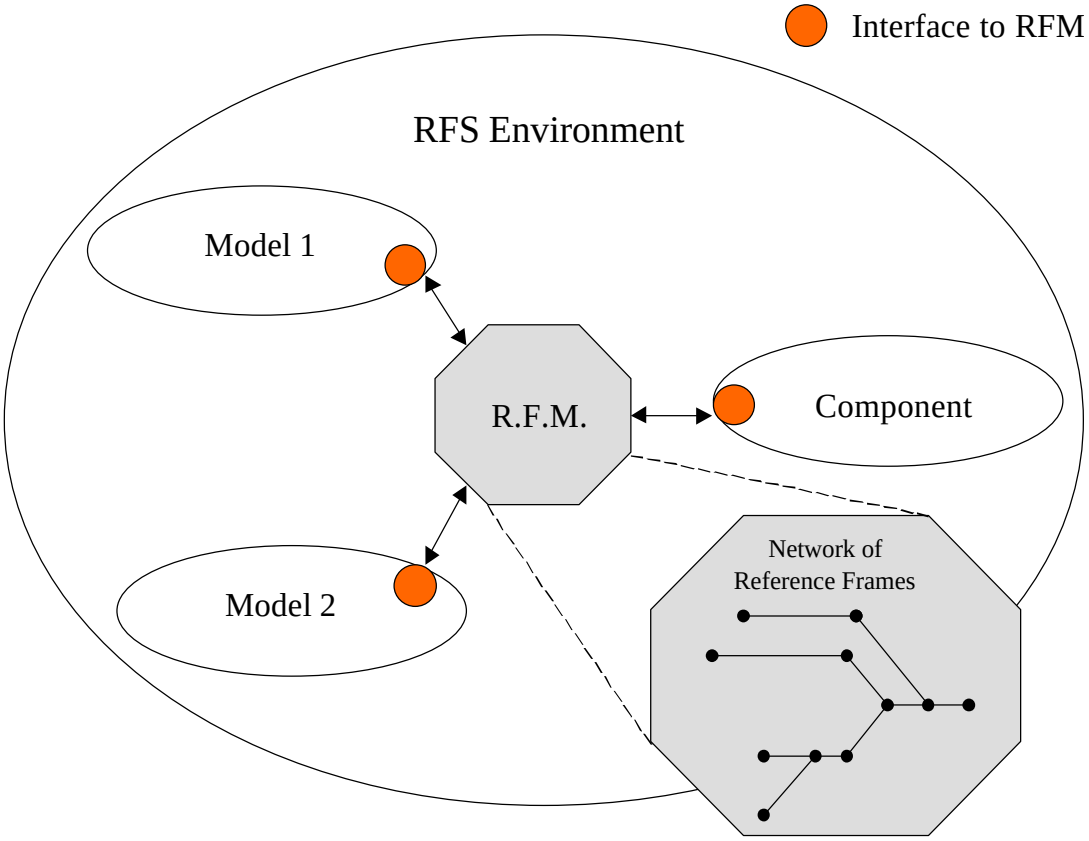


Figure 3.14: Reference Frame Manager in the Simulation Environment

The base interface class for the reference frame manager was implemented as a Reference Frame Manager class. This section describes the structure of the base classes and some of the core algorithms implemented in the Reference Frame Manager and its

operational responsibilities. The remaining implementation details of the various classes are provided in Appendix B.

3.4.1 Structure of the Reference Frame Manager

The reference frame management mechanism and the reference frames instantiated with RFS are located in the Environment Controller and Database (ECAD) Object. Specifically, the Reference Frame Manager replaces the Axis Definitions Object within the ECAD Object. All the vehicles within RFS are linked to the ECAD Object and can utilize the Reference Frame Manager to get the properties of the reference frames. The primary classes providing the standard interfaces of the reference frame management system are described below:

1. The ***Frame Definition Class*** is used to determine the interface standards and basic attributes of reference frames. All the reference frame objects used by the Reference Frame Manager inherit from this class. This base class is used to maintain the motion parameters and basic kinematics for calculating their time derivatives. Only the methods to calculate the acceleration and angular acceleration are *pure virtual functions* that need to be implemented for each reference frame class. The rest of the interfaces and functionality are implemented to create a base class that can be rapidly developed into any reference frame required by the simulation environment. The standard interfaces provide access to the reference frame's motion parameters, the identity of the reference frame and its definition frame, and allow the reference frame management mechanism to execute standard network operations on the reference frame. Other functionality within the base class allows the reference frame to be

propagated in time by the simulation environment. The 4th order Runge Kutta integration routine is also included for integrating the motion parameters.

2. The ***Frame Management Interface Class*** is used to provide the standard interfaces of the reference frame management mechanism to the simulation components such as vehicles and displays. The standard interfaces include methods that allow reference frames to be added to the network, methods to access reference frames as well as methods to calculate the kinematics and rotations between reference frames.

These interface classes are implemented to provide the reference frame management mechanism to the simulation environment in RFS through the following classes:

1. The Frame Definition Class can be implemented as ***Reference Frame Objects***. Since the interface class provides almost all the functionality of reference frames, each Reference Frame Object only needs to implement the methods that calculate its acceleration and angular acceleration with respect to a designated inertial frame. Depending on the nature of the reference frame's dynamics, a generic Reference Frame Object that can represent a family of reference frames can be developed, allowing its identity as well as the identity of its navigation and inertial frames to be set at runtime. Conversely, the Reference Frame Object may represent a specific reference frame using specific navigation and inertial frames. In these cases, functionality of the Frame Definition Class, such as the propagation of motion parameters, may be overloaded to optimize the behavior of specific reference frames.
2. The ***Reference Frame Manager (RFM)*** implements the standard interfaces of the Frame Management Interface Class as well as the functionality required to create and

manage a network using the standard network operations described in previous sections. Reference frame objects are loaded and added to the network. Paths between reference frames are created upon request and stored in a list of Frame Path Objects. Kinematics equations and rotations are assembled and executed upon request.

3. The ***Frame Path Object*** is used to store the forward and reverse paths between two reference frames. The initial, final and shared frames are used by the RFM to calculate the kinematics and rotations. Each path object is identified using the initial and final frames. Thus, there are two path objects for each pair of reference frames: only the direction of the path differs.
4. The ***Request Frame Change Object*** is used to pass relevant data between simulation components and the Reference Frame Manager when requesting motion parameters of reference frames with respect to arbitrary definition or measurement frames.

Figure 3.15 depicts the interaction of the RFM and its components with the ECAD Object in RFS while Figure 3.16 depicts the inheritance of these classes from standard RFS classes.

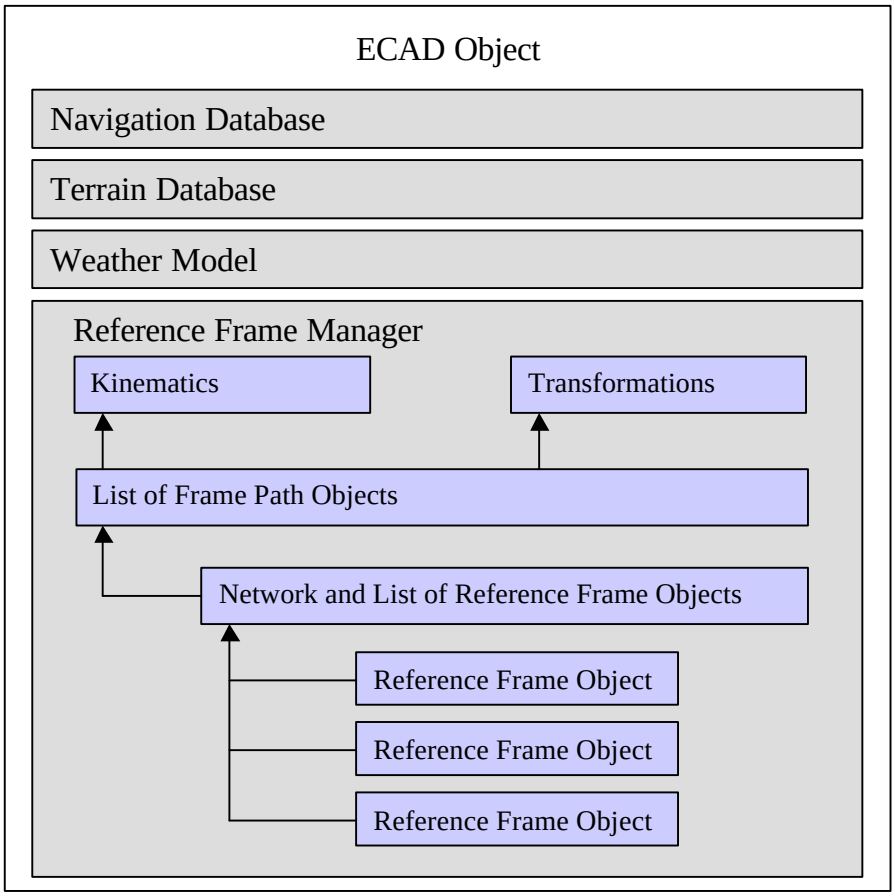


Figure 3.15: Component Interaction Diagram

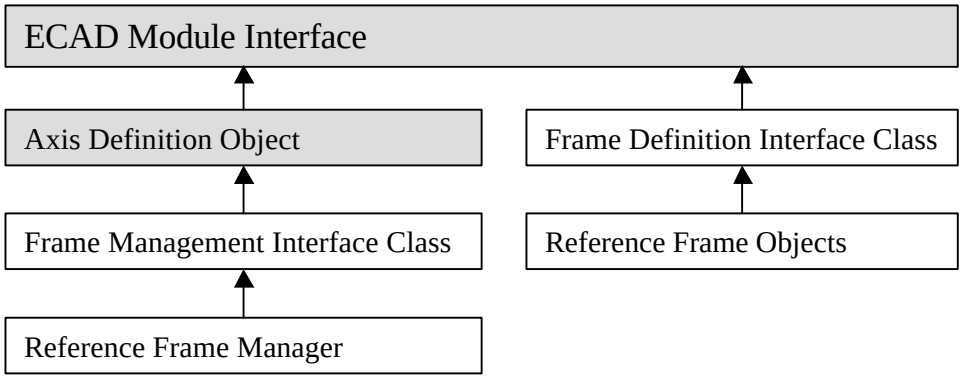


Figure 3.16: Object Inheritance Diagram

3.4.2 Operation of the Reference Frame Manager

The tasks of the Reference Frame Manager (RFM) within RFS consist of four major operations detailed below:

1. The RFM replaces the Axis Definition Object when loaded into RFS. Since the RFM inherits from the Axis Definition Object, default interfaces are still available to the vehicle objects. The various Reference Frame Objects are loaded and then registered with the RFM in a dynamic list. When a Reference Frame Object is registered with the RFM, it is added to the network.
2. The RFM maintains a network using a tree topology. The standard network operations described in Chapter 3.1.3 are used. The operations include the addition and removal of nodes from the network, grafting and pruning of trees and changing parent nodes. Each node is assigned a level within the tree using a recursive algorithm. The root node is level zero and the level of a child node is one greater than its parent node's level. The algorithm is applied to the root node of each tree within the network whenever an operation is carried out. The algorithm assigns the level of the node and applies itself to all the child nodes linked to that node.
3. When a path between a pair of Reference Frame Objects is requested, the RFM checks if the path exists in the Frame Path List. The RFM compares the pointers of the current and requested frames with the pointers of the initial and final nodes in each path stored in the list. If the path has not been generated, a Frame Path Object is created. Forward and reverse paths are created and the search algorithm described above is used to identify the shared node. Once the path is created, it is recorded in the Frame Path Object, which is then stored in the RFM for future use.

4. When a simulation component requires motion parameters to be expressed with respect to a different definition or measurement frame, it creates a Request Frame Change Object and passes it to the RFM. The Request Frame Change Object includes the name of the current and new reference frames. If the motion parameters represent the motion of an object in the current reference frame, they are copied into the request object. Otherwise the motion parameters of the current reference frame are expressed with respect to the new reference frame. If the request is for a change of measurement frames, the transformation is constructed by executing a sequence of rotations starting from the initial node to the final node as depicted in equation (3.6). The rotation matrices generated by nodes in the forward path are the transpose of their DCM, while the rotation matrices of nodes in the reverse path use their DCM. If the request requires a change in definition frame, the kinematics equations (3.10) to (3.21) are applied to nodes along the path.

CHAPTER 4

MANAGEMENT OF ROUNDOFF ERROR

Roundoff error due to the finite precision of floating-point variables and truncation errors due to numerical integration techniques are major sources of error in simulation. However, the standard method of reducing truncation error, by reducing the time step, has a detrimental effect on the roundoff error. Similarly, increasing the time step may reduce roundoff error at the cost of truncation error. Thus, a method of overcoming the coupling between the truncation error and roundoff error needs to be developed. Since the roundoff error is proportional to the magnitude of the state, a possible solution to reduce the roundoff error without adversely affecting the truncation error is to control the maximum value of the state during integration.

This chapter will discuss the development of an *intermediate frame* that can be used as the definition frame for the motion parameters of a body frame. The intermediate frame is updated when the states reach certain *critical levels.* These critical levels bound the maximum values of the motion states and are chosen such that the global roundoff error, representing the total roundoff error during the course of the simulation, is reduced. The magnitude of the critical levels affects both the update rate of the intermediate frame as well as the local roundoff error. Therefore, a large critical level reduces the number of updates but increases local roundoff error whereas a low critical level reduces the local roundoff error but increases the number of updates. Since each update also contributes to the global roundoff error, the critical levels are carefully chosen to minimize the global roundoff error.

4.1 Intermediate Frames

The reference frame manager can introduce intermediate frames that act as surrogates to the navigation frames used by dynamic models. The motion parameters of the intermediate frames are expressed with respect to the original navigation frames. The reference frame manager allows the dynamic model to switch the definition frame used by its motion parameters between the intermediate frame and the original navigation frame as required. Since the dynamics of the model typically require the motion states to be expressed in the navigation frame, the kinematics used to change the motion states' definition frame from the navigation frame to the intermediate frame and vice versa are called whenever the state derivatives need to be calculated. Therefore, the intermediate frame needs to be defined such that the computational cost and roundoff error incurred during the kinematics to and from the navigation frame are minimized.

4.1.1 Definition of Intermediate Frames

The intermediate frame is defined such that the difference in exponents between motion states of the dynamic model expressed in the intermediate frame and their incremental terms during numerical integration is bounded, thereby bounding roundoff error. The intermediate frame is updated when any element j of the model's motion states reaches its critical level $_j\underline{Cr}$; at that point, the corresponding element in the frame's motion parameters is shifted towards the vehicle by the value of that critical level. Since this is a discrete 'jump' in the motion parameters of the intermediate frame, the motion parameters of the vehicle's body frame are updated with respect to the intermediate frame and 'jump' by the same magnitude in the opposite direction. This ensures that the motion

parameters of the vehicle's body frame remain constant with respect to the original navigation frame when the intermediate frame is updated.

The intermediate frame's critical levels bound the maximum values of the motion states, thereby bounding the roundoff error during the propagation of these states. Since the states representing orientation are bounded by definition, this research will focus on bounding the states representing position and its derivative, velocity. To best reduce roundoff error in position and velocity, the intermediate frames should have the same orientation as its definition frame. If the orientations differ, the DCM of the intermediate frame will introduce additional floating-point operations across all axes. Furthermore, an angular velocity between these frames will introduce kinematics terms when switching the definition frame of the body frame's motion parameter between the intermediate frame and navigation frame. Both these operations may introduce additional roundoff error.

4.1.2 Effect of Intermediate Frames on Dynamic Modeling

The use of intermediate frames to reduce roundoff error affects several aspects of dynamic modeling, including the calculation of time derivatives of its motion states and any interaction with other simulation components. These aspects may require the model to be able to express its motion with respect to the navigation frame while maintaining its motion states with respect to the intermediate frame.

Dynamic models typically use the navigation frame as the definition frame for their body frames. Thus, the representation of these motion parameters in the navigation frame represents well-defined physical properties. Since the intermediate frame is introduced to reduce roundoff error, the motion states in the intermediate frame need to

be converted to the navigation frame to be physically meaningful. Thus, the dynamic model should be able to express its motion parameters with respect to the navigation frame at each stage of the simulation.

The motion parameters of the model's body frame are expressed with respect to the intermediate frame to control roundoff error during numerical integration. However, some of their time derivatives, specifically the acceleration and angular acceleration obtained from the kinetics equations, may be expressed with respect to the navigation frame, requiring the motion parameters to be expressed in the navigation frame. Since the acceleration and angular acceleration are expressed in the navigation frame, the kinematics of the intermediate frame with respect to the navigation frame are required to express them in the intermediate frame before they can be used in the integration routine.

For example, calculating the force of gravity on a satellite requires its position from the center of the attracting mass. Therefore, in an *earth-centered frame*, the position of the satellite has to be known with respect to the center of the earth, and not with respect to an intermediate frame that may be in close proximity to the satellite's body frame. The resulting acceleration needs to be expressed with respect to the intermediate frame, not the earth centered frame, when it is used to integrate the motion parameters of the satellite's body frame.

Thus, two sets of kinematics may be required to generate the accelerations of the body frame with respect to the intermediate frame. The first set is used to convert the motion parameters from the intermediate frame to the navigation frame, allowing the derivatives to be calculated. The second set is required to convert the derivatives from the navigation frame to the intermediate frame. If the intermediate frame maintains the

orientation of the navigation frame and does not rotate or accelerate with respect to it, the kinematics between the intermediate frame and navigation frame reduce to vector additions for position and velocity. This simplifies the first set of kinematics and eliminates the need for the second set since the accelerations are identical in the intermediate frame and navigation frame. Figure 4.1 illustrates the effect of the intermediate frame on the simulation loop for dynamic modeling.

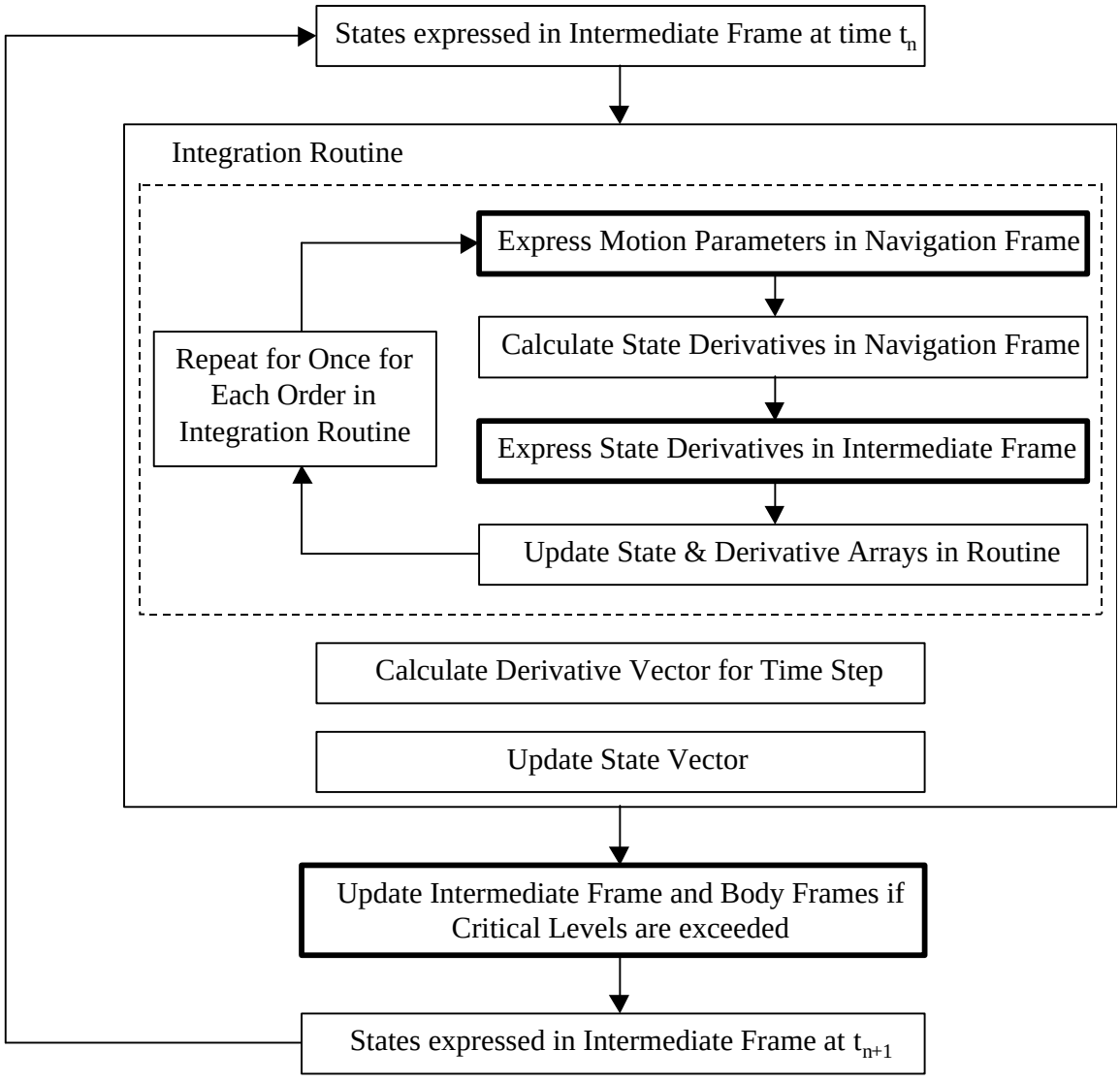


Figure 4.1: Simulation Loop With Intermediate Frame

Another aspect that needs to be considered is the interaction between multiple vehicles. Other simulation components may require the motion parameters of the dynamic model in its navigation frame or another reference frame. While this suggests that the model should store multiple values of its motion parameters with respect to different reference frames, the reference frame manager can be used by the other simulation components to express the model's motion parameters in their desired reference frame.

4.2 Critical Levels and the Reduction of Roundoff Error

Intermediate frames restrict the magnitudes of the motion states, thereby bounding local roundoff error. This section describes the role of critical levels in the operation of intermediate frames as well as their selection criteria. Estimating the local and global roundoff error through the use of intermediate frames is also described and utilized to develop an algorithm that allows critical levels to be selected adaptively during the course of the simulation.

4.2.1 Definition of Critical Levels

As a motion parameter is propagated, it may grow to the point that its magnitude is much larger than the incremental term. If the ratio of the incremental term to the motion state approaches machine accuracy, significant roundoff errors will occur. The critical level, therefore, is set to bound the magnitude of a subset of motion states relative to their incremental terms at every time step. Specifically, if the element j in the vehicle's motion states expressed with respect to the intermediate frame, ${}^{bf}_{j}\underline{X}^{IF}$, exceeds its critical level, ${}_{j}\underline{Cr}$, the corresponding element in the motion parameters of the intermediate

frame, ${}^{IF}_{j}\underline{X}^{n}$, is updated in a discrete jump. The vehicle's motion state is updated accordingly as follows:

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

$${}^{IF}_{j}\underline{X}^{n}_{IF} = {}^{IF}_{j}\underline{X}^{n}_{IF} + {}_{j}\underline{Cr} \times sign\left({}^{bf}_{j}\underline{X}^{IF}_{IF}\right) \tag{4.1}$$

$${}^{bf}_{j}\underline{X}^{IF}_{IF} = {}^{bf}_{j}\underline{X}^{IF}_{IF} - {}_{j}\underline{Cr} \times sign\left({}^{bf}_{j}\underline{X}^{IF}_{IF}\right) \tag{4.2}$$

In the equations above, the critical levels are measured in the intermediate frame because they determine the jumps taken by the intermediate frame. Since the intermediate frame has the same orientation as the navigation frame, the measurement frame used by the critical levels can be treated as either the intermediate or navigation frame. The choice of measurement frame by the motion states affects the application of the jumps to the body frame. While position is measured in the intermediate frame, the velocity of the body frame may be measured in the body frame itself. Thus, when evaluating critical levels and applying updates to the motion states, the velocity of the body frame needs to be measured in the intermediate frame and may require a rotation identical to the kinematics equation relating the time derivative of position to velocity.

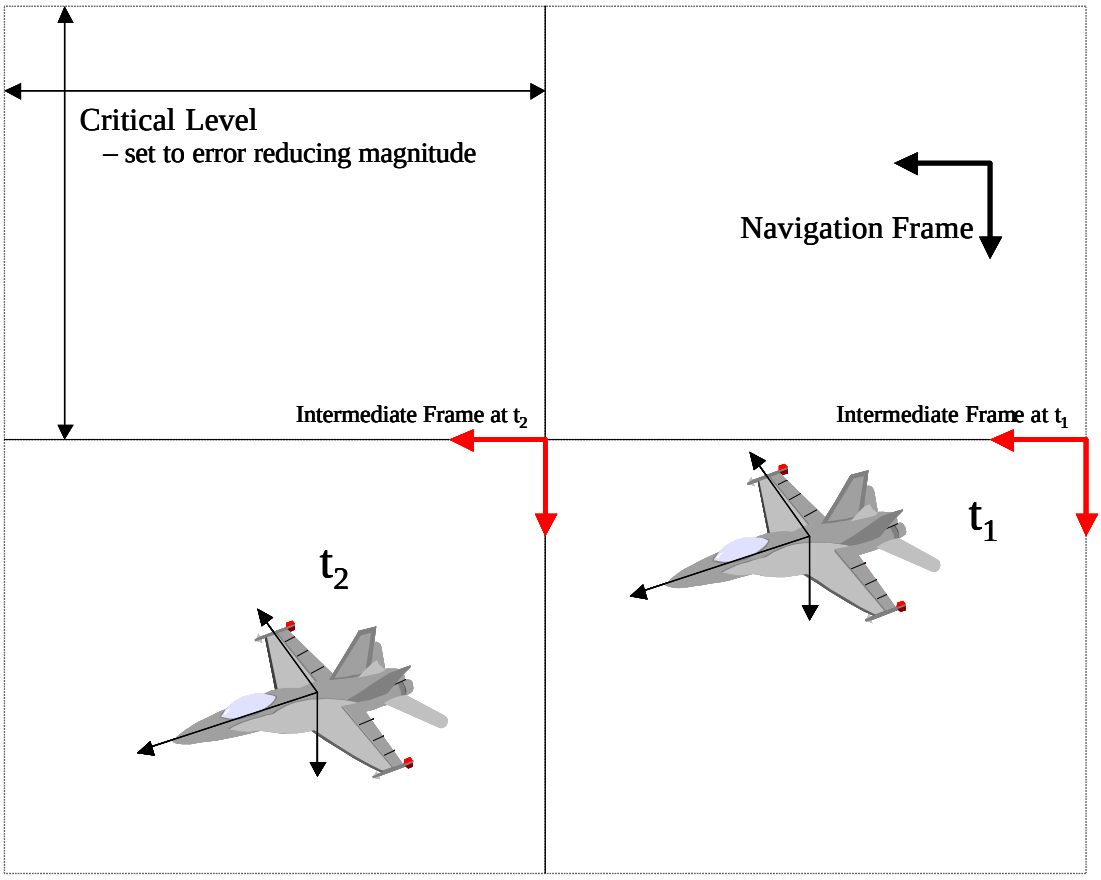


Figure 4.2: Critical Levels Regulate Updates of the Intermediate Frame

The use of critical levels to regulate the update of intermediate frames is illustrated in Figure 4.2. When a dynamic model is added to a simulation, an intermediate frame is introduced in the vicinity of the model. In the Figure 4.2, the model, represented by its body frame, is added to the simulation at t_1 and the intermediate frame is introduced in its vicinity. At time t_2, an element of the body frame's motion parameters with respect to the intermediate frame exceeds its critical level, prompting the intermediate frame to 'jump' by the critical level and update its position with respect to the navigation frame. The motion parameters of the body frame are maintained with respect to the navigation frame during the jump.

Critical levels are not constant; instead they can be selected to fit the numerical accuracy required in the simulation. The critical level of each element of a motion parameter determines the maximum local roundoff error allowed during integration. Thus, the upper limit of the critical level, $\underline{Cr}$, is a function of the maximum allowable roundoff error, $\Delta\underline{X}_{Rnd}$, per time step and can be estimated using machine accuracy, e_m:

$$\underline{Cr} < \frac{\Delta\underline{X}_{Rnd}}{e_m} \tag{4.3}$$

The lower limit of the critical level depends upon 2 distinct factors. The first is the roundoff error due to the motion parameters of the intermediate frame, $\left|{}^{IF}_{j}\underline{X}^{n} \times e_m\right|$, and represents the smallest value by which the intermediate frame can be updated without its update being lost to roundoff error. The other factor is the magnitude of the incremental term $\left|{}^{bf}_{j}\underline{\dot{X}}^{IF} \times \Delta t\right|$; if the incremental term is term is larger than the critical level, the intermediate frame will be updated at every time step. Since the error during the update reflects motion parameters in the vicinity of the body frame with respect to the navigation frame, the magnitude of the error per update is similar to the local roundoff error generated if intermediate frames are not used. Thus, updating the intermediate frame at every time step negates any benefit obtained through the use of the intermediate frames. The larger of these values is used for the lower limit:

$${}_{j}\underline{Cr} > \max\left(\left|{}^{bf}_{j}\underline{\dot{X}}^{IF}_{IF} \times \Delta t\right|, \left|{}^{IF}_{j}\underline{X}^{n}_{IF} \times e_m\right|\right) \tag{4.4}$$

4.2.2 Estimation of Roundoff Error Using Intermediate Frames

In a conventional simulation, there is a single source of roundoff error, which can be bounded as given by equations (2.43) and (2.44). The use of intermediate frames reduces the local roundoff error while introducing two additional sources of error. The first source is roundoff error per update of the intermediate frame upon reaching a critical level. The other is the roundoff error due to the propagation of the intermediate frame if it is moving with respect to the navigation frame. Thus, for the intermediate frame to be successful, the parameters that govern all these sources of error must be selected carefully.

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

$$\Delta \underline{X}_{Cr} = 2^{(\text{int})\log_2(\underline{Cr})-N+1} \approx \underline{Cr} \times e_m \tag{4.5}$$

The second source of error is incurred by updating the intermediate frames. Because the 'jump' consists of adding the critical value to the motion parameter of the intermediate frame, the roundoff error per update, $\Delta \underline{X}_U$, depends upon the magnitude of the motion parameters of the intermediate frame, ${}^{IF}\underline{X}^n$, with respect to the navigation frame:

$$\Delta \underline{X}_U = 2^{(\text{int})\log_2\left(\left|{}^{IF}\underline{X}^n\right|\right)-N+1} \approx \left|{}^{IF}\underline{X}^n_{IF}\right| \times e_m \tag{4.6}$$

The third source of error occurs if the intermediate frame is moving with respect to the navigation frame, requiring its motion parameters to be propagated and thus incurring roundoff errors. The accumulation of this error depends upon the specific implementation of the intermediate frame. If the intermediate frame only tracks the position and velocity of a vehicle's body frame, the intermediate frame's velocity only changes in discrete 'jumps' and is not directly affected by this error term. However, 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 algebraically as a function of time and constant velocity, roundoff error is induced only when its velocity is updated. This error term, $\Delta \underline{P}_{Pr}$, depends upon ${}^{IF}\underline{P}^n$, the position of the intermediate frame with respect to the navigation frame:

$$\Delta \underline{P}_{Pr} = 2^{(\text{int})\log_2\left(\left|{}^{IF}\underline{P}^n\right|\right)-N+1} \approx \left|{}^{IF}\underline{P}^n_{IF}\right| \times e_m \tag{4.7}$$

The maximum upper bound for global roundoff error, incurred during the entire simulation run, can be estimated by taking the sum of these error terms. However, their computation requires knowledge of the number of time steps and updates of the intermediate frame. Also, the vectors representing the first two error terms, $\Delta \underline{X}_{Cr}$ and $\Delta \underline{X}_U$, should be divided into error terms for position ($\Delta \underline{P}_{Cr}$ and $\Delta \underline{P}_U$) and velocity

($\Delta \underline{V}_{Cr}$ and $\Delta \underline{V}_{U}$) since the estimate for error in velocity only requires the first and second terms. In contrast, the error estimate for position is also affected by the third term at every update if the intermediate frame's position is expressed as a function of time. For $k_{\Delta t}$ time steps and ${}_{j}\underline{k}_{P}$ and ${}_{j}\underline{k}_{V}$ updates for the j^{th} elements of position and velocity of the intermediate frame respectively, the maximum upper bound of the roundoff error for the j^{th} element of position, ${}_{j}\Delta \underline{P}_{Rnd}$, and velocity, ${}_{j}\Delta \underline{V}_{Rnd}$, can be expressed as:

$$ {}_{j}\Delta \underline{P}_{Rnd} = {}_{j}\Delta \underline{P}_{Cr} \times k_{\Delta t} + {}_{j}\Delta \underline{P}_{U} \times {}_{j}\underline{k}_{P} + {}_{j}\Delta \underline{P}_{Pr} \times {}_{j}\underline{k}_{V} \tag{4.8} $$

$$ {}_{j}\Delta \underline{V}_{Rnd} = {}_{j}\Delta \underline{V}_{Cr} \times k_{\Delta t} + {}_{j}\Delta \underline{V}_{U} \times {}_{j}\underline{k}_{V} \tag{4.9} $$

Examining critical levels, their application in updating intermediate frame and the error estimates of global error, it can be noted that small critical levels will cause the intermediate frame to jump frequently, increasing the roundoff error due to updates, $\Delta \underline{X}_{U}$, and propagation, $\Delta \underline{P}_{Pr}$, but reducing the error per time step, $\Delta \underline{X}_{Cr}$. Conversely large critical levels will reduce the number of updates at the expense of increasing local roundoff error.

4.2.3 Selection of Critical Levels to Reduce Errors

The critical levels should be chosen so as to control the three error terms contributing to ${}_{j}\Delta \underline{P}_{Rnd}$ and ${}_{j}\Delta \underline{V}_{Rnd}$ in (4.8) and (4.9). The first, $\Delta \underline{X}_{Cr}$, corresponds to the magnitude of the critical levels as seen in equation (4.5). Reducing the critical level would reduce this error term.

To minimize the second, $\Delta \underline{X}_U$, a bit-wise analysis of the critical level is required. If the critical level, expressed in binary, is set to have a single non-zero bit in the mantissa with an exponent equal to or larger than the exponent of the LSB in the motion parameters of the intermediate frame, no bits are lost when updating the intermediate frame. If the motion parameters of the body frame are also measured in the intermediate frame, the error term $\Delta \underline{X}_U$ is eliminated. However, if the motion parameters are not measured in the intermediate frame, this error may not be eliminated since a rotation matrix will be applied to the critical level when updating the motion state. Thus, $\Delta \underline{P}_U$ can be eliminated since position of the body frame is measured in the intermediate frame while $\Delta \underline{V}_U$ may not be eliminated if the velocity is measured in the body frame. If this error term is present, reducing critical levels will increase the impact of this error term on global error.

The occurrence of the third term, $\Delta \underline{P}_{Pr}$, can be limited to the updates of the intermediate frame by expressing the position of the intermediate frame as an algebraic function of time and velocity. Thus, the effect of this error term on global error is proportional to the number of updates of the intermediate frame. Reducing critical levels increases the number of updates, thereby increasing the impact of $\Delta \underline{P}_{Pr}$ on global error.

The elimination of $\Delta \underline{P}_U$ allows (4.8) to be simplified such that ${}_j\Delta \underline{P}_{Rnd}$ consists of two error terms as expressed in (4.10). The first, $\Delta \underline{P}_{Cr}$, occurs at every time step while the second, $\Delta \underline{P}_{Pr}$, occurs at every update. By comparing (4.6) and (4.7), $\Delta \underline{P}_{Pr}$ can be treated as another form of $\Delta \underline{P}_U$ since both have the same magnitude and both occur when the intermediate frame is updated, allowing a common representation for ${}_j\Delta \underline{P}_{Rnd}$

and ${}_j\Delta\underline{V}_{Rnd}$, expressed by equation (4.11). It should be noted that ${}_j\underline{k}_U$ represents the update of the j^{th} element of velocity when evaluating the effect of error per update, $\Delta\underline{X}_U$, for both position and velocity.

$${}_j\Delta\underline{P}_{Rnd} = {}_j\Delta\underline{P}_{Cr} \times k_{\Delta t} + {}_j\Delta\underline{P}_{Pr} \times {}_j\underline{k}_V \tag{4.10}$$

$${}_j\Delta\underline{X}_{Rnd} = {}_j\Delta\underline{X}_{Cr} \times k_{\Delta t} + {}_j\Delta\underline{X}_U \times {}_j\underline{k}_U \tag{4.11}$$

Increasing or decreasing critical levels will reduce the impact of one error term while increasing the impact of the other on global error. The selection of critical levels that can minimize global roundoff error requires a trade off between the impact of local error and the error per update to global roundoff error. Furthermore, these terms need to be expressed as functions of the critical levels. While the impact of the local roundoff error term can be directly expressed as a function of critical level through equation (4.5), the impact of error per update on global error is through the number of updates, ${}_j\underline{k}_U$, rather than the error term expressed in (4.6). Thus, it is essential to express ${}_j\underline{k}_U$ as a function of critical levels. While it is not possible to directly express the number of updates as a function of critical levels, an approximate expression may be developed using the distance, ***S***, traversed by the j^{th} element of a motion parameters as expressed by equation (4.12). The distance, ***S***, is the sum of the absolute values of the incremental terms for the element over the entire simulation and provides a measure of the number of updates required for the intermediate frame to traverse the domain.

$$ {}_j\underline{k}_U = \frac{S}{{}_j\underline{Cr}} \tag{4.12} $$

$$ S = \sum_{i=0}^{k_{\Delta t}} \left| {}^{bf}_{j}\underline{\dot{X}}^{IF}_{IF} \times \Delta t \right|_i \tag{4.13} $$

Equation (4.11) can then be expressed as a function of critical levels. However, the error term for position is a function of critical levels for both position, ${}_j\underline{Cr}_P$, and velocity, ${}_j\underline{Cr}_V$, while the error term for velocity only depends on ${}_j\underline{Cr}_V$ as shown in equations (4.14) and (4.15).

$$ {}_j\Delta\underline{P}_{\text{Rnd}} \approx {}_j\underline{Cr}_P \times k_{\Delta t} \times \boldsymbol{e}_m + {}_j\Delta\underline{P}_U \times \frac{S}{{}_j\underline{Cr}_V} \tag{4.14} $$

$$ {}_j\Delta\underline{V}_{\text{Rnd}} \approx {}_j\underline{Cr}_V \times k_{\Delta t} \times \boldsymbol{e}_m + {}_j\Delta\underline{V}_U \times \frac{S}{{}_j\underline{Cr}_V} \tag{4.15} $$

Inspection of equations (4.14) and (4.15) shows that the critical level for position only affects the term for local roundoff error for position. Since the reduction of critical levels reduces the local roundoff error, the critical level for position can be set to its minimum limit. In contrast, the critical level for velocity affects the global roundoff error for both position and velocity. Therefore, an expression for global error of both position and velocity, Δ, as shown in equation (4.16) needs to be minimized to select suitable critical levels for velocity. The errors in this expression are scaled using the maximum magnitudes of the intermediate frame's motion states during the course of the simulation.

$$ {}_j\Delta = \frac{{}_j\underline{P}_{Rnd}}{{}^{IF}_{j}\underline{P}^{n}_{\max}} + \frac{{}_j\underline{V}_{Rnd}}{{}^{IF}_{j}\underline{V}^{n}_{\max}} \tag{4.16} $$

Equation (4.11) can then be differentiated with respect to ${}_j\underline{Cr}_V$ as depicted in equation (4.17) and the value that minimizes the global error can be computed by (4.18).

$$ \frac{d\left({}_j\Delta\right)}{d\left({}_j\underline{Cr}_V\right)} \approx \frac{e_m \times k_{\Delta t} - {}_j\Delta\underline{V}_U \times \frac{S}{{}_j\underline{Cr}_V^{\,2}}}{{}^{IF}_{j}\underline{V}^{n}_{\max}} - \frac{{}_j\Delta\underline{P}_U \times \frac{S}{{}_j\underline{Cr}_V^{\,2}}}{{}^{IF}_{j}\underline{P}^{n}_{\max}} \tag{4.17} $$

$$ {}_j\underline{Cr}_V \approx \frac{\frac{{}^{IF}_{j}\underline{V}^{n}_{\max}}{{}^{IF}_{j}\underline{P}^{n}_{\max}} \times {}_j\Delta\underline{P}_U \times {}_j\underline{k}_U + {}_j\Delta\underline{V}_U \times {}_j\underline{k}_U}{e_m \times k_{\Delta t}} \tag{4.18} $$

Equation (4.18) provides an estimate of the critical level that minimizes global roundoff error for the simulation run. However, it requires the number of updates and the number of time steps to be known. Furthermore, the terms ${}_j\Delta\underline{P}_U \times {}_j\underline{k}_U$ and ${}_j\Delta\underline{V}_U \times {}_j\underline{k}_U$ in the numerator of (4.18) represents the impact of errors from updating reference frames, treating the error from each update as a constant value that can be multiplied by the number of updates. However, (4.6) shows that this error term is proportional to the magnitude of the intermediate frame's motion parameters, which vary with time. Since the impact of the error per update can also be treated as the sum of errors per update, as expressed in (4.19), the critical level can be estimated as shown in equation (4.20):

$$ {}_j\Delta\underline{X}_U \times {}_j\underline{k}_U \approx \sum_{i=0}^{{}_j\underline{k}_U} \left({}_j\Delta\underline{X}_U \right)_i \tag{4.19} $$

$$ {}_j\underline{Cr}_V \approx \frac{\frac{{}_j^{IF}\underline{V}_{\max}^{n}}{{}_j^{IF}\underline{P}_{\max}^{n}} \times \sum_{i=0}^{{}_j\underline{k}_U} \left({}_j\Delta\underline{P}_U \right)_i + \sum_{i=0}^{{}_j\underline{k}_U} \left({}_j\Delta\underline{V}_U \right)_i}{e_m \times k_{\Delta t}} \tag{4.20} $$

Equation (4.20) allows the critical level to be calculated and updated adaptively during the course of the simulation using the number of time steps executed and the sum of the error incurred by updating the intermediate frame at any given stage of the simulation. The numerator only changes at every update of the intermediate frame while the denominator changes at every time step. Updating the critical levels at every time step may require additional computation, but ensures that the critical levels are appropriate for the motion states of the dynamics of the model.

4.3 Implementation of Intermediate Frames

Intermediate frames were implemented using the Reference Frame Manager (RFM) described in Chapter 3. The RFM allows intermediate frames to be added to its network of reference frames and can be used by dynamic models in the Reconfigurable Flight Simulator (RFS). An interface class was added to the reference frame management interface library, allowing simulation components to use intermediate frames through standard interfaces that are independent of their implementation. The implementation of the Reference Frame Manager was updated to include an Intermediate Frame class as well as an Intermediate Frame Manager class, responsible for creating and initializing intermediate frames. The descriptions of these classes as well as the modifications

required in the RFM are described in this section. The remaining implementation details are provided in Appendix B.

4.3.1 Class Definitions for Intermediate Frames

The object oriented programming paradigm was used to develop an interface class for intermediate frames. This was implemented into an Intermediate Frame Class within the Reference Frame Manager. An Intermediate Frame Manager class was also implemented in the RFM to create, initialize and destroy intermediate frames as required.

1. The ***Intermediate Frame Interface Class*** is used to determine the interface standards for intermediate frames. This interface class inherits from the Frame Definition Class described in Chapter 3, providing the standard interfaces and functionality of reference frames to the intermediate frame. The additional interfaces in this class deals with the initialization and update of the intermediate frame. Both interfaces use the identity of the body frame to access its motion parameters. The initialization interface is used to assign the body frame whose motion parameters are tracked by the intermediate frame. Since the intermediate frame does not accelerate with respect to the navigation frame, the pure virtual functions of the Frame Definition Class requiring the calculation of accelerations are implemented and return zero acceleration. Furthermore, the standard update method of using numerical integration is disabled, improving computational efficiency and restricting the introduction of roundoff error to the error per update terms described by equations (4.6) and (4.7).
2. The ***Intermediate Frame Class*** implements the Intermediate Frame Interface Class within the RFM. The standard interfaces of the interface class are implemented to initialize its motion parameters using the motion parameters of the assigned body

frame and to compare the motion parameters of the body frame with the critical levels, updating both frames if necessary. In addition to these standard interfaces, the Intermediate Frame Class also calculates and updates the critical level, adapting to the dynamics of the body frame. The critical levels are evaluated and updated at every time step using the adaptive algorithm described in Section 4.2.3.

3. The ***Intermediate Frame Manager*** is a component created and added to the RFM that is responsible for creating and initializing Intermediate Frames when requested by the RFM. The Intermediate Frame Manager creates Intermediate Frames when commanded by the RFM and maintains them in an internal list. After initializing the Intermediate Frame, the Intermediate Frame Manager registers the Intermediate Frame with the RFM. When the Intermediate Frame needs to be destroyed, either when its assigned body frame is destroyed or when the simulation terminates, the Intermediate Frame Manager is responsible for destroying the object and de-allocating the memory.

The Intermediate Frame and Intermediate Frame Manager classes are instantiated inside the RFM. Once created, the Intermediate Frames are treated as Reference Frame Objects and added to the network maintained by the RFM, as depicted in Figure 4.3.

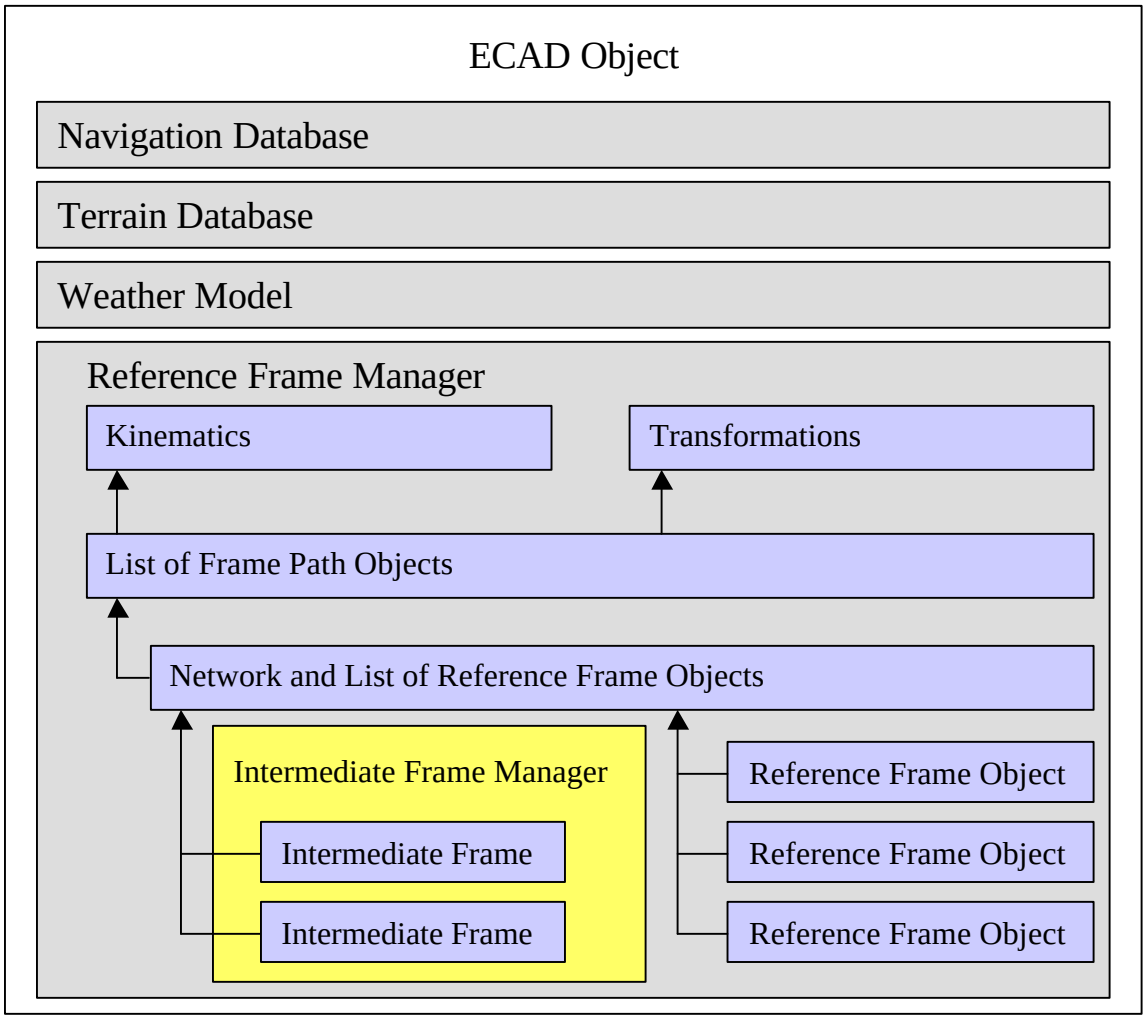


Figure 4.3: Intermediate Frames Within the RFM

4.3.2 Network Operations for Intermediate Frames

While the introduction of intermediate frames to the network may seem to require the addition of new network operations, these operations can be executed as a series of standard network operations. In particular, the introduction of the intermediate frame and its update of the body frame can be treated as combinations of several operations.

4.3.2.1 Linking Intermediate Frames with Navigation Frames and Body Frames

When an intermediate frame is created and added to the network, it acts as a surrogate to the navigation frame. Thus, it defines its motion parameters with respect to the navigation frame and becomes the definition frame of the body frame. This can be viewed as a sequence of pruning, adding and grafting operations. Once the intermediate frame is initialized, it is added as a leaf node to the navigation frame. The definition frame used by the body frame is then changed from the navigation frame to the intermediate frame. Thus, the node representing the body frame is pruned from the network and grafted to the node representing the intermediate frame, accompanied by the appropriate kinematics.

4.3.2.2 Updating Intermediate Frames and Body Frames

When an intermediate frame is updated, its motion parameters execute a discrete jump with respect to the navigation frame. Since this jump is used to limit the motion parameters of the body frame and does not reflect any physical motion of the body frame, the motion parameters of the body frame do not change with respect to the navigation frame during this update. While the intermediate frame is updated by addition or subtraction of the critical level, updating the motion parameters of the body frame requires the application of kinematics since the measurement frame of its motion parameters may not be the intermediate frame. In particular, the velocity may require a rotation based on the orientation of the body frame with respect to the intermediate frame. Instead of developing a new set of methods to handle this transformation, the standard interfaces of the Frame Definition Class and Reference Frame Manager can be used.

At the time of the update, two reference frames represent the intermediate frame before and after the update. When motion parameters of the body frame are subsequently updated, its definition frame is effectively changed from the intermediate frame before the update to the intermediate frame after the update. While this operation can be executed in the RFM by introducing a new reference frame, the kinematics can be carried out in the body frame by using the Frame Definition's interface. If ***IF1*** represents the intermediate frame before the update and ***IF2*** after the update, the motion parameters of the body frame can be expressed as follows:

$$\left[{}^{bf}C^{IF2}\right] = \left[{}^{bf}C^{IF1}\right]\left[{}^{IF1}C^{IF2}\right] \tag{4.21}$$

$${}^{bf}\underline{w}_{bf}^{IF2} = {}^{bf}\underline{w}_{bf}^{IF1} + \left[{}^{bf}C^{IF1}\right]\,{}^{IF1}\underline{w}_{IF1}^{IF2} \tag{4.22}$$

$${}^{bf}\underline{P}_{IF2}^{IF2} = \left[{}^{IF1}C^{IF2}\right]^{T}\,{}^{bf}\underline{P}_{IF1}^{IF1} + {}^{IF1}\underline{P}_{IF2}^{IF2} \tag{4.23}$$

$$\begin{aligned}{}^{bf}\underline{V}_{bf}^{IF2} = {}^{bf}\underline{V}_{bf}^{IF1} &+ \left[{}^{bf}C^{IF1}\right]\left({}^{IF1}\underline{V}_{IF1}^{IF2} + {}^{IF1}\underline{w}_{IF1}^{n} \times {}^{bf}\underline{P}_{IF1}^{IF1}\right) \\ &- \left[{}^{bf}C^{IF2}\right]\left({}^{IF2}\underline{w}_{IF2}^{n} \times {}^{bf}\underline{P}_{IF2}^{IF2}\right)\end{aligned} \tag{4.24}$$

From the equations above, it can be seen that the body frame can modify its motion parameters if it is given the motion parameters of IF1 with respect to IF2 and the angular velocity of IF2 with respect to the navigation frame. If the intermediate frame maintains the same orientation as the navigation frame and does not rotate, (4.21) and (4.22) can be ignored and (4.23) and (4.24) simplify to:

$$ {}^{bf}\underline{P}_{IF2}^{IF2} = {}^{bf}\underline{P}_{IF1}^{IF1} + {}^{IF1}\underline{P}_{IF2}^{IF2} \tag{4.25} $$

$$ {}^{bf}\underline{V}_{bf}^{IF2} = {}^{bf}\underline{V}_{bf}^{IF1} + \left[{}^{bf}C^{IF1}\right] {}^{IF1}\underline{V}_{IF1}^{IF2} \tag{4.26} $$

4.3.3 Standard Operations for Intermediate Frames within the RFM

The introduction and maintenance of intermediate frames within the RFM require several standard operations to be carried out. These include the creation and initialization of Intermediate Frame objects, updating the motion parameters of the intermediate frame objects and body frames, and adaptively calculating the critical levels. Some of these operations require the standard interface of the reference frame manager to be updated, allowing dynamic models to request intermediate frames from the RFM.

4.3.3.1 Creation and Initialization of Intermediate Frames

An intermediate frame is created when the RFM receives a request for one from a dynamic model. The RFM passes the pointer to the dynamic model's body frame to the Intermediate Frame Manager. The Intermediate Frame Manager creates an Intermediate Frame object and initializes it using the motion parameters of the body frame. The motion parameters of the Intermediate Frame are defined with respect to the body frame's definition frame, typically the navigation frame. The Intermediate Frame Manager returns the Intermediate Frame's pointer to the RFM, which subsequently registers the Intermediate Frame with the network, linking it to the nodes representing the navigation and body frames as described in Section 4.3.2.1.

The Intermediate Frame only tracks the position and velocity of the body frame. Thus, the Intermediate Frame matches the orientation of the navigation frame and does

not rotate with respect to it, requiring fewer modifications to the numerical integration routines within the dynamic model as described in Section 4.1.2.

4.3.3.2 Updating Intermediate Frames

At the end of every time step, the dynamic model requests the RFM to update the Intermediate Frame. Since critical levels are maintained in the Intermediate Frame objects, the dynamic model is unable to check when its motion parameters have exceeded their critical levels. Instead, the Intermediate Frame checks the motion parameters when it receives the update command from the RFM. If any critical levels are exceeded, the Intermediate Frame and the dynamic model's body frame are updated as described in Section 4.3.2.2.

4.3.3.3 Adaptively Selecting Critical Levels

The critical levels of the intermediate frame are selected to minimize the global roundoff error incurred as the simulation progresses. They can be initialized to error reducing values between their upper and lower bounds as expressed in (4.3) and (4.4). Error reducing values are integer powers of 2, represented by a single bit within the mantissa, as discussed in Section 4.2.3. As the simulation progresses, the critical levels are updated at every time step using equation (4.20). To ensure that these critical levels still represent error reducing values, the j^{th} element of $\underline{Cr}$ can be represented as a function of the j^{th} element of an integer vector $\underline{M}$ as depicted in equation (4.27). Consequently, the equations that adaptively calculate $\underline{Cr}$ and its upper and lower bounds can be modified to calculate ${}_j\underline{M}$ and its bounds as depicted in equations (4.28), (4.29)

and (4.30). The critical levels for position are calculated using the lower bound, expressed by (4.29) while the critical level for velocity can use the adaptive algorithm.

$$ {}_{j}\underline{Cr} = 2^{{}_{j}\underline{M}} \tag{4.27} $$

$$ {}_{j}\underline{M} < (\text{int})\log_2\left(\frac{\Delta \underline{X}_{Rnd}}{e_m}\right) \tag{4.28} $$

$$ {}_{j}\underline{M} > \max\left((\text{int})\log_2\left(\left|{}_{j}^{bf}\underline{\dot{X}}_{IF}^{IF} \times \Delta t\right|\right), (\text{int})\log_2\left(\left|{}_{j}^{IF}\underline{X}_{IF}^{n} \times e_m\right|\right)\right) \tag{4.29} $$

$$ {}_{j}\underline{M} \approx (\text{int})\log_2\left(\frac{\frac{{}_{j}^{IF}\underline{V}_{\max}^{n}}{{}_{j}^{IF}\underline{P}_{\max}^{n}} \times \sum_{i=0}^{{}_{j}\underline{k}_U}\left({}_{j}\Delta\underline{P}_U\right)_i + \sum_{i=0}^{{}_{j}\underline{k}_U}\left({}_{j}\Delta\underline{V}_U\right)_i}{e_m \times k_{\Delta t}}\right) \tag{4.30} $$

CHAPTER 5

DEVELOPMENT OF A GENERIC DYNAMIC MODEL

The ability to model reference frames as unique entities in a simulation environment encourages the development of dynamic models that are able to select their navigation and inertial frames as required. Furthermore, the ability to express motion parameters of a reference frame with respect to any other reference frame in the simulation environment can be used by dynamic models to automate many aspects of their kinetics and kinematics. Dynamic models that are able to select their navigation and inertial frames, automatically calculating the kinetics and kinematics corresponding to these reference frames, can be rapidly reconfigured to models of other vehicles, or higher fidelity models of the same vehicle, using different reference frames or requiring different kinematics fidelity.

The software implementation of kinematics, numerical integration, and reference frame transformations is, in theory, independent of the kinetics and subsystem dynamics within a model and can be reused by other models. However, these components are often re-implemented with new dynamic models, leading to additional development time and cost [27] when creating and modifying simulations. A better paradigm can allow the common elements, i.e; numerical integration, kinematics and transformation between reference frames, to be encapsulated within a common framework. The unique properties of the dynamic model, contained in the kinetics and subsystems, can be encapsulated in different model components and assembled by the framework to form the dynamic model. This chapter describes the development of a Generic Dynamic Model (GDM),

which encapsulates the common elements of 6DOF dynamic models and the identification of interfaces for representing dynamics unique to specific vehicles.

5.1 Conceptual Development of Generic Dynamic Models

Based on the structure of dynamic models described in the section 2.2.3, it is possible to identify the elements that are common to all dynamic models. These common elements can be used to form the framework of a generic dynamic model.

- The numerical integration routine is inherently independent of the dynamics as it deals with any arbitrary state vector and its corresponding derivative vector. Although the integration routine requires the derivative vector to be calculated during each stage of the routine, it is not dependent upon the implementation of the dynamics that calculate the derivative vector.
- The kinematics of the dynamic model depends only upon the motion parameters and accelerations of the reference frames used by the model. Thus, the kinematics equations can be generalized to all dynamic models.
- While the kinetics equations require several elements that are unique to specific vehicles (e.g. forces, moments, mass and inertia tensor), the equations relating these unique elements to acceleration and angular acceleration can be generalized to all dynamic models.
- The reference frame manager can also provide the kinematics and rotations between any pair of reference frames in the simulation environment.

Thus, using a reference frame manager in the simulation environment enables kinetics, kinematics and rotations to be generalized to all dynamic models. The remaining

elements in dynamic models can be considered unique to specific vehicles. These elements model the different subsystems as well as the calculation of forces and moments. The dynamic model can then be represented as a combination of a generic dynamic model and unique elements as depicted in Figure 5.1, as described in the following subsections.

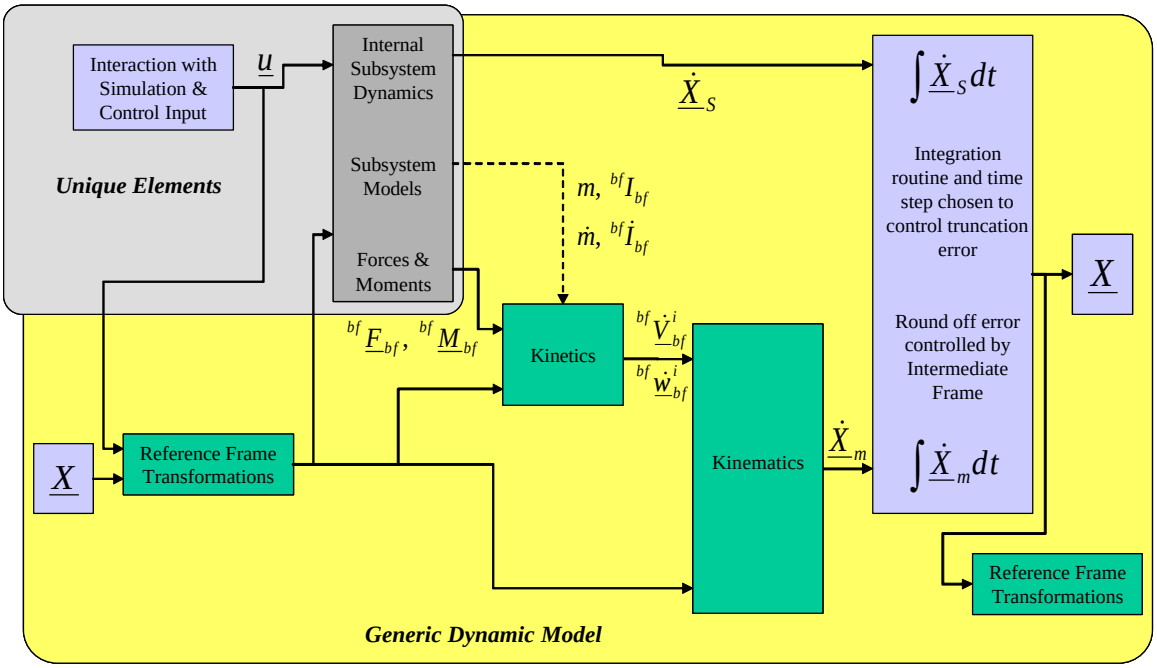


Figure 5.1: The Dynamic Model as a Combination of Unique and Generic Elements

5.1.1 Dynamic Model Elements Unique to a Vehicle

The unique elements of the dynamic model provide the parameters and equations specific to the vehicle. Each unique element only updates the state and derivative vectors propagated by the element's internal subsystem dynamics. This restriction is required to prevent multiple updates of a state that may be used by another subsystem or by the kinetics and kinematics equations. Since other elements may require the states of a

unique element in their control or dynamics calculations, each element needs to provide the identity of its states to the generic model. If an element requires the state from another element for its derivative function, it can obtain the value of the required state via the generic model. While an element may use the state of another element, it should not propagate that state. This interface standard addresses the issue of coupled dynamics within subsystems.

In addition to the internal subsystem dynamics, the unique elements may also provide parameters that provide the inputs to the kinetics equations. Foremost among these are the forces and moments on the vehicle. Each element should specify the definition and measurement frames associated with its internal body frame. Consequently, each element will need access to the reference frame manager as it may require the motion states to be expressed with respect to different definition and measurement frames. Additional parameters include the mass and inertia tensor and their rates of change. The inertial frame used in evaluating the kinetics and kinematics should also be specified at every update, as its identity may change between and during the course of simulation runs.

5.1.2 Generic Dynamic Model

The generic dynamic model collects the data from all the unique elements that are added to the model and forms the final dynamic model at runtime. While the state and derivative vectors for modeling subsystem dynamics are maintained in the unique elements, the generic model maintains the vectors for the motion states and their derivatives. The motion states correspond to the motion parameters of the body frame of the vehicle with respect to a designated navigation frame. The identity of the navigation

frame is also stored in the generic model. The overall state and derivative vectors are assembled by the generic dynamic model and combine the motion states and their derivatives with the states and derivatives of all the unique elements used by the dynamic model. These overall state and derivative vectors will be used in numerical integration.

While the derivatives of the subsystem states are calculated by the unique elements, the generic dynamic model calculates the derivatives of the motion states through the kinetics and kinematics equations. The generic model obtains and assembles the forces, moments, mass and inertia properties contributed by each unique element. These parameters are applied in the kinetics equations to calculate the acceleration of the body frame with respect to the inertial frame as expressed in (5.1) and (5.2). As in equations (2.32) and (2,33), these kinetics equations assume that the mass and inertia tensor are invariant over time. Additional terms can be introduced in these equations if the mass or inertia tensor has a time rate of change sufficiently significant to impact the modeled dynamics.

$$ {}^{bf}\underline{F}_{bf} = m\left({}^{bf}\underline{\dot{V}}^{i}_{bf} + {}^{bf}\underline{w}^{i}_{bf} \times {}^{bf}\underline{V}^{i}_{bf}\right) \tag{5.1} $$

$$ {}^{bf}\underline{M}_{bf} = {}^{bf}\underline{w}^{i}_{bf} \times \left(\left[{}^{bf}I_{bf}\right] {}^{bf}\underline{w}^{i}_{bf}\right) + \left[{}^{bf}I_{bf}\right] {}^{bf}\underline{\dot{w}}^{i}_{bf} \tag{5.2} $$

The acceleration and angular acceleration obtained by (5.1) and (5.2) are the time derivatives of the velocity and angular velocity, ${}^{bf}\underline{\dot{V}}^{i}_{bf}$ and ${}^{bf}\underline{\dot{w}}^{i}_{bf}$, measured in the body frame. This representation is required for angular acceleration since the moment equation (5.2) differentiates angular momentum and the derivative of the inertia tensor depends

upon the measurement frame used. However, an alternate representation of acceleration, depicted by equation (5.3), allows the standard kinematics representation of acceleration between reference frames, as expressed by (3.15), to be used in calculating the derivative of velocity as depicted in equation (5.4). In addition to being a more recognizable representation, it is also more computationally efficient as it requires fewer matrix operations.

$$ {}^{bf}\underline{F}_{bf} = m\frac{{}^{i}d}{dt}\left({}^{bf}\underline{V}^{i}_{bf}\right) \tag{5.3} $$

$$ \begin{aligned} {}^{bf}\dot{\underline{V}}^{n}_{bf} = {} & \frac{{}^{i}d}{dt}\left({}^{bf}\underline{V}^{i}_{bf}\right) - \left({}^{bf}\underline{w}^{n}_{bf} + 2\left(\left[{}^{bf}C^{n}\right]{}^{n}\underline{w}^{i}_{n}\right)\right)\times{}^{bf}\underline{V}^{n}_{bf} \\ & - \left[{}^{bf}C^{n}\right]\left(\frac{{}^{i}d}{dt}\left({}^{n}\underline{w}^{i}_{n}\right)\times{}^{bf}\underline{P}^{n}_{n} + {}^{n}\underline{w}^{i}_{n}\times\left({}^{n}\underline{w}^{i}_{n}\times{}^{bf}\underline{P}^{n}_{n}\right) + \frac{{}^{i}d}{dt}\left({}^{n}\underline{V}^{i}_{n}\right)\right) \end{aligned} \tag{5.4} $$

The kinematics equations of the body frame are then used to generate the derivatives of the motion states of the model, using the motion parameters of the body frame and the acceleration obtained from the kinetics equations. The time derivatives for position and orientation are based entirely upon the motion parameters of the body frame, as depicted in (5.5) and (5.6). The motion parameters of the navigation frame with respect to the inertial frame are obtained from the reference frame management mechanism. These motion parameters are used along with the acceleration calculated by the kinetics equations to calculate the derivatives for velocity and angular velocity of the body frame as depicted in (5.7) and (5.8).

$$\left[{}^{bf}\dot{C}^{n}\right] = f\left(\left[{}^{bf}C^{n}\right], {}^{bf}\underline{w}_{bf}^{n}\right) \tag{5.5}$$

$${}^{bf}\underline{\dot{P}}_{n}^{n} = \left[{}^{bf}C^{n}\right]^{T}\,{}^{bf}\underline{V}_{bf}^{n} \tag{5.6}$$

$${}^{bf}\underline{\dot{w}}_{bf}^{n} = {}^{bf}\underline{\dot{w}}_{bf}^{i} - \left(\left[{}^{bf}C^{n}\right]\,{}^{n}\underline{w}_{n}^{i}\right) \times {}^{bf}\underline{w}_{bf}^{n} - \left[{}^{bf}C^{n}\right]\,\frac{{}^{i}d}{dt}\left({}^{n}\underline{w}_{n}^{i}\right) \tag{5.7}$$

$$\begin{aligned} {}^{bf}\underline{\dot{V}}_{bf}^{n} = {}^{bf}\underline{\dot{V}}_{bf}^{i} + {}^{bf}\underline{w}_{bf}^{i} \times \left(\left[{}^{bf}C^{n}\right]\,{}^{n}\underline{V}_{n}^{i}\right) + {}^{bf}\underline{w}_{bf}^{n} \times \left(\left[{}^{bf}C^{n}\right]\left({}^{n}\underline{w}_{n}^{i} \times {}^{bc}\underline{P}_{n}^{n}\right)\right) \\ - \left(\left[{}^{bf}C^{n}\right]\,{}^{n}\underline{w}_{n}^{i}\right) \times {}^{bf}\underline{V}_{bf}^{n} - \left[{}^{bf}C^{n}\right]\left(\frac{{}^{i}d}{dt}\left({}^{n}\underline{w}_{n}^{i}\right) \times {}^{bc}\underline{P}_{n}^{n} + \frac{{}^{i}d}{dt}\left({}^{n}\underline{V}_{n}^{i}\right)\right) \end{aligned} \tag{5.8}$$

Other responsibilities of the generic dynamic model include transformation of motion parameters between reference frames, and numerical integration. The reference frame manager's ability to transform motion parameters to any reference frame in its network can be utilized by the generic dynamic model and the unique elements in the model. The generic dynamic model also provides the numerical integration routines that can be used to propagate the assembled state vector through time, using the assembled derivative vectors. While the numerical integration routine should be able to request updates to the derivative vector, it should be independent of the implementation of the unique elements that provide the derivatives or parameters that lead to the calculation of derivatives. The integration routine in the generic dynamic model should also be able to estimate the time step it would require to limit truncation error to an arbitrary value.

While the truncation error can be controlled through the selection of an appropriate time step, control of roundoff error can be achieved through the use of intermediate frames, enabling the generic model to control both truncation and numerical

error. Thus, the generic model should be able to request and initialize an intermediate frame allowing roundoff error to be controlled as described in Chapter 4.

Since the definition frame may be changed from the navigation frame to the intermediate frame, it is important for unique elements to specify their preferred definition and measurement frames for expressing motion parameters. Not only does this ensure that the motion parameters are applied correctly in each unique element, but it also enables different unique elements to express the motion parameters in reference frames that are convenient for calculating the forces and moments contributed by each element rather than the navigation frame used by the dynamic model.

5.2 Implementation of the Generic Dynamic Model

The unique elements and generic dynamic model are implemented as Unique Dynamics Components (UDC) and a Generic Dynamics Component (GDC) within the Reconfigurable Flight Simulator (RFS). The simulation environment provided by RFS also provides the Reference Frame Manager (RFM) described in Chapter 3 to all major simulation components.

Interface classes encapsulating the standard interfaces and properties of the generic model and unique elements were developed as described in the previous section. The Generic Dynamics Interface Class encapsulates the standard interfaces and properties required by the GDC while the Model Component Interface Class represents the interface standard for the Unique Dynamics Components. These interface classes were added to the reference frame management interface library, allowing these components to interact with each other through standard interfaces. The GDC implements the standard interfaces and functionality of the Generic Dynamics Interface Class while each UDC represents the

implementation of the Model Component Interface Class for specific vehicles and subsystems. A brief description of these classes and their implementation is described in this section. Full implementation details are provided in Appendix B.

5.2.1 Class Definitions for Implementing a Generic Dynamic Model

1. The ***Generic Dynamics Interface Class*** encapsulates the standard interfaces and functionality required for the successful operation of generic dynamic models. These interfaces include methods to add and remove components modeling unique elements, set the identity of the navigation and inertial frames, select the appropriate numerical integration method, and request the use of an intermediate frame to reduce roundoff error. The functionality required by the generic model described in Section 5.1.2 is implemented. This includes the assembly and initialization of the state and derivative vectors, generalized implementation of the kinetics and kinematics equations, two types of integration routines, and the introduction and initialization of the body frame and intermediate frame. The integration routines are the standard 4^{th} order Runge Kutta routine and the adaptive 4^{th} order Runge Kutta Cash Karp method with a 5^{th} order error term to calculate the time step required to control truncation error. Since the time step is controlled by the simulation architecture, the integration routine only adjusts the next time step rather than repeating the current step should the truncation error exceeds the specified limit. These integration routines can be overridden should the dynamic model require a different method.
2. The ***Model Component Interface Class*** encapsulates the interface standards for representing unique elements as described in Section 5.1.1. These interfaces include methods to get pointers to the state and derivative vectors of the subsystems, to

vectors representing the forces and moments exerted on the body by the element, and to the mass and inertia properties that the element contributes to the dynamic model. Pointers to these vectors and matrices are used since they can vary over time and it is computationally more efficient to access their addresses rather than execute function calls for each parameter. A method that updates these parameters is also included in the standard interface as are methods to provide access to the body frame, the RFM and the Generic Dynamic Component. The standard interfaces dealing with unique properties of the model are defined as *pure virtual functions* that must be implemented by the Unique Dynamics Component since these parameters are specific to each component. In contrast, the methods linking the unique component to the body frame, RFM and Generic Dynamic Component interact with standardized components in the simulation and are implemented in the interface class.

3. The ***Generic Dynamic Component (GDC) Class*** implements the Generic Dynamics Interface Class into an aircraft class that can be used in RFS. The GDC implements the standard interface requirements of the *Base Airplane Object* class in RFS. By representing the model as an aircraft class, the GDC is able to access the standard interfaces and attributes of aircraft in RFS. Not only does this enable the GDM to utilize the timing mechanisms and interactions provided by RFS, but the standard attributes of aircraft can be used to represent the shared states that may be required by different components for their calculation.
4. The ***Unique Dynamics Components (UDC)*** implement the Model Component Interface Class for specific subsystems and vehicles. Each UDC has to implement all the *pure virtual functions* of the Model Component Interface Class, ensuring that

these methods are implemented in the UDC and can be accessed by the GDC when it is assembling the dynamic model. If the UDC does not model certain parameters returned by these methods, it returns a NULL pointer, enabling the GDM to assemble the appropriate parameters. Additional interfaces can be added to the UDC, enabling it to be initialized and configured for different subsystems and scenarios.

5.2.2 Standard Operations in Assembling Generic Dynamic Models

Since the Generic Dynamics Component assembles the dynamics of the model using the properties generated by multiple Unique Dynamics Components, the standard operations required for generic dynamic models to function properly in simulation need to be discussed. The operations include the assembly of the dynamic model and the propagation of its states using numerical integration.

5.2.2.1 Assembling Unique Dynamics Components into a Generic Dynamic Model

The Generic Dynamics Component needs to assemble the state and derivative vectors for the dynamic model whenever Unique Dynamics Components are added to or removed from the model. While this typically occurs during initialization, UDCs may be added or removed during the course of the simulation. When UDCs are added or registered with a GDC, their pointers are stored by the GDC in a ‘component list’. In turn, each registered UDC is given read-only access to the GDC, the body frame and the simulation environment, including the RFM.

During the registration process, the GDC obtains vectors containing the addresses of the states and derivatives in the UDC. This allows the UDC to use standard data members of the GDC class for its states, effectively broadcasting its states to other UDCs. In addition to the state and derivative vectors, the GDC also obtains vectors containing

the addresses of forces, moments and inertia properties that are calculated by each UDC. UDCs return NULL pointers for any property not calculated by them. These pointers are stored in the GDC and used to assemble the forces and moments exerted on the dynamic model, and its total mass and inertia properties.

In this implementation, the GDC creates and registers the body frame with the RFM during initialization. An intermediate frame can be requested from the RFM, either during initialization or at any point in the simulation, if the model requires control of roundoff error. The motion parameters of the body frame are updated by the RFM. The navigation and inertial frames used by the body frame for expressing its motion parameters or for evaluating acceleration can also be set during initialization or changed during the course of the simulation. If they are changed during the simulation, the RFM carries out the appropriate transformation between the old and new navigation frames. Changing the inertial frame only affects the evaluation of acceleration and does not require the motion parameters to be expressed with respect to a different frame. While it is possible for any UDC to change the inertial frame of the GDC through its standard interface, care must be taken to ensure that only one UDC is authorized to change the inertial frame. This constraint is required to ensure that different UDCs do not try to set different inertial frames, as the identity of the inertial frame will be overridden by the last UDC to command the change. To enforce this constraint, UDCs that set the inertial frame are designated as ‘body components’ and the GDC is restricted to registering only one body component UDC.

5.2.2.2 Propagating States Using a Generic Dynamic Model

Numerical integration propagates the model's states through time. In the traditional implementation of dynamic models and integration routines, temporary arrays of vectors are used to record the states and derivatives that are progressively calculated at each stage in a multi-stage integration routine. While the number of derivative vectors corresponds to the number of stages in the routine, only a single copy of the state vector is required to represent the state at each stage. The original state vector is not modified during the individual stages but updated once the final incremental term, using the derivative vector from all stages, is assembled. This is possible because functions that calculate the derivative vectors use a well-defined state vector, allowing the functions to access the appropriate data.

However, the assembled state vector in a generic dynamic model can be of arbitrary length and the location of specific states within the vector depends upon the order in which the vector was assembled. Thus, it is not possible for the methods in the UDC to interpret the assembled state vector. If methods in the UDC require states that are not maintained by its subsystem model, it must access these shared states through the GDC, making it impractical for a copy of the state vector to be used for representing the state vector at each stage. Instead, a copy of the initial state vector is made at the start of the time step and stored until the final incremental term is assembled. Thus, the GDC uses its actual state vector to represent the states at each stage, enabling the methods in the UDCs to directly use the shared states in the GDC at each stage.

When the integration routine requests the derivatives to be calculated, the GDC requests all registered UDCs to calculate the derivatives at the current stage of the

integration. Each UDC calculates the derivatives for its subsystem states as well as the forces, moments and inertia parameters contributed to the dynamic model. While these derivatives are calculated in their respective UDCs, the parameters for calculating the kinetics of the model are assembled from all the UDCs registered with the model. The resultant forces and moments generated by the UDCs are assembled, as are their mass and inertia properties. The kinetics equations use these assembled parameters to calculate the accelerations. The kinematics equations uses these accelerations, the motion parameters of the body frame, and the motion parameters of the navigation frame with respect to the inertial frame to calculate the derivatives of the motion states.

When calculating the forces and moments, each UDC may require the motion parameters of the body frame to be expressed with respect to different reference frames, which can be achieved through the RFM. It is essential that the forces and moments use the same measurement frame as the acceleration and angular acceleration in the kinematics equations. Since forces and moments are typically expressed in the body frame, as are the acceleration and angular acceleration, additional transformations are generally not required. This property is also useful as it allows the GDC to replace the navigation frame with the intermediate frame to reduce roundoff error. The modifications to the integration routine due to the use of intermediate frame, as depicted in Figure 4.1, are automatically taken care of since each UDC transforms the motion parameters to its preferred navigation frame. The transformation of the resulting forces and moments for the kinetics equations is carried out as described above, and may not be necessary if the forces, moments and accelerations are all measured in the body frame. Thus, generic dynamic models do not require additional modifications to use intermediate frames.

CHAPTER 6

REFERENCE FRAME MANAGMENT IN PDS

Parallel and distributed simulations (PDS) often require dynamic models and other simulation components such as displays that are distributed over a large number of processors to interact with one another. These components may utilize different reference frames to express motion parameters, requiring the application of kinematics and rotations to allow their interaction. The reference frame manager's ability to generate kinematics and rotations between arbitrary reference frames in a network of reference frames can be extended to simulations distributed over multiple processors. Each processor can maintain its own network of reference frames, which is then linked to networks in other processors, allowing simulation components on one processor to interact with components on other processors using different reference frames. This would be especially useful in large-scale simulation such as air traffic control as it would allow numerous simulations with different dynamic models and components to be linked.

This chapter develops the network configurations and protocols that enable the reference frame manager to be used to handle reference frame transformations across a PDS. These configurations and protocols are developed using several design parameters. Feasible configurations and their protocols are implemented in the Reconfigurable Flight Simulator.

6.1 Managing Reference Frames in PDS

Kinematics and rotations between reference frames can only be generated if the motion parameter relating their relative motion is known. Chapter 3 dealt with the

creation of a network of reference frames in a simulation, linking each node in the network through the motion parameters of the child node with respect to the parent node. Similarly, a network comprising of all the reference frames in the PDS needs to be assembled enabling the kinematics and rotations between any pair of reference frames to be generated. The following subsections describe some of the factors that influence the development of a network of reference frames across multiple processors, including the distribution of reference frames across the network and the control mechanism used to coordinate the formation of the network and the assembly of kinematics and rotations.

6.1.1 Design Parameters for a Network of Reference Frames in PDS

A number of design parameters influence the design of a network of reference frame in a PDS. The location of reference frames across the different processors and the coordination required for identifying, generating and executing the path between the reference frames when generating the required kinematics and rotations are of particular interest.

6.1.1.1 Location of Reference Frames in PDS

The first parameter, the location of the reference frames across the different processors, provide several alternatives that can be considered. First, each processor on the network may be given all the reference frames used by all the components in the simulation. This requires a coordinated effort to determine all the reference frames that may be used and load them into the RFM on each processor. The advantage of this configuration is that the processor requesting the kinematics and rotations executes all the operations using its own network, reducing network traffic to the motion states and their appropriate reference frames. However, it also means that all the reference frames need to

be identified and loaded a priori on to each processor, which may be impractical. It may also lead to a large number of redundancies if a large number of reference frames are required in the various simulations. Thus, a large coordination effort is required to set up the configuration, allowing each processor to handle all network operations on its own, while eliminating any coordination requirements between processors for path generation at runtime.

The second option requires each processor to load the minimum number of reference frames required to assemble a network that contains all the reference frames used by the components on that processor. The RFM on each processor will only have access to these 'local' reference frames. If a component requires the motion parameters of a 'remote' reference frame that is not within its processor's RFM to be expressed with respect to its desired 'local' reference frame, the RFMs on the various processors will be required to assemble the kinematics and rotations. Identifying the path linking reference frames across multiple processors may be very computationally intensive, requiring multiple stages. However, designating a common reference frame for all the processors could greatly simplify this operation, as only the processors containing the 'local' and 'remote' frames would be involved.

The third option is for the location of the reference frames to be independent of the components that need them. Each RFM would form a network of reference frames based on the availability of reference frames rather than the requirements of simulation components on that processor. While this is the most general case with regards to the distribution of simulation components and reference frames, it may not be very practical, as propagation of dynamic models and the interactions between simulation components

on the same processor may require the identification, assembly and execution of kinematics and rotations along a path distributed over multiple processors.

6.1.1.2 Coordination Between Reference Frame Managers in PDS

The second design parameter is the method used to coordinate reference frame related operations between the processors. There are two alternatives that may be considered; a centralized system that coordinates all kinematics and transformation operations between the processors or a decentralized system where the kinematics and rotations are assembled only by the processors affected by the operation.

In a centralized system, the RFM on one processor is designated as the 'controller' of the other reference frame managers. This controller accesses the reference frames on all the processors, assembles the overall network and handles all the requests for kinematics and rotations involving multiple processors. The RFM on each processor in the simulation is linked to the controller, allowing it to access the RFM on each processor in the simulation. Figure 6.1 depicts a network comprising of four processors with processor 1 designated as the controller.

Each RFM is responsible for maintaining the motion parameters of reference frames in its processor and assembles its own network using these reference frames. The controller obtains the identity of each reference frame and its definition frame from the RFM on each processor and constructs a 'virtual' network of all the reference frames in the simulation as depicted in Figure 6.2. Each node in this virtual network records the identity of the reference frame it represents, the processors implementing it and its parent and child nodes.

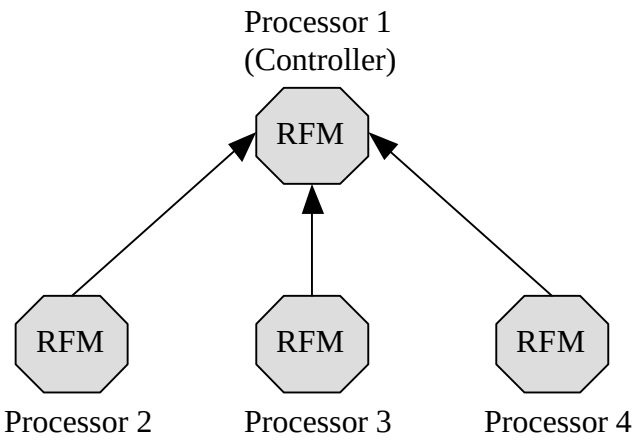


Figure 6.1: Centralized Control Links All Other Processors to 'Controller' RFM

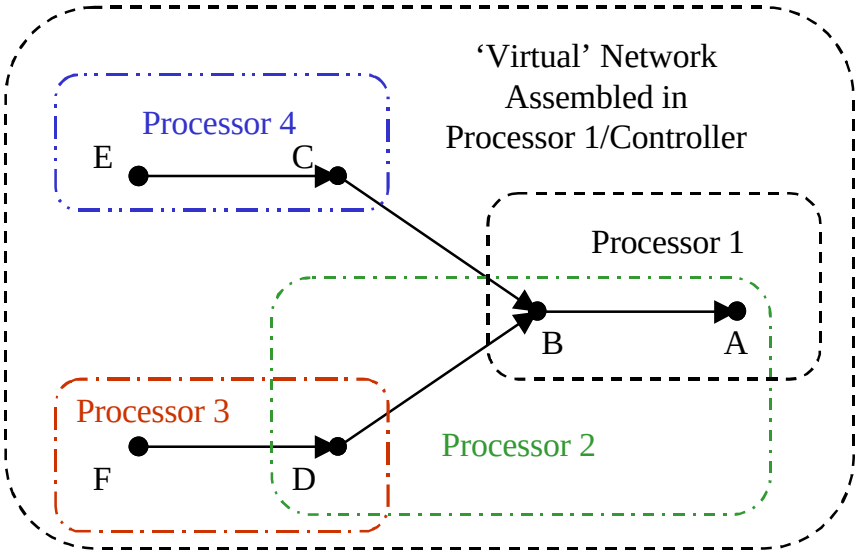


Figure 6.2: 'Virtual' Network Created in Controller RFM

When kinematics and rotations between reference frames are required, the RFM on the processor requiring the operation requests the assistance of the controller if it requires a reference frame located only on another processor. The controller assembles the path between the reference frames using its virtual network and executes the

operations requiring reference frames that are not located on the RFM requesting the operation. The motion parameters for these operations are obtained from the respective processors. If a reference frame exists on multiple processors, controller attempts to reduce the number of processors it needs to contact by identifying chains on each processor corresponding to segments of the path. Reducing the number of processors involved is important because each interaction between processors is subject to network lag, adversely affecting the performance of the simulation. The appropriate processor executes operations along these chains. Data resulting from these operations is passed to the requesting RFM, which executes the remaining operations using its own network.

In a decentralized system, only the processors implementing reference frames along the path connecting the initial and final frames systematically build the path and execute operations along it. Since reference frames are treated as nodes in the RFM, it is tempting to extend this to PDS by viewing the system as a network of nodes where each node represents the processor's RFM as depicted in Figure 6.3.

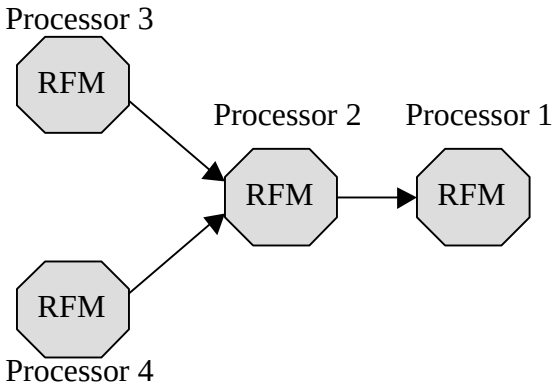


Figure 6.3: Network of RFM using Decentralized Control

The network described above implies that the links between processors are merely an extension of the links between reference frames. Thus, a link between processors in Figure 6.3 represents the link between the root node of one RFM with a node in the RFM of another processor. Such a representation assumes that the reference frames in each processor are unique to that processor and that the reference frames in the various processors can be assembled into a network depicted by Figure 6.4.

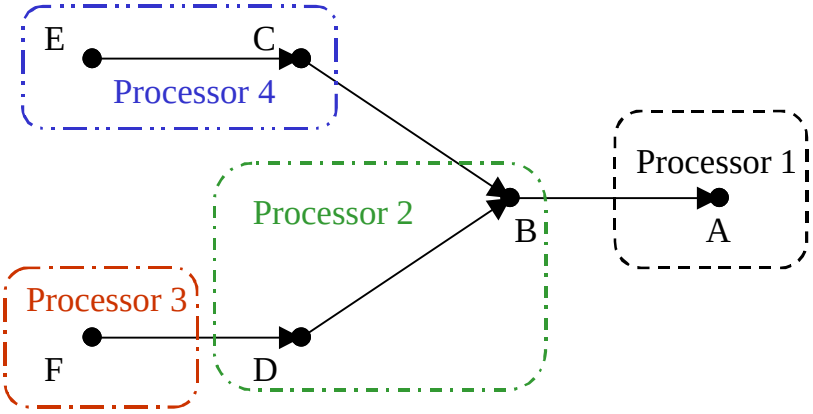


Figure 6.4: Linking RFM Can Ideally Assemble Larger Networks

In practice, the distribution of reference frames over processors as depicted in Figure 6.4 is unlikely, as common reference frames may be required by components in multiple processors or reference frames required to form a single network may not be loaded on any processor as depicted in Figure 6.5. The latter complication, depicted by the exclusion of frame C from the processors, also affects the centralized controller.

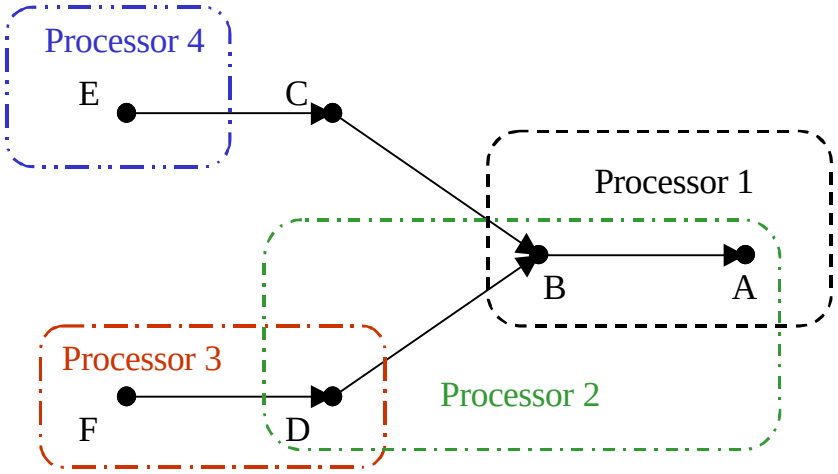


Figure 6.5: Actual Reference Frame Distribution Complicates Network Assembly

The presence of reference frames in multiple processors introduces additional complexity in assembling paths as the root node of an RFM may encounter multiple instantiations of its definition frame on different processors. In such cases, the root node's RFM will need to contact all the processors containing the root node's definition frame and attempt to determine the path that will affect the least number of processors in order to improve performance. The resulting path is stored by the RFM requesting the operation and processors affected by the operation store their respective segments. The kinematics and rotations for each segment are then executed in sequence.

While the decentralized system described above avoids the bottleneck created by using a central controller, it is an expensive method in terms of interaction between processors. An alternate approach to decentralized control is similar to the current practice of defining a common reference frame that is loaded in all processors. While the traditional approach described in Section 2.4.4 fixes this common reference frame during the development phase of PDS, reference frame management allows this to be set at runtime. The common reference frame can be designated the 'network' frame and can be

considered as the common node that links the RFMs on all the processors as depicted in Figure 6.6.

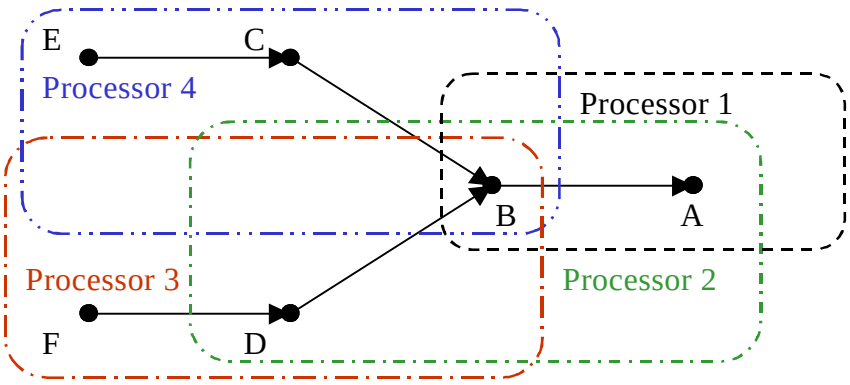


Figure 6.6: Network With a Common Reference Frame Shared by All Processors

When the network frame is designated at runtime, it is essential to ensure that any additional reference frames that may be required to form a single network in each RFM is added to the processor. The network frame does not have to be the root node in the reference frame managers. Since each RFM is able to generate a path linking the network frame to any reference frame in its local network, motion parameters expressed with respect to the network frame can be readily operated on by any RFM in the simulation. Expressing shared motion parameters in the network frame allows them to be expressed with respect to the preferred reference frame of any simulation component through the RFM on the component's processor. Since each RFM handles operations from the network frame to any reference frame in its local network, interactions between processors due to network operations are eliminated, improving network performance and scalability.

6.1.2 Population and Evaluation of the Design Space

From the design parameters described above, a total of 6 combinations are available, as shown in Table 6.1. Some of these combinations may be redundant or may be very inefficient; for example the combination C4 where each processor has all reference frames in a centralized system is redundant since a controller is not required if each processor has all the reference frames used in the simulation. In contrast, combination C3 is extremely inefficient since the lack of a central controller will imply that any reference frame operation will require a large number of interactions between processors.

Table 6.1: Design Parameters and Their Combinations for Distributed RFM

	Decentralized Control	Centralized Control
All Reference Frames are loaded	C1	C4
Only required reference frames are loaded	C2	C5
Reference frames loaded independent of requirement	C3	C6

It has been noted in Section 6.1.1.2 that a centralized controller will require several interactions between the controller and the other processors if any of the reference frames required to link the initial and final frames are not available on the processor requesting the operation. While the controller and the affected processors are executing the operation, the processor requesting the operation will not be able to proceed. Thus, centralized control incurs significant penalties to network performance, as processors will have to wait for multiple interactions to be completed before proceeding whenever the

controller is required for resolving kinematics and rotations between reference frames. This leads to the elimination of centralized control as a design option. This is supported by fact that current research indicates that bottlenecks are often caused if control is centralized when dealing with timing issues or time critical events. The implementation of the remaining design options, C1 and C2 are discussed below.

6.2 Implementation

The design options C1 and C2 as described by Table 6.1 are implemented in the Reconfigurable Flight Simulator. Both these design option use decentralized control to handle reference frame operations involving multiple processors, eliminating the need to extend the RFM implemented in Chapter 3 to include additional interfaces and functionality for assembling and utilizing the 'virtual' network described above. However, both configurations will use the RFM to manage a network of reference frames on each processor. Since the implementation of these designs involves the transfer and interpretation of data between processors, the implementation of the networking protocols in RFS are briefly described.

6.2.1 Networking in the Reconfigurable Flight Simulator

In RFS, the simulation environment includes a pointer to a networking object that can be implemented using different networking protocols. The capability in RFS to represent and execute methods as character strings allows the interface of the networking object to be expanded when necessary. The current networking object in RFS uses the HLA protocols discussed in Chapter 2 and is implemented as the HLA Networking Object. In the HLA paradigm, the instantiation of RFS on each processor is known as a federate and the entire PDS is known as a federation. The tasks of the HLA Networking

Object include the subscribing and publishing of attributes shared by individual objects, as well as sending and receiving methods, messages and events through interactions.

Since RFS primarily simulates aerospace vehicles, specific properties of these vehicles are shared as attributes through the HLA Networking Object. These attributes include the motion states of the vehicles that may change continuously throughout the simulation. Since transmitting the state vector of each vehicle to all the other federates would result in a prohibitive amount of network traffic, 'dead reckoning' algorithms are implemented through Remote Vehicle Approximation (RVA) and Remote Vehicle Monitoring (RVM) objects. These objects are used to maintain copies of each vehicle that implement basic 6DOF kinematics. The RVM is maintained by the vehicle's federate while the RVA is maintained by the other federates. When an attribute of the vehicle differs from its equivalent in the RVM by a predefined error tolerance, this attribute is updated in RVM and all corresponding RVA objects in other federates, reducing the network traffic required to update the vehicle's attributes. The representation of time in the dead reckoning models and federates also affects the accuracy of the simulation. The networking object in RFS uses the time management services of HLA to ensure that the simulation time in the different federates are synchronized to within a specified tolerance. When the RVM updates its RVA objects, the motion states corresponding to the RVM's time stamp are used. The current implementation of the network object in RFS does not account for different time stamps when updating attributes, introducing errors in RVA objects proportional to the difference in time stamps.

In the standard implementation of RFS, the motion states maintained by the RVM and RVA objects mirror the motion states of the vehicle, regardless of the reference

frames used by the vehicle. Thus, the reference frames used by the RVA and RVM are the same as those used by the vehicle, requiring all vehicles and simulation components to use identical reference frames when sharing motion states. This has been the standard approach in the development of PDS as mentioned in Chapter 2.

The introduction of the Reference Frame Manager allows the vehicles to use their preferred reference frames since all kinematics and rotations are handled by the RFM. When applying the RFM to PDS, manner in which the RFM is used depends upon the configuration of the design parameters described in this chapter. Since the network object does not account for differences in time stamps, the errors in representation of the RVA object's motion affects the kinematics calculated by the RFM irrespective of its configuration. The effect of these configurations on the interaction of the RFM with simulation components is described below.

6.2.2 Implementation of RFM in Configuration C1

Configuration C1 requires all the reference frames used in the simulation to be loaded into each RFM across all federates. Since the RFM on each federate has access to all the reference frames in the simulation, all kinematics and rotations can be carried out locally, eliminating the need for interactions between federates when assembling and executing the operation. Furthermore, the RVM and RVA objects associated with each vehicle express their motion states in the reference frames used by the vehicle. However, each vehicle would also be required to publish the identities of its definition and measurement frames. Thus, the identities of these frames are added to the attributes of the RVM and RVA objects generated by the HLA Networking Object. This configuration, depicted in Figure 6.7, would minimize the path length for kinematics and transformation

operations for any given task at the cost of configuration complexity, as all the reference frames used by all simulation components would need to be identified and loaded in each federate.

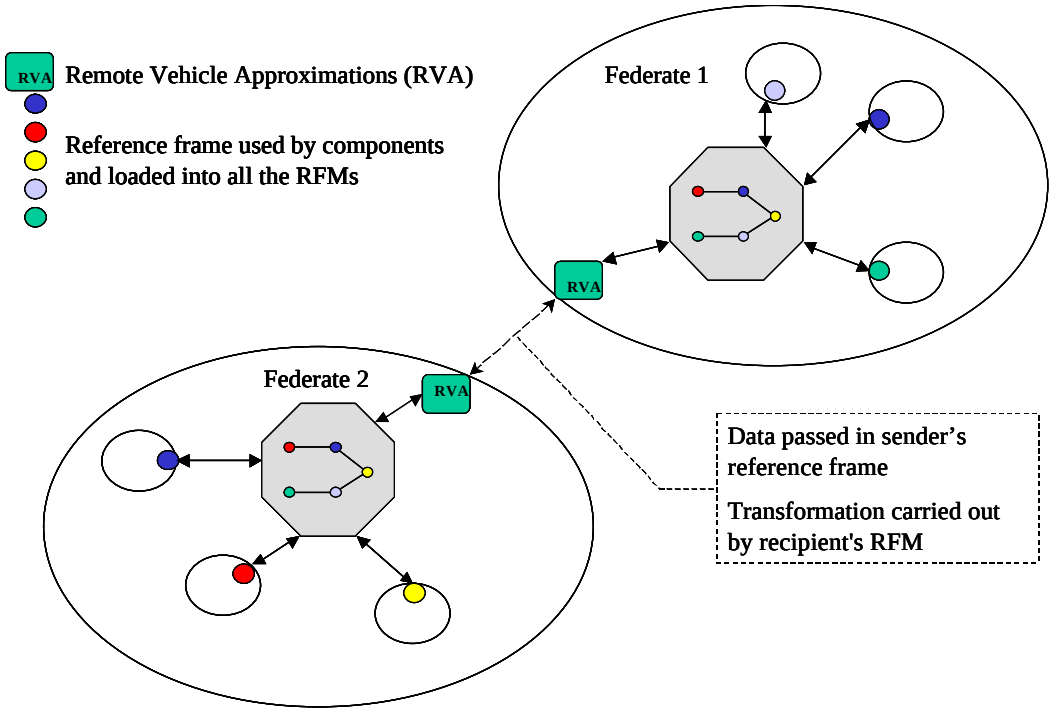


Figure 6.7: All Reference Frames Loaded in Each Federate

6.2.3 Implementation of RFM in Configuration C2

Configuration C2 requires all the reference frames used by the components in a federate to be loaded into its RFM along with a network frame that is specified and loaded into each RFM at the start of the simulation. Any additional reference frames required to link the reference frames in the federate to the network frame are also loaded. While each vehicle can maintain its motion states in its preferred reference frames, the corresponding RVM and RVA objects need to express their motion states in the network

frame. Therefore, the RVM in the HLA Networking Object is modified to use the RFM to express the motion parameters of the vehicle with respect to the network frame. Since motion states are shared in a reference frame common to all federates, kinematics and rotations can be carried out locally, eliminating the need for interactions between federates when assembling and executing the operation. This configuration, depicted in Figure 6.8, reduces configuration complexity, as the RFM uses the minimum number of reference frames to support the simulation components in the federate with the network frame and their desired reference frames. While this is similar to the current standard, a convenient network frame can be chosen at runtime and all reference frame operations are handled by the RFM.

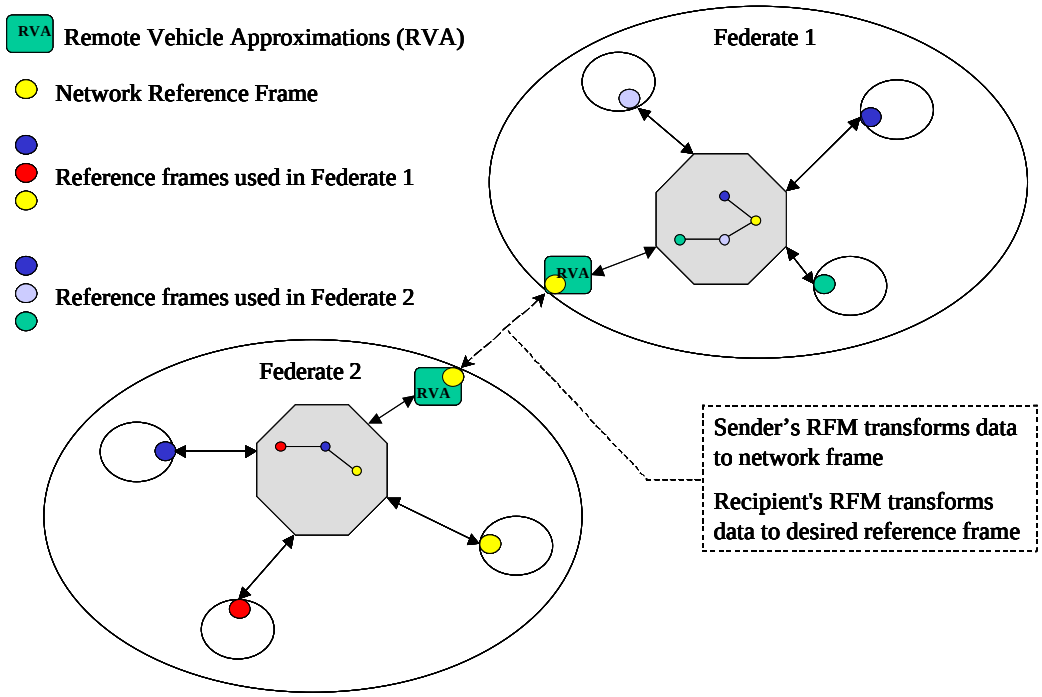


Figure 6.8: Network Frame Used to Share Motion Parameters Between Federates

CHAPTER 7

DEMONSTRATION OF REFERENCE FRAME MANAGEMENT

The implementation of the Reference Frame Manager, Intermediate Frames and Generic Dynamic Models were verified through the series of demonstrations documented in this chapter. Simulations were configured to demonstrate and verify the capabilities and benefits of Reference Frame Management and its supporting components. The reduction in roundoff error using intermediate frames and the relationship between error reduction, critical levels and time step were also analyzed. The application of the RFM to Parallel and Distributed Simulation (PDS) was verified with demonstrations of both configurations of RFM and reference frames as described in Chapter 6. This chapter describes the scenarios used as well as the costs and benefits of these components measured during the demonstrations.

7.1 Measures of the Capabilities, Benefits and Costs of RFM and its Applications

The successful demonstration of RFM and its supporting components requires measuring their capabilities and evaluating their costs and benefits. This section describes the measures of the capabilities, benefits and costs of RFM, GDM, intermediate frames and the application of RFM in PDS. The methods used to verify the capabilities and evaluate the benefits and costs are also tabulated.

In order to handle all reference frames in the simulation environment and express their motion parameters with respect to any other reference frame, the RFM has to demonstrate several capabilities. The first is the ability to assemble a network of reference frames. The second is the identification of paths connecting nodes in the

network and the third is the execution of kinematics and transformation operations along these paths. These capabilities are critical to the RFM and are verified through the successful operation of its applications. The benefits of RFM include the ability of simulation components to utilize their preferred reference frames when interacting with other simulation components, regardless of the reference frames used by the other components in expressing motion parameters. Also, the development time, cost and effort are reduced as the components do not require algorithms to calculate kinematics and rotations and do require modifications to support the addition of new components and reference frames to the simulation environment. However, the use of RFM may incur additional computational costs and numerical errors, as its calculations are generic to all reference frames rather than being optimized for specific reference frames. The methods for evaluating the capabilities, benefits and costs of RFM are tabulated in Table 7.1.

Table 7.1: Evaluating the Capabilities, Benefit & Costs of RFM

<table>
<tr><th colspan="2"></th><th>Methods for Evaluating RFM</th></tr>
<tr><td rowspan="3">Capabilities</td><td>Assembly of Network</td><td rowspan="3">• Successful operation of simulation validates capability of the RFM</td></tr>
<tr><td>Identification of path</td></tr>
<tr><td>Generation of kinematics and rotations along path</td></tr>
<tr><td rowspan="2">Benefits</td><td>Interaction of components using different reference frames</td><td>• Level of effort in enabling the interaction of components using different reference frames</td></tr>
<tr><td>Ease of developing new simulation components</td><td>• Level of effort in developing simulation component software</td></tr>
<tr><td rowspan="2">Costs</td><td>Computational load</td><td>• Runtime of scenario</td></tr>
<tr><td>Numerical accuracy</td><td>• Numerical error</td></tr>
</table>

The construction of dynamic models using a GDM and appropriate UDCs requires the GDM to demonstrate its ability to encapsulate the core services of the dynamic model and dynamically form the assembled state and derivative vectors through the addition or removal of UDCs. The GDM must also demonstrate its ability to assemble kinetics equations using the properties provided by the UDCs. The benefits of using the GDM include improved reusability of components and code, reduction in development time, effort and cost of new dynamic models and the ability to reconfigure the fidelity and dynamics of the models without the extensive modification of the software. However, the use of GDM may incur additional computational costs as well as numerical errors as the kinetics and kinematics equations are generalized for arbitrary reference frames instead of being optimized for specific reference frames. The methods for evaluating the capabilities, benefits and costs of GDM and UDCs are listed in Table 7.2.

Table 7.2: Evaluating the Capabilities, Benefit & Costs of GDM & UDC

		Methods for Evaluating GDM & UDCs
Capabilities	Encapsulation of core services	• Successful operation of simulation validates capability of the GDM
	Assembly of model's state and derivative vectors	
	Assembly of kinetics equations from UDCs	
Benefits	Ease of development for new dynamic model	• Level of effort in developing new dynamic model
	Ease of modifying dynamic models through code reuse	• Level of effort in modifying existing dynamic models for new scenarios
Costs	Computational load	• Runtime of scenario
	Numerical accuracy	• Numerical error

The successful operation of intermediate frames relies on RFM's ability to insert the intermediate frame into the network, and the intermediate frame's ability to evaluate critical levels and to update its motion parameters and the motion states of the dynamic model accordingly. The critical levels for velocity also need to be able to adapt to the dynamics of the model to minimize the roundoff errors for both position and velocity. The benefit of intermediate frames is the reduction of roundoff error at the cost of additional computational load. The methods for evaluating the capabilities, benefits and costs of intermediate frames are tabulated in Table 7.3.

Table 7.3: Evaluating the Capabilities, Benefit & Costs of Intermediate Frames

<table>
<tr><th colspan="2"></th><th>Methods for Evaluating Intermediate Frames</th></tr>
<tr><td rowspan="3">Capabilities</td><td>Insertion of intermediate frame into the network by the RFM</td><td rowspan="3">Motion of intermediate frame and critical level bounds viewed in a graphics display
Magnitude of motion states in the navigation frame and intermediate frame</td></tr>
<tr><td>Bound vehicle states through updates of intermediate frame</td></tr>
<tr><td>Monitoring of critical levels</td></tr>
<tr><td>Benefits</td><td>Reduction of roundoff error</td><td>Comparison of roundoff errors with and without intermediate frames</td></tr>
<tr><td>Costs</td><td>Computational load</td><td>Runtime of scenarios with and without intermediate frames
Number of path operations</td></tr>
</table>

The application of RFM in PDS requires the RFM on each processor to handle all reference frame operations on its processor while the networking objects need to employ the appropriate data passing protocols for identifying reference frames. If the network is configured such that each federate possesses all reference frames required by the

components in the federation, the networking object needs to demonstrate the ability to publish and subscribe the identity of the reference frames used to express the motion states of dynamic models. If the network expresses the motion parameters of all RVA and RVM objects with respect to a network frame, each networking object has to be able to express the motion states of the dynamic models with respect to the network frame when updating the RVA and RVM objects. The benefit of applying RFM to PDS is the elimination of the need to fix a common network frame during the development phase of the simulation. Therefore, the components in each federate are able to use reference frames appropriate to the scenario. The relative cost of the two methods of implementing RFM in PDS can be evaluated by comparing the complexity of their configurations as well as their computational load.

Table 7.4 Evaluating the Capabilities, Benefit & Costs of RFM in PDS

		Methods for Evaluating RFM in PDS
Capabilities	RVA subscription to the reference frame published by the RVM	• Comparison of the behavior of RVA with their corresponding dynamic models
	RVM & RVA expression of motion in the appropriate reference frames	
Benefit	Elimination of the need to fix network frame during software development	• Successful operation of both configurations verifies the elimination of pre-defined network frame
Costs	Computational load	• Number of path operations per configuration
	Configuration complexity	• Number of reference frames loaded on the processors for each configuration

7.2 Simulation Configurations for the Demonstrations

The demonstrations of RFM and its applications involved the simulation of satellites in orbit around the Earth. Three different demonstrations were used to verify and evaluate different aspects of RFM, GDM, their application to PDS and the effectiveness of intermediate frames. The first demonstration verified the operation of the RFM and GDM and evaluated their costs and benefits. The second demonstration verified the operation of RFM in PDS and compared the two methods of distributing reference frame in PDS described in Chapter 6. The final demonstration was used to verify and analyze the effectiveness of intermediate frames in reducing roundoff error. The satellite model and the scenarios for these three demonstrations are described in detail in the following subsections.

7.2.1 Satellite Dynamic Models and Reference Frames

The satellite model [36] used in all the demonstrations was implemented using the Generic Dynamic Model described in Chapter 5. The GDM was used in conjunction with a suitable UDC that modeled gravity exerted on the body by an attracting body *M* with mass M_g, expressed in equation (7.1). If the navigation frame is located at the center of mass of the attracting body, this expression is simplified, as depicted in equation (7.2):

$$ {}^{bf}\underline{F}_{bf} = -\frac{GM_g m}{\left| {}^{bf}\underline{P}_n^n - {}^{M}\underline{P}_n^n \right|^3} \left[{}^{bf}C^n \right] \left({}^{bf}\underline{P}_n^n - {}^{M}\underline{P}_n^n \right) \tag{7.1} $$

$$ {}^{bf}\underline{F}_{bf} = -\frac{GM_g m}{\left| {}^{bf}\underline{P}_M^M \right|^3} \left[{}^{bf}C^M \right] \left({}^{bf}\underline{P}_M^M \right) \tag{7.2} $$

Since the GDM represents all the common elements in the dynamic model, it did not require any modification. The UDC was developed to be a generic gravity source. Thus, the mass of the attracting body was parameterized so that it could be set at runtime through a configuration file. The model was simplified such that the only force affecting the kinetics was the force of gravity, allowing the kinetics equation to be expressed as equation (7.3).

$$\frac{{}^{i}d}{dt}\left({}^{bf}\underline{V}_{bf}^{i}\right) = -\frac{GM_g}{\left|{}^{bf}\underline{P}_{M}^{M}\right|^3}\left[{}^{bf}C^{M}\right]\left({}^{bf}\underline{P}_{M}^{M}\right) \tag{7.3}$$

Encapsulating the gravitational forces in a parameterized UDC allowed the final model to use either two body orbital mechanics or multi-body orbital mechanics by adding or removing UDCs representing the effect of different gravity sources. If N attracting bodies are used, where the center of mass of the j^{th} attracting body of mass M_j is represented by frame n_j, the resultant force on the satellite is expressed using equation (7.4). The UDC representing each attracting body uses the RFM to express the position of the satellite with respect to center of mass of the attracting body.

$$\frac{{}^{i}d}{dt}\left({}^{bf}\underline{V}_{bf}^{i}\right) = -\sum_{j=1}^{N}\frac{GM_j}{\left|{}^{bf}\underline{P}_{n_j}^{n_j}\right|^3}\left[{}^{bf}C^{n_j}\right]\left({}^{bf}\underline{P}_{n_j}^{n_j}\right) \tag{7.4}$$

If the satellite's navigation frame is not the reference frame at the center of the attracting body, the UDC is able to obtain the position of the satellite with respect to the

center of the attracting body using the RFM, allowing the satellite's navigation frame to be modified without affecting its dynamics. Similarly, if intermediate frames are introduced to control roundoff error and the position of the satellite is expressed with respect to the intermediate frame, the RFM enables each UDC to express the motion states of the satellite in the appropriate reference frame to calculate the forces.

In addition to the implementation of the dynamic model using the GDM, a control vehicle was also implemented without the GDM. Since the control model did not use RFM, the models of the reference frames used in the different configurations were included in the dynamic model. The additional functionality required by the control model to use different navigation frames were recorded to demonstrate the complexity of modifying dynamic models without the GDM.

Since the numerical accuracy of the satellites is one measure of the cost of using RFM and GDM, the local roundoff error and its maximum limit are recorded at every time step for both models. If intermediate frames are used, the error generated at their updates is also recorded in the GDM. Finally, the actual model error is evaluated at the end of the simulation run for both models by comparing the actual states of the dynamic models and their predicted states based on their orbital parameters.

Several reference frames can be used when simulating satellites in orbit around the Earth. The reference frames used in the demonstrations as the satellites' navigation and inertial frames include the non-rotating Earth Centered Inertial (ECI) frame, the rotating Earth Centered Earth Fixed (ECEF) frame and a variety of Earth Surface Fixed (ESF) frames. Since the development of reconfigurable and reusable software is one of the aspects of this research, a single class capable of representing these types of reference

frames, the Generic Orbiting Reference Frame class, was developed using the reference frame classes described in Chapter 3. The interfaces of this frame allow it to be initialized as a rotating or non-rotating frame that can be placed in orbit about another reference frame. The parameters for its orbit can also be set at runtime. Thus, different instantiations of the same class can be used as the ECI, ECEF and different ESF frames. The definition frame and inertial frame for this class are also set at runtime through the configuration file, allowing a network of reference frames to be rapidly created.

7.2.2 Demonstration 1: Capabilities of the RFM and GDM

The objectives of this demonstration were to demonstrate the capabilities of the RFM and GDM and evaluate their benefits and costs. In particular, the benefits of RFM and GDM with respect to software reuse and interoperability of simulation components and their costs with respect to computational and numerical cost were evaluated using the methods and measures listed in Table 7.5.

This demonstration involved the simulation of satellites in geo-stationary orbit around the Earth. Canonical units [37] were used to express the motion parameters as well as simulation time. The semi-major axis of the orbit was set to 6.6107 DU while the eccentricity and inclination were set to 0.0 degrees. 10 geo-stationary satellites were propagated in each simulation run for one orbit. The initial true anomalies of these satellites were randomized at the start of the simulation, effectively randomizing their positions and velocities along the orbit. A total of 10 simulation runs, corresponding to 100 satellites, were executed per condition. Twenty-four conditions were created by four independent variables: the identity of the navigation frame, the use of RFM & GDM, the use of a display component requiring data from all satellites and the time step. These

independent variables and their different levels are summarized in Table 7.6. A full factorial of these independent variables produces a total of 36 conditions. However, the use of the display component did not significantly affect the operation of the RFM and GDM. Thus, its use was restricted to the larger time step and configurations using only the control model or the RFM & GDM, resulting in the 24 conditions.

Table 7.5: Methods and Metrics in Demonstration 1

	Verification & Evaluation Method	**Measures**
Capabilities of GDM & RFM	• Successful operation validates capability of RFM & GDM • Compare the trajectories of the generic and control models	• Motion states of the GDM & control models
Interoperability of simulation components using different reference frames	• Compare the level of effort required to ensure that the generic model and control model can interact with different reference frames	• Description of functionality required
Ease of developing new simulation components and dynamic models	• Compare the level of effort required to develop the generic and control models	• Lines of code (SLOC) • Description of functionality required • Description of development effort
Ease of modifying and reusing existing models in different scenarios	• Compare the level of effort required to change the navigation frame used by the generic and control models	
Numerical cost	• Compare the error incurred by the generic and control models when using different navigation frames	• Final error: Distance between actual and predicted position
Computational cost	• Compare the runtimes of scenarios using generic and control models	• Runtime

Table 7.6 Independent Variables and Their Levels in Demonstration 1

Independent Variable	Levels
Navigation Frame	ECI, ECEF, ESF
Use of RFM & GDM	Control Model, RFM & Control Model, RFM & GDM
Use of display component	Display not used, display used to view satellite motion
Time Step	1.06795 TU, 0.106795 TU

The satellites used the ECI frame as their inertial frame while their navigation frame was based on the demonstration conditions. The display provided a 3 dimensional visualization of the trajectory. Figure 7.1 provides a schematic of the satellites and reference frames in the demonstration scenario.

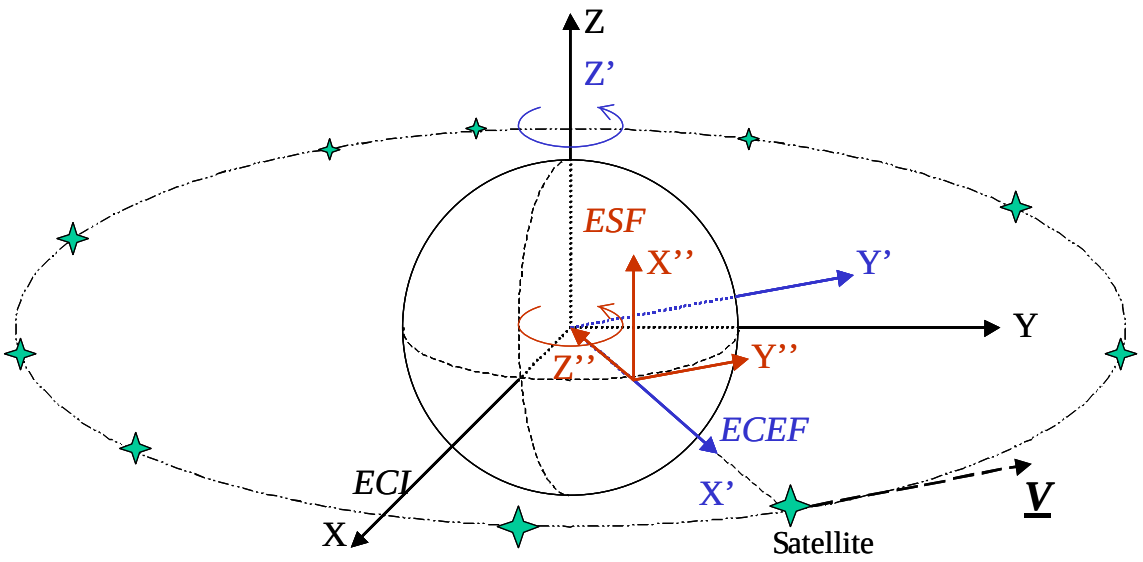


Figure 7.1: Schematic of 10 Satellites and the Reference Frames in Demonstration 1

7.2.3 Demonstration 2: RFM in PDS

The objectives of this demonstration were to verify the appropriate use of RFM by the RVM in PDS and compare the costs of the two configurations described in Chapter 6 for using RFM in PDS. The methods and measures used to verify the success of the demonstration and evaluate the two configurations are listed in Table 7.7.

Table 7.7: Methods and Metrics in Demonstration 2

	Verification & Evaluation Method	Measures
Appropriate use of RFM by RVM & RVA in PDS	• Verify behavior of RVA in the visualization display	• Motion of the RVA & dynamic model
Network frame not fixed during the development of simulation software	• Successful operation of both configurations verifies the elimination of pre-defined network frame	• None
Configuration complexity	• Compare the number of reference frames loaded on each federate for both configuration	• Number of reference frames
Computational load	• Compare the number of path operations required by each federate for both configurations	• Number of path operations

As in demonstration 1, the scenario for demonstration 2 involved the simulation of satellites in geo-stationary orbit around the Earth. In this case, four geo-stationary satellites were propagated using four processors for ten orbits. The initial true anomalies of these satellites were set to 0°, 90°, 180° and 270°. Figure 7.2 depicts these satellites after they have been propagated by approximately an eighth of their orbits from their initial conditions in a counter clockwise direction.

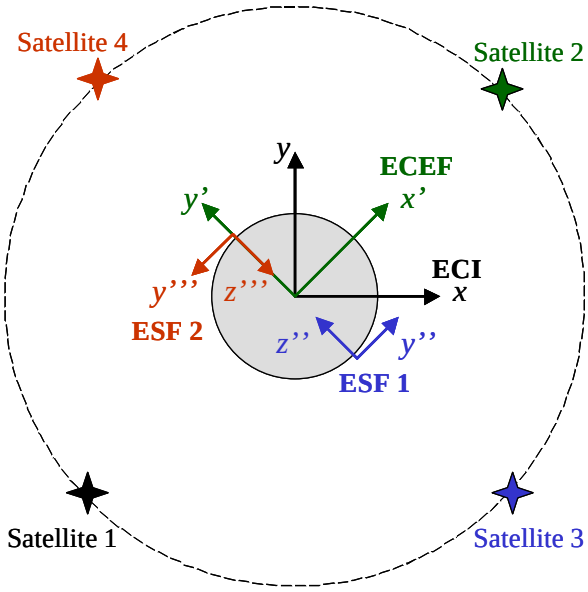


Figure 7.2: Schematic of Satellites and Reference Frames in Demonstration 2

Four different reference frames were used in the simulation environment as shown in Figure 7.2: an ECI frame, an ECEF frame and two ESF frames. Each satellite used a different navigation frame. Each processor was responsible for propagating one of the satellites and operating a display of all the satellites in the simulation. The reference frame used by each display corresponded to the navigation frame used by the satellite on its processor. Table 7.8 lists the reference frames used by the satellites and displays on each federate.

Table 7.8: Reference Frames Used by Satellites & Displays on Each Federate

Processor	Satellite Model	Navigation Frame	Display Frame
Federate 1	Satellite 1	ECI	ECI
Federate 2	Satellite 2	ECEF	ECEF
Federate 3	Satellite 3	ESF 1	ESF 1
Federate 4	Satellite 4	ESF 2	ESF 2

This demonstration was executed twice with a time step of 1.06795 TU and synchronized using the time management services of HLA. The two configurations detailed in Chapter 6 were tested. In the first configuration, the RFM on each federate was provided with all the reference frames used in the entire federation, allowing the motion parameters of satellites to be expressed with respect to any other reference frame in the federation without the need for a common reference frame. In the second configuration, only the reference frames required to support the dynamic model on that processor were used within each federate and the ECI frame was set as the common network frame.

7.2.4 Demonstration 3: Intermediate Frames

The objectives of this demonstration were to verify the operation of intermediate frames, evaluate their ability to reduce roundoff errors over a range of conditions, and assess their computational cost. The methods and measures used to verify the operation of intermediate frames and evaluate their effect on roundoff error are listed in Table 7.9.

As in demonstration 1, this demonstration involved the stochastic simulation of 10 satellites orbiting the Earth. The values for the semi-major axis and inclination were identical to demonstration 1, at 6.6107 DU and 0.0 degrees respectively. The initial true anomaly of each satellite was also set to 180°, and the longitude of perigee of each satellite's orbit was randomized at the start of the simulation. Therefore each satellite was initialized at the apogee of its orbit and the orientation of each orbit was randomized with respect to the ECI frame, as illustrated in Figure 7.3.

Table 7.9: Methods and Metrics in Demonstration 3

	Verification & Evaluation Method	Measures
Intermediate frame's ability to bound the vehicle's motion states with critical levels	• View motion of intermediate frame and critical level bounds in a graphics display • Compare vehicle's motion states in the navigation frame and intermediate frame	• Motion of the intermediate frame and critical level bounds • Motion states of the vehicle
Reduction in roundoff error for the vehicle's motion states	• Compare roundoff error incurred by models with and without intermediate frames for different conditions • Check for error reduction using a statistical paired t-test	• Global error • Limit for global roundoff error
Computational cost	• Compare the runtimes and number of path operations with and without intermediate frames	• Number of path operations • Runtime

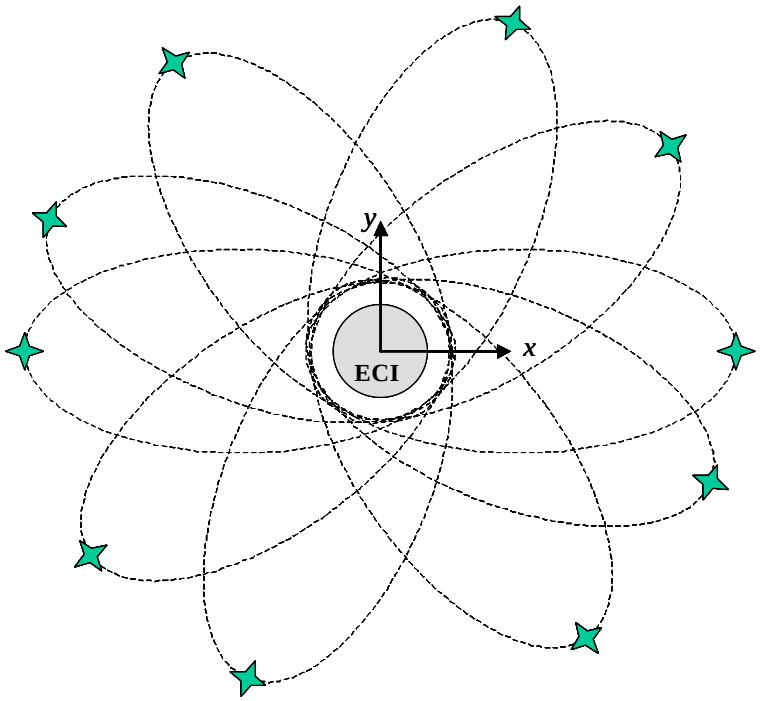


Figure 7.3: Schematic of 10 Satellites With Randomized Longitudes of Perigee

As in demonstration 1, 10 satellites were simulated per run and a total of 10 simulation runs, corresponding to 100 satellites, were executed per condition. The independent variables used to determine each condition included the use of intermediate frames, the values of critical levels, the eccentricity of the orbit and the time step. These independent variables and their different levels are listed in Table 7.10. While a full factorial of these independent variables produces a total of 240 conditions, only 68 conditions were used in the demonstration as the primary concern lay in the effectiveness of the adaptive critical level. A full factorial of the remaining independent variables using adaptive critical levels provides 48 conditions. The fixed critical levels were only used to demonstrate the need for critical levels to adapt to time steps and therefore restricted to fixed time steps and a single eccentricity of 0.00, providing the remaining 20 conditions.

Table 7.10 Independent Variables and Their Levels in Demonstration 3

Independent Variable	**Levels**
Use of Intermediate Frame	GDM only, GDM with Intermediate Frame
Critical level for velocity	2^{-3}, 2^{-7}, 2^{-12}, 2^{-17}, Adaptive Critical Level
Eccentricity of orbit, e	0.00, 0.25, 0.60, 0.85
Time Step, Δt (TU)	1.07, 1.07×10^{-1}, 1.07×10^{-2}, 1.07×10^{-3}, 1.07×10^{-4}, Adaptive

7.3 Results

This section deals with the analysis and observation of the results obtained from the three demonstrations. In particular, the benefits of RFM and GDM with regards to the development and interoperability of simulation components, the relative costs and merits of distributing reference frames in PDS, and the effectiveness of intermediate frames in

reducing numerical error are discussed. The computational and numerical costs of RFM and GDM are also discussed along with the computational cost of improving numerical accuracy with intermediate frames. A full tabulation of these results is provided in Appendix C.

7.3.1 Demonstration 1 Results: Capabilities of the RFM and GDM

The successful operation of demonstration 1 was evaluated by comparing the behavior of the simulated satellites with their predicted behavior using two body orbital mechanics. The motion of the control and generic satellites were observed with the visualization display for all the navigation frames. Since the motion of both the control and generic models matched the expected trajectories, the demonstration was deemed a success, verifying the capabilities of the RFM and GDM. In addition, the actual position and velocity errors of each model were recorded after one orbit. These errors refer to the magnitudes of the vectors relating the satellite's initial position and velocity to its position and velocity after one orbit, expressed by equations (7.5) and (7.6). The errors for both control and generic models were less than 0.01% in all the conditions.

$$\Delta P = \left| {}^{bf}\underline{P}_n^n - \underline{P}_0 \right| \tag{7.5}$$

$$\Delta V = \left| {}^{bf}\underline{V}_{bf}^n - \underline{V}_0 \right| \tag{7.6}$$

The successful operation of this demonstration verified key aspects of the RFM and GDM. Since the satellites used different navigation and inertial frames, they required the RFM to provide the kinematics and rotations between the navigation and inertial

frames. This in turn required the successful construction and maintenance of the network of reference frames, the identification of paths linking reference frames, and generation of the requested kinematics and rotations. Furthermore, the successful operation of the RFM was a prerequisite for the successful operation of the generic dynamic models.

The successful operation of generic dynamic models, indicated by comparing the behavior of the satellite models with their predicted behavior as described above, also verified several key aspects of the GDM. The first aspect was the GDM's ability to assemble the model, obtain forces and moments from a UDC and apply the kinetics and kinematics for the appropriate navigation and inertial frames, resulting in the assembly of the final state and derivative vectors. The second aspect was the GDM's ability to provide core services such as numerical integration. The final aspect was the GDM's ability to interact with the simulation environment and its ability to obtain kinematics and rotations from the RFM.

In addition to verifying specific capabilities of the RFM and GDM through the demonstration, the software development benefits were observed by comparing the development effort required for the control model and the generic model. The reduction in development effort was noted by comparing the number of lines of code as well as model components and functionality required in developing the GDM and control versions of the satellite model, as illustrated in Table 7.11. The number of lines of code required refers to the specific development of the models and does not include the code required by the standard RFS infrastructure or the code used record the errors within the model for evaluating the costs of RFM &GDM or the effectiveness of intermediate frames. The development effort that would be required to further modify the existing

model or enable it to interact with components using different reference frames is also listed in Table 7.11.

Table 7.11: Reduction in Development Effort With RFM & GDM

	Using GDM & UDC	Control Model
Number of Lines of Code	200	1000
Components and model functionality needed to develop the satellite model	1. Satellite initialization methods in GDM 2. Forces in UDC	1. Satellite initialization methods 2. Forces in model 3. States & derivatives 4. Integration routine 5. Propagate model of ECEF frame 6. Kinematics and rotations between body frame, ESF frame, ECEF frame and ECI frame
Functionality required: 1. To modify the dynamics of the model 2. To use a different navigation frame	1. Modify or code new dynamics in UDC 2. Change navigation frame through configuration file	1. Modify or code new dynamics in model 2. Model new navigation frame 3. Update the state & derivative array 4. Kinematics and rotations of new navigation frame
Functionality required to enable interaction with components using a different reference frame	1. RFM provides kinematics and rotations of new reference frame	1. Model of the new reference frame 2. Kinematics and rotations of the new reference frame

The numerical cost of RFM & GDM was observed by comparing the actual error at the end of each orbit for the control and generic models. This was carried out for both

time steps and all three navigation frames. The navigation frame used was of particular interest as the number of arithmetic operations carried out by the RFM when calculating the kinematics and rotations for the GDM is directly proportional to the transformation path length between the satellite's body frame and its inertial frame. Therefore, the path length when the navigation frame is set to the ECI, ECEF and ESF frames is 1, 2 and 3 respectively. Figure 7.4 depicts the 90th percentile, median and 10th percentile of actual errors in position for both the control model and generic model when the time step was set to 1.06795 TU.

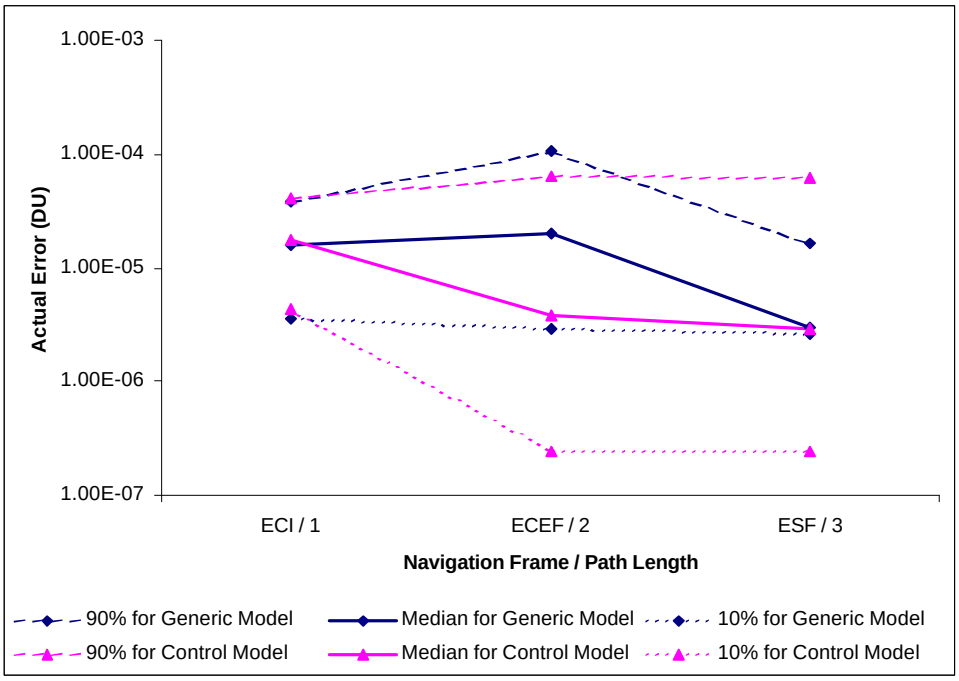


Figure 7.4: Actual Position Error for a Time Step of 1.06795 TU

When the navigation frame is set to the ECI frame, the path length is 1 and the actual error of the control model and generic model are identical, as seen in Figure 7.4. As the path length increases due to the use of the ECEF frame or ESF frame as the

navigation frame, the control model and generic model use different implementations of kinematics between the inertial and navigation frame. Since the kinematics in the control model is specific to each pair of reference frames, it requires fewer arithmetic operations and incurs fewer roundoff errors than the generic model, which needs to implement the complete set of kinematics equations. This numerical cost is further highlighted with the use of a smaller time step, increasing the effect of roundoff error. Figure 7.5 illustrates this error when the time step is set to 0.106795 TU.

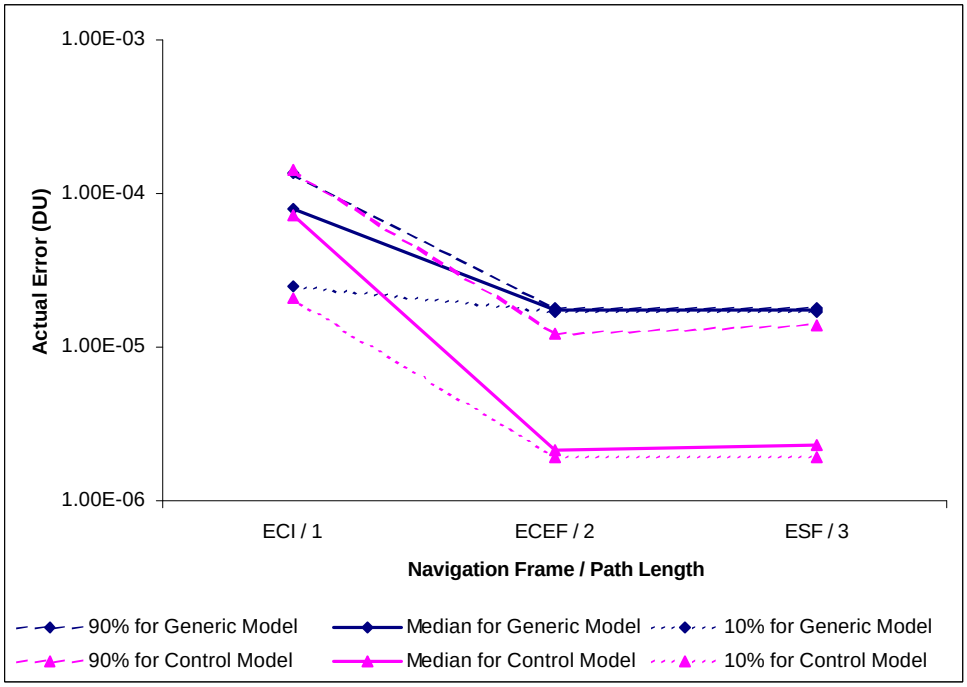


Figure 7.5: Actual Position Error for a Time Step of 0.106795 TU

At the smaller time step, the generic model has a significantly larger error than the control model when the path length is greater than 1. However, while the generic model does incur a larger roundoff error than the control model for longer path lengths, the

relative error incurred by both models is still less than 0.01% of the magnitude of its position vector.

The computational cost of RFM and GDM was observed by comparing the runtimes of conditions with control model, control model and RFM, and GDM and RFM. The runtimes for conditions using only the control model and conditions using the control model with RFM were almost identical. When the control model was used with the RFM, the RFM instantiated and propagated the reference frames in the simulation. However, the control model did not utilize the RFM. Thus, the computational load of modeling and propagating the ECI, ECEF and ESF frames was not significant compared to the computational load of simulating 10 satellite models. When generic models were used, however, the runtimes increased significantly when the visualization display was not used. This increase is observed in Figure 7.6, which depicts the runtimes of the control model and GDM, with and without the visualization display, for a time step of 1.0675 TU. The computational load is increased when the GDM is used because the RFM needs to search the network in addition to assembling the kinematics and rotations. The significance of the additional computational load depends upon the overall computational load of the simulation. In the case of simple satellite models, this additional load can be significant. However, as the computational load increases with the complexity of the model or other simulation components, the effect of this additional computational load is less significant, as seen in Figure 7.6 where the runtime is almost identical for the control and generic models when the visualization display is added to the simulation.

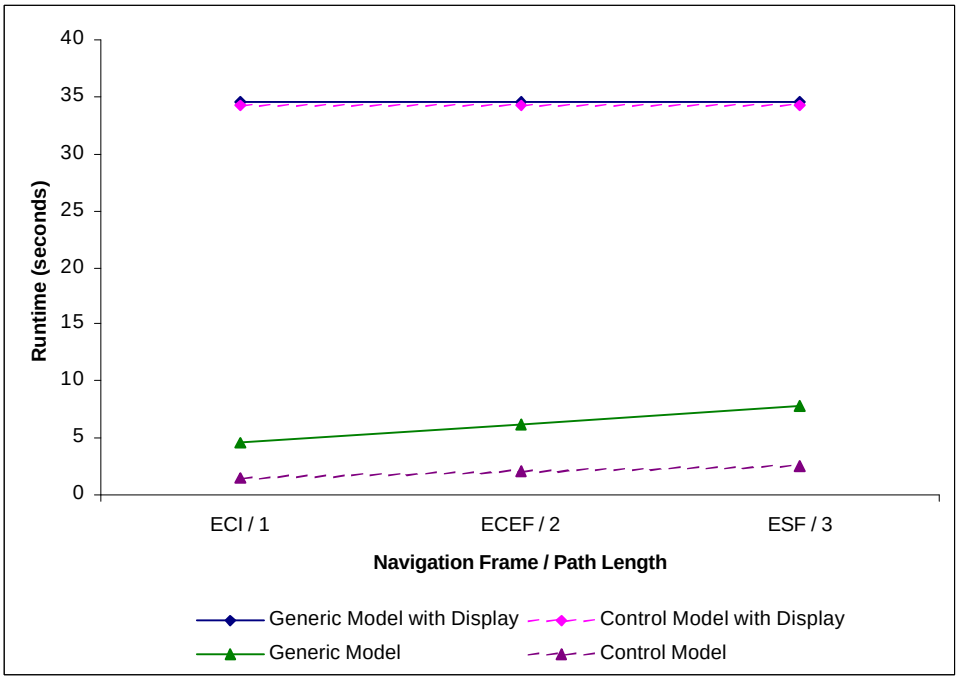


Figure 7.6: Computational Cost of RFM & GDM

7.3.2 Demonstration 2 Results: RFM in PDS

The successful operation of demonstration 2 was evaluated by comparing the behavior of the RVAs with their corresponding dynamic models. The motions of the generic satellites and RVAs on the four processors were observed with the visualization displays on all four processors. Since all four satellites were in geo-stationary orbit, their relative positions were always constant with respect to one another. Furthermore, the display frames for three federates were earth fixed frames, ensuring that the satellites and their RVAs would be stationary with respect to these display frames, facilitating the ease of visually observing and verifying their trajectories through the displays.

The successful operation of this demonstration verified the ability of the RFM to be utilized effectively in PDS. The HLA Networking Object's ability to utilize the RFM

in different configurations was also verified. In addition to verifying the operation of RFM in PDS, this demonstration also allowed the relative costs of using different configurations of distributing reference frames to be evaluated. Figure 7.7 depicts the number of path operations executed by each federate for the two configurations while Figure 7.8 depicts the number of reference frames loaded on each federate for each configuration, including the body frames.

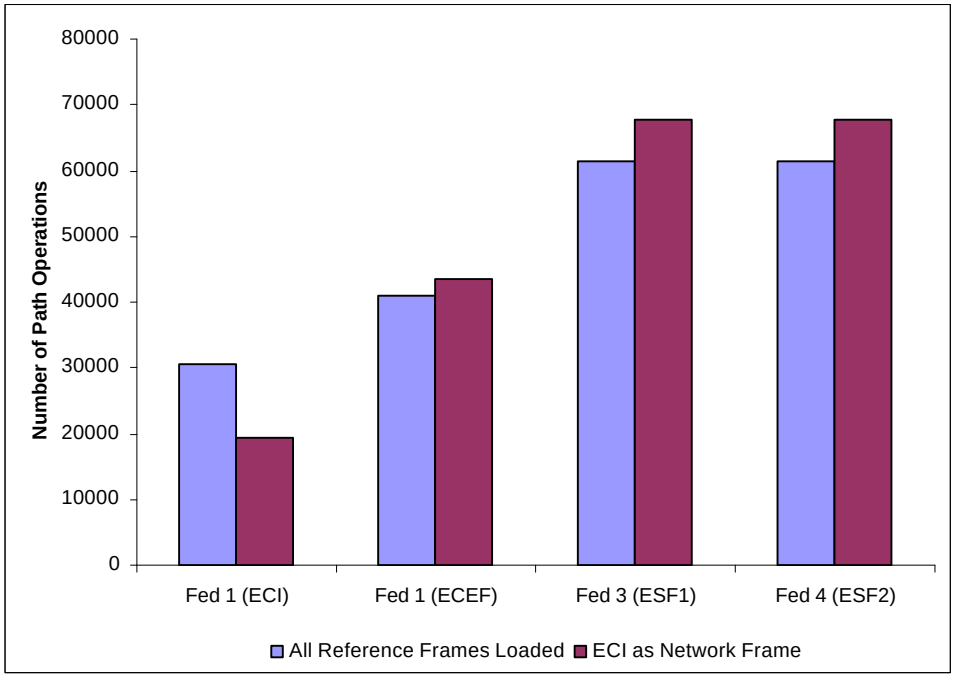


Figure 7.7: Number of Path Operations for the Two Configurations

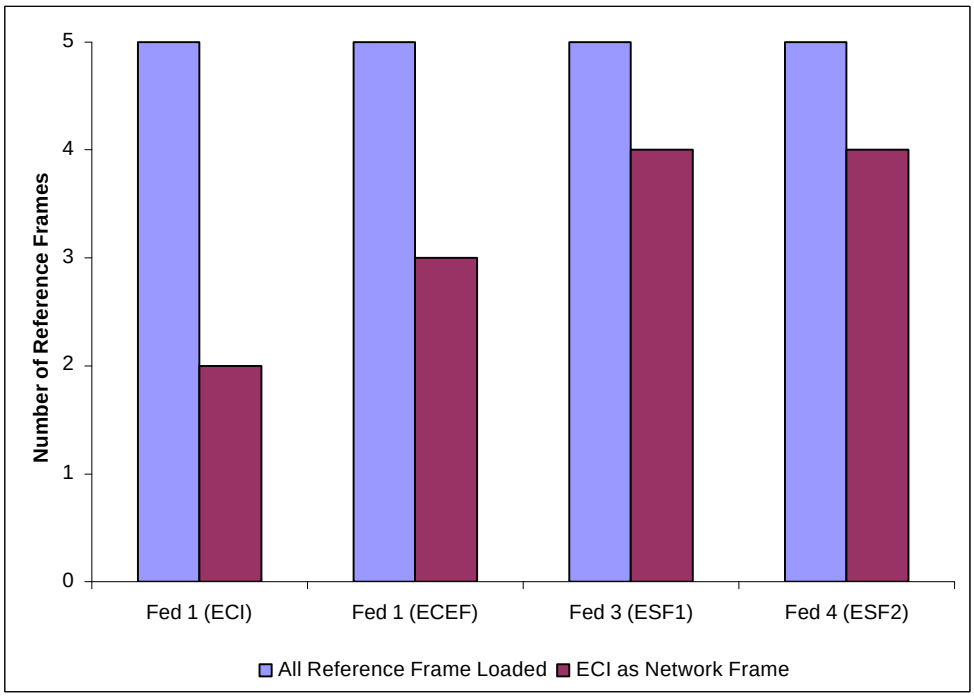


Figure 7.8: Number of Reference Frames on Each Federate

It was observed that the configuration where all reference frames were loaded required approximately 50% more reference frames in the federation while the configuration utilizing a network frame required a total of approximately 2% more path operations over all federates for the scenario used in this demonstration. The effect of using a network frame on number of path operations for individual federates is described below in greater detail.

The number of path operations depends upon the reference frames used by the dynamic model, the RVM and the display in each federate. The number of path operations required by the satellite model on each federate was independent of the two configurations and was solely dependent on the path length between the model's navigation frame and inertial frame. The number of path operations required by the RVM was dependent on the configuration. If the network frame was not required, the RVM

would not require any path operations; otherwise the number of operations would depend upon the path length between the network frame and the navigation frame used by its dynamic model. Therefore, the use of a network frame would increase the number of path operations required by each RVM. The number of path operations required by the display was dependent upon navigation frame of the dynamic model and RVAs as well as its own display frame. Therefore, the effect of using a network frame on the number of path operations required by the display depended upon the actual navigation frames of the RVAs and the display frame. In the case of federate 1, using the ECI frame as the network frame significantly reduced the number of operations required since the ECI frame was also the display frame, minimizing the path lengths between the RVAs and the display frame. In contrast, the number of path operations required by the other federates was increased. Therefore, the choice of network frame can affect the computational load on each federate. The operation of each component needs to be examined to determine the effect of the network frame on its computational load. Consequently, comparing the computational costs of the two configurations described in Chapter 6 requires a careful examination of the reference frame used as well as their frequency of use by each component within the federation.

7.3.3 Demonstration 3 Results: Intermediate Frame

Demonstration 3 was used to verify the operation of the intermediate frame and its ability to bound the magnitude of a vehicle's motion states. It was also used to examine the effectiveness of intermediate frames in reducing roundoff error. The effect of different values for critical levels on error reduction was also examined, along with the effect of time step size on the adaptive critical level.

The operation of the intermediate frame was verified by comparing the states of a satellite in an elliptical orbit with respect to its navigation frame and its intermediate frame. The magnitude of the satellite's position vector with respect to both reference frames is depicted in Figure 7.9. It can be seen that the magnitude of the position vector with respect to the intermediate frame was several magnitudes smaller than its magnitude with respect to the navigation frame. The sudden changes are due to the update of the vehicle's motion states whenever the critical levels of the intermediate frame are exceeded. Thus, the ability of the intermediate frame to bound the motion states of the dynamic model through updates based on critical levels was verified.

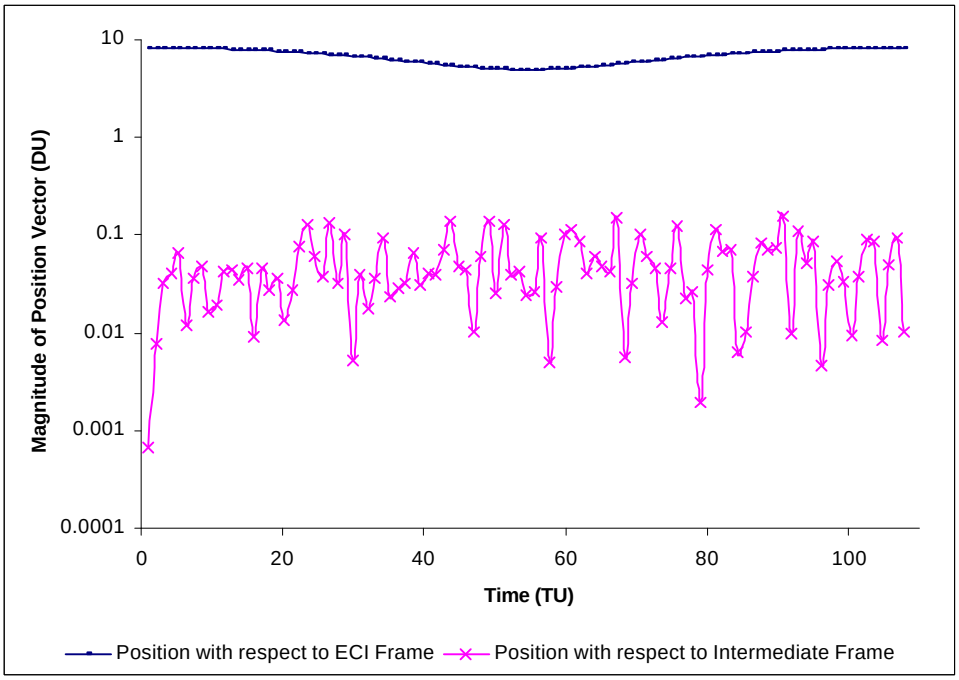


Figure 7.9: Magnitude of Position Vector for Eccentricity of 0.25

The effectiveness of intermediate frames in reducing roundoff error for different time steps was analyzed by comparing the actual error and the theoretical limit on global

error of satellites using intermediate frames with 'control' satellites that did not use intermediate frames. The theoretical error limits included the truncation error calculated by the Runge-Kutta-Cash-Karp integration routine and roundoff error, estimated by the methods described in Chapters 2 and 4. A one-tailed statistical paired t-test was carried out to check if the use of intermediate frames significantly reduced errors. To ensure a paired test, the same seed was used to generate the random numbers for each condition. Since 100 satellites were simulated, the test statistic was 2.369 (α of 0.01). The null hypothesis, expressed by equation (7.7) states that the error in a model using an intermediate frame is the same the error in the control model. The alternate hypothesis, expressed by equation (7.8), states that the error in the model using an intermediate frame is less than the error in the control model. The dependent variable, *d*, was the difference in position error, ΔP, between the models using intermediate frames and the corresponding control models. The test statistic used is expressed by equation (7.11).

$$H_0 : \Delta P_{IF} - \Delta P_{No\ IF} = 0 \tag{7.7}$$

$$H_1 : \Delta P_{IF} - \Delta P_{No\ IF} < 0 \tag{7.8}$$

$$d = \Delta P_{IF} - \Delta P_{No\ IF} \tag{7.9}$$

$$S_D = \sqrt{\sum_{i=1}^{100} \left(d_i - \bar{d}\right)^2} \tag{7.10}$$

$$t_0 = \frac{d}{S_D / \sqrt{100}} \tag{7.11}$$

Since the alternate hypothesis is less than zero, the null hypothesis is rejected if the test statistic is less than –2.369. The test statistic for actual error and theoretical error limit was calculated for the conditions with adaptive critical levels. The test statistic for actual error is tabulated in Table 7.12 while the test statistic for error limit is tabulated in Table 7.13.

Table 7.12 Test Statistic for Actual Error

	e = 0.00	e = 0.25	e = 0.60	e = 0.85
$\Delta t = 1.07$	-4.43	-4.42	1.57	-2.34
$\Delta t = 1.07\times10^{-1}$	-12.88	-9.90	-11.18	-7.58
$\Delta t = 1.07\times10^{-2}$	-13.70	-13.24	-12.05	-10.35
$\Delta t = 1.07\times10^{-3}$	-9.83	-12.04	-10.23	-11.44
$\Delta t = 1.07\times10^{-4}$	-13.10	-16.25	-13.38	-10.93
Adaptive Step	-5.73	-5.12	-6.46	-7.01

Table 7.13 Test Statistic for Theoretical Error Limit

	e = 0.00	e = 0.25	e = 0.60	e = 0.85
$\Delta t = 1.07$	-730	-332	-403	-242
$\Delta t = 1.07\times10^{-1}$	-4922	-329	-430	-468
$\Delta t = 1.07\times10^{-2}$	-18164	-314	-442	-495
$\Delta t = 1.07\times10^{-3}$	-112559	-312	-434	-491
$\Delta t = 1.07\times10^{-4}$	-196006	-310	-431	-491
Adaptive Step	-444	-225	-347	-157

From these tables, it can be observed that the null hypothesis is rejected for both the actual error and theoretical error limit in all conditions except for the actual error when the eccentricity is 0.6 and 0.85 for the fixed time step of 1.07 TU. The rejection of

the null hypothesis at those eccentricities with the use of adaptive time steps suggests that this was probably due to size of the time step at perigee, as highly eccentric orbits require small time steps at perigee. Therefore, the ability of the intermediate frames to reduce roundoff error for a variety of conditions is verified.

The effectiveness of the intermediate frames in reducing roundoff error was evaluated by comparing the mean actual error and mean theoretical error limit of satellites using intermediate frames with control models that did not use intermediate frames, as illustrated in Figure 7.10. The effectiveness of the intermediate frame improves at smaller time steps where the roundoff error increases in the control model. At the smallest time step, the numerical error of the model using an intermediate frame was two orders of magnitude less than the numerical error in the control model.

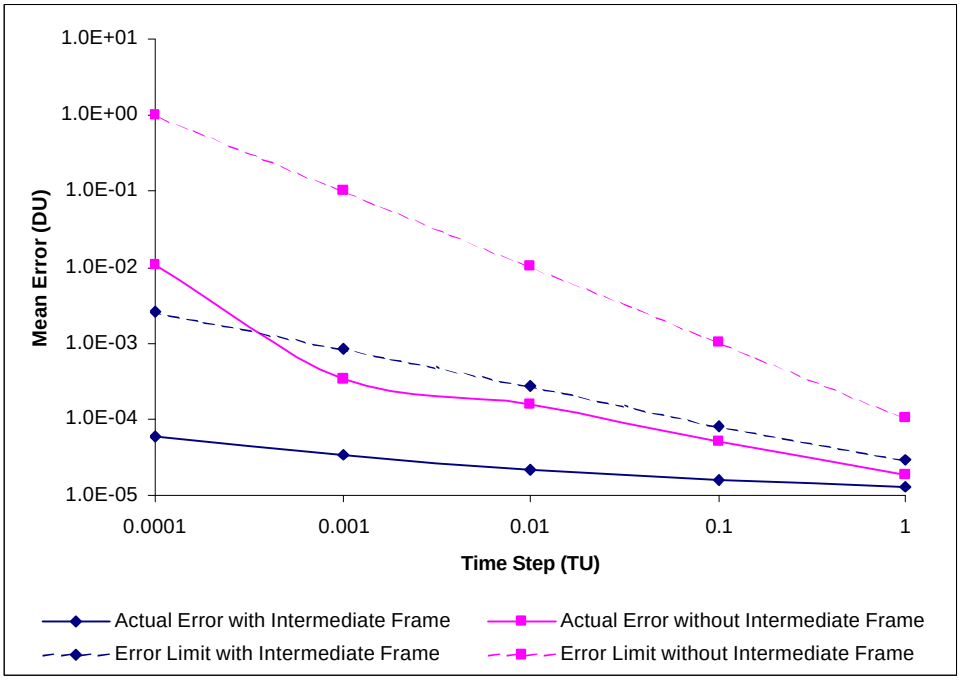


Figure 7.10: Position Error for Different Time Steps

The importance of selecting an appropriate critical level was demonstrated by comparing the error reduction achieved using the adaptive critical level described in Chapter 4 and four fixed critical levels for velocity. The mean actual error using the adaptive time step was compared with the mean actual errors using these fixed critical levels as well as the errors from the corresponding control model. The eccentricity of the orbit was restricted to 0.0 and only fixed time steps were used.

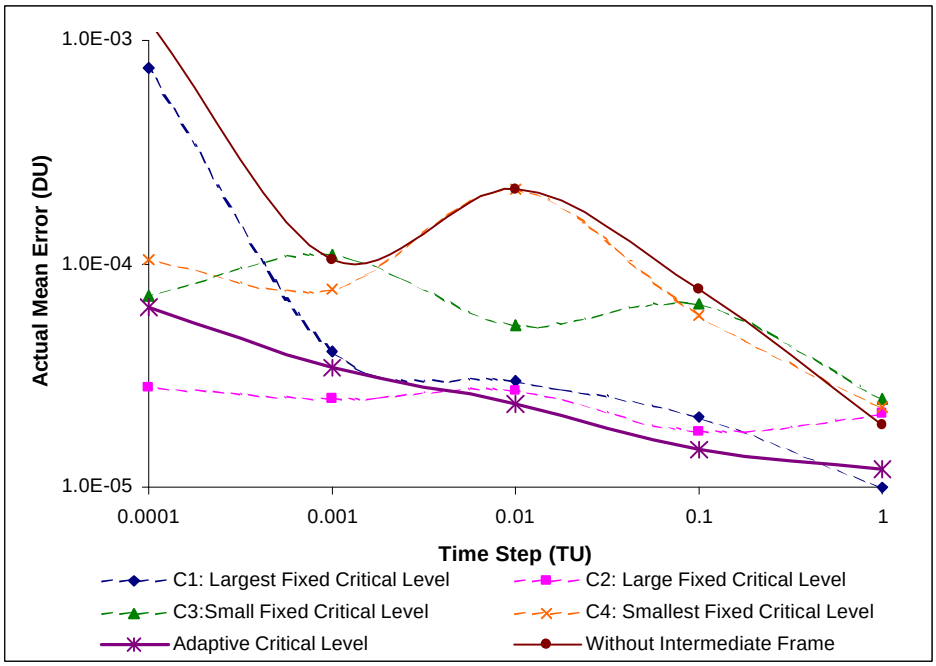


Figure 7.11: Effect of Different Critical Levels on Error

Figure 7.11 illustrates the need to adapt the critical level to time step, as the larger critical levels are effective at large time steps but rapidly lose their effectiveness at smaller time steps. Conversely, small critical levels are effective at small time steps but ineffective at larger time steps. The adaptive critical level, on the other hand, varies its

magnitude based on the time step used by the simulation scenario, as illustrated in Figure 7.12, enabling it to remain effective over a large range of time steps.

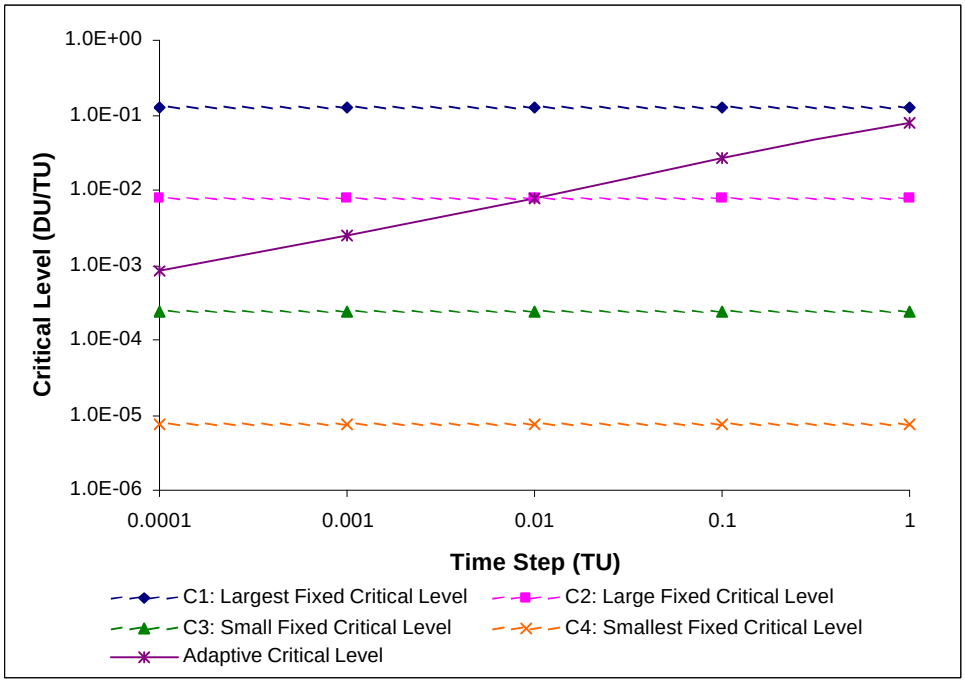


Figure 7.12: Effect of Time Step on Adaptive Critical Level

The computational cost of intermediate frames was estimated by examining the number of path operations and the runtime. It was observed that, while the control model required 6 path operations per time step, corresponding to the number of derivative calls in the RKCK integration routine, the model using intermediate frames required 20 path operations per time step. Use of the intermediate frame increased the path length by 1. Since the transformations path was called twice per derivative call, the first during the calculation of forces in the UDS and the second during the calculation of kinematics by the GDM, an additional 12 path operations were used in the 6 derivative calls required by the RKCK routine per time step. The dynamic model also published its motion states

with respect to the navigation frame, adding an additional 2 path operations during the time step. While the number of path operations with the use of intermediate was approximately 2.3 times larger than the number required by the control model, the effective computational cost was much smaller and was estimated by comparing the runtimes of the different conditions, as depicted in Figure 7.13. The runtimes of configurations using intermediate frames was about 60% longer than the configurations using the control model.

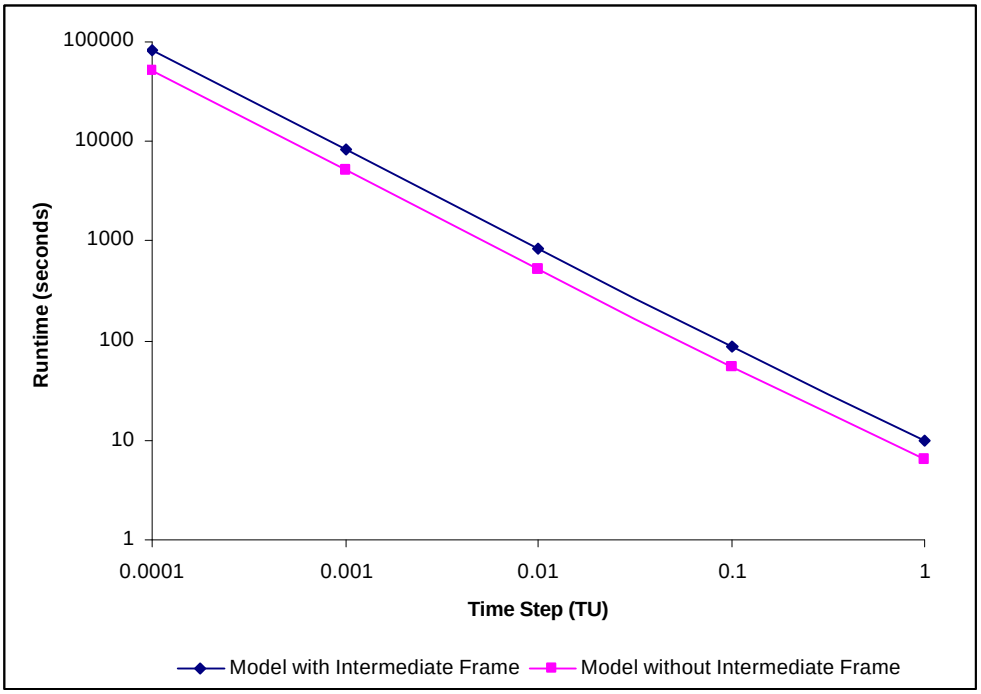


Figure 7.13: Computational Cost of Intermediate Frames

CHAPTER 8

CONCLUSION

8.1 Summary

Reference frame definitions are usually considered intrinsic to the dynamic model in most aerospace simulations due to modeling decisions early during their development. This thesis has treated reference frames as unique entities that can be defined with respect to other reference frames. The focus of this thesis was to develop a mechanism that allows a network of reference frames to be formed, enabling the calculation of the kinematics and rotations between any pair of reference frames within the network. This mechanism also enables dynamic models to be viewed as a combination of generic properties common to all models and unique properties that give dynamic models their unique behavior. When implemented in a simulation environment, this mechanism allows a simulation component to express the motion parameters of other components in its preferred reference frame. Dynamic models are able select their reference frames from the simulation environment. Furthermore, reference frames can also be created to facilitate the reduction of roundoff error. Parallel and distributed simulations (PDS) also benefit from this mechanism, as the individual components within each federate can use their preferred reference frames.

The first three objectives of this research were met by defining reference frames using their motion parameters, allowing them to be treated as nodes in a modifiable network assembled at runtime in the simulation environment by the Reference Frame Manager. Specifically, reference frames were conceptually treated as entities whose

motion was defined with respect to other reference frames. Each reference frame was defined with respect to a single definition frame through its motion parameters. Definition with respect to a single definition frame ensures that a network of reference frames provides a unique path linking any pair of arbitrary reference frames in network, ensuring consistent kinematics and rotations. This definition of reference frames allows them to be treated as nodes in a network of reference frames. These nodes are linked through their motion parameters. Standard operations were defined for the network to facilitate the manipulation of nodes, enabling an extensible network of reference frames to be maintained. When implemented in a simulation environment, the properties of each node and its operations within the network facilitate the definition of reference frames as unique entities with standard set of properties and interfaces. The standard interfaces of the reference frames and the network allow dynamic models to access all the reference frames in the simulation.

Once a network of reference frames is formed, the paths linking reference frames are identified. Since the network uses a tree topology, a search algorithm that uses the levels in the tree identifies the path linking the nodes. The network uses these paths to assemble the kinematics and rotations between reference frames. When implemented in a simulation environment, the Reference Frame Manager uses these algorithms to assemble the kinematics and rotations upon request by dynamic models and other simulation components. The conceptual development and implementation of these algorithms satisfied the fourth objective of this research.

The next two objectives of this research, the reduction of roundoff error through reference frames, were achieved through the development of intermediate frames. These

intermediate frames act as surrogates of the navigation frame and are defined so that the motion states of the vehicle with respect to the intermediate frame are bounded. If the vehicle's motion states exceed specific critical values, the motion parameters of the intermediate frame and motion states of the vehicle are updated. These critical values for velocity can adapt to the behavior of the dynamic model to minimize the growth of roundoff error for both position and velocity.

The next two objectives, dealing with the representation of dynamic models, were met through the development of generic dynamic models and the identification of elements unique to specific vehicles. Specifically, the paradigm required by dynamic models to access reference frames through the network of reference frames encouraged the development of a generic dynamic component that encapsulates the common elements within dynamic models. This generic dynamics component provides the numerical integration routines, a standard representation of kinetics, kinematics and rotations between reference frames through the network of reference frames. The unique elements provide the subsystem models as well as the forces, moments and inertia properties required by the kinetics. These unique elements can be added to the generic dynamic model to assemble a model of the vehicle.

The final objective was met by developing the data passing protocols required to use a network of reference frames in PDS. These protocols primarily require dead reckoning models to either publish the identity of their reference frame or use the reference frame management mechanism to express their motion parameters in a common reference frame, which can be set at runtime. The reference frame management mechanism enabled several configurations to be developed regarding the use of reference

frames in PDS. Two configurations in particular provide a trade off between the number of reference frames required in each federate and the computational load of handling kinematics and rotations between reference frames on each federate.

The standard interfaces for the reference frame management system, intermediate frames and generic dynamic models were implemented as an interface library. This was then instantiated as the Reference Frame Manager using the object-oriented approach in the Reconfigurable Flight Simulator, which provides the simulation environment for the RFM. The RFM is used to create a network of reference frames encouraging the creation of different reference frame objects in RFS, which can then be added by the RFM to its network of reference frames.

The architecture of RFS, specifically, the design of the Environment Controller and Database Object (ECAD) allows the RFM to be added to the simulation environment and enables any simulation component to access the RFM and its network of reference frames. The RFM provides a common interface where dynamic models and simulation components can request the kinematics and rotations between the reference frames in the network.

The RFM can also create intermediate frames upon request through an Intermediate Frame Manager, which initializes the intermediate frames and assigns them to specific vehicles upon request. The RFM updates its network so that the vehicle's body frame uses the intermediate frame as its definition frame, which in turn defines its motion parameters with respect to the vehicle's original navigation frame.

The generic dynamic model was implemented in RFS as the GDM, which also instantiated the standard interfaces of the Base Airplane Object. The GDM implements

the functionality common to all 6DOF dynamic models, including integration routines with adaptive time steps, kinematics, kinetics and rotations that obtain the motion parameters of reference frames from the RFM. It also maintains a list of UDCs that provide the forces, moments, inertia properties and subsystem dynamics unique to specific vehicles.

The HLA Networking Object in RFS was modified to utilize the RFM, allowing it to express the motion states of RVA and RVM objects with respect to any network frame set at runtime. It is also able to publish the identity of the navigation frames used by dynamic models and maintains the motion states of the RVA and RVM objects with respect to the corresponding navigation frames.

The RFM, GDM, intermediate frames and the modifications to the HLA Networking Object were verified though a series of demonstrations simulating the trajectories of satellites through a series of Monte Carlo simulations. The results of these simulation runs showed that intermediate frames significantly reduce roundoff error and are especially effective with small time steps. Thus, it is possible to manage total error without having to sacrifice roundoff error or truncation error.

8.2 Contributions of Work

1. This thesis has shown that a significant portion of kinematics of rigid bodies can be captured by the simulation environment through a network of reference frames. Expressing the motion of each reference frame with respect to other reference frames allows them to be treated as nodes in a network of reference frames and enables the calculation of kinematics and rotations between any pair of reference frames within

the network. Standard network operations facilitate the growth and modification of the network through the addition and removal of reference frames.

2. The implementation of this network in the simulation environment ensures that simulation components can interact with each other regardless the reference frames used to express motion, encouraging the reconfiguration and reuse of simulation components in different scenarios and simulation, reducing the time, effort and cost of developing simulation components for a large variety of applications.
3. This thesis has also shown that a dynamic model may be viewed as a combination of generic elements and elements unique to specific vehicle models. Elements common to all dynamic models can be encapsulated in generic model. The generic model is able to integrate state and derivative vectors of arbitrary size and automate the kinetics and kinematics of the model using the reference frame management mechanism, which also handles transformations between arbitrary pairs of reference frames. The unique elements of the model provide the forces, moments, inertia properties and subsystem dynamics. The generic model is able to assemble the final model when given vehicle specific UDCs, encouraging software reuse and reducing development effort, time and cost.
4. The roundoff error can be reduced without adversely affecting the truncation error or changing the time step. Intermediate frames can be used to reduce the roundoff error and help manage the total error in numerical integration. Critical levels control the behavior of the intermediate frames and can be adapt to the behavior of the dynamic model to minimize the growth of roundoff error. The development of a method to control roundoff error independent of time step and the control of truncation error

allows the total numerical error to be controlled, improving the accuracy of long duration simulations where both sources of numerical error need to be controlled. Scenarios where time step may be constrained by other factors, leading to unacceptably large errors, could also benefit from the use of intermediate frames.

5. This thesis has shown how a reference frame management system can be used in distributed simulation to eliminate the need to force all components to publish their motion states in a pre-determined a reference frame. Each component is able to use its preferred reference frame and the reference frame management mechanism and networking component handle any reference frame transformations that may be required in publishing the motion parameters of dynamic models. The ability for each simulation component to express motion in its preferred reference frame without the need to fix a common reference frame during the software development phase improves the interoperability of simulations and facilitates their reconfiguration to include different components and scenarios.

8.3 Future Directions

- This research has assumed that reference frames express their motion using a Cartesian coordinate system. Consequently, all kinematics and rotations assembled by the network are expressed in Cartesian coordinates. However, some reference frames may support multiple coordinate systems. Therefore, the assembly of a network of reference frames using a variety of coordinate systems and the generation of kinematics and rotations between reference frames using different coordinate systems could be examined.

- The intermediate frames in this research were restricted to the reduction of roundoff error in position and velocity. The effect of orientation and angular velocity on roundoff error can be studied and may allow the intermediate frames to be expanded to include orientation and angular velocity for reducing roundoff error. Similarly, the representation of states in simulation can be examined and methods to reduce roundoff error in non-motion states could be developed, facilitating the reduction of numerical error in the dynamic model's entire state vector, including its subsystems.
- While the development of a generic dynamic model allows the inertial frame to be set or changed at runtime, the dynamic switching of the inertial frames based on the fidelity requirements of the dynamic models needs to be fully explored. The acceleration of the model with respect to different reference frames could be compared to identify an inertial or Newtonian reference frame providing sufficient model fidelity for a particular scenario and application[38]. For example, the acceleration of a vehicle with respect to its inertial frame's parent and child frames could be used to regulate the dynamic switching of inertial frames.
- While two configurations for distributing reference frames in a PDS were demonstrated, methods to measure the benefits of each configuration were not available. Developing metrics to measure the performance of these configurations would allow the appropriate configuration to be selected for different scenarios. In particular, measures of configuration complexity due to the coordination and identification of reference frames required on each federate need to be developed.

APPENDIX A

THE RECONFIGURABLE FLIGHT SIMULATOR

A.1 The Reconfigurable Flight Simulator Architecture

The Reconfigurable Flight Simulator (RFS) uses the object oriented programming approach to implement a simulation environment that can be easily expanded and modified to suit the needs of the user. The object oriented programming approach facilitates code reuse and encourages a modular design to create greater flexibility and simplicity in developing components for the simulation. The primary RFS application sets up the base command line interface and default objects that are used to manage the simulation. The vehicle modules, controllers and displays are loaded from dynamic link libraries through the command line interface or script files. Thus, a large variety of modules and simulation configurations can be loaded as required. Figure A.1 illustrates the concept of using different aircraft and display modules with RFS to run the required simulation scenario.

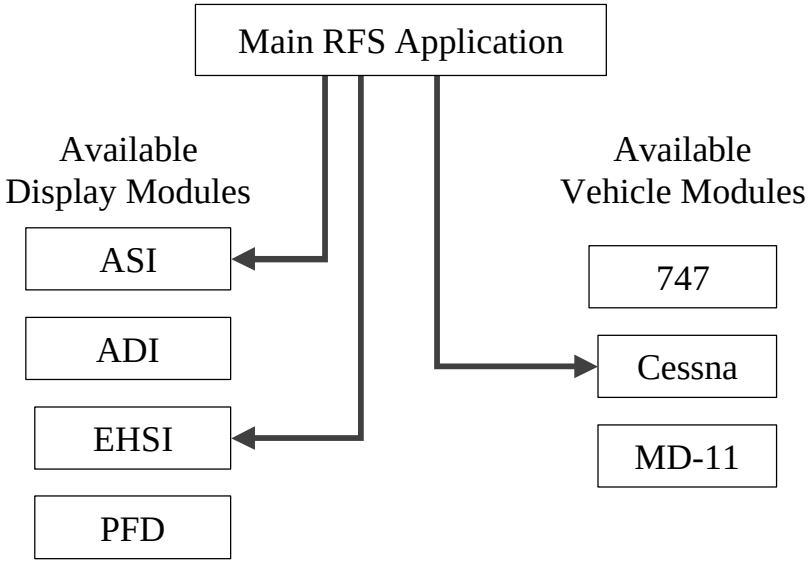


Figure A.1: Modular Architecture of RFS

The primary components of the simulation are the Simulation Object, the Event Dispatcher, the Master Simulation Controller, the Environment Controller and Database (ECAD) Object, the Timer Object, the I/O List, the Vehicle List and the Controller, Events and Measurement (CEM) Objects List as illustrated in Figure A.2. Additional objects include the I/O Objects, the Vehicle Objects and the CEM Objects, which are added to their respective lists. Each of these objects is created through a Library Manager and the Library/Factory Objects.

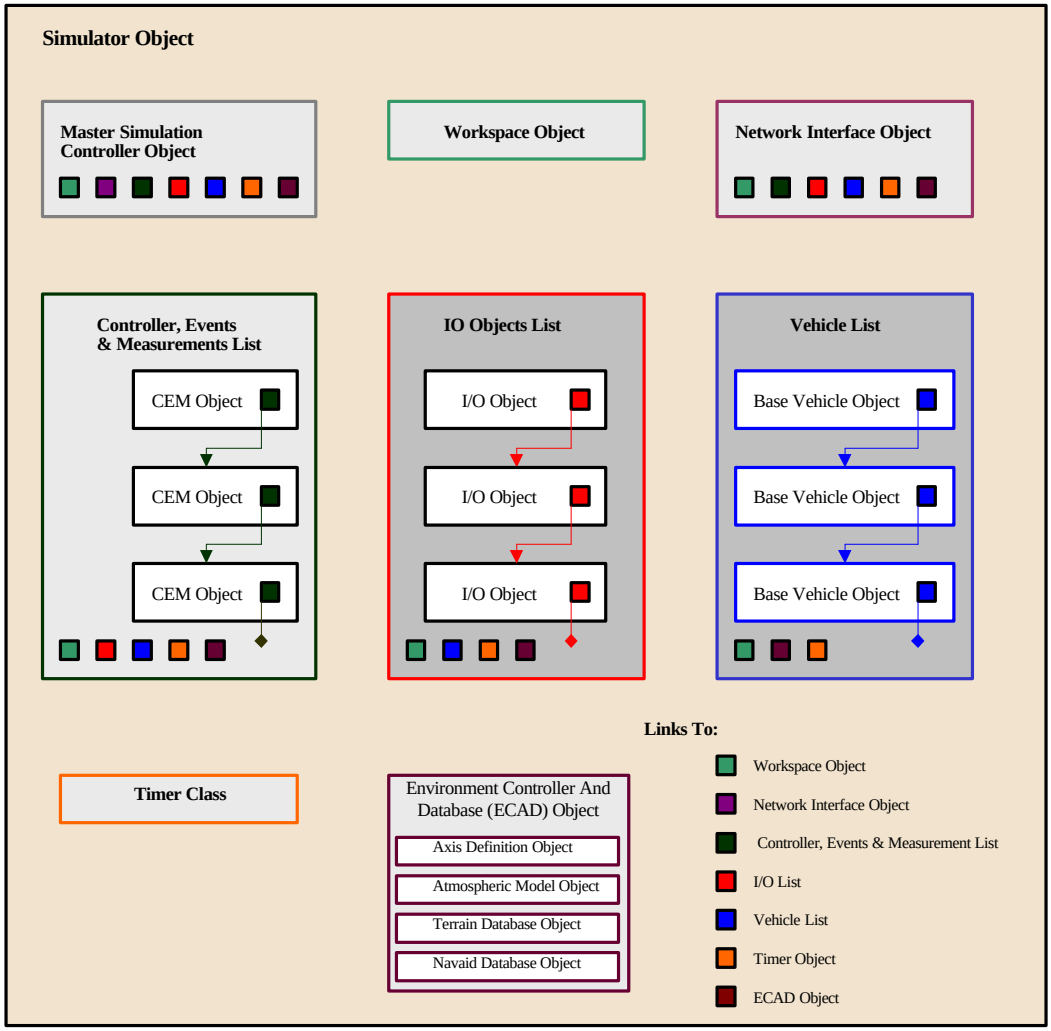


Figure A.2: RFS Component Interaction

The Simulation Object and Event Dispatcher are responsible for setting up the simulation and linking the different modules. The Simulation Object is the main object in RFS and is responsible for maintaining the various modules in the simulation. The Event Dispatcher is used as a communications device to pass event messages to all the objects in the simulation such as the addition and removal of any object from the simulation.

The Master Simulation Controller and the Timer Object are responsible for the timing issues of the simulation. The Master Simulation Controller is responsible for passing the time steps to the individual list objects. The Timer Object is responsible for generating the timing for the simulation. Both the Master Simulation Controller and the Timer Object can be overridden and replaced if required.

The ECAD object is responsible for setting up the simulation environment. This object is composed of four objects: the Axis Definition Object, the Terrain Object Database, the Atmospheric Model and the Navigation Database. Each of these components can be overridden if necessary.

The Input/Output or I/O Objects provide the interfaces for the development of displays and controls for user interaction. Thus, common I/O Objects are the cockpit displays and 'Out the Window' displays for aircraft simulation, virtual control stick modules and hardware interface modules for flight yokes as well as other visualization tools such as graphing displays. The I/O Objects are managed using the I/O List Object. The I/O List adds the I/O Object to the simulation loop and calls the I/O Object at every time step. The I/O List can be overridden if so desired.

The Vehicle Objects contain the dynamic models for the vehicles being simulated. These vehicles can include ground vehicles, aircraft or spacecraft. A derived class from

the Base Vehicle class is the Base Airplane class. The Vehicle List is responsible for adding the Vehicle Objects to the simulation and calling the Vehicle Objects at every time step.

The CEM is a very powerful class that has access to most of the objects in the simulation. It is primarily used for controlling vehicle objects, for generating events in a discrete time simulation and for creating measurement objects. The CEM List is used to maintain the CEM objects and calls them at every time step.

A.2 Types of Interfaces in RFS

There are 2 major types of interfaces that are available to objects within RFS. These interfaces are used to facilitate interaction between different objects. The first type is the standard interface provided by the base classes stored in the Simulation Foundation Class (SFC) Library. The second type is an extensible interface, called the Object Data/Method Extension or OD/ME interface that represents methods and data variables using character strings.

The standard interfaces in the base classes address most of the functionality that is needed for each class. However, the derived classes may require additional methods that are not defined in the base class. Furthermore, different classes may need to interact with each other using these additional methods. If the concept of a standard interface were used, the header files of the derived classes would need to be included in the header files of the classes using the interface. This would create a dependency between the classes. A change in one of the classes would require all dependant classes to be recompiled with the updated header file of the changed class.

The OD/ME interface reads the methods and data variables as text strings and passes these strings from the calling object to the called object. These strings are then reconstructed through the interface and the method is executed or the data is passed back to the calling object. This interface provides tremendous flexibility in terms of adding new interfaces and variables. The only drawback is that this interface computationally more expensive than the standard interface by approximately an order of magnitude.

The SFC Library is built upon the OD/ME interface so that all the classes that use the SFC Library have the ability to use the OD/ME interface. A command line interpreter is also built upon the OD/ME interface and acts as the portal for the user to interact with the simulator. The command line interpreter can be used to load script files, libraries and even call the OD/ME methods of an object within the simulator.

A.3 The Environment Controller and Database or ECAD

The Environment Controller and Database or ECAD object is responsible for setting up the simulation environment. This object is composed of four components that control different aspects of the environment and can be overridden if necessary. This allows the simulation environment to be tailored to the simulation requirements. The simulation environment consists of the axis definition used by the vehicles, the weather model as well as terrain and navigational data.

The Axis Definition Object is responsible for maintaining the reference frames used by the vehicle objects. This object is better than the typical implementation of reference frames in simulation in that the object provides the vehicles with a limited choice of reference frames and their transformations. The default implementation includes a Cartesian *flat earth surface frame* and an *earth centered earth fixed frame*

using latitude, longitude and altitude. Despite the limited frames available, this approach does provide a certain degree of independence between the vehicle models and reference frames. This object can be overridden to handle an arbitrary number of reference frames as well as their kinematics and rotations.

The other components of the ECAD object are the Atmospheric Model, the Terrain Object Database and the Navigation Database. The Atmospheric Model contains the method declarations to allow a rudimentary atmospheric model to be implemented. The current implementation models basic properties such as atmospheric temperature and density. The Terrain Object Database is currently an empty class that can be overridden to provide terrain information. The Navigation Database provides information on navigation aids such as VOR, TACAN and ADF transmitters. It can be overridden as required to include the navigation aids for different regions.

APPENDIX B

CLASS DESCRIPTIONS OF THE REFERENCE FRAME MANAGER

B.1 Base Classes and Standard Interfaces for Managing Reference Frames

The base classes used to define the basic attributes and interface standards for reference frames, a reference frame management mechanism, intermediate frames and generic dynamic models are compiled into a Frame Definition and Management library that can be accessed by dynamic models and other simulation components. This library ensures that access to the RFM and its services is independent of its implementation of. The library defines the following base classes with their associated interfaces:

1. Frame Definition Class
2. Frame Manager Interface Class
3. Intermediate Frame Interface Class
4. Generic Dynamics Interface Class
5. Model Component Interface Class

B.1.1 Frame Definition Class

The Frame Definition class inherits from the ECAD Module Interface class and is the base class used to determine the interface standards, basic attributes and functionality of reference frames implemented in RFS. The primary responsibility of the Frame Definition class is to maintain the motion parameters of the reference frame. The basic kinematics and integration routine can be generalized to all reference frames and is implemented in the base class, while the methods to calculate the derivatives of velocity and angular velocity are unique to specific reference frames and are consequently represented by *pure virtual functions* that need to be implemented in the derived class of each reference frame.

The rest of the interfaces and functionality are implemented to create a base class that can be rapidly developed into any reference frame required by the simulation environment. The ECAD Module Interface provides access to the OD/ME interface, allowing reference frames to customize their interfaces to different simulation requirements. The standard interfaces and functionality implemented in the Frame Definition Class are listed in Table B.1.

Table B.1: Standard Interfaces and Functionality for Frame Definition

Standard Interfaces Implemented in the Derived Class *(Pure Virtual Functions)*	– Calculate the derivatives for velocity and angular velocity
Standard Interfaces Implemented in the Base Class	– Obtain and set the identity of the reference frame and its definition frame – Obtain and set the level of the node in an enumerated tree – Add and remove child nodes – Access or initialize motion parameters – Access the derivatives for velocity and angular velocity – Update motion parameters due to a discrete change in the definition frame – Access and override default propagation of motion parameters
Implementation of Internal Functionality	– Calculate kinematics with respect to the definition frame – RK4 integration routine using the simulation's time step

All the standard interfaces can be overridden if required. In particular, the default method of propagating motion parameters is applicable for body frame since the dynamic

model typically handles this functionality. Thus, the dynamic model updates the motion parameters of the body frame at the end of its time step.

B.1.2 Frame Manager Interface Class

The Frame Manager Interface Class defines the standard interfaces of the reference frame manager available to simulation components. The Frame Manager Interface Class inherits from the Axis Definition Object in RFS and has all the standard methods and variables of the Axis Definition Object, ensuring that the Reference Frame Manager does not adversely affect simulations components that do not utilize it.

The standard interfaces defined by the Frame Manager Interface Class, listed in Table B.2, include methods that allow reference frames to be added or removed from the network, methods to access existing reference frames or request intermediate frames as well as methods to calculate the kinematics and rotations between reference frames. Unlike the Frame Definition Class, all the standard interfaces of the Frame Manager Interface Class are *pure virtual functions* that need to be implemented to build a Reference Frame Manager.

Table B.2: Standard Interfaces and Functionality for Frame Manager Interface

Standard Interfaces Implemented in the Derived Class *(Pure Virtual Functions)*	– Register and remove reference frames from the network – Check if a specific reference frame is registered with the network – Get the number of reference frames registered with the network – Access any registered reference frame – Express motion parameters with respect to any registered reference frame – Change the parent node representing a reference frame's definition frame – Create, initialize and update an intermediate frame upon request
Standard Interfaces Implemented in the Base Class	– NONE
Implementation of Internal Functionality	– NONE

B.1.3 Intermediate Frame Interface Class

The Intermediate Frame Interface Class inherits from the Frame Definition Class described in Chapter 3, providing the standard interfaces and functionality of reference frames to the intermediate frame. Since the intermediate frame does not accelerate with respect to the navigation frame, the time derivatives of velocity and angular velocity are set to zero. Furthermore, the standard update method of using numerical integration is disabled, improving computational efficiency and restricting the introduction of roundoff error to errors per update.

The additional interfaces in this class deals with the initialization and update of the intermediate frame. Both interfaces use the identity of the body frame to access its motion parameters. Since these interfaces require critical levels to set and update the

motion parameters of the intermediate frame, they are represented by *pure virtual functions*, allowing their implementation to be dictated by the requirements of the simulation. These additional interfaces are listed in Table B.3.

Table B.3: Standard Interfaces and Functionality for Intermediate Frame Interface

Standard Interfaces Implemented in the Derived Class *(Pure Virtual Functions)*	– Initialize intermediate frame using a specified body frame – Update the motion parameters and critical levels of the intermediate frames if critical levels are exceeded
Standard Interfaces Implemented in the Base Class	– Set derivatives of velocity and angular velocity to zero
Implementation of Internal Functionality	– Disable default RK4 propagation of motion parameters

B.1.4 Generic Dynamics Interface Class

The Generic Dynamics Interface Class inherits from the Base Airplane Object in RFS and defines the standard interfaces and functionality required for the encapsulating the generic elements of dynamic models. These interfaces include methods to add and remove components modeling unique elements, set the identity of the navigation and inertial frames, select the appropriate numerical integration method as well as request the use of an intermediate frame to reduce roundoff error. Since these interfaces are generic to all dynamic models, they are implemented in the base class. The only *pure virtual functions* present are those inherited from the Base Airplane Object.

The basic functionality of the base class includes the assembly and initialization of the state and derivative vectors, generalized implementation of the kinetics and kinematics equations, two types of integration routines as well as the introduction and

initialization of the body frame and intermediate frame. The kinetics equations are assembled from the force, moment and inertia properties obtained from each UDC. The integration routines are the standard 4th order Runge Kutta routine and the adaptive 4th order Runge Kutta Cash Karp method with a 5th order error term to calculate the time step required to control truncation error. Since the time step is controlled by the simulation architecture, the integration routine calculates the next time step rather than repeating the current step when the truncation error exceeds the specified limit

Table B.4: Standard Interfaces and Functionality for Generic Dynamics Interface

Standard Interfaces Implemented in the Derived Class *(Pure Virtual Functions)*	– Pure virtual functions required by the Base Airplane Object
Standard Interfaces Implemented in the Base Class	– Add or remove Model Component Interface objects representing UDCs – Set the navigation and inertial frames – Select type of integration routine – Request an intermediate frame from the Reference Frame Manager
Implementation of Internal Functionality	– Allocate and initialize the Body Frame – Link the Body Frame with the Reference Frame Manager – Assemble state and derivative vectors – Assemble kinetics and inertia parameters from all UDCS – Calculate kinetics and kinematics – Propagate state vector using selected type of numerical integration

B.1.5 Model Component Interface Class

The Model Component Interface Class forms the base class for Unique Dynamics Components (UDC) and defines the interface standards for representing the subsystem dynamics, forces, moments and inertia properties of the model. The standard interfaces include methods to get pointers to the state and derivative vectors of the subsystems, methods to get the pointers to the force vectors, moment vectors exerted on the body frame and inertia properties that need to be added to the dynamic model. Pointers to these vectors and matrices are used since they can vary over time and it is computationally more efficient to access their addresses rather than execute function calls for each parameter. Other standard interfaces update these parameters and provide access to the body frame, the RFM and the Generic Dynamic Component.

The standard interfaces dealing with unique properties of the model are defined as *pure virtual functions* that must be implemented by the UDC since these parameters are specific to each component. In contrast, the methods linking the UDC to the body frame, RFM and Generic Dynamic Component interact with standardized components in the simulation and are implemented in the base class.

Table B.5: Standard Interfaces and Functionality for Model Component Interface

Standard Interfaces Implemented in the Derived Class *(Pure Virtual Functions)*	– Generate and return state and derivative array for subsystem dynamics – Get pointers to forces and moments – Get pointers to mass, mass rate and center of mass – Get pointers to moments and products of inertia and their rates – Update or calculate these parameters at the current time
Standard Interfaces Implemented in the Base Class	– Set link to Generic Dynamic Interface – Set link to Reference Frame Manager – Set link to Body Frame
Implementation of Internal Functionality	– NONE

B.2 Class Description of the Reference Frame Manager

The base classes contained in the Frame Definition and Management Library defines the base classes and standard interfaces required for creating and managing a network of reference frame. These base classes and interfaces are implemented in RFS as the Reference Frame Manager, which is loaded into the ECAD object.

The Frame Manager Interface Class and Intermediate Frame Interface Class are implemented to form the actual Reference Frame Manager. The Frame Manager Interface Class defines the standard interface for the RFM. Since the RFM creates and initializes intermediate frames, it implements the Intermediate Frame Interface Class to create intermediate frames and manage their critical levels. When implemented, the Reference Frame Manager implements two new classes that it uses to manage and utilize its

network of reference frames: the Frame Path Class and the Intermediate Frame Manager. Thus, the RFM consists of the following classes:

1. Reference Frame Manager
2. Frame Path Class
3. Intermediate Frame Class
4. Intermediate Frame Manager

B.2.1 Reference Frame Manager

The Reference Frame Manager (RFM) implements the standard interfaces of the Frame Management Interface Class as well as the functionality required to create and manage a network. Since the RFM inherits from the Frame Manager Interface Class, it replaces the Axis Definition Object in RFS.

The primary operations of the RFM include the management of a network of reference frames, the identification and generation of paths linking pairs of reference frames, the evaluation of kinematics and rotations along these paths, and the generation and maintenance of intermediate frames through an Intermediate Frame Manager. Table B.6 describes the functionality required for the implementation of each of these operations in the RFM.

Table B.6: Implementation of Functionality for Reference Frame Manager

Network Operations	– Add and remove nodes – Graft and prune of trees – Update list of root nodes (distinct trees) – Recursively set the level of each node in all trees starting from the root nodes – Check if a node is within a specific tree
Path Generation	– Check if a path is available – Check if a path is feasible – Assemble forward and reverse paths – Manage list of paths
Kinematics & rotations	– Assemble kinematics along the forward and reverse paths – Assemble rotations along the forward and reverse paths
Intermediate Frame Management	– Command creation and deletion of intermediate frames by Intermediate Frame Manager – Insert intermediate frame between the body frame and its navigation frame using network operations

The RFM creates and manages a network of reference frames using the standard network operations described in Chapter 3. These operations are typically called when reference frames are added or removed from the simulation environment or when the model of a reference frame's motion parameters is modified and needs to use a new definition frame.

The reference frames are added to the RFM through their pointers, which are maintained in a dynamic list. The network topology is represented by a list of root nodes, indicating the number distinct trees, and each reference frames links to its parent and

child nodes. When a new reference frame is added, the RFM checks if the pointer to the reference frame's definition frame is present in its dynamic list. If the pointer is available, the new reference frame is linked to its definition frame. Otherwise, the new reference frame is placed in the root nodes list. The RFM then checks if any reference frames in the root nodes list use the new reference frame is their definition frame. If so, these root nodes are removed from the root nodes list and linked to the new reference frame, effectively grafting the root node's tree to the new reference frame. When linking reference frames, a pointer to the parent node is set in the child node and the child node's pointer is added to a list of child nodes maintained by each parent node. Furthermore, the level of the child node and the nodes in its sub-tree are reset recursively. The algorithm for setting the level of a node is illustrated in Figure B.1.

The removal of reference frames from the RFM follows a similar procedure. If the node to be removed is a root node, its pointer is removed from the root node list. Otherwise, the link to its parent frame is removed. Thus, the pointer to its parent node is cleared while its own pointer is removed from the parent node's list of child nodes. If the node has any child nodes, the links to the child nodes are removed and the child nodes are placed in the root nodes list. The levels of the child nodes and their sub-trees are leveled recursively as described in Figure B1. Similarly, changing the definition frame is achieved by a series of kinematics, pruning and grafting operations. The motion parameters are expressed with respect to the new definition frame, the link to the old definition frame is removed, the reference frame is linked to the node representing the new definition frame and the levels of the node and it sub-tree are updated recursively.

```
// Recursively update the level of the node as well as its child nodes
void    FrameManager::f_updateNodeLevel(FrameDefinition* lpFrame)
{
        // Get the definition frame
        lpDefinitionFrame = lpFrame→getLpDefinitionFrame( ) ;

        // Check if the frame is a root node
        if(lpDefinitionFrame != NULL)
                // Enumeration of a frame is greater than its definition frame by 1
                nodeLevel = lpDefinitionFrame→getFrameLevel() + 1 ;
        else
                // Set enumeration of root node to zero
                nodeLevel = 0 ;

        // Assign the enumeration to the frame
        lpFrame→setFrameLevel(nodeEnumeration) ;

        // Check if there are any child nodes
        numberOfChildNodes = lpFrame→getNumberOfChildNodes() ;

        // Recursively update the child nodes
        for(counter = 0; counter < numberOfChildNodes; counter++)
        {
                lpChildNode = lpFrame→getLpChildNode(counter) ;
                f_updateNodeLevel(lpChildNode) ;
        }
}
```

Figure B.1: Algorithm to Recursively Set Node Levels in the RFM

Once a network is formed, paths linking nodes can be identified and generated using the search algorithm described in Chapter 3. Since a total of $\boldsymbol{n^2-n}$ paths can be generated, only those paths required for kinematics and rotations are generated upon request. Once generated, these paths are stored in a list of ***Frame Path Objects***. If a dynamic model or simulation component requires the kinematics or rotations between a pair of reference frames, the RFM checks if the path has already been generated and stored in its list. If the path is not found, the RFM checks if the nodes representing the

initial and final reference frames are in the same tree since a path cannot link nodes in different trees. If the nodes are in the same tree, the RFM executes the search algorithm to identify the shared node as illustrated in Figure B.2. The shared node links the forward and reverse paths allowing the path to be assembled, as described in Chapter 3, and stored in the path list. The kinematics and rotations are assembled recursively along the path using equations (3.10) to (3.21). Since these paths depend upon the network topology, when a reference frame is removed or has its definition frame changed, the RFM checks for and deletes all paths containing this reference frame.

When a dynamic model requests the use of an intermediate frame to control roundoff error, the RFM commands the Intermediate Frame Manager to generate and initialize the intermediate frame. The definition frame of the intermediate frame is set to the navigation frame of the dynamic model. The RFM then adds this intermediate frame to the network and links it to the model's navigation frame. The definition frame used by the body frame is changed using grafting and pruning operations to link the body frame to the intermediate frame. The motion parameters of the body frame are updated using kinematics and rotations between the intermediate and navigation frames.

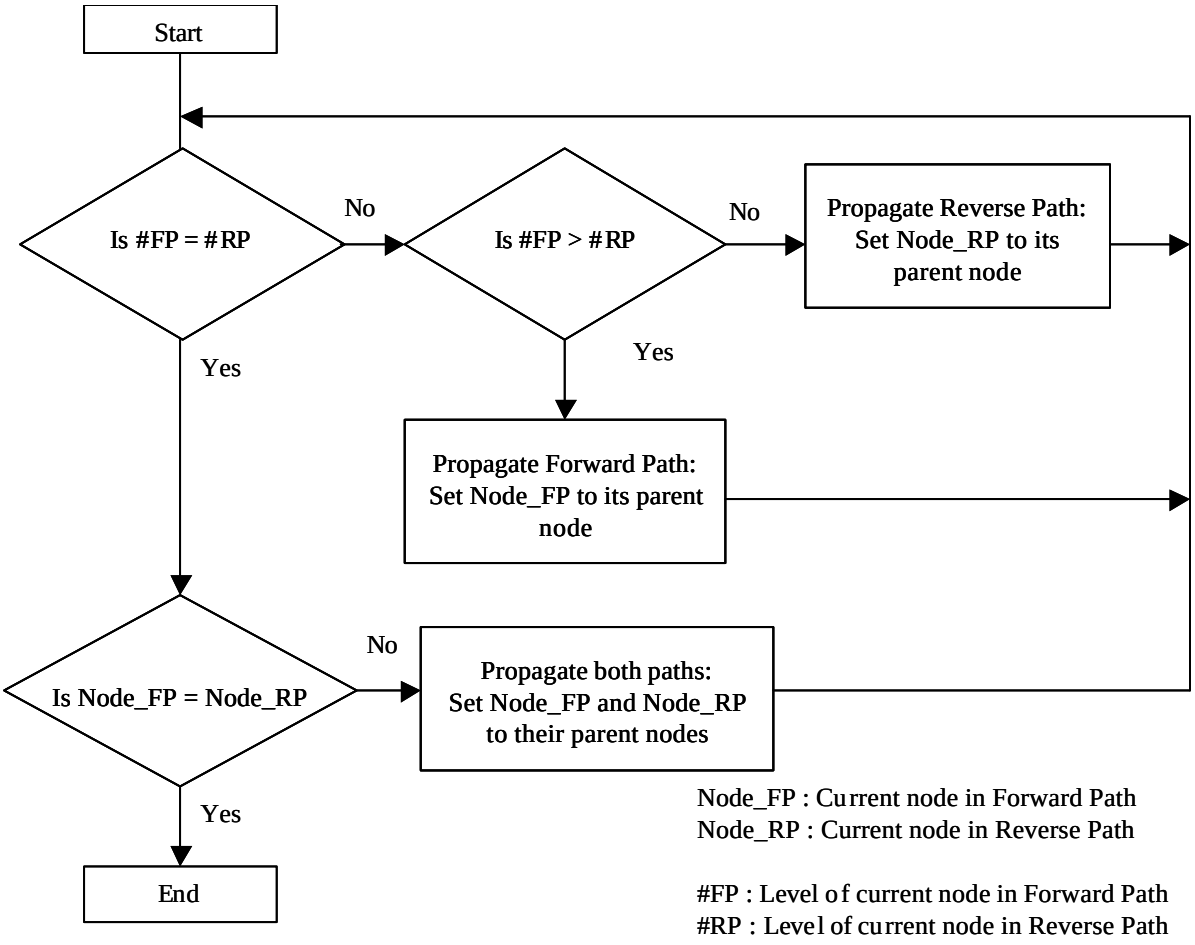


Figure B.2: Algorithm to Identify the Shared Node

B.2.2 Frame Path Class

The Frame Path object stores the sequence of nodes linking the initial node to the final node. The initial, final and shared nodes are recorded in addition to the forward and reverse paths. The forward path is the sequence of nodes linking the initial node to the shared node while the reverse path links the final node to the shared node. Each path object is identified using the initial and final frames. The shared node, as defined in Chapter 3, is the node in the path with the lowest level and can be viewed as the root

node of the sub-tree containing the path. Thus, there are two path objects for each pair of reference frames as each path is unidirectional.

The data stored in the Frame Path class is initialized and maintained by the RFM during path generation. It should be noted that the forward and reverse paths store the nodes contributing their motion parameters to the kinematics and transformation calculations. Thus, the shared node is not included in either path. This implies that either path may be empty if the initial or final node is the shared node.

In addition to these data members, the Frame Path class also contains interfaces to manage the paths and assist the RFM in checking for paths. Nodes are added to the forward and reverse paths through their pointers. Methods to check if a specified node is the initial or final node enable the RFM to determine if a path exists. Another method checks if a specified node exists within the path, either within the forward or reverse paths or as its shared node. This allows the RFM to delete paths containing nodes that are removed from the simulation.

B.2.3 Intermediate Frame Class

The Intermediate Frame Class implements the Intermediate Frame Interface Class within the RFM. The standard interfaces of the interface class are implemented to initialize its motion parameters using the motion parameters of the assigned body frame and to compare the motion parameters of the body frame with the critical levels, updating both frames if necessary. In addition to these standard interfaces, the Intermediate Frame Class also calculates and updates the critical level, adapting to the dynamics of the body frame. The critical levels are updated at every time step.

The Intermediate Frame is initialized using the body frame of the dynamic model that requested it. The intermediate frame uses the body frame's definition frame as its own definition frame. The motion parameters of the intermediate frame are then initialized to error reducing magnitudes in the vicinity of the body frame. These magnitudes are multiples of critical levels for each motion parameter. Initial critical levels can be estimated using the initial states and derivatives of the body frame. After the intermediate frame is initialized, it is linked to its definition frame and the body frame in the network by the RFM. Unlike typical navigation frames, which may support numerous body frames, the intermediate frame supports only a single body frame, which it uses as its child node.

The intermediate frame is commanded to evaluate its parameters at very time step by its child node's dynamic model. Like the body frame, the motion parameters of the intermediate frame do not use the default numerical method included in the Frame Definition class for propagation through time. In this implementation, the intermediate frame does not rotate with respect to its definition frame and matches its orientation. The velocity is only updated when its critical level is exceeded and the position per time step is calculated using the velocity and time of last update. The limits of the critical levels are also calculated and the critical levels are updated to satisfy these limits.

Once the motion parameters of the intermediate frame are updated, the motion parameters of the body frame are compared with the critical levels. If they exceed their critical levels, the motion parameters of both the intermediate frame and the body frame are updated, as described in Chapter 4. If the intermediate frame is updated or a specified number of time steps have elapsed without an update, the critical levels are recalculated

to minimize the roundoff error, as described in Chapter 4. The upper limit for roundoff error due to every update is also estimated.

B.2.4 Intermediate Frame Manager

The Intermediate Frame Manager is responsible for creating and initializing Intermediate Frames when requested by the RFM. The IFM dynamically allocates the memory required for the intermediate frame and places the pointer to the intermediate frame in a list. The intermediate frame is allowed to initialize itself based on the body frame of the dynamic model.

Since several dynamic models may require intermediate frames, each intermediate frame is given a unique name, based on the total number of frames generated during run time. Thus, the first intermediate frame generated would be named “IF0000” while the 100th frame would be named “IF0099”.

The IFM is also responsible for de-allocating and destroying the intermediate frame when it is no longer in use. When the Intermediate Frame needs to be destroyed, either when its assigned body frame is destroyed or when the simulation terminates, the Intermediate Frame Manager is responsible for destroying the object and de-allocating the memory.

APPENDIX C

DEMONSTRATION RESULTS

C.1 Results From Demonstration 1

Table C.1: Runtime for Scenarios with a Time Step of 1.06795 TU

	Display Not Used			Display Used	
	Control Only	Control & RFM	GDM & RFM	Control Only	GDM & RFM
ECI	1.438	1.453	4.562	34.297	34.500
ECEF	2.047	2.078	6.219	34.328	34.500
ESF	2.454	2.500	7.813	34.328	34.500

Table C.2: Runtime for Scenarios with a Time Step of 0.106795 TU

	Control Only	Control & RFM	GDM & RFM
ECI	6.125	6.125	35.781
ECEF	12.109	12.125	52.344
ESF	16.469	16.531	68.141

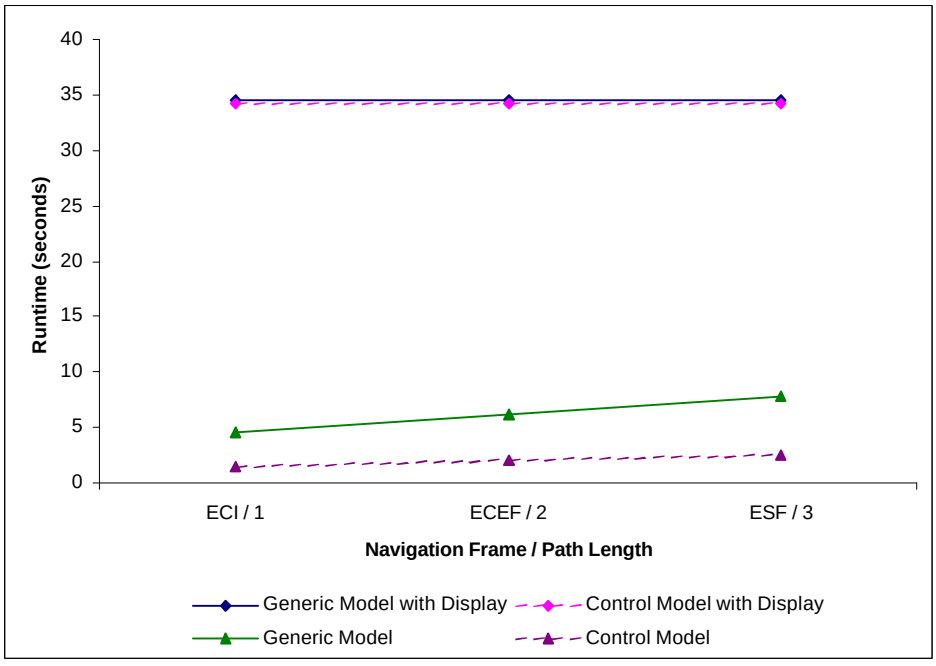


Figure C.1: Runtimes of GDM and Control Models for Time Step of 1.06795 TU

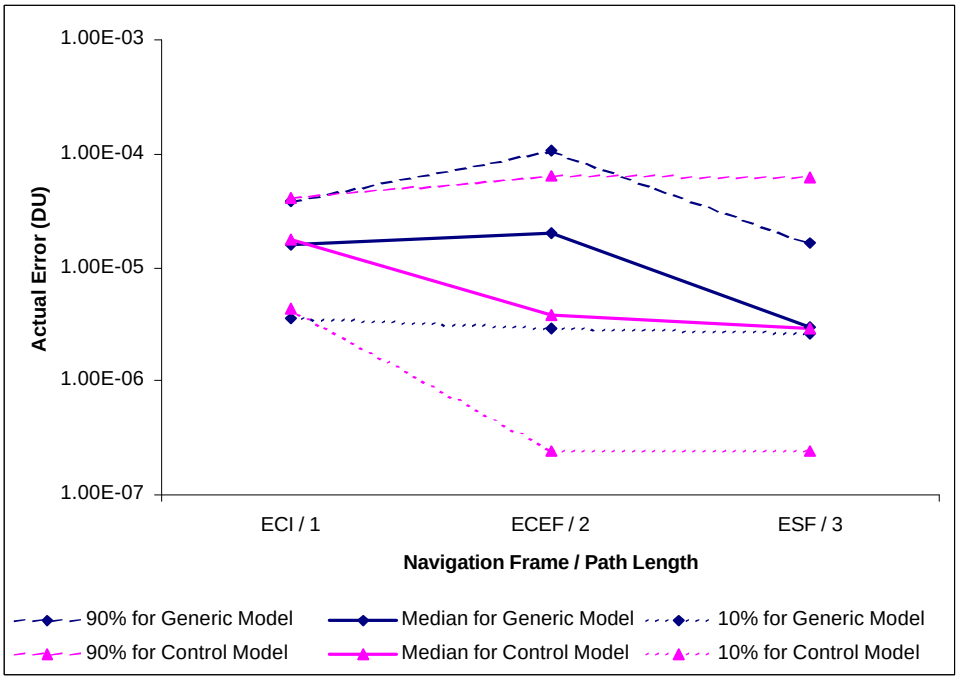


Figure C.2: 90th, Median and 10th Percentile Position Error at 1.06795 TU

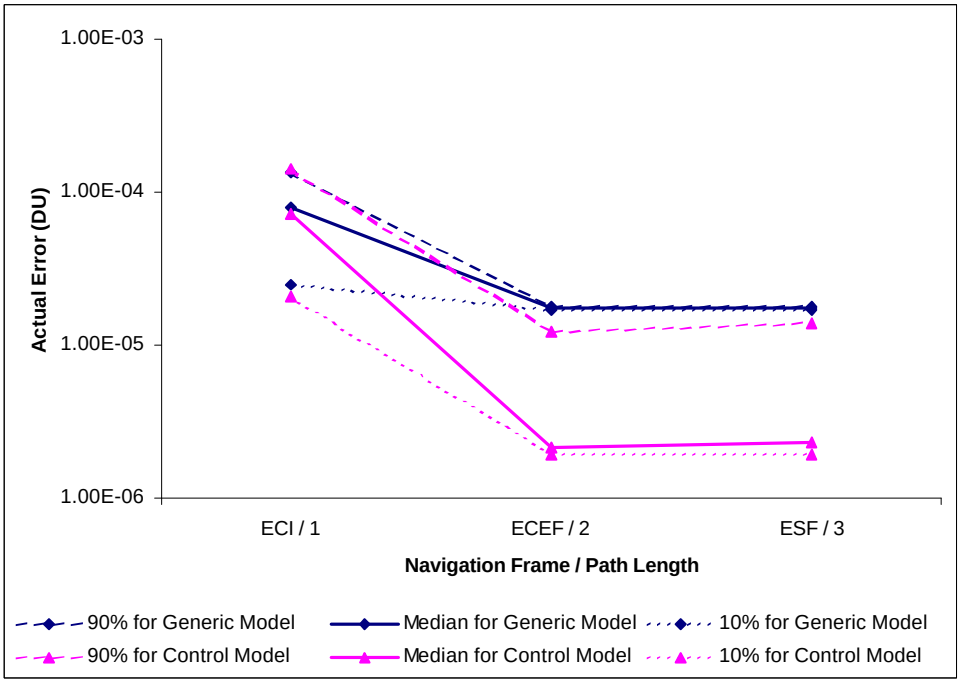


Figure C.3: 90th, Median and 10th Percentile Position Error at 0.106795 TU

C.2 Results From Demonstration 2

Table C.3: Reference Frames Loaded on Each Federate (excluding Body Frames)

	Configuration 1: All Reference Frames Loaded	Configuration 2: ECI Set As Network Frame
Federate 1	ECI, ECEF, ESF1, ESF2	ECI
Federate 2	ECI, ECEF, ESF1, ESF2	ECI, ECEF
Federate 3	ECI, ECEF, ESF1, ESF2	ECI, ECEF, ESF1
Federate 4	ECI, ECEF, ESF1, ESF2	ECI, ECEF, ESF2

Table C.4: Number of Path Operations for Configuration 1

	Forward Path Operations	Reverse Path Operations	Total Path Operations
Federate 1	30697	0	30697
Federate 2	38415	2560	40975
Federate 3	51227	10220	61447
Federate 4	51233	10220	61453
Total	**171572**	**23000**	**194572**

Table C.5: Number of Path Operations for Configuration 2

	Forward Path Operations	Reverse Path Operations	Total Path Operations
Federate 1	19236	0	19236
Federate 2	35862	7666	43528
Federate 3	52527	15344	67871
Federate 4	52521	15332	67853
Total	**160146**	**38342**	**198488**

Both configurations executed 1283 time steps.

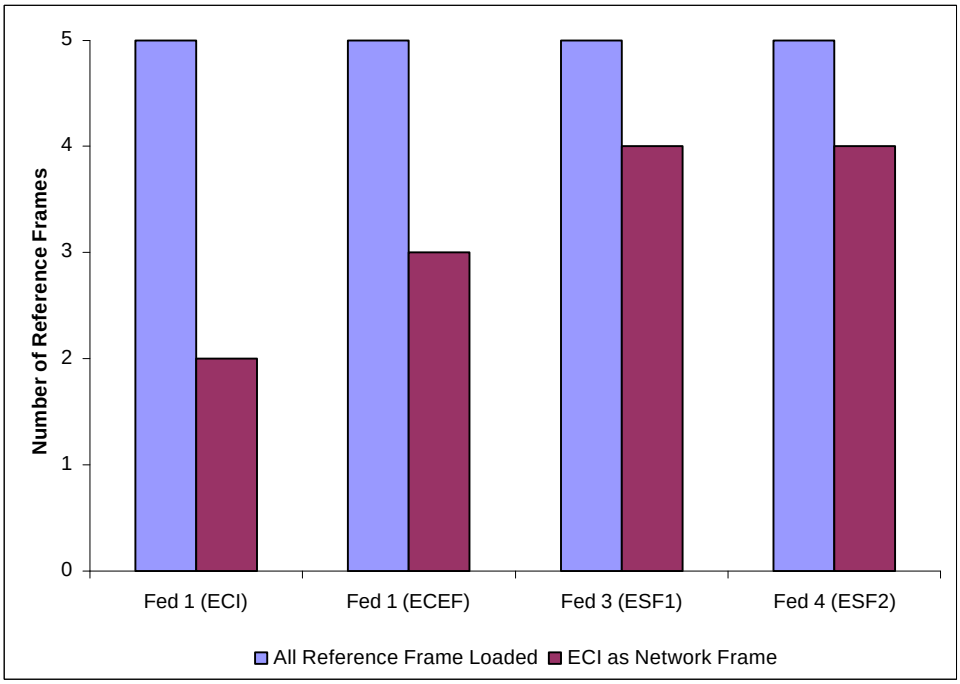


Figure C.4: Number of Reference Frames Loaded (including Body Frame)

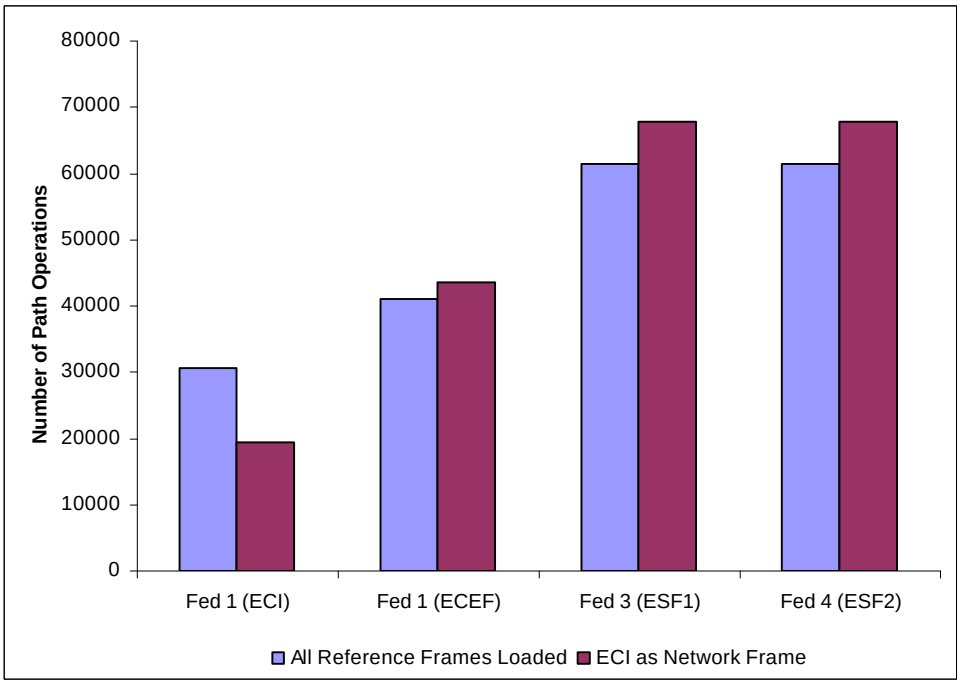


Figure C.5: Total Number of Path Operations

C.3 Results From Demonstration 3

Table C.6: Runtime for Control Model

	Eccentricity			
Time Step	e = 0.00	e = 0.25	e = 0.60	e = 0.85
1.06795	6.52	6.33	6.33	6.34
1.06795×10^{-1}	54.25	53.17	53.17	53.16
1.06795×10^{-2}	521.25	520.41	520.66	519.83
1.06795×10^{-3}	5190.47	5192.78	5193.75	5185.24
1.06795×10^{-4}	51920.70	51915.25	51935.58	51901.42
Adaptive	9.86	10.08	11.48	13.03

Table C.7: Runtime Using Intermediate Frame With Adaptive Critical Levels

	Eccentricity			
Time Step	e = 0.00	e = 0.25	e = 0.60	e = 0.85
1.06795	10.30	9.88	9.89	9.88
1.06795×10^{-1}	88.78	86.72	86.80	86.72
1.06795×10^{-2}	853.11	850.39	849.24	847.64
1.06795×10^{-3}	8329.36	8326.61	8338.56	8334.30
1.06795×10^{-4}	83049.12	83044.23	83069.59	83007.06
Adaptive	11.14	12.59	15.16	17.81

Table C.8: Runtime Using Intermediate Frame With Fixed Critical Levels

	Critical Level for Velocity			
Time Step	2^{-3}	2^{-7}	2^{-12}	2^{-17}
1.06795	10.08	9.86	9.88	9.86
1.06795×10^{-1}	88.13	85.33	86.75	86.72
1.06795×10^{-2}	843.86	829.66	852.91	853.55
1.06795×10^{-3}	8293.92	8264.77	8297.69	8519.27
1.06795×10^{-4}	82768.84	82647.20	82678.88	83538.14

Table C.9: Number of Time Steps for Adaptive Time Step

Eccentricity	Without Intermediate Frames	With Intermediate Frames
e = 0.00	790	668
e = 0.25	842	800
e = 0.60	974	980
e = 0.85	1121	1171

Table C.10: Mean Critical Level for Position

	Eccentricity			
Time Step	e = 0.00	e = 0.25	e = 0.60	e = 0.85
1.06795	5.16E-02	5.45E-02	6.14E-02	8.74E-02
1.06795×10^{-1}	1.95E-03	1.96E-03	2.12E-03	2.97E-03
1.06795×10^{-2}	7.01E-05	6.90E-05	9.44E-05	1.27E-04
1.06795×10^{-3}	4.46E-05	4.60E-05	5.66E-05	6.48E-05
1.06795×10^{-4}	4.45E-05	4.59E-05	5.65E-05	6.48E-05
Adaptive	1.05E-01	8.09E-02	6.24E-02	5.97E-02

Table C.11: Mean Critical Level for Velocity

	Eccentricity			
Time Step	e = 0.00	e = 0.25	e = 0.60	e = 0.85
1.06795	8.00E-02	8.40E-02	9.40E-02	1.13E-01
1.06795×10^{-1}	2.73E-02	2.78E-02	2.93E-02	4.02E-02
1.06795×10^{-2}	7.80E-03	7.42E-03	9.31E-03	1.30E-02
1.06795×10^{-3}	2.48E-03	2.55E-03	2.95E-03	4.17E-03
1.06795×10^{-4}	8.21E-04	8.35E-04	9.10E-04	1.31E-03
Adaptive	1.02E-01	9.82E-02	9.82E-02	1.36E-01

Table C.12: Statistic for Paired t-Test for Difference in Actual Error

	Eccentricity			
Time Step	e = 0.00	e = 0.25	e = 0.60	e = 0.85
1.06795	-4.43	-4.42	1.57	-2.34
1.06795×10^{-1}	-12.88	-9.90	-11.18	-7.58
1.06795×10^{-2}	-13.70	-13.24	-12.05	-10.35
1.06795×10^{-3}	-9.83	-12.04	-10.23	-11.44
1.06795×10^{-4}	-13.10	-16.25	-13.38	-10.93
Adaptive	-5.73	-5.12	-6.46	-7.01

Table C.13: Statistic for Paired t-test for Difference in Theoretical Error Limit

	Eccentricity			
Time Step	e = 0.00	e = 0.25	e = 0.60	e = 0.85
1.06795	-730	-332	-403	-242
1.06795×10^{-1}	-4922	-329	-430	-468
1.06795×10^{-2}	-18164	-314	-442	-495
1.06795×10^{-3}	-112559	-312	-434	-491
1.06795×10^{-4}	-196006	-310	-431	-491
Adaptive	-444	-225	-347	-157

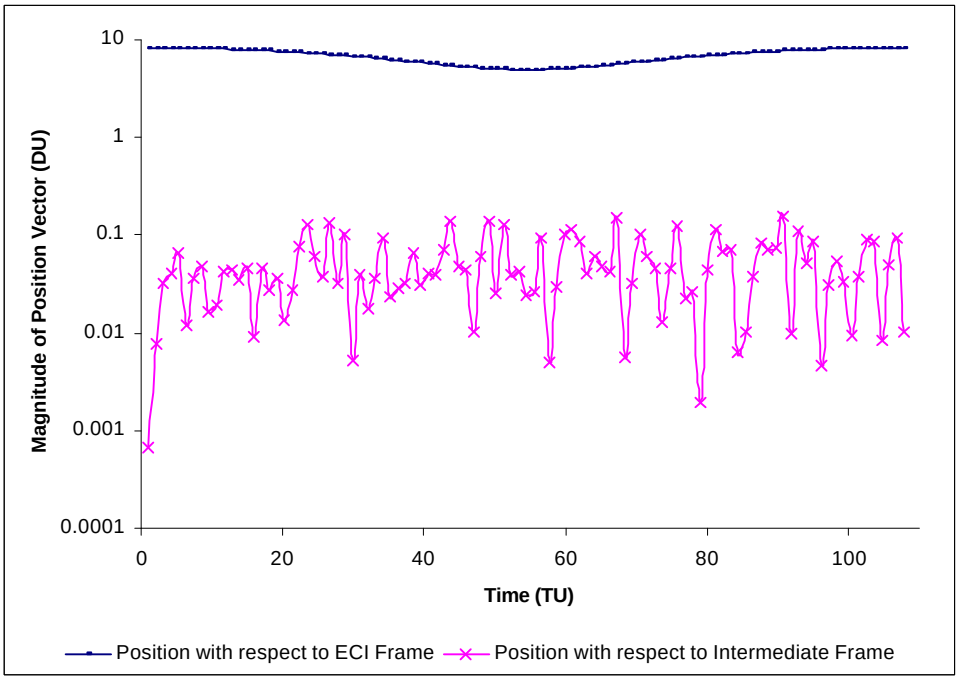


Figure C.6: Magnitude of Position Vector of a Satellite With e = 0.25

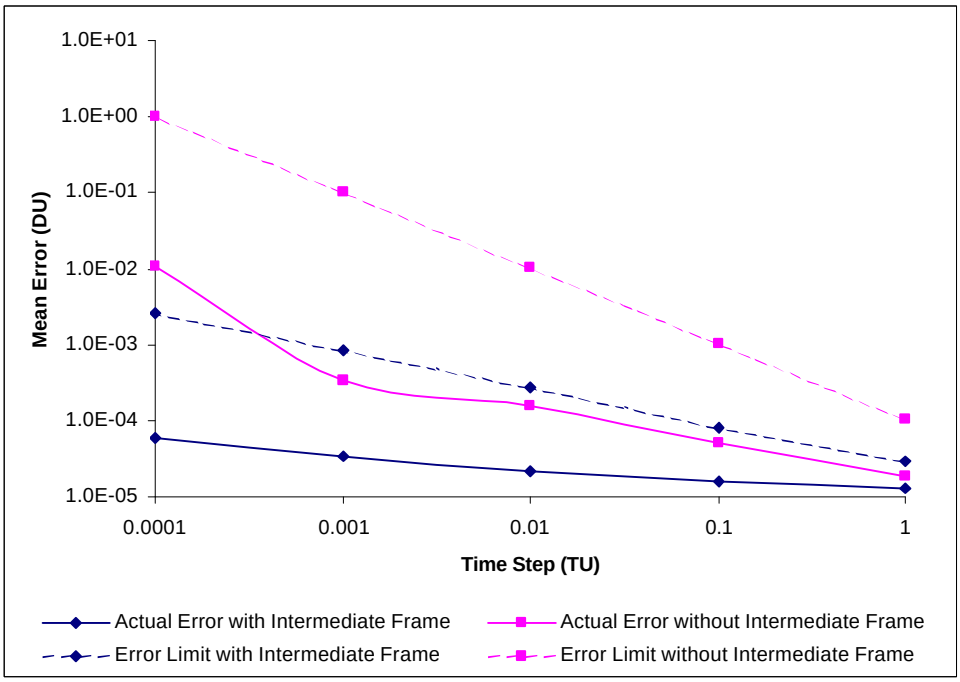


Figure C.7: Mean Position Errors for Different Time Steps

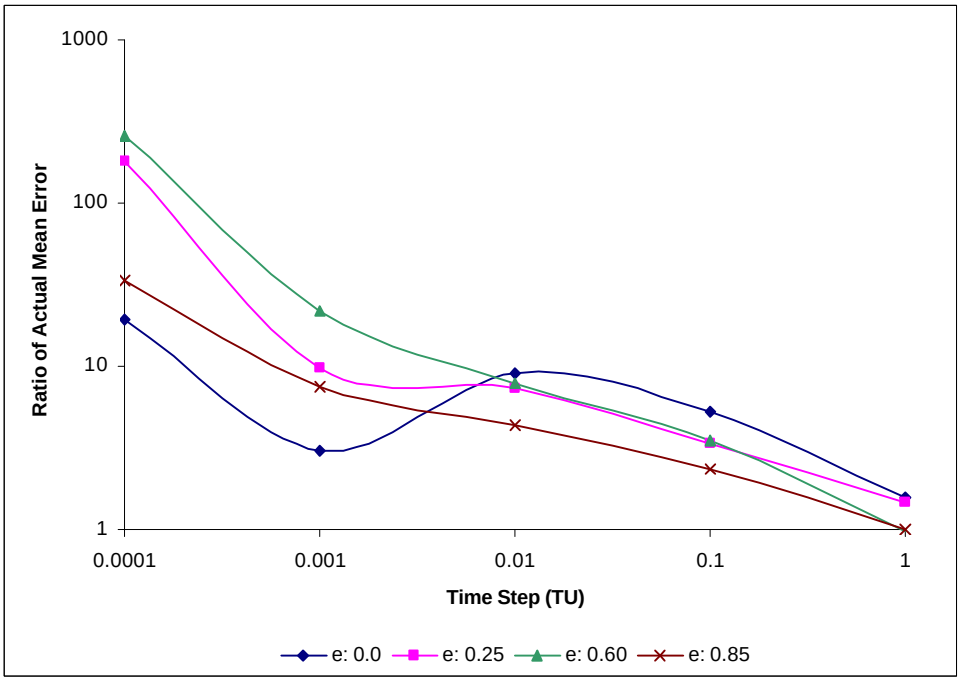


Figure C.8: Ratio of Mean Actual Errors for Position at Different Eccentricities

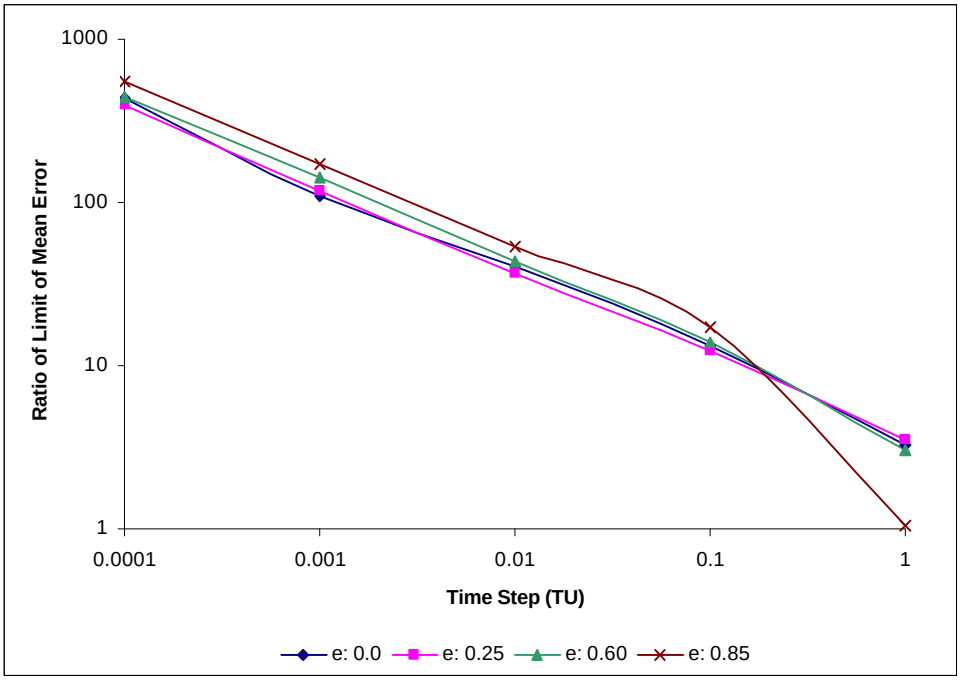


Figure C.9: Ratio of Mean Error Limits for Position at Different Eccentricities

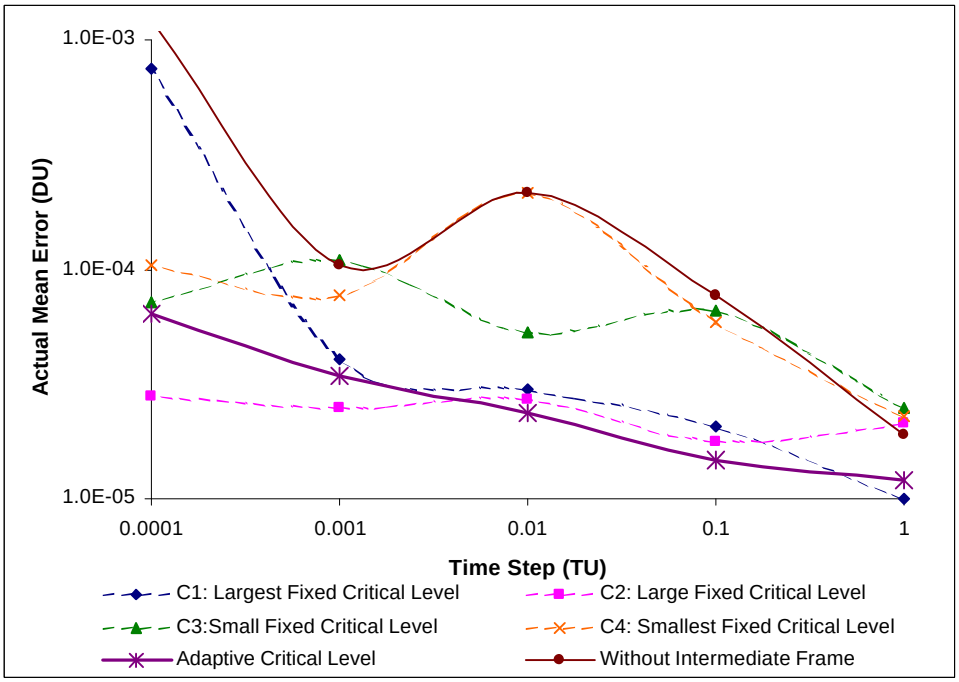


Figure C.10: Mean of Actual Errors for Position With Different Critical Levels

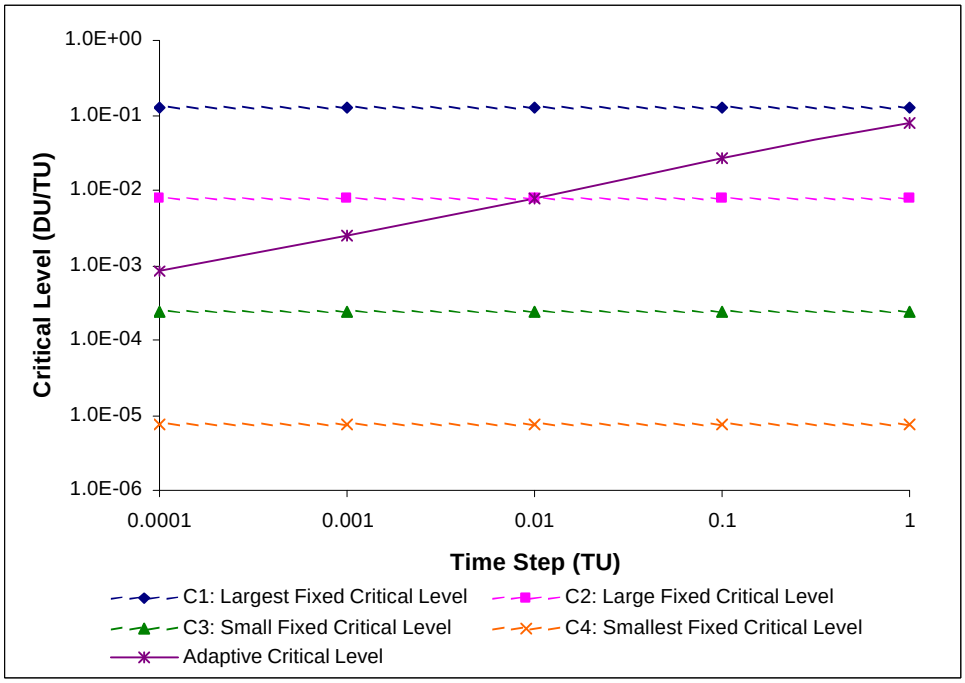


Figure C.11: Variation of Adaptive Critical Level for Velocity With Time Step

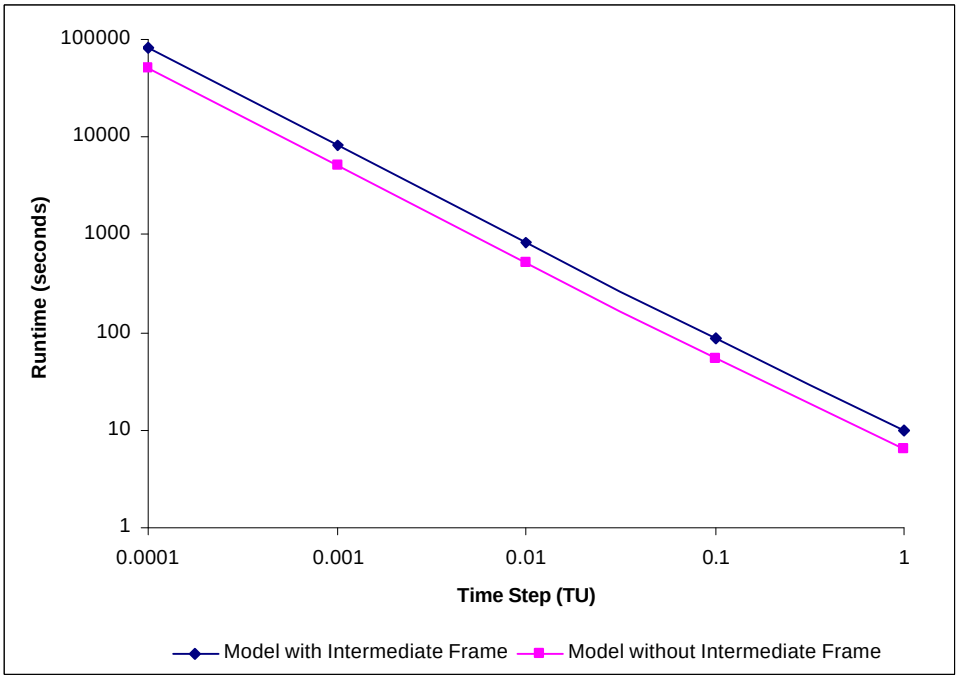


Figure C.12: Mean Runtimes for Scenarios in Demonstration 3

REFERENCES

[1] Foster, L., "Fidelity in Modeling and Simulation", Proceedings of the 1997 Spring Simulation Interoperability Workshop, Vol. 1, Inst. for Simulation and Training, Orlando, 1997, pp 473-481.

[2] Lim, W. C., "Effects of Reuse on Quality, Productivity, and Economics," IEEE Software, Vol. 11, No. 5, 1994, pp. 23-30.

[3] Poulin, J. S., Caruso, J. M., Hancock, D. R., "The Business Case for Software Reuse," IBM Systems Journal, Vol. 32, No. 4, 1993, pp. 567-594.

[4] Press, W.H., et al., Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, New York, 1997.

[5] Ginsberg, J. H., Advanced Engineering Dynamics, Second Edition, Cambridge University Press, Cambridge, England, UK, 1998.

[6] Zipfel, P. H., Modeling and Simulation of Aerospace Vehicle Dynamics, AIAA Education Series, American Inst. of Aeronautics and Astronautics, Reston, 2000.

[7] Maxwell, E.A., General Homogenous Coordinates in Space of Three Dimensions, First Edition, Reprinted, University Press, Cambridge, 1959.

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

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

[10] Denavit, J., Hartenberg R.S., "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices," Transactions of the ASME, Journal of Applied Mechanics, Vol. 22, American Society of Mechanical Engineers, New York, 1955, pp. 215-221.

[11] Stevens, B. L., and Lewis, F. L., Aircraft Control and Simulation, John Wiley & Sons, Inc, New York, 1992.

[12] Beer, F.P., Johnston Jr., E. R., Vector Mechanics for Engineers: Dynamics, Third SI Metric Edition, McGraw-Hill Ryerson Limited, Toronto, 1999.

[13] Ascher, U. M., and Petzold, L. R., Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[14] Fujimoto, R. M., Parallel and Distributed Simulation Systems, John Wiley & Sons, Inc., New York, 2000.

[15] Miller, D. C., Thorpe, J. A., "SIMNET: The Advent of Simulator Networking," Proceedings of the IEEE, Vol. 83, Issue 8, Inst. of Electrical and Electronics Engineers, New York, 1995, pp. 1114-1123.

[16] Hofer, R. C., Loper, M. L., "DIS Today," Proceedings of the IEEE, Vol. 83, Issue 8, Inst. of Electrical and Electronics Engineers, New York, 1995, pp. 1124-1137.

[17] Foley, P. G., Mamaghani, F., Birkel, P. A., "The Synthetic Environment Data Representation and Interchange Specification (SEDRIS) Development Project", www.sedris.org/pap_dlds.htm (Accessed October 30,2005).

[18] Calvin, J. O., Weatherly, R., "An Introduction to the High Level Architecture (HLA) Runtime Infrastructure (RTI)", 14th Workshop on Standards for the Interoperability of Distributed Simulations, Vol. 2, Inst. for Simulation and Training, Orlando, 1996, pp 705-715.

[19] Stark, T. S., Weatherly, R., Wilson, A., "The High Level Architecture (HLA) Interface Specification and Application Programmer's Interface", 14th Workshop on Standards for the Interoperability of Distributed Simulations, Vol. 2, Inst. for Simulation and Training, Orlando, 1996, pp 851-856.

[20] Toms, R. M., Birkel, P. A., "Choosing a Coordinate Framework for Simulations", www.sedris.org/pap_dlds.htm (Accessed October 30,2005).

[21] Burchfiel, J., 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, 1990.

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

[23] Elking, E., McDonald, B., "GCS/Langley Coordinate Conversion", AIAA Paper 2001-4131, Aug. 2001.

[24] Toms, R. M., "Efficient Transformations from Geodetic to UTM Coordinate Systems", 15th Workshop on the Interoperability of Distributed Interactive Simulations, Vol. 1, Inst. for Simulation and Training, Orlando, 1996, pp 223-228.

[25] Toms, R. M., Smith, K. I., "SEDRIS Coordinate Transformation Services", www.sedris.org/pap_dlds.htm (Accessed October 30,2005).

[26] Devanbu, P., Karstu, S., Melo, W., Thomas, W., "Analytical and Empirical Evaluation of Software Reuse Metrics," Proceedings of the 18th International Conference on Software Engineering, 1996, pp. 189-199.

[27] Madden, M. W., "Measuring Reuse in a Simulation Framework," AIAA Journal of Aerospace Computing, Information and Communication, Vol. 1, No. 8, 2004, pp. 320-340.

[28] Frakes, W., Terry, C., "Reuse Level Metrics," Proceedings of the Third International Conference on Software Reuse: Advances in Software Reusability, 1994, pp. 139-148.

[29] Poulin, J. S., "Measuring Software Reusability," Proceedings of the Third International Conference on Software Reuse: Advances in Software Reusability, 1994, pp. 126-138.

[30] Sayers, M. W., "Symbolic Computer Language for Multibody Systems," AIAA Journal of Guidance, Control and Dynamics, Vol. 14, No. 6, 1991, pp. 1153-1163.

[31] Tanenbaum, A. S., Computer Networks, Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[32] Vágó, I., Graph Theory Application to the Calculation of Electrical Networks, Elsevier Science Publishing Company, Inc, Amsterdam, The Netherlands, 1985.

[33] Truss, J., Discrete Mathematics for Computer Scientists, Addison Wesley Longman Limited, Harlow, England, UK, 1999.

[34] Palmer, M. J., Sinclair, R. B., Advanced Networking Concepts, Course Technology, Cambridge, MA, 1997.

[35] Ippolito, C. A., Pritchett, A.R., “Software Architecture for a Reconfigurable Flight Simulator,” AIAA Paper 2000-4501, Aug. 2000.

[36] Hale, F.J., Introduction to Space Flight, Prentice-Hall, New Jersey, 1994, pp. 343.

[37] Bate, R.R., Mueller, D.D., White, R.R., “Canonical Units”, Fundamentals of Astrodynamics, Dover Publications, Inc., New York, 1971, pp. 40-43.

[38] Kane, T.R., Levinson, D.A., “Secondary Newtonian Reference Frames”, Dynamics: Theory and Applications, McGraw-Hill, Inc., New York, 1985, pp. 166-169.