The Transformation of Oscillatory Equations in Six Degree of Freedom Re-entry Trajectory Models with Coordinate Transformations

by

George P. Davailus

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

APPROVED:

Calvin J. Ribbens

Calvin J. Ribbens, Chairman

Eugene Augustus Cooper Jr

Eugene A. Cooper

Layne T. Watson

Layne T. Watson

July, 1994

Blacksburg, Virginia

The Transformation of Oscillatory Equations in Six Degree of Freedom Re-entry Trajectory Models with Coordinate Transformations

by

George P. Davailus

Committee Chairman: Calvin J. Ribbens

Computer Science

(ABSTRACT)

Currently, many missile fire control systems use a three degree of freedom (3-DOF) trajectory model. The three degrees of freedom represent the linear motion of the missile in three dimensions. A 6-DOF model adds roll, pitch, and yaw, or angular motion in three dimensions to the first three degrees of freedom. Because more of the missile's attributes are modeled, a 6-DOF model is more accurate than a 3-DOF model. For the same reason, a 3-DOF model is easier to develop and executes faster. Also, because a 3-DOF model ignores the seemingly random angular motion, the step sizes used to integrate 3-DOF models are larger.

The goal of this project is to develop a 6-DOF re-entry model with the accuracy of a 6-DOF model with conventional equations of motion and computational speed at least comparable to the 3-DOF model. This can be achieved by transforming the equations that compute the effects of angular motion so that they are better conditioned. Essentially, this is done by fitting a sine wave to the oscillating state variables representing the orientation and angular rates, namely the quaternions and the angular velocity. This thesis shows the results of transforming the oscillating variables of the state vector.

ACKNOWLEDGEMENTS

The author would like to thank the following people for their suggestions, allowing me to bounce ideas off of them, and/or reviewing this document: Barry Bressler, Mark Flenner, Roger Gray, John Lawton, Craig Martel, Manny Prescini, and Frank Regan; all from the Naval Surface Warfare Center Dahlgren Division. Additionally, the Naval Surface Warfare Center Dahlgren Division provided computer and other resources, without which this thesis would not have been possible. Finally, the author would like to thank the committee chairman Calvin Ribbens, and the other members of the committee Layne Watson and Gusty Cooper for allowing me to bounce ideas off of them, and/or reviewing this document.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Background

The computation of trajectories that involve rigid bodies moving through the atmosphere requires the numerical solution of a large set of ordinary differential equations

$$x(0) = x_0, \tag{1.1}$$

$$\frac{dx}{dt} = f(x,t). \tag{1.2}$$

One way to model this trajectory is to use a three degree of freedom (3-DOF) simulation, which is essentially a point-mass model computing only the linear motion. A second way is to use a six degree of freedom (6-DOF) simulation, which computes both the linear and angular motion of the rigid body. For typical 6-DOF simulations, the components of the unknown vector x include the three components of the angular velocity—pitch, yaw, and roll—and the four components of the *quaternion* vector. The quaternion vector is a way to represent a transformation from one frame of reference to another. Depending on the trajectory, all of these components can oscillate very rapidly. This pattern is regular enough that when the angular velocity and quaternions are graphed versus time they look very much like sine waves. For a detailed derivation of Equations (1.1–1.2) and a detailed discussion of 3- and 6-DOF simulations see Regan [19].

It is possible, however, to give a very high level view of the initial conditions, state equations, and major parameters of 3- and 6-DOF simulations in this chapter. The major parameters in a 3-DOF simulation include the mass, speed, latitude, longitude, azimuth, and flight path angle of the rigid body, the dynamic pressure the atmosphere exerts on the rigid body, and the gravitational and aerodynamic forces experienced by the rigid body. The azimuth is the direction the rigid body is heading in, measured from due north. For example, a rigid body heading due east would have an azimuth of 90. The flight path angle is a measure of the steepness of the rigid body's dive. A large, negative flight path angle means a steep dive. A small, negative flight path angle means a shallow dive. The initial conditions for a 3-DOF simulation are as follows:

$$\begin{aligned}\vec{P} &= \begin{pmatrix} (h_{rb} + r_e)\cos\phi\cos\lambda \\ (h_{rb} + r_e)\cos\phi\sin\lambda \\ (h_{rb} + r_e)\sin\phi \end{pmatrix} \\ \vec{V} &= T\begin{pmatrix} S \\ 0 \\ 0 \end{pmatrix}\end{aligned}$$

where $\vec{P}$ is the rigid body's position, r_e is the radius of the earth, h_{rb} is the altitude of the rigid body, ϕ is the rigid body's latitude, λ is the rigid body's longitude, $\vec{V}$ is the rigid body's velocity, S is the rigid body's speed, and T is a transformation matrix computed from the product of rotational matrices based on latitude, longitude, azimuth, and the flight path angle. More details on rotational and transformation matrices are given in Chapter 3. Generally, in a 3-DOF simulation only position and velocity are members of the state vector. Typical state equations are as follows:

$$\begin{aligned}\dot{\vec{P}} &= \vec{V} \\ \dot{\vec{V}} &= \vec{g} + \frac{1}{m_{rb}}\vec{A},\end{aligned}$$

where $\vec{g}$ is the gravitational acceleration and $\vec{A}$ is the aerodynamic force. Major parameters for a 6-DOF simulation include the center of mass, center of pressure, surface area, and diameter of the rigid body, the axial and normal aerodynamic forces, the pitching and pitch damping moments, and the angle of attack. The axial force is caused by the airflow over the front of the rigid body. The normal force, also known as lift, is caused by air flowing perpendicular to the rigid body. The pitching moment is an aerodynamic moment influencing the angular velocity of the rigid body. The pitch damping moment causes the rigid body to attempt to snap back to an earlier orientation, in much the same way that a stretched spring will attempt to snap back to its earlier position. The angle of attack is defined as the angle between the direction the rigid body is pointed in and the direction the rigid body is moving in. The state vector, in addition to the position and velocity, includes an angular velocity vector ($\vec{\omega}$), and an orientation vector called a quaternion ($\vec{q}$). As for initial conditions, the components of the angular velocity vector are given initial numerical values, and the quaternion vector is ultimately computed from a transformation matrix, similar to the one referred to above. The additional state equations are given as follows:

$$\begin{aligned}\dot{\vec{\omega}} &= I^{-1}(\vec{M} - \vec{\omega} \times (I\vec{\omega})) \\ \dot{\vec{q}} &= \frac{1}{2}Q\vec{\omega}\end{aligned}$$

where I^{-1} is the inverse moment of inertia matrix, $\vec{M}$ is the aerodynamic moment vector, I is the moment of inertia matrix, and Q is a matrix composed entirely of elements of the quaternion vector. Also, the state equation for the velocity derivative vector has a transformation matrix composed entirely of quaternion components added to it.

The state vector x is an element of $\Re^n$, where $n \geq 13$. This state vector in 6-DOF simulations includes at least position, velocity, and angular velocity vectors, which are elements of $\Re^3$, and a unit vector called a quaternion, which is an element of $\Re^4$. Since these quantities are changing continuously, one must numerically integrate them to determine their value at a particular point in time. For further details on the actual structure of a typical state vector, see Regan [19]. When numerically integrating the state vector, one either calculates the initial values of the dependent variables from known quantities, or these values are themselves known at the beginning of the trajectory, in this case, at a particular time t. One can then use a numerical integration scheme (e.g., Runge-Kutta) to compute the values of the dependent variables at the next interval; i.e., at time $t + \Delta t$. Typically, 3-DOF simulations are much simpler than 6-DOF simulations, because the angular motion is not computed in a 3-DOF simulation. Since the angular motion is ignored in a 3-DOF simulation, the state vector of a 3-DOF simulation is usually an element of $\Re^6$. Thus, computing the derivatives of a 3-DOF model requires many fewer operations than computing the derivatives for a similar 6-DOF model. 3-DOF simulations can also generally use larger integration step sizes than 6-DOF simulations, because angular motion is ignored. Thus, 6-DOF models have high accuracy with long computational times, and 3-DOF models yield less accuracy, but execute faster. Some situations desire the accuracy of the 6-DOF with the speed of the 3-DOF, or at least close to the speed of a 3-DOF. For example, real-time software in missiles, and CFD (computational fluid dynamics) codes. For real-time software, speed is the critical factor, so 3-DOF simulations are usually preferred.

By transforming the equations used in these types of problems to a more tractable form, 6-DOF simulations can be placed on a better footing in comparison with their 3-DOF counterparts. There are two areas to investigate for potential improvements in computational efficiency of 6-DOF simulations.

1. The integration methods can be investigated from two perspectives.

   (1) Reduce the number of derivative calculations per iteration.

   (2) Use a more accurate and/or stable method, allowing the integration step size to be increased.

2. The differential equations can be investigated from two perspectives.

   (1) Simply reduce the cost per derivative calculation.

   (2) Transform the equations so that they are better conditioned, thereby allowing the step size to be increased.

These two options are not mutually exclusive. For example, changing the differential equations may improve the conditioning of the formulation, which in turn allows a different integration method to be used. When examining the differential equations of a 6-DOF simulation for potential improvements, the equations corresponding to coordinate transformations and angular motion are good candidates for transformation.

The goal of this project is to improve the computational efficiency of a 6-DOF re-entry model (i.e., a rigid body re-enters the atmosphere and falls to the ground) to improve the feasibility of executing in a real-time environment. This can be accomplished by pursuing the second option (mentioned above) of investigating the differential equations. That is, the tractability of the model solution can be improved by transforming the equations used to compute the coordinate transformations and the angular motion so that their conditioning is improved, thus increasing the numerical stability and efficiency of the overall formulation. Section 1.2 sketches the general approach of improving the tractability of the model solution. Only one Runge-Kutta integration method is used by the 6-DOF model described in this thesis (see Appendix A for details), and the state vector used is an element of $\Re^{13}$.

1.2 Research Issues

6-DOF type problems, such as trajectories at high mach numbers with viscous flows, are computationally intense. The reason for this is that angular motion often involves oscillations that restrict the integration step size. Our approach will attempt to take advantage of these oscillations, rather than be hindered by them. Essentially, this can be accomplished by fitting a sine wave to the oscillating functions. With these oscillating functions in mind, it is desirable to know how much the step size can be increased by taking these oscillations into account. This will depend on the stability of the frequencies and amplitudes of the oscillating functions.

One potential difficulty with the transformation methods being proposed here is that the atmosphere is not uniform. At an altitude of roughly 18000 ft, half of the atmosphere is above you and half is below; at 36000 ft, three quarters is below and one quarter above; and at 54000 ft seven eighths is below and one eighth is above. The airflow is laminar in the upper portion of the atmosphere; i.e., the flow is smooth and predictable. In the lower part of the atmosphere, however, the airflow is turbulent. As a result, the frequencies and amplitudes of these oscillating functions may change nonlinearly, which could cause problems for this method, especially in the lower part of the atmosphere.

This thesis will describe three transformations that can be made to the trajectory formulation. Two of these transformations replace the oscillating state variables one for one, making the state vector an element of $\Re^{13}$. The third transformation replaces three state variables with four other state variables, making the state vector an element of $\Re^{14}$. We will compare these methods with the unmodified formulation. Hybrid methods (i.e., transformations in the upper part of the atmosphere and the unmodified formulation in the lower part of the atmosphere) will also be considered.

To summarize, an approach that takes the oscillatory nature of angular motion into account appears to be based on sound principles, but it may not work out as well as might be expected. Nevertheless, any reduction in the execution time of real time software is useful, and this technique is worth investigating even if it yields only a small part of its potential.

1.3 Thesis Organization

Chapter 2 deals with previous literature in the general area of oscillating functions and trajectory simulations, and the lack of literature dealing specifically with the integration of oscillating state variables in 6-DOF models (most attempts to improve the execution time of 6-DOF models do not get to the crux of the problem: the oscillating state variables, mainly due to their seeming intractability). Chapter 3 gives an overview of the quaternion vector and an approach that deals with its oscillatory nature. Chapter 4 gives an overview of the angular rates and two approaches that deal with their oscillatory nature. Chapter 5 describes the 6-DOF model that was used, how the numerical experiments were conducted, and the results. Chapter 6 concludes and summarizes the thesis. In addition, there are two appendices. Appendix A describes the algorithm used to simulate the missile trajectory.

Appendix B presents results supplementing the results from Chapter 5.

Chapter 2
Review of Literature

The main point to consider when dealing with previous literature is that there is not much research involving improving the tractability of 6-DOF models. Since the primary impediment to a rapidly executing 6-DOF model is the rapid oscillations of the angular motion, special integration methods that deal with oscillating functions should be considered. These include the following.

1. Fourier methods, including Fourier series, immediately come to mind when attempting to solve these problems. Weinan [24] discusses the use of the Fourier-Galerkin method in the solution of Navier-Stokes equations, which are commonly used in aerodynamic problems.

2. Laplace transforms are also often used in aerodynamic problems. Derrick and Grossman [9] describe their use in engineering mathematics in general.

3. Bessel functions. Regan [19] describes the use of Bessel functions in computing the value of the angle of attack of a nonrotating rigid body. The angle of attack is the angle between the velocity vector and the direction the missile is pointed, and has a significant effect on the aerodynamic forces and moments. Martin [13] does the same for rotating rigid bodies.

These three methods will be discussed further in Chapter 4. In general, however, even advanced engineering books do not come equipped with details describing how to apply these methods to specific and complex aerodynamic problems. Regan and other sources develop equations for the nutational and precessional frequencies, both of which refer to certain aspects of angular motion, but they usually say very little about the ***amplitudes*** of angular motion, and often many aspects of angular motion are ignored altogether. Nayfeh [15] [16] wrote two reports that cover similar subject matter. One other special method deserves some mention: stochastic differential equations. Stochastic methods have been used to compute the initial angular velocity (due to uncertainty in the initial wobble), and to approximate 6-DOF behavior in a 3-DOF model. However, using stochastic methods to solve for highly oscillatory state variables can be something else entirely, and requires that at least reasonable estimates of the frequency and envelope, or amplitude of the oscillating variables be obtainable. The two state vector variables that deal specifically with 6-DOF behavior, and especially the angular velocity, are not always very well behaved, and therefore good estimates of the frequency and amplitude cannot be counted on. As a result, these oscillating state variables would have to be integrated first, obtaining an average value for the state variable, and using a random variable to mimic the oscillatory behavior. This process would have to be repeated because of the exponential atmosphere (i.e., once for altitudes between 0 and 10,000 feet, once for altitudes between 10,000 and 20,000 feet). As a result, it would not be feasible to apply stochastic processes to the problem at hand. For further information, Papoulis [17] has a section on Kalman filtering and a chapter on harmonic analysis of stochastic processes.

In fact, articles in journals on 3-DOF approximations to the angular motion contained in 6-DOF models are more common than articles dealing directly with 6-DOF models. Luke [11], for example, shows a method for approximating the angular rates using the thrust vector. But with a re-entry body, one is dealing with an uncontrolled vehicle (i.e., there is no thrust). Lawton and Bushong [2] use the thrust vector, which is a reasonably good approximation for the direction the missile is pointed in, and the velocity vector to compute the angle of attack. But again, the re-entry body has no thrust, so this approach has problems. For that matter, most rigid bodies are missiles or rockets and have thrust. This is one reason not much literature on optimizing 6-DOF re-entry models exists.

There are methods that appear fairly general. Summerset and Chowkwanyun [23] devised a method that takes advantage of parallel processors, allowing the simultaneous computation of multiple trajectories. This is done by using Jacobian matrices. However, this investigation concerns the optimization of a single trajectory, so this approach is not directly relevant. Also, Giannakoglou, Chaviaropoulos, and Papailiou[6] devised a method for separating the rotational and nonrotational portions of transonic airflows. But this article only refers to airflows in two dimensions. In three dimensions, the spin, pitch, and yaw are intertwined in a complex manner. A change in the spin rate of a rigid body could affect its pitch, and vice-versa. Therefore, this approach does not appear to hold much promise, either.

There are a number of reasons for the scarcity of technical literature in this area. Angular motion is difficult to visualize, and as a result, is not exactly a popular topic for discussion. Trying to imagine a rigid body simultaneously spinning, pitching, and yawing, and predicting what will happen next, one can get an idea of how counter-intuitive angular motion can seem.

Secondly, 3-DOF models usually provide a close approximation to 6-DOF models, and since 3-DOF models are much faster than 6-DOF models, the 3-DOF models are the ones usually used to control the actual rockets and missiles. Thus, there might not seem to be any point in making the 6-DOF models more efficient. For zero angle of attack, the 3-DOF model is essentially equivalent to the 6-DOF model. However, if the angle of attack is not zero, a lifting force must be taken into account. This lifting force is difficult to estimate in a 3-DOF re-entry model, due to the difficulty in approximating the angle of attack for a 3-DOF model; but this is simple to do in a 6-DOF model. This is one example where a 6-DOF model must be used.

Thirdly, and surprisingly, attempts to improving the execution times of 6-DOF simulations are often met with indifference. The usual approach to solving a 6-DOF model is by brute force. Concerns with computational efficiency are often answered by advising the purchase of a newer, faster computer. Although by no means universal, this attitude of apathy is difficult to understand in light of the current environment of tight budgets in the aerospace community, but it can at least in part be attributed to the esoteric nature of the subject matter. Admittedly, 6-DOF models will never be as fast as their 3-DOF counterparts, but 6-DOF models are used to confirm the validity of 3-DOF models, among other things, and any savings in execution time should be welcomed. To quote Ernest Rutherford, "We are short of money, so we must think."

Mathematicians and other scientists may be able to assist the aerodynamic community, however, by applying their knowledge to specific engineering problems. For example, Solovyov, Volpert, and Davtyan [22] discuss the flow of liquids with changing viscosity, an important problem in many 6-DOF models. Muraskin [14] discusses sine curves within sine curves. Both of these articles refer to a repeating pattern of oscillations (e.g., a small amplitude in the first period, a large amplitude in the second period, then back to the small amplitude). But all these approaches do not find an optimal solution to computing aperiodic functions dealing with coordinate transformations and angular rates that have no repeating pattern and an oscillating amplitude.

There is a great deal of literature dealing with stiff ordinary differential equations that can be considered. Chartier [3] describes methods that use parallel techniques which are L-stable. Shroff and Keller [21] describe an iteration method for stabilizing systems of equations. Lambert [10] and Shampine and Gordon [8] devote an entire chapter to stiff methods in general. Gear [5] has a book on ordinary differential equations in general. Shampine [20] also describes methods for testing for stiffness and ways of dealing with it. Additionally, Gaffney [4] describes the performance of several numerical methods for the integration of stiff oscillatory ordinary differential equations and the use of the Jacobian matrix to smooth out the equations. All of these articles and books deal with numerical methods, but the main thrust of this investigation is reformulating the differential equations. Petzold [18] does suggest a reformulation, but this involves transforming the differential equations to a form where the period of the function is constant and the change in amplitude is linear. A perfectly constant frequency and a linearly changing amplitude cannot be counted upon in this problem. Acton [1] also has some sections on stiff methods in general.

Acton [1] also points out that many analysts, when confronted with stiff equations, will blindly integrate with standard methods, and when they fail, will resort to methods specially designed for stiff systems, rather than transforming the equations into a potentially more tractable form. He mentions the second order equation $\ddot{w} + Lw = 0$, where $L \gg 0$, being a large, positive, slowly varying function. He then notes that when solving a function like this, developing an alternate method in which the analyst can utilize, rather than be hindered by, the oscillatory nature of the function is very worthwhile. In this regard, he is absolutely correct, and this thesis follows his approach. Madelung [12] details a similar method. The solution he proposes is to integrate the amplitude and phase angle of the oscillating function, rather than integrating the function directly. He then sketches out a very general method for going about this. The subsequent chapters in this thesis will describe three methods along these very lines that can be applied to 6-DOF models.

Chapter 3

Quaternion Transformations

This chapter deals with the missile orientation and its oscillatory nature in this problem. The first section deals with the different coordinate systems needed to develop the 6-DOF model. The second section describes the transformations of vectors from one coordinate system to another. Section 3.3 describes the different sets of state variables that can be used to represent the transformations. Finally, in Section 3.4 a way to transform the state variables representing the orientation to a more tractable form is described. Numerical results from implementing this transformation are given in Chapter 5.

3.1 Coordinate Systems

When one considers a 6-DOF simulation of some object, such as a missile or a reentry body, flying relative to the earth, there are several different coordinate systems (or frames) necessary to construct the simulation. By having several coordinate frames, the simulation can compute vector quantities in the frame in which they are most naturally represented, and then transform the vector quantities from one coordinate system to another. These transformation techniques become part of the differential equations. The following are three typical coordinate frames.

1. The **geocentric frame.** This coordinate system is an inertial frame used to compute the flight of the object relative to the earth. The inertial frame does not rotate with the earth; rather, the earth rotates within this frame. The origin of this frame is the center of the earth. The z-axis goes through the north and south poles. The x- and y-axes lie in the equatorial plane. For this model, the x-axis goes through 0

and 180 degrees longitude, and the y-axis 90 and -90 degrees longitude. The rotation of the earth is taken into account by a rotational matrix computed from the earth's angular rate. The geocentric frame is used to compute values such as the gravitational acceleration.

2. The **object or rigid body frame**. The origin of this frame is the center of mass of the object. The x-axis goes through the front and back of the object, with the positive half going through the front. The z-axis initially goes through the top and bottom of the missile, with the positive half going through the bottom of the missile. The y-axis initially goes through the left and right sides of the missile, with the positive half going through the right side when looking from the rear. The object frame is used to compute quantities such as the angle of attack and the angular velocity.

3. The **object's velocity frame**. This frame is also known as the wind or aeroballistic frame. The velocity frame is chosen so that initially the y and z-components of the velocity vector with respect to this frame are equal to zero. The velocity frame is used to compute quantities such as the aerodynamic drag on the object.

In solving equations of motion, the formulas must often change the frame of reference of various vector quantities by using a coordinate transformation. Strictly speaking, this transformation does not change the magnitude or direction of a vector, but the values of the vector components do vary as the frame is rotated.

3.2 Directional Cosine Matrices

One method for changing coordinate systems is by using directional cosine matrices (DCMs). To change the coordinate frame of a vector, rotate the coordinate system the vector is currently represented in through up to three angles, also known as Euler angles, about any of the axes. At most, three rotations are required to bring any two coordinate systems into coincidence. Given two coordinate systems a and b, one can solve for the three

Euler angles θ_1, θ_2, θ_3. A positive rotation about an axis a is defined as a clockwise rotation viewed along the positive the a axis. A rotation through an angle θ_1 about the x-axis can be represented by a rotational matrix

$$\Gamma_1 = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & \cos\theta_1 & \sin\theta_1 \\ 0.0 & -\sin\theta_1 & \cos\theta_1 \end{pmatrix} .$$

Notice that $\Gamma_1(3,2)$ is negative, rather than $\Gamma_1(2,3)$. This is due to the placement of the three axes and the definition of a positive rotation. A rotation about the z-axis would require a similar change in sign. A rotation through an angle θ_2 about the y-axis can be represented by another rotational matrix

$$\Gamma_2 = \begin{pmatrix} \cos\theta_2 & 0.0 & -\sin\theta_2 \\ 0.0 & 1.0 & 0.0 \\ \sin\theta_2 & 0.0 & \cos\theta_2 \end{pmatrix} .$$

A rotation through an angle θ_3 about the z-axis can be represented by yet another rotational matrix

$$\Gamma_3 = \begin{pmatrix} \cos\theta_3 & \sin\theta_3 & 0.0 \\ -\sin\theta_3 & \cos\theta_3 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} .$$

The product of the above three matrices $\Gamma_1\Gamma_2\Gamma_3$ is a DCM

$$\begin{pmatrix} \cos\theta_2\cos\theta_3 & \cos\theta_2\sin\theta_3 & -\sin\theta_2 \\ \sin\theta_1\sin\theta_2\cos\theta_3 - \cos\theta_1\sin\theta_3 & \sin\theta_1\sin\theta_2\sin\theta_3 + \cos\theta_1\cos\theta_3 & \sin\theta_1\cos\theta_2 \\ \cos\theta_1\sin\theta_2\cos\theta_3 + \sin\theta_1\sin\theta_3 & \cos\theta_1\cos\theta_2\sin\theta_3 - \sin\theta_1\cos\theta_3 & \cos\theta_2\cos\theta_1 \end{pmatrix} .$$

Multiplying a vector by the transpose of the above matrix ($\Gamma_3^T\Gamma_2^T\Gamma_1^T$, since all three matrices are orthogonal) transforms the vector to the desired frame of reference, assuming that the first rotation is about the z-axis, the second rotation is about the y-axis, and the third rotation is about the x-axis. The order of the rotations is important—a fact that is easy to illustrate with a simple example. Suppose you are flying level in an airplane. Pulling back on the stick and going straight up and then banking 90 degrees to the right produces a different orientation than banking 90 degrees to the right first and then pulling back on the stick. Multiplying the above DCM by a vector simply rotates the *vector* from one coordinate system to another. If the rotation of the *coordinate system* is needed, it is necessary to multiply the transpose of the above DCM by the vector. Also, since the magnitude of a vector should not change simply due to a coordinate transformation, the transformation matrix must be orthogonal. Take the two coordinate systems a and b, and a vector $\vec{v}_a$ in the a coordinate system. To rotate $\vec{v}_a$ from a to b, multiply the above DCM by $\vec{v}_a$.

$$DCM \cdot \vec{v}_a$$

To transform $\vec{v}_a$ to the b coordinate system, multiply by the transpose of the above DCM [19].

$$DCM^T \cdot \vec{v}_a = \vec{v}_b$$

3.3 Updating the DCM

A 6-DOF formulation needs transformation matrices to transform vectors from the geocentric to the object frame, and vice-versa. Since the object has continuously changing angular motion, the matrices are constantly changing. One such matrix is the velocity to rigid body matrix. Aerodynamic effects are most easily computed in the velocity frame. But one needs to know what those effects would be in the body frame, and ultimately in the geocentric frame. Fortunately, one rotation about the x-axis of the velocity frame is sufficient to bring these two frames into coincidence. Therefore, only one rotational matrix is needed, and it is unnecessary to update this matrix using integration.

The rigid body to geocentric matrix is a different story. Because of the simultaneous spinning, pitching, and yawing in relation to the earth, three rotational matrices are needed to compute this transformation matrix, and all three must be updated with each integration cycle. One way to handle this is to make the transformation matrix a state variable, integrating it with each iteration. Accurately updating the directional cosine matrix as a state variable is difficult, due to problems with maintaining orthogonality. There are two methods that can be used to avoid this problem: Euler angles and quaternions.

As mentioned in the preceding section, Euler angles are the angles through which the three axes are rotated. The Euler angles could be included as state variables, with three new equations, but this method has its own problems. The derivatives for the Euler angles are given as follows:

$$\dot{\theta}_1 = \omega_1 + \omega_2 \sin\theta_1 \tan\theta_2 + \omega_3 \cos\theta_1 \tan\theta_2,$$

$$\dot{\theta}_2 = \omega_2 \cos\theta_1 - \omega_3 \sin\theta_1,$$

$$\dot{\theta}_3 = \frac{\omega_2 \sin\theta_1 + \omega_3 \cos\theta_1}{\cos\theta_2},$$

where ω is the angular velocity in the object frame. As can be seen there are singularities in the derivatives for θ_1 and θ_3. Although there are ways to get around this, many 6-DOF models simply use quaternions.

A quaternion is a 4×1 unit vector $\vec{q}$ which represents an angle of rotation and three direction angles. Using quaternions, a transformation matrix can easily be computed. Because it is easier to maintain the unity of a vector than to maintain the orthogonality of a matrix, and because there are no singularities that must be dealt with, quaternions are generally the preferred method for updating the coordinate transformations as the numerical integration proceeds. A detailed discussion of quaternions is beyond the scope of this thesis. Interested readers are referred to Appendix D of Regan [19] for further information. In the remainder of this section we will give sufficient details to understand how the quaternion vector is used in trajectory calculations, including some analogies to help with an intuitive understanding of their role.

The initial quaternion computations, calculated at time $t = 0$, are as follows.

$$q_1 = \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\cos\frac{\theta_3}{2} - \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\sin\frac{\theta_3}{2},$$

$$q_2 = \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\sin\frac{\theta_3}{2} - \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\cos\frac{\theta_3}{2},$$

$$q_3 = \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\cos\frac{\theta_3}{2} + \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\sin\frac{\theta_3}{2},$$
$$q_4 = \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\cos\frac{\theta_3}{2} + \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\sin\frac{\theta_3}{2},$$

where θ_1, θ_2, and θ_3 are the three Euler angles, also computed at time $t = 0$, from the initial transformation matrix (see Appendix A for further details).

The updated geocentric to object transformation matrix, which transforms a vector in the geocentric coordinate system to a vector in the object or rigid body coordinate system, is given by

$$T_{gc2o} = \begin{pmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{pmatrix} .$$

If the unity of the quaternion is maintained, the DCM computed from it is guaranteed to be orthogonal. The derivative of the quaternion is computed as follows.

$$\dot{\vec{q}} = \begin{pmatrix} 0.5(\omega_1q_4 + \omega_3q_2 - \omega_2q_3) \\ 0.5(\omega_2q_4 - \omega_3q_1 + \omega_1q_3) \\ 0.5(\omega_3q_4 + \omega_2q_1 - \omega_1q_2) \\ 0.5(-\omega_1q_1 - \omega_2q_2 - \omega_3q_3) \end{pmatrix}$$

where ω is the angular velocity vector in the object frame.

There is another way to view quaternions. They can be represented by three directional angles α, β, and γ, which represent one half of the angle formed between the x-axes, y-axes, and z-axes of the two coordinate systems, respectively, and the angle of rotation δ. The quaternion can be computed from these angles as follows.

$$\vec{q} = \begin{pmatrix} \cos\alpha\sin\frac{\delta}{2} \\ \cos\beta\sin\frac{\delta}{2} \\ \cos\gamma\sin\frac{\delta}{2} \\ \cos\frac{\delta}{2} \end{pmatrix}$$

Note that there is a redundancy here in that a vector in four dimensional space is representing a transformation in three dimensional space. For each redundancy in the state

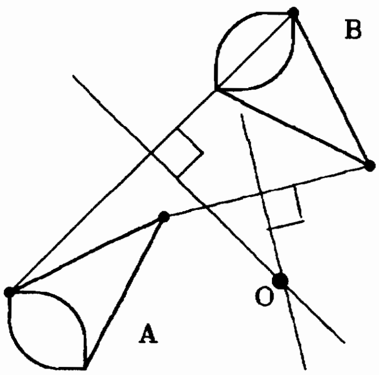

Figure 3.1: Two dimensional analogy for the quaternion vector.

variables representing the orientation, a constraint is placed upon them. In the case of the quaternion vector, the constraint turns out to be $||\vec{q}||_2 = 1$.

To understand how quaternions can be represented this way, a two dimensional analogy from Glashow [7] may be useful (see Figure 3.1).

Imagine two identical objects A and B in a plane, not overlapping, and with two different orientations. Then draw two lines, one connecting the front ends of the two objects, the other connecting the back ends. Bisect each line with a line perpendicular to it. Call the point where the two bisecting lines intersect O. Rotating the entire plane about the point O through some angle θ so that the final position of the first object is the same as the original position of the second object is equivalent to transforming the coordinate system of the first object. This view could be represented by

$$\begin{pmatrix} \theta \\ a \\ b \end{pmatrix},$$

where θ is the angle of rotation, a is the length of the line segment connecting the front ends of the two objects, and b is the length of the line segment connecting the back ends of the two objects. a and b are analogous to the three directional angles of the quaternion α,

β, and γ.

Here is another view of quaternions. In two-dimensional space, only one Euler angle is required to bring the two coordinate systems into coincidence, say θ. One analog of the quaternion is then

$$\vec{q} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

with the constraint $q_1^2 + q_2^2 = \cos^2\theta + \sin^2\theta = 1$.

For three dimensions the procedure is more difficult. Instead of rotating a plane about a point, a three dimensional space must be rotated about a line. Picture two coordinate systems that are not coincident. The line to be rotated about must be equilangular about the x-axes, y-axes, and z-axes of the two coordinate systems. The angles formed by the line and the three axes are the directional angles. Rotating about this line through the angle of rotation is equivalent to transforming a vector from one coordinate system to another.

3.4 Transforming the Quaternions to a More Tractable Form

Typically, when integrating an oscillating function, many points are needed to follow each oscillation if instability is to be avoided. For example, Acton [1] recommends eight to ten points. When integrating a sine wave, or for that matter, any oscillating function (such as the angular velocity and quaternion vector of a typical 6-DOF model), the integration step size is severely restricted if the frequency is large, that is, if there are a large number of crests in a given period of time. In the case of a rigid body moving through the atmosphere, it is easy to understand why. When a rigid body moves through the atmosphere, it pitches up and down, and yaws back and forth. In choosing a step size to use in a general integration method, one assumes that the rigid body has constant pitch and yaw rates throughout that time period. If the step size is made too large, then the simulation will indicate that the body will eventually tip up and begin tumbling out of control, which does not realistically happen.

Since the step size is restricted by the rapid variations of some of the state vector components, those components should be transformed into something that varies more slowly. That something is the frequency (or possibly phase) and amplitude of the angular velocity and quaternions. Since the angular velocity and quaternions often follow semi-regular patterns, the frequencies and amplitudes of the functions should vary much more slowly than the actual functions. One could then fit a sine wave to the oscillating functions, using trigonometric functions to convert the amplitudes and frequencies into angular velocity and quaternions. This method will henceforth be referred to as the semi-periodic quaternion method. This approach has great potential to enable significant increases in the integration step size, thereby reducing the execution time and computing costs of trajectory simulations.

For an arbitrary oscillatory function g, one can make the following transformation:

$$g(t) = A(t) \cdot \sin(P(t)),$$

where $A(t)$ is the amplitude function and $P(t)$ is the phase function. With quaternions the transformation is as follows:

$$\vec{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = \begin{pmatrix} A_1(t) \cdot \sin(P_1(t)) \\ A_2(t) \cdot \sin(P_2(t)) \\ A_3(t) \cdot \sin(P_3(t)) \\ A_4(t) \cdot \sin(P_4(t)) \end{pmatrix}. \tag{3.1}$$

With angular velocity there is a similar transformation:

$$\vec{\omega} = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \begin{pmatrix} A_5(t) \cdot \sin(P_5(t)) \\ A_6(t) \cdot \sin(P_6(t)) \\ A_7(t) \cdot \sin(P_7(t)) \end{pmatrix}, \tag{3.2}$$

where ω_1 is the spin or roll rate of the rigid body, ω_2 is the pitch, and ω_3 is the yaw. Hereafter, q will represent $\vec{q}$, and ω will represent $\vec{\omega}$. Also, q_n will represent the nth component of q, and ω_n will represent the nth component of ω. Equations (3.1–3.2) leave one additional unknown for each equation (a net increase of 7 unknowns), but no additional equations.

However, with a rapidly spinning rigid body, the following situation exists:

$$\omega_1 \gg \omega_2, \omega_3.$$

The derivatives for q_1 and q_4 are given by the following:

$$\begin{aligned}\dot{q}_1 &= \frac{1}{2}(q_4 \cdot \omega_1 - q_3 \cdot \omega_2 + q_2 \cdot \omega_3),\\ \dot{q}_4 &= \frac{1}{2}(-q_1 \cdot \omega_1 - q_2 \cdot \omega_2 - q_3 \cdot \omega_3).\end{aligned}$$

The above equations and their derivation are given in Regan [19]. Since $\omega_1 \gg \omega_2, \omega_3$, it is reasonable to approximate $\dot{q}_1$ and $\dot{q}_4$ as follows:

$$\begin{aligned}\dot{q}_1 &\approx \frac{\omega_1}{2} q_4, &(3.3)\\ \dot{q}_4 &\approx -\frac{\omega_1}{2} q_1. &(3.4)\end{aligned}$$

It was stated earlier that the quaternions look very much like sine waves. In fact, experiments with quaternion behavior in 6-DOF models suggest that q_1 and q_4 are related in a special way, and similarly for q_2 and q_3. In particular, note that

$$\begin{aligned}\frac{d}{dt}\left(\sin\left(\frac{\omega}{2} \cdot t\right)\right) &= \frac{\omega}{2} \cos\left(\frac{\omega}{2} \cdot t\right), &(3.5)\\ \frac{d}{dt}\left(\cos\left(\frac{\omega}{2} \cdot t\right)\right) &= -\frac{\omega}{2} \sin\left(\frac{\omega}{2} \cdot t\right), &(3.6)\end{aligned}$$

for some constant ω. The similarity between Equations (3.3–3.4) and (3.5–3.6), plus the observed behavior of the quaternions in typical calculations, make it reasonable to assume that q_1 and q_4 have the same amplitude and are 90 degrees out of phase. A similar relation holds for q_2 and q_3, and therefore one can make a similar assumption. Tables 3.1 and 3.2 show the values of the quaternions and angular velocity, respectively, at specific times for a typical calculation. Note that when q_1 reaches its peak or valley, q_4 is moving through zero and vice-versa. The same is true for q_2 and q_3. Hence the rationale for making the above assumption.

Placing this in the context of a rapidly spinning rigid body, one can understand why this is a good approximation. The spin rate is the primary angular rate, and the body does not pitch and yaw much. As a result, the orientation of the x-axis of the rigid body frame changes very slowly in relation to the earth, while the y-axis and z-axis rotate about the x-axis very rapidly, but will otherwise change very slowly. One can then let the orientation of

Table 3.1: Typical quaternion values over time.

time (sec)	q_1	q_2	q_3	q_4
0.00000	0.00000	-0.99144	0.00000	0.13053
0.03125	0.04193	-0.94342	0.30480	0.12426
0.06250	0.07652	-0.80402	0.58007	0.10591
0.09375	0.10548	-0.58673	0.79915	0.07730
0.12500	0.12420	-0.31263	0.94082	0.04111
0.15625	0.13094	-0.00824	0.99136	0.00093
0.18750	0.12494	0.29693	0.94587	-0.03941
0.21875	0.10687	0.57334	0.80876	-0.07597
0.25000	0.07837	0.79421	0.59332	-0.10520
0.28125	0.04229	0.93814	0.32041	-0.12431
0.31250	0.00203	0.99120	0.01647	-0.13133
0.34375	-0.03844	0.94824	-0.28906	-0.12573
0.37500	-0.07524	0.81344	-0.56659	-0.10786
0.40625	-0.10483	0.59984	-0.78922	-0.07963
0.43750	-0.12424	0.32814	-0.93541	-0.04357
0.46875	-0.13176	0.02466	-0.99097	-0.03337
0.50000	-0.12641	-0.28120	-0.95056	0.03730
0.53125	-0.10898	-0.55982	-0.81805	0.07436
0.56250	-0.08085	-0.78421	-0.60632	0.10423
0.59375	-0.04504	-0.93262	-0.33584	0.12413
0.62500	-0.00472	-0.99071	-0.03284	0.13192

Table 3.2: Typical angular velocity values over time.

time (sec)	spin rate ω_1	pitch rate ω_2	yaw rate ω_3
0.00000	20.000000	0.005000	-0.003000
0.03125	20.000000	0.002734	-0.005081
0.06250	20.000000	-0.000291	-0.005039
0.09375	20.000000	-0.003254	-0.004763
0.12500	20.000000	-0.004546	-0.002071
0.15625	20.000000	-0.005724	0.000699
0.18750	20.000000	-0.003911	0.003026
0.21875	20.000000	-0.002010	0.005402
0.25000	20.000000	0.000844	0.004821
0.28125	20.000000	0.001090	0.005659
0.31250	20.000000	0.004612	0.001486
0.34375	20.000000	0.005525	0.001633

the x-axis in relation to the earth be the slowly changing amplitude variable that Acton [1] refers to, and let the rotation of the other two axes be the frequency variable. Equation (3.1) can therefore be written as

$$q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = \begin{pmatrix} A_1(t)\cdot\sin(P_1(t)) \\ A_2(t)\cdot\cos(P_2(t)) \\ A_2(t)\cdot\sin(P_2(t)) \\ A_1(t)\cdot\cos(P_1(t)) \end{pmatrix}. \tag{3.7}$$

The initial phases and amplitudes are computed from Equation (3.7):

$$\begin{aligned} A_1(0) &= \sqrt{q_1^2(0)+q_4^2(0)}, \\ A_2(0) &= \sqrt{q_2^2(0)+q_3^2(0)}, \\ P_1(0) &= \arctan\left(\frac{q_1(0)}{q_4(0)}\right), \\ P_2(0) &= \arctan\left(\frac{q_3(0)}{q_2(0)}\right). \end{aligned}$$

The derivatives of these phases and amplitudes are given as follows.

$$\begin{aligned}
\dot{A}_1 &= \frac{2q_1\dot{q}_1 + 2\dot{q}_4q_4}{2\sqrt{q_1^2+q_4^2}} = \frac{q_1\dot{q}_1 + \dot{q}_4q_4}{A_1}, \\
\dot{A}_2 &= \frac{2q_2\dot{q}_2 + 2\dot{q}_3q_3}{2\sqrt{q_2^2+q_3^2}} = \frac{q_2\dot{q}_2 + \dot{q}_3q_3}{A_2}, \\
\dot{P}_1 &= \frac{\frac{q_4\dot{q}_1 - \dot{q}_4q_1}{q_4^2}}{1+\frac{q_1^2}{q_4^2}} = \frac{q_4\dot{q}_1 - \dot{q}_4q_1}{A_1^2}, \\
\dot{P}_2 &= \frac{\frac{q_2\dot{q}_3 - \dot{q}_2q_3}{q_2^2}}{1+\frac{q_3^2}{q_2^2}} = \frac{q_2\dot{q}_3 - \dot{q}_2q_3}{A_2^2}.
\end{aligned}$$

Note that there is a problem with singularities if $A_i = 0$. But there is a way to avoid this, by differentiating Equation (3.7). Note the following if $A_1 = 0$:

$$\begin{aligned}
\dot{q}_1 &= \dot{A}_1\sin(P_1) + A_1\dot{P}_1\cos(P_1) = \dot{A}_1\sin(P_1) \\
\dot{q}_4 &= \dot{A}_1\cos(P_1) - A_1\dot{P}_1\sin(P_1) = \dot{A}_1\cos(P_1)
\end{aligned}$$

So if $A_1 = 0$, use the following equations instead.

$$\begin{aligned}
P_1(0) &= \arctan\left(\frac{-q_4}{q_1}\right), \\
\dot{A}_1 &= \sqrt{\dot{q}_1^2 + \dot{q}_4^2}, \\
\dot{P}_1 &= \frac{\ddot{q}_4\dot{q}_1 - \dot{q}_4\ddot{q}_1}{\dot{A}_1^2}.
\end{aligned}$$

Details on the second derivative of the quaternion are given in the next chapter. Singularity problems with A_2 can be handled in a similar manner. The only time problems with singularities occur is when the x-axes of the two coordinate systems are parallel. To understand why, recall the definition of a quaternion vector:

$$\vec{q} = \begin{pmatrix} \cos\alpha\sin\frac{\delta}{2} \\ \cos\beta\sin\frac{\delta}{2} \\ \cos\gamma\sin\frac{\delta}{2} \\ \cos\frac{\delta}{2} \end{pmatrix}.$$

Recall that the angle δ represents the angle of rotation. The directional angle α is the angle formed between the line of rotation and the x-axis of both coordinate systems, the directional angle β is the angle formed between the line of rotation and the y-axis of both coordinate systems, and the directional angle γ is the angle formed between the line of rotation and the z-axis of both coordinate systems. In other words, the line of rotation *bisects* the angle between the x-axis of both coordinate systems (to yield α), and similarly for both β and γ. In order for one of the amplitudes to be equal to zero, either q_1 and q_4 must both equal zero, or both q_2 and q_3 must equal zero. There is only one way for both q_1 and q_4 to be equal to zero: $\gamma = 90$, $\delta = 180$. This means that the line of rotation is perpendicular to the x-axis of both coordinate systems. $\delta = 180$ means that the x-axis must be flipped 180 degrees about some line of rotation to bring the two coordinate systems into coincidence. This being the case, the two x-axes must have been parallel to begin with. There are two ways for both q_2 and q_3 to be equal to zero: $\delta = 0$ and $\beta = 90$, $\gamma = 90$. In the first case the angle of rotation is zero, so the two coordinate systems are coincident, and therefore the x-axes are obviously parallel. In the second case the line of rotation is perpendicular to the y-axes and z-axes of both coordinate systems. Therefore, the line of rotation must be perpendicular to the yz-plane of both coordinate systems. As a result the two yz-planes are parallel to each other, and therefore the two x-axes are parallel to each other. So the only time one of the amplitudes is zero is when the x-axes of the two coordinate systems are parallel. If only the y- or z-axes of the two coordinate systems are aligned, the only effect is that the two amplitudes are equal. So, if using the above equations does not solve the problem, the geocentric frame can simply be rotated 90 degrees about the y- or z-axes when one of the quaternion amplitudes goes below a certain value, say 0.1. This could be implemented by multiplying a rotational matrix about the y- or z-axis by the position and velocity vectors in the geocentric frame and the geocentric to missile transformation matrix. From the new transformation matrix, the new quaternion vector and its derivative would be computed. Then the quaternion amplitudes, phases, and their derivatives would be computed, and numerical integration would begin again. If one of the quaternion amplitudes was zero, rotating the geocentric coordinate system about the the y- or z-axis would have the effect of making the two amplitudes equal. For realistic 6-DOF models, effects such as gravity and the rotation of the earth depend on latitude, so rotating about the z-axis is probably a better choice.

Note that because of Equation (3.7) the number of unknowns has been reduced by four. We can therefore compute the initial conditions for the phases and amplitudes of the quaternion vector and then proceed with numerical integration as one normally would.

Chapter 4

Angular Rate Transformations

This chapter deals with the rate of change of the missile orientation, also referred to as the angular rates, and its oscillatory nature in this problem. The second section deals with one way to transform the state variables representing the angular rates to a more tractable form. Section 4.3 deals with a second way to transform the state variables representing the angular rates to a more tractable form. The fourth, fifth, and sixth sections, respectively, deal with the application of Laplace transforms, Fourier series, and Bessel functions to this problem.

4.1 Angular Rates

The previous chapter described how the quaternions could be transformed, but the angular velocity still needs attention. Because the rigid body is constantly spinning, pitching, and yawing, the quaternion derivative must be computed as specified in the previous chapter. Numerical integration is used to update the quaternion vector. Only the quaternion and angular velocity elements are used to compute the quaternion vector derivative. Since the angular velocity is continuously changing, numerical integration must be used to update it as well. This is a more difficult transformation to make, since the solution of the angular velocity, and especially the pitch and yaw, is much less tractable than the solution of the quaternion vector.

Since the spin rate dominates over the pitch and yaw rates, and the roll moment is negligible or zero, the x-component of the angular velocity will remain fairly constant. Therefore, while it may be desirable to transform ω_1, it is not absolutely necessary. That leaves the pitch and yaw rates. This chapter will cover two possible methods for transforming the pitch and yaw, but first, some definitions are needed.

The moment of inertia matrix is defined as

$$I = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix}.$$

The diagonal elements of the matrix represent the inertia that must be overcome in the respective plane when rotating about the respective axis. To understand what nonzero off-diagonal elements represent physically, imagine an airplane with one fuel tank on each wing. Suppose one tank is full, and the other is empty, and therefore the airplane has an uneven weight distribution along the y-axis. This is represented by nonzero off diagonal elements in the matrix. The significance of this is that if the pilot pulls back on the stick to climb higher, the air will push down on both wings with equal force, but since one wing contains less mass, that wing will undergo greater acceleration. The aircraft will therefore begin to roll.

Since the matrix I represents the moment of inertia, it is symmetric. In other words, an equal amount of inertia must be overcome whether rotating a rigid body clockwise or counter-clockwise about any of the axes. A rigid body is said to have rotational symmetry if it is represented by a matrix I where $I_{yy} = I_{zz}$. A rigid body is said to have configurational, or body symmetry if all the off diagonal elements of I are zero in the body frame. A rigid body's principal axis frame is defined as the frame such that the moment of inertia matrix in that frame has no off-diagonal elements that are nonzero. If the body has configurational symmetry, then the body and principal axis frames are coincident.

The derivative of the angular velocity, the angular acceleration, is given by

$$\dot{\omega}_{rb} = I^{-1}(\vec{M} - \omega_{rb} \times (I\omega_{rb})),$$

where $\vec{M}$ is the moment vector. Just as aerodynamic forces influence the linear or kinematic motion of the rigid body, moments influence the angular motion of the rigid body. If the rigid body has both rotational and configurational symmetry, it would be tempting to transform the pitch and yaw rates in a manner similar to the quaternions, with the pitch and yaw being 90 degrees out of phase. This would make the pitch and yaw orthogonal, for reasons similar to that for the quaternion pairs: the orientation of the x-axis of the rigid body frame changes very slowly in relation to the earth, while the y-axis and z-axis rotate about the x-axis very rapidly, but will otherwise change very slowly. But here, the moment vector could exert a great deal of influence on the angular acceleration, rendering any orthogonality assumptions essentially invalid. Additionally, for an asymmetric body, the pitch and yaw rates might not be orthogonal, and the amplitudes and frequencies could oscillate in such a manner as to render the formulation unstable. The problem must be approached in a different manner, one that takes the angular velocity ω out of the state vector and replaces it with some quantity that is better behaved. In other words, ω should be transformed to a vector quantity that is more tractable. One such vector quantity is the angular momentum, or $\vec{H}$. The other vector quantity described in this chapter is the derivative of the quaternion, $\dot{q}$. In Chapter 5 the experimental results for both approaches will be given.

4.2 Angular Momentum

One way to transform the angular velocity to a more tractable form is to use the angular momentum, which is defined as

$$\vec{H} = I\omega.$$

One could then use the corrollary of the law of conservation of angular momentum, which states that the change in angular momentum is directly proportional to the torque applied to the body. Since H_1 is fairly constant and H_2 and H_3 are orthogonal to each other, there is a transformation for the angular momentum similar to the quaternions:

$$\vec{H} = \begin{pmatrix} H_1 \\ A_h(t)\sin(P_h(t)) \\ A_h(t)\cos(P_h(t)) \end{pmatrix}.$$

The initial amplitude and phase equations are given by

$$\begin{aligned} A_h &= \sqrt{H_2^2 + H_3^2}, \\ P_h &= \arctan\left(\frac{H_2}{H_3}\right). \end{aligned}$$

The reason for the orthogonality of H_2 and H_3 is similar to that for the quaternion pairs: the orientation of the x-axis in relation to the earth changes very little, while the y-axis and z-axis rotate about the x-axis very rapidly, but will otherwise remain mostly unchanged. One can then let the orientation of the y- and z-axes be represented by the frequency variable. The amplitude and frequency of the angular momentum should be much more well behaved than the angular velocity, so this method holds some promise. To compute the updated angular velocity, simply multiply the inverse moment of inertia matrix by the updated angular momentum.

$$\omega_{i+1} = I^{-1}\vec{H}_{i+1}.$$

The derivative of the angular momentum is given by

$$\dot{\vec{H}} = \vec{M} - \omega_{rb} \times \vec{H},$$

where $\vec{M}$ is the moment vector. The phase and amplitude derivatives are given by

$$\begin{aligned} \dot{A}_h &= \frac{H_2\dot{H}_3 + \dot{H}_2H_3}{A_h}, \\ \dot{P}_h &= \frac{H_3\dot{H}_2 - \dot{H}_3H_2}{A_h^2}. \end{aligned}$$

In theory, one would have to be concerned about singularities here as well, but in practice there is always some pitch and yaw motion involved. The three above equations would replace the state equation for $\dot{\omega}$:

$$\dot{\omega}_{rb} = I^{-1}(\vec{M} - \omega_{rb} \times (I\omega_{rb}))$$

The approach described in this section will hereafter be referred to as the angular momentum transformation.

4.3 A Second Order Quaternion Equation

There is another possibility for transforming the angular velocity. One could compute the second derivative of the quaternions, and derive a second order equation. First define the matrix Ω.

$$\Omega = \begin{pmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{pmatrix}.$$

The quaternion first and second derivatives are given by the following:

$$\begin{aligned} \dot{q} &= \frac{1}{2}\Omega q & (4.1) \\ \ddot{q} &= \frac{1}{2}\dot{\Omega} q + \frac{1}{2}\Omega\dot{q}. & (4.2) \end{aligned}$$

Substituting $\frac{1}{2}\Omega q$ for $\dot{q}$ in the second order equation, we can rewrite it as follows.

$$\ddot{q} + J \cdot q = \vec{0} \qquad (4.3)$$

where $J = -\frac{1}{2}\left(\dot{\Omega} + \frac{1}{2}\Omega^2\right)$ is given by

$$J = \frac{1}{2}\begin{pmatrix} \frac{1}{2}\omega_T & -\dot{\omega}_3 & \dot{\omega}_2 & -\dot{\omega}_1 \\ \dot{\omega}_3 & \frac{1}{2}\omega_T & -\dot{\omega}_1 & -\dot{\omega}_2 \\ -\dot{\omega}_2 & \dot{\omega}_1 & \frac{1}{2}\omega_T & -\dot{\omega}_3 \\ \dot{\omega}_1 & \dot{\omega}_2 & \dot{\omega}_3 & \frac{1}{2}\omega_T \end{pmatrix},$$

and $\omega_T = \|\omega\|_2^2$ is the square of the total angular velocity. This looks very much like a simple harmonic oscillator. Unfortunately, J varies with time, although the diagonal elements significantly dominate the matrix. This leaves two options. The first one is to use a method that assumes that the first derivative does not appear in the equation. But doing this would require integrating the angular acceleration to update the angular velocity, and the point of this chapter is to find a way to avoid doing just that.

The second option is to reformulate the second derivative equation. First define the matrix Q (unitless).

$$Q = \begin{pmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{pmatrix}.$$

Equations (4.1–4.2) can be rewritten in the following way:

$$\begin{aligned} \dot{q} &= \frac{1}{2}Q\omega \\ \ddot{q} &= \frac{1}{2}\dot{Q}\omega + \frac{1}{2}Q\dot{\omega}. \end{aligned}$$

We now have a second order differential equation that can be transformed simply by taking the derivative of Equation (3.7). There is only one problem remaining: computing the updated angular velocity. However, this is easily solved. Because the columns of Q are orthogonal to each other, one can simply multiply the transpose of Q by the updated quaternion derivative:

$$\omega_{i+1} = 2Q_{i+1}^T\dot{q}_{i+1}.$$

The key point to keep in mind here is that although this approach does not allow the avoidance of the computation of ω or $\dot{\omega}$, it allows the model to avoid *numerically integrating* $\dot{\omega}$ to update ω. That can be accomplished with the above equation. Since either the angular momentum or the quaternion derivative vectors can be expected to be more tractable than the angular velocity, a sine wave would have a better chance of having a good fit with either alternative vector quantity than with the angular velocity, and as a result, updating ω and $\dot{\omega}$ could probably be done more accurately. The second derivatives of the quaternion phases and amplitudes are given as follows.

$$\begin{aligned}
\ddot{A}_1 &= \frac{(q_1\ddot{q}_1 + \dot{q}_1^2 + q_4\ddot{q}_4 + \dot{q}_4^2)A_1 - (q_1\dot{q}_1 + q_4\dot{q}_4)\dot{A}_1}{A_1^2}, \\
\ddot{A}_2 &= \frac{(q_2\ddot{q}_2 + \dot{q}_2^2 + q_3\ddot{q}_3 + \dot{q}_3^2)A_2 - (q_2\dot{q}_2 + q_3\dot{q}_3)\dot{A}_2}{A_2^2}, \\
\ddot{P}_1 &= \frac{(q_4\ddot{q}_1 - \ddot{q}_4 q_1)A_1 - (q_4\dot{q}_1 - \dot{q}_4 q_1)2\dot{A}_1}{A_1^3}, \\
\ddot{P}_2 &= \frac{(q_2\ddot{q}_3 - \ddot{q}_2 q_3)A_2 - (q_2\dot{q}_3 - \dot{q}_2 q_3)2\dot{A}_2}{A_2^3}.
\end{aligned}$$

The above four equations would replace the state equation for $\dot{\omega}$. Again, there are problems with singularities. If $A_1 = 0.0$, then use the following equations instead.

$$\ddot{\omega} \approx -\dot{\omega} \times \vec{H} - \omega \times \dot{\vec{H}}, \tag{4.4}$$

$$q^{(3)} = \frac{1}{2}Q\ddot{\omega} + \dot{Q}\dot{\omega} + \frac{1}{2}\ddot{Q}\omega, \tag{4.5}$$

$$\ddot{A}_1 = \frac{\dot{q}_1\ddot{q}_4 + \ddot{q}_1\ddot{q}_4}{\dot{A}_1}, \tag{4.6}$$

$$\ddot{P}_1 = \frac{(q_4^{(3)}\dot{q}_1 - \dot{q}_4 q_1^{(3)})\dot{A}_1 - (\ddot{q}_4\dot{q}_1 - \dot{q}_4\ddot{q}_1)\ddot{A}_1}{\dot{A}_1^3}. \tag{4.7}$$

In the upper portion of the atmosphere, Equation (4.4) is a good approximation for $\ddot{\omega}$, since $-\vec{H} \times \omega$ is the driving force behind $\dot{\omega}$. In the lower part of the atmosphere, though, there may be problems with this approach. Singularity problems with A_2 can be handled in a similar manner, or this problem could be handled by rotating the geocentric system about the y- or z-axis. Henceforth, this approach will be referred to as the $\dot{q}$ transformation.

Now both the quaternions and the angular velocity have been transformed, and numerical integration can now proceed as would normally be done, but this time, hopefully, with significantly larger step sizes. There are, however, some integration techniques that assume a periodic or oscillatory form which should be discussed.

4.4 Laplace Transforms

Laplace transforms can sometimes lessen the intractability of a differential equation by converting it to an algebraic equation. The Laplace transform of some function is defined as follows.

$$F(s) = \int_0^\infty e^{-st} f(t)dt.$$

Equation (4.3) is the one that needs to be transformed. Taking the first element in the quaternion we have:

$$F(s) = \int_0^\infty e^{-st}(\ddot{q}_1 + \frac{1}{4}\omega_T q_1 - \dot{\omega}_3 q_2 + \dot{\omega}_2 q_3 - \dot{\omega}_3 q_4)dt.$$

The question is now how can this be integrated? The variable that is being transformed is q. With ω and $\dot{\omega}$ being oscillating variable coefficients, this seems difficult at best. In particular, Derrick and Grossman [9] note that the variable coefficients must be polynomials to apply the transform. This is because the coefficients can be eliminated using integration by parts. Even if there were a way to transform functions with nonpolynomial, oscillating coefficients like ω and $\dot{\omega}$, there would be a great deal of difficulty in applying the inverse transform. Specifically, the trouble would lie in recognizing the transforms of basic functions amongst the maze of variables. For these reasons, Laplace transforms do not appear to be a worthwhile approach.

4.5 Fourier Series

A Fourier series can be represented by the following infinite series:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx),$$

where $a_0, \ldots, a_n$ and $b_1, \ldots, b_n$ are constants [9]. The thing to keep in mind is that Fourier series are best suited for periodic functions. But the 6-DOF model must be solved in an exponential atmosphere. As a result, the amplitudes of the quaternions and angular velocity oscillate and the frequency can change enough to be significantly aperiodic. Therefore, these oscillating functions really cannot be considered periodic, and in some cases the solutions are transient or even mildly chaotic.

One could attempt to use different Fourier series in different portions of the atmosphere. But due to the exponential nature of the atmosphere several Fourier series would be needed, especially in the lower altitudes. Even within this context, the results would not be very satisfactory, since the quaternions and angular velocity could be significantly aperiodic even in a small portion of the trajectory, say, from 50,000 to 40,000 feet. Even if this could be overcome, a different set of Fourier series would be needed for each distinct trajectory. In other words, the set of Fourier series needed for a shallow dive, high velocity trajectory would be different from the set of series needed for a steep dive, low velocity trajectory. This is hardly satisfactory, given that angular momentum and quaternion derivative transformations have the advantage of not being specific to only a small portion of the atmosphere, or to any one trajectory. Because of this, Fourier series are not always recommended for integration of oscillating state variables, and do not seem to be promising in this context.

4.6 Bessel Functions

A Bessel equation of order p is given by

$$x^2\ddot{y} + x\dot{y} + (x^2 - p^2)y = 0.$$

If a differential equation is not in the form of a Bessel function, it can often be converted to a Bessel function, even if the equation has variable coefficients. This is done by substituting for both the dependent variable and the variable coefficient so that it takes the form of a Bessel equation. But the variable coefficient ω, having a transient and sometimes mildly chaotic solution, is not even a sine wave, so it is difficult to envision what substitutions to make. It was stated earlier that Regan [19] gives a solution to the angle of attack using Bessel functions. The equation he uses is given as follows:

$$\ddot{\alpha} - \frac{\rho V^2 C_p S d}{I_{22}}\alpha = 0,$$

where α is the angle of attack, ρ is the air density, V is the rigid body's velocity, C_p is the pitching moment coefficient, S is the surface area of the rigid body, d is the diameter of the rigid body, and I_{22} is from the moment of inertia matrix. Of these, S, d, and I_{22} are invariant, ρ is known at each altitude, V can be computed using a 3-DOF model, and C_p is computed from the missile's configuration, which is invariant, and the angle of attack. So this is fairly close to being a closed form solution, and can be converted to a Bessel function by first changing the independent variable from time to altitude. There are two problems with applying this technique to the problem at hand. First, a rigid body with rotational symmetry (i.e., $I_{22} = I_{33}$) is assumed. Second, the solution of ω is nowhere near to being closed form, as can be inferred from above. So even though Bessel functions are used in many areas of aerodynamics, the problem studied here does not seem to be a good candidate for a method based on Bessel functions.

Chapter 5

Numerical Experiments

5.1 Methodology

The transformations of the quaternions and angular velocity were tested on a 6-DOF model of the author's own making. This model includes an exponential atmosphere with aerodynamic forces and moments. Important parameters for the rigid body include mass, diameter, and surface area. Spherical gravity is used for the Earth, and the Earth is assumed to be nonrotating. The atmosphere is not modeled with complete accuracy, but with sufficient accuracy to test the validity of the transformations. The change in airflow from laminar to turbulent is made at a single point, rather than scaling the effect gradually in a small transition region. Also, the rigid body is assumed not to lose mass (in an actual model, the outer skin of the rigid body would begin burning up). Configurational symmetry is assumed, and rotational asymmetry is assumed. For details on the algorithm(s) used, see Appendix A. The software was run on a Vax 6430, and the operating system was VMS 5.4.1. The software is written in C. The standard fourth order Runge-Kutta is used for all the transformed problems and for the original formulation for purposes of consistency, but it would be reasonable to suspect that a less stable numerical method unable to handle the original formulation could solve one of the transformed formulations.

A word about test cases: all re-entry bodies have a feasibility region known as the V-gamma map. This is a two dimensional plot, with the velocity (V) being the y-coordinate and the re-entry angle (gamma) being the x-coordinate. The re-entry angle is the angle formed between the velocity vector and the plane going through the rigid body's center of mass that is parallel to the plane tangent to the surface of the earth at the rigid body's latitude and longitude. For a steep dive, the rigid body will have a large, negative re-entry angle. For a shallow dive, the rigid body will have a small, negative re-entry angle. Points lying within the V-gamma map correspond to feasible trajectories; i.e., to velocities and re-entry angles that will allow re-entry to occur. Many V-gamma maps have several corners on them. This particular 6-DOF model has a V-gamma map with four corners (steep dive, high velocity; steep dive, low velocity; shallow dive, high velocity; and shallow dive, low velocity). Points outside this map are not considered to be valid. If the dive is too shallow, the rigid body will bounce off or burn up in the atmosphere. If the dive is too steep, the wind shear could result in large, chaotic oscillations and tear the rigid body apart. If the rigid body is moving too fast through the atmosphere, it will bounce off or burn up in the atmosphere. If the rigid body is moving too slowly, the atmosphere will have more time to act on the rigid body, making the oscillation of the amplitude of the oscillating function large, and ultimately causing the rigid body to start tumbling. The four corners on the map are therefore the points that stress the 6-DOF model the most. If the transformations are valid for these four points, then the transformations are very likely to be valid for the rest of the V-gamma map. Therefore, the four corners of the map were the points selected for testing. The oscillations of the amplitudes for trajectories in the interior of the V-gamma map are not expected to be as rapid as those at the top of the V-gamma map, because larger velocities tend to increase the dynamic pressure more, which is one of the driving forces behind the frequency of the oscillating amplitudes. On the other hand, slower velocities allow the atmosphere more time to act on the rigid body, making the oscillating amplitudes more pronounced. So, oscillations for trajectories in the interior of the V-gamma map are not expected to be as badly behaved as those at the bottom of the V-gamma map, either.

5.2 Results

Tables 5.2 and 5.1 present the error in position and velocity, respectively, for each of the methods. The unmodified formulation is labeled simply as Unmodified. The angular

Table 5.1: Velocity error (in feet/sec) at 100,000 feet.

	Unmodified		Alternative 1		Alternative 2	
Step	Absolute	Relative	Absolute	Relative	Absolute	Relative
1/16	$0.00 \cdot 10^{0}$	$0.00 \cdot 10^{0}$	$4.15 \cdot 10^{-2}$	$3.19 \cdot 10^{-6}$	$3.99 \cdot 10^{-2}$	$3.07 \cdot 10^{-6}$
1/8	$6.64 \cdot 10^{-2}$	$5.10 \cdot 10^{-6}$	$6.46 \cdot 10^{-2}$	$4.97 \cdot 10^{-6}$	$6.42 \cdot 10^{-2}$	$4.94 \cdot 10^{-6}$
1	unstable	unstable	$4.62 \cdot 10^{-2}$	$3.55 \cdot 10^{-6}$	$7.06 \cdot 10^{-2}$	$5.43 \cdot 10^{-6}$
5/4	unstable	unstable	$7.27 \cdot 10^{-2}$	$5.59 \cdot 10^{-6}$	unstable	unstable
4	unstable	unstable	$1.93 \cdot 10^{-1}$	$1.48 \cdot 10^{-5}$	unstable	unstable
8	unstable	unstable	$9.68 \cdot 10^{-1}$	$7.45 \cdot 10^{-5}$	unstable	unstable
10	unstable	unstable	$2.00 \cdot 10^{1}$	$1.54 \cdot 10^{-3}$	unstable	unstable
24	unstable	unstable	unstable	unstable	unstable	unstable

momentum transformation (described in Section 4.2) is labeled as Alternative 1, and the $\dot{q}$ transformation (described in Section 4.3) is labeled as Alternative 2. Each method computed the same trajectory from an initial altitude of 500,000 feet down to a final altitude of 100,000 feet. The errors reported in Tables 5.2 and 5.1 were computed by using the two-norm of the difference between the position and velocity vectors computed by the modified formulations and the position and velocity vectors computed by the "true" solution; that is, the unmodified formulation at half its original step size.

$$||p_{true} - p_{mod}||_2 \tag{5.1}$$

$$||v_{true} - v_{mod}||_2 \tag{5.2}$$

Note that this is the absolute error, as opposed to the relative error. The absolute error is most appropriate because of the need to know the accuracy of the rigid body (i.e., the actual distance the missile landed from its target). For completeness, however, the relative error is also reported in Tables 5.2 and 5.1.

Although the tables reflect the results from only one corner of the V-gamma map (steep dive, low velocity), similar results were obtained for the other three corners (see Appendix B). Generally, the smaller the integration step size, the more accurate the solution, since more equations are being solved to determine the trajectory. Therefore, in order to compare

Table 5.2: Position error (in feet) at 100,000 feet.

	Unmodified		Alternative 1		Alternative 2	
Step	Absolute	Relative	Absolute	Relative	Absolute	Relative
1/16	$0.00 \cdot 10^{0}$	$0.00 \cdot 10^{0}$	$2.74 \cdot 10^{-1}$	$1.30 \cdot 10^{-8}$	$1.92 \cdot 10^{-1}$	$9.14 \cdot 10^{-9}$
1/8	$7.17 \cdot 10^{-3}$	$3.42 \cdot 10^{-10}$	$2.73 \cdot 10^{-1}$	$1.30 \cdot 10^{-8}$	$2.72 \cdot 10^{-1}$	$1.30 \cdot 10^{-8}$
1	unstable	unstable	$2.77 \cdot 10^{-1}$	$1.32 \cdot 10^{-8}$	$1.05 \cdot 10^{0}$	$5.02 \cdot 10^{-8}$
5/4	unstable	unstable	$2.69 \cdot 10^{-1}$	$1.28 \cdot 10^{-8}$	unstable	unstable
4	unstable	unstable	$4.24 \cdot 10^{-1}$	$2.02 \cdot 10^{-8}$	unstable	unstable
8	unstable	unstable	$8.69 \cdot 10^{-1}$	$4.14 \cdot 10^{-8}$	unstable	unstable
10	unstable	unstable	$1.30 \cdot 10^{1}$	$6.19 \cdot 10^{-7}$	unstable	unstable
24	unstable	unstable	unstable	unstable	unstable	unstable

the two alternative methods with the original, the unmodified model was used to compute the trajectory at two step sizes, one of which was equal to one half of the other. Note that the unmodified method becomes unstable above an eighth of a second, the second alternative becomes unstable above one second, and the first alternative becomes unstable at 24 seconds. However, even for the first alternative, unacceptable inaccuracy begins creeping in at a step size of ten seconds. If the step size is increased further, the error will bounce around crazily until the region of instability is reached.

Table 5.3 presents the execution times in CPU seconds for each of the methods. A VAX C timer function was used to compute the execution times. Note that there seems to be some extra overhead with the angular momentum transformation (e.g., going from a step size of an eighth of a second to one second only makes the model seven times faster, not eight). Similarly for the $\dot{q}$ transformation.

After 100,000 feet, the two transformation alternatives slowly stray from the unmodified model. This may be due to the change in the airflow around the rigid body from laminar (smooth and predictable) to turbulent. This is reflected in the model by changing the computation of the aerodynamic forces and moments. The solution to this problem is therefore to switch methods, from one of the alternative methods to the original at or

Table 5.3: Execution time ending the simulation at 100,000 feet.

Step size (sec)	Unmodified	Alternative 1	Alternative 2
1/16	3.20	3.34	3.28
1/8	1.59	1.61	1.67
1	unstable	0.23	0.22
5/4	unstable	0.17	unstable
4	unstable	0.06	unstable
8	unstable	0.04	unstable
10	unstable	0.03	unstable
24	unstable	unstable	unstable

slightly below 100,000 feet. If the formulation overshoots 100,000 feet by a small amount (maybe a few thousand feet), then integration can simply be continued. If the formulation overshoots 100,000 feet by a large amount, the integration can be backed up, the state vector can be interpolated based on the current and previous altitudes, and then the integration can be restarted. Since most of the trajectory is above 100,000 feet for problems of interest, the cost of switching to the slower method at low altitudes should not be too great. In the example computation, the time of flight from 100,000 feet to impact is only 20 percent of the entire flight regime for the steep dive, low velocity V-gamma case.

Table 5.4 shows the error in velocity and position using the alternative methods to 100,000 feet and then switching to the original formulation for the remainder of the trajectory. The rigid body is considered to re-enter the atmosphere at 500,000 feet, so the alternative methods can be used on 80 percent of the trajectory. The errors were obtained by comparing the values of the position and velocity vectors of each of the labeled methods with the unmodified formulation at a step size of one sixteenth of a second. The unmodified formulation presented in the table used a step size of an eighth of a second and is labeled as Unmodified. Using the angular momentum transformation to 100,000 feet and then switching to the unmodified formulation is labeled as hybrid method 1. Above 100,000 feet hybrid method 1 uses a step size of 8.0 seconds. Using the $\dot{q}$ transformation to 100,000 feet and

Table 5.4: Position and velocity errors at sea level for the different methods.

Method	Hybrid method 1	Hybrid method 2	Unmodified formulation
velocity error	$3.25 \cdot 10^0$	$6.48 \cdot 10^0$	$9.99 \cdot 10^{-1}$
position error	$2.43 \cdot 10^{-1}$	$8.56 \cdot 10^{-1}$	$1.41 \cdot 10^0$
execution time	0.44	0.63	2.00

then switching to the unmodified formulation is labeled as hybrid method 2. Above 100,000 feet hybrid method 2 uses a step size of one second. The quaternion derivative transformation is ill-conditioned for step sizes above one second. This may be because the oscillation of the pitch/yaw amplitude caused problems in the quaternion derivatives, while the angular momentum pitch/yaw amplitude remains fairly steady due to partial conservation of angular momentum. From Table 5.4 it is evident that both hybrid methods outperform the unmodified formulation in both time and the accuracy in position. The improvement in accuracy may be because the hybrid methods computed the orientation and angular rate variables better than the unmodified formulation presented in the table by fitting a sine wave to the oscillating state variables.

Potentially, even larger step sizes could be taken at higher altitudes. Tables 5.5 and 5.6 show the results of further increasing the step sizes above 300,000 and 400,000 feet, respectively, for the angular momentum transformation. Below 300,000 feet, both tables have a step size of 8.0 seconds, and below 100,000 feet both tables switch to the unmodified formulation. Between 400,000 and 300,000 feet, Table 5.6 uses a step size of 12.0 seconds. Note that there was no noticeable improvement in execution time for either of the tables, possibly due to the fairly coarse granularity of the timer function. Increasing the step size was also attempted for altitudes between 200,000 and 300,000 feet. However, even a slight increase of the step size to ten seconds resulted in a velocity error of 39.3 feet per second, and a position error of 130.8 feet, which is simply unacceptable. A step size of 18.0 seconds above 400,000 feet and/or a step size of 16.0 seconds above 300,000 feet causes the solution of the differential equations to move into the unacceptable range. So, for the example con-

Table 5.5: Error in position and velocity at sea level using the angular momentum transformation with variable step sizes above 300,000 feet.

Method	Step size of 12.0	Step size of 16.0
velocity error	$1.64 \cdot 10^1$	$1.05 \cdot 10^1$
position error	$1.96 \cdot 10^0$	$7.54 \cdot 10^0$
execution time	0.44	0.44

Table 5.6: Error in position and velocity errors at sea level using the angular momentum transformation with variable step sizes above 400,000 feet.

Method	Step size of 16.0	Step size of 18.0
velocity error	$1.57 \cdot 10^1$	$1.67 \cdot 10^1$
position error	$1.28 \cdot 10^0$	$5.96 \cdot 10^0$
execution time	0.44	0.44

sidered here, it is best to use a step size of 16.0 seconds above 400,000 feet, a step size of 12.0 seconds between 300,000 and 400,000 feet, a step size of 8.0 seconds between 100,000 and 300,000 feet, and a step size of 0.125 seconds between sea level and 100,000 feet.

The $\dot{q}$ transformation becomes ill-conditioned at step sizes above 1.0 second even above 400,000 feet, so there is no point in trying to use larger step sizes at higher altitudes. However, even though the $\dot{q}$ transformation is ill-conditioned above step sizes of 1.0 second, it is more than three times as fast as the unmodified formulation when combined with the unmodified formulation below 100,000 feet, and more than seven times as fast as the unmodified formulation above 100,000 feet.

It is not all that unrealistic to know when to switch formulations. In this case, turbulent airflow is represented by a change in the aerodynamic coefficients at 100,000 feet. So if a transformation does not work, it is fairly simple to trace where it fails by printing out the values of the state variables at 100,000 feet and at impact. Another area where the transformation could fail is when the inertial forces overwhelm the viscous forces. This is also easily determined, and in an actual model would be at approximately 100,000 feet. It is not all that unrealistic to know when to switch step sizes, either. This is often done by trial and error, but switching the step size every 50,000 to 100,000 feet is reasonable.

The angular momentum transformation performs even better than the $\dot{q}$ transformation, being more than fifty times as fast above 100,000 feet, and approximately 4.5 times as fast as the unmodified formulation alone when combined with the unmodified formulation below 100,000 feet. The angular momentum transformation was also nearly 100 times as fast above 300,000 feet, and more than 100 times as fast above 400,000 feet. This is a very significant success, and at least for this 6-DOF model, the angular momentum transformation is the superior choice. A word of warning for readers who want to attempt this on an actual 6-DOF model, however: the angular momentum transformation will not necessarily be better in all cases, and interested readers are strongly encouraged to try both the angular momentum and $\dot{q}$ transformations, and make their own determinations.

Chapter 6

Conclusions and Summary

This project has determined that the execution time of 6-DOF models can be reduced significantly by transforming the differential equations to a form that can be numerically integrated more efficiently. Two methods were successful.

1. Integrate the spin angular momentum, along with the pitch/yaw angular momentum amplitude and phase, and integrate the phases and amplitudes of the quaternions by pairing them together and assuming each element of the pair is 90 degrees out of phase with the other above 100,000 feet. Using this method, the step size can be made 64 times larger. After 100,000 feet the formulation must go back to the original equations. Overall, the computation time was reduced by a factor of four and a half.

2. Integrate the phases and amplitudes of the quaternions by pairing them together and assuming each element of the pair is 90 degrees out of phase with the other, and integrating the derivatives of the quaternion amplitudes and phases above 100,000 feet. Using this method, the step size can be made eight times larger. After 100,000 feet the formulation must go back to the original equations. With this approach, the total simulation time was reduced by a factor of three.

The methods do not work quite as well below 100,000 feet due to the turbulence of the lower part of the atmosphere. This does not quite get a 6-DOF model up to the speed of a 3-DOF model, but it still represents a significant improvement over the old 6-DOF model.

To sum up, there are three main points to consider when comparing the old 6-DOF model and the semi-periodic quaternion models:

1. The transformations required are relatively simple to make. Essentially, a sine wave is being fitted to the oscillating function.

2. The transformed models yield accuracy similar to the old model (the position difference between the models is within one or two feet).

3. The transformed models yield significant improvements in execution time (3.0 to 4.5 times as fast).

As to why this apparently has never been attempted before, it should again be noted that angular motion can often seem counter-intuitive, and that journal articles generally deal with 3-DOF approximations of 6-DOF models rather than with 6-DOF models directly, and that re-entry simulations represent a unique situation in the rigid body world in that there are few characteristics (e.g., thrust) present in missiles that can be used in approximating the angular motion of re-entry bodies. Hopefully, research such as this can aid in bridging the gap between mathematics and engineering.

Appendix A
The Algorithm

The algorithm consists of four subroutines: the initialization/main routine, the integration subroutine, the coordination transformation subroutine, and the equations of motion subroutine, which handles the aerodynamic forces and moments.

When the initialization is complete, the algorithm goes into a while loop. For each iteration of the loop, the integration subroutine is called, and then the rigid body's altitude is checked to determine if it has gone below sea level. If so, and if the rigid body's altitude is not within one foot of sea level, the integration is backed up, the step size is interpolated based on the rigid body's current and previous altitudes, and the integration is restarted. When the missile's altitude comes within one foot of sea level, the program is halted.

The integration subroutine first calls the quaternion subroutine and then the equations of motion subroutine. The quaternion subroutine is given below.

A.1 Initialization

First, initialize the step size (sec), the rigid body's speed (ft/s), mass (slugs), length (ft), and diameter (ft).

$$\begin{aligned}\Delta t &= 0.125\\ v_{rb} &= v_0\\ m_{rb} &= 50.0\\ l_{rb} &= 8.0\\ d_{rb} &= 2.0\end{aligned}$$

Next, compute the rigid body's half cone angle (rad).

$$\theta_h = \arctan\left(\frac{d_{rb}}{2l_{rb}}\right)$$

Next, compute the rigid body's surface area (ft^2).

$$s_{rb} = \pi\sqrt{l_{rb}^2 + \frac{d_{rb}^2}{2}}$$

Initialize the rigid body's angular velocity (rad/s).

$$\omega_{rb} = \begin{pmatrix} 20.0 \\ 0.05 \\ -0.03 \end{pmatrix}$$

Initialize the rigid body's latitude (rad).

$$\phi = 0.0$$

Initialize the rigid body's longitude (rad).

$$\lambda = 0.0$$

Initialize the rigid body's azimuth (rad). This is the heading of the missile with respect to due north.

$$A_z = 0.0$$

Initialize the re-entry or flight path angle (rad).

$$\gamma = \frac{\gamma_0 \pi}{180.0}$$

Initialize the rigid body's angle of attack (rad). This is the angle between the direction the missile is pointed in and the velocity vector.

$$\alpha = 0.0$$

Initialize the radius of the earth (ft).

$$r_e = 2.089997 \cdot 10^7$$

Initialize the rigid body's altitude (ft).

$$h_{rb} = 5.0 \cdot 10^5$$

Initialize the rigid body's position vector in the geocentric frame (ft).

$$\vec{P}_{gc} = \begin{pmatrix} (h_{rb} + r_e)\cos\phi\cos\lambda \\ (h_{rb} + r_e)\cos\phi\sin\lambda \\ (h_{rb} + r_e)\sin\phi \end{pmatrix}$$

Initialize the rigid body's velocity vector in the geocentric frame (ft/s).

$$\vec{V}_{gc} = v_{rb} \begin{pmatrix} \cos\phi\cos\lambda\sin\gamma - \cos\gamma\sin\lambda\sin A_z - \cos\gamma\cos\lambda\sin\phi\cos A_z \\ \cos\phi\sin\lambda\sin\gamma + \cos\gamma\cos\lambda\sin A_z - \cos\gamma\cos\lambda\sin\phi\cos A_z \\ \sin\phi\sin\gamma + \cos\gamma\sin A_z\cos\phi \end{pmatrix}$$

Initialize the geodetic vertical unit vector.

$$\vec{h}_{gc} = \begin{pmatrix} \cos\phi\cos\lambda \\ \cos\phi\sin\lambda \\ \sin\phi \end{pmatrix}$$

Compute the unit vector of the geocentric velocity vector.

$$\vec{u}_1 = \frac{\vec{V}_{gc}}{v_{rb}}$$

Compute a second unit vector.

$$\vec{u}_2 = \frac{\vec{u}_1 \times \vec{h}_{gc}}{||\vec{u}_1 \times \vec{h}_{gc}||_2}$$

Compute a third unit vector.

$$\vec{u}_3 = \frac{\vec{u}_1 \times \vec{u}_2}{||\vec{u}_1 \times \vec{u}_2||_2}$$

Initialize the geocentric to velocity transformation matrix.

$$T_{gc2v} = (\vec{u}_1, \vec{u}_2, \vec{u}_3)$$

Initialize the velocity to missile rotational matrix using α.

$$T_{v2m} = T(\alpha)_y$$

Initialize the geocentric to missile transformation matrix.

$$T_{gc2m} = T_{v2m} T_{gc2v}$$

Initialize the missile to geocentric transformation matrix.

$$T_{m2gc} = T_{gc2m}^T$$

Initialize the Euler angles (rad).

$$\begin{aligned}
\theta_3 &= \arctan\left(\frac{T_{gc2m_{31}}}{T_{gc2m_{11}}}\right) \\
\theta_1 &= \arctan\left(\frac{T_{gc2m_{23}}}{T_{gc2m_{22}}}\right) \\
\theta_2 &= \arctan\left(\frac{\cos\theta_1 T_{gc2m_{21}}}{T_{gc2m_{22}}}\right)
\end{aligned}$$

Initialize the quaternion vector.

$$\begin{aligned}
q_1 &= \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\cos\frac{\theta_3}{2} + \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\sin\frac{\theta_3}{2} \\
q_2 &= \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\sin\frac{\theta_3}{2} + \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\cos\frac{\theta_3}{2} \\
q_3 &= \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\cos\frac{\theta_3}{2} + \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\sin\frac{\theta_3}{2} \\
q_4 &= \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\cos\frac{\theta_3}{2} + \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\sin\frac{\theta_3}{2}
\end{aligned}$$

Initialize the moment of inertia matrix (ft^2 slugs).

$$I_{rb} = \begin{pmatrix} 8.0 & 0.0 & 0.0 \\ 0.0 & 63.9 & 0.0 \\ 0.0 & 0.0 & 64.1 \end{pmatrix}$$

Initialize the gravitational constant (lb ft^2 slug^{-1}).

$$G = 3.438191 \cdot 10^{-8}$$

Initialize the earth's mass (slugs).

$$m_e = 4.105551 \cdot 10^{23}$$

Initialize the inverse moment of inertia matrix (ft^{-2} slugs^{-1}).

$$I_{rb}^{-1} = (I_{rb})^{-1}$$

If using the angular momentum transformation, compute the initial angular momentum, and pitch/yaw amplitude and phase.

$$\begin{aligned} \vec{H} &= I_{rb}\omega_{rb} \\ A_h &= \sqrt{H_2^2 + H_3^2} \\ P_h &= \arctan\left(\frac{H_2}{H_3}\right) \end{aligned}$$

If not using the unmodified formulation, compute the initial amplitudes and phases of the quaternions.

$$\begin{aligned} A_1(0) &= \sqrt{q_1^2(0) + q_4^2(0)} \\ A_2(0) &= \sqrt{q_2^2(0) + q_3^2(0)} \\ P_1(0) &= \arctan\left(\frac{q_1(0)}{q_4(0)}\right) \\ P_2(0) &= \arctan\left(\frac{q_3(0)}{q_2(0)}\right) \end{aligned}$$

For details on handling singularities, see Section 3.3. If using the $\dot{q}$ transformation, compute the initial quaternion derivative, and the derivatives of the quaternion phases and amplitudes.

$$Q = \begin{pmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{pmatrix}$$

$$\begin{aligned}
\dot{q} &= \frac{1}{2}Q\omega_{rb} \\
\dot{A}_1 &= \frac{q_1\dot{q}_4 + \dot{q}_1 q_4}{A_1} \\
\dot{A}_2 &= \frac{q_2\dot{q}_3 + \dot{q}_2 q_3}{A_2} \\
\dot{P}_1 &= \frac{q_4\dot{q}_1 - \dot{q}_4 q_1}{A_1^2} \\
\dot{P}_2 &= \frac{q_2\dot{q}_3 - \dot{q}_2 q_3}{A_2^2}
\end{aligned}$$

For details on handling singularities, see Section 3.3. Initialize the state vector. If using the unmodified formulation use the following state vector.

$$SV_i = \vec{V}_{rb}, i = 1,2,3$$

$$SV_i = \vec{P}_{rb}, i = 4,5,6$$

$$SV_i = \omega_{rb}, i = 7,8,9$$

$$SV_i = q, i = 10,11,12,13$$

If using the angular momentum transformation, use the following state vector.

$$SV_i = \vec{V}_{rb}, i = 1,2,3$$

$$SV_i = \vec{P}_{rb}, i = 4,5,6$$

$$\begin{aligned}
SV_7 &= H_1 \\
SV_8 &= A_h \\
SV_9 &= P_h \\
SV_{10} &= A_1 \\
SV_{11} &= A_2 \\
SV_{12} &= P_1 \\
SV_{13} &= P_2
\end{aligned}$$

If using the $\dot{q}$ transformation, use the following state vector.

$$SV_i = \vec{V}_{rb}, i = 1,2,3$$

$$SV_i = \vec{P}_{rb}, i = 4,5,6$$

$$\begin{aligned} SV_7 &= \dot{A}_1 \\ SV_8 &= \dot{A}_2 \\ SV_9 &= \dot{P}_1 \\ SV_{10} &= \dot{P}_2 \\ SV_{11} &= A_1 \\ SV_{12} &= A_2 \\ SV_{13} &= P_1 \\ SV_{14} &= P_2 \end{aligned}$$

A.2 Integration

The integration subroutine is a fourth order Runge-Kutta, the equations for which are as follows:

$$\begin{aligned} \vec{k}_1 &= \vec{f}(t, \vec{y}_i) \\ \vec{k}_2 &= \vec{f}(t + \frac{\Delta t}{2.0}, \vec{y}_i + \vec{k}_1 \frac{\Delta t}{2.0}) \\ \vec{k}_3 &= \vec{f}(t + \frac{\Delta t}{2.0}, \vec{y}_i + \vec{k}_2 \frac{\Delta t}{2.0}) \\ \vec{k}_4 &= \vec{f}(t + \Delta t, \vec{y}_i + \vec{k}_3 \Delta t) \\ \vec{y}_{i+1} &= \vec{y}_i + \frac{1}{6}\Delta t(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) \end{aligned}$$

The derivatives of the state vector are computed for each k_i.

A.3 Coordinate Transformations

If the standard formulation is being used, get q and ω_{rb} from the state vector. Otherwise get the quaterion amplitudes and phases from the state vector and compute the updated quaternion.

$$\vec{q} = \begin{pmatrix} A_1(t) \cdot \sin(P_1(t)) \\ A_2(t) \cdot \sin(P_2(t)) \\ A_3(t) \cdot \sin(P_3(t)) \\ A_4(t) \cdot \sin(P_4(t)) \end{pmatrix}$$

If the angular momentum transformation is being used then get the spin angular momentum and pitch/yaw angular momentum amplitude and phase from the state vector, and compute the updated angular momentum and velocity.

$$\begin{aligned} \vec{H} &= \begin{pmatrix} H_1 \\ A_3(t)\sin(P_3(t)) \\ A_3(t)\cos(P_3(t)) \end{pmatrix} \\ \omega_{rb} &= I_{rb}^{-1}\vec{H} \end{aligned}$$

Compute the matrix Q.

$$Q = \begin{pmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{pmatrix}$$

If the $\dot{q}$ transformation is being used then get the quaternion amplitude and phase derivatives from the state vector, compute $\dot{q}$, and compute the updated angular velocity.

$$\begin{aligned} \dot{q} &= \begin{pmatrix} \dot{A}_1 \sin P_1 + A_1\dot{P}_1 \cos P_1 \\ \dot{A}_2 \cos P_2 - A_2\dot{P}_2 \sin P_2 \\ \dot{A}_2 \sin P_2 + A_2\dot{P}_2 \cos P_2 \\ \dot{A}_1 \cos P_1 - A_1\dot{P}_1 \sin P_1 \end{pmatrix} \\ \omega_{rb} &= 2Q^T\dot{q} \end{aligned}$$

Compute the updated transformation matrices.

$$\begin{aligned} T_{gc2m} &= \begin{pmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{pmatrix} \\ T_{m2gc} &= T_{gc2m}^T \end{aligned}$$

If using the standard formulation or the angular momentum transformation, compute $\dot{q}$.

$$\dot{q} = \frac{1}{2}Q\omega_{rb}$$

If using the standard formulation, place $\dot{q}$ into the state vector. If using the angular momentum transformation, compute the quaternion amplitude and phase derivatives.

$$\begin{aligned} \dot{A}_1 &= \frac{q_1\dot{q}_4 + \dot{q}_1q_4}{A_1} \\ \dot{A}_2 &= \frac{q_2\dot{q}_3 + \dot{q}_2q_3}{A_2} \\ \dot{P}_1 &= \frac{q_4\dot{q}_1 - \dot{q}_4q_1}{A_1^2} \\ \dot{P}_2 &= \frac{q_2\dot{q}_3 - \dot{q}_2q_3}{A_2^2} \end{aligned}$$

For details on handling singularities, see section 3.3. If not using the standard formulation, place the quaternion amplitude and phase derivatives into the state vector.

A.4 Equations of Motion

First, get the position and velocity vectors from the state vector. Compute the altitude (ft).

$$a_{rb} = ||P_{gc}||_2 - r_e$$

Compute the air density (slugs/ft^3).

$$\rho = \frac{2.5033163 \cdot 10^{-6}}{2^{\frac{a_{rb}}{18000.0}}}$$

Compute the gravitational acceleration in the geocentric frame (ft/s^2).

$$\vec{g}_{gc} = -\frac{\vec{P}_{gc} m_e G}{||\vec{P}||_2^3}$$

Compute the dynamic pressure on the rigid body (lbs/ft^2).

$$\bar{q} = 0.5\rho ||V_{gc}||_2^2$$

Compute the velocity in the rigid body frame (ft/s).

$$\vec{V}_m = T_{gc2m} \vec{V}_{gc}$$

Compute the angle of attack (rad).

$$\alpha = \arctan\left(\frac{\sqrt{V_{m_2}^2 + V_{m_3}^2}}{V_{m_1}}\right)$$

Compute the velocity to missile rotational angle (rad).

$$\theta_v = \arctan\left(\frac{V_{m_2}}{V_{m_3}}\right)$$

Compute the axial force coefficient. This represents the air flow over the rigid body's nosecone.

$$C_a = 2\sin^2\theta_h + \sin^2\alpha(1.0 - 3.0\sin\alpha)$$

Compute the normal force coefficient. This represents the air flow perpendicular to the rigid body, and is also referred to as lift.

$$C_n = \cos^2\theta_h \sin 2\alpha$$

Compute the pitching moment coefficient. This moment arises from the normal force.

$$C_p = -\frac{1.0}{3000.0}\tan\theta_h \sin 2\alpha$$

Compute the pitch damping moment coefficient. Just as a damping force causes a spring to attempt to snap back to its original position, a pitch damping moment causes a rigid body to attempt to snap back to its original orientation.

$$C_d = -\frac{\sin\alpha}{\sin^2\theta_h}$$

If the angle of attack is ten degrees or more, then compute the pitch damping moment differently.

$$C_d = -\frac{\sin\frac{\pi}{18.0}}{\sin^2\theta_h}$$

If the rigid body's altitude is less than 100,000 feet, then multiply all four coefficients by ten. This is done to simulate turbulence. Then compute the aerodynamic force vector in the missile frame (lbs).

$$\vec{F}_{rb} = -\bar{q}s_{rb}\begin{pmatrix} C_a \\ C_n\sin\theta_v \\ C_n\cos\theta_v \end{pmatrix}$$

If not using the angular momentum transformation, compute the angular momentum (ft^2 slug / sec) in the rigid body frame.

$$\vec{H} = I_{rb}\omega_{rb}$$

Compute the derivative of the angular momentum (ft^2 slug / s^2) in the rigid body frame.

$$\dot{\vec{H}} = \bar{q}s_{rb}d_{rb}\begin{pmatrix} 0.0 \\ \frac{C_d\omega_2 d_{rb}}{\|V_{gc}\|_2} + C_p\cos\theta_v \\ \frac{C_d\omega_3 d_{rb}}{\|V_{gc}\|_2} + C_p\sin\theta_v \end{pmatrix} - \omega_{rb}\times\vec{H}$$

If not using the angular momentum transformation, compute the angular acceleration (rad/s^2).

$$\dot{\omega}_{rb} = I_{rb}^{-1}\dot{\vec{H}}$$

Compute the acceleration in the geocentric frame (ft/s^2).

$$\vec{A}_{gc} = \vec{g}_{gc} + T_{m2gc}\frac{\vec{F}_m}{m_{rb}}$$

Place the acceleration and velocity vectors in the state vector. If using the unmodified formulation, place the angular acceleration vector in the state vector. If using the angular momentum transformation, compute the pitch/yaw amplitude and phase derivatives, and place them in the state vector along with the derivative of the spin angular momentum.

$$\begin{aligned}
\dot{A}_h &= \frac{H_2\dot{H}_3 + \dot{H}_2 H_3}{A_H} \\
\dot{P}_h &= \frac{H_3\dot{H}_2 - \dot{H}_3 H_2}{A_H^2}
\end{aligned}$$

If using the $\dot{q}$ transformation, compute the second derivatives of the quaternions, then compute the second derivatives of the quaternion amplitudes and phases, and place them in the state vector.

$$\begin{aligned}
\ddot{q} &= \frac{1}{2}Q\dot{\omega}_{rb} + \frac{1}{2}\dot{Q}\omega_{rb} \\
\ddot{A}_1 &= \frac{(q_1\ddot{q}_1 + \dot{q}_1^2 + q_4\ddot{q}_4 + \dot{q}_4^2)A_1 - (q_1\dot{q}_1 + q_4\dot{q}_4)\dot{A}_1}{A_1^2} \\
\ddot{A}_2 &= \frac{(q_2\ddot{q}_2 + \dot{q}_2^2 + q_3\ddot{q}_3 + \dot{q}_3^2)A_2 - (q_2\dot{q}_2 + q_3\dot{q}_3)\dot{A}_2}{A_2^2} \\
\ddot{P}_1 &= \frac{(q_4\ddot{q}_1 - \ddot{q}_4 q_1)A_1 - (q_4\dot{q}_1 - \dot{q}_4 q_1)2\dot{A}_1}{A_1^3} \\
\ddot{P}_2 &= \frac{(q_2\ddot{q}_3 - \ddot{q}_2 q_3)A_2 - (q_2\dot{q}_3 - \dot{q}_2 q_3)2\dot{A}_2}{A_2^3}
\end{aligned}$$

For details on handling singularities, see Section 4.2. At this point, control is turned back over to the integration subroutine, and the appropriate state variables are updated.

Appendix B

Results from Other Test Cases

This appendix contains results from the other three test cases from the V-gamma map (steep dive, high velocity; shallow dive, low velocity; shallow dive, high velocity). The tables for each test case are the same ones used in Chapter 5. One should note the similarities among all test cases. For example, the unmodified formulation is always unstable for a step size above one eighth of a second, the $\dot{q}$ transformation is always unstable for a step size above one second, and the angular momentum transformation is always unstable for a step size of 24 seconds. Also, unacceptable errors creep into the angular momentum transformation solution for a step size of ten seconds or more. Finally, the percentage reduction in execution time is approximately the same for the two transformations in all test cases.

The tables below contain data for the following three cases:

Case 2. Steep dive, high velocity, Tables B.1–B.6.

Case 3. Shallow dive, low velocity Tables B.7–B.12.

Case 4. Shallow dive, high velocity Tables B.13–B.18.

Table B.1: Case 2: Velocity error (in feet/sec) at 100,000 feet.

	Unmodified		Alternative 1		Alternative 2	
Step	**Absolute**	**Relative**	**Absolute**	**Relative**	**Absolute**	**Relative**
1/16	$0.00 \cdot 10^{0}$	$0.00 \cdot 10^{0}$	$1.69 \cdot 10^{-1}$	$1.07 \cdot 10^{-5}$	$1.28 \cdot 10^{-1}$	$8.06 \cdot 10^{-6}$
1/8	$8.00 \cdot 10^{-5}$	$5.06 \cdot 10^{-9}$	$1.69 \cdot 10^{-1}$	$1.07 \cdot 10^{-5}$	$2.64 \cdot 10^{-2}$	$1.67 \cdot 10^{-6}$
1	unstable	unstable	$1.69 \cdot 10^{-1}$	$1.0700 \cdot 10^{-5}$	$3.98 \cdot 10^{-2}$	$2.52 \cdot 10^{-6}$
5/4	unstable	unstable	$1.69 \cdot 10^{-1}$	$1.07 \cdot 10^{-5}$	unstable	unstable
4	unstable	unstable	$5.20 \cdot 10^{-1}$	$3.29 \cdot 10^{-5}$	unstable	unstable
8	unstable	unstable	$4.13 \cdot 10^{-1}$	$2.61 \cdot 10^{-5}$	unstable	unstable
10	unstable	unstable	$1.41 \cdot 10^{2}$	$8.92 \cdot 10^{-3}$	unstable	unstable
24	unstable	unstable	unstable	unstable	unstable	unstable

Table B.2: Case 2: Position error (in feet) at 100,000 feet.

	Unmodified		Alternative 1		Alternative 2	
Step	**Absolute**	**Relative**	**Absolute**	**Relative**	**Absolute**	**Relative**
1/16	$0.00 \cdot 10^{0}$	$0.00 \cdot 10^{0}$	$8.87 \cdot 10^{-4}$	$4.22 \cdot 10^{-11}$	$7.70 \cdot 10^{-4}$	$3.65 \cdot 10^{-9}$
1/8	$2.61 \cdot 10^{-2}$	$1.24 \cdot 10^{-9}$	$2.64 \cdot 10^{-2}$	$1.26 \cdot 10^{-9}$	$2.64 \cdot 10^{-2}$	$1.26 \cdot 10^{-9}$
1	unstable	unstable	$1.35 \cdot 10^{-2}$	$6.44 \cdot 10^{-10}$	$3.98 \cdot 10^{-2}$	$1.89 \cdot 10^{-9}$
5/4	unstable	unstable	$1.15 \cdot 10^{-2}$	$5.46 \cdot 10^{-10}$	unstable	unstable
4	unstable	unstable	$4.24 \cdot 10^{-2}$	$2.02 \cdot 10^{-8}$	unstable	unstable
8	unstable	unstable	$6.29 \cdot 10^{-1}$	$2.99 \cdot 10^{-8}$	unstable	unstable
10	unstable	unstable	$1.07 \cdot 10^{2}$	$5.09 \cdot 10^{-6}$	unstable	unstable
24	unstable	unstable	unstable	unstable	unstable	unstable

Table B.3: Case 2: Execution time ending the simulation at 100,000 feet.

Step size (sec)	Unmodified	Alternative 1	Alternative 2
1/16	2.07	2.20	2.18
1/8	1.05	1.16	1.07
1	unstable	0.17	0.16
5/4	unstable	0.15	unstable
4	unstable	0.05	unstable
8	unstable	0.03	unstable
10	unstable	0.03	unstable
24	unstable	unstable	unstable

Table B.4: Case 2: Position and velocity errors at sea level for the different methods.

Method	Hybrid method 1	Hybrid method 2	Unmodified formulation
velocity error	$3.78 \cdot 10^0$	$7.69 \cdot 10^0$	$1.35 \cdot 10^0$
position error	$6.32 \cdot 10^{-1}$	$1.58 \cdot 10^0$	$4.81 \cdot 10^{-1}$
execution time	0.30	0.43	1.38

Table B.5: Case 2: Error in position and velocity at sea level using the angular momentum transformation with variable step sizes above 300,000 feet.

Method	Step size of 12.0	Step size of 16.0
velocity error	$7.70 \cdot 10^0$	$1.75 \cdot 10^1$
position error	$1.23 \cdot 10^0$	$1.27 \cdot 10^1$
execution time	0.30	0.30

Table B.6: Case 2: Error in position and velocity at sea level using the angular momentum transformation with variable step sizes above 400,000 feet.

Method	Step size of 16.0	Step size of 18.0
velocity error	$7.50 \cdot 10^0$	$1.68 \cdot 10^1$
position error	$1.36 \cdot 10^0$	$1.42 \cdot 10^1$
execution time	0.30	0.30

Table B.7: Case 3: Velocity error (in feet/sec) at 100,000 feet.

	Unmodified		Alternative 1		Alternative 2	
Step	Absolute	Relative	Absolute	Relative	Absolute	Relative
1/16	$0.00 \cdot 10^{0}$	$0.00 \cdot 10^{0}$	$7.49 \cdot 10^{-1}$	$4.7344 \cdot 10^{-5}$	$2.09 \cdot 10^{-1}$	$1.3199 \cdot 10^{-5}$
1/8	$1.27 \cdot 10^{-4}$	$8.03 \cdot 10^{-9}$	$7.38 \cdot 10^{-1}$	$4.67 \cdot 10^{-5}$	$2.09 \cdot 10^{-1}$	$1.32 \cdot 10^{-5}$
1	unstable	unstable	$7.42 \cdot 10^{-1}$	$4.69 \cdot 10^{-5}$	$2.12 \cdot 10^{-1}$	$1.34 \cdot 10^{-5}$
5/4	unstable	unstable	$7.58 \cdot 10^{-1}$	$4.79 \cdot 10^{-5}$	unstable	unstable
4	unstable	unstable	$7.93 \cdot 10^{-1}$	$5.01 \cdot 10^{-5}$	unstable	unstable
8	unstable	unstable	$2.55 \cdot 10^{0}$	$1.61 \cdot 10^{-4}$	unstable	unstable
10	unstable	unstable	$1.72 \cdot 10^{1}$	$1.09 \cdot 10^{-3}$	unstable	unstable
24	unstable	unstable	unstable	unstable	unstable	unstable

Table B.8: Case 3: Position error (in feet) at 100,000 feet.

	Unmodified		Alternative 1		Alternative 2	
Step	Absolute	Relative	Absolute	Relative	Absolute	Relative
1/16	$0.00 \cdot 10^{0}$	$0.00 \cdot 10^{0}$	$4.02 \cdot 10^{-1}$	$1.91 \cdot 10^{-8}$	$6.65 \cdot 10^{-2}$	$3.17 \cdot 10^{-9}$
1/8	$3.84 \cdot 10^{-2}$	$1.83 \cdot 10^{-9}$	$4.47 \cdot 10^{-1}$	$2.13 \cdot 10^{-8}$	$3.57 \cdot 10^{-2}$	$1.70 \cdot 10^{-9}$
1	unstable	unstable	$3.84 \cdot 10^{-1}$	$1.83 \cdot 10^{-8}$	$7.15 \cdot 10^{-2}$	$3.41 \cdot 10^{-9}$
5/4	unstable	unstable	$6.41 \cdot 10^{-1}$	$3.05 \cdot 10^{-8}$	unstable	unstable
4	unstable	unstable	$6.04 \cdot 10^{-1}$	$2.87 \cdot 10^{-8}$	unstable	unstable
8	unstable	unstable	$1.13 \cdot 10^{0}$	$5.39 \cdot 10^{-8}$	unstable	unstable
10	unstable	unstable	$5.05 \cdot 10^{1}$	$2.41 \cdot 10^{-6}$	unstable	unstable
24	unstable	unstable	unstable	unstable	unstable	unstable

Table B.9: Case 3: Execution time ending the simulation at 100,000 feet.

Step size (sec)	Unmodified	Alternative 1	Alternative 2
1/16	3.40	3.47	3.45
1/8	1.86	1.93	1.90
1	unstable	0.34	0.32
5/4	unstable	0.30	unstable
4	unstable	0.19	unstable
8	unstable	0.15	unstable
10	unstable	0.19	unstable
24	unstable	unstable	unstable

Table B.10: Case 3: Position and velocity errors at sea level for the different methods.

Method	Hybrid method 1	Hybrid method 2	Unmodified formulation
velocity error	$2.66 \cdot 10^{0}$	$3.80 \cdot 10^{0}$	$2.62 \cdot 10^{-0}$
position error	$5.63 \cdot 10^{-1}$	$6.32 \cdot 10^{-1}$	$6.88 \cdot 10^{-1}$
execution time	0.49	0.74	2.22

Table B.11: Case 3: Error in position and velocity at sea level using the angular momentum transformation with variable step sizes above 300,000 feet.

Method	Step size of 12.0	Step size of 16.0
velocity error	$1.37 \cdot 10^{0}$	$1.44 \cdot 10^{1}$
position error	$7.70 \cdot 10^{-1}$	$1.95 \cdot 10^{1}$
execution time	0.49	0.49

Table B.12: Case 3: Error in position and velocity at sea level using the angular momentum transformation with variable step sizes above 400,000 feet.

Method	Step size of 16.0	Step size of 18.0
velocity error	$1.43 \cdot 10^{0}$	$1.57 \cdot 10^{1}$
position error	$8.11 \cdot 10^{-1}$	$6.90 \cdot 10^{0}$
execution time	0.49	0.49

Table B.13: Case 4: Velocity error (in feet/sec) at 100,000 feet.

	Unmodified		Alternative 1		Alternative 2	
Step	Absolute	Relative	Absolute	Relative	Absolute	Relative
1/16	$0.00 \cdot 10^{0}$	$0.00 \cdot 10^{0}$	$8.13 \cdot 10^{-1}$	$5.14 \cdot 10^{-5}$	$3.01 \cdot 10^{-1}$	$1.90 \cdot 10^{-9}$
1/8	$1.34 \cdot 10^{-4}$	$8.47 \cdot 10^{-9}$	$8.11 \cdot 10^{-1}$	$5.13 \cdot 10^{-5}$	$2.52 \cdot 10^{-1}$	$1.59 \cdot 10^{-5}$
1	unstable	unstable	$8.22 \cdot 10^{-1}$	$5.19 \cdot 10^{-5}$	$3.03 \cdot 10^{-1}$	$1.92 \cdot 10^{-5}$
5/4	unstable	unstable	$8.42 \cdot 10^{-1}$	$5.33 \cdot 10^{-5}$	unstable	unstable
4	unstable	unstable	$1.74 \cdot 10^{0}$	$1.10 \cdot 10^{-4}$	unstable	unstable
8	unstable	unstable	$8.81 \cdot 10^{0}$	$5.57 \cdot 10^{-4}$	unstable	unstable
10	unstable	unstable	$4.76 \cdot 10^{1}$	$3.01 \cdot 10^{-3}$	unstable	unstable
24	unstable	unstable	unstable	unstable	unstable	unstable

Table B.14: Case 4: Position error (in feet) at 100,000 feet.

	Unmodified		Alternative 1		Alternative 2	
Step	Absolute	Relative	Absolute	Relative	Absolute	Relative
1/16	$0.00 \cdot 10^{0}$	$0.00 \cdot 10^{0}$	$2.54 \cdot 10^{-1}$	$1.21 \cdot 10^{-8}$	$2.74 \cdot 10^{-2}$	$1.31 \cdot 10^{-9}$
1/8	$2.01 \cdot 10^{-2}$	$9.57 \cdot 10^{-10}$	$2.38 \cdot 10^{-1}$	$1.13 \cdot 10^{-8}$	$2.61 \cdot 10^{-1}$	$1.24 \cdot 10^{-8}$
1	unstable	unstable	$8.22 \cdot 10^{-1}$	$3.91 \cdot 10^{-8}$	$3.36 \cdot 10^{-2}$	$1.60 \cdot 10^{-9}$
5/4	unstable	unstable	$3.03 \cdot 10^{-1}$	$1.44 \cdot 10^{-8}$	unstable	unstable
4	unstable	unstable	$1.13 \cdot 10^{0}$	$5.39 \cdot 10^{-8}$	unstable	unstable
8	unstable	unstable	$4.00 \cdot 10^{0}$	$1.90 \cdot 10^{-7}$	unstable	unstable
10	unstable	unstable	$1.17 \cdot 10^{2}$	$5.58 \cdot 10^{-6}$	unstable	unstable
24	unstable	unstable	unstable	unstable	unstable	unstable

Table B.15: Case 4: Execution time ending the simulation at 100,000 feet.

Step size (sec)	Unmodified	Alternative 1	Alternative 2
1/16	2.59	2.67	2.65
1/8	1.46	1.49	1.48
1	unstable	0.23	0.21
5/4	unstable	0.19	unstable
4	unstable	0.07	unstable
8	unstable	0.05	unstable
10	unstable	0.05	unstable
24	unstable	unstable	unstable

Table B.16: Case 4: Position and velocity errors at sea level for the different methods.

Method	Hybrid method 1	Hybrid method 2	Unmodified formulation
velocity error	$1.55 \cdot 10^{-1}$	$3.90 \cdot 10^{-1}$	$3.92 \cdot 10^{-4}$
position error	$4.87 \cdot 10^{-1}$	$1.57 \cdot 10^{0}$	$4.39 \cdot 10^{-2}$
execution time	0.41	0.59	1.82

Table B.17: Case 4: Error in position and velocity at sea level using the angular momentum transformation with variable step sizes above 300,000 feet.

Method	Step size of 12.0	Step size of 16.0
velocity error	$1.86 \cdot 10^{0}$	$1.98 \cdot 10^{1}$
position error	$4.45 \cdot 10^{-1}$	$1.24 \cdot 10^{1}$
execution time	0.41	0.41

Table B.18: Case 4: Error in position and velocity at sea level using the angular momentum transformation with variable step sizes above 400,000 feet.

Method	Step size of 16.0	Step size of 18.0
velocity error	$1.68 \cdot 10^{0}$	$1.93 \cdot 10^{1}$
position error	$1.08 \cdot 10^{0}$	$1.02 \cdot 10^{1}$
execution time	0.41	0.41

REFERENCES

[1] F. S. Acton, "Numerical Methods That Work," Harper & Row, New York, 1970, pps. 143-155.

[2] P. M. Bushong, J. A. Lawton, "Fast Solution to an Optimal Trajectory Problem," AAS/AIAA Spaceflight Mechanics Meeting, American Astronautical Society, February 1994, AAS Paper 94-131.

[3] P. Chartier, "L-Stable Parallel One-Block Methods for Ordinary Differential Equations," SIAM Journal of Numerical Analysis, Society for Industrial and Applied Mathematics, April 1994, pps. 552-574.

[4] P. W. Gaffney, "A Performance Evaluation of Some FORTRAN Subroutines for the Solution of Stiff Oscillatory Ordinary Differential Equations," ACM Transactions on Mathematical Software, Vol. 10, No. 1, March 1984, pps. 58-72.

[5] C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations," Prentice-Hall, Englewood Cliffs, N.J., 1971.

[6] K. Giannakoglou, P. Chaviaropoulos, K. D. Papailiou, "Computation of Rotational Transonic Flows Using a Decomposition Method," AIAA Journal, American Institute of Aeronautics and Astronautics, Inc., October 1988, pps. 1175-1180.

[7] S. L. Glashow, "Interactions," Warner Books, Inc., New York, 1988, pps. 155-157.

[8] M. K. Gordon, L. F. Shampine, "Computer Solution of Ordinary Differential Equations," W. H. Freeman & Co., San Francisco, 1975, pps. 126-145.

[9] S. I. Grossman, W. R. Derrick, "Advanced Engineering Mathematics", Harper & Row, New York, 1988, pps. 188-243.

[10] J. D. Lambert, "Computational Methods in Ordinary Differential Equations," John Wiley & Sons Ltd., New York, pps. 135-156; 217-249.

[11] R. A. Luke, "Software Design of 6-DOF Guidance Approximations in 3-DOF Ascent Simulations," SIMULATION, Simulation Councils, Inc., July 1993, pps. 22-35.

[12] E. Madelung, "On a Method for a Fast Numerical Solution of Differential Equations of Second Order," Z. Physik, 1931, Vol. 67, pps. 516-518.

[13] J. J. Martin, "Atmospheric Re-entry: An Introduction to its Science and Engineering," Prentice Hall, Tappan, N.J., 1966, pps. 23-82.

[14] M. Murashkin, "Oscillations Within Oscillations," Applied Mathematics and Computations, Elsevier Science Publishing Co., Inc., January 1993, pps. 45-82.

[15] A. H. Nayfeh, "Nonlinear Motion of an Asymmetic Body with Variable Roll Rates," Sandia National Labs Technical Report, 1972.

[16] A. H. Nayfeh, "Nonlinear Resonances in the Motion of Rolling Re-entry Bodies," Virginia Polytechnic Institute and State University Technical Report, Blacksburg, Va., 1971.

[17] A. Papoulis, "Probability, Random Variables, and Stocahstic Processes," McGraw-Hill, Inc., New York, 1965, pps. 423-426; 453-473.

[18] L. R. Petzold, "An Efficient Numerical Method for Highly Oscillatory Ordinary Differential Equations," SIAM Journal of Numerical Analysis, Society for Industrial and Applied Mathematics, June 1981, pps. 455-479.

[19] F. J. Regan, "Re-Entry Vehicle Dynamics," AIAA, New York, 1984, pps. 141-234; 397-408.

[20] L. F. Shampine, "Evaluation of a Test Set for Stiff ODE Solvers," ACM Transactions on Mathematical Software, Vol. 7, No. 4, December 1981, pps. 409-420.

[21] G. M. Shroff, H. B. Keller, "Stabilization of Unstable Procedures: the Recursive Projection Method," SIAM Journal of Numerical Analysis, Society for Industrial and Applied Mathematics, August 1993, pps. 1099-1120.

[22] S. E. Solovyov, V. A. Volpert, S. P. Davtyan, "Radially Symmetric Flow of Reacting Liquid with Changing Viscosity," SIAM Journal of Applied Mathematics, Society for Industrial and Applied Mathematics, August 1993, pps. 907-914.

[23] T. K. Summerset, R. M. Chowkwanyun, "Trajectory Optimization on Multiprocessors: A Comparison of Three Implementation Strategies," AIAA Atmospheric Flight Mechanics Conference, American Institute of Aeronautics and Astronautics, Inc., 1990, pps. 12-23.

[24] E. Weinan, "Convergence of Fourier Methods for the Navier-Stokes Equations," SIAM Journal of Numerical Analysis, Society for Industrial and Applied Mathematics, June 1993, pps. 650-674.

VITA

George P. Davailus was born August 4, 1968 in Scranton, Pennsylvania. In May 1990 he obtained a B.S. in Mathematics from Pennsylvania State University. Immediately afterwards, he began working at the Naval Surface Warfare Center at Dahlgren, Virginia as a government mathematician. In January 1992, he was admitted to the Virginia Polytechnic Institute and State University for graduate studies in computer science.

George P. Davailus