1994

NPSNET: Flight Simulation Dynamic Modeling Using Quaternions

Cooke, Joseph M.

Cooke, Joseph M., Zyda, Michael J., Pratt, David R. and McGhee, Robert B. “NPSNET: Flight Simulation Dynamic Modeling Using Quaternions,” Presence , Vol 1., No. 4, pp. 404-420.
http://hdl.handle.net/10945/41038

Joseph M. Cooke
Michael J. Zyda
David R. Pratt
Robert B. McGhee
Naval Postgraduate School
Department of Computer Science,
Code CS/Zk
Monterey, California 93943-5100

NPSNET: Flight Simulation Dynamic Modeling Using Quaternions

Abstract

The Naval Postgraduate School (NPS) has actively explored the design and implementation of networked, real time, three-dimensional battlefield simulations on low-cost, commercially available graphics workstations. The most recent system, NPSNET, has improved in functionality to such an extent that it is considered a low-cost version of the Defense Advanced Research Project Agency's (DARPA) SIMNET system. To reach that level, it was necessary to economize in certain areas of the code so that real time performance occurred at an acceptable level. One of those areas was in aircraft dynamics. However, with "off-the-shelf" computers becoming faster and cheaper, real-time and realistic dynamics are no longer an expensive option. Realistic behavior can now be enhanced through the incorporation of an aerodynamic model. To accomplish this task, a prototype flight simulator was built that is capable of simulating numerous types of aircraft simultaneously within a virtual world. Besides being easily incorporated into NPSNET, such a simulator also provides the base functionality for the creation of a general purpose aerodynamic simulator that is particularly useful to aerodynamics students for graphically analyzing differing aircraft's stability and control characteristics. This system is designed for use on a Silicon Graphics workstation and uses the GL libraries. A key feature of the simulator is the use of quaternions for aircraft orientation representation to avoid singularities and high data rates associated with the more common Euler angle representation of orientation.

1 Introduction

The current state of the art in simulation technology has provided today's military with many valuable training experiences that could not have been obtained elsewhere and, as a result, has greatly increased survivability and readiness. From flight simulators, which allow a pilot to explore the edge of the flight envelope without endangering crew or multimillion dollar assets, to battlefield simulators, which allow entire fighting divisions to practice command and control without having to incur the enormous costs of running a full blown field exercise, computer simulation has become a way of doing business within the military.

One simulation system designed by the Defense Advanced Research Projects Agency (DARPA) is the Simulation Networking system (SIMNET) (Thorpe, 1987). SIMNET is a networked battlefield simulator that allows multiple user interaction on the battlefield at many different levels. Vehicle simulators, such as tanks and aircraft, connect to the network and become part of a three-dimensional world. At the Naval Postgraduate School (NPS), an effort to develop a SIMNET-type system based on commercially available, general purpose,

Presence, *Volume 1, Number 4, Fall 1992*

graphic workstations has been active for a number of years. This system, NPSNET, consists of Silicon Graphics workstations attached to a local area Ethernet (Zyda, Pratt, Monahan, & Wilson, 1992). Eventually, NPSNET will become a node on the SIMNET network.

The speed of the computer platforms on which NPSNET now runs has increased significantly since its inception. It soon became evident that more realistic vehicle dynamics was desirable and would substantially improve system behavior. Therefore, this work is the result of the research done and the methodology used for providing this additional functionality to NPSNET's aircraft simulations.

There are several issues to be addressed when incorporating an aerodynamic model into a computer simulation. The complexity of the aerodynamic model, which orientation model to use, and how aircraft data should be represented in the system are the critical issues and are the center of focus in this work. Of primary importance is that the complexity of the model fits the objective of the simulation. A complete aerodynamic model that includes fully articulated control surfaces and airflow divergence patterns over the aircraft would seriously affect real time performance on any computer. Such models are usually computed in non-real time on supercomputers and are not appropriate for use on low-cost graphics workstations. On the other hand, modeling the dynamics of an aircraft kinematically, so that the aircraft's velocity and orientation are a linear result of control input, does not reproduce the nuances of aircraft motion and response that a user of a flight simulation would expect. The aerodynamic model's complexity must provide as much realism as possible without reducing the frame rate below an acceptable level.

The choice of orientation model considered is between the Euler angle or quaternion approach. Which one to use has been the subject of heated debate among computer scientists (Goldiez & Lin, 1991). Either model can be used to represent orientation, but, depending on the simulation's objectives, one method has certain advantages over the other. It basically comes down to determining which approach provides the right tool for the job (Shoemake, 1985).

Because each type of aircraft exhibits its own specific aerodynamics and handling characteristics, it is desirable to change these characteristics depending on the type of aircraft simulated. One solution is to base the aerodynamic model on the aircraft's stability coefficients, inertial coefficients, and airframe specifications, all of which are available in most aerodynamic stability and control textbooks. Stability coefficients provide a very accurate model of aircraft flight behavior. However, care must be taken when modeling some of the newer generation fighters. To improve maneuverability, these aircraft have been designed aerodynamically neutral to unstable. Their stability coefficients reflect this instability.

2 Coordinate Systems and Terminology

Coordinate systems and the method in which they are described vary to a great extent depending on the application and the preference of the user. In aircraft simulations, coordinate systems fall into two broad classes, "body" coordinates and "earth" or "inertial" coordinates (Rolfe & Staples, 1986). Body coordinates have their origin based at the center of gravity and continually move with the aircraft. Inertial coordinates, on the other hand, are defined with respect to the earth and have their origin positioned at some suitable location such as the center of the simulated world. Other coordinate systems exist, based on parameters such as the flight path and angle of attack. However, the aerodynamic model presented in this paper is based on geometric body and world coordinate systems. In general, all aerodynamic forces, accelerations, and velocities are calculated in the body coordinate system first, and then converted to the world coordinate system prior to updating an aircraft's position and altitude.

Figure 1 shows the generally accepted convention for labeling of the axes in the two coordinate systems (Anderson, 1989). Body coordinates are defined with the origin at the center of gravity (CG), the x axis along the fuselage pointing out the nose of the aircraft, the y axis along the wing-line pointing out the right wing, and the z axis pointing out the bottom of the plane. World coordinates are defined with the origin based at a fixed

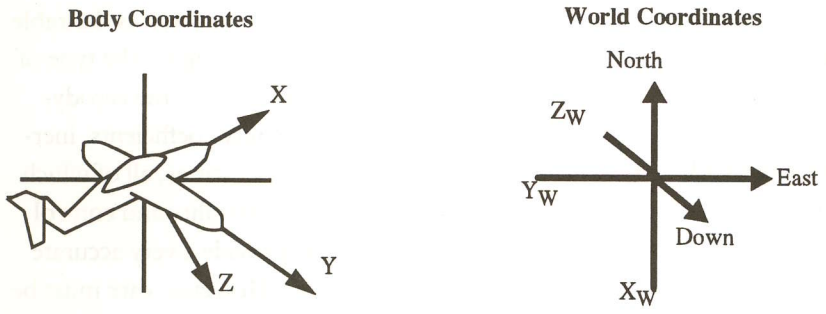

Figure 1. *World and body coordinate systems.*

X_w, Y_w, Z_w	Aircraft location in world coordinates (feet)
U_w, V_w, W_w	Aircraft velocity in world coordinates (ft/sec)
ψ, θ, ϕ	Azimuth, Elevation, Roll in world coordinates (radians)

Figure 2. *Terms defined within the world coordinate system.*

U, V, W	Linear velocity along X, Y, and Z body axes (ft/sec)
P, Q, R	Angular velocity around X,Y, and Z body axes (rad/sec)
V_τ	Resultant velocity Vector $\sqrt{U^2 + V^2 + W^2}$
V_ε	Wind velocity across tail of aircraft
$\dot{U}, \dot{V}, \dot{W}$	Linear acceleration (ft/sec^2)
$\dot{P}, \dot{Q}, \dot{R}$	Angular acceleration (rad/sec^2)
F_x, F_y, F_z	Forces acting on aircraft
L, M, N	Moments about the X, Y, and Z axes
α	Angle of attack[$\tan^{-1}$ (W/U)]
β	Sideslip [$\tan^{-1}$ (V/U)]

Figure 3. *Terms defined within the aircraft body coordinate system.*

point on the ground, the *x* axis pointing north, the *y* axis pointing east, and the *z* axis pointing down. Because of its limited effect, the curvature of the earth is usually ignored.

In a dynamics model, velocities, accelerations, and forces are described in both world and body coordinate systems. Without an explicit description of the variables used to describe the model, confusion can arise. Most terms described in this paper refer to the geometric body axes. However if a reference is made to the world coordinate system the subscript "w" is used (Fig. 2).

Terms without the "w" subscript relate to body axes and include linear and angular velocities, accelerations, forces, moments, angle of attack, and sideslip angle (Fig. 3). Note that the direction of angular accelerations and velocities and moment terms are defined using the right-hand rule around their respective axes (Fig. 4).

The aircraft control surfaces such as elevator, ailerons, and rudder are defined as a rotation in radians around their respective hinge points on the aircraft. When a control surface is flush with the aircraft, the angle of deflection is zero (Fig. 5).

Within the aerodynamic model, the particular aircraft being modeled is characterized by certain dimensional characteristics. A description of these terms is included in Figure 6.

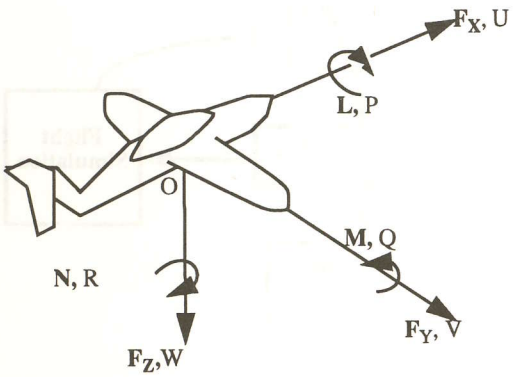

Figure 4. *Notation with respect to body axes.*

δe elevator deflection positive down (radians).
A positive δe produces a positive lift and a negative pitch moment.

δa aileron deflection positive left (radians).
A positive δa produces a negative roll moment.

δr positive nose left (radians).A positive δr produces a positive sideforce and a negative yaw moment.

Figure 5. *Terminology defining aircraft controls.*

3 Aerodynamic Model

The mathematical model presented takes forces, control inputs, and aircraft specifications as inputs, generating linear and angular velocities in aircraft body coordinates as outputs (Fig. 7). Based on a classical representation of linear aerodynamics and utilizing the total force Eqs. (3.18)–(3.20), all forces associated with lift and drag are calculated utilizing aerodynamic stability derivatives (Roskam, 1979).

Stability derivatives, first used over a half-century ago, assume that all aerodynamic forces and moments can be expressed as a function of the instantaneous value of the perturbation variables (Nelson, 1989). The perturbation variables are the instantaneous changes from the reference conditions of translational velocities, angular velocities, control deflections, and their derivatives. For example, the term $\delta X/\delta u$ is the stability derivative defining the change in X force with respect to the change in forward speed. This derivative can be expressed in terms of a nondimensional coefficient C_{Xu} as follows:

$$\frac{\delta X}{\delta u} = C_{Xu}\frac{1}{u_0}QS \tag{3.1}$$

where

$$C_{Xu} = \frac{\delta C_X}{\delta(u/u_0)} \tag{3.2}$$

and Q is the dynamic pressure, $\frac{1}{2}\rho V_\tau^2$, where ρ is the air density at the aircraft altitude.

Figure 8 lists the nondimensional coefficients used in this model. These coefficients are generally broken down into three categories, lateral, longitudinal, and control. The longitudinal coefficients represent forces effecting the longitudinal axes of the aircraft, while the lateral coefficients represent forces affecting the lateral axes of the aircraft. Nondimensional coefficients, generated in actual aircraft testing, are available for most aircraft. By using these coefficients in combination with the dynamics equations, it is possible to build a general use flight simulator.

Using the nondimensional coefficients, lift, drag, and sideforce are calculated as follows:

$$L' = \left[C_{L0} + C_{L\alpha}\alpha + C_{Q}Q\frac{c}{2V_\tau} + C_{L\dot{\alpha}}\dot{\alpha}\frac{c}{2V_\tau} + C_{L\delta e}\delta e\left[\frac{(V_\tau + \Delta V\epsilon)}{V_\tau}\right]^2\right]\frac{\rho V_\tau^2 S}{2} \tag{3.3}$$

$$D = \left[C_{D0} + C_{D\alpha}\alpha + C_{D\delta e}\delta e\left[\frac{(V_\tau + \Delta V\epsilon)}{V_\tau}\right]^2\right]\frac{\rho V_\tau^2 S}{2} \tag{3.4}$$

$$S_F = [C_{Y\beta}\beta + C_{Y\delta r}\delta r]\frac{\rho V_\tau^2 S}{2} \tag{3.5}$$

Once lift, drag, and sideforce are calculated, these forces are translated into forces along the aircraft X, Y, and Z axes as shown in Eqs. (3.6) through (3.8). The terms F_{AX}, F_{AY}, and F_{AZ} represent the resultant aerodynamic forces.

$$F_{AZ} = -L'\cos\alpha - D\sin\alpha \tag{3.6}$$

$$F_{AX} = L'\sin\alpha - D\cos\alpha - S_F\sin\beta \tag{3.7}$$

$$F_{AY} = S_F\cos\beta \tag{3.8}$$

S	surface area of wing (ft^2)
b	wing span (ft)
c	chord length (ft)
w	weight (lbs)
I_{xx}	roll inertia (slug-ft^2)
I_{yy}	pitch inertia (slug-ft^2)
I_{zz}	yaw inertia (slug-ft^2)

Figure 6. *Aircraft dimensional specifications.*

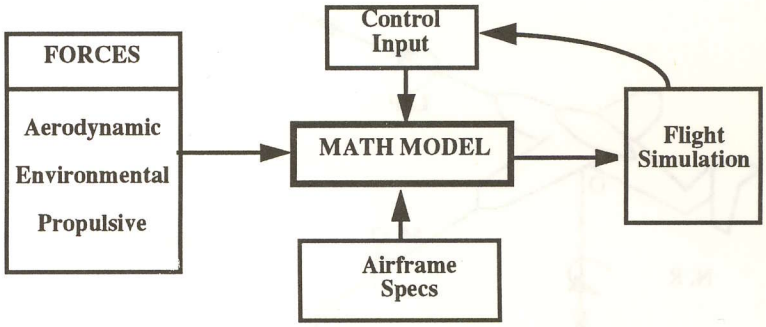

Figure 7. *Basic aerodynamic model.*

The aerodynamic moments represent the torque forces about the center of the aircraft and are determined in the following equations:

$$L_A = \left[C_{L\beta}\beta + C_{LP}P\frac{b}{2V_T} + C_{LR}R\frac{b}{2V_T} + C_{L\delta a}\delta a + C_{L\delta r}\delta r\right]\frac{\rho V_T^2 Sb}{2} \quad (3.9)$$

$$M_A = \left[C_{M0} + C_{M\alpha}\alpha + C_{MQ}Q\frac{c}{2V_T} + C_{M\dot{\alpha}}\dot{\alpha}\frac{c}{2V_T} + C_{M\delta e}\delta e\left[\frac{(V_T + \Delta V\epsilon)}{V_T}\right]^2\right]\frac{\rho V_T^2 Sc}{2} \quad (3.10)$$

$$N_A = \left[C_{N\beta}\beta + C_{NP}P\frac{b}{2V_T} + C_{NR}R\frac{b}{2V_T} + C_{N\delta a}\delta a + C_{N\delta r}\delta r\right]\frac{\rho V_T^2 Sb}{2} \quad (3.11)$$

The forces and moments that result from the above calculations are added to other forces and moments at this time:

$$F_X = F_{AX} + F_{\text{Thrust}} \quad (3.12)$$

$$F_Y = F_{AY} \quad (3.13)$$

$$F_Z = F_{AZ} \quad (3.14)$$

$$L = L_A + L_{\text{Torque}} \quad (3.15)$$

$$M = M_A + M_{\text{Thrust}} + M_{\text{Gyro}} \quad (3.16)$$

$$N = N_A + N_{\text{Thrust}} + N_{\text{Gyro}} \quad (3.17)$$

Engine forces such as thrust, torque, and gyroscopic effect as well as environmental forces such as wind shear can have anywhere from a minor to a significant effect on the forces and moments along all axes of the aircraft (Roskam, 1979). However, to limit this complexity of the model, some simplifications are made. Engine thrust is limited to the *X*-axis only and no calculations are made for torque or gyroscopic effect since one of the author's experiences as a pilot, and reference to the relevant literature, indicates that these are second-order effects for high-performance aircraft.

The total force equations are used to determine the linear acceleration of the aircraft (Nelson, 1989):

$$\dot{U} = VR - WQ - g\sin\theta + \frac{F_X}{m} \quad (3.18)$$

$$\dot{V} = WP - UR + g\sin\phi\cos\theta + \frac{F_Y}{m} \quad (3.19)$$

$$\dot{W} = UQ - VP + g\cos\phi\cos\theta + \frac{F_Z}{m} \quad (3.20)$$

The total moment equations are used to derive the equations for solving angular acceleration:

$$L = I_{XX}\dot{P} - I_{XZ}\dot{R} - I_{XZ}PQ + (I_{ZZ} - I_{YY})RQ \quad (3.21)$$

$$M = I_{YY}\dot{Q} + (I_{XX} - I_{ZZ})PR + (I_{XZ}(P^2 - R^2) \quad (3.22)$$

$$N = I_{ZZ}\dot{R} - I_{XZ}\dot{P} + (I_{YY} - I_{XX})PQ + I_{XZ}QR \quad (3.23)$$

However, prior to solving for either *P* or *R*, an interim step is required:

$$L'' = L + I_{XZ}PQ - (I_{ZZ} - I_{YY})RQ \quad (3.24)$$

Longitudinal Coefficients

C_{Lo}	Reference Lift at zero angle of attack
C_{Do}	Reference Drag at zero angle of attack
$C_{L\alpha}$	Lift curve slope
$C_{D\alpha}$	Drag curve slope
C_{Mo}	Pitch moment
$C_{M\alpha}$	Pitch moment due to angle of attack
C_{LQ}	Lift due to pitch rate
C_{MQ}	Pitch moment due to pitch rate
$C_{L\dot{\alpha}}$	Lift due to angle of attack rate
$C_{M\dot{\alpha}}$	Pitch moment due to angle of attack rate

Lateral Coefficients

$C_{Y\beta}$	Side force due to sideslip
$C_{L\beta}$	Dihedral effect
C_{LP}	Roll damping
C_{LR}	Roll due to yaw rate
$C_{N\beta}$	Weather cocking stability
C_{NP}	Rudder adverse yaw
C_{NR}	Yaw damping

Control Coefficients

$C_{L\delta e}$	Lift due to elevator
$C_{D\delta e}$	Drag due to elevator
$C_{M\delta e}$	Pitch due to elevator
$C_{L\delta a}$	Roll due to aileron

Figure 8. *Aircraft specification notation.*

$$N' = N - (I_{YY} - I_{XX})PQ - I_{XZ}RQ \tag{3.25}$$

Therefore,

$$\dot{P} = (L''I_{ZZ} - N'I_{XZ})/(I_{XX}I_{ZZ} - I_{XZ}^2) \tag{3.26}$$

$$\dot{Q} = (M - (I_{XX} - I_{ZZ})PR - I_{XZ}(P^2 - R^2))/I_{YY} \tag{3.27}$$

$$\dot{R} = (N'I_{XX} + L''I_{XZ})/(I_{XX}I_{ZZ} - I_{XZ}^2) \tag{3.28}$$

are the equations for angular acceleration.

Linear and angular velocities are determined by numerically integrating the accelerations. The trapezoidal rule, sometimes referred to as the modified Euler method or the first-order predictor–corrector method, is used. The general method for this integration technique is as follows (Press, Flannery, Teukolsky, & Vetterling, 1990):

$$P_n = P_{n-1} + \left(\frac{\dot{P}_{n-1} + \dot{P}_n}{2}\right) dt \tag{3.29}$$

where

P_n = new value of P

P_{n-1} = previous value of P

$\dot{P}_n$ = predicted rate-of-change of P

$\dot{P}_{n-1}$ = previous rate-of-change of P

dt = integration step size

4 Position Updates

Because position updates usually occur in a world coordinate system, the aircraft's linear velocities must be converted into world position rates by applying the following transformation, which effectively rotates the $(U\ V\ W)$ vector by the Euler angles (Roskam, 1979). The order of transformation is important when utilizing this method and proceeds as follows:

$$\begin{bmatrix} U_\phi \\ V_\phi \\ W_\phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix} U_{\phi\theta} \\ V_{\phi\theta} \\ W_{\phi\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} U_\phi \\ V_\phi \\ W_\phi \end{bmatrix} \tag{4.2}$$

$$\begin{bmatrix} U_W \\ V_W \\ W_W \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_{\phi\theta} \\ V_{\phi\theta} \\ W_{\phi\theta} \end{bmatrix} \tag{4.3}$$

Integrating the resultant velocity vector, now in world coordinates, by the time step of the program, a position update is obtained [Eq. (4.4)]:

$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} = \begin{bmatrix} X_{W_{\text{old}}} \\ Y_{W_{\text{old}}} \\ Z_{W_{\text{old}}} \end{bmatrix} + \begin{bmatrix} U_W \\ V_W \\ W_W \end{bmatrix} dt \tag{4.4}$$

This technique is revisited in the next section.

5. Methods of Orientation

The aerodynamic model generates rotational velocities relative to the fixed aircraft body coordinate system. But just as position updates could only be determined after converting linear velocities into world coordinates, orientation updates require conversion of angular velocities in a similar manner. Three methods exist for defining the conversion of angular velocities to orientations in world coordinates, each having its own particular advantages and disadvantages (Goldiez & Lin, 1991).

The most popular of these three methods is known as the Euler method. Using a sequence of three angles, the Euler method provides an intuitive description of aircraft attitude in world space (Rolfe, 1986). These angles consist of the familiar azimuth angle ψ, the elevation angle θ, and the roll angle ϕ. The next method, which has become popular in recent years, is the quaternion method. Based on the unit sphere, the quaternion method provides an elegant method of defining rotations through the use of four parameters. Three of the coordinates describe the axis of rotation while the fourth is determined by the angle through which the rotation occurs (Shoemake, 1985). The third method of defining orientation is the direction cosine matrix. The direction cosines relate the aircraft body axis frame to the world reference frame. Direction cosines, as used in flight simulation, are generally determined from either Euler angles or quaternions and are utilized for transformations between axes. However, an alternative approach (not discussed in this paper) is to use incremental rotation matrices to update rotation matrices (Paul, 1981). A disadvantage of this approach is that repeated incremental rotation matrix multiplication can result in drift requiring periodic renormalization of the direction cosine matrix (Funda, Taylor, & Paul, 1990).

Each method has its own particular advantages and disadvantages and their use depends on the application and the implementation. Because NPSNET is a networked simulator, the orientation model used must not only render orientations in the world of the aircraft being piloted, but also of other aircraft in the world, either flying autonomously or piloted remotely across the network.

6 Euler Method

The most common method of defining an aircraft's orientation in world space is by the Euler attitude angles. Starting with the aircraft's axis origin aligned with the world's axis origin, the Euler angles specify three successive rotations to bring the world coordinates into alignment with the aircraft. The fact that there exist 12 possible ways to define rotations, each with potentially

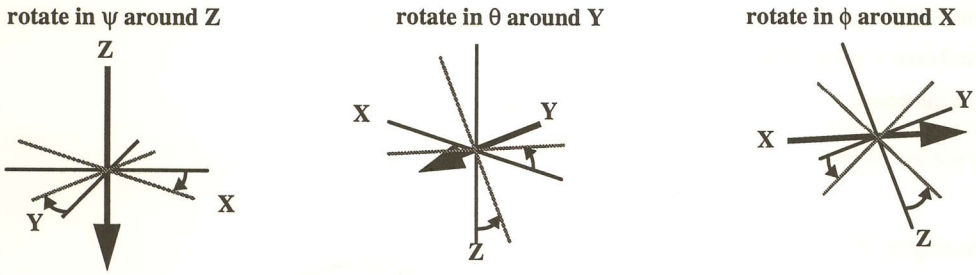

Figure 9. *Euler attitude angle rotation.*

different results, means that the order of these rotations is important. Euler chose the convention of rotating first about the z axis, then about the new x axis, and finally about the new z axis. This convention exists in celestial mechanics, applied mechanics, and molecular and solid state physics. The convention used in quantum mechanics, nuclear physics, and particle physics, chooses to rotate first about the z, then the new y, and finally the new z (Burchfiel, 1990). The convention most often used in graphics is standard to aerospace engineers and has been proposed for use by SIMNET (UCF/IST 1990). Using the right-hand rule, rotations are made, first, about the z axis by the angle ψ, then about the new y axis by angle θ, and finally about the new x axis by angle ϕ (Fig. 9).

The range of values the attitude angles can take are

$$\psi = \pm\pi \qquad \theta = \pm\frac{\pi}{2} \qquad \phi = \pm\pi$$

The aerodynamic model generates velocities in body coordinates. As seen in Eqs. (4.1)–(4.3), the linear body rates are transformed into world rates by application of the Euler angles. What follows is a method for obtaining these angles.

There is a direct relationship between Euler attitude angles and the angular velocity of the aircraft around its body axes (Nelson, 1989). From this relationship, the rates of change of the attitude angles can be derived:

$$\dot{\phi} = P + Q \sin\phi \tan\theta + R \cos\phi \tan\theta \qquad (6.1)$$

$$\dot{\theta} = Q \cos\phi - R \sin\phi \qquad (6.2)$$

$$\dot{\psi} = Q \sin\phi \sec\theta + R \cos\phi \sec\theta \qquad (6.3)$$

The inverse of the above equations are

$$P = \dot{\phi} - \dot{\psi} \sin\theta \qquad (6.4)$$

$$Q = \dot{\theta} \cos\phi + \dot{\psi} \sin\phi \cos\theta \qquad (6.5)$$

$$R = -\dot{\theta} \sin\phi + \dot{\psi} \cos\phi \cos\theta \qquad (6.6)$$

Equations (6.1) through (6.3) are also known as the gimbal equations and are quite commonly used in simulation. However, a problem exists when pitch, θ, goes through the vertical. That is, where pitch becomes $\pm(\pi/2)$. At that point $\dot{\phi}$ and $\dot{\psi}$ become undefined. Implementing a flight dynamics model capable of complete vertical maneuvering necessitates "fixing" the code so a division by zero does not occur.

7 Direction Cosines

In the case of a flight simulation, transforming between body coordinates and world coordinates is done quite frequently. A convenient way to represent the transformation between two coordinate systems is with the direction cosines. Using matrix notation and the direction cosines (a, b, c), the transformation from body to world axes is expressed by

$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad (7.1)$$

X, Y, and Z represent vectors of any kind, such as force, velocity, and acceleration. The inverse relationship, converting world coordinates to body coordinates, is the transpose:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} \qquad (7.2)$$

$$a_1 = \cos\theta\cos\psi \qquad b_1 = \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi \qquad c_1 = \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi$$

$$a_2 = \cos\theta\sin\psi \qquad b_2 = \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi \qquad c_2 = \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi$$

$$a_3 = -\sin\theta \qquad b_3 = \sin\phi\cos\theta \qquad c_3 = \cos\phi\cos\theta$$

Figure 10. *Direction cosines in terms of Euler angles.*

In terms of the Euler attitude angles, the direction cosines for the above transformations are shown in Figure 10.

It should be noted that there are 12 ways in which Euler angles can be defined, and, as a result, just as many ways to compute the direction cosines, although the values finally obtained are independent of the choice of Euler angles.

The direction cosines are needed for transformations between coordinate systems, whether Euler angles or quaternions are used to specify orientation. [Although, as explained in Funda et al. (1990), an alternative to matrix multiplication using direction cosines is to transform points by quaternion multiplication.] Direction cosines were already used for transforming linear velocities in body coordinates to world coordinates. By multiplying the transformation matrices of [Eqs. (4.1)–(4.3)] into one matrix, the result would be identical to the transformation matrix of Eq. (7.1). By using direction cosines, the need for determining the intermediate velocities is eliminated.

8 Quaternion Method

An alternate method that has gained popularity in the graphics community in the mid-1980s is through the use of the unit quaternion. Not a new method, quaternions have been around for over a century. Augmenting the "four-parameter method," they have been useful to aerodynamic engineers for some time and are still the method of choice for describing spacecraft orientation (Mitchell & Rodgers, 1965). Discovered by Sir William Rowan Hamilton in 1843 as a result of a search for a generalization of complex numbers, quaternions provide an efficient means for updating orientations (Shoemake, 1985).

There are numerous ways to interpret the quaternion mathematically. They can be described as an algebraic

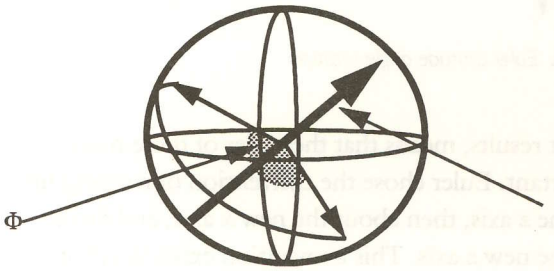

Figure 11. *Quaternion orientation.*

quantity,

$$w + ix + jy + kz \tag{8.1}$$

as a point in three-dimensional projective space (w, x, y, z), as a linear transformation of four space (matrix), or as a scalar plus 3-vector:

$$(w, \mathbf{v}) \qquad \mathbf{v} = i\mathbf{x} + j\mathbf{y} + k\mathbf{z} \tag{8.2}$$

The best notation depends on their intended use. The most intuitive approach is to view the quaternion as a scalar plus 3-vector [Eq. (8.2)]. However, for algebraic manipulation, Eq. (8.1) generally becomes more useful.

A common way of defining quaternion orientation is in combination with Euler's theorem which states that the orientation of a rigid body can be described as a rotation about an axis $\mathbf{v}$ by rotation angle Φ (Fig. 11) (Goldstein, 1980). Constraining the axis vector $\mathbf{v}$ to be of unit magnitude, the quaternion becomes

$$Q = \cos\frac{\Phi}{2}, \mathbf{v}\sin\frac{\Phi}{2} \tag{8.3}$$

This representation is always of unit magnitude such that

$$w^2 + x^2 + y^2 + z^2 = 1 \tag{8.4}$$

Prior to defining how to rotate a rigid body using the unit quaternion, it is necessary to review some of the mathematics associated with the quaternion. For the

$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} = \begin{bmatrix} 1-2\sin^2 A \sin^2 \frac{1}{2}D & 2\cos A \cos B \sin^2 \frac{1}{2}D \\ & -2\cos C \sin \frac{1}{2}D \cos \frac{1}{2}D \\ 2\cos A \cos B \sin^2 \frac{1}{2}D & 1-2\sin^2 \frac{1}{2}D \sin^2 B \\ +2\cos C \cos \frac{1}{2}D \sin \frac{1}{2}D & \\ 2\cos A \cos C \sin^2 \frac{1}{2}D & 2\cos B \cos C \sin^2 \frac{1}{2}D \\ -2\cos B \sin \frac{1}{2}D \cos \frac{1}{2}D & +2\cos A \sin \frac{1}{2}D \cos \frac{1}{2}D \end{bmatrix} \begin{bmatrix} 2\cos A \cos C \sin^2 \frac{1}{2}D \\ +2\cos B \sin \frac{1}{2}D \cos \frac{1}{2}D \\ 2\cos B \cos C \sin^2 \frac{1}{2}D \\ -2\cos A \cos \frac{1}{2}D \sin \frac{1}{2}D \\ 1-2\sin^2 C \sin^2 \frac{1}{2}D \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Figure 12. *Four parameter method.*

purpose of a flight simulation, an understanding of the multiplication and the quaternion derivative is necessary. More comprehensive reviews are available in Shoemake (1985), Goldstein (1980), Funda et al. (1991), and Chou (1992).

Most applications involving quaternions make use of the mathematics associated with their multiplication. Similar to the algebra associated with imaginary numbers, quaternions have three imaginary units, i, j, and k and are noncommunitive under multiplication with

$$i^2 = j^2 = k^2 = -1 \tag{8.5}$$

and

$$ij = k = -ji \qquad jk = i = -kj \qquad ki = j = -ik \tag{8.6}$$

In algebraic notation, the product of quaternion Q multiplied by quaternion Q_1 is

$$\begin{aligned} QQ_1 &= (w + ix + jy + kz)(w_1 + ix_1 + jy_1 + kz_1) \\ &= (ww_1 - xx_1 - yy_1 - zz_1) \\ &\quad + i(xw_1 + wx_1 - zy_1 + yz_1) \\ &\quad + j(yw_1 + zx_1 - wy_1 + xz_1) \\ &\quad + k(zw_1 + yx_1 - xy_1 + wz_1) \end{aligned} \tag{8.7}$$

In vector notation:

$$QQ_1 = (w, \mathbf{v})(w_1, \mathbf{v}_1) = ww_1 - \mathbf{v} \cdot \mathbf{v}_1, \\ w\mathbf{v}_1 + w_1\mathbf{v} + \mathbf{v} \times \mathbf{v}_1 \tag{8.8}$$

The results of the above multiplication is a rotation from the orientation represented by Q to the new cumulative orientation of Q and Q_1 in quaternion terms.

Multiplication provides a method of orientation extrapolation that can be of benefit in a networked simulation. If Q_1 represents a finite rotation based on an integral time step, and Q represents the cumulative rotation, then a repeated multiplying of Q by Q_1 will result in a smooth rotation across a series of update frames (Burchfiel, 1990).

One frame of axes (body coordinates) can be brought into coincidence with a reference frame by a single rotation D about a fixed axis making angles A, B, and C with a second reference frame (world coordinates). The four parameters A, B, C, and D, therefore, define the orientation of the aircraft body in world coordinates (Rolfe & Staples, 1986). The transformation matrix relating body to world coordinates using these four parameters is shown in Figure 12. While this matrix involves four angles and appears to be more complex than the Euler angle matrix of Figure 10, it can be simplified by making the substitutions:

$$\begin{aligned} q_0 &= \cos 1/2D \\ q_1 &= \cos A \sin 1/2D \\ q_2 &= \cos B \sin 1/2D \\ q_3 &= \cos C \sin 1/2D \end{aligned} \tag{8.9}$$

The transformation matrix then becomes

$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{8.10}$$

which represents the transform based on the unit quaternion. This result is used for determining position updates as well as orientation updates.

In case the angles A, B, C, D are not initially known, values for q_1, q_2, q_3, and q_4 can be obtained by a straightforward process from the initial direction cosine matrix (Funda et al., 1991). To update the resulting quaternion from angular accelerations, the following equations are used:

$$\begin{aligned} \dot{q}_0 &= -1/2(q_1P + q_2Q + q_3R) \\ \dot{q}_1 &= 1/2(q_0P + q_2R - q_3Q) \\ \dot{q}_2 &= 1/2(q_0Q + q_3P - q_1R) \\ \dot{q}_3 &= 1/2(q_0R + q_1Q - q_2P) \end{aligned} \tag{8.11}$$

Because of the constraint that the quaternion be of unit value and assuming an integration step size of less than 1, the above set of equations become

$$\begin{aligned} \dot{q}_0 &= -1/2(q_1P + q_2Q + q_3R) + \lambda q_0 \\ \dot{q}_1 &= 1/2(q_0P + q_2R - q_3Q) + \lambda q_1 \\ \dot{q}_2 &= 1/2(q_0Q + q_3P - q_1R) + \lambda q_2 \\ \dot{q}_3 &= 1/2(q_0R + q_1Q - q_2P) + \lambda q_3 \end{aligned} \tag{8.12}$$

where λ is an integration drift correction gain given by

$$\lambda = 1 - (q_0^2 + q_1^2 + q_2^2 + q_3^2) \tag{8.13}$$

Alternatively, Eq. (8.11) can be integrated without drift correction providing that periodic normalization to unit magnitude is accomplished (Funda et al., 1990).

Many of the auxiliary computations involved with a flight simulation require the use of Euler angles. It is to be emphasized, however, that knowledge of Euler angles is not required to obtain the direction cosines of Eq. (8.10). In fact, Eq. (8.11) can be used to update the direction cosine matrix without the calculation of any trigonometric functions at all.

When needed, Euler angles can be obtained from the transformation matrix in Eq. (8.10) via the following method. Because pitch is limited to $\pm\pi/2$, cos (θ) is always positive. As a result, obtaining these angles is relatively simple. The evaluation angle is derived from the transformation matrix [Eq. (7.1)] as follows:

$$\theta = \text{asin}\ (-a_3) \tag{8.14}$$

To obtain the azimuth (ψ) angle it must be noted that, since cos (θ) is always positive, the sign value of a_2 always reflects the sign value of sin (ψ):

$$a_2 = \cos\theta \sin\psi \tag{8.15}$$

Therefore:

$$\psi = \text{acos}\left(\frac{a_1}{\cos\theta}\right) \cdot (\text{sign}\ [a_2]) \tag{8.16}$$

The roll (ϕ) angle is similarly obtained:

$$\phi = \text{acos}\left(\frac{c_3}{\cos\theta}\right) \cdot (\text{sign}\ [b_3]) \tag{8.17}$$

9 Advantages and Disadvantages

Quaternions and Euler angles have their own advantages and disadvantages. Euler angles use only three components instead of four to represent orientation. If one were to send quaternions over a network in place of Euler angles, as has been proposed for SIMNET (Burchfiel, 1990), network traffic would increase. However, in cases where angular rates remain constant for long periods of time, by extrapolating orientation updates with quaternions, it would be necessary to send an update only when an angle rate change occurs.

As pointed out above, quaternions can be computed directly from the dynamic equations, bypassing the computation of transcendental functions necessary in computing Euler angles. If each transcendental function costs approximately 20 arithmetic operations then the net cost for deriving a rotation update using the Euler method is 94 operations (Burchfiel, 1990). This can be compared to 42 operations using the quaternion method. In flight simulation, Euler angles are sometimes necessary for use in other simulator functions such as cockpit displays, etc. This means that approximately another 64 operations are necessary, making the Euler method slightly more computationally efficient. However, in many cases it is not necessary to compute the angles at each orientation update. Network updates should only occur when a rate change occurs. Additionally, radars, gyros, and attitude indicators can be updated at slower rates. The bottom line is that orientations can be achieved very efficiently utilizing quaternions, without calculating Euler angles. When Euler angles are needed for other aircraft functions, then quaternions become less efficient, depending on how often they need to be computed (Table 1).

The most significant advantage of quaternions is that no singularity exists when the elevation angle (θ) passes through $\pm\pi/2$. In the Euler method, $\dot{P}$ and $\dot{R}$ both become undefined in this situation due to division by zero. Techniques exist, however, for working around this singularity. Truncating values as $\pm\pi/2$ is approached will avoid this problem. If the elevation angle is truncated at values of 89.99 and 90.01 then a 0.02 degree rotation skip results (Goldiez & Lin, 1991). Depending on the speed of the program and the rotation rates desired, this may not be noticeable. However, in higher fidelity simulations, where a slow vertical maneuver is executed, it is a factor. Regardless of the significance of these effects, however, this approach has the disadvantage of introducing nonunique values for Euler angles.

Numerous aircraft operate autonomously within the prototype simulation, changing very little in angular velocity. When this simulator is eventually networked to other workstations, quaternions will provide a way of forward interpolating rotations, thereby eliminating the need for the continued transmission of update packets. If updates are eventually needed, the quaternion rate can quickly and easily be converted into Euler angles.

Table 1. *Efficiency Comparison of Euler and Quaternion Methods*

Operation	Euler	Quaternion
Derivation	96	42
Creating rotation matrix	20	32
Total	116	74
Euler angle conversion	0	64
Total calculations	116	138

As the number of aircraft increase in the simulated world, the number of calculations necessary for orientation updates begins to multiply. Since only the currently piloted aircraft makes use of the Euler angles for additional simulation functions, calculating Euler angles is not necessary for all other aircraft in the simulation. Therefore, it becomes more efficient to utilize quaternions for defining these rotations, saving approximately 42 arithmetic operations per update per aircraft.

10 Overall System Layout

To satisfy the prototype's basic requirements, the overall structure of the system is designed as shown in Figure 13. An aircraft data file makes it simple to create new aircraft, position these aircraft, and designate their handling characteristics. It is also used to initialize the flight parameters of the autonomous aircraft. The program data structure contains information describing the current state of the aircraft and its design specifications. The aerodynamic model is used for updating the piloted aircraft's body rates. The nondynamic model also outputs a set of velocities, but unlike the dynamic model, it determines these velocities in accordance with a predetermined script. The orientation model converts body rates to position and orientation in world space. Euler angles are then determined for the piloted aircraft and used to update cockpit displays. Autonomous aircraft do

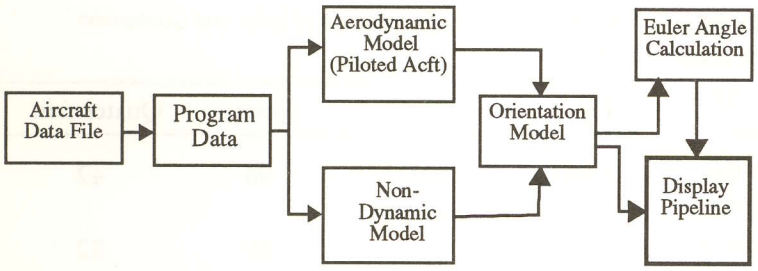

Figure 13. *Prototype flight simulator basic structure.*

```
Flight Record
    1                          /id number/
    1                          /type aircraft/
    200.0                      /airspeed/
    -950.0                     /posx/
    300.0                      /altitude/
    0.0                        /posz/
    0.0                        /heading/
    0.0                        /initial angle of bank/
    3.0                        /initial gforce/
    1                          /status:0-piloted,1-levelturn,2-g controlled turn/
Specification Record
    4                          /type aircraft-- A4/
    1                          /jet or prop jet:1 prop:0/
    27.5                       /b/
    260.0                      /S/
    10.8                       /c/
    546.0                      /m/
    8090.0                     /Ix/
    25900.0                    /Iy/
    29200.0                    /Iz/
    1300.0                     /Ixz/
    0000.0                     /max thrust/
    8000.0                     /mil thrust or horsepower/
    0.03 0.3                   /CDo /CDa
    0.28 3.45 0.0 0.72 0.36    /CLo /CLa /CLq /CLda /CLde
    0.0 -3.6 -0.38 -1.1 -0.5   /CMo /CMq /CMa /CMda /CMde
    -0.98 0.17                  /CYb /CYdr
    -0.12 -0.26 0.14 0.08 -0.105  /CLb /CLp /CLr /CLda /CLdr
    0.25 0.022 -0.35 0.06 0.032   /CNb /CNp /CNr /CNda /CNdr
    0.2618 -0.5236 0.5236      /deflection limits of rud, ail, elevator (radians)/
```

Figure 14. *Example aircraft data file.*

not require the calculation of Euler angles since they need not display instrument readings to a human pilot and therefore bypass this function.

11 Implementation Details

Data records within the aircraft data file are divided into two categories, flight records and aircraft specification records. The flight record contains information describing the position of an aircraft and its initial flight parameters (Fig. 14). The specification record contains the dimensional characteristics and stability coefficients describing a particular aircraft. By manipulating the data in the specification record, one can change an aircraft's basic design as desired. To link an aircraft to a particular set of specification data, all that is necessary is to match the "type aircraft" information within the two records. This system allows one specification record to be used for an unlimited number of flight records.

```
int       ID;              --number assignment
int       type;            --aircraft type
int       status;          --piloted:0 or autonomous level turn:1 climbing turn:2
typeptr   Tptr;            --pointer to aircraft specification data structure
float     Forces[3];       --forces in X,Y,Z dir
float     Torques[3];      --torques around X,Y,Z axis
float     linear_vel[3] ;  --velocity in X,Y,Z direction
float     angular_vel[3];  --angular velocity around X,Y,Z axes
float     linear_accel[3]; --linear accelerations
float     angular_accel[3]; --angular accelerations
float     sideslip;        --sideslip or beta angle
float     ang_atk;         --angle of attack
float     d_ang_atk;       --angle of attack rate
float     lift;            --total lift
float     drag;            --total drag
double    Q[4];            --quaternion
Matrix    H;               --direction cosine matrix
float     euler_angles[3]; --euler angles angles in radians - yaw,pitch,roll
float     pos[3];          --world position in X, Y, Z
float     ref[3];          --look direction
float     vel_world[3];    --velocities in world position - X,Y,Z
float     gfor;            --amount of g force
float     rpm;             --engine rpm
float     elev;            --flight control positions
float     eltrim;          --elevator trim
float     rud;             --rudder position
float     ail;             --aileron position
float     thro;            --throttle position
int       flaps;           --flap position
int       gear;            --landing gear position
```

Figure 15. *Aircraft flight data structure.*

A flight data structure provides a global source of information on the state of each aircraft in the simulation. The information contained here is necessary for operation of the aerodynamic and orientation models, and for updating cockpit displays (Fig. 15). Note that the fourth item in this structure is a pointer to another data structure containing aircraft specification data. Not shown, this structure contains information identical to that found in the specification record of the data input file (Fig. 14). Maintaining flight information in two separate files saves some storage space by allowing more than one aircraft to use a single set of specification data.

The throttle in an aircraft is the pilot's primary means to control the engine. As such, a mapping of throttle position to engine rpm must be devised that incorporates delays associated with engine spool-up characteristics. In large, high-speed simulators, rpm and engine data are retrieved from engine-specific lookup tables. Because tables such as these are engine specific, the following simpler, generic method was devised.

$$\Delta\text{rpm} = (\text{rpm}_{\text{desired}} - \text{rpm}_{\text{current}})\varphi dt \qquad (11.1)$$

where

$\text{rpm}_{\text{desired}}$ = throttle position × throttle gain

dt = delta time

φ = engine spool-up gain factor (inverse of time-constant)

Since applications using this simulator include both jet and propeller aircraft, it was decided that allowances should be made for the differing characteristics of the two types of engines. Jet engines are generally rated in terms of thrust (lb), while propellers are rated in terms of horsepower (ft-lb/s) (Anderson, 1989). Since the aerodynamic model uses thrust in terms of lb, it can use the data for jets directly. However, propeller driven aircraft require the following conversion:

```
aerodynamic model
        read aircraft velocity
        time step for aero model = timestep_factor * velocity
        for computed_time_step loop
                do aero calculations
                update aircraft state variables
        end loop
exit aerodynamic model
```

Figure 16. *Algorithm for computing time step.*

$$T = \frac{550\eta HP}{V\tau}\frac{\rho}{\rho_0} \qquad (11.2)$$

where

η = propeller efficiency (usually around .8)

HP = engine rated horsepower

$\frac{\rho}{\rho_0}$ = density altitude ratio where ρ_0 is the density at sea level

A normal aircraft control stick exhibits two degrees of freedom, left–right for aileron control and back–forward for elevator control. Therefore, control inputs from the spaceball were limited to these directions. The maximum deflection of the control stick is information entered via the specification records. It is a simple procedure to read deflection data from the spaceball and linearly map it to a control deflection somewhere between ±max obtainable deflection. Rudder deflection was not simulated since rudder control is not normally used in jet aircraft after takeoff.

12 Speed of Aerodynamic Model

The aerodynamic model exhibits a tendency to "blow up" if the time step between aerodynamic calculations becomes too great. This becomes very evident when the aircraft speed and rotation rates increase. The solution to this problem is to run the aerodynamic model at a faster rate than the rest of the system. The trick is to determine how fast. "Smart" integration schemes can be found in the literature that increase or decrease the number of time steps according to the amount of numerical divergence present in the integrated values (Press et al., 1990).

A more simple method is used in this program to solve for this problem. Because there exists a direct relationship between an aircraft's speed and its tendency to produce a numerical instability, the time step was adjusted in direct relationship with the aircraft's airspeed. Prior to updating body rates in the aerodynamic model, aircraft airspeed is measured and the number of time steps is calculated (Fig. 16).

Increasing the time step does not decrease performance of the overall system. The speed of the aerodynamic model in the current implementation ranges from 17.6 to 18.8 msec. Figures 17 and 18 illustrate the complexity of the graphics portion of the overall simulation model. Running at approximately 120 msec, the graphics pipeline remains the limiting factor in this system.

13 Conclusions and Future Work

The techniques presented in this paper have proven to be an effective method for implementing a graphical dynamic flight simulation on a matrix-based graphics computer in real-time. Like most research and academic projects, this aircraft simulator is structured to allow for the addition of more detailed functionality. Current work includes the integration of a weapons delivery system and avionics suite. The orientation model functions developed during the course of this paper have become part of the C program library at the Naval Postgraduate School, thereby providing an alternate and more flexible tool for manipulating solid objects in a graphical environment. Integration of this orientation model into other dynamic simulation systems is also under investigation.

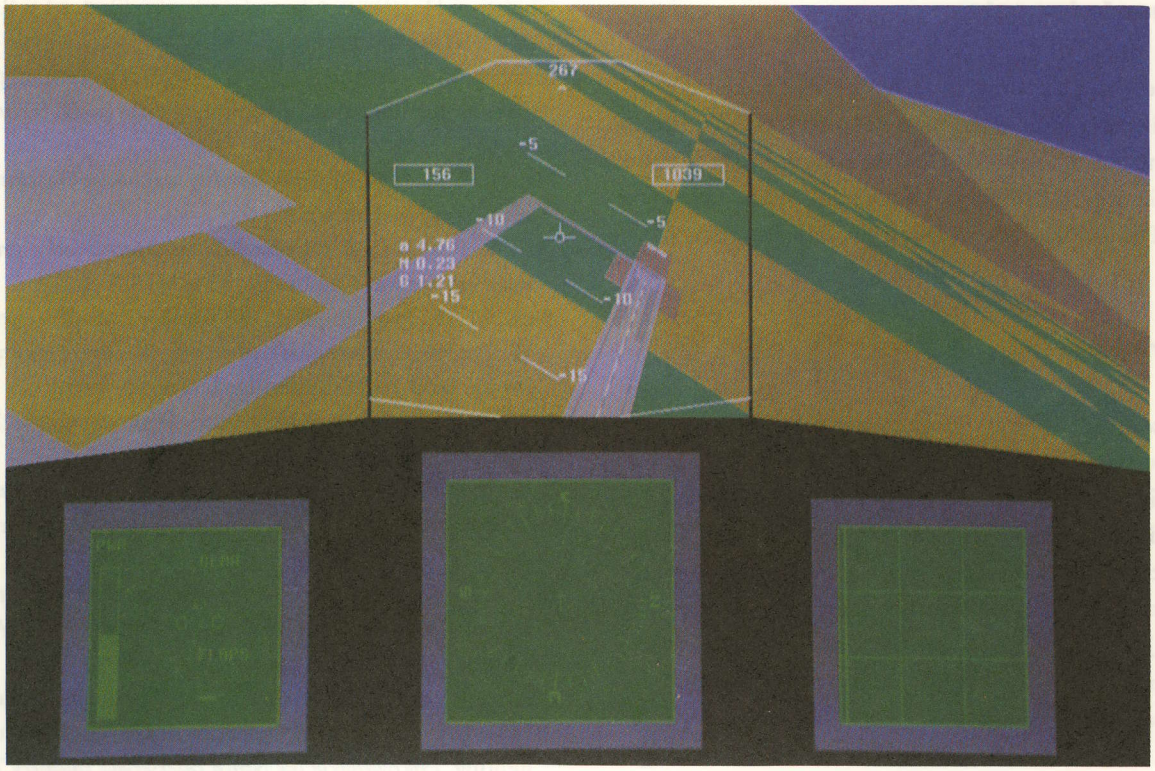

Figure 17. *Looking down towards runway.*

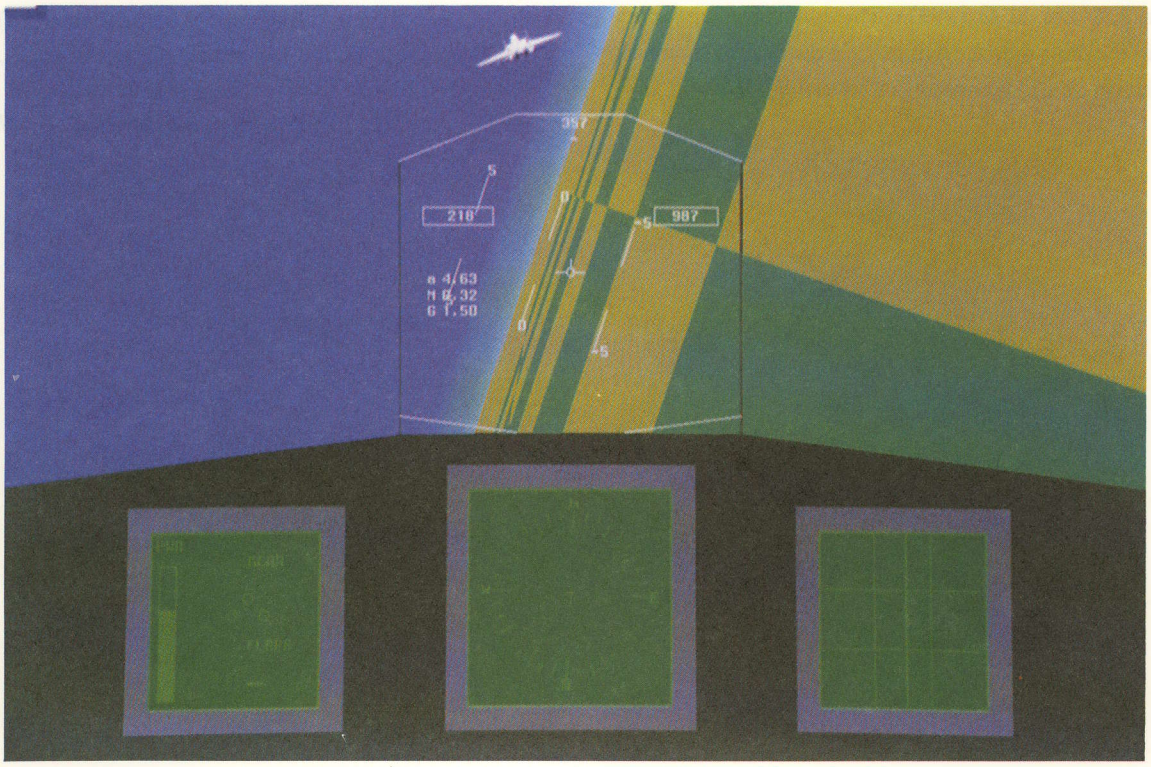

Figure 18. *Closing in.*

Acknowledgments

This work was supported by DARPA/ASTO, the Defense Modeling and Simulation Office, USA HQDA AI Center in the Pentagon, USA STRICOM, the National Science Foundation (Grant BCS-9109989), USA TRAC-Monterey, and the Naval Postgraduate School Direct Funding Program.

References

Anderson, J. D. (1989). *Introduction to flight.* New York: McGraw-Hill.

Burchfiel, J. (1990). The advantages of using quaternions instead of Euler angles for representing orientation. White Paper ASD-91-001, Third Workshop on Standard for the Interoperability of Defense Simulations, Orlando, FL.

Chou, J. C. K. (1992). Quaternion kinematics and dynamic differential equations. *IEEE Transactions on Robotics and Automation, 8*(1), 53–64.

Funda, J., Taylor, R. H. & Paul, R. P. (1990). On homogeneous transforms, quaternions, and computational efficiency. *IEEE Transactions on Robotics and Automation, 6*(3), 382–387.

Goldiez, B. & Lin, K.-C. (1991). The orientation representation in the draft military standard for distributed interactive simulation. *AAI,* Orlando, FL: University of Central Florida.

Goldstein, H. (1980). *Classical mechanics,* (2nd. ed.). Reading, MA: Addison-Wesley.

Mitchell, E. E., & Rodgers, A. E. (1965). Quaternion parameters in the simulation of a spinning rigid body. *Simulation, 18*(6).

Nelson, R. C. (1989). *Flight stability and automatic control.* New York: McGraw-Hill.

Paul, R. P. (1981). *Robot manipulators: Mathematics, programming and control.* Cambridge, MA: MIT Press.

Press, W. H., Flannery, B. P., Teukolsky, S. A. & Vetterling, W. T. (1990). *Numerical recipes in C: The art of scientific computing.* New York: Cambridge University Press.

Rolfe, J. M., & Staples, K. J. (1986). *Flight simulation.* New York: Cambridge University Press.

Roskam, J. (1979). *Airplane flight dynamics and automatic flight controls.* Roskam Aviation and Engineering Corporation.

Shoemake, K. (1985). Animating rotation with quaternion curves. *Computer Graphics, 19*(3), 245–254, SIGGRAPH Conference Proceedings.

Thorpe, J. A. (1987). The new technology of large scale simulator networking: Implications for mastering the art of warfighting. *Proceedings of the Ninth Interservice Industry Training Systems Conference.*

UCF/IST (1990). Military Standard (DRAFT) for Protocol Data Units for Distributed Interactive Simulation. University of Central Florida Institute for Simulation and Training.

Zyda, M. J., Pratt, D. R., Monahan, J. G. & Wilson, K. P. (1992). NPSNET: Constructing a 3D virtual world. *Proceedings of the 1992 Symposium on Interactive 3D Graphics.*