2000-11

Rigid Body Dynamics, Inertial Reference Frames, and Graphics Coordinate Systems: A Resolution of Conflicting Conventions and Terminology

McGhee, Robert

NPS-MV-01-002

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rigid Body Dynamics, Inertial Reference Frames, and Graphics Coordinate Systems: A Resolution of Conflicting Conventions and Terminology

by

Robert B. McGhee
Eric Robert Bachmann
Michael J. Zyda

November 2000

Prepared for: NPS MOVES Academic Group

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 2000	Technical Report

4. TITLE AND SUBTITLE
Rigid Body Dynamics, Inertial Reference Frames, and Graphics Coordinate Systems: A Resolution of Conflicting Conventions and Terminology

5. FUNDING
ARO, N6M

6. AUTHOR(S)
McGhee, Robert B., Bachmann, Eric R., and Zyda, Michael J.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-MV-01-002

9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The field of rigid body dynamics has a long tradition of study in physics and engineering, and has achieved a high degree of standardization and clarity of exposition. In comparison, computer graphics is a relatively new and rapidly growing field with competing standards and still evolving terminology. The authors believe that there is much to be gained by examining commonalities and differences between these two fields with regard to choice of coordinate systems and methods used for representing rigid body motion. Toward that end, in what follows, the basic ideas of rigid body dynamics are first developed in coordinate-free form. The standard coordinate system used in aircraft and ship dynamics (north, east, down) is then introduced and utilized to derive rigid body dynamics in a form suitable for computation, the "Newton Euler" equations. After this, the coordinate system associated with OpenGL, and other graphics API's, is compared to the standard dynamics coordinates. Finally, it is shown that it is possible to bring the dynamics and graphics perspectives on representation and display of rigid body motion into harmony through the use of a simple constant inverse homogeneous transformation matrix.

14. SUBJECT TERMS
Newton-Euler Equations, Rigid-Body Dynamics

15. NUMBER OF PAGES
15

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM David R. Ellison, USN
Superintendent

Richard Elster
Provost

This report was prepared for __NPS_MOVES Academic Group____
and funded by__Army Research Office (ARO), Modeling and Simulation Office (N6M)

This report was prepared by:

_____
Robert B. McGhee
Professor

_____
Eric R. Bachmann
Research Assistant Professor

_____
\Michael J. Zyda
Professor Professor

Reviewed by:

_____
Michael J. Zyda
Chair, MoVES Academic Group

Released by:

_____
D. W. Netzer
Associate Provost and
Dean of Research

Rigid Body Dynamics, Inertial Reference Frames, and Graphics Coordinate Systems: A Resolution of Conflicting Conventions and Terminology

R. B. McGhee, E. R. Bachmann, and M. J. Zyda

Naval Postgraduate School, Monterey, CA 93943
Email: mcghee@cs.nps.navy.mil

Abstract

The field of rigid body dynamics has a long tradition of study in physics and engineering, and has achieved a high degree of standardization and clarity of exposition. In comparison, computer graphics is a relatively new and rapidly growing field with competing standards and still evolving terminology. The authors believe that there is much to be gained by examining commonalities and differences between these two fields with regard to choice of coordinate systems and methods used for representing rigid body motion. Toward that end, in what follows, the basic ideas of rigid body dynamics are first developed in coordinate-free form. The standard coordinate system used in aircraft and ship dynamics (north, east, down) is then introduced and utilized to derive rigid body dynamics in a form suitable for computation, the "Newton Euler" equations. After this, the coordinate system associated with OpenGL, and other graphics API's, is compared to the standard dynamics coordinates. Finally, it is shown that it is possible to bring the dynamics and graphics perspectives on representation and display of rigid body motion into harmony through the use of a simple constant inverse homogeneous transformation matrix.

1. Introduction

In a previous paper [1] involving two of the authors, a quaternion approach to flight dynamics modeling and simulation was presented as a means for avoiding the Euler angle singularities associated with a vertical aircraft attitude. Since that time, the authors have received many requests for additional information and for further explanations of the mathematics and physical concepts underlying our simulation. In considering these requests, we have come to the realization that there are serious conflicts in terminology and notational conventions between the fields of physics and aerospace systems on the one hand, and graphics and virtual reality on the other. We have also noted that our students at the Naval Postgraduate School tend to mentally compartmentalize their knowledge of these two fields, and often become confused when forced to confront both when faced with physically based modeling and simulation in a virtual reality system. The purpose of this paper is to deal with these issues. We hope we have written a kind of "Rosetta Stone" paper which will facilitate communication between two areas of science and technology which have evolved somewhat separately, but have now come into a kind of "cultural collision."

The present paper begins with a discussion of the meaning of the term "inertial frame" and the realization of modern physics that this idea is somewhat circular. That is, the very idea of rigid body "motion" is relativistic, and cannot be defined in absolute terms. The concept of coordinate-free vector calculus is then introduced, and subsequently used to derive the "Newton-Euler" equations governing rigid body motion. As part of this discussion, it is shown that three apparently contradictory definitions of the most common set of Euler angles used to specify rigid body orientation are in fact the same, and are in complete agreement regarding their mathematical representation. In an appendix associated with this part of the paper, the non-orthogonal transformations linking body angular velocity to Euler angle rates are derived and explained. These transformations are generally unknown in the field of computer graphics, but are central to the Euler angle singularity problem of rigid body dynamics. A second appendix to this section shows how quaternion representation of rigid body orientation both eliminates the singularity problem and greatly simplifies coordinate transformations in comparison to the more usual matrix methods.

The second part of this paper explains computer graphics coordinate systems, and highlights conflicts with dynamics conventions. These conflicts are serious, and can lead to fatally flawed simulations, misunderstood results, and even loss of life or property when embodied in real physical systems. Fortunately, there is a solution to these problems involving a simple, but little understood, mathematical transformation. The authors hope our results will prove helpful in resolving the cultural and mathematical conflicts between the fields of computer graphics and rigid body dynamics.

2. Rigid Body Dynamics

Inertial Frames and Particle Dynamics

Newton's famous "Second Law of Motion" for mechanical systems is most often expressed as

$$f = ma \tag{1}$$

where f is "applied force" (or "contact" force), m is mass, and a is acceleration. A more precise statement of this law is

$$F + G = \frac{d}{dt}mV \qquad (2)$$

where F is a three-dimensional applied force vector, G is a three-dimensional "action at a distance" force vector (gravity, electrostatic, electromagnetic, etc.), and V is a three dimensional velocity vector.

A careful statement of either form of this law also requires the caveats that position and its time derivatives must be measured with respect to an "inertial frame" and that m is a point mass or "particle." However, this statement is deceptive because it turns out that there are no point masses in nature and that what constitutes an inertial frame depends on the physical and temporal scale of the problem being addressed. In fact, the only way to define an inertial frame is to say that it is a frame of reference attached to one or more physical bodies of a scale much larger than the problem under study, and such that the Second Law of Motion applies to an acceptable degree of accuracy for motion of a particle with respect to this frame. That is, the view of modern physics is that the very concept of motion cannot be defined except as the "relative displacement" of one body with respect to another. This was established in 1887 by the famous Michaelson-Morley experiment [2], which disproved the existence of an invisible "ether" relative to which all bodies or waves move.

To make the above notion of an inertial frame more understandable, it is useful to examine some specific examples. Consider, for instance, the interior of a vehicle moving at constant speed in an unchanging direction with respect to the Earth. Then for objects moving around inside the vehicle, the Second Law holds to a degree that is usually acceptable. That is, movement of humans and objects inside the vehicle appear to be unaffected by its motion. On the other hand, it is common knowledge that if the vehicle turns or accelerates longitudinally, coffee spills, passengers fall down, and other obvious phenomena make it clear that the Second Law no longer applies with respect to motion relative to the interior of the vehicle. To deal with an accelerating vehicle, it is usually possible to take the surface of the Earth as an inertial frame, so long as vehicle speed relative to the Earth is far below orbital velocity. When this is not the case, it is conventional to describe vehicle (satellite) motion in a solar frame of reference.[3] Likewise, solar system motion is usually studied relative to a galactic reference frame, etc.

Coordinate-Free Rigid Body Dynamics

All of the above discussion applies the Second Law to particles. However, this is actually an idealization, since real physical objects have finite density and therefore nonzero volume. An important special case of a body with volume is the "rigid body" idealization which assumes that a body of mass m also has a shape that cannot be changed. That is, such bodies are solid and completely inelastic. Fortunately, it can be shown that the Second Law also applies to rigid bodies so long as F is interpreted as the vector sum of all forces acting on the body, G represents forces exerted by a uniform gravitational field, and V is the velocity of the "center of gravity" (also called the "cg" or "centroid") of the body. The center of gravity is in turn defined as the unique point (possibly interior to the body) at which it is "balanced" in any orientation in the presence of a uniform gravitational field. Mathematically, this is equivalent to defining the cg as the unique point at which the first moment of mass is zero about any axis passing though the point [4].

To account for the motion of any point on the body other than the cg, it is necessary to deal with "rotational" dynamics (Figure1). Specifically, if V is the *translational* velocity of the cg, ω is the *angular* velocity vector (vector rate of rotation) of the body, and R_p is a vector from the cg to any point p on the body, then

$$V_p = V + \omega \times R_p \qquad (3)$$

where x denotes the vector cross-product [4]. Thus, while the angular velocity measured at any point on a rigid body is the same as at any other point on the body, this is not true of translational velocity. Granted these facts, it is natural to wonder if there is a rotational form of the Second Law. Not surprisingly, this is the case, with the law taking the form

$$M = \frac{d}{dt}H \qquad (4)$$

where H is the *angular momentum* vector and M is the vector sum of all "moments" acting on the body produced by forces whose "line of action" does not pass through the cg of the body. All of these terms require further explanation.

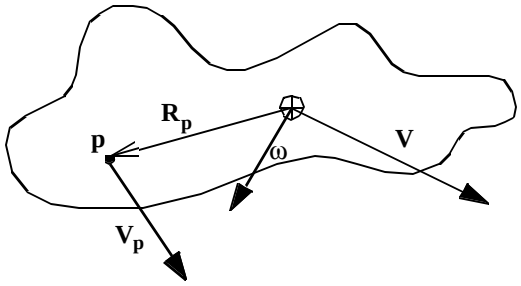

Figure 1: Rotational and Translational Motion of a Point on a Rigid-Body

The moment of a force acting on a rigid body is given by

$$M_i = R_i \times F_i \qquad (5)$$

where F_i is any applied force vector, and R_i is any vector extending from the cg of the body to a line collinear with F_i (the "line of action"). The *total moment* (or simply the moment) M is the vector sum of all such individual moments (Figure 2). The concept of angular momentum is somewhat more elusive, and is best developed for arbitrary rigid bodies after the selection of a specific coordinate system attached to the body. However, for the special case of a rigid body composed of a finite set of point masses (of course, with all pairwise distances between points constant), or "a system of particles," the definition is relatively straightforward. Specifically, the "translational" (or "lin-

ear") momentum of any particle i in a system of particles is defined as $m_i V_i$. The *angular momentum* or *moment of momentum* of such a particle is defined as

$$H_i = R_i \times m_i V_i \tag{6}$$

If H is defined as the sum of all H_i, then it is a straightforward exercise in vector calculus to show that Eq. (4) holds for a system of particles [4].

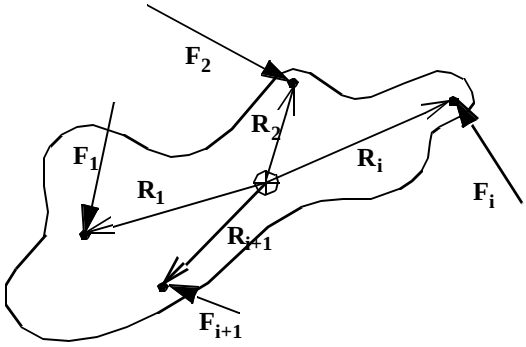

Figure 2: Total Moment of Force Acting on a Rigid-Body

Rigid Body Orientation

The above discussion is based on geometrical concepts and vector calculus, and treats rigid body dynamics at the highest possible level of abstraction consistent with mathematical rigor. However, computer simulation of rigid body dynamics must in the end reduce to scalar calculations, since this is the inherent nature of computer arithmetic. To permit this, and yet retain a level of abstraction suitable for human reasoning, it is conventional for computer simulation of rigid body motion to choose a coordinate system attached to an appropriate inertial frame, and then to express all vectors in component form relative to these coordinates. For motion on or near the surface of the Earth, at speeds far below orbital velocity, a commonly used coordinate system is the local "flat Earth" system with an arbitrarily selected origin on the surface of the Earth, near to the bodies to be simulated, and with coordinate axes x, y, and z directed in the local north, east and down directions respectively (Figure 3). To specify orientation, it is also necessary, for each rigid body, to specify a "body fixed" coordinate system attached to the body. This is also an *xyz* system with x conventionally "out the nose", z "out the belly," and y "out the right side." (Figure 4) The superscript or subscript "*E*" is often used to designate Earth coordinates, while "*B*" is typically used to signify body coordinates. This convention will be followed in the remainder of this paper.

The *reference orientation* for a rigid body is one in which all of its body-fixed axes are aligned with the corresponding Earth axes. In general, a body can be moved away from this orientation by rotating it about one or more of the Earth-fixed axes. If the order of rotations is first about a north axis, then about an east axis, and finally about a down axis, the associated angles are denoted by the reserved words *bank*, *elevation*, and *azimuth* respectively. When

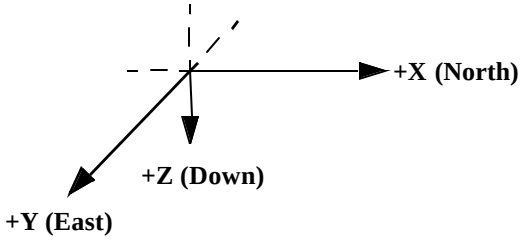

Figure 3: Conventional Aerospace North, East, Down Earth-Fixed Coordinate System

these terms are used, it is also required that the absolute value of elevation be not greater than $\pi/2$. When using the above set of "Euler angles," there are also reserved symbols for each angle; namely, bank is designated by ϕ, elevation by θ, and azimuth by ψ. In aircraft terminology, bank is sometimes referred to as "roll." However, as shown in appendix A, this "roll" angle is not the integral of the rate of rotation about the body-fixed nose axis so we avoid this confusing terminology. Observing these conventions is extremely important when individuals with a computer graphics background attempt to communicate with others working in the fields of engineering, navigation, air or sea traffic control, military operations, etc. Failure to do so not only endangers the accuracy of simulation systems, but can even lead to loss of equipment or human life when actual military or civil operations are involved.

It is important to realize that the above described set of Euler angles is not unique. In particular, rotating first about a north axis is an arbitrary (although universal in many fields) convention. That is, instead of the rotation order *xyz* used for the bank, elevation, azimuth system, any of the six permutations of these axis orders (such as *yxz*, for example) could just as well have been chosen as a standard (but none have been other than *xyz*). Less obviously, yet another suitable set of axes, used in physics to study the motion of spinning bodies such as tops, is the set *zyz*. If these Euler angles are used, the first rotation is about a local

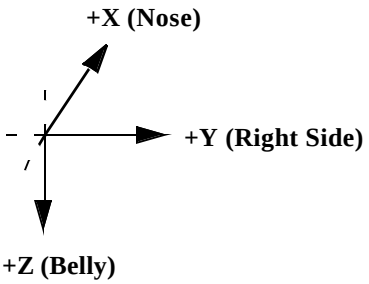

Figure 4: Conventional Aerospace Body-Fixed Coordinate System

down axis and is called *spin*, the second is about an east axis and is called *tilt*, and the third is again about a vertical axis and is called *precession*. Evidently, a *zxz* system would work just as well for this class of problems and, like the *zyz* system, can be used to specify the orientation of any rigid body, whether spinning or not. Likewise *xyx*, *xzx*, *yxy*, and *yzy* systems are perfectly general. In summary, taking all of the above into account, there are twelve possible distinct choices for Euler angles. Only three of these (*xyz*, *zxz*, *zyz*) are in common use, and only these have reserved words used to name the corresponding angles.

In mechanics, Euler angle rotations are often defined with respect to body-fixed axes rather than earth-fixed axes. At first, such an approach to specification of orientation would seem to potentially introduce yet another set of twelve Euler angles. Fortunately, and somewhat surprisingly, this is not the case. Instead, mechanical experiments (physics, not mathematics!) show that if the temporal order of rotations is reversed, body axis rotations yield exactly the same orientation as Earth axis rotations. Specifically, starting with a given body in its reference orientation, if it is first rotated through the azimuth angle about its belly axis, then through elevation about its right side axis, and finally through the bank angle about its nose axis, the final orientation of the body will be exactly the same as if these rotations had been performed in the reverse order about Earth axes [5]. This little appreciated fact can also lead to great confusion in discussions between individuals from different fields of science and technology.

While the above described physical facts are certainly not intuitively obvious, and may even appear to be paradoxical, there is yet another way of defining Euler angles which helps to resolve the apparent conflict. This third approach is derived from the terminology and practice of naval gunnery and field artillery, and predates the work of Newton and Euler by several centuries. In particular, to aim an artillery piece, it is necessary to tilt the gun barrel upward through an "elevation" angle so that a projectile will travel the desired distance (or "range") when the gun is fired. It is also necessary to rotate the gun carriage to a proper "azimuth" angle so it points toward the target. Finally, in most modern guns, when the projectile is fired, the "rifling" in the gun tube imparts a "roll rate" (or "spin") to the projectile to stabilize its flight toward a target. If the azimuth, elevation, and bank axes all intersect in a common point (true for some guns, but not all), then the mechanism that moves the gun is called a "gimbal" system. Thus, gimbal systems provide a mechanical means for achieving the rotations (without translation) discussed in the preceding paragraphs. Evidently, unlike the two previous definitions of Euler angles above, the "temporal" order of gimbal rotations does not matter. That is, the gun is "aimed" at the same point regardless of whether it is first rotated in azimuth, and then in elevation, or vice versa. If there is an actual adjustable bank angle (as in multibarrelled guns), the temporal order of this rotation is also of no significance.

A mechanical (or geometrical) proof of the equivalence of Earth-fixed axis rotations and body-fixed axis rotations (of course, with temporal reversal of order) can be accomplished through the following (real or mental) experiment. Consider a rigid body mounted on a gimbal system and in its reference configuration (nose axis north, side axis east, belly axis down). Now suppose bank (inner) gimbal rotation is accomplished first. In this case, since the nose gimbal axis is aligned with the Earth's north axis, this rotation corresponds to an Earth fixed *x*-axis rotation. Since this rotation does not involve elevation or azimuth, these axes are still aligned with the east and down axes respectively. That is, banking a gun tube has no effect on the gun azimuth or elevation Euler angles. This being the case, subsequent elevation of the gun will be about an east axis, and finally, azimuth gimbal axis rotation will be about down axis. This means that *xyz* rotations about Earth-fixed north, east, and down axes, in that order, are equivalent to gimbal rotations. Now suppose the rigid body is again in its reference configuration and that it is first rotated about its belly axis. Since this axis is aligned with the azimuth gimbal axis, this is an azimuth rotation. After this, the body side axis no longer points east, but it is still aligned with the elevation gimbal axis, so rotation about the side axis is an elevation rotation. Finally, by the same arguments, rotation about the body nose axis is in fact a bank gimbal rotation. This proves that Earth-fixed axis rotations are physically equivalent to body-fixed axis rotations providing the temporal order of rotations is reversed in the two cases[1]. For the interested reader, an on-line Virtual Reality Modeling Language (VRML) simulation of this "mechanical" proof is available at http://npsnet.org/~bachmann/orientation/orientation.wrl. Figure 5 below shows a typical frame from this simulation.

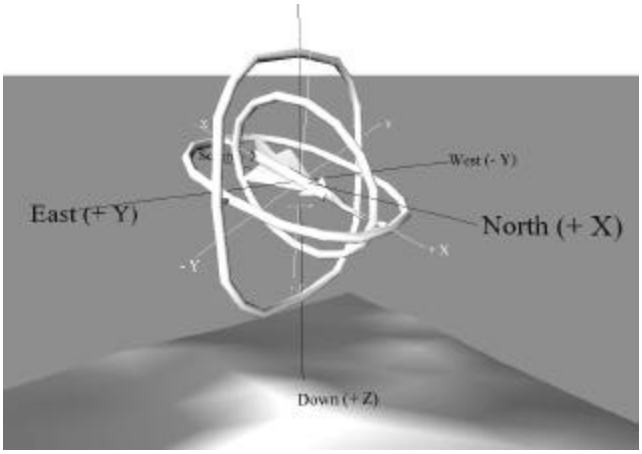

Figure 5: Virtual Gimbals - 3D Simulation

Angular and Linear Momentum for Arbitrary Rigid Bodies

In order to deal with arbitrary rigid body rotational dynamics, it is useful to define six "moment of inertia" constants derived from the mass distribution of the body relative to a specified body-fixed coordinate system (nose, side, belly coordinates). These are: [5],[6]

1. Unfortunately, some confusion on this point still exists in the literature. Specifically, the term "Euler angle" is sometimes used exclusively for body axis rotations and "fixed angle" is reserved for Earth axis rotations [5]. From the above discussion this distinction is evidently unnecessary.

$$I_{xx} = \int (y^2 + z^2)\,dm \tag{7}$$

$$I_{yy} = \int (x^2 + z^2)\,dm \tag{8}$$

$$I_{zz} = \int (x^2 + y^2)\,dm \tag{9}$$

$$I_{xy} = \int xy\,dm \tag{10}$$

$$I_{xz} = \int xz\,dm \tag{11}$$

$$I_{yz} = \int yz\,dm \tag{12}$$

where *dm* is an infinitesimal mass element and it is understood that the integral extends over the entire volume of the body. With these definitions, the "moment of inertia matrix" for the body is defined as:

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \tag{13}$$

Using *I*, if the angular velocity of the rigid body in the body-fixed coordinate system is

$$^B\omega = \begin{bmatrix} p & q & r \end{bmatrix}^T \tag{14}$$

where *p*, *q*, and *r* are the rotational rates about the body *x*, *y*, and *z* axes respectively, then it can be shown [5] that the angular momentum of the body is given by:

$$H = I\begin{bmatrix} p & q & r \end{bmatrix}^T \tag{15}$$

The components of $^B\omega$ are called "roll-rate", "yaw-rate", and "pitch-rate" respectively. In mechanics, these are reserved words while the symbols in Eq. (15) are likewise reserved[1]. In a similar way, there is a reserved set of symbols for the linear velocity of the center of gravity of a rigid body expressed in body coordinates. These are:

$$^BV = \begin{bmatrix} u & v & w \end{bmatrix}^T \tag{16}$$

The corresponding reserved names for these three linear velocity components are "surge", "sway", and "heave" respectively [6]. Using Eq. (16), linear momentum can be expressed in body coordinates as:

$$^BP = m\begin{bmatrix} u & v & w \end{bmatrix}^T \tag{17}$$

1. The terms "pitch" and "yaw" are sometimes used as being synonymous with elevation and azimuth respectively. However, since as shown in Appendix A, the elevation Euler angle is not the integral of pitch rate, and the azimuth Euler angle is likewise not the integral of yaw rate, the authors believe that this usage is misleading, and it has been avoided in this paper.

Newton-Euler Equations

With the above definitions of linear and angular momentum, the linear and angular behavior of an arbitrary rigid body under the influence of applied forces and moments can be determined by numerical integration of Eq. (2) and Eq. (4). However, before this can be done, it is necessary to choose a suitable "state vector" for such a body. While this vector is not unique, the usual choice is:

$$x = \begin{bmatrix} x_E & y_E & z_E & \phi & \theta & \Psi & u & v & w & p & q & r \end{bmatrix}^T \tag{18}$$

In this equation, all symbols have been defined above except for the first three, which stand for north position, east position, and down position respectively. The corresponding "state derivative", needed for numerical integration, is:

$$\dot{x} = \frac{dx}{dt} \tag{19}$$

Careful examination of the above two equations reveals some surprising mathematical difficulties arising from the fact that the last six components of the state vector are velocities relative to the Earth expressed in the (potentially moving) body-fixed coordinate system, while the first six components are the integral of these velocities, but only after they are transformed to Earth coordinates. To solve this problem, two transformation matrices, *R* and *T*, are introduced so that:

$$\frac{d}{dt}\begin{bmatrix} x_E & y_E & z_E \end{bmatrix}^T = R\begin{bmatrix} u & v & w \end{bmatrix}^T \tag{20}$$

and

$$\frac{d}{dt}\begin{bmatrix} \phi & \theta & \Psi \end{bmatrix}^T = T\begin{bmatrix} p & q & r \end{bmatrix}^T \tag{21}$$

The matrix *T* is derived in the attached Appendix A. The matrix *R* is given by

$$R = R_z(\Psi)R_y(\theta)R_x(\phi) \tag{22}$$

where the matrices on the right side of this equation are also given in Appendix A. It is important to realize that this value for R applies regardless of which of the three physical rotation methods discussed above is employed [5].

There is yet another more subtle problem associated with the above choice of state variables due to the fact that body-fixed coordinates do not in general define an inertial frame, and Newton's laws of motion (Eq. (2) and Eq. (4)) are valid only with respect to such a frame. To solve this problem, it is necessary to recognize that when a vector is represented in coordinate form relative to a rotating coordinate system, its derivative has two components; a "rate of growth" component relating to changes in the vector as viewed from the rotating frame of reference, and a "rate of transport" component due to the rotation of the moving coordinate system relative to the selected inertial frame. With respect to translational velocity, this fact can be observed in driving an automobile in which speed (rate of growth of velocity vector) is varied by use of the vehicle brakes and accelerator, while heading (direction of the velocity vector) is controlled by the steering wheel. Mathematically, this means that [2]

$$\frac{dV}{dt} = \begin{bmatrix}\dot{u} & \dot{v} & \dot{w}\end{bmatrix} + \begin{bmatrix}p & q & r\end{bmatrix} \times \begin{bmatrix}u & v & w\end{bmatrix} \quad (23)$$

Note that this expression gives linear acceleration relative to an earth-fixed inertial frame, but in body-fixed coordinates. Thus, referring to Eq. (2), it follows that:

$$\begin{bmatrix}F_x & F_y & F_z\end{bmatrix}^T + R^T \begin{bmatrix}0 & 0 & mg\end{bmatrix}^T = m\frac{dV}{dt} \quad (24)$$

and consequently,

$$\begin{bmatrix}\dot{u} & \dot{v} & \dot{w}\end{bmatrix}^T =$$

$$\frac{1}{m}\begin{bmatrix}F_x & F_y & F_z\end{bmatrix}^T + R^T \begin{bmatrix}0 & 0 & g\end{bmatrix}^T - \begin{bmatrix}p & q & r\end{bmatrix}^T \times \begin{bmatrix}u & v & w\end{bmatrix}^T \quad (25)$$

The above provides one of the two equations for the velocity part of $\dot{x}$. The corresponding rotational equations are obtained by noting that, since I is a constant matrix, it follows that:

$$\frac{dH}{dt} = \frac{d}{dt} I \begin{bmatrix}p & q & r\end{bmatrix}^T$$

$$= I\begin{bmatrix}\dot{p} & \dot{q} & \dot{r}\end{bmatrix}^T + \begin{bmatrix}p & q & r\end{bmatrix}^T \times I\begin{bmatrix}p & q & r\end{bmatrix}^T \quad (26)$$

Thus, from Eq. (4),

$$\begin{bmatrix}M_x & M_y & M_z\end{bmatrix}^T = I\begin{bmatrix}\dot{p} & \dot{q} & \dot{r}\end{bmatrix}^T + \begin{bmatrix}p & q & r\end{bmatrix}^T \times I\begin{bmatrix}p & q & r\end{bmatrix}^T \quad (27)$$

and therefore,

$$\begin{bmatrix}\dot{p} & \dot{q} & \dot{r}\end{bmatrix}^T = I^{-1}\begin{bmatrix}M_x & M_y & M_z\end{bmatrix}^T - I^{-1}\begin{bmatrix}p & q & r\end{bmatrix}^T \times I\begin{bmatrix}p & q & r\end{bmatrix}^T \quad (28)$$

Taken all together, Eq. (20), Eq. (21), Eq. (25), and Eq. (28) define all of the elements of $\dot{x}$, and therefore provide a complete characterization of rigid body dynamics in a form suitable for solution by numerical integration. Since Eq. (25) and Eq. (28) are derived from Newton's Second law, and make use of Euler angles to specify orientation, these two equations taken together are usually called the "Newton-Euler" equations [5]. A complete implementation of numerical solution of the Newton-Euler equations, including a selection of numerical integration formulas and all necessary vector-matrix computations, is available in ANSI Common Lisp at http://npsnet.org/~bachmann/.

Euler Angle Singularities

While the Newton Euler equations above appear to provide a complete characterization of rigid body motion, there remains a hidden problem having to do with Euler angle singularities. Specifically, when the nose unit vector of a rigid body points straight up (or down), the bank and azimuth gimbal axes are collinear. This means that neither bank nor azimuth angles are uniquely defined, but rather, only their difference (nose up) or sum (nose down) can be specified uniquely. This problem is also manifested in an even more serious way with respect to Euler angle rates since, as can be seen in Appendix A, the non-orthogonal body rate to Euler rate transformation matrix (T in Eq. (21)) is singular for this orientation ($\theta = \pm 90°$). There are two common solutions to this problem. One is to integrate body rotation rates for a short time interval, and then compute an "incremental" R matrix for the resulting small angles. If rates are integrated in Earth coordinates, then (from the discussion of rigid body rotation in earlier paragraphs) the incremental matrix multiplies R on the left. On the other hand, if rates are integrated in body coordinates, the multiplication is on the right. Either way, however, this method avoids the singularity in the T matrix since Eq. (21) is not used at all. Nevertheless, this is only an approximate method since the angle increments used are not Euler angle increments. Appendix B describes a more precise method involving transformation of rotation rates in body coordinates into quaternion rates, and subsequent integration of these rates to get a quaternion representation of orientation. Of course, the Euler angle singularity problem arises only for rigid bodies which are capable of assuming a vertical orientation.

3. COMPUTER GRAPHICS

Coordinate Axis Conventions

Computer graphics grew out of a desire to present the numbers produced as the output of digital computation in a form more easily understood. This need was perhaps most strongly felt by users of "Fortran", the first widely adopted scientific and engineering computer language. The first graphics facilities generally available to Fortran users were "line printer" plots obtained by using alphanumeric characters to represent values of two-dimensional data [7]. While high resolution "flat bed" mechanical plotters were also provided to some Fortran users, this was a very expensive alternative and was not generally available on an interactive basis to users of time-shared computers. This situation began to improve in the 1970's with the introduction of CRT terminals and minicomputers. In this time period, specialized "vector" graphics devices became available to represent computer output in real time as a set of straight lines ("vectors") drawn on an analog oscilloscope display. This in turn lead to the development of "wire frame" 3D graphical displays. This form of output was so computationally demanding that specialized "graphics computers" were developed to provide high resolution images in real time [8]. Subsequently, raster graphics display devices improved to such an extent that they generally displaced vector graphics display devices in all but a few highly specialized applications. At the same time, processor improvements lead to the replacement of specialized graphics computers by graphics workstations providing color 3D graphics with hidden surface elimination in real time [9]. Today, advances in personal computer hardware and the availability of general purpose graphics API's such as OpenGL have made real-time 3D graphics available to all computer users at a very low cost. This in turn has made 3D over the Internet feasible, and has lead to distributed virtual reality simulation systems [10].

The coordinate systems used by most graphics API's evolved from simple 2D graphics on a computer display screen surface. This lead to a different orientation of the coordinate axes than those depicted in Figure 3 or Figure 4. It remains a widespread convention in the graphics field to use the reserved symbol "*x*" as a default label for the horizontal coordinate and "*y*" for the vertical coordinate. In order to provide a basis for the application of perspective transformations within a 3D "viewing volume", a

"*z*" axis directed out of the display screen was added. Since, positive *y* is up on the display surface, and positive *x* is to the right, it follows that the *xyz* axes of this system define a set of "right handed" coordinates as depicted in Figure 6.

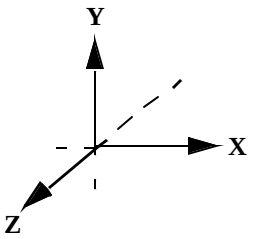

Figure 6: Typical Coordinate System Used by a Graphics API

Graphics Transformations and Coordinate Systems

In OpenGL, and in 3D graphics languages in general, rigid body surfaces are approximated by polygons which in turn are described by vertex locations. The vertices which describe an object are normally positioned relative to a coordinate system which has a fixed relation to the object itself. In OpenGL, this fixed coordinate system is termed *object coordinates* [11] and follows the convention depicted in Figure 6.

Several transformations are performed upon the vertex data as the associated polygons are processed by the rendering pipeline. Each of the transformations discussed here is performed using a 4 x 4 "homogeneous transform" matrix which takes the vertex coordinates from one coordinate system to another. Graphics systems usually use homogeneous transform matrices to treat rotation and translation in a uniform way [11]. Thus, both successive rotations *and* translations are accomplished by multiplication of 4x4 matrices.

Objects are positioned and oriented within a scene or relative to each other by *modeling transformations*. Modeling transformations take the form of a homogenous transform matrix *M*. The matrix *M* generally performs a rotation, translation, or both. The transformed vertex coordinates, v_w, of a vertex *v* are produced through the multiplication

$$v_w = Mv \tag{29}$$

The result of the transformation expressed by Eq. (29) is often termed *world coordinates* [12].

To individuals with a background in dynamics, it comes as a surprise to learn that OpenGL and similar languages make no mention of Euler angles. Instead, the orientation of an object is represented directly by its rotation matrix. The "elementary" rotation matrices R_x, R_y, and R_z are algebraically identical to the matrices defined in Appendix A, but the corresponding rotation angles are *not* called bank, elevation, and azimuth since these are reserved words used only for Earth coordinate axis rotations. Moreover, in OpenGL, no names at all are assigned to rotation angles. This is partly due to the fact that the idea of rotation about workstation coordinates is extended in OpenGL to be about *any* axis, with the axis being defined as a vector expressed in workstation coordinates. Rotations are therefore defined as "vector-angle pairs". Such a pair is very close to the concept of a quaternion as described in Appendix B.

Since rotations can be about any axis in OpenGL, there is no simple way to accumulate by numerical integration even one scalar angle, much less three "Euler" angles as in dynamics. Instead, all rotations in OpenGL are accomplished by simply pre-multiplying the current rotation matrix by any new rotation matrix. Thus, the idea of "Euler angle rates" does not even arise, and the *T* matrix derived in Appendix A is not used in OpenGL. This has the very significant result that the singularity of the *T* matrix (no inverse) that occurs at elevation angles of plus or minus $\pi/2$ has no relevance in graphics (unless Euler angles are needed for instrument display simulation as in aircraft flight simulation [1]). On the other hand, when rotation is represented directly by rotation matrices there is no concept of a state vector in a form suitable for rigid-body dynamics.

Once modeling transformations have been completed and all objects have been positioned in world coordinates, *viewing transformations* are performed to position the view volume. Positioning of the view volume is analogous to positioning and aiming a camera. The viewing transformations take the form of a 4 x 4 inverse homogenous transformation matrix. Application of modeling and then viewing transformations takes a vertex from world coordinates to *eye coordinates*. If the matrix V^{-1} represents the rotations and translations needed to position the view volume, then the coordinates of the vertex *v* in eye coordinates are given by

$$v_e = V^{-1}Mv \tag{30}$$

The matrix V^{-1} represents a combination of rotations and translations which are mathematically identical to the rotations and translations which are applied as modeling transformations. The only real difference is the order in which the transformations are applied to *v*. All modeling transformations described by *M* must be completed before the viewing transformations of V^{-1}. Thus in Eq. (30), the matrix *M* appears farthest to the right and nearest to *v*. This order is opposite the order the transformations appear in the application code, but is the actual order in which the transformations are carried out by the rendering pipeline.[11]

In effect, viewing transformations perform a translation from world coordinates into the eye coordinates of the "camera." It should be understood that a modeling transformation that rotates an object clockwise is equivalent to a viewing transformation that rotates the object counterclockwise. Thus, graphics programmers often find it easier to think of viewing transformations as moving all the objects in the world relative to a stationary camera instead of moving the camera position relative to stationary objects. Individuals with a background in dynamics may at times find this way of thinking alarming since it is clearly impossible in the physical world. In the mathematical world of graphics however, the two are equivalent.

Following the viewing transformation into eye coordinates, the OpenGL rendering pipeline subjects vertex data to several addition transformations which are strictly graphics related. The "projection transformation" produces

"clip coordinates." Subsequently, "normalized device coordinates" result from "perspective division." Finally, "window coordinates" are produced by the "viewport transformation." [11]

4. UNITING GRAPHICS AND DYNAMICS COORDINATE AND MOTION CONVENTIONS

In comparing graphics and dynamics conventions, it can be seen that graphics object coordinates and the body-fixed coordinate system described previously in Section 2 for use in representation of orientation under aerospace conventions serve similar purposes. The world coordinate system discussed above corresponds to the aerospace earth-fixed coordinate system in the same way. However, since graphics grew from the display of simple 2D images on a computer display screen surface, the orientation and arrangement of the coordinate axes do not match.

Though all systems are right-handed, comparison of Figure 6 to Figures 3 and 4 shows that the vertical axis under graphics conventions is the *y* axis with positive being up, while the vertical axis under aerospace conventions is the *z* axis with the positive direction being down. If it is desired that an existing graphics pipeline perform the rendering, this lack of correspondence makes is impossible to follow traditional aerospace conventions in physically based graphics applications without some type of modification.

Vertex locations expressed in a traditional north, east, down coordinate system can be translated to a *y* axis up system such as used in graphics API's using the inverse homogenous transform

$$H^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{31}$$

The matrix H^{-1} is the homogeneous transform from north, east, down coordinates to *y* axis up coordinates. Using this inverse it is possible to work with object coordinates and perform all modeling transformations with respect to a *z* axis down coordinate system such as depicted in Figures 3 and 4. Viewing transformations may also be completed relative to a conventional north, east, down coordinate system. In this case Eq. (30) becomes

$$v_e = H^{-1}V^{-1}Mv \tag{32}$$

The results of Eq. (32), may then be processed by a *y* axis up graphics pipeline. Note, that the inverse transform is applied to the vertex data last after all modeling and viewing transformations have be completed. With this approach, the Newton-Euler equations remain valid and the graphics and dynamics points of view are brought into harmony.

5. SUMMARY AND CONCLUSIONS

The fields of rigid body dynamics and computer graphics have confronted many of the same physical and mathematical problems, but have developed more or less independently and from quite different perspectives. With the advent of distributed virtual reality systems, these two fields have experienced a cultural collision which has exposed serious difficulties in terminology and coordinate conventions which have in turn inhibited dialogue between these two communities and have sometimes lead to incorrect or unnecessarily complicated simulations. In particular, singularity problems associated with Euler angle representation of rigid body orientation are little appreciated by the graphics community, and are often dealt with by mathematically incorrect "hacks". We hope this paper clarifies this problem and provides easily understood alternatives. We also hope that we have made the elegant and efficient use of quaternions for representation of rigid body orientation more accessible.

So far as the dynamics community is concerned, the use of workstation coordinates for dynamic simulation studies has proved baffling, since the Newton-Euler equations for rigid body motion are usually applied in a "north, east, down" Earth fixed coordinate system. We have shown that a simple fixed coordinate transformation resolves this problem both conceptually and with respect to actual graphics API code. Finally, we believe that we have brought together the physics and mathematics of rigid body dynamics in a simplified form which has previously been unavailable from a single source. We hope that future development of graphics API's will be affected by the results we have presented, and that careful choice of coordinates and terminology in such systems will serve to achieve a greater degree of harmony with the established conventions of physics and aerospace engineering.

6. ACKNOWLEDGEMENTS

The authors wish to acknowledge the support of the Navy Modeling and Simulation Management Office (N6M) and the Army Research Office (ARO) under Proposal No. 40410-MA for making this work possible.

7. REFERENCES

[1] Cooke, J. M., Zyda, M. J., Pratt, D. R., McGhee, R. B., "NPSNET: Flight Simulation Modeling Using Quaternions", *Presence*, Vol. 1, No. 4, Fall, 1992, pp. 404-420

[2] Synge, J. L., Griffith, B. A., *Principles of Mechanics*, Second Edition, McGraw-Hill Book Company, Inc., New York, New York, 1949.

[3] Sellers, J. J., *Understanding Space: An Introduction to Astronautics*, McGraw-Hill, Inc., New York, New York, 1994.

[4] Phillips, H. B., *Vector Analysis*, John Wiley & Sons, Inc., New York, New York, 1933.

[5] Craig, J., *Introduction to Robotics: Mechanics and Control*, Second Edition, Addison-Wesley Publishing

company, Inc., Menlo Park, California, 1989.

[6] Fossen, T. I., *Guidance and Control of Ocean Vehicles*, John Wiley & Sons Ltd.,Chichester, England, 1994.

[7] Chan, S. P., Chan S. Y., Chan S. G., *Analysis of Linear Networks and Systems: A Matrix-Oriented approach with Computer Applications*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1972.

[8] Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J. F., Phillips, R. L.,*Computer Graphics: Principles and Practice*, Second Edition in C, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1996.

[9] Zyda, M. J., McGhee, R. B., Ross R. S., Smith, D. B., Streyle D. G., "Flight Simulators for Under $100,000," IEEE Computer Graphics and Applications, Vol. 8, No. 1, January 1988, pp. 19-27.

[10] Zyda M., Singhal, S., *Networked Virtual Environments, Design and Implementation*, ACM Press, New York, New York, 1999.

[11] Woo, M., Neider J., Davis, T., and Shreiner, D., *OpenGL Programming Guide*, Third Edition, Addison-Wesley, Inc., Reading, Massachusetts, 1999.

[12] Hearn, D., Baker, M. P., *Computer Graphics: C Version*, Second Edition, Prentice Hall, Inc. Upper Saddle River, New Jersey, 1997.

[13] Kuipers, J. B., Quaternions and Rotation Sequences, Princeton University Press, Princeton, New Jersey, 1999.

Appendices

Appendix A: Derivation of Body Angular Rates to Euler Angle Rates Relationship

Let R_x, R_y and R_z be elementary rotation matrices about the local north, east and down axes respectively. Also, let ${}^B\omega$ be the angular rate of a rigid body measured in body coordinates and $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ the corresponding Euler angle rates of the body (bank, elevation, and azimuth rates, respectively). The angular velocity of a rigid body in earth coordinates, is then given by

$$ {}^E\omega = \begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix} + R_z\begin{bmatrix}0\\0\\\dot{\theta}\end{bmatrix} + R_zR_y\begin{bmatrix}\dot{\phi}\\0\\0\end{bmatrix} \tag{33} $$

Evidently

$$ {}^B\omega = \begin{bmatrix}p\\q\\r\end{bmatrix} = [R_zR_yR_x]^{-1}\ {}^E\omega = [R_zR_yR_x]^T\ {}^E\omega \tag{34} $$

Substituting Eq. (33) into Eq. (34), and using the inverse law of transposed matrices produces:

$$ {}^B\omega = R_x{}^TR_y{}^TR_z{}^T\ {}^E\omega = R_x{}^{-1}R_y{}^{-1}R_z{}^{-1}\ {}^E\omega \tag{35} $$

$$ = R_x{}^TR_y{}^TR_z{}^T\begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix} + R_x{}^TR_y{}^T\begin{bmatrix}0\\\dot{\theta}\\0\end{bmatrix} + R_x{}^T\begin{bmatrix}\dot{\phi}\\0\\0\end{bmatrix} \tag{36} $$

Since, [11]

$$ R_x = \begin{bmatrix}1 & 0 & 0\\ 0 & \cos\varphi & -\sin\varphi\\ 0 & \sin\varphi & \cos\varphi\end{bmatrix} \tag{37} $$

$$ R_y = \begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix} \tag{38} $$

$$ R_z = \begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix} \tag{39} $$

it follows from the first term of Eq. (36), that the rotational rate about an earth fixed down axis expressed in body coordinates is given by:

$$ {}^B\dot{\psi} = R_x{}^TR_y{}^TR_z{}^T\begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix} = R^T\begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix} = \dot{\psi}\begin{bmatrix}-\sin\theta\\ \sin\varphi\cos\theta\\ \cos\varphi\cos\theta\end{bmatrix} \tag{40} $$

In a similar manner, the following are obtained from the second and third terms of Eq. (36) respectively:

$$ {}^B\dot{\theta} = R_x{}^TR_y{}^T\begin{bmatrix}0\\\dot{\theta}\\0\end{bmatrix} = \dot{\theta}\begin{bmatrix}0\\\cos\varphi\\-\sin\varphi\end{bmatrix} \tag{41} $$

$$ {}^B\dot{\phi} = R_x{}^T\begin{bmatrix}\dot{\phi}\\0\\0\end{bmatrix} = \dot{\phi}\begin{bmatrix}1\\0\\0\end{bmatrix} \tag{42} $$

To obtain expressions for body rates in terms of Euler rates and angles, Eq. (40), Eq. (41) and Eq. (42) can be combined to produce

$$ p = \dot{\phi} - \dot{\psi}\sin\theta \tag{43} $$

$$ q = \dot{\theta}\cos\varphi + \dot{\psi}\sin\varphi\cos\theta \tag{44} $$

$$ r = -\dot{\theta}\sin\varphi + \dot{\psi}\cos\varphi\cos\theta \tag{45} $$

In order to solve for $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ in terms of *p*, *q*, and *r*, it is useful to note that Eq. (44) and Eq. (45) involve only $\dot{\theta}$, and $\dot{\psi}$. Thus, multiplying Eq. (44) by *cos* φ and Eq. (45) by *-sin* φ and adding produces the result

$$ \dot{\vartheta} = q\cos\varphi - r\sin\varphi \tag{46} $$

Substituting this result into Eq. (45) yields:

$$\dot{\psi} = q\frac{\sin\varphi}{\cos\varphi} + r\frac{\cos\varphi}{\cos\theta} = q\sec\theta\sin\varphi + r\sec\theta\cos\varphi \tag{47}$$

Finally, using this result in Eq. (43),

$$\dot{\varphi} = p + \dot{\psi}\sin\theta = p + q\tan\theta\sin\varphi + r\tan\theta\cos\varphi \tag{48}$$

In matrix form, these results can be rewritten as:

$$\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \tan\theta\sin\varphi & \tan\theta\cos\varphi\\ 0 & \cos\varphi & -\sin\varphi\\ 0 & \sec\theta\sin\varphi & \sec\theta\cos\varphi\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} = T\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{49}$$

where *sec* $\theta = 1 / cos\ \theta$. Evidently, the *T* matrix is singular for $\theta = \pm 90°$.

Appendix B: Quaternion Representation of Rigid Body Orientation

A "quaternion" is a four dimensional vector with an associated "quaternion product". There are three commonly used notations for quaternions. First of all, the "component" form is simply:

$$q = (q_0\ \ q_1\ \ q_2\ \ q_3) \tag{50}$$

where the parentheses denote a list, or equivalently, a vector. Another notation is the "flag" form:

$$q = q_0 + iq_1 + jq_2 + kq_3 \tag{51}$$

In this form, the symbols i, j, and k, are flags which in fact stand for the "unit vectors"

$$i = (0\ \ 1\ \ 0\ \ 0) \tag{52}$$

$$j = (0\ \ 0\ \ 1\ \ 0) \tag{53}$$

$$k = (0\ \ 0\ \ 0\ \ 1) \tag{54}$$

Equivalently, and more commonly, the flag form can be written:

$$q = w + ix + jy + kz \tag{55}$$

The third form of representation is:

$$q = w + su \tag{56}$$

or

$$q = w + v \tag{57}$$

where s is a scalar and u is any three-dimensional unit vector. In this representation, *w* is often called the "real" part and *v* the "vector" part of the quaternion. Evidently, these three representations are equivalent, and conversion from one form to another is trivial.

The quaternion product is defined as:

$$\begin{aligned} q_1q_2 = {} & (w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2)\\ & + i(x_1w_2 + w_1x_2 - z_1y_2 + y_1z_2)\\ & + j(y_1w_2 + z_1x_2 + w_1y_2 - x_1z_2)\\ & + k(z_1w_2 - y_1x_2 + x_1y_2 + w_1z_2) \end{aligned} \tag{58}$$

From this definition, evidently

$$i^2 = (0\ \ 1\ \ 0\ \ 0)(0\ \ 1\ \ 0\ \ 0) = (-1\ \ 0\ \ 0\ \ 0) \tag{59}$$

which is often abbreviated simply to:

$$i^2 = -1 \tag{60}$$

Likewise, again from the definition of the quaternion product,

$$j^2 = k^2 = -1 \tag{61}$$

and

$$ij = k = -ji \tag{62}$$

$$jk = i = -kj \tag{63}$$

$$ki = j = -ik \tag{64}$$

Equivalently, and more elegantly, Eq. (62), (63), and (64) can be summarized as:

$$ijk = -1 \tag{65}$$

Using these results, an alternative (but equivalent) definition of the quaternion product is often presented by taking Eq. (60), (61), and (65) as a given (axiomatic) "flag algebra", and then deriving Eq. (58)[1]. This approach is also useful for quaternion analysis, since all the rules of scalar arithmetic and scalar calculus apply to the flag representation of a quaternion. This is analogous to the conventional use of the symbol *i* (or *j*) to flag the "imaginary" part (second component) of a complex number (two dimensional vector with associated "complex" product).

The above defined flag algebra exhibits some of the characteristics of the corresponding flag algebras for two dimensional vectors (Eq. (60)), and some of those of three dimensional vectors(Eq. (65)). This leads to the third common definition of the quaternion product (equivalent to the two above), which is most easily derived using flag algebra. Specifically, if two quaternions are written in the form of Eq. (57), then:

$$q_1q_2 = (w_1w_2 - v_1 \cdot v_2 + w_1v_2 + w_2v_1 + v_1 \times v_2) \tag{66}$$

1. This approach is taken in [1], but the equation corresponding to Eq. (58) above has some typographical errors. Demonstration of the correctness of Eq. (58) is an easy exercise in quaternion flag algebra.

where x represents the standard vector "cross" product and "." stands for the standard vector "dot" product (square of vector length). The presence of the cross product term in this equation shows that, unlike the complex product, the quaternion product is not commutative.

In order to use quaternions to specify orientation, it is necessary to redefine the state vector for a rigid body as:

$$x = (x_e \;\; y_e \;\; z_e \;\; \varphi \;\; \theta \;\; \psi \;\; u \;\; v \;\; w \;\; q_0 \;\; q_1 \;\; q_2 \;\; q_3) \tag{67}$$

Since this is a vector of length 13, whereas the previously defined state vector (using Euler angles) was of length 12, it is conventional to apply the constraint that q is a unit vector. That is,

$$q \cdot q = {q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2 = 1 \tag{68}$$

When this is done, then it can be shown [1] that

$$\dot{q} = \frac{1}{2} q {}^B\omega = \frac{1}{2} q(0 \;\; p \;\; q \;\; r) \tag{69}$$

where, of course the indicated product is a quaternion product and the q component of ${}^B\omega$ is yaw rate, and not the quaternion q.

Given $\dot{q}$, q can be determined by numerical integration (followed by periodic normalization for correction of integration drift errors) and then used to compute a corresponding rotation matrix, thereby avoiding altogether the singularities of the body rate to Euler rate transformation matrix discussed in Appendix A. While this is done in [1], a more straightforward way is to simply use quaternion rotation of vectors by means of the equation [13]:

$${}^E v = q v q^{-1} \tag{70}$$

where (for unit quaternions)

$$q^{-1} = (q_0 \;\; -q_1 \;\; -q_2 \;\; -q_3) \tag{71}$$

A geometrical interpretation of this result can be obtained by rewriting q as:

$$q = \cos\frac{\alpha}{2} + u \sin\frac{\alpha}{2} \tag{72}$$

where u is a unit vector. In this form, u is the axis of rotation of any rigid body motion, and alpha is the angle of rotation. Code for this transformation and all other quaternion operations discussed in this appendix is available in ANSI Common Lisp at http://npsnet.org/~bachmann/.

From Eq. (70) and (71), evidently -q accomplishes the same rotation as q. If it is desired to eliminate this ambiguity, from Eq. (72), and if the range of values for α is restricted to $-\pi \leq \alpha \leq \pi$, then the real part of q will be non-negative. In this case, from Eq. (68),

$$q_0 = \sqrt{1 - {q_1}^2 - {q_2}^2 - {q_3}^2} \tag{73}$$

and q_0 can be eliminated from the state vector given by Eq. (67). This approach amounts to constraining quaternions to lie on a unit four dimensional hemisphere. Such quaternions can be referred to as being in their "positive real form" since q_0 is non-negative.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5100

3. Research Office, Code 09 1
   Naval Postgraduate School
   Monterey, CA 93943-5138

4. Dr. Michael Coyle 1
   Program Manager, Engineering Sciences Directorate,
   Mathematical and Computer Sciences Division
   U.S. Army Research Office
   P.O. Box 12211, 4300 S. Miami Blvd.
   Research Triangle Park, NC 27709

5. CNO, N6M1 1
   2000 Navy Pentagon
   Washington, DC 20350-2000