STRAPDOWN SYSTEM ALGORITHMS

AD-P003 621

By

Paul G. Savage
President
Strapdown Associates, Inc.
Woodbridge Plaza, Suite 150
10201 Wayzata Blvd.
Minnetonka, Minnesota 55343

SUMMARY

This paper addresses the attitude determination, acceleration transformation, and attitude/heading output computational operations performed in modern-day strapdown inertial navigation systems. Contemporary algorithms are described for implementing these operations in real-time computers. The attitude determination and acceleration transformation algorithm discussions are based on the two-speed approach in which high frequency coning and sculling effects are calculated with simplified high speed algorithms, with results fed into lower speed higher order algorithms. This is the approach that is typically used in most modern-day strapdown systems. Design equations are included for evaluating the performance of the strapdown computer algorithms as a function of computer execution speed and sensor assembly vibration amplitude/frequency/phase environment.

Both direction cosine and quaternion based attitude algorithms are described and compared in light of modern-day algorithm accuracy capabilities. Orthogonality and normalization operations are addressed for potential attitude algorithm accuracy enhancement. The section on attitude data output algorithms includes a discussion on roll/yaw Euler angle singularities near high/low pitch angle conditions.

1. INTRODUCTION

The concept of strapdown inertial navigation was originated more than thirty years ago, largely from an analytical standpoint. The theoretical analytical expressions for processing strapdown inertial sensor data to develop attitude, velocity, and position information were reasonably well understood in the form of continuous matrix operations and differential equations. The implementation of these equations in a digitial computer, however, was invariably keyed to severe throughput limitations of original airborne digitial computer technology. As a result, many of the strapdown computational algorithms originated during these early periods were inherently limited in accuracy, particulary under high frequency dynamic motion. A classical test for algorithm accuracy during this early period was how well the algorithm computed attitude under cyclic coning motion as the coning frequency approached the computer update cycle frequency.

In the late 1960's and early 1970's, several analytical efforts addressed the problem of splitting the strapdown computation process into low and high speed sections (7, 8, 10). The low speed section contained the bulk of the computational equations, and was designed to accurately account for low frequency large amplitude dynamic motion effects (e.g., vehicle maneuvering). The high speed computation section was designed with a small set of simple algorithms that would accurately account for high frequency small amplitude dynamic motion (e.g., vehicle vibrations). Splitting the computational process in this manner allowed the bulk of the strapdown algorithms to be iterated at reasonable speeds compatible with computer throughput limitations. The high speed algorithms were simple enough that they could be mechanized individually with special purpose electronics, or as a minor high speed loop in the main processor.

Over the past ten years, the structure of most strapdown algorithms has evolved into this two speed structure. The techniques have been refined today so that fairly straight-forward analytical design methods can be used to define algorithm analytical forms and computational rates to achieve required levels of performance in specified dynamic environments.

This paper describes the algorithms used today in most modern-day strapdown inertial navigation systems to calculate attitude and transform acceleration vector measurements from sensor to navigation axes. The algorithms for integrating the transformed accelerations into velocity and position data are not addressed because it is believed that these operations are generic to inertial navigation in general, not only strapdown inertial navigation.

For the algorithms discussed, the analytical basis is presented together with a discussion on general design methodology used to develop the algorithms for compatibility with particular user accuracy and environmental requirements.

2. STRAPDOWN COMPUTATION OPERATIONS

Figure 1 depicts the computational elements implemented by software algorithms in typical strapdown inertial navigation systems. Input data to the algorithms is provided from a triad of strapdown gyros and accelerometers. The gyros provide precision measurements of strapdown sensor coordinate frame ("body axes") angular rotation rate relative to nonrotating inertial space. The accelerometers provide precision measurements of 3-axis orthogonal specific force acceleration along body axes.

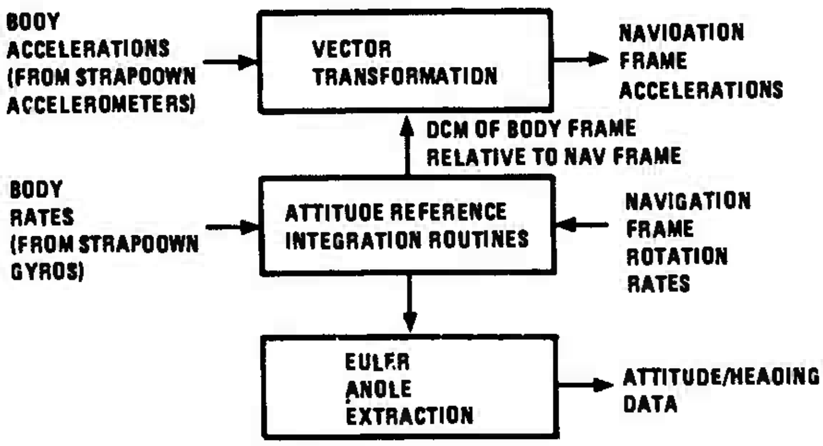

FIGURE 1 - STRAPDOWN ATTITUDE REFERENCE OPERATIONS

The strapdown gyro data is processed on an iterative basis by suitable integration algorithms to calculate the attitude of the body frame relative to navigation coordinates. The rotation rate of the navigation frame is an input to the calculation from the navigation section of the overall computation software. Typical navigation coordinate frames are oriented with the z-axis vertical and the x, y, axes horizontal.

The attitude information calculated from the gyro and navigation frame rate data is used to transform the accelerometer specific force vector measurements in body axes to their equivalent form in navigation coordinates. The navigation frame specific force accelerations are then integrated in the navigation software section to calculate velocity and position. The velocity/position computational algorithms are not unique to the strapdown mechanization concept, hence, are not treated in this paper. Several texts treat the velocity/position integration algorithms in detail (1, 2, 3, 4, 12).

Figure 1 also shows an Euler Angle Extraction function as part of the strapdown attitude reference operations. This algorithm is used to convert the calculated attitude data into an output format that is more compatible with typical user requirements (e.g., roll, pitch, heading Euler angles).

3. STRAPDOWN ATTITUDE INTEGRATION ALGORITHMS

The attitude information in strapdown inertial navigation systems is typically calculated in the form of a direction cosine matrix or as an attitude quaternion. The direction cosine matrix is a three-by-three matrix whose rows represent unit vectors in navigation axes projected along body axes. As such, the element in the i^{th} row and j^{th} column represents the cosine of the angle between the navigation frame i-axis and body frame j-axis. The quaternion is a four-vector whose elements are defined analytically (5, 9) as follows:

$$\begin{aligned}
a &= (\alpha_x/\alpha)\ \sin\ (\alpha/2) \\
b &= (\alpha_y/\alpha)\ \sin\ (\alpha/2) \\
c &= (\alpha_z/\alpha)\ \sin\ (\alpha/2) \\
d &= \cos\ (\alpha/2)
\end{aligned} \qquad (1)$$

where

$\alpha_x, \alpha_y, \alpha_z$ = Componante of an engla vector $\underline{\alpha}$.
α = Magnitude of $\underline{\alpha}$.

Tha $\underline{\alpha}$ vector is defined to have direction and magnitude such thet if the nevigetion frame wae roteted ebout $\underline{\alpha}$ through en angle α, it would be roteted into elignment with the body frame. The $\underline{\alpha}$ rotation angla vector end ita quaternion equivalent (a, b, c, d, from equetione (1)), or tha direction cosine matrix, uniquely defina tha ettitude of the body axea relative to navigation exes.

3.1 Direction Cosina Updeting Algorithms

3.1.1 Direction Cosina Updating Algorithm For Body Rotetions

Tha diraction cosine matrix can be updeted for body frame gyro sensad motion in tha strepdown computer by executing the following claasicel diraction cosine matrix chein rule elgorithm on e repetativa basis:

$$C(m+1) = C(m)\,A(m) \tag{2}$$

where

C(m) = Direction cosina matrix relating body to navigetion axaa at the m^{th} computer cycle time

A(m) = Direction cosine martix that transforms vectora from body coordinetas et the $(m+1)^{th}$ computer cycle to body coordinetes et the m^{th} computer cycle.

It is well known (9) thet:

$$A(m) = I + f_1(\underline{\phi}x) + f_2(\underline{\phi}x)^2 \tag{3}$$

where

$$f_1 = \frac{\sin\phi}{\phi} = 1 - \phi^2/3! + \phi^4/4! \quad -\cdots$$

$$f_2 = \frac{1-\cos\phi}{\phi^2} = 1/2! - \phi^2/4! + \phi^4/6! \quad -\cdots$$

$$\phi^2 = \phi_x{}^2 + \phi_y{}^2 + \phi_z{}^2 \tag{4}$$

$$(\underline{\phi}x) \overset{\Delta}{=} \begin{bmatrix} 0 & -\phi_z & \phi_y \\ \phi_z & 0 & -\phi_x \\ -\phi_y & \phi_x & 0 \end{bmatrix}$$

I = 3 x 3 unity matrix

ϕ_x, ϕ_y, ϕ_z = Components of $\underline{\phi}$.

$\underline{\phi}$ = Angle vector with direction end magnitude such thet e rotetion of the body frame about $\underline{\phi}$ through an angle equal to tha magnituda of $\underline{\phi}$ will rotate tha body frame from its oriantation at computar cycle m to its oriantation at computer cycle m+1. The $\underline{\phi}$ vector is computad for eech computer cycle m by processing the dete from the strepdown gyros. The elgorithm for computing $\underline{\phi}$ will ba describad subsequantly.

The "order" of the algorithm defined by equations (2) through (4) is determined by the number of terms carried in the f_1, f_2 expansions. A fifth order algorithm, for example, retains sufficient terms in f_1 and f_2 such that A(m) contains all $\underline{\phi}$ term products out to fifth order. Hence, f_1 would be truncated after the ϕ^4 term and f_2 would be truncated after the ϕ^2 term to retain fifth order accuracy in A(m). The order of accuracy required is determined by system accuracy requirements under maximum rate input conditions when $\underline{\phi}$ is a maximum. The computation iteration rate is typically selected to assure that $\underline{\phi}$ remains small at maximum rate (e.g., 0.1 radians). This assures that the number of terms required for accuracy in the f_1, f_2 expansions will be reasonable.

3.1.2 Direction Cosine Updating Algorithm For Navigation Frame Rotations

Equation (2) is used to update the direction cosine matrix for gyro sensed body frame motion. In order to update the direction cosines for rotation of the navigation coordinate frame, the following classical direction cosine matrix chain rule algorithm is used:

$$C(n+1) = B(n)\, C(n) \qquad (5)$$

where

B(n) = Direction cosine matrix that transforms vectors from navigation axes at computer cycle n to navigation axes at computer cycle (n+1).

The equation for B(n) parallels equation (3):

$$B(n) = I - (\underline{\theta}x) + 0.5(\underline{\theta}x)^2 \qquad (6)$$

with

$$(\underline{\theta}x) \stackrel{\Delta}{=} \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix} \qquad (7)$$

where

$\theta_x, \theta_y, \theta_z$ = Components of $\underline{\theta}$.

$\underline{\theta}$ = Angle vector with direction and magnitude such that a rotation of the navigation frame about $\underline{\theta}$ through an angle equal to the magnitude of $\underline{\theta}$ will rotate the navigation frame from its orientation at computer cycle n to its orientation at computer cycle n+1. The $\underline{\theta}$ vector is computed for each computer cycle n by processing the navigation frame rotation rate data from the navigation software section (12).

It is important to note that the n cycle (for navigation frame rotation) and m cycle (for body frame rotation) are generally different, n typically being executed at a lower iteration rate than m. This is permissable because the navigation frame rotation rates are considerably smaller than the body rates, hence, high execution rates are not needed to maintain $\underline{\theta}$ small to reduce the order of the iteration algorithm. The algorithm represented by equations (5) and (6) is second order in $\underline{\theta}$. Generally, first order is of sufficient accuracy, and the $(\underline{\theta}x)^2$ term need not be carried in the actual software implementation.

3.2 Quaternion Updating Algorithms

3.2.1 Quaternion Transformation Properties

The updating algorithms for the attitude quaternion can be developed through an investigation of its vector transformation properties (5, 9). We first introduce nomenclature that is useful for describing quaternion algebraic operations. Referring to equation (1), the quaternion with components a, b, c, d, can be described as:

$$u = ai + bj + ck + d \qquad (8)$$

where

a,b,c = Components of the "vector" part of the quaternion.

i,j,k = Quaternion vector operators analagous to unit vectors along orthogonal coordinate axes.

d = "Scalar" part of the quaternion.

We also define rules for quaternion vector operator products as:

$$\begin{matrix} ii = -1 & ij = k & ji = -k \\ jj = -1 & jk = i & kj = -i \\ kk = -1 & ki = j & ik = -j \end{matrix}$$

With the above definitions, the product w of two quaternions (u and v) becomes:

$$w = uv = (ai + bj + ck + d)\,(ei + fj + gk + h)$$

$$\begin{aligned} = \; & aeii + afij + agik + ahi \\ & + beji + bfjj + bgjk + bhj \\ & + ceki + cfkj + cgkk + chk \\ & + dei + dfj + dgk + dh \end{aligned}$$

$$\begin{aligned} = \; & (ah + de + bg - cf)i \\ & + (bh + df + ce - ag)j \\ & + (ch + dg + af - be)k \\ & + (dh - ae - bf - cg) \end{aligned}$$

or in "Four-vector" matrix form:

$$w \triangleq \begin{vmatrix} e' \\ f' \\ g' \\ h' \end{vmatrix} = \begin{bmatrix} d & -c & b & a \\ c & d & -a & b \\ -b & a & d & c \\ -a & -b & -c & d \end{bmatrix} \begin{vmatrix} e \\ f \\ g \\ h \end{vmatrix}$$

We also define the "complex conjugate" of the general quaternion u in equation (8) as:

$$u^* \triangleq -ai - bj - ck + d$$

We now define a quaternion operator h(m) for the body angle change $\underline{\phi}$ over computer cycle m as:

$$h(m) = \begin{matrix} (\phi_x/\phi)\sin(\phi/2) \\ (\phi_y/\phi)\sin(\phi/2) \\ (\phi_z/\phi)\sin(\phi/2) \\ \cos(\phi/2) \end{matrix} \qquad (9)$$

where the elements in the above column matrix refer to the i, j, k, and scalar components of h. We also define a general vector $\underline{v}$ with components v_x, v_y, v_z, and a corresponding quaternion v having the same vector components with a zero scalar component:

$$v = \begin{vmatrix} v_x \\ v_y \\ v_z \\ 0 \end{vmatrix}$$

Using the above definitions and the general rules for quaternion algebra, it is readily demonstrated by substitution and trigonometric manipulation that:

$$v' \triangleq h(m)\; v\; h(m)^* = A'(m)\; v \qquad (10)$$

where

$$A'(m) \triangleq \begin{bmatrix} A(m) & 0 \\ 0 & 0 \end{bmatrix}$$

$$v' \triangleq \begin{bmatrix} v_x' \\ v_y' \\ v_z' \\ 0 \end{bmatrix}$$

$A(m)$ = As defined in (3).

Equation (10), therefore, is the quaternion form of the vector transformation equation that transforms a vector from body coordinates at computer cycle (m+1) to body coordinates at computer cycle m:

$$\underline{v}' = A(m)\ \underline{v} \tag{11}$$

where

$\underline{v}', \underline{v}$ = "Three-vector" form of v' and v (i.e., with components v_x', v_y', v_z' and v_x, v_y, v_z).

$\underline{v}$ = The general vector $\underline{v}$ in body coordinates at computer cycle (m+1).

$\underline{v}'$ = The general vector $\underline{v}$ in body coordinates at computer cycle m.

3.2.2 Quaternion Updating Algorithm For Body Motion

Equation (10) with its equation (11) dual can be used to define analagous vector transformation operations between body coordinates and navigation coordinates at computer cycle m as:

$$\begin{aligned} v'' &= q(m)\ v'\ q(m)^* \\ \underline{v}'' &= C(m)\ \underline{v}' \end{aligned} \tag{12}$$

where

$q(m)$ = Quaternion relating body axes to navigation axes at computer cycle m.

$\underline{v}'$ = The vector $\underline{v}$ in navigation coordinates.

$\underline{v}''$ = The vector $\underline{v}$ in body coordinates at computer cycle m.

v', v'' = Quaternion ("Four vector") form of $\underline{v}'$, $\underline{v}''$.

The q quaternion has four elements (i.e., a, b, c, d) that are updated for body motion $\underline{\phi}$ at each computer cycle m. The updating equation is easily derived by substituting equation (10) into (12):

$$v'' = q(m)\ h(m)\ v\ h(m)^*\ q(m)^*$$

Using the definition for the quaternion complex conjugate, it is readily demonstrated that:

$$h(m)^*\ q(m)^* = (q(m)\ h(m))^*$$

Thus,

$$v'' = q(m)\ h(m)\ v\ (h(m)\ q(m))^*$$

But we can also write the direct expression:

$$v'' = q(m+1)\ v\ q(m+1)^*$$

Therefore, by direct comparison of the latter two equations:

$$q(m+1) = q(m)\ h(m) \tag{13}$$

Equation (13) is the quaternion equivalent to direction cosine updating equation (2). For computational purposes, $h(m)$ as defined in equations (9) is equivalently:

$$h(m) = \begin{vmatrix} f_3\ \phi_x \\ f_3\ \phi_y \\ f_3\ \phi_z \\ f_4 \end{vmatrix}$$

$$f_3 = \frac{\sin(\phi/2)}{\phi} = 0.5(1 - (0.5\phi)^2/3! + (0.5\phi)^4/5! - \cdots) \tag{14}$$

$$f_4 = \cos(\phi/2) = 1 - (0.5\phi)^2/2! + (0.5\phi)^4/4! - \cdots)$$

$$(0.5\phi)^2 = 0.25\ (\phi_x{}^2 + \phi_y{}^2 + \phi_z{}^2)$$

The "order" of the equation (13) and (14) updating algorithm depends on the order of ϕ terms carried in h which depends on the truncation point used in f_3 and f_4. The rationale for selecting the algorithm order and associated algorithm iteration rate is directly analagous to selection of the direction cosine updating algorithm order (discussed previously).

3.2.3 Quaternion Updating Algorithm For Navigation Frame Rotation

Equation (13) with (14) is used to update the quaternion for body frame motion sensed by gyros. In order to update the quaternion for rotation of the navigation coordinate frame, an algorithm analagous to equation (5) (for the direction cosine matrix) is used with a navigation frame rotation quaternion r:

$$q(n+1) = r(n)\ q(n)$$

$$r(n) = \begin{vmatrix} -0.5\ \theta_x \\ -0.5\ \theta_y \\ -0.5\ \theta_z \\ 1-0.5(\theta/2)^2 \end{vmatrix} \tag{15}$$

$$(\theta/2)^2 = 0.25\ (\theta_x{}^2 + \theta_y{}^2 + \theta_z{}^2)$$

where

$\theta_x, \theta_y, \theta_z$ = Components of $\underline{\theta}$ as defined previously for equations (6) and (7).

The development of equation (15) parallels the development of (13). The equation for r(n) is a truncated form of the theoretical exact analytical expression (analagous to the second order truncated form of equation (14)). The θ^2 term in equation (15) generally is not required for accuracy (due to the smallness of $\underline{\theta}$ in typical applications).

As for the direction cosine updating algorithm for navigation frame motion, the equivalent quaternion updating algorithm (equation (15)) updating cycle n need not be processed as fast as the body rate cycle m to maintain equivalent accuracy. This is due to the considerably smaller navigation frame rotation rates compared to body rotation rates.

3.2.4 Equivalancies Between Direction Cosine And Quaternion Elements

The analytical equivalency between the elements of the direction cosine matrix and the attitude quaternion can be derived by direct expansion of equations (12). If we define the elements of q as:

$$q = \begin{vmatrix} a \\ b \\ c \\ d \end{vmatrix}$$

equation (12) becomes after expansion, factorization of v', and neglecting the scalar part of the v" and v' quaternion vectors (i.e., carrying only the vector components $\underline{v}$" and $\underline{v}$'):

$$\underline{v}'' = \begin{bmatrix} (d^2 + a^2 - b^2 - c^2) & 2(ab - cd) & 2(ac + bd) \\ 2(ab + cd) & (d^2 + b^2 - c^2 - a^2) & 2(bc - ad) \\ 2(ac - bd) & 2(bc + ad) & (d^2 + c^2 - a^2 - b^2) \end{bmatrix} \underline{v}' \qquad (16)$$

Defining C in equation (12) as:

$$C = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

equation (16) when compared with (12) shows that:

$$\begin{aligned} C_{11} &= d^2 + a^2 - b^2 - c^2 \\ C_{12} &= 2(ab - cd) \\ C_{13} &= 2(ac + bd) \\ C_{21} &= 2(ab + cd) \\ C_{22} &= d^2 + b^2 - c^2 - a^2 \\ C_{23} &= 2(bc - ad) \\ C_{31} &= 2(ac - bd) \\ C_{32} &= 2(bc + ad) \\ C_{33} &= d^2 + c^2 - a^2 - b^2 \end{aligned} \qquad (17)$$

The converse of equation (17) is somewhat more complicated. Using the property (from equation (1)) that :

$$a^2 + b^2 + c^2 + d^2 = 1$$

the converse of equation (17) can be shown (11) to be computable from the following sequence of operations:

$$\begin{aligned} T_r &= C_{11} + C_{22} + C_{33} \\ P_1 &= 1 + 2C_{11} - T_r \\ P_2 &= 1 + 2C_{22} - T_r \\ P_3 &= 1 + 2C_{33} - T_r \\ P_o &= 1 + T_r \end{aligned}$$

If $P_1 = \max(P_1, P_2, P_3, P_o)$, then:

$$\begin{aligned} a &= 0.5\, P_1{}^{1/2}\, \mathrm{sign}(a_{previous}) \\ b &= (C_{21} + C_{12})/4a \\ c &= (C_{13} + C_{31})/4a \\ a &= (C_{32} - C_{23})/4a \end{aligned}$$

If $P_2 = \max(P_1, P_2, P_3, P_o)$, then:

$$\begin{aligned} b &= 0.5\, P_2{}^{1/2}\, \mathrm{sign}(b_{previous}) \\ c &= (C_{32} + C_{23})/4b \\ d &= (C_{13} - C_{31})/4b \\ a &= (C_{21} + C_{12})/4b \end{aligned} \qquad (18)$$

If $P_3 = \max(P_1, P_2, P_3, P_o)$, then:

$$\begin{aligned} c &= 0.5\, P_3{}^{1/2}\, \mathrm{sign}(c_{previous}) \\ d &= (C_{21} - C_{12})/4c \\ a &= (C_{13} + C_{31})/4c \\ b &= (C_{32} + C_{23})/4c \end{aligned}$$

If $P_o = \max(P_1, P_2, P_3, P_o)$, then:

$$\begin{aligned} d &= 0.5\, P_4{}^{1/2}\, \mathrm{sign}(d_{previous}) \\ a &= (C_{32} - C_{23})/4d \\ b &= (C_{13} - C_{31})/4d \\ c &= (C_{21} - C_{12})/4d \end{aligned}$$

3.3 The Computation Of $\underline{\phi}$

3.3.1 Continous Form

The $\underline{\phi}$ "body attitude change" vector is calculated by processing data from the strapdown gyros. Under situations where the angular rotation rate vector (sensed by the gyros) lies along a fixed direction (i.e., is nonrotating in inertial space), the $\underline{\phi}$ vector is equal to the simple integral of the angular rate vector over the time interval from computer cycle m to computer cycle (m+1):

$$\underline{\phi} = \int_{t_m}^{t_{m+1}} \underline{\omega}\, dt \qquad \text{for cases when } \underline{\omega} \text{ is nonrotating.} \tag{19}$$

where

$\underline{\omega}$ = Angular rate vector sensed by the strapdown gyros.

Under general motion conditions (when $\underline{\omega}$ may be rotating), equation (19) has the more complex form (as derived in (10) or alternatively, in Appendix A):

$$\underline{\alpha}(t) = \int_{t_m}^{t} (\underline{\omega} + 1/2\ \underline{\alpha} \times \underline{\omega} + \frac{1}{\alpha^2} (1 - \frac{\alpha \sin \alpha}{(1-\sin\alpha)})\ \underline{\alpha} \times (\underline{\alpha} \times \underline{\omega}))dt$$

$$\underline{\phi} = \underline{\alpha}(t=t_{m+1}) \tag{20}$$

It can verified by power series expansion that to first order,

$$(1/\alpha^2)\ (1 - \frac{\alpha \sin \alpha}{(1-\cos\alpha)}) = \frac{1}{12}$$

Hence, $\alpha(t)$ in equation (20), to third order accuracy in α can be approximated by:

$$\underline{\alpha}(t) = \int_{t_m}^{t} (\underline{\omega} + 1/2\ \underline{\alpha} \times \underline{\omega} + \frac{1}{12}\ \underline{\alpha} \times (\underline{\alpha} \times \underline{\omega}))dt \tag{21}$$

A second order expression for $\underline{\alpha}(t)$ can be obtained from (21) by dropping the 1/12 term. An even simpler expression for $\underline{\alpha}(t)$ is obtained by dropping the 1/12 term, and approximating the $\underline{\alpha}$ term in the integral by the direct integral of $\underline{\omega}$:

$$\underline{\beta}(t) = \int_{t_m}^{t} \underline{\omega}\, dt$$

$$\delta\underline{\beta}(t) = 1/2 \int_{t_m}^{t} \underline{\beta} \times \underline{\omega}\, dt \tag{22}$$

$$\underline{\phi} = \underline{\beta}(t=t_{m+1}) + \delta\underline{\beta}(t=t_{m+1})$$

An interesting characteristic about equation (22) is that its accuracy is in fact comparable to that of third order equation (21). In other words, the simplifying assumption of replacing α with β in the 1/2 $\underline{\alpha} \times \underline{\omega}$ term is in fact equivalent to introducing an error in equation (21) that to third order, equals the 1/12 $\underline{\alpha} \times (\underline{\alpha} \times \underline{\omega})$ term. This property can be verified by simulation as well as analytical expansion under hypothesized angular motion conditions.

Equation (22) is the equation that is mechanized in software in most modern-day strapdown inertial navigation systems to calcuate $\underline{\phi}$. It can be demonstrated analytically and by simulation that for representative vehicle angular motion and vibration, equation (22) faithfully calculates $\underline{\phi}$ to accuracy levels that are compatible with high performance strapdown inertial navigation system requirements.

For situations where $\underline{\omega}$ is nonrotating, the $\delta\underline{\beta}$ term in (22) is zero and ϕ equals the simple time integral or $\underline{\omega}$ over the computer interval m (i.e., the equation (19) approximation). For situations where $\underline{\omega}$ is rotating (a situation defined analytically as "coning"), the $\underline{\delta\beta}$ term is nonzero and must be calculated and used as a correction to the $\underline{\omega}$ integral to properly calculate $\underline{\phi}$.

It is important to note that the accuracy by which equation (22) approximates (20) is dependant on $\underline{\phi}$ being small (e.g., less than 0.1 radian). In order to protect the accuracy of this approximation, the computer iteration rate must be high enough that $\underline{\phi}$ remains small under maximum vehicle rotation rate conditions.

3.3.2 Recursive Algorithm Form

The implementation of equation (22) in a digital computer implies that a higher speed integration summing operation be performed during each body motion attitude update cycle. A computational algorithm for the integration function can be derived by first rewriting equation (22) in the equivalent incremental updating form:

$$\underline{\beta}(t) = \underline{\beta}(\ell) + \int_{t_\ell}^{t} \underline{\omega}\, dt$$

$$\underline{\delta\beta}(\ell+1) = \underline{\delta\beta}(\ell) + 1/2 \int_{t_\ell}^{t_{\ell+1}} \underline{\beta}(t) \times \underline{\omega}\, dt \qquad (23)$$

$$\underline{\beta}(\ell+1) = \underline{\beta}(t=t_{\ell+1})$$

$$\underline{\phi} = \underline{\beta}(t=t_{m+1}) + \underline{\delta\beta}(t=t_{m+1})$$

with initial conditions:

$$\underline{\beta}(t=t_m) = 0$$
$$\underline{\delta\beta}(t=t_m) = 0 \qquad (24)$$

where

ℓ = High speed computer cycle within the m body rate update cycle.

The integrals in (23) can be replaced by analytical forms that are compatible with gyro input data processing if $\underline{\omega}$ is replaced by a generalized time series expansion. For equations (23), it is sufficient to approximate $\underline{\omega}$ over the ℓ to $\ell+1$ time interval as a constant plus a linear ramp:

$$\underline{\omega} = \underline{A} + \underline{B}\,(t - t_\ell) \qquad (25)$$

where

$\underline{A}$, $\underline{B}$ = Constant vectors.

Substituting (25) in (23), and recognizing with the equation (25) approximation that:

$$\underline{A}(t_{\ell+1} - t_\ell) = 1/2\,\left(\underline{\Delta\theta}(\ell) + \underline{\Delta\theta}(\ell-1)\right)$$

$$1/2\,\underline{B}(t_{\ell+1} - t_\ell)^2 = 1/2\,\left(\underline{\Delta\theta}(\ell) - \underline{\Delta\theta}(\ell-1)\right)$$

where by definition:

$$\underline{\Delta\theta}(\ell) \triangleq \int_{t_\ell}^{t_{\ell+1}} \underline{\omega}\, dt$$

yields the desired final form for the $\underline{\phi}$ updating algorithm:

$$\underline{\delta\beta}(\ell+1) = \underline{\delta\beta}(\ell) + 1/2\,(\underline{\beta}(\ell) + 1/6\,\underline{\Delta\theta}(\ell-1)) \times \underline{\Delta\theta}(\ell)$$

$$\underline{\Delta\theta}(\ell) = \int_{t_\ell}^{t_{\ell+1}} \underline{\omega}\,dt = \sum_{t_\ell}^{t_{\ell+1}} \underline{d\theta} \qquad (26)$$

$$\underline{\beta}(\ell+1) = \underline{\beta}(\ell) + \underline{\Delta\theta}(\ell)$$

$$\underline{\phi} = \underline{\beta}(t=t_{m+1}) + \underline{\delta\beta}(t=t_{m+1})$$

with initial conditions:

$$\underline{\beta}(t=t_m) \triangleq \underline{\beta}(\ell=0) = 0$$

$$\underline{\delta\beta}(t=t_m) \triangleq \underline{\delta\beta}(\ell=0) = 0$$

where

$\underline{d\theta}$ = Gyro output pulse vector. Each component (x,y,z) represents the occurance of a rotation through a specified fixed angle increment about the gyro input axis.

$\underline{\Delta\theta}$ = Gyro output pulse vector count from ℓ to $\ell+1$.

The computational algorithm described by equation (26) is used on a recursive basis to calculata $\underline{\phi}$ once each m cycle. After $\underline{\phi}$ is calculated, the $\underline{\beta}$ and $\underline{\delta\beta}$ functions are reset for the next m cycle $\underline{\phi}$ calculation. The iteration rate for ℓ within $\overline{m}$ is maintained at a high enough rate to properly account for anticipated dynamic $\underline{\omega}$ motion effects. Section 6. describes analytical techniques that can be used to aeeess the adequacy of the ℓ iteration rate under dynamic angular rate conditiona.

3.4 The Computation Of $\underline{\theta}$

The $\underline{\theta}$ vector in equations (6) and (15) is computed as a simple integral of navigation frame angular rate over the n cycle iteration period:

$$\underline{\theta} = \int_{t_n}^{t_{n+1}} \underline{\Omega}\,dt \qquad (27)$$

where

$\underline{\Omega}$ = Navigation frame rotation rate as calculated in the navigation software section (12).

Standard recursive integration algorithms can be uaad to calculate $\underline{\theta}$ in equation (27) (e.g., trapezoidal) over the time intervel from n to n+1. The update rate for tha integration algorithm is eelacted to be compatible with software accuracy requirements in the anticipated dynamic maneuver environment for the user vehicle.

3.5 Orthogonality And Normalization Algorithms

Most strapdown attitude computation techniquas periodically employ self-consistancy correction algorthms as an outer-loop function for accuracy enhancement. If the basic attituda data is computed in the form of a direction cosine matrix, the self-consistancy check is that the rows should be orthogonal to each other and equal to unity in magnitude. This condition is besed on tha fact that the rows of the direction cosine matrix represent unit vectors along orthogonal navigation coordinate frame axea as projected in body axes. For the quaternion, the self-consistancy check is thet the sum of the squares of the quaternion elemants be unity (this can be verified by operation on equation (1)).

3.5.1 Direction Cosine Orthogonalization And Normalization

The test for orthogonality between two direction cosine rows is that the dot product be zero. The error condition, than is:

$$E_{ij} = C_i C_j^T \tag{28}$$

where

C_i = i^{th} row of C

C_j = j^{th} row of C

T = Transpose

A calculated orhogonality error E_{ij} can be corrected by rotating C_i and C_j relative to each other about an axie perpendicular to both by the srror angle E_{ij}. Since it is not known whether C_i or C_j is in error, it is assumed that each are equally likely to be generating the error, and each is rotated by half of E_{ij} to correct the error. Hence, the orthogonality correction algorithm is:

$$\begin{aligned} C_i(n+1) &= C_i(n) - 1/2\ E_{ij}\ C_j(n) \\ C_j(n+1) &= C_j(n) - 1/2\ E_{ij}\ C_i(n) \end{aligned} \tag{29}$$

It is easily verified using (29) that an orthogonality error E_{ij} originally present in $C_i(n)$ and $C_j(n)$ is no longer present in $C_i(n+1)$ and $C_j(n+1)$ after application of equation (29).

The unity condition on C_i (i.e., normality) can be tested by comparing the magnitude squared of C_i with unity:

$$E_{ii} = 1 - C_i C_i^T \tag{30}$$

A measured normality error E_{ii} can be corrected with:

$$C_i(n+1) = C_i(n) - 1/2\ E_{ii}\ C_i\ (n) \tag{31}$$

Equations (28) through (31) can be used to measure and correct orthogonality and normalization errors in the direction cosine matrix. In combined matrix form, the overall measurement/correction operation is sometimes written as:

$$C_{n+1} = C_{n+1/2}\ (I - C_n\ C_n^T)\ C_n \tag{32}$$

3.5.1.1 Rows or Columns - The previous discussion addressed the problem of orthogonalizing and nomalizing the rows of a direction cosine matrix C. In combined form, equation (32) shows that the corrsction is:

$$\delta C = 1/2\ (I - CC^T)\ C \tag{33}$$

Equation (33) can be operated upon by premultiplication with C postmultiplication by C^T, and combining terms. The result is:

$$\delta C = 1/2\ C\ (I - C^T C) \tag{34}$$

The $(I - C^T C)$ term in (34) is the error matrix based on testing orthogonality and normality of the columns of C. Thus, if the rows of C are orthonormalized (i.e., δC is nulled), the columns of C will also be implicitly orthonormalized. The inverse applies if the columns are directly orthonormalized with (34). The question that remains is, which is preferred? The answer is related to the real time computing problem associated with the calculation and correction of orthogonalization and normalizstion errors.

Ideally, the orthogonalization and normalization operations are performed as an outer loop function in a strapdown navigation computer so as not to impact computer throughput requirements. A computstional organization that facilities such an approach divides the orthonormalization operations into submodules that are executed on successive passes in the outer-loop software path. A logical division of the orthonormalization operations into submodules is as defined by equations (28), (29), (30), and (31).

This implies that measurement and correction of orthogonalization and normslization effects are performed at different times in the computing cycle. Such an approach is only valid if the orthogonality and normalizations errors (i.e., E_{ij} and E_{ii}) remain reasonably stable as a function of time.

To assess the time stability of the orthogonality/normalizstion error is to investigate the rate of change of the bracketed terms in equations (33) and (34). For convenience, these will be defined as:

$$E_R \stackrel{\Delta}{=} (I - CC^T)$$
$$E_C \stackrel{\Delta}{=} (I - C^TC) \qquad (35)$$

The time derivative of (35) is:

$$\dot{E}_R = - \dot{C}C^T - C\dot{C}^T$$
$$\dot{E}_C = - \dot{C}^TC - C^T\dot{C} \qquad (36)$$

Expressions for $\dot{C}$ and $\dot{C}^T$ can be developed by returning to equations (2), (3), (5), and (6). These equations can be rearranged to show that over a given time interval, the change in C is given by:

$$\Delta C = C(A - I) + (B - I)C$$

which with (3) and (4) becomes to first order:

$$\Delta C = C(\underline{\phi}x) - (\underline{\theta}x)C \qquad (37)$$

Dividing by the time interval for the change in C, recognizing that $\underline{\phi}$ and $\underline{\theta}$ are approximately integrals of $\underline{\omega}$ and $\underline{\Omega}$ over the time interval, and letting the time interval go to zero in the limit, yields the classical equation for the rate of change of C:

$$\dot{C} = C(\underline{\omega}x) - (\underline{\Omega}x)C \qquad (38)$$

where

$(\underline{\omega}x)$, $(\underline{\Omega}x)$ = Skew symmetric matrix form of vectors $\underline{\omega}$, $\underline{\Omega}$.

The transpose of (38) is :

$$\dot{C}^T = - (\underline{\omega}x)\, C^T + C^T\, (\underline{\Omega}x) \qquad (39)$$

We now substitute (38) and (39) into (36). After combining terms and applying equations (35), the final result is:

$$\dot{E}_R = E_R\, (\underline{\Omega}x) - (\underline{\Omega}x)\, E_R$$
$$\dot{E}_C = E_C\, (\underline{\omega}x) - (\underline{\omega}x)\, E_C \qquad (40)$$

Equations (40) show that the rate of change of E_R is proportional to E_R and the navigation frame rotation rate $\underline{\Omega}$, whereas the rate of change of E_C is proportional to E_C and the body rotation rate $\underline{\omega}$. Since $\underline{\omega}$ is generally much larger than $\underline{\Omega}$, $\dot{E}_C$ is generally larger than $\dot{E}_R$. It can be concluded that E_R is more stable over time, hence, orthonormalizing the direction cosine matrix rows (based on the E_R measurement) is the preferred computational approach if the real time computing problem is taken into account.

3.5.2 Quaternion Normalization

The quaternion is normalized by measuring its magnitude squared compared to unity, and adjusting each element proportionally to correct the normalization error. The normalization error is given by:

$$E_q = q\, q^* - 1 \qquad (41)$$

It is easily verified using the rules for quaternion algebra that E_q equals the sum of the squares of the elements of q minus 1. The correction algorithm is given by:

$$q_{(n+1)} = q_{(n)} - 1/2\, E_q\, q_{(n)} \qquad (42)$$

3.6 Direction Cosine Versus The Quaternion For Body Attitude Referencing

The tradeoff between direction cosine versus quaternion parameters as the primary attitude reference data in strapdown inertial systems has been a popular area of debate between strapdown analysts over the past three decades. In its original form, the tradeoff centered on the relative accuracy between the two methods in accounting for body angular motion. These tradeoffs invariably evolved from the differential equation form of the direction cosine and quaternion updating equations and investigated the accuracy of equivalent algorithms for integrating these equations in a digital computer under hypothesized body angular motion. Invariably, the body motion investigated was coning motion at various frequencies relative to the computer update frequency. For these early studies, the tradeoffs generally demonstrated that for comparable integration algorithms, the quaternion approach generated solutions that more accurately replicated the true coning motion for situations where the coning frequency was within a decade of the computer update frequency.

As presented in this paper, both the quaternion and direction cosine updating algorithms have been based on processing of a body angle motion vector $\underline{\phi}$ which accounts for all dynamic motion effects including coning. These updating algorithms (equation (2) and (3) for direction cosines and (13) and (14) for the quaternion) represent exact solutions for the attitude updating process for a given input angle vector $\underline{\phi}$. Consequently, the question of accuracy for different body motion can no longer be considered a viable tradeoff area. The principle tradeoffs that remain between the two approaches are the computer memory and throughput requirements associated with each in a strapdown navigation system.

In order to assess the relative computer memory and throughput requirements for quaternion parameters versus direction cosines, the composite of all computer requirements for each must be assessed. In general, these can be grouped into three major computional areas:

1. Basic updating algorithm
2. Normalization and orthogonalization algorithms
3. Algorithms for conversion to the direction cosine matrix form needed for acceleration transformation and Euler angle extraction

<u>Basic Updating Algorithms</u> - The basic updating algorithm for the quaternion parameters is somewhat simpler than for direction cosines as expansion of equations (2) and (3) compared with (13) and (14) would reveal. This results in both a throughput and memory advantage for the quaternion approach. Part of this advantage arises because only four quaternion elements have to be updated compared to nine for direction cosines. The advantage is somewhat diminished if it is recognized that only two rows of direction cosines (i.e., 6 elements) need actually be updated since the third row can then be easily derived from the other two by a cross-product operation (i.e., the third row represents a unit vector along the z-axis of the navigation frame as projected in body axes. The first two rows represent unit vectors along x and y navigation frame axes. The cross-product of unit vectors along x and y navigation axes equals the unit vector along the z-navigation axis).

<u>Normalization And Orthogonalization Algorithms</u> - The normalization and orthogonalization operations associated with direction cosines are given by equation (28) through (31). The quaternion normalization equation is given by equations (41) and (42).

The normalization equation for the quaternion is generally simpler to implement than the orthogonalization and normalization equations for the direction cosines. If only two rows of the direction cosine matrix are updated (as described in the previous paragraph) the direction cosine orthogonalization and normalization operations required are half that dictated by (28) through (31), but are still more than required by (41) and (42) for the quaternion. Since the orthonormalization operations would in general be iterated at low rate, no throughput advantage results for the quaternion. Some memory savings may be realized, however.

A key factor that must be addressed relative to orthonormalization tradeoffs is whether or not orthonormalization is actually needed at all. Clearly, if the direction cosine or quaternion updating algorithms were implemented perfectly, orthonormalization would not be required. It is the author's contention that, in fact, the accuracy requirements for strapdown systems dictate that strapdown attitude updating software cannot tolerate any errors whatsoever (compared to sensor error effects). Therefore, if the attitude updating software is designed for negligible drift and scale factor error (compared to sensor errors) it will also implicity exhibit negligible orthogonalization and/or normalization errors.

The above argument is valid if the effect of orthonormalization errors in strapdown attitude data is no more detrimental to system performance than other software attitude error effects. This is in fact the case, as detailed error analyses would reveal. Since modern-day general purpose computers used in today's strapdown inertial navigation systems have the capability to implement attitude updating algorithms essentially perfectly within a reasonable throughput and memory requirement, it is the author's opinion that orthonormalization error correction should not be needed, hence, is not a viable tradeoff area relative to the use of quaternion parameters versus direction cosines.

<u>Algorithms For Conversion To The Direction Cosine Matrix</u> - If the basic calculated

attitude data is direction cosines directly, no conversion process is required. For cases where only two rows of direction cosines are updated, the third row must be generated by the cross-product between the two rows calculated. For example:

$$\begin{aligned} C_{31} &= C_{12}\,C_{23} - C_{13}\,C_{22} \\ C_{32} &= C_{13}\,C_{21} - C_{11}\,C_{23} \\ C_{33} &= C_{11}\,C_{22} - C_{12}\,C_{21} \end{aligned} \qquad (43)$$

For quaternion parameters, equation (17) must be implemented to develop the direction cosine matrix, a significantly more complex operation compared with (43) for the two row direction cosine approach. Since direction cosine elements are generally required at high rate (for acceleration transformation and Euler angle output extraction) both a throughput and memory penalty is accrued for the quaternion approach. The penalty is compounded if the calculated direction cosine outputs are required to greater than single precision accuracy (including computational round-off error). For noise-free acceleration transformation operations (such as may be needed to effect an accurate system calibration) double-precision accuracy is needed. The result is that equation (17) for the quaternion versus (43) for direction cosines would have to be implemented in double-precision imposing a significant penalty for the more complex quaternion conversion process.

<u>Tradeoff Conclusions</u> - From the above qualitative discussion, it is difficult to draw hard conclusions regarding a preference for direction cosines versus quaternion parameters for attitude referencing in strapdown inertial systems. Pros and cons exist for each in the different tradeoff areas. Quantitative comparisons based on actual software sizing and computer loading studies have led to similar inconclusive results. Fortunately, today's computer technology is such that the slight advantage one attitude parameter approach may have over the other in any particular application is insignificant compared with composite total strapdown inertial system throughput and memory software requirements. Hence, ultimate selection of the attitude approach can be safely made based on "analyst's choice".

4. STRAPDOWN ACCELERATION TRANSFORMATION ALGORITHMS

The acceleration vector measurement from the accelerometers in a strapdown inertial system is transformed from body to navigation axes through a mechanization of the classical vector tranformation equation:

$$\underline{a}^N = C\,\underline{a} \qquad (44)$$

where

$\underline{a}$ = Specific force acceleration measured in body axes by the strapdown accelerometers

$\underline{a}^N$ = Specific force acceleration with components evaluated along navigation axes.

The implementation of equation (44) is accomplished on a repetative basis as a recursive algorithm in a digital computer such that its integral properties are preserved at the computer cycle times. In this manner, the velocity which is formed from the integral of (44) will be accurate under dynamic conditions in which $\underline{a}^N$ may have erratic high frequency components. The recursive algorithm for (44) must account for the effects of body rotation (and secondarily, rotation of the navigation coordinate frame) as well as variations in $\underline{a}$ over the computer iteration period.

4.1 <u>Acceleration Transformation Algorithm That Accounts For Body Rotation Effects</u>

To develop an algorithm for equation (44) that preserves its integral properties, we begin with its integral over a computer cycle:

$$\underline{u}^N = \int_{t_m}^{t_{m+1}} C\,\underline{a}\,dt \qquad (45)$$

where

$\underline{u}^N$ = Change in the integral of equation (44) (or specific force velocity change) over a computer cycle m

The velocity vector in the navigation computer is generated by summing the $\underline{u}^N$'s corrected for Coriolis and gravity effects.

The C matrix in (45) is a continuous function of time in the interval from t_m to t_{m+1}. An equivalent form for C in terms of its value at the computer update time (m) is:

$$C = C(m)\,A(t) \qquad (46)$$

whare

$C(m)$ = Value of C at t_m

$A(t)$ = Direction cosine matrix that traneform vectore from body axee at tima t to the body attitude at the etart time for the computation interval t_m.

Equation (46) with the dafinition for A(t) above accounte for tha affect of gyro seneed body motion over the computar intarval. The next section will diecuas the correction usad to account for tha small rotation of the navigation frame ovar the computer interval.

Subatituting (46) in (45) and expanding:

$$\underline{u}^N = C(m) \int_{t_m}^{t_{m+1}} A(t)\, \underline{a}\, dt$$

We now uee a first order approximation for A(t) as given by equation (3), with $\underline{\phi}$ traatad as a function of time in the interval as definad to first ordar in equation (22):

$$\underline{\phi}(t) = \underline{\beta}(t) = \int_{t_m}^{t} \underline{\omega}\, dt$$

Thus,

$$A(t) \approx I + (\underline{\beta}(t)\times) \qquad (47)$$

and

$$\underline{u}^N \approx C(m) \int_{t_m}^{t_{m+1}} (I + (\underline{\beta}(t)\times))\, \underline{a}\, dt$$

$$= C(m) \left(\int_{t_m}^{t_{m+1}} \underline{a}\, dt + \int_{t_m}^{t_{m+1}} (\underline{\beta}(t) \times \underline{a})\, dt\right)$$

We now dafine

$$\underline{u} \overset{\Delta}{=} \int_{t_m}^{t_{m+1}} \underline{a}\, dt$$

Hence,

$$\underline{u}^N = C(m) \left(\underline{u} + \int_{t_m}^{t_{m+1}} (\underline{\beta}(t) \times \underline{a})\, dt\right) \qquad (48)$$

with

$$\underline{\beta}(t) = \int_{t_m}^{t} \underline{\omega}\, dt$$

$$\underline{u} = \int_{t_m}^{t_{m+1}} \underline{a}\, dt$$

An alternative form of (48) can also ba derived through diract application of the integration by parts rula to tha integral term in tha equation (48) $\underline{u}^N$ axpression.:

$$\underline{u}^N = C(m) \left(\underline{u} + 1/2\, \underline{\beta} \times \underline{u} + 1/2 \int_{t_m}^{t} (\underline{\beta}(t) \times \underline{a} + \underline{u}(t) \times \underline{\omega})\, dt\right) \qquad (49)$$

with

$$\underline{\beta}(t) = \int_{t_m}^{t} \underline{\omega}\, dt$$

$$\underline{u}(t) = \int_{t_m}^{t} \underline{a}\, dt$$

$$\underline{\beta} = \underline{\beta}(t=t_{m+1})$$

$$\underline{u} = \underline{u}(t=t_{m+1})$$

Equations (48) and (49) are algorithmic forms of equation (44) that can be used to calculate $\underline{u}^N$ in the strapdown computer exactly (within the approximation of equation (47)). These equations show that the specific force velocity change in navigation coordinates is approximately equal to the integrated output from the strapdown accelerometer ($\underline{u}$) over the computer cycle, times the direction cosine matrix which was valid at the previous computer update time. Correction terms are applied to account for body rotation. In general, the correction term involves an integral of the interractive effects of angular $\underline{\omega}$ and linear $\underline{a}$ motion over the update cycle. The integral terms have been coined "sculling" effects.

The equation (49) form of the $\underline{u}^N$ equation includes a $1/2\ \underline{\beta} \times \underline{u}$ term which can be evaluated at t_{m+1} as the simple cross-product of integrated gyro and accelerometer measurements (i.e., without a dynamic integral operation). Furthermore, it is easily demonstrated that for approximately constant angular rates and accelerations over the computer cycle, the integral term in (49) is identically zero. This forms the basis for an approximate form of (49) which is valid under benign flight conditions (i.e., using equation (49) without including the integral term). The $1/2\ \underline{\beta} \times \underline{u}$ term in (49) is sometimes denoted as "rotation compensation".

4.1.1 Incremental Form of Transformation Operations and Sculling Terms

In a severe dynamic environment, equations (48) or (49) would be implemented explicitly with the integral terms mechanized as a high speed digital algorithmic operation within the t_m to t_{m+1} update cycle. The integral terms we are dealing with are from (48) and (49):

$$\underline{S}_1 \triangleq \int_{t_m}^{t_{m+1}} (\underline{\beta}(t) \times \underline{a})\, dt$$

$$\underline{S}_2 \triangleq 1/2 \int_{t_m}^{t_{m+1}} (\underline{\beta}(t) \times \underline{a} + \underline{u}(t) \times \underline{\omega})\, dt \qquad (50)$$

With the equation (50) definitions, (48) and (49) become:

$$\underline{u}^N = C(m)\ (\underline{u} + \underline{S}_1) \qquad (51)$$

or

$$\underline{u}^N = C(m)\ (\underline{u} + 1/2\ \underline{\beta} \times \underline{u} + \underline{S}_2) \qquad (52)$$

Recursive algorithms for $\underline{S}_1$ or $\underline{S}_2$ can be derived by first rewriting (50) in the equivalent form:

$$\underline{\beta}(t) = \underline{\beta}(\ell) + \int_{t_\ell}^{t} \underline{\omega}\, dt$$

$$\underline{u}(t) = \underline{u}(\ell) + \int_{t_\ell}^{t} \underline{a}\, dt$$

$$\underline{\gamma}_1(\ell+1) = \underline{\gamma}_1(\ell) + \int_{t_\ell}^{t_{\ell+1}} (\underline{\beta}(t) \times \underline{a})\, dt$$

$$\underline{\gamma}_2(\ell+1) = \underline{\gamma}_2(\ell) + 1/2 \int_{t_\ell}^{t_{\ell+1}} (\underline{\beta}(t) \times \underline{a} + \underline{u}(t) \times \underline{\omega})\, dt \qquad (53)$$

$$\underline{\beta}(\ell+1) = \beta(t=t_{\ell+1})$$

$$\underline{u}(\ell+1) = \underline{u}(t=t_{\ell+1})$$

$$\underline{S}_1 = \underline{\gamma}_1(t=t_{m+1})$$

$$\underline{S}_2 = \underline{\gamma}_2(t=t_{m+1})$$

with initial conditiona

$$\begin{aligned} \underline{\beta}(t=t_m) &= 0 \\ \underline{u}(t=t_m) &= 0 \\ \underline{\gamma}_1(t=t_m) &= 0 \\ \underline{\gamma}_2(t=t_m) &= 0 \end{aligned} \tag{54}$$

where

ℓ = High spaed computer cycle within m lowar speed computation cycle.

The intagrals in (53) can be replaced by analytical forms that are compatible with gyro and accelerometer input data proceasing if $\underline{\omega}$ and $\underline{a}$ are replaced by a generalized tima saries expansion. For equations (53), it is sufficient to approximate $\underline{\omega}$ and $\underline{a}$ over the ℓ to $\ell+1$ time interval as conatanta. Using this approximation in (53) yialda the final algorithm forms. For $\underline{S}_1$, the companion to equation (51), the algorithm ia:

$$\underline{\gamma}_1(\ell+1) = \underline{\gamma}_1(\ell) + \left(\underline{\beta}(\ell) + 1/2\ \underline{\Delta\theta}(\ell)\right) \times \underline{\Delta v}(\ell)$$

$$\underline{\beta}(\ell+1) = \underline{\beta}(\ell) + \underline{\Delta\theta}(\ell)$$

where

$$\underline{\Delta\theta}(\ell) = \int_{t_\ell}^{t_{\ell+1}} \underline{\omega}\ dt = \sum_{t_\ell}^{t_{\ell+1}} \underline{d\theta}$$

$$\underline{\Delta v}(\ell) = \int_{t_\ell}^{t_{\ell+1}} \underline{a}\ dt = \sum_{t_\ell}^{t_{\ell+1}} \underline{dv}$$

and

$$\underline{S}_1 = \underline{\gamma}_1(t=t_{m+1}) \tag{55}$$

For equation (51):

$$\underline{u}(\ell+1) = \underline{u}(\ell) + \Delta v(\ell)$$

$$\underline{u} \triangleq \underline{u}(t=t_{m+1})$$

with initial conditions:

$$\underline{\beta}(t=t_m) \triangleq \underline{\beta}(\ell=0) = 0$$

$$\underline{\gamma}_1(t=t_m) \triangleq \underline{\gamma}_1(\ell=0) = 0$$

where

$\underline{d\theta}$, $\underline{dv}$, = Gyro and accelerometer output pulse vectors. Each component (x, y, z) reprasents tha occurance of a rotation through a specifiad angle about the gyro input axis (for $\underline{d\theta}$ components) or an accelaration through a specific force velocity change along the acceleromater input axis (for $\underline{dv}$ components).

$\underline{\Delta\theta}$, $\underline{\Delta v}$, = Gyro and accelaromater pulse vector counts from ℓ to $\ell+1$.

For the alternative $\underline{S}_2$ form, the companion to equation (52), the algorithm is:

$$\underline{\gamma}_2(\ell+1) = \underline{\gamma}_2(\ell) + 1/2\,(\underline{\beta}(\ell) \times \underline{\Delta v}(\ell) + \underline{u}(\ell) \times \underline{\Delta\theta}(\ell))$$

$$\underline{\beta}(\ell+1) = \underline{\beta}(\ell) + \Delta\theta(\ell)$$

$$\underline{u}(\ell+1) = \underline{u}(\ell) + \underline{\Delta v}(\ell)$$

whsra

$$\underline{\Delta\theta}(\ell) = \int_{t_\ell}^{t_{\ell+1}} \underline{\omega}\, dt = \sum_{t_\ell}^{t_{\ell+1}} \underline{d\theta}$$

$$\underline{\Delta u}(\ell) = \int_{t_\ell}^{t_{\ell+1}} \underline{a}\, dt = \sum_{t_\ell}^{t_{\ell+1}} \underline{dv}$$

and (56)

$$\underline{S}_2 = \underline{\gamma}_2(t=t_{m+1})$$

For equations (52):

$$\underline{\beta} = \underline{\beta}(t=t_{m+1})$$

$$\underline{u} = \underline{u}(t=t_{m+1})$$

with initial conditions:

$$\underline{\beta}(t=t_m) \stackrel{\Delta}{=} \underline{\beta}(\ell=0) = 0$$

$$\underline{u}(t=t_m) \stackrel{\Delta}{=} \underline{u}(\ell=0) = 0$$

$$\underline{\gamma}_2(t=t_m) \stackrel{\Delta}{=} \underline{\gamma}_2(\ell=0) = 0$$

Equations (51) with (55), or (52) with (56) are computational algorithms that can be used to calculate tha navigation frama specific forca velocity changes. Two iteration rates ars implied: a basic m cycle rate, and a higher speed ℓ cycle rate within each m cycls.

Ths m cycle rata is selacted to ba high enough to protact the approximation of neglecting tha $(\underline{\beta}(t)\times)^2$ term in A(t) (contraat equation (47) with the equation (3) exact form for A). This design condition is typically avaluated under maximum expected linear acceleration/angular rats envalope conditions for tha particular application. Typically, the m cycla rete required for accuracy in tha attituda updating algorithms is alao sufficiant for accuracy requirements in the m cycle of the acceleration transformation algorithms.

Tha ℓ cycle rata within m is set high enough to proparly account for anticipated composite dynamic $\underline{\omega}$, $\underline{a}$ effacts. Section 6. dascribas analytical tachniques that can be used to assess tha edequacy of the $\underline{S}$ iteration rata for the sculling computation under dynamic input conditions.

4.1.3 Acceleration Transformation Algorithms Basad on Quaternion Attitude Dete

Equations (51) or (52) were based on the use of direction cosine data (C) in the strapdown computer. If the basic attitude data is calculatad in the form of e quaternion, the equivalent C matrix for transformation can ba calculated using equations (17). Alternatively, the quaternion data can be epplied directly in the implementation of the tranformation operation through application of equetion (12) to equations (51) and (52):

$$u^N = q(m)\,(u + S_1)\,q(m)^* \qquad (57)$$

or

$$u^N = q(m)\,(u + S_2')\,q(m)^*$$

(58)

$$\underline{S}_2' \stackrel{\Delta}{=} 1/2\,\underline{\beta} \times \underline{u} + \underline{S}_2$$

where u and the terms in the middle brackets are the quaternion form of the vector of the same nomenclature defined as having the first three terms (i.e., vector components) equal to the vector elements, and the fourth scalar term equal to zero. The $\underline{S}_1$ and $\underline{S}_2$ terms are calculated as defined by equations (55) and (56).

4.2 Acceleration Transformation Algorithm Correction For Navigation Frame Rotations

The acceleration transformation algorithms represented by equation (51), (52) or (57), (58) with (55), (56) neglects the effect of navigation frame rotation. In general, this is a minor correction term that can be easily accounted for at the n cycle update rate (i.e., the computer cycle rate used to update the attitude data for the effect of navigation frame rotations). It can be shown through a development similar to that leading to equation (52), that the correction algorithm for local navigation frame motion is given to first order by:

$$\Delta\underline{u}^N(n) = -\,1/2\ \underline{\theta} \times \underline{v}(n) \tag{59}$$

where

$\Delta\underline{u}^N(n)$ = Correction to the value of $\underline{u}^N$ computed in the m cycle that occurs at the current n cycle time. (Note: the m cycle is within the lower speed n cycle time frame).

$\underline{v}(n)$ = Summation of $\underline{u}(m)$ over the n cycle update period.

$\underline{\theta}$ = Integral of the navigation frame angular rotation rate over the n cycle period (as described in Sections 3.1.2 and 3.4)

5. EULER ANGLE EXTRACTION ALGORITHMS

If the body attitude relative to navigation axes is defined in terms of three successive Euler angle rotations ψ, θ, ϕ about axes z, y, x respectively (from navigation to body axes), it can be readily demonstrated (9) that the relationship between the direction cosine elements and Euler angles is given by:

$$
\begin{aligned}
C_{11} &= \cos\theta \cos\psi \\
C_{12} &= -\cos\phi \sin\psi + \sin\phi \sin\theta \cos\psi \\
C_{13} &= \sin\phi \sin\psi + \cos\phi \sin\theta \cos\psi \\
C_{21} &= \cos\theta \sin\psi \\
C_{22} &= \cos\phi \cos\psi + \sin\phi \sin\theta \sin\psi \\
C_{23} &= -\sin\phi \cos\psi + \cos\phi \sin\theta \sin\psi \\
C_{31} &= -\sin\theta \\
C_{32} &= \sin\phi \cos\theta \\
C_{33} &= \cos\phi \cos\theta
\end{aligned}
\tag{60}
$$

For conditions where $/\theta/ \neq \pi/2$ the inverse of equations (60) can be used to evaluate the Euler angles from the direction cosines:

$$
\begin{aligned}
\phi &= \tan^{-1} \frac{C_{32}}{C_{33}} \\
\theta &= -\tan^{-1} \frac{C_{31}}{\sqrt{(1-C_{31}{}^2)}} \\
\psi &= \tan^{-1} \frac{C_{21}}{C_{11}}
\end{aligned}
\tag{61}
$$

For situations where $/\theta/$ approaches $\pi/2$, the ϕ and ψ equations in (61) become indeterminate because the numerator and denominator approach zero simultaneously (see equations (60)). Under these conditions, an alternative equation for ϕ, ψ can be developed by first applying trigonometric algebra to equations (61) to obtain:

$$
\begin{aligned}
C_{23} + C_{12} &= (\sin\theta - 1)\sin(\psi + \phi) \\
C_{13} - C_{22} &= (\sin\theta - 1)\cos(\psi + \phi) \\
C_{23} - C_{12} &= (\sin\theta + 1)\sin(\psi - \phi) \\
C_{13} + C_{22} &= (\sin\theta + 1)\cos(\psi - \phi)
\end{aligned}
\qquad (62)
$$

Taking appropriate reciprocals of sine, cosine terms in (62) and applying the inverse tangent function:

For θ near $+\pi/2$

$$\psi - \phi = \tan^{-1} \frac{C_{23} - C_{12}}{C_{13} + C_{22}}$$

$$(63)$$

For θ near $-\pi/2$

$$\psi + \phi = \tan^{-1} \frac{C_{23} + C_{12}}{C_{13} - C_{22}}$$

Equations (63) can be used to obtain expressions for the sum or difference of ψ and ϕ under conditions where $/\theta/$ is near $\pi/2$. Explicit separate solutions for ψ and ϕ cannot be found under the $/\theta/ = \pi/2$ condition because ψ and ϕ both become angle measures about parallel axes (about vertical), hence, measure the same angle (i.e., a degree of rotational freedom is lost, and only two Euler angles, $\theta = \pm\,\pi/2$ and ψ or ϕ define the body to navigation frame attitude). Under $/\theta/$ near $\pi/2$ conditions, θ or ψ can be arbitrarily selected to satisfy another condition, with the unspecified variable calculated from (63). As an example, ψ might be set to a constant at the value it had from equations (61) when the $/\theta/$ near $\pi/2$ region was entered. This selection avoids jumps in ψ as the solution equation is transitioned from the (61) to the (63) form.

6. ALGORITHM PERFORMANCE ASSESSMENT

The division of the attitude updating and acceleration transformation algorithms into high and low speed loops for body motion effects (ℓ and m rates) provides for flexibility in selection of the iteration rates to maintain overall algorithm accuracy at system specified performance levels. The ℓ and m rate algorithms have been designed such that the high rate ℓ loop consists of simple computations that can be iterated at the high rate needed to properly account for high frequency vibration effects. The m rate loop algorithms, on the other are more complicated, based on computationally exact solutions.

Iteration rates for the m loop are selected to maintain accuracy under maximum maneuver induced motion conditions. The m loop iteration rate to maintain accuracy under maximum maneuver conditions can be easily evaluated analytically, or by simulation, through comparision of the actual algorithm solution with the Taylor series truncated forms selected for system mechanization. Iteration rates for the ℓ loop are selected to maintain accuracy under anticipated vibratory environmental conditions.

6.1 Vibration Environment Assessment

A fundamental calculation that should be performed prior to the analysis of ℓ loop algorithm iteration rate requirements is an assessment of the dynamic inputs that must be measured by the algorithms. In essence, this consists of an evaluation of the continuous (i.e., infinitely fast iteration rate) form of the algorithms in question under dynamic input conditions. The specific continuous form equations of interest are equations (22) for $\underline{\delta\beta}$ and (50) for $\underline{S}_1$ or $\underline{S}_2$.

6.1.1 $\underline{\delta\beta}$ Dynamic Environment Assessment (Coning)

We repeat equations (22) for $\underline{\delta\beta}$ evaluated at $t = t_{m+1}$:

$$\underline{\beta}(t) = \int_{t_m}^{t} \underline{\omega}\, dt$$

$$\delta\underline{\beta}(t=t_{m+1}) = 1/2 \int_{t_m}^{t_{m+1}} \underline{\beta}(t) \times \underline{\omega}\, dt \qquad (64)$$

end analyse the solution for $\delta\beta(t=t_{m+1})$ under general cyclic motion at frequency f in axes x and y with angular amplitudes θ_x, θ_y and relative phase angle ϕ such that:

$$\int_0^t \underline{\omega}\, dt = (\theta_x \sin(2\pi ft),\ \theta_y \sin(2\pi ft+\phi),\ 0)^T$$

$$\underline{\omega} = 2\pi f\,(\theta_x \cos(2\pi ft),\ \theta_y \cos(2\pi ft+\phi),\ 0)^T \qquad (65)$$

Substituting (65) in (64), expanding through application of appropriate trigonometric identities, and carrying out the indicated integrals analytically between the assigned limits, yields zero for the x, y components and the following for the z component of $\delta\underline{\beta}(t=t_{m+1})$:

$$\delta\beta_z(t=t_{m+1}) = \pi\,\theta_x\,\theta_y\,(\sin\phi)\,f\,\left((t_{m+1} - t_m) - \frac{\sin 2\pi f(t_{m+1} - t_m)}{2\pi f}\right)$$

Defining the m cycle time interval as T_m, the latter exression is equivalently:

$$\delta\beta_z = \pi\,\theta_x\,\theta_y\,(\sin\phi)\,f\,T_m\left(1 - \frac{\sin 2\pi fT_m}{2\pi fT_m}\right) \qquad (66)$$

Hence, even though the ω rate is cyclic in two axes as defined by equation (65) in x and y, the value for $\delta\beta_z$ is a constant proportional to the sine of the phase angle between the x, y angular vibrations. Under conditions where $\phi = 0$ (defined as "rocking" motion), $\delta\beta_z$ is zero. Under conditions where $\phi = \pi/2$, $\delta\beta_z$ is maximum. The equation (65) rate when $\phi = \pi/2$ has been termed "coning motion" due to the characteristic response of the z axis under this motion which describes a cone in inertial space.

Equation (66) can be put into a "drift rate" form by dividing the $\delta\beta_z$ angle by the time interval T_m over which it was evaluated:

$$\delta\dot{\beta}_z = \pi\,\theta_x\,\theta_y\,(\sin\phi)\,f\left(1 - \frac{\sin 2\pi fT_m}{2\pi fT_m}\right) \qquad (67)$$

Equation (67) is a fundamental equation that can be used to assess the magnitude of $\delta\dot{\beta}_z$ that must be accounted for by the $\delta\beta$ computer algorithm under discrete frequency input conditions. If $\delta\dot{\beta}_z$ is small relative to system performance requirements, it can be neglected, and the ℓ loop algorithm for $\delta\beta$ need not be implemented.

Equation (67) describes how $\delta\dot{\beta}_z$ can be calculated for a discrete input vibration frequency f. In a more general case, the input rate is composed of a mixture of frequencies in x and y at different phase angles ϕ for each. If the source of the generalized angular vibration is random input noise to the strapdown system, the x, y motion is colored by the transmission characteristics of the noise input to the x, y angular response. A more general development of equation (67) that accounts for the latter effects shows that the comparable equation for $\delta\dot{\beta}_z$ is given by:

$$\delta\dot{\beta}_z = \int_0^\infty \omega\, A_x(\omega)\, A_y(\omega)\, \sin\left(\phi_{Ay}(\omega) - \phi_{Ax}(\omega)\right)\left(1 - \frac{\sin\omega T_m}{\omega T_m}\right) P_{nn}(j\omega)\, d\omega \qquad (68)$$

where

$A_x(\omega)$, $A_y(\omega)$ = Amplitude of transfer function relating system input vibration noise to angular attitude response of sensor assembly about x, y axes.

$\phi_{Ax}(\omega)$, $\phi_{Ay}(\omega)$ = Phase of transfer function relating system input vibration noise to angular attitude response of sensor assembly about x, y axes.

$P_{nn}(j\omega)$ = Power spectral density of input vibration noise.

ω = Fourier frequency (rad/sec)

Note: Mean squared vibration energy = $\int_0^\infty P_{nn}(j\omega)\, d\omega$

Equation (68) can be used to assess the extent of random spectrum dynamic angular environment to be measured by the $\delta\beta$ computational algorithm. The $\delta\beta_z$ value calculated by (68) measures the composite correlated coning drift in the sensor assembly that must be calcuated to accurately account for the actual motion present. If the $\delta\beta_z$ magnitude calculated from (68) is small compared to other systems error budget effects, the mechanization of an algorithm to calculate $\delta\beta$ is not needed (i.e., can be approximated by zero).

The extension of equations (67) and (68) to y, z or z, x axis angular vibration motion should be obvious.

6.1.2 $\underline{S}_1$, $\underline{S}_2$ Dynamic Environment Assessment (Sculling)

We repeat equations (50) with $\underline{u}$ and $\underline{\beta}$ from (48) and (49):

$$\underline{\beta}(t) = \int_{t_m}^{t} \underline{\omega}\, dt$$

$$\underline{u}(t) = \int_{t_m}^{t} \underline{a}\, dt$$

$$\underline{S}_1 = \int_{t_m}^{t_{m+1}} (\underline{\beta}(t) \times \underline{a})dt \tag{69}$$

$$\underline{S}_2 = 1/2 \int_{t_m}^{t_{m+1}} (\underline{\beta}(t) \times \underline{a} + \underline{u}(t) \times \underline{\omega})dt$$

and analyse the $\underline{S}_1$, $\underline{S}_2$ solutions under general cycle motion at frequency f in axes x, y with angular amplitude θ_x about axis x and acceleration amplitude D_y along axis y at relative phase ϕ such that:

$$\int_0^t \underline{\omega}\, dt = (\theta_x \sin(2\pi ft),\ 0,\ 0)^T$$

$$\underline{\omega} = (2\pi f\ \theta_x \cos(2\pi ft),\ 0,\ 0)^T \tag{70}$$

$$\underline{a} = (0,\ D_y \sin(2\pi ft+\phi),\ 0)^T$$

Substituting (70) in (69), expanding through application of appropriate trigonometric identities, and carrying out the indicated integrals analytically between the assigned limits, yields zero for the x, y components and the following for the z component of $\underline{S}_1$ and $\underline{S}_2$:

$$S_{2z} = 1/2\ T_m\ \theta_x\ D_y\ (\cos\phi)\ \left(1 - \frac{\sin \pi f T_m}{2\pi f T_m}\right) \tag{71}$$

$$S_{1z} = 1/2\ (\underline{\beta} \times \underline{u})_z + S_{2z} \tag{72}$$

where

$(\underline{\beta} \times \underline{u})_z$ = z - component of $\underline{\beta} \times \underline{u}$ evalulated at $t = t_{m+1}$.

Hence, even though the $\underline{\omega}$ and $\underline{a}$ inputs are cyclic in two axes as defined in equations (70), the value for S_{2z} is a constant proportional to the cosine of the phase angle between the x angular vibration and y linear acceleration vibration. Under conditions where $\phi = \pi/2$, $\dot{S}_{2z}$ is zero. Under conditions where $\phi = 0$, $\dot{S}_{2z}$ is a maximum. Equation (70) motion when $\phi = 0$ has been termed "sculling motion" due to the analogy with the characteristic angular movement and acceleration forces imparted to an oar used to propel a boat from the stern. Note also that S_{1x} is equal to S_{2x} plus the correction term (rotation compensation) measured as the cross-product of the simple angular rate and linear acceleration integrals taken over the m computation cycle. (See equations (48) and (49) for definitions).

Equation (71) for S_{2x} can be put into an "acceleration bias" form by dividing the velocity change correction S_{2z} by the time interval T_m over which it was evaluated:

$$\dot{S}_{2x} = 1/2\ \theta_x\ D_y\ (\cos\phi)\left(1 - \frac{\sin 2\pi f T_m}{2\pi f T_m}\right) \tag{73}$$

Equation (73) (with (72) for $\dot{S}_{1z}$) is a fundamental equation that can be used to assess the magnitude of $\dot{S}_{2z}$ that must be accounted for by the S_1 or S_2 computer algorithm under discrete frequency input conditions. If $\dot{S}_{2x}$ is small relative to system performance requirements, it can be neglected, and the ℓ loop algorithm for calculating S_1 or S_2 need not be implemented. Under the latter conditions, S_1 would be set equal to the cross-product term in (72) which makes the basic equation (51) and (52) transformation algorithms identical.

Equation (73) describes how $\dot{S}_{2x}$ can be calculated with a discrete input vibration frequency f for angular motion about x and linear motion along y. In a more general case, the input rates and accelerations are composed of mixtures of angular and linear motion about x and y at different frequencies and relative phase angles. If the source of the generalized vibration motion is random input noise to the strapdown system, the x, y angular and linear motion is colored by the transmission characteristics of the noise input to the x, y angular and linear response. A more general development of equation (73) that accounts for the latter effects show that the comparable equation for $\dot{S}_{2z}$ is given by:

$$\dot{S}_{2z} = \int_0^\infty \Big(A_y(\omega)\, B_x(\omega) \cos(\phi_{Ay}(\omega) - \phi_{Bx}(\omega)) - A_x(\omega)\, B_y(\omega) \cos(\phi_{Ax}(\omega) - \phi_{By}(\omega))\Big) \left(1 - \frac{\sin\omega T_m}{\omega T_m}\right) P_{nn}(j\omega)\, d\omega \tag{74}$$

where

$A_x(\omega)$, $A_y(\omega)$, $\phi_{Ax}(\omega)$, $\phi_{Ay}(\omega)$, $P_{nn}(j\omega)$, ω = As defined previously.

$B_x(\omega)$, $B_y(\omega)$, $\phi_{Bx}(\omega)$, $\phi_{By}(\omega)$ = x, y, amplitude/phase linear acceleration response of the sensor assembly to the input vibration.

Equation (74) can be used to assess the extent of random spectrum dynamic motion environment to be measured by the S_1 or S_2 computational algorithms. The $\dot{S}_{2z}$ value calculated by (74) measures the composite correlated sculling acceleration bias in the sensor assembly that must be calculated to accurately account for the actual motion present. If the $\dot{S}_{2z}$ magnitude calculated from (74) is small compared to other system error budget effects, the mechanization of an algorithm to calculate S_1 or S_2 in the high rate ℓ loop is not needed (i.e., S_2 can be approximated by zero in (52) or S_1 can be set equal to the cross-product term in (52)).

The extension of equations (73) and (74) for y, z or z, x axis vibration motion should be obvious.

6.2 Algorithm Accuracy Assessment

The accuracy of the computation algorithm for $\underline{\delta\beta}$ or $\underline{S_1}$, $\underline{S_2}$ can be assessed by comparing their solutions to the comparable continuous form solutions developed in Section 6.1 under identical input conditions.

6.2.1 $\underline{\delta\beta}$ Coning Algorithm Error Assessment

The computational algorithm for calculating $\underline{\delta\beta}$ in a strapdown system is given by equation (26). A measure of the accuracy of the equation (26) algorithm can be obtained by analytically calculating the solution generated from (26) under assumed cyclic motion and comparing this result to the equivalent solution obtained from the idealized continuous algorithm described in Section 6.1. For a discrete frequency vibration input, the equation (65) motion can be used analytically in equation (26) to calculate the algorithm solution for $\underline{\delta\beta}$ at $t = t_{m+1}$ (i.e., analagous to the equation (67) solution for the continuous (infinitely fast) algorithm. After much algebraic manipulation, it can be demonstrated that the algorithm solution for $\underline{\delta\beta}$ as calculated from equation (26) under equation (65) input motion, has zero x, y components, with a z component rate given by:

$$\dot{\delta\beta}_{zALG} = \pi\, \theta_x\, \theta_y\, (\sin\phi) \left((1 + 1/3\,(1 - \cos 2\pi f T_\ell)) \frac{\sin 2\pi f T_\ell}{2\pi f T_\ell} - \frac{\sin 2\pi f T_m}{2\pi f T_m} \right) \tag{75}$$

where

$\dot{\delta\beta}_{zALG}$ = Recursive algorithm solution for $\delta\beta_z$ rate

T_ℓ = Time interval for high speed ℓ computer iteration cycle

Equation (75) for the $\underline{\delta\beta}$ discrete recursive algorithm solution of equation (26) is directly analagous to the equation (67) solution of the equation (22) continuous $\underline{\delta\beta}$ algorithm. It is easily verified that (75) reduces to (67) as T_ℓ approaches zero.

The error in the $\underline{\delta\beta}$ algorithm is measured by the difference between (67) and (75); i.e.:

$$e(\dot{\delta\beta}_z) = \pi\, f\, \theta_x\, \theta_y\, (\sin\phi)\left((1 + 1/3\,(1 - \cos 2\pi f T_\ell)) \frac{\sin 2\pi f T_\ell}{2\pi f T_\ell} - 1 \right) \tag{76}$$

where

$e(\dot{\delta\beta}_z)$ = Error rate in the equation (26) algorithm.

Equation (76) can be used to assess the error in the equation (26) $\underline{\delta\beta}$ algorithm caused by finite iteration rate (i.e., the effect of T_ℓ) under discrete frequency input conditions.

Under random vibration input conditions, the equation (26) algorithm can be analyzed to obtain the more general solution for the $\dot{\delta\beta}_{zALG}$ rate:

$$\dot{\delta\beta}_{zALG} = \int_0^\infty \omega\, A_x(\omega)\, A_y(\omega)\, \sin(\phi_{Ay}(\omega) - \phi_{Ax}(\omega)) \left((1 + 1/3\,(1 - \cos\omega T_\ell)) \frac{\sin \omega T_\ell}{\omega T_\ell} - \frac{\sin \omega T_m}{\omega T_m} \right) P_{nn}(j\omega)\, d\omega \tag{77}$$

The $\delta\beta$ algorithm error under random inputs is the difference between the equation (77) discrete solution and the equivalent continuous equation (68) solution form. The result is:

$$e(\dot{\delta\beta}_z) = \int_0^\infty \omega\, A_x(\omega)\, A_y(\omega)\, \sin(\phi_{Ay}(\omega) - \phi_{Ax}(\omega)) \left((1 + 1/3\,(1 - \cos\omega T_\ell)) \frac{\sin \omega T_\ell}{\omega T_\ell} - 1 \right) P_{nn}(j\omega)\, d\omega \tag{78}$$

Equations (76) and (78) can be used to assess the error in the equation (26) $\underline{\delta\beta}$ algorithm caused by finite iteration rate under discrete or random vibration input conditions. The extension of equations (76) and (78) to y, z or z, x axis effects should be obvious.

6.2.2 $\underline{S}$ Sculling Algorithm Error Assessment

The computational algorithm for calculating $\underline{S}_1$ or $\underline{S}_2$ is given by equations (55) and (56). A measure of the accuracy of these algorithms can be obtained by analytically calculating the solution generated from (55) or (56) under assumed cyclic motion and comparing the result to the equivalent solution obtained from the continuous algorithm as described in Section 6.1.2. For a discrete frequency vibration input, the equation (70) motion can be used analytically in equation (55) and (56) to calculate the algorithm solution for $\underline{S}_1$, $\underline{S}_2$ (i.e., analogous to the equation (72) and (73) solution for the continuous (infinitely fast) algorithm). After much algebraic manipulation, it can be demonstrated that the algorithm solution for $\underline{S}_1$ and $\underline{S}_2$ as calculated from equations (55) and (56) under equation (70) input motion, has zero x, y components, with a z component rate given by:

$$\dot{S}_{2zALG} = 1/2\ \theta_z\ D_y\ (\cos\phi)\ \left(\frac{\sin 2\pi f T_\ell}{2\pi f T_\ell} - \frac{\sin 2\pi f T_m}{2\pi f T_m} \right) \tag{79}$$

$$S_{1zALG} = 1/2\ (\underline{\beta} \times \underline{u})_z + S_{2zALG} \tag{80}$$

where

S_{1zALG}, S_{2zALG} = Recursive algorithm solutions for S_{1z}, S_{2z}.

Equations (79) and (80) for the $\underline{S}_1$, $\underline{S}_2$ discrete recursive algorithm solution is directly analogous to the equations (73) and (72) solution to the continuous $\underline{S}_1$, $\underline{S}_2$ algorithm. It is easily verified that (79) and (80) reduce to (73) and (72) as T_ℓ approaches zero.

The error in the $\underline{S}_1$, $\underline{S}_2$ algorithm is measured by the difference between (79), (80) and (73), (72) ; i.e.,

$$e(\dot{S}_{1z}) = e(\dot{S}_{2z}) = 1/2\ \theta_z\ D_y\ (\cos\phi)\ \left(1 - \frac{\sin 2\pi f T_\ell}{2\pi f T_\ell}\right) \tag{81}$$

where

$e(\dot{S}_{1z})$, $e(\dot{S}_{2z})$ = Error rate in the equation (55) and (56) algorithm solutions.

Equation (81) can be used to assess the error in the equation (55) and (56) algorithms caused by finite iteration rate (i.e., the effect of T_ℓ) under discrete frequency input conditions.

Under random vibration input conditions, the equation (55) and (56) algorithms can be analyzed to obtain the more general solution for S_{1z}, S_{2z}:

$$\begin{aligned} \dot{S}_{2z} = \int_0^\infty & \Big(A_y(\omega)\ B_x(\omega)\cos(\phi_{Ay}(\omega) - \phi_{Bx}(\omega)) \\ & - A_x(\omega)\ B_y(\omega)\cos(\phi_{Ax}(\omega) - \phi_{By}(\omega))\Big) \left(\frac{\sin \omega T_\ell}{\omega T_\ell} - \frac{\sin \omega T_m}{\omega T_m}\right) P_{nn}(j\omega)\ d\omega \end{aligned} \tag{82}$$

$$S_{1z} = 1/2\ (\underline{\beta} \times \underline{u})_z + S_{2z}$$

The S_{1z}, S_{2z} algorithm error under vibration is the difference between the equation (82) discrete solutions and the equivalent continuous equation (74) with (72) forms:

$$\begin{aligned} e(\dot{S}_{1z}) = e(\dot{S}_{2z}) = \int_0^\infty & \Big(A_y(\omega)\ B_x(\omega)\cos(\phi_{Ay}(\omega) - \phi_{Bx}(\omega)) \\ & - A_x(\omega)\ B_y(\omega)\cos(\phi_{Ax}(\omega) - \phi_{By}(\omega))\Big) \left(1 - \frac{\sin \omega T_\ell}{\omega T_\ell}\right) P_{nn}(j\omega)\ d\omega \end{aligned} \tag{83}$$

Equation (82) and (83) can be used to assess the error in the equation (55) and (56) algorithms caused by finite iteration rate under discrete or random vibration input conditions. The extension of equation (83) to y, z or z, x axis effects should be obvious.

7. CONCLUDING REMARKS

The strapdown computational algorithms and associated design considerations presented in this paper are representative of the algorithms being used in most modern-day strapdown inartial navigation aystems. The unique characteristic of the attituda and transformation algorithms pressnted is the separation of each into a complex low speed and simple high speed computation section. Due to the simplicity of the high speed calculations they can be executed at the high rates necesrary to properly account for high frequecy but generally low amplituda vibratory effects without posing an insurmountable throughput burdan on the computar. The lower speed calculations which contain the bulk of the computational equations can than be executed at a fairly modest update rate selected to properly account for lower frequency but larger magnitude maneuver induced motion effects. Perhaps the principal advantage of the algorithm forms presented, is their ability to be analyzed for accuracy using straight-forward analytical techniques. This allows the algorithms to be easily tailored and evaluated for given applications as a function of anticipated dynamic environments and user accuracy requirements.

REFERENCES

1. Pitman, George R. Jr., ed., Inertial Guidance, John Wiley and Sons, New York, 1962.

2. Leondes, Cornelius T., ed., Guidance and Control of Aeroapace Vehicles, McGraw-Hill, 1963.

3. Macomber, George R. and Fernandes, Manuel, Inertial Guidance Engineering, Prentice-Hall Englewood Cliffs, New Jersey, 1962.

4. Britting, Kenneth R., Inertial Navigation System Analysis, John Wiley and Sons, New York, 1971.

5. Morse, Philip M. and Feshback, Herman, Methods of Theoretical Physics, McGraw-Hill, 1953.

6. A Study of Critical Computational Problems Associated with Strapdown Inertial Navigation Systems, NASA Report CR-968, April 1968.

7. Savage, P.G., "A New Second-Order Solution for Strapped-Down Attitude Computation", AIAA/JACC Guidance & Control Conference, Seattle, Washington, August 15 - 17, 1966.

8. Jordan, J.W., "An Accurate Strapdown Direction Cosine Algorithm", NASA TN D-5384, September 1969.

9. Mckern, Richard A., A Study of Transformation Algorithms For Use In A Digital Computer, Massachusetts Institute of Technology Master's Thesis, Department of Aeronautics and Astronatics, January 1968.

10. Bortz J.E., "A New Mathematical Formulation for Strapdown Inertial Navigation, "IEEE Transactions on Aerospace and Electronic Systems, Volume AES-7, No. 1, January 1971.

11. Shepperd, Stanley W., "Quaternion from Rotation Matrix", AIAA Journal of Guidance and Control, Vol. 1, No. 3, May-June 1978.

12. Savage, P.G., Introduction To Strapdown Inertial Navigation Systems, June 1, 1983 (Third Printing); Third Strapdown Associates Open Seminar On Strapdown Inertial Navigation Systems, Marquette Inn, Minneapolis, Minnesota, November 14 - 18, 1983.

APPENDIX A

DERIVATION OF $\dot{\underline{\phi}}$ EQUATION

A differential equation for the rate of change of the $\underline{\phi}$ vector can be derived from the equivalent quaternion rate equation. The quaternion h in equations (13) and (14) is the quaternion equivalent to the $\underline{\phi}$ rotation angle vector. A differential equation for h can be derived from the incremental equivalent to (13) that describes how h changes over a short time period Δt (from t_ℓ to $t_{\ell+1}$) within the larger time interval from t_m to t_{m+1}:

$$h(\ell+1) = h(\ell)\, p(\ell) \tag{A1}$$

where

$$p = \begin{vmatrix} g_3 \alpha_x \\ g_3 \alpha_y \\ g_3 \alpha_z \\ g_4 \end{vmatrix}$$

$$g_3 = \frac{\sin(\alpha/2)}{\alpha} \qquad g_4 = \cos(\alpha/2) \tag{A2}$$

$\underline{\alpha}$ = Rotation angle vector associated with the small rotation of the body over the short computer time interval from ℓ to $\ell+1$ within the larger interval from m to m+1.

$\alpha_x, \alpha_y, \alpha_z, \alpha$ = Components and magnitude of $\underline{\alpha}$.

Equation (A1) is equivalently:

$$\frac{h(\ell+1) - h(\ell)}{\Delta t} = h(\ell)\,\frac{p(\ell)-1}{\Delta t} \tag{A3}$$

$$\Delta t \stackrel{\Delta}{=} t_{\ell+1} - t_\ell$$

The basic definition of angular rate states that for small Δt,

$$\underline{\alpha} \approx \underline{\omega}\,\Delta t$$
$$\alpha \approx \omega\,\Delta t \tag{A4}$$

Hence, for small Δt, $\underline{\alpha}$ is small, and therefore, from (A2),

$$g_3 \approx 1/2$$
$$g_4 \approx 1 - \frac{\alpha^2}{2} \approx 1 - \frac{\omega^2 \Delta t^2}{2} \tag{A5}$$

Using mixed vector/scalar notation, substitution of (A4) and (A5) in (A2) yields:

$$p = g_3\,\underline{\alpha} + g_4$$
$$\approx 1/2\,\underline{\omega}\,\Delta t + 1 - \frac{\omega^2 \Delta t^2}{2}$$

Substituting in (A3) obtains:

$$\frac{h(\ell+1) - h(\ell)}{\Delta t} \approx h(\ell)\left(1/2\,\underline{\omega} + 1/2\,\omega^2 \Delta t\right)$$

In the limit as $\Delta t \to 0$, the latter reduce to the derivative form:

$$\dot{h} = 1/2\,h\,\underline{\omega} \tag{A6}$$

We now return to (14) and express h as a function of $\underline{\phi}$ in mixed vector/scaler notation:

$$h = f_3\,\underline{\phi} + f_4$$
$$f_3 = \frac{\sin(\phi/2)}{\phi} \tag{A7}$$
$$f_4 = \cos(\phi/2)$$

Substituting in (A6),

$$\dot{h} = 1/2\,f_3\,\underline{\phi}\,\underline{\omega} + 1/2\,f_4\,\underline{\omega} \tag{A8}$$

It is readily demonstrated by algebraic expansion and using the rules of quaternion algebra that $\underline{\phi}\,\underline{\omega}$ in (A8) is equivalently:

$$\underline{\phi}\,\underline{\omega} = \underline{\phi} \times \underline{\omega} - \underline{\phi} \cdot \underline{\omega}$$

Differentiation of (A7) shows that:

$$\dot{h} = \dot{f}_3\,\underline{\phi} + f_3\,\dot{\underline{\phi}} + \dot{f}_4$$

$$\dot{f}_3 = 1/2\ \frac{\cos\phi/2}{\phi}\,\dot{\phi} - \frac{\sin\phi/2}{\phi^2}\,\dot{\phi}$$

$$= \frac{\dot{\phi}}{\phi}\,(1/2\ f_4 - f_3)$$

$$\dot{f}_4 = -\,1/2\,(\sin\phi/2)\,\dot{\phi} = -\,1/2\,\phi\,\dot{\phi}\,f_3$$

Hence, with (A8),

$$\dot{h} = f_3\,\dot{\underline{\phi}} + \frac{\dot{\phi}}{\phi}\,(1/2\ f_4 - f_3)\,\underline{\phi} - 1/2\,\phi\,\dot{\phi}\,f_3$$

$$= 1/2\ f_3\,(\underline{\phi} \times \underline{\omega}) - 1/2\ f_3\,\underline{\phi} \cdot \underline{\omega} + 1/2\ f_4\,\omega$$

Dividing by f_3 and solving for $\dot{\underline{\phi}}$:

$$\dot{\underline{\phi}} = 1/2\ \frac{f_4}{f_3}\,\underline{\omega} + 1/2\,(\underline{\phi} \times \underline{\omega}) - \frac{\dot{\phi}}{\phi}\,(1/2\ \frac{f_4}{f_3} - 1)\,\underline{\phi} + 1/2\,\phi\,\dot{\phi} - 1/2\,\underline{\phi} \cdot \underline{\omega} \qquad \text{(A9)}$$

Equation (A9) is now separated into its vector and scalar components:

$$\dot{\underline{\phi}} = 1/2\ \frac{f_4}{f_3}\,\underline{\omega} + 1/2\,(\underline{\phi} \times \underline{\omega}) - \frac{\dot{\phi}}{\phi}\,(1/2\ \frac{f_4}{f_3} - 1)\,\underline{\phi} \qquad \text{(A10)}$$

$$1/2\,\phi\,\dot{\phi} = 1/2\,\underline{\phi} \cdot \underline{\omega}$$

The scalar equation is equivalently:

$$\frac{\dot{\phi}}{\phi} = \frac{1}{\phi^2}\,\underline{\phi} \cdot \underline{\omega}$$

Substituting in the vector part of (A10) yields:

$$\dot{\underline{\phi}} = 1/2\ \frac{f_4}{f_3}\,\omega + 1/2\,(\underline{\phi} \times \underline{\omega}) - \frac{1}{\phi^2}\,(1/2\ \frac{f_4}{f_3} - 1)\,(\underline{\phi} \cdot \underline{\omega})\,\underline{\phi}$$

Using the vector triple product rule, it is easily demonstrated that:

$$(\underline{\phi} \cdot \underline{\omega})\,\underline{\phi} = \underline{\phi} \times (\underline{\phi} \times \underline{\omega}) + \phi^2\,\underline{\omega}$$

Substituting,

$$\dot{\underline{\phi}} = 1/2\ \frac{f_4}{f_3}\,\underline{\omega} + 1/2\,\underline{\phi} \times \underline{\omega} - (1/2\ \frac{f_4}{f_3} - 1)\,\underline{\omega} + \frac{1}{\phi^2}\,(1 - \frac{f_4}{2f_3})\,\underline{\phi} \times (\underline{\phi} \times \underline{\omega})$$

Combining terms:

$$\dot{\underline{\phi}} = \underline{\omega} + 1/2\,\underline{\phi} \times \underline{\omega} + \frac{1}{\phi^2}\,(1 - \frac{f_4}{2f_3})\,\underline{\phi} \times (\underline{\phi} \times \underline{\omega})$$

Using the definition for f_4 and f_3 from (A7), it can be shown by trigonometric manipulation that the bracketed coefficient in the latter expression is equivalently:

$$1 - \frac{f_4}{2f_3} = \frac{1}{\phi^2}\,(1 - \frac{\phi\,\sin\phi}{2(1-\cos\phi)})$$

Substitution yields the final expression for $\dot{\underline{\phi}}$:

$$\dot{\underline{\phi}} = \underline{\omega} + 1/2\ \underline{\phi} \times \underline{\omega} + \frac{1}{\phi^2}\left(1 - \frac{\phi \sin \phi}{2(1-\cos\phi)}\right) \underline{\phi} \times (\underline{\phi} \times \underline{\omega}) \qquad \text{(A11)}$$

Equation (20) in the main text is the integral from of (A11) over a computer cycle (from t_m to t_{m+1}).