Chapter 11

Strapdown navigation system computation

11.1 Introduction

The analytical equations which must be solved in order to extract attitude, velocity and position information from the inertial measurements provided by the gyroscopes and accelerometers in a strapdown system have been described in Chapter 3. This chapter is concerned with the real time implementation of these equations in a computer. The main computing tasks, those of attitude determination, specific force resolution and solution of the navigation equation, are indicated in the block diagram shown in Figure 11.1.

The most demanding of the processing tasks, in terms of computer loading, are the attitude computation and the specific force vector resolution. In the presence of high frequency motion, the implementation of these tasks in real time creates a substantial computing burden for the strapdown system computer, even with modern processors. The navigation processing task, which is common to all types of inertial navigation system, both strapdown and platform mechanisations, is less demanding computationally. In addition to these tasks, the determination of attitude in terms of Euler rotations is required in some applications.

Early attempts to produce a strapdown inertial navigation system were limited, in part, by the computer technologies available at the time. Apart from the physical size of early computers which delayed the development of strapdown systems, particularly for airborne applications, the lack of computing speed, or throughput, was a major obstacle to the achievement of fast and accurate attitude computation. As a result, the performance which could be achieved in early strapdown systems was limited, particularly under high frequency motion conditions.

Such difficulties prompted much effort to be directed towards the development of efficient computing algorithms, and in particular, the splitting of the strapdown computing processes into low- and high-speed segments. The low-speed calculations are designed to take account of low frequency, large amplitude, body motions arising from vehicle manoeuvres, whilst the high-speed section involves a relatively simple
algorithm which is designed to keep track of the high frequency, low amplitude, motions of the vehicle. Contemporary algorithms often adopt this approach for both the attitude computation and the specific force resolution.

11.2 Attitude computation

As indicated previously, it is the computation of attitude which is particularly critical in a strapdown system. It should therefore come as no surprise to find that it is this topic which has been the subject of much study [1–5]. In many applications, the dynamic range of the angular motions to be taken account of can be very large, varying from a few degrees per hour to 2000° per second or more. In addition, the system may be subjected to high frequency dynamic motion in some applications. For example, a strapdown system in a guided missile may be subjected to such motion as a result of body bending and rocket motor induced vibration. The ability of the strapdown algorithm to keep track of body attitude accurately in a severe vibratory environment may well be the critical factor in determining its performance, if accurate navigation is to be achieved.

The conventional approach to attitude determination is to compute the direction cosine matrix, relating the vehicle body reference frame to the reference co-ordinate system, or its quaternion equivalent, using a numerical integration scheme. It is theoretically possible to calculate body attitude sufficiently accurately, even in the presence of high frequency angular motion, provided that the computational frequency is sufficiently high. However, in practice this may impose an intolerable burden on the processor.

An alternative formulation advocated by Bortz [1] involves the representation of changes in attitude as a rotation vector. As described below, this approach allows
the attitude computation to be split conveniently into low- and high-speed sections, denoted here as the $k$-cycle and $j$-cycle rates. The lower-speed part of the calculation is designed to take account of the relatively low frequency, large amplitude, body motion arising as a result of vehicle manoeuvres. The high-speed section involves a simple algorithm, which is designed to track the high frequency, low amplitude, motions of the vehicle.

Using this approach, coning motion at frequencies near to the lower computational rate may be accounted for, without the need to increase the speed at which the bulk of the computation is implemented.

11.2.1 Direction cosine algorithms

In order to update the direction cosine matrix, $C$, defined in Chapter 3, it is required to solve a matrix differential equation of the form:

$$\dot{C} = C\Omega$$  \hspace{1cm} (11.1)

where $\Omega$ is a skew symmetric matrix formed from the elements of the turn rate vector $\omega$. For clarity, the subscripts and superscripts used earlier have been omitted in the following development.

Over a single computer cycle, from time $t_k$ to $t_{k+1}$, the solution of the equation may be written as follows:

$$C_{k+1} = C_k \exp \int_{t_k}^{t_{k+1}} \Omega \, dt$$  \hspace{1cm} (11.2)

Provided that the orientation of the turn rate vector, $\omega$, remains fixed in space over the update interval, we may write:

$$\int_{t_k}^{t_{k+1}} \Omega \, dt = [\omega \times]$$  \hspace{1cm} (11.3)

Hence, eqn. (11.2) becomes:

$$C_{k+1} = C_k \exp[\sigma \times] = C_k A_k$$  \hspace{1cm} (11.4)

where $C_k$ represents the direction cosine matrix which relates body to reference axes at the $k$th computer cycle, and $A_k$ the direction cosine matrix which transforms a vector from body coordinates at the $k+1$th computer cycle to body coordinates at the $k$th computer cycle. The variable $\sigma$ is an angle vector with direction and magnitude such that a rotation of the body frame about $\sigma$, through an angle equal to the magnitude of $\sigma$, will rotate the body frame from its orientation at computer cycle $k$ to its position at computer cycle $k+1$. The components of $\sigma$ are denoted by $\sigma_x, \sigma_y$ and $\sigma_z$ and its magnitude is given by:

$$\sigma = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}$$  \hspace{1cm} (11.5)
and
\[
\sigma \times = \begin{bmatrix}
0 & -\sigma_z & \sigma_y \\
\sigma_z & 0 & -\sigma_x \\
-\sigma_y & \sigma_x & 0
\end{bmatrix}
\] (11.6)

Expanding the exponential term in eqn. (11.4) gives:
\[
A_k = I + [\sigma \times] + \frac{[\sigma \times]^2}{2!} + \frac{[\sigma \times]^3}{3!} + \frac{[\sigma \times]^4}{4!} + \cdots
\] (11.7)

and using eqn. (11.6) it can be shown that:
\[
[\sigma \times]^2 = \begin{bmatrix}
-\sigma_y^2 + \sigma_z^2 & \sigma_x \sigma_y & \sigma_x \sigma_z \\
\sigma_x \sigma_y & -\sigma_x^2 + \sigma_z^2 & \sigma_y \sigma_z \\
\sigma_x \sigma_z & \sigma_y \sigma_z & -\sigma_x^2 + \sigma_y^2
\end{bmatrix}
\]
\[
[\sigma \times]^3 = -(\sigma_x^2 + \sigma_y^2 + \sigma_z^2) [\sigma \times]
\]
\[
[\sigma \times]^4 = -(\sigma_x^2 + \sigma_y^2 + \sigma_z^2) [\sigma \times]^2
\]

Thus, we may write:
\[
A_k = I + [\sigma \times] + \frac{[\sigma \times]^2}{2!} - \frac{\sigma_x^2 [\sigma \times]^2}{3!} - \frac{\sigma_y^2 [\sigma \times]^2}{4!} + \cdots
\]
\[
= I + \left[ 1 - \frac{\sigma_x^2}{3!} + \frac{\sigma_y^4}{5!} - \cdots \right] [\sigma \times] + \left[ \frac{1}{2!} - \frac{\sigma_x^2}{4!} + \frac{\sigma_y^4}{6!} - \cdots \right] [\sigma \times]^2
\]

which may be written as follows:
\[
A_k = I + \frac{\sin \sigma}{\sigma} [\sigma \times] + \frac{(1 - \cos \sigma)}{\sigma^2} [\sigma \times]^2
\] (11.10)

Provided that \(\sigma\) is the angle vector as defined above, eqn. (11.10) provides an exact representation of the attitude matrix which relates body attitude at times \(t_{k+1}\) and \(t_k\). If it was possible to implement this equation perfectly, it would yield an orthogonal matrix which need only be evaluated when transformation of the measured specific force vector is required. In practice of course, it is necessary to truncate the mathematical functions in eqn. (11.10), in order to produce an algorithm which can be implemented in real time. Following eqn. (11.9), \(A_k\) may be calculated using:
\[
A_k = I + a_1 [\sigma \times] + a_2 [\sigma \times]^2
\] (11.11)

where
\[
a_1 = 1 - \frac{\sigma_x^2}{3!} + \frac{\sigma_y^4}{5!} - \cdots
\]
and
\[ a_2 = \frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!} - \cdots \]

The direction cosine matrix may therefore be updated for body motion, as sensed by the strapdown gyroscopes, using a recursive algorithm based on eqns. 11.4 and 11.9. The order of the algorithm will be determined by the number of terms included in eqn. (11.9). For example, both of the infinite series would be truncated at the \( \sigma^2 \) level in a fourth order algorithm. The computation rate should be selected to ensure that the magnitude of \( \sigma \) remains small at the maximum turn rate, thus avoiding the need to include a large number of terms in the expression for \( A_k \).

11.2.1.1 The definition of attitude computation errors

The computed attitude matrix, written here as \( \hat{A} \), may be expressed in terms of the true attitude matrix, \( A \), and an error matrix \( E \) as follows:

\[
\hat{A} = A[I + E]
\]

Rearranging, we have:

\[
E = A^T \hat{A} - I
\]

Substituting for \( A \) and \( \hat{A} \) following eqns. 11.10 and 11.11, we may write:

\[
E = \left[ I - \frac{\sin \sigma}{\sigma} [\sigma \times] + \frac{(1 - \cos \sigma)}{\sigma^2} [\sigma \times]^2 \right] \left[ I + a_1[\sigma \times] + a_2[\sigma \times]^2 \right] - I
\]

\[
= (\sigma a_1 \cos \sigma - \sin \sigma + \sigma^2 a_2 \sin \sigma) \frac{[\sigma \times]}{\sigma}
\]

\[
+ (1 - \cos \sigma - \sigma a_1 \sin \sigma + \sigma^2 a_2 \cos \sigma) \frac{[\sigma \times]^2}{\sigma^2}
\]

The first term in eqn. (11.14) represents a skew symmetric matrix, following the form of \( [\sigma \times] \), and is denoted below by the symbol \( U \). The second term is symmetric, following \( [\sigma \times]^2 \), and may be represented by a symmetric matrix \( S \). Hence, we may write:

\[
E = U + S
\]

as shown in Reference 5.

\( A \) is an orthogonal matrix, if the equation \( A^T A = I \) is satisfied. For the computed matrix, we may write:

\[
\hat{A}^T \hat{A} = [I + E]^T [I + E]
\]

Ignoring second- and higher-order terms gives:

\[
\hat{A}^T \hat{A} = I + E + E^T
\]
Substituting for $E$ and $E^T$ in terms of their symmetric and skew symmetric components, and noting that $S^T = S$ and $U^T = -U$, gives:

$$\hat{A}^T\hat{A} = I - 2S$$  \hspace{1cm} (11.18)

We can also write:

$$A^T\hat{A} = I + S + U$$  \hspace{1cm} (11.19)

From the last two equations, the following conclusions can be drawn:

$S$ represents the deviation of the matrix $\hat{A}$ from the orthogonal form. The diagonal elements of $S$ are termed the scale errors, whilst the off-diagonal terms represent the skew errors [5].

If $S$ is zero, $\hat{A}$ becomes an orthogonal matrix representing a co-ordinate rotation, which is different from that defined by $A$. $U$ provides a measure of the difference between the two rotations.

A single parameter, $D_{dc}$, the root sum square of the upper or lower off-diagonal elements of $U$, divided by the computer update interval $\delta t$, may be used as a measure of the drift in the computed attitude matrix. The parameter $D_{dc}$ may be used to assess the accuracy of the various orders of attitude algorithm considered.

The matrix $U$ is defined as:

$$U = (\sigma a_1 \cos \sigma - \sin \sigma + \sigma^2 a_2 \sin \sigma) \frac{[\sigma \times]}{\sigma}$$  \hspace{1cm} (11.20)

In the case of a single $x$-axis rotation, where $\sigma = [\sigma \ 0 \ 0]^T$, we have:

$$D_{dc} = \frac{1}{\delta t}(\sigma a_1 \cos \sigma - \sin \sigma + \sigma^2 a_2 \sin \sigma)$$  \hspace{1cm} (11.21)

where

$$a_1 = 1, a_2 = 0 \quad \text{is a first-order algorithm},$$

$$a_1 = 1, a_2 = 0.5 \quad \text{is a second-order algorithm},$$

$$a_1 = 1 - (\sigma^2/6), a_2 = 0.5 \quad \text{is a third-order algorithm}.$$  

11.2.1.2 Example

Consider cases in which the maximum size of the angular increment ($\sigma_{\text{max}}$) is 0.1 and 0.05 rad. If the maximum angular rate of the body is 10 rad/s, these figures correspond to update intervals of 0.01 and 0.005 s.

The drift errors in the computed attitude are shown in Table 11.1 for these two cases using different orders of algorithm.

The substantial improvement in accuracy which may be achieved by reducing the update interval is clearly seen. The large reduction in the drift rate obtained through the inclusion of third-order terms, occurs as a result of the cancellation of both the third- and fourth-order terms in the expression for drift error with this level of truncation. In many applications, high turn rates will not normally be sustained for long periods, the mean rates expected being substantially lower than the figure of
10 rad/s considered in the above analysis. In such cases, the mean drift errors resulting from imprecise attitude computation will be considerably smaller than the figures quoted in the table. It may therefore be feasible to use first- or second-order update algorithms for some applications.

It is assumed here that the major part of the attitude computation described above, along with some other aspects of the navigation processing to be described later, will be implemented at the lower $k$-cycle data rate. However, some parts of the computation may need to be performed at higher rates, whilst others can be carried out less frequently, as discussed below.

### 11.2.2 Rotation angle computation

A further limitation on the accuracy of the direction cosine matrix updates is the accuracy with which the rotation angle, $\sigma$, can be determined. Consider first the case where the direction of the angular rate vector, $\omega$, remains fixed in space over an update interval. In this case, $\sigma$ is determined quite simply as the integral of $\omega$ over the computer cycle, $k$:

$$\sigma = \int_{t_k}^{t_{k+1}} \omega \, dt$$  \hspace{1cm} (11.22)

that is, $\sigma$ is the sum of the incremental measurements provided directly by some gyrosopes over the time interval $t_k$ to $t_{k+1}$. The relationship between $\sigma$ and $\omega$ can also be expressed as $d\sigma/dt = \omega$ in this situation.

In general, however, it is not possible to determine $\sigma$ precisely by simply summing the measurements of incremental angle. If the direction of $\omega$ does not remain fixed in space but is rotating, as occurs in the presence of coning motion, for example, then following Bortz [1], we may write:

$$\dot{\sigma} = \omega + \dot{\epsilon}$$  \hspace{1cm} (11.23)
in which \( \omega \) represents the inertially measurable angular motion and \( \dot{e} \) is a component of \( \sigma \) owing to the non-inertially measurable motion, known as the non-commutativity rate vector.

An expression for \( \sigma \) under general motion conditions, that is, where the motion is not restricted to a single-axis, may be derived by differentiating eqn. (11.10), writing \( dA/dt = A[\omega \times] \) and manipulating vectors formed from the resulting expressions as described in Reference 1 to give:

\[
\dot{\sigma} = \omega + \frac{1}{2} \sigma \times \omega + \frac{1}{\sigma^2} \left[ 1 - \frac{\sigma \sin \sigma}{2(1 - \cos \sigma)} \right] \sigma \times \sigma \times \omega \tag{11.24}
\]

As a result of non-commutativity effects, the final orientation after a series of rotations is dependent on both the individual rotations and the order in which they have occurred. The above equation indicates how the history of previous rotations (\( \sigma \)) and the current angular rate (\( \omega \)) affect the build up of the non-commutativity term (\( e \)).

A practical implementation would require the right-hand side of eqn. (11.24) to be truncated. For instance, writing the sine and cosine terms as series expansions and ignoring terms higher than third order in \( \sigma \), the above equation may be written as

\[
\dot{\sigma} = \omega + \frac{1}{2} \sigma \times \omega + \frac{1}{12} \sigma \times \sigma \times \omega \tag{11.25}
\]

In Reference 3, the following algorithm is proposed:

\[
\delta \alpha_{k+1} = \int_{t_k}^{t_{k+1}} \alpha \times \omega \, dt
\]

where

\[
\alpha = \int_{t_k}^{t} \omega \, dt
\]

\[
\sigma = \alpha_{k+1} + \delta \alpha_{k+1} \tag{11.26}
\]

and \( \alpha_{k+1} \) represents the sum of the incremental angle outputs provided by the gyroscopes from \( t_k \) to \( t_{k+1} \). In general, where significant levels of angular vibration are present, it will be necessary to solve eqn. (11.26) at a higher rate, denoted the \( j \)-cycle update rate, than that at which the direction cosine matrix is updated. It will be necessary to select a \( j \)-cycle update rate which is sufficiently fast to ensure that the value of \( \sigma \) obtained using eqn. (11.26) agrees well with the true value, given by eqn. (11.24), in the presence of the maximum body rates and vibratory motion.

### 11.2.3 Rotation vector compensation

In this section, an expression is derived for the drift error in the computed attitude (\( \delta \alpha \)) arising in the presence of coning motion. Coning refers to the motion which arises when a single-axis of a body describes a cone, or some approximation to a cone, in space. Such motion results from the simultaneous application of angular oscillations about two orthogonal axes of the system where the oscillations are out of phase.
It is assumed here that the body is oscillating at a frequency \( f \) about the \( x \) and \( y \) axes. The amplitudes of the \( x \) and \( y \) motions are \( \theta_x \) and \( \theta_y \), respectively. Additionally, a phase difference of \( \phi \) is assumed to exist between the two channels. Thus, we may write:

\[
\omega = 2\pi f [\theta_x \cos 2\pi ft \quad \theta_y \cos(2\pi ft + \phi) \quad 0]^T
\]

and

\[
\alpha = [\theta_x \{\sin 2\pi ft - \sin 2\pi ft_k\} \quad \theta_y \{\sin(2\pi ft + \phi) - \sin(2\pi ft_k + \phi)\} \quad 0]^T
\]

Substituting for \( \alpha \) and \( \omega \) in the \( \delta \alpha \) equation gives:

\[
\delta \alpha = \pi f \int_{t_k}^{t_{k+1}} \begin{bmatrix}
\theta_x \{\sin 2\pi ft - \sin 2\pi ft_k\} \\
\theta_y \{\sin(2\pi ft + \phi) - \sin(2\pi ft_k + \phi)\} \\
0
\end{bmatrix}
\times
\begin{bmatrix}
\theta_x \cos 2\pi ft \\
\theta_y \cos(2\pi ft + \phi) \\
0
\end{bmatrix}
dt
\]

which yields a \( z \)-component,

\[
\delta \alpha_z = \pi f \theta_x \theta_y \sin \phi \int_{t_k}^{t_{k+1}} \{1 - \cos 2\pi f (t - t_k)\} \, dt
\]

Integrating between the appropriate limits, we have:

\[
\delta \alpha_z = \pi f \theta_x \theta_y \sin \phi \left[ t_{k+1} - t_k - \frac{\sin 2\pi f (t_{k+1} - t_k)}{2\pi f} \right]
\]

Writing \( t_{k+1} - t_k = \delta t \) gives:

\[
\delta \alpha_z = \pi f \theta_x \theta_y \delta t \sin \phi \left[ 1 - \frac{\sin 2\pi f \delta t}{2\pi f \delta t} \right] \tag{11.27}
\]

Thus, although the rate \( \omega \) is cyclic about the \( x \) and \( y \) axes, a \( z \)-component of \( \delta \alpha \) arises, which is a constant proportional to the sine of the phase angle and the amplitude of the motion. It can be seen, from the above equation, that \( \delta \alpha \) is maximised when \( \phi = \pi/2 \). Under such conditions, the motion of the body is referred to as coning, owing to the motion of the \( z \)-axis which describes a cone as space.

Over the time interval \( k\delta t \), the drift error in the computed attitude, which arises if the above correction term is not applied, may be expressed as:

\[
\delta \dot{\alpha}_z = \pi f \theta_x \theta_y \sin \phi \left[ 1 - \frac{\sin 2\pi f \delta t}{2\pi f \delta t} \right] \tag{11.28}
\]

If \( \delta \dot{\alpha}_z \) is small compared with the overall system performance requirement, the need to implement the correction terms described is avoided.
11.2.3.1 Example
Consider a situation where the body exhibits coning motion at a frequency, $f$, of 50 Hz. The angular amplitudes of the motion in $x$ and $y$ is taken to be $0.1^\circ$. If the attitude update frequency is 100 Hz, that is, $\delta t = 0.01$ s, the resulting drift is $100^\circ$/h. By increasing the computational frequency to 500 Hz, the drift figure falls to $\sim 6^\circ$/h.

A more general development of the above equation allows an RMS value for $\delta \alpha_z$ to be calculated in the presence of a given vibration spectrum [3].

11.2.4 Body and navigation frame rotations
Returning now to the continuous form of the attitude eqn. (11.1), the vector $\omega$ represents the turn rate of the body with respect to the navigation reference frame. When navigating with respect to the local geographic frame, this equation takes the following form, as discussed in Chapter 3:

$$\dot{C}_b^n = C_b^n \Omega_{ib}^b - \Omega_{in}^n C_b^n$$

The first term, $C_b^n \Omega_{ib}^b$, is a function of the body rates, as sensed by the strapdown gyroscopes, whilst the second term, $-\Omega_{in}^n C_b^n$, is a function of the lower navigation frame rates. The updating of the direction cosine matrix to take account of the body motion, that is, the solution of the equation $dC_b^n/dt = C_b^n \Omega_{ib}^b$ may be accomplished using eqns. 11.4 and 11.11, as described above.

A similar algorithm may be used to take account of navigation frame rotations. In order to update the direction cosine matrix for navigation frame rotations, an equation similar to eqn. (11.4) may be used, in which $A$ is replaced with a navigation frame rotation direction cosine matrix, $B$, as follows:

$$C_{l+1}^n = B_l C_l^n$$

where $B_l$ represents the direction cosine matrix relating navigation axes at time $t_{l+1}$ to navigation axes at time $t_l$. $B_l$ may be expressed in terms of a rotation vector $\theta$ as follows:

$$B_l = I + \frac{\sin \theta}{\theta} [\theta \times] + \frac{1 - \cos \theta}{\theta^2} [\theta \times]^2$$

where $\theta$ is a rotation vector with magnitude and direction such that a rotation of the navigation frame about $\theta$, through an angle equal to the magnitude of $\theta$, will rotate the navigation frame from its orientation at time $t_l$ to its position at time $t_{l+1}$. The angle $\theta$ may be written as:

$$\theta = \int_{t_l}^{t_{l+1}} \omega_{in}^n \, dt$$

In view of the fact that the navigation frame turn rates will generally be much slower than the body rates, direction cosine updates for rotations of the navigation frame may be implemented at a much slower rate, denoted the $l$-cycle update rate. Additionally, the mathematical functions on which the algorithm is based can be truncated at a lower level.
11.2.4.1 Example

Considering first-order terms only in eqn. (11.31), and applying navigation frame updates at 1 s intervals to take account of rotation of the Earth, the net drift error can be maintained at a negligibly low level. In some short-range missile applications, in which angular rate measurement accuracies of several hundred degrees per hour may be acceptable, the need to take account of navigation frame rotations, for example, Earth’s rate at 15°/h, becomes superfluous.

11.2.5 Quaternion algorithms

Using the quaternion attitude representation, it is required to solve the equation:

\[ \dot{q} = \frac{1}{2} q \cdot p \]  

(11.33)

where \( p = [0, \omega^T] \) as defined in Section 3.4.3.2. This equation may be expressed in matrix form as:

\[ \dot{q} = \frac{1}{2} W q \]  

(11.34)

where

\[ W = \begin{bmatrix}
0 & -\omega_x & -\omega_y & -\omega_z \\
\omega_x & 0 & \omega_z & -\omega_y \\
\omega_y & -\omega_z & 0 & \omega_x \\
\omega_z & \omega_y & -\omega_x & 0
\end{bmatrix} \]  

(11.35)

and \( \omega_x, \omega_y, \) and \( \omega_z \) are the components of \( \omega \).

For the situation in which the orientation of the rate vector, \( \omega \), remains fixed over a computer update interval, the solution to the above equation may be written as:

\[ q_{k+1} = \left[ \exp \left( \frac{1}{2} \int_{t_k}^{t_{k+1}} W \, dt \right) \right] q_k \]  

(11.36)

\[ \int_{t_k}^{t_{k+1}} W \, dt = \Sigma = \begin{bmatrix}
0 & -\sigma_x & -\sigma_y & -\sigma_z \\
\sigma_x & 0 & \sigma_z & -\sigma_y \\
\sigma_y & -\sigma_z & 0 & \sigma_x \\
\sigma_z & \sigma_y & -\sigma_x & 0
\end{bmatrix} \]  

(11.37)

to give

\[ q_{k+1} = \exp \left( \frac{\Sigma}{2} \right) q_k \]  

(11.38)

By expanding the exponential term and following a development similar to that used to obtain the direction cosine solution above, it can be shown that the exponential term may be written in quaternion form as:

\[ q_{k+1} = q_k \cdot r_k \]  

(11.39)
where

\[
\mathbf{r}_k = \begin{bmatrix}
a_c \\
a_s \sigma_x \\
a_s \sigma_y \\
a_s \sigma_z
\end{bmatrix}
\]  
(11.40)

\[
a_c = \cos\left(\frac{\sigma}{2}\right) = 1 - \frac{(0.5\sigma)^2}{2!} + \frac{(0.5\sigma)^4}{4!} - \ldots
\]  
(11.41)

\[
a_s = \frac{\sin(\sigma/2)}{\sigma} = 0.5 \left(1 - \frac{(0.5\sigma)^2}{3!} + \frac{(0.5\sigma)^4}{5!} - \ldots\right)
\]  
(11.42)

and

\[
(0.5\sigma)^2 = 0.25(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)
\]

By comparison with the quaternion definition given in Chapter 3, eqn. (3.32), it can be seen that \(\mathbf{r}_k\) is a quaternion representing a rotation of magnitude \(\sigma\), about a vector \(\sigma\). This is the quaternion which transforms from body axes at time \(t_{k+1}\) to body axes at time \(t_k\), whilst \(\mathbf{q}_k\) represents the quaternion relating body to navigation axes at time \(t_k\). Therefore, the quaternion \(\mathbf{q}\) may be updated for body motion, as sensed by the strapdown gyroscopes, using eqns. (11.39)–(11.42) recursively. The parameter \(\sigma\) is determined as described in Section 11.2.1.

As for the direction cosine algorithm, the update interval is selected to ensure that \(\sigma\) remains small at the maximum body rate, thus avoiding the need to retain a large number of terms in the expressions for \(a_c\) and \(a_s\). The order of the quaternion updating algorithm will be determined by the truncation point selected in eqns. 11.41 and 11.42.

### 11.2.5.1 Definition of attitude errors

In order to quantify the performance of the quaternion update algorithm, a drift parameter \(D_q\) is defined following a development parallel to that used in Section 11.2.1 to determine the drift error in the direction cosine update algorithm. The error in the computed quaternion \(\mathbf{r}\) may be expressed in terms of the true and computed quaternions, denoted \(\mathbf{r}\) and \(\hat{\mathbf{r}}\), respectively, as follows:

\[
\delta\mathbf{r} = \mathbf{r}^* \cdot \hat{\mathbf{r}}
\]  
(11.43)

For a single \(x\)-axis rotation, \(\mathbf{r} = [\cos(0.5\sigma) \sin(0.5\sigma) 0 0]\), \(\hat{\mathbf{r}} = [a_c \sigma a_s 0 0]\) and

\[
\delta\mathbf{r} = \begin{bmatrix}
a_c \cos(0.5\sigma) + \sigma a_s \sin(0.5\sigma) \\
\sigma a_s \cos(0.5\sigma) - a_c \sin(0.5\sigma) \\
0 \\
0
\end{bmatrix}
\]  
(11.44)
This may be expressed as a direction cosine error matrix in accordance with eqn. (3.59). Following the procedure used to define the direction cosine drift in Section 11.2.1, an expression for the quaternion drift may be defined in terms of the off-diagonal elements of the error matrix, viz:

\[
D_q = \frac{2}{\delta t} \{a_c \cos(0.5\sigma) + \sigma a_s \sin(0.5\sigma)\} \{\sigma a_s \cos(0.5\sigma) - a_c \sin(0.5\sigma)\}
\]

\[
= \frac{1}{\delta t} \{2\sigma a_c a_s \cos \sigma - a_c^2 \sin \sigma + \sigma^2 a_s^2 \sin \sigma\}
\]

(11.45)

where \(a_c\) and \(a_s\) are selected according to the order of the algorithm needed as follows:

\[
a_c = 1, \ a_s = 0.5 \quad \text{is a first-order algorithm,}
\]

\[
a_c = 1 - \frac{(0.5\sigma)^2}{2}, \ a_s = 0.5 \quad \text{is a second-order algorithm,}
\]

\[
a_c = 1 - \frac{(0.5\sigma)^2}{2}, \ a_s = 0.5 \left(1 - \frac{(0.5\sigma)^2}{6}\right) \quad \text{is a third-order algorithm.}
\]

### 11.2.5.2 Example

The drift in the attitude computed using different orders of the quaternion algorithm is summarised in Table 11.2, using the same conditions considered in Section 11.2.1 to evaluate the performance of the direction cosine algorithm.

Comparing these results with those tabulated in Section 11.2.1, it can be seen that the quaternion drift figures are smaller than those obtained using direction cosines. This arises principally because the quaternion equations involve the expansion of terms in \(\sin(\sigma/2)\) and \(\cos(\sigma/2)\), whereas the direction cosine equations contain similar terms in \(\sigma\). This also accounts for the correspondence between the quaternion drift figures which arise when \(\sigma\) is 0.1 rad and the direction cosine drifts when \(\sigma\) is 0.05 rad. It is therefore apparent that the quaternion representation yields the more accurate attitude solution, for a given level of truncation, in the presence of a single-axis rotation.

In order to update the quaternion for navigation frame rotations, an equation similar to eqn. (11.39) may be used, in which \(r\) is replaced with a navigation frame.

<table>
<thead>
<tr>
<th>Order of algorithm</th>
<th>Attitude drift error (°/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\sigma_{max} = 0.1) rad</td>
</tr>
<tr>
<td>1</td>
<td>1720</td>
</tr>
<tr>
<td>2</td>
<td>860</td>
</tr>
<tr>
<td>3</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 11.2 Drift in the computed attitude using different orders of the quaternion algorithm
rotation quaternion, \( \mathbf{p} \), as follows:

\[
\mathbf{q}_{l+1} = \mathbf{p}_l \cdot \mathbf{q}_l
\]  

(11.46)

where \( \mathbf{q}_l \) represents the quaternion relating body to navigation axes at computer cycle \( l \), and \( \mathbf{p}_l \) is the quaternion which transforms from body axes at time \( t_{l+1} \) to body axes at time \( t_l \). The quaternion \( \mathbf{p}_l \) may be expressed in terms of the rotation vector \( \mathbf{\theta} \), as follows:

\[
\mathbf{p}_l = \begin{bmatrix} b_c \\ b_s \theta_x \\ b_s \theta_y \\ b_s \theta_z \end{bmatrix}
\]  

(11.47)

where

\[
b_c = \cos \left( \frac{\theta}{2} \right) = 1 - \frac{(0.5\theta)^2}{2!} + \frac{(0.5\theta)^4}{4!} - \ldots
\]  

(11.48)

\[
b_s = \frac{\sin(\theta/2)}{\theta} = 0.5 \left( 1 - \frac{(0.5\theta)^2}{3!} + \frac{(0.5\theta)^4}{5!} - \ldots \right)
\]  

(11.49)

As in the case of the direction cosines, it is assumed that these equations may be implemented at the reduced rate, the \( l \)-cycle, compared with the body motion updates. Additionally, a low order algorithm will usually be sufficient to provide accurate navigation frame updates.

In the situation where the quaternion attitude representation is selected, the following equation may be used to compute the direction cosine matrix \( C^b_\mathbf{p} \) for use in the acceleration vector transformation algorithm, which is discussed in Section 11.3.

\[
C^b_\mathbf{p} = \begin{pmatrix} 1 - 2(c^2 + d^2) & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & 1 - 2(b^2 + d^2) & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & 1 - 2(b^2 + c^2) \end{pmatrix}
\]  

(11.50)

11.2.6 Orthogonalisation and normalisation algorithms

It is common practice, in the implementation of strapdown attitude algorithms, to apply self-consistency checks in an attempt to enhance the accuracy of the computed direction cosines or quaternion parameters. The rows of the direction cosine matrix represent the projection of unit vectors which lie along each axis of the orthogonal reference co-ordinate frame in the body frame. It follows therefore, that the rows of the direction cosine matrix should always be orthogonal to one another and that the sum of the squares of the elements in each row should equal unity. In the case of the quaternion representation, the self-consistency test is to check that the sum of the squares of the four parameters is unity.
Self-consistency checks may be applied as part of the attitude algorithm to ensure that the above conditions are satisfied. If required, it is usually sufficient to carry out these checks at a relatively low rate. This part of the computation may be carried out at the -cycle frequency.

11.2.6.1 Direction cosine checking

The condition for orthogonality of the \(i\)th and \(j\)th rows of the direction cosine matrix, denoted \(C_i\) and \(C_j\), is that their dot product should equal zero, that is, \(C_iC_j^T = 0\). In practice, this is not necessarily the case, and we define:

\[
\Delta_{ij} = C_iC_j^T
\]  \hspace{1cm} (11.51)

where \(\Delta_{ij}\) is an angle error defined about an axis perpendicular to \(C_i\) and \(C_j\), the orthogonality error between the two rows.

Since either row, \(C_i\) or \(C_j\), is equally likely to be in error, the correction is apportioned equally between them using:

\[
\hat{C}_i = C_i - \frac{1}{2} \Delta_{ij}C_j
\]  \hspace{1cm} (11.52)

\[
\hat{C}_j = C_j - \frac{1}{2} \Delta_{ij}C_i
\]  \hspace{1cm} (11.53)

where the \(\hat{\cdot}\) notation denotes the corrected quantity.

Normalisation errors may be identified by comparing the sum of the squares of the elements in a row with unity, that is,

\[
\Delta_{ii} = 1 - C_iC_i^T
\]  \hspace{1cm} (11.54)

and corrections applied using:

\[
\hat{C}_i = C_i - \frac{1}{2} \Delta_{ii}C_i
\]  \hspace{1cm} (11.55)

An alternative approach is to operate on the columns of the direction cosine matrix, in a similar manner to that described above for the rows, as described in Reference 3.

11.2.6.2 Quaternion normalisation

The quaternion can be normalised by comparing the sum of the squares of its elements with unity. The normalisation error is given by:

\[
\Delta q = 1 - q \cdot q^*
\]  \hspace{1cm} (11.56)
The quaternion may be normalised by dividing each element by $\sqrt{(q \cdot q^*)}$. Thus, we may write:

$$q = \frac{q}{\sqrt{(q \cdot q^*)}}$$

$$= \{1 - \Delta q\}^{-0.5} q$$

$$\approx \left\{1 + \frac{1}{2} \Delta q\right\} q \quad (11.57)$$

It should be noted that the orthogonalisation and normalisation cannot correct for errors that have occurred in the previous computation. For example, an error arising in a single element of the quaternion will be 'spread' amongst all of the elements, as a result of a normalisation correction. In the opinion of the authors, such techniques should be used with caution. Emphasis should be placed on the design of the basic attitude update algorithm, rather than placing any reliance on the normalisation process described here, which may serve only to compound fundamental errors in the update algorithm.

### 11.2.7 The choice of attitude representation

The relative benefits of using direction cosines or quaternion parameters to represent attitude has received considerable attention in the published literature [3–5]. The results have largely been found to be inconclusive, although the quaternion method does offer some advantage because, inherently, it gives rise to an orthogonal attitude matrix. In addition, the analysis, given in the preceding sections, has shown the accuracy of attitude computation obtained using quaternions to be superior to the accuracy which may be achieved using the direction cosine representation. These factors account, in part at least, for the popularity of the quaternion method in recent years.

However, the final selection of attitude algorithm is unlikely to be made solely on the grounds of computational accuracy. It is the computing burden and the memory requirements which will be the major factors in determining the method to be adopted for a particular application. In this context, there is still a tendency to opt for the quaternion approach, primarily because there are fewer parameters to update, four quaternions as opposed to nine direction cosines. However, when account is taken of the overall strapdown computing task, including the resolution of the measured specific force vector, the relative benefits of the two methods become less clear cut.

### 11.3 Acceleration vector transformation algorithm

This section is concerned with the computational algorithm required to resolve the measured specific force acceleration components into the navigation reference frame. Care must be taken to allow for changes in body rotation occurring over a computer update interval and a two-speed algorithm may be required for applications where
the system is called upon to operate in a highly vibratory environment [6]. For many applications, a relatively low speed algorithm may be sufficient to resolve accelerations associated with vehicle manoeuvres, whilst a high speed correction term may be included to take account of vibration.

11.3.1 Acceleration vector transformation using direction cosines

The measured specific force vector, \(f^b\), is expressed in the navigation co-ordinate frame using:

\[
f^n = C^n_b f^b \tag{11.58}
\]

where \(C^n_b\) is the direction cosine matrix which transforms from body to reference axes, as defined earlier.

An algorithm to implement this function, which accepts incremental velocity measurements, may be developed by first integrating eqn. (11.58) to give:

\[
u^n = \int_{t_k}^{t_{k+1}} f^n \, dt \tag{11.59}
\]

where \(u^n\) represents the change in velocity, expressed in the navigation frame, over the computer cycle \(t_k\) to \(t_{k+1}\). The velocity vector, \(v^n\), may be determined by summing the values of \(u^n\) computed at each cycle, and correcting for Coriolis accelerations and the effects of gravity as described later, in Section 11.4. The matrix \(C^n_b\) varies continuously with time over the update interval and may be written in terms of the matrix \(C^n_k\), the value of \(C^n_b\) at time \(t_k\), and \(A\), a matrix representing the transformation from body axes at time \(t\) to body axes at the start of the update interval, \(t_k\), that is,

\[
C^n_b = C^n_k A \tag{11.60}
\]

Substituting for \(C^n_b\) in eqn. (11.58) gives:

\[
u^n = C^n_k \int_{t_k}^{t_{k+1}} A f^b \, dt \tag{11.61}
\]

Following eqn. (11.11), \(A\) may be expressed in the form:

\[
A = I + [\alpha \times] + 0.5[\alpha \times]^2 - \cdots
\]

where

\[
\alpha = \int_{t_k}^{t} \omega^b \, dt
\]

Substituting for \(A\) in eqn. (11.61) gives:

\[
u^n = C^n_k \int_{t_k}^{t_{k+1}} [f^b + \alpha \times f^b + 0.5[\alpha \times]^2 f^b - \cdots] \, dt \tag{11.62}
\]
If second- and higher-order terms are ignored, we may write:

$$u^n = C_k \left[ \int_{t_k}^{t_{k+1}} f^b \, dt + \int_{t_k}^{t_{k+1}} \alpha \times f^b \, dt \right]$$  \hspace{1cm} (11.63)

If we now write:

$$v = \int_{t_k}^{t} f^b \, dt$$

and integrate the cross product term in eqn. (11.63) by parts, it can be shown that

$$u^n = C_k \left( v_{k+1} + \frac{1}{2} \alpha_{k+1} \times v_{k+1} + \frac{1}{2} \int_{t_k}^{t_{k+1}} (\alpha \times f^b - \omega^b \times v) \, dt \right)$$  \hspace{1cm} (11.64)

where $\alpha_{k+1} = \alpha$ evaluated over the time interval $t_k$ to $t_{k+1}$, by summing the incremental angle measurements over this period. Similarly, $v_{k+1} = v$ evaluated over the same time interval, by summing the incremental velocity measurements provided by the inertial measurement unit.

In eqn. (11.64), we have an expression for $u^n$ containing three terms:

- the sum of the incremental velocity measurements produced by the inertial measurement unit;
- a cross product of the incremental angle, accumulated from $t_k$ to $t_{k+1}$, and the incremental velocity change over the same period. In the literature, this term is referred to as the rotation correction;
- a dynamic integral term.

In the situation where $f^b$ and $\omega^b$ remain constant over the update interval, we have $\alpha = \omega^b t$ and $v = f^b t$. Substituting for $\alpha$ and $v$ in eqn. (11.48), it can be shown easily that the integral term becomes identically zero under such conditions. However, if $f^b$ and $\omega^b$ vary significantly over the update interval, it may be necessary to evaluate the integral in order to provide compensation for the dynamic motion. In this case, the rate at which the integral is evaluated will need to be well above the frequency of the dynamic motion, that is, at the $j$-cycle update rate referred to in the previous section. An example to illustrate this effect is given in the following section.

11.3.2 Rotation correction

In order to illustrate the need for the rotation correction term in a missile application using strapdown technology, consider a situation in which such a vehicle is manoeuvring in a lateral plane such that, $f^b = [0 \ t \ 0]^{T}$ and $\omega^b = [0 \ 0 \ \omega]^{T}$, that is, simultaneously accelerating and rotating in the yaw plane.

If the acceleration transformation update interval is $\delta t$, then, following eqn. (11.61), the true velocity change over the time interval from $t = 0$ to $t = \delta t$ is
given by:

\[
\mathbf{u}^n = \int_0^{\delta t} \begin{bmatrix}
\cos \omega t & \sin \omega t & 0 \\
-\sin \omega t & \cos \omega t & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
f \\
f \\
f
\end{bmatrix}
dt
\]

\[
\begin{bmatrix}
f \\
f \\
0
\end{bmatrix}
\]

Differencing the expressions for \( \mathbf{u}^{n'} \) and \( \mathbf{u}^n \) gives:

\[
\delta \mathbf{u}^n = \mathbf{u}^{n'} - \mathbf{u}^n
\]

\[
= \begin{bmatrix}
\frac{f}{\omega} (1 - \cos \omega \delta t) & \frac{f}{\omega} (1 - \sin \omega \delta t) & 0 \\
\frac{f \omega \delta t^2}{2} & - \frac{f \omega \delta t^2}{2} & - \frac{f \omega \delta t^2}{6}
\end{bmatrix}^T
\]

An equivalent acceleration error may be written as:

\[
\frac{\delta \mathbf{u}^n}{\delta t} = \begin{bmatrix}
\frac{f \omega \delta t}{2} & - \frac{f \omega \delta t}{2} & - \frac{f \omega \delta t}{6}
\end{bmatrix}^T
\]

\[11.3.2.1 \text{ Example} \]

Take the case of a missile which is accelerating laterally at 20g, whilst travelling at 800 m/s. The associated turn rate is \( \sim 0.25 \text{ rad/s} \). Substituting for \( f \) and \( \omega \) in the above expression, and assuming an update interval of 0.01 s, gives an \( x \)-component of acceleration bias of 25 milli-g. In the presence of a sustained missile manoeuvre, a bias of this magnitude would give rise to significant velocity and position errors, particularly in the case of medium to long-range missile systems. The figure derived above may be put into context by comparing it with the magnitudes of acceleration measurement error which may be acceptable for such applications, typically \( \sim 5-10 \text{ milli-g} \), sometimes less.

It can be shown that the simple rotation correction term, given in eqn. (11.64), is able to compensate for the second-order velocity error which appears in the above example. Higher-order corrections may be applied if considered necessary, the relevant expressions being derived by including extra terms in the expansion of the transformation matrix \( \mathbf{A} \), by substituting in eqn. (11.61).
11.3.3 Dynamic correction

In calculating the change in velocity over a computer cycle, expressed in the navigation co-ordinate frame, the following correction term may need to be applied:

\[ \delta u^n = \frac{1}{2} \int_{t_k}^{t_{k+1}} (\alpha \times a^b - \omega^b \times v) \, dt \]  (11.67)

A good test of the performance of the vector transformation algorithm is its ability to cope with sculling motion. Sculling is characterised by the simultaneous application of in-phase components of angular and linear oscillatory motion with respect to two orthogonal axes. Such motion can be particularly detrimental to system performance if the computational frequency is too low, or if sculling corrections are not applied.

An expression for \( \delta u^n \) is now derived under conditions where the body performs an angular oscillation about the \( x \)-axis, whilst simultaneously oscillating linearly in the \( y \) direction. Writing

\[ \alpha = \int_{t_k}^{t'} \omega^b \, dt \]

where

\[ \omega^b = [2\pi f \theta_x \cos 2\pi ft \ 0 \ 0]^T \]

and

\[ v = \int_{t_k}^{t'} a^b \, dt \]

where

\[ a^b = [0 \ Ay \sin(2\pi ft + \phi) \ 0]^T \]

in which \( Ay \) is the amplitude of the cyclic acceleration. Substituting in the equation for \( \delta u^n \) yields a \( z \)-component

\[ \delta u^n_z = \frac{1}{2} \theta_x A_y \cos \phi \int_{t_k}^{t_{k+1}} \{1 - \cos 2\pi f(t - t_k)\} \, dt \]

\[ = \frac{1}{2} \theta_x A_y \cos \phi \left(t_{k+1} - t_k - \frac{\sin 2\pi f(t_{k+1} - t_k)}{2\pi f}\right) \]  (11.68)

Writing \( t_{k+1} - t_k = \delta t \) gives

\[ \delta u^n_z = \frac{1}{2} \theta_x A_y \delta t \cos \phi \left(1 - \frac{\sin 2\pi f \delta t}{2\pi f \delta t}\right) \]  (11.69)

It can be seen that \( \delta u^n_z \) is maximised when \( \phi = 0 \). Under such conditions, the motion of the body is referred to as sculling. Over the time interval \( \delta t \), an effective acceleration error arises if the above correction term is not applied. This error may be expressed as

\[ \delta \dot{u}^n_z = \frac{1}{2} \theta_x A_y \cos \phi \left(1 - \frac{\sin 2\pi f \delta t}{2\pi f \delta t}\right) \]  (11.70)
If $\delta \dot{u}_p^t$ is small compared with the overall system performance requirement, the need to implement the correction term is avoided.

### 11.3.3.1 Example

Consider a situation in which the body exhibits sculling motion at a frequency, $f$, of 50 Hz. The angular amplitude, $\theta_x$, of the motion is taken to be 0.1°, whilst the cyclic acceleration, with respect to an orthogonal axis, is taken to vary between $\pm A_y$ where $A_y = 10g$.

If the attitude update frequency is 100 Hz, that is, $\delta t = 0.01$ s, the resulting acceleration error is 8.7 milli-g. By increasing the computational frequency to 500 Hz, the magnitude of the error is reduced to $\sim 0.5$ milli-g, the required precision being determined by the actual application, and possibly, its particular phase of flight.

### 11.3.4 Acceleration vector transformation using quaternions

In cases where the attitude is computed in quaternion form, the acceleration vector transformation may be effected using the quaternion parameters directly, using the following equation in place of eqn. (11.64):

$$u^n = q \cdot \left( v_{k+1} + \frac{1}{2} \alpha_{k+1} \times v_{k+1} + \frac{1}{2} \int_{t_k}^{t_{k+1}} (\alpha \times f^b - \omega^b \times v) \, dt \right) \cdot q^*$$

(11.71)

Alternatively, the direction cosine matrix, $C_b^n$, may be calculated from the quaternion parameters, using eqn. (11.50), and the acceleration transformation performed as described in Section 11.3.1.

### 11.4 Navigation algorithm

The computational processes required to determine vehicle velocity and position are not unique to strapdown systems and are described in many of the standard inertial navigation texts. More recently, Bar-Itzhack [7] has considered the benefits of using different computational rates for different parts of the navigation computation. For example, terms involving Earth’s rate do not need to be evaluated as often as terms which are functions of the body rate. Such considerations are also taken account of in the analysis which follows.

The velocity and position equations, given in Chapter 3, may be expressed in integral form as follows:

$$v^n = \int_0^t f^n \, dt - \int_0^t [2\omega_{ie} + \omega_{en}] \times v^n \, dt + \int_0^t g \, dt$$

(11.72)

$$x^n = \int_0^t v^n \, dt$$

(11.73)
It is required to evaluate the integral terms within the navigation processor, in order
to determine vehicle velocity and position. The vectors in the above equations may
be expressed in component form, as follows:

\[ \mathbf{v}^n = [v_N \ v_E \ v_D]^T \]
\[ \mathbf{x}^n = [x_N \ x_E \ -h]^T \]
\[ \omega_{ie} = [\Omega \cos L \ 0 \ -\Omega \sin L]^T \]
\[ \omega_{en} = \begin{bmatrix} \frac{v_E}{R_0 + h} & -\frac{v_N}{R_0 + h} & -\frac{v_E \tan L}{R_0 + h} \end{bmatrix}^T \]
\[ \mathbf{g} = [0 \ 0 \ g]^T \]

The above expressions apply for navigation in the vicinity of the Earth, in the local
vertical geographic frame. The first integral term in eqn. (11.72) represents the sum
of the velocity changes over each update cycle, \( u^n \):

\[ u^n = \int_{t_k}^{t_{k+1}} \mathbf{f}^n \, dt \]

\( u^n \) may be determined using the eqn. (11.48), developed in the previous section. Since
this term is a function of vehicle body attitude, \( \mathbf{f}^n = C_b^e \mathbf{f}^b \), it must be calculated at a
rate which is high enough to take account of vehicle dynamic motion. Including the
gravity contribution, the velocity vector may be updated over the time interval \( t_k \) to
\( t_{k+1} \) using:

\[ \mathbf{v}^n_{k+1} = \mathbf{v}^n_k + u^n + \mathbf{g}\delta t \]

The second integral term in eqn. (11.50) includes the Coriolis correction. In general,
the contribution to \( \mathbf{v}^n \) is small compared with the other terms in the equation. Since
the rate of change of the magnitude of the Coriolis term will be relatively low, it is
sufficient to apply this correction at the relatively low \( l \)-update rate, referred to in
Section 11.3, using a fairly simple algorithm, viz:

\[ \mathbf{v}^n_{l+1} = [I - 2\Omega_{ie}\delta t_l - \Omega_{en}\delta t_l] \mathbf{v}^n_l \] (11.74)

where

\[ \Omega_{ie} = [\omega_{ie} \times], \]
\[ \Omega_{en} = [\omega_{en} \times], \]
\[ \mathbf{v}^n_l = \text{velocity vector at time } t_l, \]
\[ \delta t_l = t_{l+1} - t_l. \]

Finally, position may be derived by integrating the velocity vector, as shown in
eqn. (11.73).

The choice of integration scheme will, of course, be dependent on the application.
For relatively short range, low accuracy, applications, a low order scheme such as
rectangular or trapezoidal integration is likely to be adequate. Equations for updating position over the time interval $t_k$ to $t_{k+1}$ are shown below.

Rectangular integration:

$$x_{k+1}^n = x_k^n + v_k^n \delta t \quad (11.75)$$

Trapezoidal integration:

$$x_{k+1}^n = x_k^n + \left(\frac{v_k^n + v_{k+1}^n}{2}\right) \delta t \quad (11.76)$$

For aircraft and ship-borne inertial system applications, in which the performance requirements are more demanding, a higher order integration scheme such as Simpson's rule or fourth-order Runge–Kutta integration [8] may be needed. Simpson’s rule:

$$x_{k+1}^n = x_k^n + \left(\frac{v_{k-1}^n + 4v_k^n + v_{k+1}^n}{6}\right) \delta t \quad (11.77)$$

It is often required to determine position relative to the Earth in terms of height above the Earth and the angular orientation of the current local vertical navigation reference frame with respect to the Earth frame; commonly expressed as latitude and longitude. To avoid mathematical singularities, the angular position parameters may be expressed in the form of a direction cosine matrix relating the navigation (n) and Earth (e) frames. The position direction cosine matrix and height propagate in accordance with the following differential equations:

$$\dot{h} = -v_D \quad (11.78)$$

$$\dot{C}_n^e = C_n^e \Omega_{en}^n \quad (11.79)$$

where $\Omega_{en}^n = [\omega_{en}^n \times]$ and $\omega_{en}^n = [\dot{\ell} \cos L - \dot{L} \quad -\dot{\ell} \sin L]^{T}$ as described in Chapter 3.

An algorithm for implementation in the navigation software can be formulated based on the integral of the above equation.

$$h_l = h_{l-1} + \Delta h_l \quad (11.80)$$

$$C_{nl+1}^e = C_{nl}^e \Delta C_{l+1}^l \quad (11.81)$$

where $\Delta C_{l+1}^l$ represents the direction cosine matrix relating navigation axes at time $t_{l+1}$ to navigation axes at time $t_l$. $\Delta C_{l+1}^l$ may be expressed in terms of a rotation vector $\theta$ as follows:

$$\Delta C_{l+1}^l = I + \frac{\sin \zeta}{\zeta} [\zeta \times] + \frac{1 - \cos \zeta}{\zeta^2} [\zeta \times]^2 \quad (11.82)$$

where $\zeta$ is a rotation vector with magnitude and direction such that a rotation of the navigation frame about $\zeta$, through an angle equal to the magnitude of $\zeta$, will rotate the navigation frame from its orientation at time $t_l$ to its position at time $t_{l+1}$.

$$\zeta \approx \int_{t_l}^{t_{l+1}} \omega_{en}^n \, dt \quad (11.83)$$
To achieve precision updating of the rotation angle vector $\xi$ in the presence of rotations of the position vector from computer cycle $l$ to $l + 1$, an algorithm of the form used for velocity updating may be used; as described by Savage [9]. The resulting algorithm includes terms to compensate for the effects of position vector rotation and the combined dynamic effect of angular rate and specific force which gives rise to rectification errors. The term 'scrolling' has been used to refer to the dynamic error effect [11, 12], which is analogous to the sculling effect which affects the updating of the velocity vector under dynamic conditions.

11.5 Summary

Strapdown navigation computation involves the determination of vehicle attitude, velocity and position from measurements of angular rate and specific force, obtained from a set of inertial instruments rigidly mounted in the vehicle. Measurements made by the inertial sensors are used in various equations to provide the desired navigation information. In the foregoing discussion, three distinct areas of strapdown computation have been described, namely, attitude computation, transformation of the specific force vector and navigation computation. Techniques have been outlined which may be used to implement each of these functions, in real time, together with some analysis of their application.

In this chapter, the equations to be implemented in an inertial navigation system processor have been described. Algorithms may be developed, based on the equations given, which will accept and process inertial measurements in incremental form directly. In order to achieve the desired real time solution of the various algorithms, it is proposed that the strapdown computation should be split into low-, medium- and high-frequency sections as follows:

**Low-frequency computation (l-cycle).** Certain parts of the strapdown navigation equations involve terms which vary very slowly with time, 'Earth's rate' terms for instance. It is therefore sufficient to implement these sections of the algorithms at relatively low rates. In particular, the updating of the computed attitude for navigation frame rotations, and the application of Coriolis corrections to the navigation calculations, may be carried out at the low rate. In addition, algorithms for attitude orthogonalisation and normalisation, if required, may be implemented at the low rate.

**Medium-frequency computation (k-cycle).** It is intended that the bulk of the computation should be implemented at the medium rate which will need to be selected to cope with the large amplitude dynamic motion, arising as a result of vehicle manoeuvres. This includes the updating of the quaternions or direction cosines, the transformation of the acceleration vector and the solution of the major part of the navigation equations.

**High-frequency computation (j-cycle).** In order to cope with vibratory motion, coning and sculling, for instance, it may be necessary to carry out some relatively simple calculations at high speed. This allows compensation for variations in the applied
angular rate and linear acceleration occurring between the updates which take place at a lower frequency.

The various computational functions to be implemented are summarised in Figure 11.2. It is assumed that the inertial measurement unit will be capable of providing outputs at intervals, consistent with the highest computational frequency which is required.

In modern strapdown systems, emphasis is placed on the use of more precise algorithms based, wherever possible, on closed-form solutions of the analytically exact integral solutions to the attitude and navigation differential equations described in this chapter. Resulting improvements in computational precision allow the strapdown algorithms to be validated more easily, since test results can be expected to be in good agreement with analytical solutions under defined motion conditions. The general use of more precise algorithms has become possible through the application of modern computer technology, in which fast processing speed and long floating point word lengths combine to allow such algorithms to be implemented. The reader wishing to gain greater insight into the mathematical derivation of modern strapdown algorithms and processing techniques is referred to the various publications by Paul Savage who has worked and written extensively on this subject [9–12].

References

8 JEFFREY, A.: ‘Mathematics for engineers and scientists’ (Nelson, 1979)
12 SAVAGE, P.G.: ‘Strapdown system computational elements’. NATO RTO Lecture Series No. 232, October 2003