Quaternions: An Introduction with Octave m-file examples

A. S. Hodel a.s.hodel@eng.auburn.edu

Revision : 1.1 (c) 1998

Acknowledgement

The author gratefully thanks the Flight Controls and Dynamics Branch at Marshall Space Flight Center for their useful discussions that led to this document and m-file suite. Citations are given where they were available to me. I welcome any suggestions for additional reference material to be placed in the bibliography.

Contents

1 Quaternions

A convenient way to represent the relative orientation of two frames is through the use of quaternions.

1.1 Definition

Definition 1.1 A quaternion is a 4-vector defined by

$$q = a\hat{\imath} + b\hat{\jmath} + c\hat{k} + d, \;\; a, b, c, d \in \mathrm{I\!R}$$

where $\hat{\imath}$, $\hat{\jmath}$, $\hat{k}$ are quantities satisfying $\hat{\imath}^2 = \hat{\jmath}^2 = \hat{k}^2 = -1$ and

$$\begin{array}{ccc} \hat{\imath}\hat{\jmath} = \hat{k} & \hat{\jmath}\hat{k} = \hat{\imath} & \hat{k}\hat{\imath} = \hat{\jmath} \\ \hat{\jmath}\imath 1 = -\hat{k} & \hat{k}\hat{\jmath} = -\hat{\imath} & \hat{\imath}\hat{k} = -\hat{\jmath} \end{array}$$

Remark 1.1 Notice that $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ are scalar quantities equal to $\sqrt{-1}$; however, $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ do not commute under multiplication. Therefore quaternions do not commute under multiplication ($q_1 q_2 \neq q_2 q_1$ in general). This is consistent with the interpretation of quaternions as spatial rotations (to be discussed in §1.4.2.

M-file `quaternion` Create/extract quaternion information:

```
quaternion: construct or extract a quaternion
w = a*i + b*j + c*k + d from given data.

calling formats:
[a,b,c,d]    = quaternion(w)              -or-
[vv,theta] = quaternion(w)
w            = quaternion(a,b,c,d)
w            = quaternion(vv,theta)
```

(`vv,theta` format to be explained later)

1.2 Multiplication of quaternions

From Definition 1.1, we have

$$\begin{aligned} q_1 q_2 &= (a_1\hat{\imath} + b_1\hat{\jmath} + c_1\hat{k} + d_1)(a_2\hat{\imath} + b_2\hat{\jmath} + c_2\hat{k} + d_2) \\ &= (a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2)i + (c_1 a_2 - a_1 c_2 + b_1 d_2 + d_1 b_2)\hat{\jmath} \\ &\quad + (a_1 b_2 - b_1 a_2 + c_1 d_2 + d_1 c_2)\hat{k} + (-a_1 a_2 - b_1 b_2 - c_1 c_2 + d_1 d_2) \end{aligned} \tag{1.1}$$

Define $v_1 = \begin{bmatrix} a_1 & b_1 & c_1 \end{bmatrix}^T$ and $v_2 = \begin{bmatrix} a_2 & b_2 & c_2 \end{bmatrix}^T$. Then [Mul] equation (1.1) may be rewritten as

$$q_1 q_2 = d_1 d_2 + d_1 v_2 + d_2 v_1 - \langle v_1, v_2 \rangle + (v_1 \times v_2) \tag{1.2}$$

where the cross product

$$(v_1 \times v_2) \triangleq \left\| \begin{bmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{bmatrix} \right\|$$

This vector-based interpretation will become more important in §1.4.2.

M-file `qmult`

```
function c = qmult(a,b)
multiply two quaternions
```

Definition 1.2 The *conjugate* of a quaternion $q = a\hat{\imath} + b\hat{\jmath} + c\hat{k} + d$ is $q^* = -a\hat{\imath} - b\hat{\jmath} - c\hat{k} + d$. Notice that $qq^* = q^*q = a^2 + b^2 + c^2 + d^2 \triangleq |q|^2$. For quaternions q with $qq^* \neq 0$ we define the *inverse quaternion* $q^{-1} = \dfrac{q^*}{qq^*}$. Notice that $qq^{-1} = q^{-1}q = 1$.

M-file `qinv` Compute the inverse of a quaternion

```
function b = qinv(a)
return the inverse of a quaternion
a =  [w,x,y,z] = w*i + x*j + y*k + z
qmult(a,qinv(a)) = 1 = [0 0 0 1]
```

Example 1.1 Compute and multiply the inverse quaternion:

```
octave:1> q = quaternion(1,2,3,4); qi = qinv(q)
qi = -0.033333  -0.066667  -0.100000   0.133333
octave:2> qmult(qi,q)
ans = 0  0  0  1
octave:3> qmult(q,qi)
ans = 0  0  0  1
```

In both cases the product is the identiy quaternion $0\hat{\imath} + 0\hat{\jmath} + 0\hat{k} + 1 = 1$.

1.3 Quaternions as rotations

Definition 1.3 A *unit quaternion* $q = a\hat{\imath} + b\hat{\jmath} + c\hat{k} + d$ satisfies $\left\| \begin{bmatrix} a & b & c & d \end{bmatrix} \right\|_2 = 1$.

Remark 1.2 Unit quaternions may be considered as representations of a rotation of θ degrees about a given unit vector v (see routine `quaternion`). The transformation is accomplished by multiplying $v_r = qvq^{-1}$, where $v = \begin{bmatrix} x & y & z & 0 \end{bmatrix}$.

With this interpretation, unit quaternions may be written as

$$q = \left(ai + b\hat{\jmath} + c\hat{k}\right) \sin(\theta/2) + \cos(\theta/2) \triangleq \left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}, \theta \right) \tag{1.3}$$

Notice that this implies that $\begin{bmatrix} a & b & c \end{bmatrix}$ is also a unit vector. This is consistent with multiplication by the identity quaterion $q = 1$, since $\cos(0) = 1$ and $\sin(0) = 0$.

Example 1.2 Unit quaternions may be multiplied to obtain a single equivalent coordinate rotation. Let q_3 be a 90° rotation about the $x-$axis. Follow this rotation by q_2, a 90° rotation about the (new) y axis. Finally, follow this rotation by q_1, a 90° rotation about the (newly revised) z axis. Then $q = q_1 q_2 q_3$ is a 90 degree rotation about the original y-axis.

```
octave:1> degrees = pi/180;
octave:2> q1 = quaternion([0,0,1],90*degrees);
octave:3> q2 = quaternion([0,1,0],90*degrees);
octave:4> q3 = quaternion([1,0,0],90*degrees);
octave:5> q = qmult(q1,qmult(q2,q3));
octave:6> [vv,th] = quaternion(q)
vv = 0 1 0
th = 1.5708
octave:7> th/degrees
ans = 90.000
```

The rotation about a vector axis interpretation of quaternions requires care: the axis of rotation is taken based on the coordinate frame for which the quaternion was written. For example, a rotation around the inertial y-axis followed by a rotation around the inertial x axis can be written as

$$q = q_x q_y = \left(\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \theta_x \right) \left(\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T, \theta_y \right)$$

This is inconsistent with the definition of Euler angles, in which each angle is defined by a frame based upon the previous rotation. This problem is addressed by coordinate transformation by quaternions:

M-file `qtrans` Transform a quaternion in one frame to the coordinate basis of another frame.

```
function v = qtrans(v,q)
transform the vector v (in quaternion form) by the quaternion q;
v = [x y z 0], q = transformation quaternion
returns v = q*v/q
```

Remark 1.3 Given a quaternion q_f describing the rotation from an inertial frame (x_1, y_1, z_1) to second frame (x_2, y_2, z_2), a quaternion q_r defined in the second frame may be applied by multiplying $q_f q_r {q_f}^{-1}$:

1. Transform (rotate) frame 2 (x_2, y_2, z_2) back to the inertial frame.
2. Perform the rotation q_r.
3. Rotate back to the second frame via q_f.

That is, if $q_r = \left(\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T, \theta_r \right)$ is a rotation about the y axis in the frame (x_2, y_2, z_2), the quaternion $q_f q_r {q_f}^{-1}$ is the equivalent rotation in the frame (x_1, y_1, z_1).

Example 1.3 Quaternion algebra $-90°$ rotation about x followed by $90°$ rotation about revised y and z axes is equivalent to a rotation about the $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ vector.

```
qx  = quaternion([1,0,0],-pi/2);    # elementary rotations
qy1 = quaternion([0,1,0],pi/2);
qz2 = quaternion([0,0,1],pi/2);

qyi = qtrans(qy1,qx);               # transform back to original coordinates
qzi = qtrans(qtrans(qz2,qy1),qx);
qzi2= qtrans(qz2,qyi);              # NOT THE SAME as qzi!
qzi3= qtrans(qz2,qmult(qyi,qx));    # This matches  qzi

[qyiv,thyi] = quaternion(qyi)       # check vectors, angles
[qziv,thzi] = quaternion(qzi)
[qzi2v,thz2i] = quaternion(qzi2)
[qzi3v,thz3i] = quaternion(qzi3)

qtot = qmult(qzi,qmult(qyi,qx))
qtotchk = qmult(qx, qmult(qy1,qz2))
```

Output:

```
qyiv = 0.00000   0.00000  -1.00000
thyi = 1.5708
```

q_{y_1} in the q_x frame is equivalent to a rotation about the original $-z$ axis.

```
qziv = 1.0000e+00   1.5701e-16   3.4863e-32
thzi = 1.5708
qzi2v = 1.5701e-16   -1.5701e-16    1.0000e+00
thz2i = 1.5708
qzi3v = 1.0000e+00   1.9626e-16   3.9252e-17
thz3i = 1.5708
```

One may be tempted to view q_{y_i} as the rotation to the basis frame of q_{z_2}, but this is incorrect. q_{y_i} merely represents the rotation q_{y_1} in the reference frame; both rotations q_x and q_{y_i} must be used to back-transform q_{z_2} into the reference frame. This is shown as follows:

$$\begin{aligned}
q_{y_i} &= q_x q_{y_1} {q_x}^{-1} \\
q_{z_i} &= q_x \left(q_{y_1} q_{z_2} {q_{y_1}}^{-1} \right) {q_x}^{-1} \\
q_{z_i 2} &= q_{y_i} q_{z_2} {q_{y_i}}^{-1} \\
&= q_x q_{y_1} {q_x}^{-1} q_{z_2} \left(q_x q_{y_1} {q_x}^{-1} \right)^{-1} = q_x q_{y_1} {q_x}^{-1} q_{z_2} q_x {q_{y_1}}^{-1} {q_x}^{-1} \neq q_{z_i} \\
q_{z_i 3} &= \left(q_{y_i} q_x \right) q_{z_2} \left(q_{y_i} q_x \right)^{-1} = q_{y_1} q_x q_{z_2} {q_x}^{-1} {q_{y_1}}^{-1} = q_{z_i}
\end{aligned}$$

```
qtot = -5.5511e-17    7.0711e-01    1.2326e-32    7.0711e-01
qtotchk = -5.5511e-17    7.0711e-01    5.5511e-17    7.0711e-01
```

1.4 Application of quaternions in coordinate frame transformations

Interpretations of quaternions.

1.4.1 Euler angles

(yaw-pitch-roll) may be used to construct an inertial-to-body quaterion q_{bi} as follows. The yaw rotation is about the z axis in the inertial frame

$$q_y = \left(\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, \psi \right)$$

The pitch rotation is about the y axis in frame 1 (y-axis after the yaw rotation is made)

$$q_{p_1} = \left(\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \theta \right)$$

The roll rotation is made about the x axis in frame 2

$$q_{r_2} = \left(\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \varphi \right)$$

In order to construct a single quaterion q_{bi} relating the inertial to body axis rotation, rotations q_{p_1} and q_{r_2} must be expressed in the inertial frame:

$$\begin{aligned} q_p &= q_y q_{p_1} {q_y}^{-1} \\ q_r &= q_y q_{r_1} {q_y}^{-1} = q_y q_{p_1} q_{r_2} {q_{p_1}}^{-1} {q_y}^{-1} \end{aligned}$$

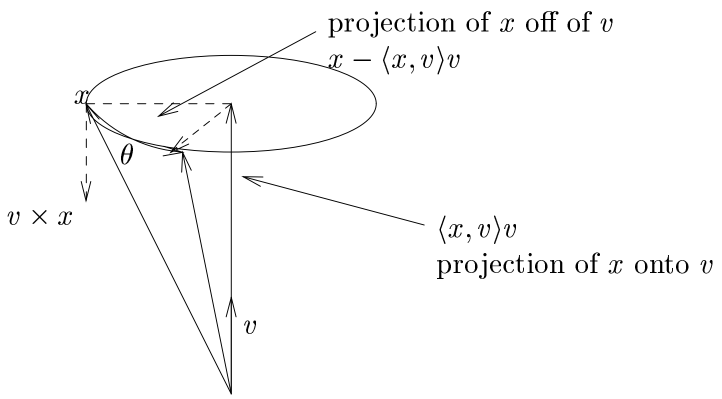
and so

$$q_{bi} = q_r q_b q_y = \left(q_y q_{p_1} q_{r_2} {q_{p_1}}^{-1} {q_y}^{-1} \right) \left(q_y q_{p_1} {q_y}^{-1} \right) y_y = q_y q_{p_1} q_{r_2}$$

Notice that the order of the rotations is reversed when the untransformed rotations q_y, q_{p_1}, q_{r_2} are used instead of quaternions transformed to the inertial frame q_r, q_b, q_y.

1.4.2 Coordinate transformation matrices and quaternions

A unit quaternion (v, θ) may be interpreted as a rotation of θ degrees about the axis v. Application of this rotation to a vector x is illustrated below:



The vector $x_e = \langle x, v \rangle\, v$ is the projection of x onto v and is unchanged by the rotation. The vector $x_{e\perp} = x - x_e$ is orthogonal to v and is rotated by θ degrees about x. The plane of the rotation is spanned by $x_{e\perp}$ and $v \times x$ Thus the rotated vector $x - r$ may be written as

$$x_r = x_e + (x - x_e)\cos(\theta) + (v \times x)\sin(\theta). \tag{1.4}$$

where the first term reflects the unchanged portion of x and the second two terms denote the rotation in the plane normal to v.

The coordinate transformation applied to x may be computed by applying equation (1.4) to the x, y, z unit vectors. The above application involves the rotation of a vector about an axis. Since

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ x \end{bmatrix} = (v_2 z - v_3 y)\hat{\imath} + (v_3 x - v_1 z)\hat{\jmath} + (v_1 y - v_2 x)k$$

we may write

$$\begin{aligned} x_r &= v_1 v + (x - v_1 v)\cos\theta + \begin{bmatrix} 0 \\ v_3 \\ -v_2 \end{bmatrix} \sin\theta \\ y_r &= v_2 v + (x - v_2 v)\cos\theta + \begin{bmatrix} -v_3 \\ 0 \\ v_1 \end{bmatrix} \sin\theta \\ z_r &= v_3 v + (x - v_3 v)\cos\theta + \begin{bmatrix} v_2 \\ -v_1 \\ 0 \end{bmatrix} \sin\theta \end{aligned}$$

(The signs are reversed since we're moving the coordinate axes, not the vector.)

An alternative derivation for the transformation matrix from equation (1.4) is as follows. Define $\bar{q} = \text{imag}(q) = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T = v\sin(\theta/2)$, $\|v\| = 1$. Then

$$\begin{aligned} x_r &= vv^T x + (x - vv^T x)\cos(\theta) - ((v \times e_1)x_1 + (v \times e_2)x_2 + (v \times e_3)x_3)\sin(\theta) \\ &= \left(I\cos\theta + vv^T(1 - \cos\theta) + (v \times I)\sin\theta\right) x \end{aligned}$$

where $v \times I$ is defined by the column-by-column vector cross product. From equation 1.3 and the trigonometric half-angle formulae we have

$$\begin{aligned} \cos\theta &= \cos^2(\theta/2) - \sin^2(\theta/2) = 1 - 2\sin^2(\theta/2) = 1 - 2\left(1 - \text{real}(q)^2\right) \\ \sin\theta &= 2\cos(\theta/2)\sin(\theta/2) = 2\text{real}(q)\sin(\theta/2) = 2q_4\sin(\theta/2). \end{aligned}$$

Substituting the above into x_r we obtain

$$\begin{aligned} x_r &= \left[I(2q_4^2 - 1) + 2vv^T \left(\frac{\sin^2(\theta/2)}{\sin^2(\theta/2)} \right) (1 - q_4^2) + 2(v \times I) q_4 \sin\theta/2 \right] x \\ &= \left[I(2q_4^2 - 1) + 2\bar{q}\bar{q}^T \frac{1 - q_4^2}{1 - \cos^2(\theta/2} + 2(\bar{q} \times I) q_4 \right] x \\ &= \left[I(2q_4^2 - 1) + 2\bar{q}\bar{q}^T + 2(\bar{q} \times I) q_4 \right] x \\ &= (2q_4^2 - 1)x + 2\bar{q}\bar{q}^T x + 2q_4(\bar{q} \times x) \end{aligned}$$

Remark 1.4 The above derivation treats a quaternion as a transformation upon a vector within a linear space. An alternative convention is used in [RW91], in which the quaternion is treated as a transformation upon the basis vectors of a frame. The latter convention results in a sign change of the angle of rotation, i.e.,

$$x_r = (2q_4^2 - 1)x + 2\bar{q}\bar{q}^T x + 2q_4(\bar{q} \times x)$$

Use of the latter convention results in a similar change in sign in the time derivative of the quaternion, discussed in 1.5.

M-file `qtransv` Rotate a vector

```
qtransv is the function defined from: /home3/hodel/oct/quat/qtransv.m

 vr = qtransv(vv,qr)
 rotate a 3-vector as specified by quaternion qr
 q = (ee,th)  (vector, angle notation)
 vr = (vv . ee)*ee [projection on ee unchanged ]
      + [vv - (vv . ee)*ee] cos(th)  [what's left gets scaled by cosine]
      + (vv x ee) sin(th)            [and the sine term completes the rotation]
```

M-file `qtransvmat` Obtain 3×3 rotation matrix from a quaternion

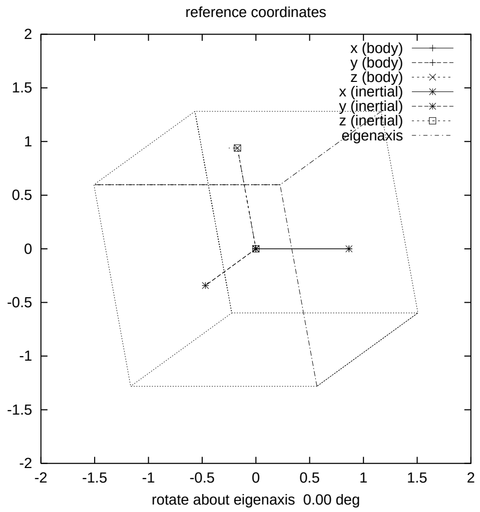
```
[xv,yv,zv] = qtransvmat(q)
xm = qtransvmat(q), xm = [xv yv zv]
compute x,y,z axes rotated per specified quaternion q
```

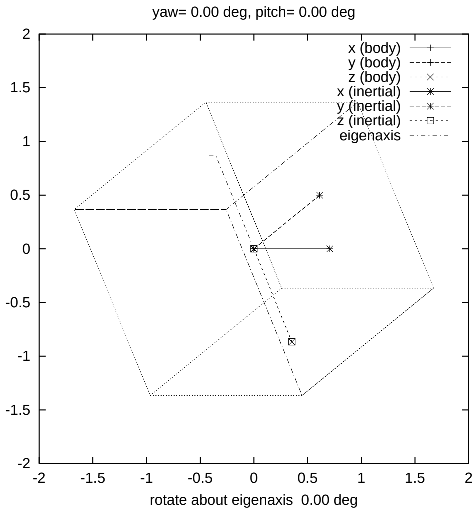
Example 1.4 Use of quaternions to view body/inertial frame transformations:

```
degrees = pi/180;    daz = 30*degrees;    del = -20*degrees;
qazimuth = quaternion([0,0,1],daz);
qelevation = quaternion([cos(daz),sin(daz),0],del);
qview = qmult(qelevation,qazimuth);

th = 0:5:20;
ov = ones(size(th));
myth = [th,max(th)*ov ; 0*ov,th];
myth = [[0:5:20] , 20*ones(1,4),20*ones(1,4); ...
        zeros(1,5), 5:5:20, 20*ones(1,4); ...
        zeros(1,5), zeros(1,4), [5:5:20]];

# inertial frame quaternion
qin = quaternion([1 0 0],180*degrees);
for kk=1:length(myth(1,:))
  figure(kk)
  thy = myth(1,kk);   thp = myth(2,kk);   thr = myth(3,kk);
```

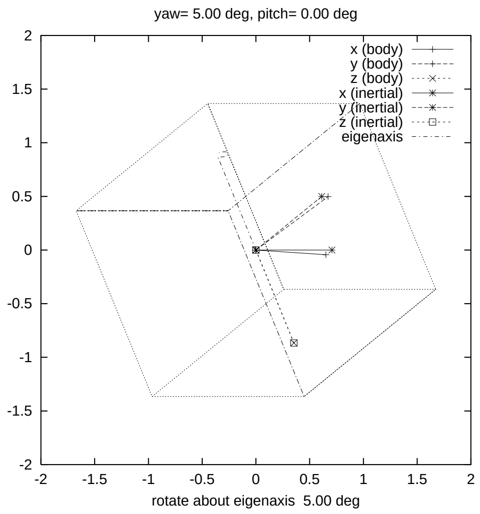
```
  # avoid transformations by reversing the order of multiplication!
  qyi = quaternion([0,0,1],thy*degrees);
  qp1 = quaternion([0,1,0],thp*degrees);
  qr2 = quaternion([1,0,0],thr*degrees);
  qbi = qmult(qyi,qmult(qp1,qr2));

  printf("yaw=%8.4f, pitch=%8.4f, \n    qbi = (%8.4f)i + (%8.4e)j + (%8.4f)k + (
%8.4f)\n",thy,thp, ...
        qbi(1), qbi(2), qbi(3), qbi(4));
  [vv,th] = quaternion(qbi);
  printf("      = (vector) = [%8.4f %8.4f %8.4f], th=%5.2f deg\n", ...
        vv(1), vv(2), vv(3), th*180/pi);

  # transform qbi to reference coordinates
  qb = qmult(qin,qbi);
  title(sprintf("yaw=%5.2f, pitch=%5.2f, roll=%5.2f (deg)",thy,thp,thr))
  coordinate_plot(qin,qb,qview);
  gset terminal postscript eps
  eval(sprintf("gset output 'fig%d.eps'",kk));
  replot
  gset terminal x11
endfor
```

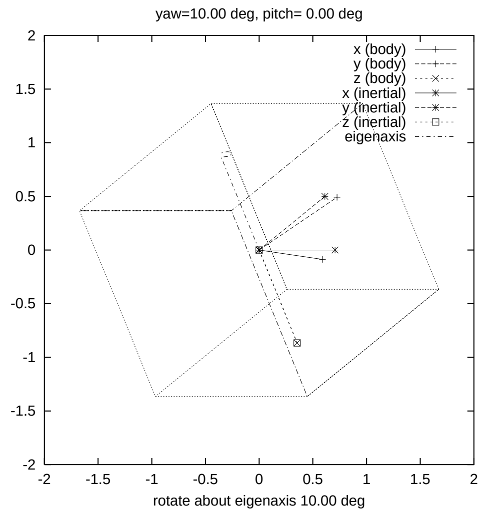
Results:[1]

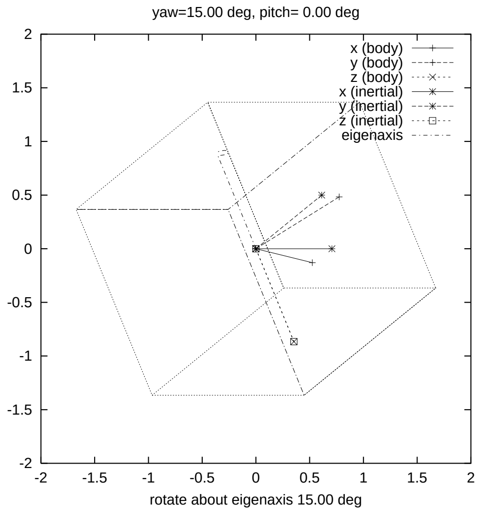


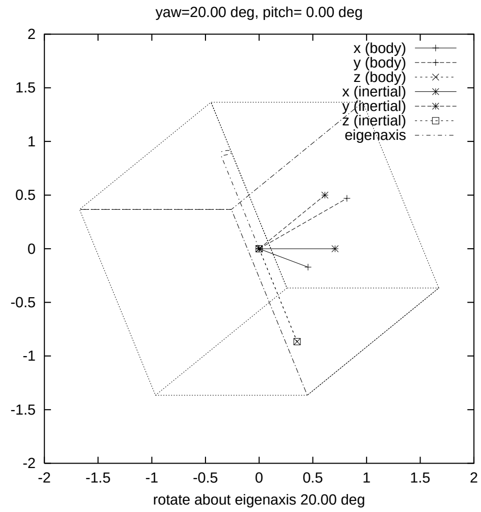


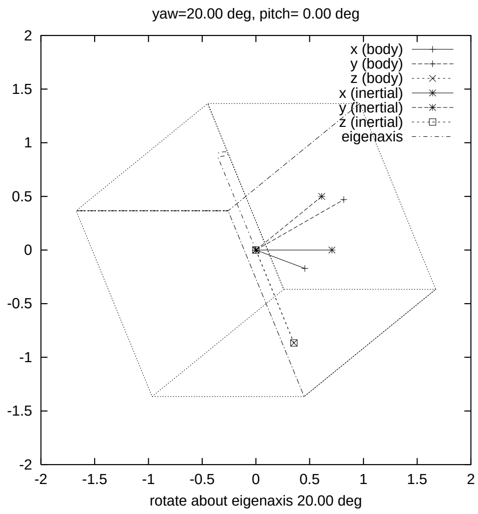
[1] due to details of visualization software, the $x - y - z$ axes do not appear as a right-handed coordinate system. Sorry.

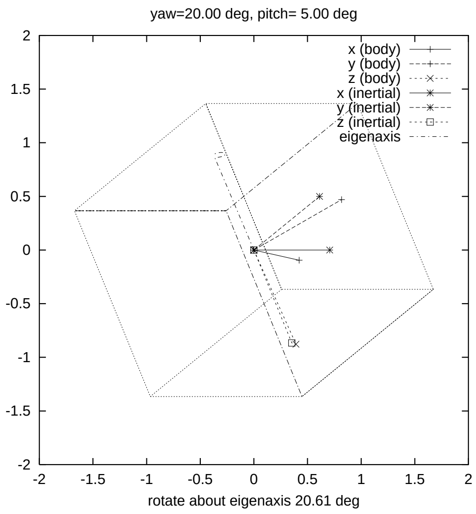

yaw= 5.00 deg, pitch= 0.00 deg
x (body)
y (body)
z (body)
x (inertial)
y (inertial)
z (inertial)
eigenaxis
rotate about eigenaxis 5.00 deg

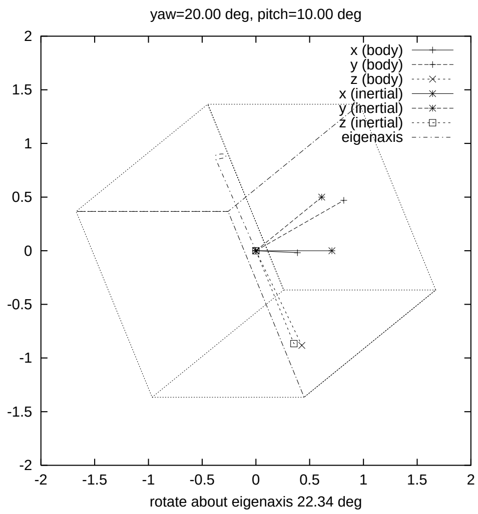

yaw=10.00 deg, pitch= 0.00 deg
x (body)
y (body)
z (body)
x (inertial)
y (inertial)
z (inertial)
eigenaxis
rotate about eigenaxis 10.00 deg

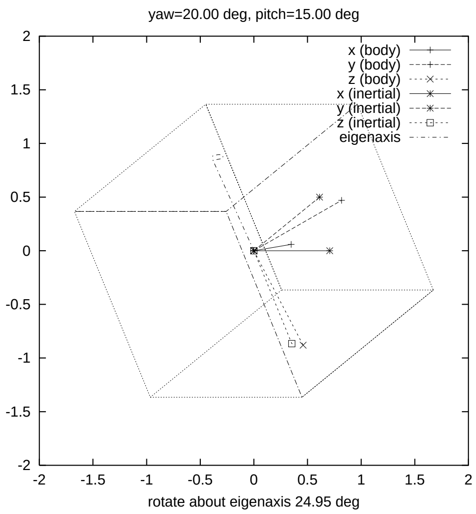

yaw=15.00 deg, pitch= 0.00 deg
x (body)
y (body)
z (body)
x (inertial)
y (inertial)
z (inertial)
eigenaxis
rotate about eigenaxis 15.00 deg

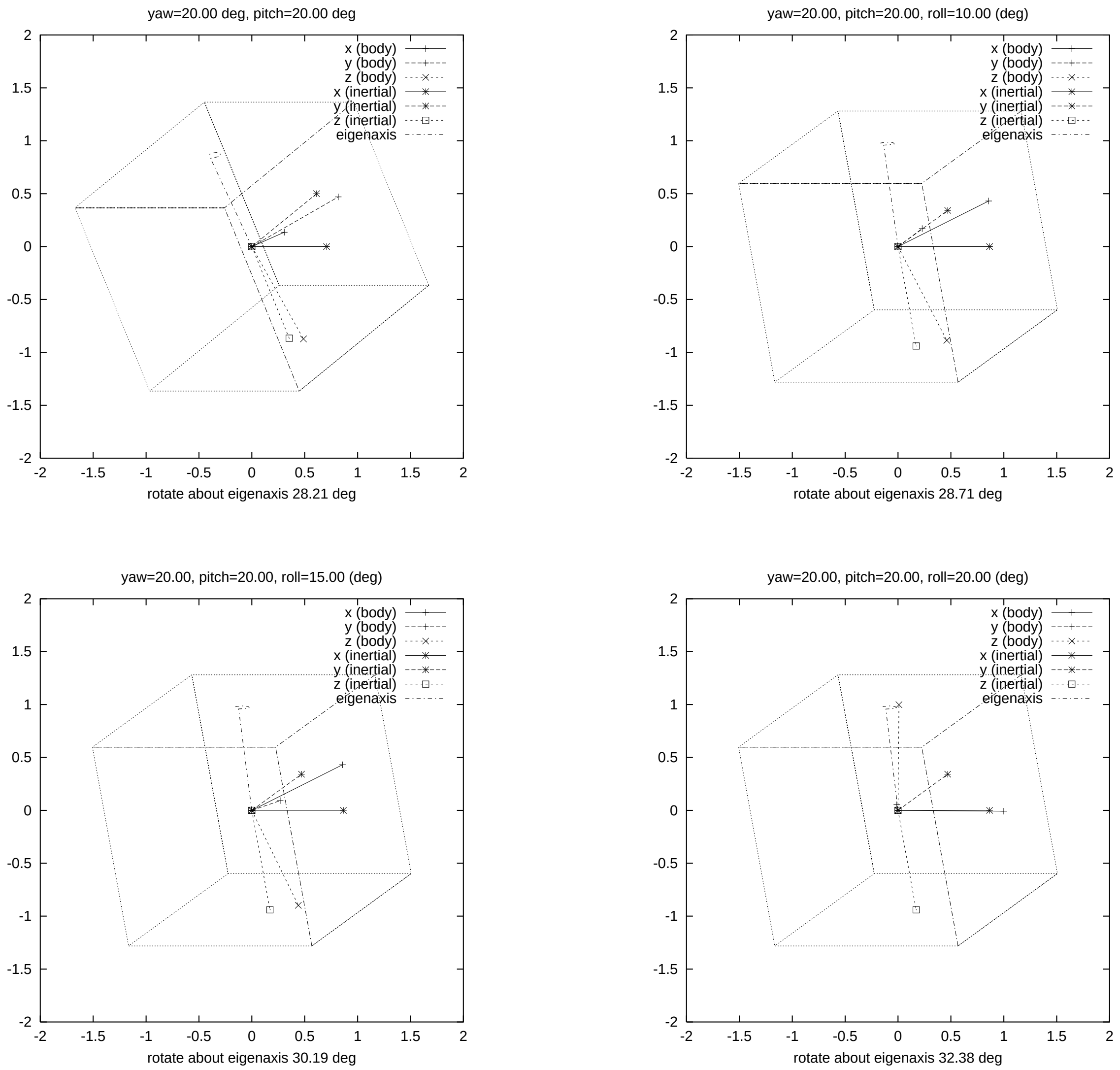

yaw=20.00 deg, pitch= 0.00 deg
x (body)
y (body)
z (body)
x (inertial)
y (inertial)
z (inertial)
eigenaxis
rotate about eigenaxis 20.00 deg


yaw=20.00 deg, pitch= 0.00 deg
x (body)
y (body)
z (body)
x (inertial)
y (inertial)
z (inertial)
eigenaxis
rotate about eigenaxis 20.00 deg


yaw=20.00 deg, pitch= 5.00 deg
x (body)
y (body)
z (body)
x (inertial)
y (inertial)
z (inertial)
eigenaxis
rotate about eigenaxis 20.61 deg

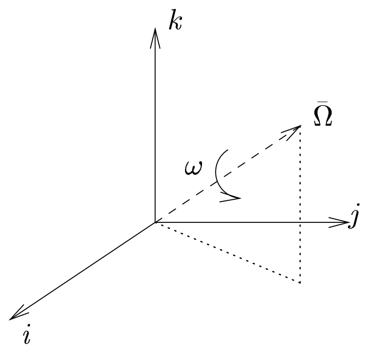

yaw=20.00 deg, pitch=10.00 deg
x (body)
y (body)
z (body)
x (inertial)
y (inertial)
z (inertial)
eigenaxis
rotate about eigenaxis 22.34 deg


yaw=20.00 deg, pitch=15.00 deg
x (body)
y (body)
z (body)
x (inertial)
y (inertial)
z (inertial)
eigenaxis
rotate about eigenaxis 24.95 deg



1.5 Time derivative of a quaternion

In this subsection we adopt the vector notation for quaternions, i.e. (with some abuse of notation) $q = (\bar{v}, \theta) = \bar{v} + \theta$ as in equation (1.3). Let $q_b(t) = (\bar{q}_b(t), \theta_b(t))$ be a quaternion relating the coordinates of a body (rotating) frame to a fixed frame, and suppose that the body frame is rotating with fixed angular velocity ω about the unit vector $\bar{\Omega} = \omega_1 \hat{i} + \omega_2 \hat{j} + \omega_2 \hat{k}$ in the fixed frame. (`qtrans` may be used to perform the deriviation in the body frame.)



Then Euler's approximation of a rotation over Δt seconds is $q_r = \bar{\Omega}\sin(\omega\Delta t/2) + \cos(\omega\Delta t/2)$. For Δt sufficiently small, $\cos(\omega\Delta t/2) \approx 1$ and $\sin(\omega\Delta t/2) \approx \omega\Delta t/2$, so q_r may be approximated as

$$q_r \approx 1 + \left(\frac{\omega\Delta t}{2}\right)\bar{\Omega}$$

Thus

$$\begin{aligned}
q_b(t+\Delta t) &= q_b(t)\left(1 + \left(\frac{\omega\Delta t}{2}\right)\bar{\Omega} + \cdots\right) \\
q_b(t+\Delta t) - q_b(t) &= q_b(t)\left(\frac{\omega\Delta t}{2}\right)\bar{\Omega} + \cdots \\
\frac{q_b(t+\Delta t) - q_b(t)}{\Delta t} &= \left(\frac{\omega}{2}\right) q_b(t)\bar{\Omega} + \cdots
\end{aligned}$$

We take the limit at $\Delta t \to 0$ to obtain

$$\frac{dq_b}{dt} = \left(\frac{\omega}{2}\right) q_b(t)\bar{\Omega} = \frac{\omega}{2}\left(q_{b,4}\bar{\Omega} - (\bar{q}_b \cdot \bar{\Omega}) - \bar{\Omega} \times \bar{q}_b\right)$$

where $x \cdot y = \sum x_\ell y_\ell$ is the vector inner (dot) product and $x \times y = \left\| \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \right\|$ is the vector cross product. In matrix form this is

$$\begin{bmatrix} \dot{q}_{b1} \\ \dot{q}_{b2} \\ \dot{q}_{b3} \\ \dot{q}_{b4} \end{bmatrix} = \frac{\omega}{2} \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} q_{b1} \\ q_{b2} \\ q_{b3} \\ q_{b4} \end{bmatrix}$$

Notice that the 4×4 matrix is skew symmetric (hence its matrix exponential is orthogonal).

References

[Mul] Larry Mullins. Course 4160: Quaternions. Course notes.

[RW91] M. H. Rheinfurth and H. B. Wilson. Methods of applied dynamics. Technical Report NASA RP-1262, NASA, George C. Marshall Space Flight Center, 1991.

Index