Attitude representation

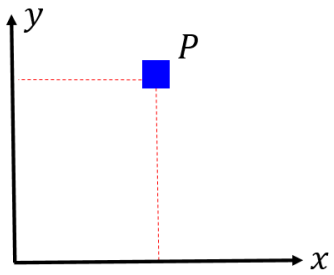
The most common ways to represent attitude are: rotation matrix, Euler angles, and Quaternion

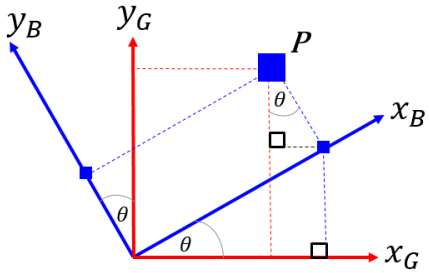
There following **factors** affect the final equations/matrices that represent attitude and perform coordinate transformations:

- Wether the rotation matrix represnt body-fixed frame to world frame transformation or vice versa (the relationship between them is the transpose).
- Wether the rotation matrix active vs assive. A passive rotation matrix takes the space associated with the rotated system and evaluate it wrt to the fixed system (wolrd to body), while an active rotation evaluates the fixed system wrt to the rotated one (body to world).
- Wether rotation sequence is internsic or extrinsic (the relationship is reverse order)
- Wether the angles are positive clockwise or counterclockwise

PASSIVE rotations vs **ACTIVE** rotations

- A **PASSIVE** rotation matrix ${}^{B}R_{G}$, converts the co-ordinates of a point expressed in a fixed **G-frame**, into the co-ordinates of the same point expressed in the new **B-frame**.
- vb = pR3X(psi) * pR2Y(theta) * pR1Z(phi) * vg
- vb = bRg * vg

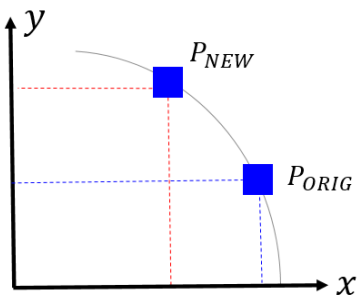


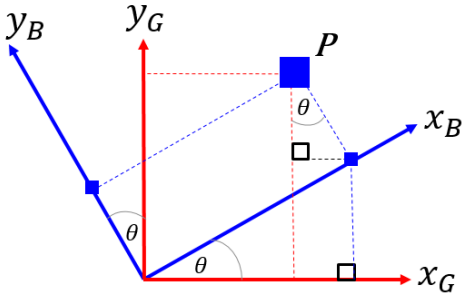


$$\begin{bmatrix} x_P \\ y_P \end{bmatrix}_B = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} x_P \\ y_P \end{bmatrix}_G$$

$${}^{B}P = {}^{B}R_{G} * {}^{G}P$$

- An **ACTIVE** rotation matrix ${}^{G}R_{B}$, allows us to calculate the position of the new point in B-frame relative to the G-frame, ie: ${}^{G}P$.
- vg = inv(pR1Z(phi)) * inv(pR2Y(theta)) * inv(pR3X(psi)) * vb
- vg = gRb * vb





$$\begin{bmatrix} x_P \\ y_P \end{bmatrix}_G = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} x_P \\ y_P \end{bmatrix}_B$$

$${}^{G}\mathrm{P} = {}^{G}\mathrm{R}_{\mathrm{B}} * {}^{B}\mathrm{P}$$

- Passive rotation is related to active rotation by the transpose:

$$ {}^{B}R_{G} = {}^{G}R_{B}^{T} $$

Rotation matrix (Direction Cosine Matrix)

- Uses a 3×3 matrix to represent the linear transform mapping from one coordinate frame to another rotated coordinate frame.

$$ R = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} $$

- Some authors prefer to write the matrix that maps from the body-flxed coordinates to the world coordinates; others prefer the matrix that maps from the world coordinates to the body-flxed coordinates.
- *rotation matrix (here)* encodes the attitude of a rigid body to be the matrix that when pre-multiplied by a vector expressed in the world coordinates yields the same vector expressed in the body-flxed coordinates $\acute{z}$ is in world frame, z is in body frame.

$$ \mathbf{z}' = R\,\mathbf{z} $$
$$ \mathbf{z} = R^T \mathbf{z}'. $$

- Rotation matrices are a complete representation of a 3D orientation, thus there is no singularity in that model.
- A *coordinate rotation* is a rotation about a single coordinate axis. Enumerating the *x*-, *y*-, and *z*-axes with 1,2,and 3, the coordinate rotations (passive) are:

$$R_1(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$R_2(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

$$R_3(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- A rotation matrix may also be referred to as a *direction cosine matrix* , because the elements of this matrix are the cosines of the unsigned angles between the body-flxed axes and the world axes.
- An example of 3 successive PASSIVE rotations
-

Euler angles

Euler angles are widely used because they are easy to understand. The three parameters : Roll, Pitch and Yaw define rotations around the fixed frame's axes :

- Roll (φ) : Rotation around X axis defined between [-π ; π] ;
- Pitch (θ) : Rotation around Y axis defined between [-π/2 ; π/2] ;
- Yaw (ψ) : Rotation around Z axis defined between [-π ; π].

Gimbal lock

Euler angles suffer from a singularity called "Gimbal lock", when Pitch approaches ± π/2,we do not advise to use Euler angles if the device has to be used in a wide range of orientations. Quaternions and rotation matrices do not have any singularity.

Rotation Sequence

Three coordinate rotations in sequence can describe any rotation. The function that maps an Euler angle vector to its corresponding rotation matrix

$$R = X(\alpha)Y(\beta)Z(\gamma)$$

The table below summarizes some of the popular rotation sequences:

1. 1, 2, 3 represent the angles *α*, *β* and *γ*, i.e. the angles corresponding to the first, second and third elemental rotations respectively.
2. *X*, *Y*, *Z* are the matrices representing the elemental rotations about the axes *x*, *y*, *z* of the fixed frame (e.g., *X*1 represents a rotation about *x* by an angle *α*).
3. *s* and *c* represent sine and cosine (e.g., *s*1 represents the sine of *α*).

Proper Euler angles	Tait–Bryan angles
$X_1Z_2X_3 = \begin{bmatrix} c_2 & -c_3s_2 & s_2s_3 \\ c_1s_2 & c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 \\ s_1s_2 & c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$	$X_1Z_2Y_3 = \begin{bmatrix} c_2c_3 & -s_2 & c_2s_3 \\ s_1s_3 + c_1c_3s_2 & c_1c_2 & c_1s_2s_3 - c_3s_1 \\ c_3s_1s_2 - c_1s_3 & c_2s_1 & c_1c_3 + s_1s_2s_3 \end{bmatrix}$
$X_1Y_2X_3 = \begin{bmatrix} c_2 & s_2s_3 & c_3s_2 \\ s_1s_2 & c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 \\ -c_1s_2 & c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$	$X_1Y_2Z_3 = \begin{bmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \\ s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 & c_1c_2 \end{bmatrix}$
$Y_1X_2Y_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & s_1s_2 & c_1s_3 + c_2c_3s_1 \\ s_2s_3 & c_2 & -c_3s_2 \\ -c_3s_1 - c_1c_2s_3 & c_1s_2 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$	$Y_1X_2Z_3 = \begin{bmatrix} c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 & c_2s_1 \\ c_2s_3 & c_2c_3 & -s_2 \\ c_1s_2s_3 - c_3s_1 & c_1c_3s_2 + s_1s_3 & c_1c_2 \end{bmatrix}$
$Y_1Z_2Y_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_1s_2 & c_3s_1 + c_1c_2s_3 \\ c_3s_2 & c_2 & s_2s_3 \\ -c_1s_3 - c_2c_3s_1 & s_1s_2 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$	$Y_1Z_2X_3 = \begin{bmatrix} c_1c_2 & s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 \\ s_2 & c_2c_3 & -c_2s_3 \\ -c_2s_1 & c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 \end{bmatrix}$
$Z_1Y_2Z_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 \\ c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 \\ -c_3s_2 & s_2s_3 & c_2 \end{bmatrix}$	$Z_1Y_2X_3 = \begin{bmatrix} c_1c_2 & c_1s_2s_3 - c_3s_1 & s_1s_3 + c_1c_3s_2 \\ c_2s_1 & c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 \\ -s_2 & c_2s_3 & c_2c_3 \end{bmatrix}$
$Z_1X_2Z_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 & s_1s_2 \\ c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 & -c_1s_2 \\ s_2s_3 & c_3s_2 & c_2 \end{bmatrix}$	$Z_1X_2Y_3 = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & -c_2s_1 & c_1s_3 + c_3s_1s_2 \\ c_3s_1 + c_1s_2s_3 & c_1c_2 & s_1s_3 - c_1c_3s_2 \\ -c_2s_3 & s_2 & c_2c_3 \end{bmatrix}$

Refer to https://en.wikipedia.org/wiki/Euler_angles for more details

Euler angle rates from body rates (pqr)

This depends on the euler angle sequenc used

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \left[\begin{array}{c|c|c} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{array}\right] \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{L}_I^B \dot{\mathbf{\Theta}}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{L}_B^I \boldsymbol{\omega}_B$$

Quaternions

- Quaternions can be represented by Eignaxis representation which has a rotation axis $\vec{E}$ and a rotation angle θ around this axis.
- the relationship between the quaternion and the Eigenaxis parameters $\left\{\theta, \vec{E}\right\}$

$$\begin{bmatrix} q_0 \\ q_X \\ q_Y \\ q_Z \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ E_X.\sin\left(\frac{\theta}{2}\right) \\ E_Y.\sin\left(\frac{\theta}{2}\right) \\ E_Z.\sin\left(\frac{\theta}{2}\right) \end{bmatrix}$$

where:

- $\left(1 = E_X{}^2 + E_Y{}^2 + E_Z{}^2\right)$ and $\left(1 = q_0{}^2 + q_X{}^2 + q_Y{}^2 + q_Z{}^2\right)$

Quaternion inverse

Note: Inverse is the same as conjugate if it is a unit quaternion, hence make sure to nromalize quaternion.

if $q = \begin{bmatrix} q_0 \\ q_X \\ q_Y \\ q_Z \end{bmatrix}$ then $q^{-1} = \begin{bmatrix} q_0 \\ -q_X \\ -q_Y \\ -q_Z \end{bmatrix}$

Quaternion multiplication

$$q \otimes p = Q(q)p = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Quaternion transformation

To transform a vector (i.e. position, velocity,etc.) from a certain coordinate frame to another coordinate frame use the following formula

$$P_B = q \otimes P_A \otimes q^{-1}$$

where q represent the rotation from A to B

To perform the transformation in the opposite direction, inverse the quaternion then apply the same formula above

Quaternion to DCM conversion

- Rotation matrix can be extracted from quaternion. The following rotation matrix represent transformation from the world frame to body-fixed frame.
- Note: this transformation is independent of Euler angle sequences. You could see it differently only if the quaternion is represented differently (i.e. q0 is swapped with q3)

$${}^{B}R_{G} = \begin{pmatrix} {q_0}^2 + {q_X}^2 - {q_Y}^2 - {q_Z}^2 & 2\,q_0\,q_Z + 2\,q_X\,q_Y & 2\,q_X\,q_Z - 2\,q_0\,q_Y \\ 2\,q_X\,q_Y - 2\,q_0\,q_Z & {q_0}^2 - {q_X}^2 + {q_Y}^2 - {q_Z}^2 & 2\,q_0\,q_X + 2\,q_Y\,q_Z \\ 2\,q_0\,q_Y + 2\,q_X\,q_Z & 2\,q_Y\,q_Z - 2\,q_0\,q_X & {q_0}^2 - {q_X}^2 - {q_Y}^2 + {q_Z}^2 \end{pmatrix}$$

The derivatives of the quaternion parameters

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_X} \\ \dot{q_Y} \\ \dot{q_Z} \end{bmatrix} = B \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_B$$

$$B = \begin{pmatrix} -\frac{q_x}{2} & -\frac{q_y}{2} & -\frac{q_z}{2} \\ \frac{q_0}{2} & -\frac{q_z}{2} & \frac{q_y}{2} \\ \frac{q_z}{2} & \frac{q_0}{2} & -\frac{q_x}{2} \\ -\frac{q_y}{2} & \frac{q_x}{2} & \frac{q_0}{2} \end{pmatrix}$$

- Note the B matrix can map the quaternion rates to the angular velocity in body frame. Similar approach can produce the mapping to angular velocity in world frame
- We can also express $\dot{q}$ in an alternate form

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_X} \\ \dot{q_Y} \\ \dot{q_Z} \end{bmatrix} = A \cdot \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix}$$

$$A = \begin{pmatrix} 0 & -\frac{\omega_x}{2} & -\frac{\omega_y}{2} & -\frac{\omega_z}{2} \\ \frac{\omega_x}{2} & 0 & \frac{\omega_z}{2} & -\frac{\omega_y}{2} \\ \frac{\omega_y}{2} & -\frac{\omega_z}{2} & 0 & \frac{\omega_x}{2} \\ \frac{\omega_z}{2} & \frac{\omega_y}{2} & -\frac{\omega_x}{2} & 0 \end{pmatrix}$$

Quaternion to Euler angles

- Quaternions can be converted to Euler angles. Different equations are available depending on the euler angle sequence, for example below is for sequence 1,2,3 (ZYX)

$$\mathbf{q}_{123}(\phi, \theta, \psi) = \begin{bmatrix} c_{\phi/2} c_{\theta/2} c_{\psi/2} + s_{\phi/2} s_{\theta/2} s_{\psi/2} \\ -c_{\phi/2} s_{\theta/2} s_{\psi/2} + c_{\theta/2} c_{\psi/2} s_{\phi/2} \\ c_{\phi/2} c_{\psi/2} s_{\theta/2} + s_{\phi/2} c_{\theta/2} s_{\psi/2} \\ c_{\phi/2} c_{\theta/2} s_{\psi/2} - s_{\phi/2} c_{\psi/2} s_{\theta/2} \end{bmatrix}$$

Derivative of rotation matrix

- Recall that skew symmetric matrix is defined as:

$$\vec{\omega}^{\wedge} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

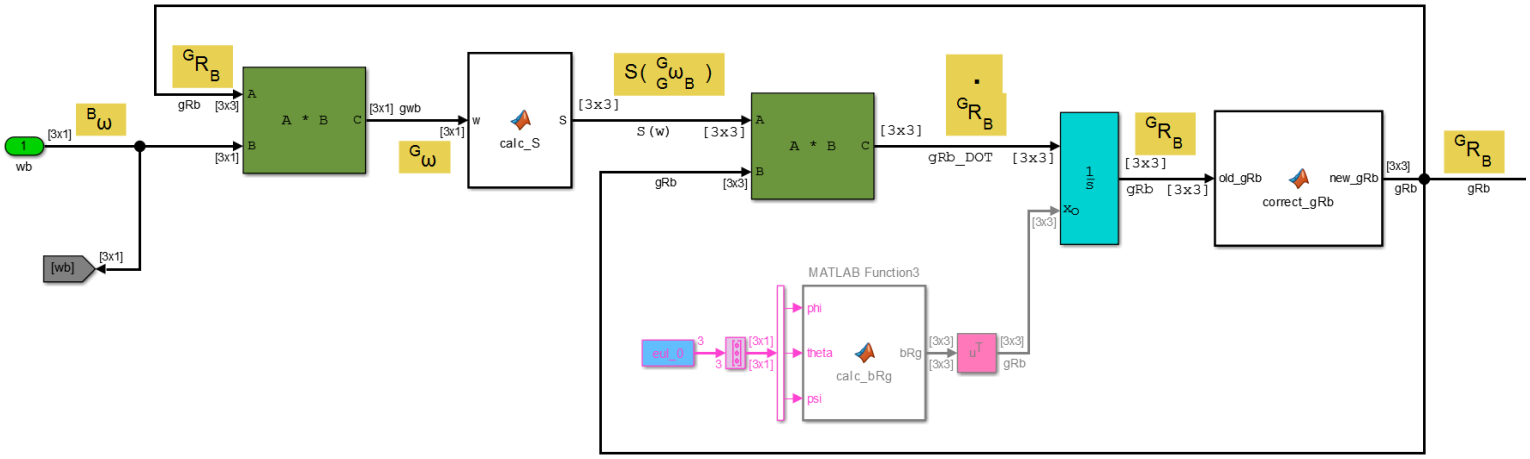
- Also recall that ske symmetric matrix converts vector cross product to matrix multiplication $\hat{A}\boldsymbol{B} = \boldsymbol{A} \times \boldsymbol{B}$ where $A = [x, y, z]^T$

The derivative is related to the angular velocity by:

$$\dot{\mathbf{R}} = \vec{\omega}\hat{}\mathbf{R}$$

The velocity is related to position: ${}^{G}v \;= {}^{G}\omega_{B}\hat{}\,.\,{}^{G}p(t)$

The significance of this expression for ${}^{G}\dot{R}_{B}$, is that we can use it as a mechanism to propagate the orientation of a rigid body. Using this approach there are NO singularities that would prevent us from computing ${}^{G}R_{B}(t)$. The implementation of this algorithm would look something like this:



Note that to compute ${}^{G}R_{B}(t=0)$, we could let the user specify an initial Euler angle configuration. Also, the **`calc_S`** function would be:

```
function S = calc_S(w)
%#codegen

x = w(1);
y = w(2);
z = w(3);

S = [    0,  -z,   y;
         z,   0,  -x;
        -y,   x,   0;
    ];
```

When we integrate ${}^{G}\dot{R}_{B}$, we would expect that the orthogonality relationship of ${}^{G}R_{B}(t)\;.\,{}^{G}R_{B}(t)^{T} = I$ would still be true. And it almost is ! The combination of "numerically" intergrating a derivative and the finite word size of computer calculations, means that we end up with a relationship that is more like this:

$${}^{G}R_{B}(t)\;.\,{}^{G}R_{B}(t)^{T} = I + \delta e \;\text{,where}\quad \delta e \quad \text{is a matrix of small terms}$$

We can reduce this δe term using a correction technique such as the one proposed by William Premerlani and Paul Bizard (https://wiki.paparazziuav.org/w/images/e/e5/DCMDraft2.pdf).

```
function new_gRb = correct_gRb(old_gRb)
%#codegen

% The correction technique implementde here is the one described by
% William Premerlani and Paul Bizard:
%   see:  https://gentlenav.googlecode.com/files/DCMDraft2.pdf

row_1   = old_gRb(1,:);
row_2   = old_gRb(2,:);

% look at the error associted with row_1 and row_2 they are supposed to be
% orthogonal, ie: their DOT product should be ZERO
e       = sum( row_1 .* row_2);

% distrubute the error across X and Y
new_row_1 = row_1  - (e/2)*row_2;
new_row_2 = row_2  - (e/2)*row_1;

% compute Z
new_row_3 = cross( new_row_1, new_row_2 );

% normlaize all rows so that their MAG is UNITY
%:  NOTE:  norm([3 4]) is 5
new_row_1 = new_row_1 / norm(new_row_1);
new_row_2 = new_row_2 / norm(new_row_2);
new_row_3 = new_row_3 / norm(new_row_3);

% assemble the NEW gRb matrix
new_gRb = [new_row_1; new_row_2; new_row_3];

end
```

Attitude representation using Euler angles

```
syms t
syms phi(t) theta(t) psi(t) p(t) q(t) r(t) % roll pitch yaw p q r
syms phi_dot(t)  theta_dot(t)  psi_dot(t) % euler rates
syms p_dot(t) q_dot(t) r_dot(t) % body rates
```

Let's use the Z - X - Y Euler angle convention

```
R1_z = [
      cos(psi),  -sin(psi),   0;
     sin(psi),  cos(psi),   0;
          0   ,     0    ,   1;

    ];
R2_x = [
         1   ,       0     ,     0      ;
         0   ,  cos(phi)   ,  -sin(phi);
         0   ,  sin(phi)  , cos(phi);
    ];
R3_y = [
         cos(theta) ,   0     , sin(theta);
            0       ,   1     ,     0    ;
        -sin(theta)  ,   0     , cos(theta);
```

```
    ];
```

Body to world(inertial) frame rotation

```
wRb = R1_z*R2_x*R3_y % body to world
```

wRb(t) =

$$\begin{pmatrix} \cos(\psi(t))\cos(\theta(t)) - \sin(\phi(t))\sin(\psi(t))\sin(\theta(t)) & -\cos(\phi(t))\sin(\psi(t)) & \cos(\psi(t))\sin(\theta(t)) + \cos(\theta(\\ \cos(\theta(t))\sin(\psi(t)) + \cos(\psi(t))\sin(\phi(t))\sin(\theta(t)) & \cos(\phi(t))\cos(\psi(t)) & \sin(\psi(t))\sin(\theta(t)) - \cos(\psi(\\ -\cos(\phi(t))\sin(\theta(t)) & \sin(\phi(t)) & \cos(\phi(t))\,\mathrm{co} \end{pmatrix}$$

```
bRw=transpose(wRb) % world to body
```

bRw(t) =

$$\begin{pmatrix} \cos(\psi(t))\cos(\theta(t)) - \sin(\phi(t))\sin(\psi(t))\sin(\theta(t)) & \cos(\theta(t))\sin(\psi(t)) + \cos(\psi(t))\sin(\phi(t))\sin(\theta(t)) \\ -\cos(\phi(t))\sin(\psi(t)) & \cos(\phi(t))\cos(\psi(t)) \\ \cos(\psi(t))\sin(\theta(t)) + \cos(\theta(t))\sin(\phi(t))\sin(\psi(t)) & \sin(\psi(t))\sin(\theta(t)) - \cos(\psi(t))\cos(\theta(t))\sin(\phi(t)) \end{pmatrix}$$

Orthogonality

rotation matrix is orthogonal

$$R^T(t)R(t) = R(t)R^T(t) = 1$$

Position & Velocity

```
syms R(t) R_t(t) % R and R transpose
```

Let's differentiate both sides of $R^T(t)R(t) = R(t)R^T(t) = 1$

```
exp = diff(R_t*R ==1,t) == diff(R*R_t ==1,t)
```

exp(t) =

$$\left(R(t)\frac{\partial}{\partial t}R_t(t) + R_t(t)\frac{\partial}{\partial t}R(t) = 0\right) = \left(R(t)\frac{\partial}{\partial t}R_t(t) + R_t(t)\frac{\partial}{\partial t}R(t) = 0\right)$$

Position

Let's write the position equation as

$$q(t) = R(t)p$$

where p is position in fixed-body frame, q is position in inertial frame and R body to inertial transform

```
syms q_w(t)  p_b
position = q_w==R*p_b
```

position(t) = $q_w(t) = p_b\,R(t)$

Velocity

If we differentiate, the result is

$$\dot{q} = \dot{R}p$$

velocity in inertial frame — *position in body-fixed frame*

```
diff(position,t)
```

```
ans(t) =
```

$$\frac{\partial}{\partial t} q_w(t) = p_b \frac{\partial}{\partial t} R(t)$$

Pre-multiply by rotation matrix transpose R^T results in

$$R^T \dot{q} = R^T \dot{R} p \qquad \hat{\omega}^b$$

velocity in body-fixed frame — *encodes angular velocity in body-fixed frame*

which relates body angular rates (ω_b) to body-fixed velocity (transitional).

$$\omega_b = R^T(t)\, \dot{R}(t)$$

If we use $p = R^T q$, we get

$$\dot{q} = \dot{R} R^T q \qquad \hat{\omega}^s$$

velocity in inertial frame — *encodes angular velocity in inertial frame*

which relates inertial angular rates (ω_s) to inertial velocity (transitional).

$$\omega_s = \dot{R}(t) R^T(t)$$

Examples

Example 1: angular velocity for rotation around Z-axis

```
R = R1_z
```

R(t) =

$$\begin{pmatrix} \cos(\psi(t)) & -\sin(\psi(t)) & 0 \\ \sin(\psi(t)) & \cos(\psi(t)) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
R_dot = diff(R,t)
```

R_dot(t) =

$$\begin{pmatrix} -\sin(\psi(t))\frac{\partial}{\partial t}\psi(t) & -\cos(\psi(t))\frac{\partial}{\partial t}\psi(t) & 0 \\ \cos(\psi(t))\frac{\partial}{\partial t}\psi(t) & -\sin(\psi(t))\frac{\partial}{\partial t}\psi(t) & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

```
omega_b = simplify(R.'*R_dot)
```

omega_b(t) =

$$\begin{pmatrix} 0 & -\frac{\partial}{\partial t}\psi(t) & 0 \\ \frac{\partial}{\partial t}\psi(t) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

```
omega_w = simplify(R_dot*R.')
```

omega_w(t) =

$$\begin{pmatrix} 0 & -\frac{\partial}{\partial t}\psi(t) & 0 \\ \frac{\partial}{\partial t}\psi(t) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Which results in a angular velocity around z-axis only ($\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$

Example 2: angular velocity for rotation sequence Z-X

```
R = R1_z*R2_x
```

R(t) =

$$\begin{pmatrix} \cos(\psi(t)) & -\cos(\phi(t))\sin(\psi(t)) & \sin(\phi(t))\sin(\psi(t)) \\ \sin(\psi(t)) & \cos(\phi(t))\cos(\psi(t)) & -\cos(\psi(t))\sin(\phi(t)) \\ 0 & \sin(\phi(t)) & \cos(\phi(t)) \end{pmatrix}$$

```
R_dot = simplify(diff(R,t));
omega_b = simplify(R.'*R_dot)
```

omega_b(t) =

$$\begin{pmatrix} 0 & -\cos(\phi(t))\,\frac{\partial}{\partial t}\,\psi(t) & \sin(\phi(t))\,\frac{\partial}{\partial t}\,\psi(t) \\ \cos(\phi(t))\,\frac{\partial}{\partial t}\,\psi(t) & 0 & -\frac{\partial}{\partial t}\,\phi(t) \\ -\sin(\phi(t))\,\frac{\partial}{\partial t}\,\psi(t) & \frac{\partial}{\partial t}\,\phi(t) & 0 \end{pmatrix}$$

Another manual way using $\omega_b = (R_z R_x)^T * \left(\dot{R}_z R_x + R_z \dot{R}_x\right)$

```
% another way to calculate it manually
w_b2 = simplify( (R1_z*R2_x).'*(diff(R1_z,t)*R2_x + R1_z*diff(R2_x,t)) )
```

```
w_b2(t) =
```

$$\begin{pmatrix} 0 & -\cos(\phi(t))\,\frac{\partial}{\partial t}\,\psi(t) & \sin(\phi(t))\,\frac{\partial}{\partial t}\,\psi(t) \\ \cos(\phi(t))\,\frac{\partial}{\partial t}\,\psi(t) & 0 & -\frac{\partial}{\partial t}\,\phi(t) \\ -\sin(\phi(t))\,\frac{\partial}{\partial t}\,\psi(t) & \frac{\partial}{\partial t}\,\phi(t) & 0 \end{pmatrix}$$

```
omega_w = simplify(R_dot*R.')
```

```
omega_w(t) =
```

$$\begin{pmatrix} 0 & -\frac{\partial}{\partial t}\,\psi(t) & \sin(\psi(t))\,\frac{\partial}{\partial t}\,\phi(t) \\ \frac{\partial}{\partial t}\,\psi(t) & 0 & -\cos(\psi(t))\,\frac{\partial}{\partial t}\,\phi(t) \\ -\sin(\psi(t))\,\frac{\partial}{\partial t}\,\phi(t) & \cos(\psi(t))\,\frac{\partial}{\partial t}\,\phi(t) & 0 \end{pmatrix}$$

Example 3: angular velocity for rotation sequence Z-X-Y

```
R = R1_z*R2_x*R3_y
```

```
R(t) =
```

$$\begin{pmatrix} \cos(\psi(t))\cos(\theta(t)) - \sin(\phi(t))\sin(\psi(t))\sin(\theta(t)) & -\cos(\phi(t))\sin(\psi(t)) & \cos(\psi(t))\sin(\theta(t)) + \cos(\theta(\\ \cos(\theta(t))\sin(\psi(t)) + \cos(\psi(t))\sin(\phi(t))\sin(\theta(t)) & \cos(\phi(t))\cos(\psi(t)) & \sin(\psi(t))\sin(\theta(t)) - \cos(\psi(\\ -\cos(\phi(t))\sin(\theta(t)) & \sin(\phi(t)) & \cos(\phi(t))\,\mathrm{co} \end{pmatrix}$$

```
R_dot = simplify(diff(R,t));
omega_b = simplify(R.'*R_dot);
omega_b = omega_b(t)
```

```
omega_b =
```

$$\begin{pmatrix} \\ \\ \\ (\sin(\psi(t))\sin(\theta(t)) - \cos(\psi(t))\cos(\theta(t))\sin(\phi(t))) & \left(\cos(\psi(t))\cos(\theta(t))\frac{\partial}{\partial t}\psi(t) - \sin(\psi(t))\sin(\theta(t))\right. \end{pmatrix}$$

Another manual way using $\omega_b = (R_z R_x R_y)^T * \left(\dot{R}_z R_x R_y + R_z \dot{R}_x R_y + R_z R_x \dot{R}_y\right)$

```
% another way to calculate it manually
w_b2 = simplify( (R1_z*R2_x*R3_y).'*(diff(R1_z,t)*R2_x*R3_y + R1_z*diff(R2_x,t)*R3_y + R1_z*R2_
```

w_b2(t) =

$$\begin{pmatrix} \\ \\ \\ (\sin(\psi(t))\sin(\theta(t)) - \cos(\psi(t))\cos(\theta(t))\sin(\phi(t))) & \left(\cos(\psi(t))\cos(\theta(t))\frac{\partial}{\partial t}\psi(t) - \sin(\psi(t))\sin(\theta(t))\right. \end{pmatrix}$$

```
omega_w = simplify(R_dot*R.');
omega_w = omega_w(t)
```

omega_w =

$$\begin{pmatrix} \\ (\cos(\psi(t))\sin(\theta(t)) + \cos(\theta(t))\sin(\phi(t))\sin(\psi(t))) & \left(\cos(\psi(t))\sin(\theta(t))\frac{\partial}{\partial t}\psi(t) + \cos(\theta(t))\sin(\psi(t))\right. \\ \\ \end{pmatrix}$$

TO confirm, substitute a zero rotation around Y, we should get the same as Z-X angular velocity

```
simplify(subs(omega_b,[theta(t)],[0]))
```

ans =

$$\begin{pmatrix} 0 & -\cos(\phi(t))\frac{\partial}{\partial t}\psi(t) & \sin(\phi(t))\frac{\partial}{\partial t}\psi(t) \\ \cos(\phi(t))\frac{\partial}{\partial t}\psi(t) & 0 & -\frac{\partial}{\partial t}\phi(t) \\ -\sin(\phi(t))\frac{\partial}{\partial t}\psi(t) & \frac{\partial}{\partial t}\phi(t) & 0 \end{pmatrix}$$

Euler rates to body rates conversion matrix

Extract the vector from the skew-symmetric matrix

```
w_x = simplify(omega_b(3,2));
w_y = simplify(omega_b(1,3));
w_z = simplify(omega_b(2,1));
wb = [w_x;w_y;w_z];
```

Substitute with shorter symbols

```
syms phi_t theta_t psi_t phi_dot_t theta_dot_t psi_dot_t
cur_labels = [phi(t), theta(t), psi(t),diff(phi(t), t), diff(theta(t), t),diff(psi(t), t) ];
new_labels = [phi_t, theta_t, psi_t, phi_dot_t, theta_dot_t, psi_dot_t];
wb_new = subs(wb,cur_labels,new_labels);
% simplify the expressions
wb_new = simplify(wb_new)
```

```
wb_new =
```

$$\begin{pmatrix} \dot{\phi}_t \cos(\theta_t) - \dot{\psi}_t \cos(\phi_t) \sin(\theta_t) \\ \dot{\theta}_t + \dot{\psi}_t \sin(\phi_t) \\ \dot{\phi}_t \sin(\theta_t) + \dot{\psi}_t \cos(\phi_t) \cos(\theta_t) \end{pmatrix}$$

References

- Direction Cosine Matrix IMU: Theory William Premerlani and Paul Bizard: https://wiki.paparazziuav.org/w/images/e/e5/DCMDraft2.pdf
- Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors, James Diebel (2006)

- `Bradley Horton : 01-Mar-2016, bradley.horton@mathworks.com.au`