Explore **PASSIVE** rotations and **EULER rates**

In this task we're going to look at how the EULER rates of a rigid body can be determined from the BODY rates of the rigid body. We'll see that there are certain angular poses that result in a matrix singularity which in turn prevents us from transforming from body rates to Euler rates. This task demonstrates how PASSIVE rotation matrices can be applied. The key learning outcomes from this sesion will allow you to answer the following questions:

- What is a PASSIVE rotation matrix ?
- How do i construct a Direction Cosine Matrix (DCM) from a given rotation sequence ?
- What's the relationship between body rates and Euler rates (of a given sequence) ?
- Understand the presense of singularity poses for a given sequense.

Why are we doing this ?

- Before we can create a 6-DOF model of a vehicle (eg: a quadcopter), we need to get comfortable with certain concepts. Concepts such as PASSIVE rotation matrices and Euler rates.

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

We know that rotations don't commute !

In the movie below BOTH rotations involve a YAW, PITCH and ROLL of 90,30 60 degrees. But the one on the left does the sequence in the order of YPR .... while the one on the right does it in reverse, ie: in the order of RPY

```
disp('<a href="matlab:bh_play_movie">CLICK ME to PLAY a MOVIE</a>')
```

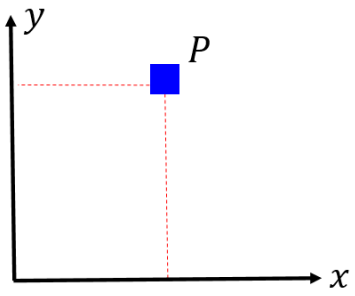
CLICK ME to PLAY a MOVIE

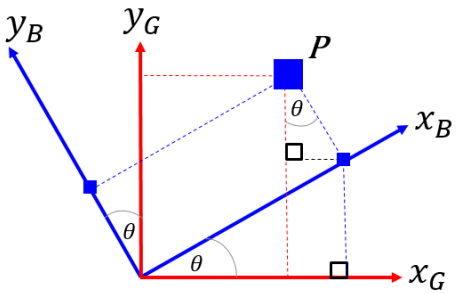
Review the concept of PASSIVE rotation matrices :

Consider the following scenario:

- We have a data point P.
- We have a fixed frame called the **G-fame.**
- We know the *(x,y)* co-ordinates of the point P in this **G-fame** and refer to this as ${}^{G}P$.
- We then rotate the **B-frame** by θ relative to the fixed **G-frame**.

We now want to know what the co-ordinate of the point P is relative to this new **B-frame,** ie: what is ${}^{B}P$? This scenario is shown in the figure below:

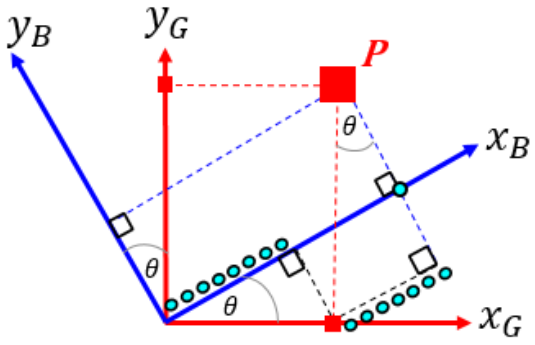


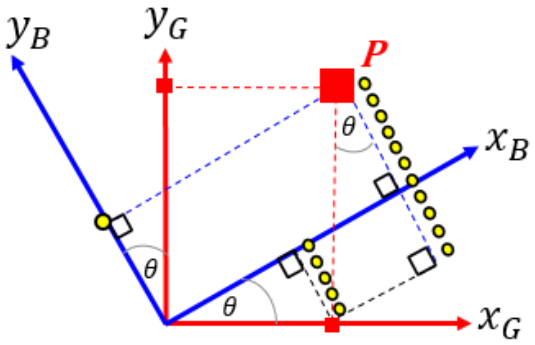


$$\begin{bmatrix} x_P \\ y_P \end{bmatrix}_B = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} x_P \\ y_P \end{bmatrix}_G$$

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

A **PASSIVE** rotation matrix, 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**.





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

An example of 3 successive PASSIVE rotations

Say we have a fixed G-frame. We start by having our B-frame co-incident with G, and then we start to rotate the B-frame. Specifically, we're going to apply 3 LOCAL axes rotations which will result in a newly orientated B-frame. Assume that we apply these 3 successive rotations in the following order:

1. R1Z occurs 1st about the LOCAL **Z** body axis (ϕ), aka **YAW**
2. R2Y occurs 2nd about the LOCAL **Y** body axis (θ), aka **PITCH**
3. R3X occurs 3rd about the LOCAL **X** body axis (ψ), aka **ROLL**

We can express a vector defined in the G axis to it's corresponding description in the B axis, using a sequence of **PASSIVE** rotation matrices, ie:

$$^{B}\mathbf{v} = \mathbf{R3X}(\psi_x) \times \mathbf{R2Y}(\theta_y) \times \mathbf{R1Z}(\phi_z) \times {}^{G}\mathbf{v}$$

OR, in a more compact form as:

$$^{B}\mathbf{v} = {}^{B}\mathbf{R}_{G} \times {}^{G}\mathbf{v}$$

Create a passive rotation object

```
syms phi theta psi
OBJ_P = bh_rot_passive_G2B_CLS({'D1Z', 'D2Y', 'D3X'}, [phi, theta, psi], 'SYM')
```

```
OBJ_P =
  bh_rot_passive_G2B_CLS with properties:

        ang_units: SYM
    num_rotations: 3
          dir_1st: D1Z
          dir_2nd: D2Y
          dir_3rd: D3X
          ang_1st: [1x1 sym]
          ang_2nd: [1x1 sym]
          ang_3rd: [1x1 sym]
```

Here are the PASSIVE rotation matrices:

Here's $\mathbf{R1Z}(\boldsymbol{\phi}_z)$:

```
R1Z = OBJ_P.get_R1
```

```
R1Z =
```

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

Here's $\mathbf{R2Y}(\boldsymbol{\theta}_y)$:

```
R2Y = OBJ_P.get_R2
```

```
R2Y =
```

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

Here's $\mathbf{R3X}(\boldsymbol{\psi}_x)$

```
R3X = OBJ_P.get_R3
```

```
R3X =
```

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

Calculate the Direction Cosine Matrix ${}^{B}R_{G}$

Recall we earlier said: ${}^{B}v = {}^{B}R_{G} * {}^{G}v$

```
bRg = R3X * R2Y * R1Z
```

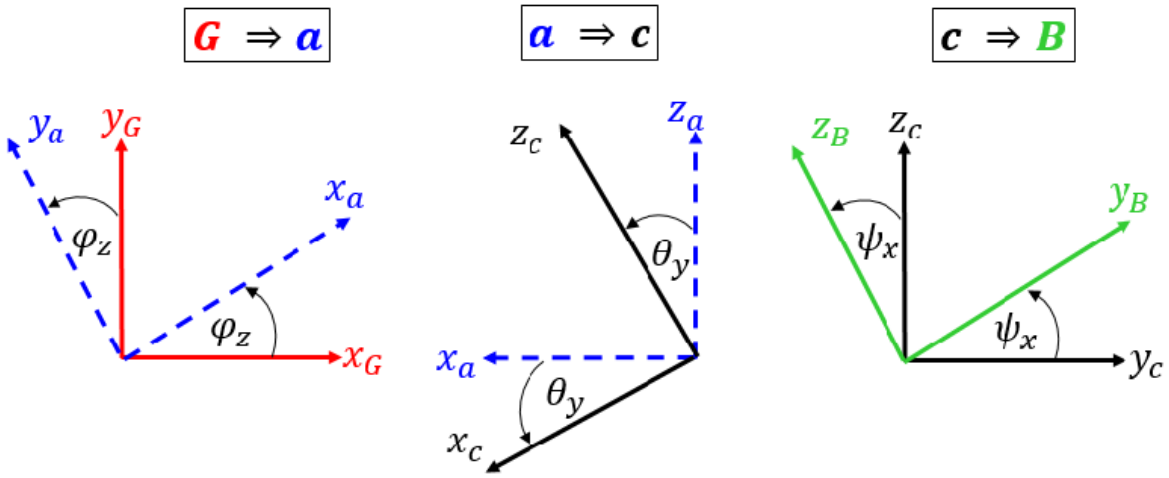
bRg =

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

Explore the relationship between BODY rates and EULER rates

As we apply these local frame rotations of (ϕ, θ, ψ), we can represent the **angular rates** $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ of the rotating frames into the LOCAL frame co-ordinates. These local frame co-ordinates can then be converted into co-ordinates expressed in the final B frame. For example, during each of the local axes rotations we can think of there being a START frame and an END frame:

Rotation	START frame	End frame	Angular rate vector Associated with Rotation
R1Z(φ)	G-frame	a-frame	$\begin{pmatrix} 0 \\ 0 \\ \varphi\_DOT \end{pmatrix}$ in G
R2Y(θ)	a-frame	c-frame	$\begin{pmatrix} 0 \\ \theta\_DOT \\ 0 \end{pmatrix}$ in a
R3X(ψ)	c-frame	B-frame (the body frame)	$\begin{pmatrix} \psi\_DOT \\ 0 \\ 0 \end{pmatrix}$ in c



We can express each of the local frame angular velocities into their corresponding components in the final B frame - and we'll use PASSIVE rotation matrices to do this:

```
syms phi_dot  theta_dot  psi_dot

aRg = R1Z;
cRa = R2Y;
bRc = R3X;
```

Here's the portion of the BODY frame rate associated with our $\dot{\phi}$ rate:

```
wb_part_1 = bRc * cRa * aRg * [0;0;phi_dot]   % convert local G into B
```

```
wb_part_1 =
```

$$\begin{pmatrix} -\varphi_{dot}\sin(\theta) \\ \varphi_{dot}\cos(\theta)\sin(\psi) \\ \varphi_{dot}\cos(\psi)\cos(\theta) \end{pmatrix}$$

Here's the portion of the BODY frame rate associated with our $\dot{\theta}$ rate:

```
wb_part_2 = bRc * cRa *       [0;theta_dot;0] % convert local a into B
```

```
wb_part_2 =
```

$$\begin{pmatrix} 0 \\ \theta_{dot}\cos(\psi) \\ -\theta_{dot}\sin(\psi) \end{pmatrix}$$

Here's the portion of the BODY frame rate associated with our $\dot{\psi}$ rate:

```
wb_part_3 = bRc *             [psi_dot;0;0]   % convert local c into B
```

```
wb_part_3 =
```

$$\begin{pmatrix} \psi_{dot} \\ 0 \\ 0 \end{pmatrix}$$

The total angular velocity expressed in the BODY B frame is therefore

We can now construct the total angular velocity vector expressed in components of the final B frame.

$$^{B}_{G}\omega_{b} \equiv \omega_{b} = f(\dot{\phi}, \dot{\theta}, \dot{\psi}, \phi, \theta, \psi)$$

```
wb = wb_part_1 + wb_part_2 + wb_part_3
```

wb =

$$\begin{pmatrix} \psi_{dot} - \varphi_{dot}\sin(\theta) \\ \theta_{dot}\cos(\psi) + \varphi_{dot}\cos(\theta)\sin(\psi) \\ \varphi_{dot}\cos(\psi)\cos(\theta) - \theta_{dot}\sin(\psi) \end{pmatrix}$$

We can write the angular velocity vector ω_b as a MATRIX equation

Let's say that: $\omega_{b} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}$, we can write a matrix equation of the form **A.x = b** that describes the relationship between the body rates ω_b and the Euler rates: $A \times \begin{pmatrix} \phi_{dot} \\ \theta_{dot} \\ \psi_{dot} \end{pmatrix} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}$

```
syms p q r

x = [phi_dot, theta_dot, psi_dot].';

[A,b] = equationsToMatrix(  wb(1)==p, ...
                            wb(2)==q, ...
                            wb(3)==r, ...
                            x);
```

Just to clarify this, let me put each of the 3 elements side by side as [**A, x, b] ,** where:

$$\begin{matrix} A & \times & x & = & b \\ \downarrow & & \downarrow & & \downarrow \end{matrix}$$

$$A \times \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} {}^{B}\omega_{x} \\ {}^{B}\omega_{y} \\ {}^{B}\omega_{z} \end{pmatrix}$$

```
[A, x, b]
```

ans =

$$\begin{pmatrix} -\sin(\theta) & 0 & 1 & \varphi_{dot} & p \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 & \theta_{dot} & q \\ \cos(\psi)\cos(\theta) & -\sin(\psi) & 0 & \psi_{dot} & r \end{pmatrix}$$

ATTENTION: The SINGULARITY between BODY rates and EULER rates

From the Matrix equation computed above there is actually an angle that causes the determinant of **A** to be ZERO, and hence prevents us from solving for the Euler rates (at that angle) iff we know the body rates ω_b. The angle that causes this problem is the rotation about the local Y axis, ie: the angle θ.

Specifically it is when $\theta = 90^o$. We can see this by first computing the determinant of *A* .

```
det_A = simplify( det(A) )
```

det_A = $-\cos(\theta)$

And then solving for its roots.

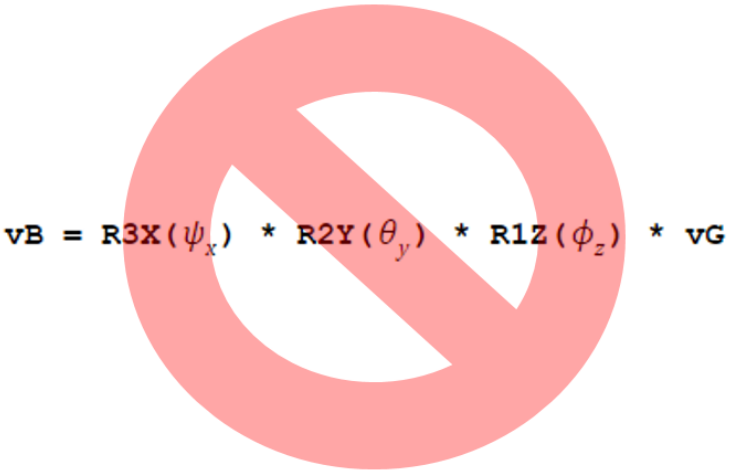
```
[bad_theta, param_s, cond_s] = solve( det_A ==0, theta, 'ReturnConditions', true )
```

bad_theta = $\frac{\pi}{2} + \pi k$

param_s = k

cond_s = $k \in \mathbb{Z}$

So this tells us that as soon as our vehicle has a pitch angle of $\theta = 90^o$, that our chosen Euler angle sequence simply canNOT be used to convert FROM body rates ω_b TO Euler rates. So? So, if you think your vehicle will pitch by $\theta = 90^o$ ... ***AND you're wanting to calculate EULER rates from body rates*** .... then you'll need to consider an alternate form of describing your vehicle's pose (eg: quaternions, or integrating directly the DCM)



Take a moment !

In the previous section we used the function **`det()`** to calculate the determinant of a matrix. .... but what if we didn't know that **`det()`** was the function to use ?

Well, the MATLAB HELP browser is an incredible resource to use. try this simple text search

```
doc determinant of a matrix
```

Let's compute Euler rates from our body rates ω_b

Assuming our vehicle does NOT have a pitch angle of $\theta = 90^o$, then we can use the results of the previous section to calculate the Euler rates from our body rates ω_b.

$$\begin{pmatrix} \phi_{dot} \\ \theta_{dot} \\ \psi_{dot} \end{pmatrix} = A^{-1} * \omega_b \text{ where } \omega_b = \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

```
euler_rates = A \ [p; q; r];
euler_rates = simplify(euler_rates)
```

```
euler_rates =
```

$$\begin{pmatrix} \frac{r\cos(\psi)+q\sin(\psi)}{\cos(\theta)} \\ q\cos(\psi)-r\sin(\psi) \\ \frac{p\cos(\theta)+r\cos(\psi)\sin(\theta)+q\sin(\psi)\sin(\theta)}{\cos(\theta)} \end{pmatrix}$$

Create a REusable function from this expresion:

We can automatically convert these derived "symbolic" expressions into either MATLAB functions OR Simulink blocks(as a MATLAB Function block). So let's look at the following specific example:

- our relationship for converting FROM our body rates ${}^{B}\omega$ to our EULER rates

```
FILENAME        = 'bh_autogen_euler_rates';
INPUT_VAR_ORDER = {'theta', 'psi', 'p', 'q', 'r'};
```

Here is a MATLAB function:

```
matlabFunction(euler_rates,'File',FILENAME, ...
               'Optimize',false, 'Vars', INPUT_VAR_ORDER);

% look at the autogenerated file
dbtype(FILENAME)
```

```
1     function euler_rates = bh_autogen_euler_rates(theta,psi,p,q,r)
2     %BH_AUTOGEN_EULER_RATES
3     %    EULER_RATES = BH_AUTOGEN_EULER_RATES(THETA,PSI,P,Q,R)
4     
5     %    This function was generated by the Symbolic Math Toolbox version 7.0.
6     %    21-Jul-2016 09:23:20
7     
8     euler_rates = [(r.*cos(psi)+q.*sin(psi))./cos(theta);q.*cos(psi)-r.*sin(psi);(p.*cos(theta)+r.*cos
```

Here is the Simulink Block:

```
FILENAME        = 'SIM_bh_autogen_bRg';
BLOCK_NAME      = [FILENAME,'/my_euler_rates'];
new_system(FILENAME)
open_system(FILENAME)

matlabFunctionBlock(BLOCK_NAME, euler_rates, ...
                    'Optimize',false, 'Vars', INPUT_VAR_ORDER);
```

```
Evaluating callback 'PostLoadFcn' for simulink
Callback: setsysloc_simulink(bdroot)
Evaluating callback 'LoadFcn' for simulink/Sources/Waveform Generator
Callback: set_param(gcb,'LoadFlag','1');


Evaluating callback 'LoadFcn' for simulink/Sources/Signal Builder
Callback: sigbuilder_block('load');
Evaluating callback 'LoadFcn' for simulink/Sinks/XY Graph
Callback: sfunxy([],[],[],'LoadBlock')
Evaluating callback 'LoadFcn' for simulink/Model-Wide Utilities/Model Info
Callback: slcm LoadBlock;
Evaluating callback 'LoadFcn' for simulink/Math Operations/Slider Gain
```

Callback: slideg Load

```
BLOCK_NAME      = [FILENAME,'/bRg'];
INPUT_VAR_ORDER = {'phi','theta','psi'};
matlabFunctionBlock(BLOCK_NAME, bRg, ...
                    'Optimize',false, 'Vars', INPUT_VAR_ORDER);
```