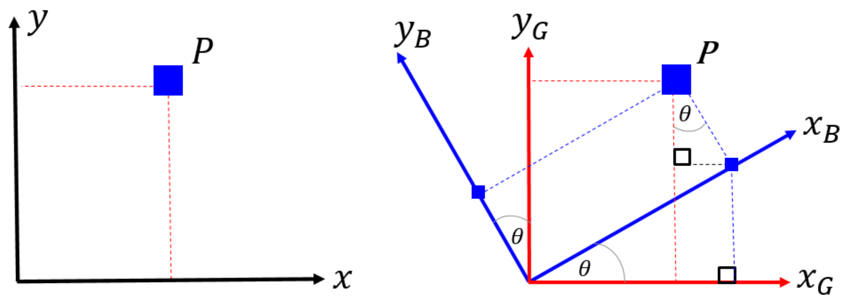
Explore **PASSIVE** rotations which Transform a G vec into a B vec

In this tutorial, we're going to explore the concept of **PASSIVE** rotation matrices.



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

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

Why are we doing this ?

- Rotation matrices are used heavily in Mechanical, Robotic and Aeronautical engineering applications.
- Often students can get confused when they read the term "Rotation matrix". In many/most cases, this confusion can be reduced by emphasizing a rotation matrix as being either PASSIVE or ACTIVE.

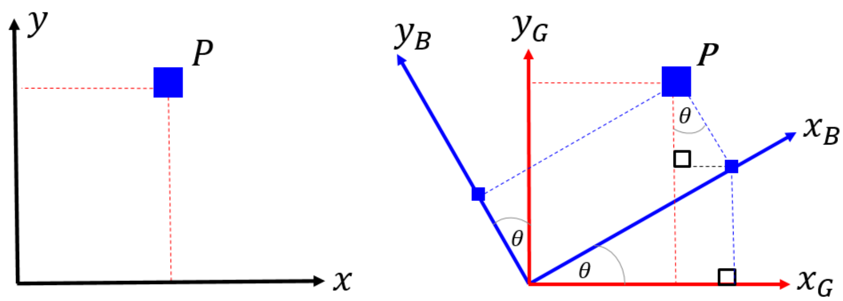
`Bradley Horton : 01-Mar-2016, bradley.horton@mathworks.com.au`

Introduction:

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** relative to the fixed **G-frame**.

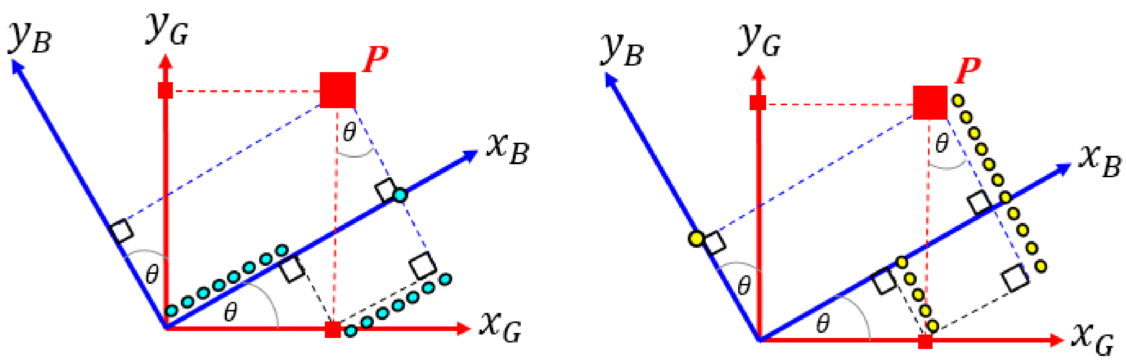
We now want to know what the co-ordinate of the point P is relative to this new **B-frame,** ie: what is ${}^{B}P$? This scenario is shown in the figure below:



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

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

A **PASSIVE** rotation matrix ${}^{B}R_{G}$, 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**.



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

A concrete example - part 1:

Consider the specific case of ${}^{G}P = \begin{pmatrix} 3 \\ 1 \\ 0 \end{pmatrix}$ and a B-frame rotated by 60 degrees relative to Z axis of the G-frame

```
gP    = [3,1,0]';
alpha = 60*pi/180;

bRg   = [   cos(alpha), sin(alpha),   0;
           -sin(alpha), cos(alpha),   0;
                     0,          0,   1];
```

So now apply the passive rotation matrix and calculate ${}^{B}P$

```
bP = bRg * gP
```

```
bP =
          2.36602540378444
         -2.09807621135332
                         0
```

A concrete example - part 2:

We can implement the formula for this passive rotation matrix ${}^{B}R_{G}$ into a MATLAB class called <**`bh_rot_passive_G2B_CLS`**>. This will allow us to reuse the formula over and over again. So repeating the previous example we have:

```
gP     = [3, 1, 0]';

OBJ_PR = bh_rot_passive_G2B_CLS({'D1Z'}, alpha, 'RADIANS');

R      = OBJ_PR.get_R1();
bP     = R * gP
```

```
bP =
          2.36602540378444
         -2.09807621135332
                         0
```

An example of 3 successive PASSIVE rotations

In the previous example we considered just 1 rotation. Consider now the scenario of performing a sequence of rotations. As before, 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:

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

OR, in a more compact form as:

$$\mathbf{{}^{B}v = {}^{B}R_{G} \times {}^{G}v}$$

Let's explore:

Let's explore these 3 passive rotations using the MATLAB class <**`bh_rot_passive_G2B_CLS`**> that we used earlier. Note in the code below how we are stating that the 1st rotation ϕ is about the local Z axis (ie: D1Z), and the second rotation θ is then around the local Y axis (ie: D2Y), and the 3rd rotation ψ is then around the local X axis (ie: D3X). In the example below we're alos going to use "symbolic" variables for our rotation angles.

```
OBJ_B = bh_rot_passive_G2B_CLS({'D1Z', 'D2Y', 'D3X'}, [sym('phi'), sym('theta'), sym('psi')],
```

```
OBJ_B =
  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]
```

The symbolic PASSIVE rotation matrices

```
R1 = OBJ_B.get_R1
```

R1 =

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

```
R2 = OBJ_B.get_R2
```

R2 =

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

```
R3 = OBJ_B.get_R3
```

R3 =

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

Here are some compound PASSIVE rotation matrices - part 1

```
R2R1       = OBJ_B.get_R2R1
```

R2R1 =

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

Note that "R2R1" is the same thing as "R2*R1":

```
diff_mat    = R2R1 - R2*R1    % this should be zero
```

diff_mat =

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Here are some compound PASSIVE rotation matrices - part 2

```
R3R2R1     = OBJ_B.get_R3R2R1
```

R3R2R1 =

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

Note that "**R3R2R1**" is the same thing as "**R3*R2*R1**":

```
diff_mat_B = R3R2R1 - R3*R2*R1    % this should be zero
```

diff_mat_B =

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Here's the PASSIVE rotation matrix ${}^{B}R_{G}$

```
bRg = R3*R2*R1
```

bRg =

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

Transform a vector in G, into its components in B

```
vG    = [1,0,0]';
bRg   = OBJ_B.get_R3R2R1;
vB    = bRg*vG
```

```
vB =
```

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

Transform a vector in G, into its components in B - alternate syntax

```
vG              = [1,0,0]';
vB_2nd_approach = OBJ_B.apply_R3R2R1(vG);
```

Note what the "**`apply_R3R2R1(vG)`**" method does the same thing as "**`R3R2R1 * vG`**"

```
diff_vB         = vB - vB_2nd_approach    % this should be zero
```

```
diff_vB =
```

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

Next steps:

If you found this tutorial interesting, there are 2 others that you may want to look at:

- **bhLIVE_TUT_rot_passive_G2B_example_3_euler_rates_CONCEPT.mlx** : In this tutorial we see an application of PASSIVE rotation matrices that comes from the modelling of aerial vehicles.
- **bhLIVE_TUT_rot_ACTIVE_B2G_example_1_CONCEPT.mlx** : In this tutorial we introduce the concept of the ACTIVE rotation matrix.