Explore ACTIVE rotations applied to a BODY-FIXED frame

Contents

Recall our discussion on PASSIVE rotations

Say we start with a G-frame. We're going to apply 3 LOCAL axes rotations which will result in a newly orientated frame called the 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 into it's corresponding description in the B axis, using a **PASSIVE** rotation matrix, ie:

vB = R3X(ψ_x) * R2Y(θ_y) * R1Z(ϕ_z) * vG

OR, in a more compact form as:

`vB = bRg * vG`

Now define what we mean by ACTIVE rotations

Continuing on from the previous section, we can now write:

vG = $R1Z(\phi_z)^{-1}$ ***** $R2Y(\theta_y)^{-1}$ ***** $R3X(\psi_x)^{-1}$ *** vB**

vG = $R1Z(\phi_z)^T$ ***** $R2Y(\theta_y)^T$ ***** $R3X(\psi_x)^T$ *** vB**

vG = $R1Z(-\phi_z)$ ***** $R2Y(-\theta_y)$ ***** $R3X(-\psi_x)$ *** vB**

If we now define the following **ACTIVE** rotation matrices:

1. **a_R1Z(**ϕ_z**) =** $R1Z(\phi_z)^{-1}$ **=** $R1Z(-\phi_z)$
2. **a_R2Y(**θ_y**) =** $R2Y(\theta_y)^{-1}$ **=** $R2Y(-\theta_y)$
3. **a_R3X(**ψ_x**) =** $R3X(\psi_x)^{-1}$ **=** $R3X(-\psi_x)$

Then we can write:

vG = a_R1Z(ϕ_z**) * a_R2Y(**θ_y**) * a_R3(**ψ_x**) * vB**

Or in a more compact form:

`vG = gRb * vB`

where it should be clear that:

gRb == $(bRg)^{-1}$ **==** $(bRg)^T$

Let's explore these ACTIVE rotations

```
OBJ_A = bh_rot_active_B2G_CLS({'D1Z', 'D2Y', 'D3X'}, [sym('phi'), sym('theta'), sym('psi')], 'SYM')
```

```
OBJ_A =

  bh_rot_active_B2G_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 ACTIVE rotation matrices

```
aR1     = OBJ_A.get_active_R1
aR2     = OBJ_A.get_active_R2
aR3     = OBJ_A.get_active_R3
```

```
aR1 =

[ cos(phi), -sin(phi), 0]
[ sin(phi),  cos(phi), 0]
[        0,         0, 1]


aR2 =

[  cos(theta), 0, sin(theta)]
[           0, 1,          0]
[ -sin(theta), 0, cos(theta)]


aR3 =

[ 1,        0,         0]
[ 0, cos(psi), -sin(psi)]
```

```
[ 0, sin(psi),  cos(psi)]
```

Here are some compound ACTIVE rotation matrices - part 1

```
aR1R2    = aR1*aR2

diff_mat = aR1R2 - OBJ_A.get_active_R1R2 % this should be a ZERO matrix
```

```
aR1R2 =

[ cos(phi)*cos(theta), -sin(phi), cos(phi)*sin(theta)]
[ cos(theta)*sin(phi),  cos(phi), sin(phi)*sin(theta)]
[         -sin(theta),         0,          cos(theta)]


diff_mat =

[ 0, 0, 0]
[ 0, 0, 0]
[ 0, 0, 0]
```

Here are some compound ACTIVE rotation matrices - part 2

```
aR1R2R3 = aR1*aR2*aR3

diff_mat = aR1R2R3 - OBJ_A.get_active_R1R2R3 % this should be a ZERO matrix
```

```
aR1R2R3 =

[ cos(phi)*cos(theta), cos(phi)*sin(psi)*sin(theta) - cos(psi)*sin(phi), sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta)]
[ cos(theta)*sin(phi), cos(phi)*cos(psi) + sin(phi)*sin(psi)*sin(theta), cos(psi)*sin(phi)*sin(theta) - cos(phi)*sin(psi)]
[         -sin(theta),                              cos(theta)*sin(psi),                              cos(psi)*cos(theta)]
```

```
diff_mat =

[ 0, 0, 0]
[ 0, 0, 0]
[ 0, 0, 0]
```

Here is ACTIVE rotation matrix gRb

Here is the compound ACTIVE rotation matrix:

```
gRb = aR1*aR2*aR3
```

```
gRb =

[ cos(phi)*cos(theta), cos(phi)*sin(psi)*sin(theta) - cos(psi)*sin(phi), sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta)]
[ cos(theta)*sin(phi), cos(phi)*cos(psi) + sin(phi)*sin(psi)*sin(theta), cos(psi)*sin(phi)*sin(theta) - cos(phi)*sin(psi)]
[         -sin(theta),                              cos(theta)*sin(psi),                              cos(psi)*cos(theta)]
```

Recall the PASSIVE rotation matrix bRg

Note how the inverse of the **ACTIVE** gRb is just the **PASSIVE** bRg which we computed during our discussion on PASSIVE rotations

```
bRg = inv(gRb);
simplify(bRg)
```

```
ans =

[                              cos(phi)*cos(theta),                              cos(theta)*sin(phi),         -sin(theta)]
[ cos(phi)*sin(psi)*sin(theta) - cos(psi)*sin(phi), cos(phi)*cos(psi) + sin(phi)*sin(psi)*sin(theta), cos(theta)*sin(psi)]
[ sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta), cos(psi)*sin(phi)*sin(theta) - cos(phi)*sin(psi), cos(psi)*cos(theta)]
```

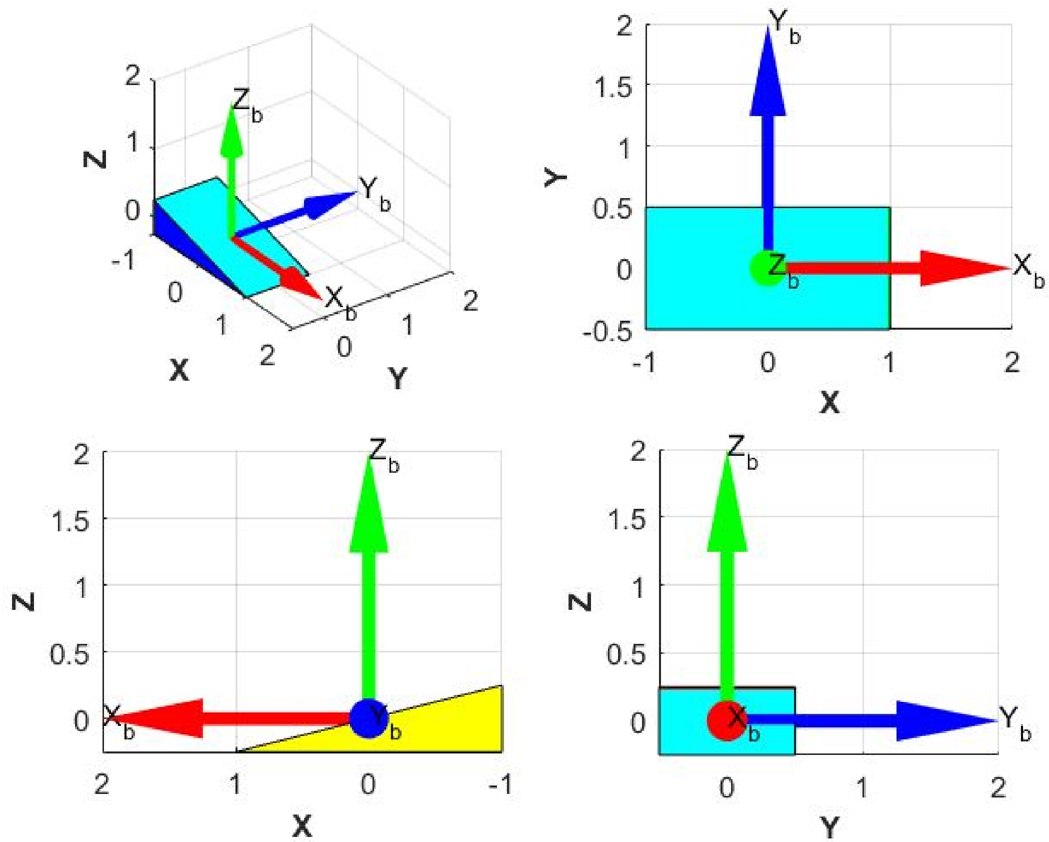
Define some geometry(co-ordinates) of a vehicle

```
% this will be the "toy" system that we'll rotate in space
veh_OBJ = bh_vehicle_CLS()
```

```
veh_OBJ = 

  bh_vehicle_CLS with properties:

    FaceAlpha: 1
          gRb: [3x3 double]
      X_b_col: [18x1 double]
      Y_b_col: [18x1 double]
      Z_b_col: [18x1 double]
      X_g_col: [18x1 double]
      Y_g_col: [18x1 double]
      Z_g_col: [18x1 double]
```

Show the vehicle in it's original pose

```
figure();
hax(1) = subplot(2,2,1);  veh_OBJ.plot_3D(hax(1));
hax(2) = subplot(2,2,2);  veh_OBJ.plot_XY(hax(2));
hax(3) = subplot(2,2,3);  veh_OBJ.plot_XZ(hax(3));
hax(4) = subplot(2,2,4);  veh_OBJ.plot_YZ(hax(4));
```



Define the ACTIVE rotation sequence and angles

We'd like to subject the vehicle to a series of rotations applied to a body fixed co-ordinate frame attached to the vehicle.

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**

```
degs_yaw  = 90;
degs_pitch= 30;
degs_roll = 60;
```

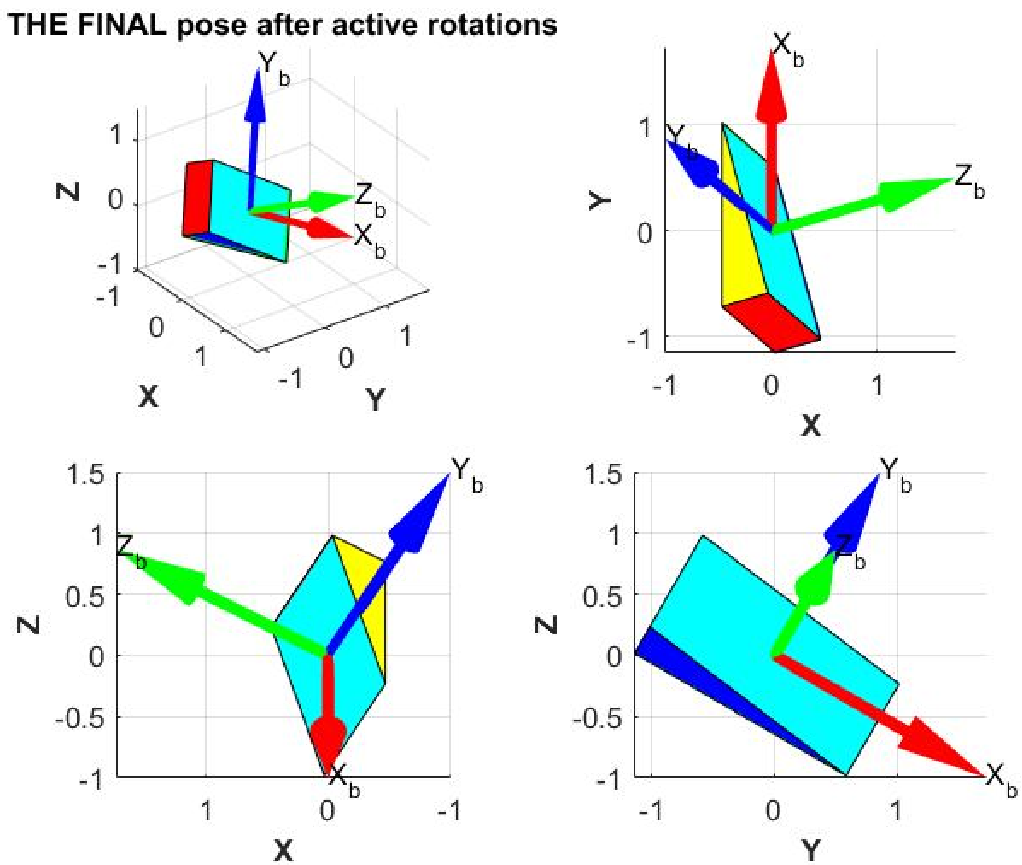
```
arot_OBJ  = bh_rot_active_B2G_CLS({'D1Z','D2Y','D3X'}, ...
                                   [degs_yaw, degs_pitch, degs_roll], ...
                                   'DEGREES')
```

```
arot_OBJ =

  bh_rot_active_B2G_CLS with properties:

        ang_units: DEGREES
    num_rotations: 3
          dir_1st: D1Z
          dir_2nd: D2Y
          dir_3rd: D3X
          ang_1st: 90
          ang_2nd: 30
          ang_3rd: 60
```

Now apply this ACTIVE rotation sequence to the vehicle

```
% get each of the active rotation matrices
aR1 = arot_OBJ.get_active_R1();
aR2 = arot_OBJ.get_active_R2();
aR3 = arot_OBJ.get_active_R3();

% chain them together in the correct ACTIVE order
aR1R2R3 = aR1 * aR2 * aR3;

% get the current G frame geometry data of the vehicle
[X,Y,Z] = veh_OBJ.get_G_XYZ();
v_mat   = [ X(:), Y(:), Z(:) ]';  % a 3xN matrix

% now apply the complete ACTIVE rotation matrix to our vehicle data
new_XYZ = aR1R2R3 * v_mat;

% store this new rotated vehicle data
veh_OBJ       =  veh_OBJ.set_G_XYZ(new_XYZ(1,:)', new_XYZ(2,:)', new_XYZ(3,:)');

% store the DCM so that we can draw the body fixed frame
veh_OBJ.gRb = arot_OBJ.get_active_R;
```

```
% plot the new rotated vehicle
figure();
hax(1) = subplot(2,2,1);  veh_OBJ.plot_3D(hax(1));
hax(2) = subplot(2,2,2);  veh_OBJ.plot_XY(hax(2));
hax(3) = subplot(2,2,3);  veh_OBJ.plot_XZ(hax(3));
hax(4) = subplot(2,2,4);  veh_OBJ.plot_YZ(hax(4));
title(hax(1), 'THE FINAL pose after active rotations')
```

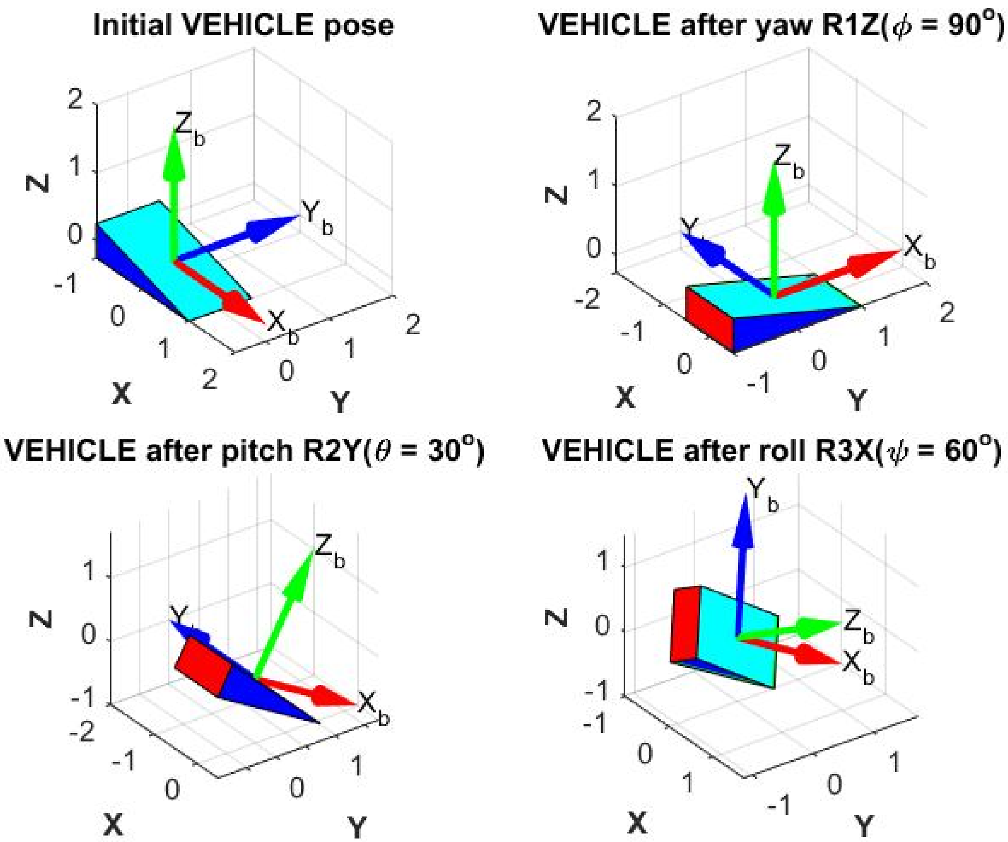


REPEAT what we just did ... BUT let's show the progressive rotations

```
veh_OBJ = bh_vehicle_CLS();
figure();
clear hax
```

```
% Here's the vehicle in its ORIGINAL pose
hax(1) = subplot(2,2,1);  veh_OBJ.plot_3D(hax(1));
title(hax(1),'Initial VEHICLE pose')

% apply the 1st active rotation
clear veh_OBJ
veh_OBJ = bh_vehicle_CLS();                   % ORIG pose is starting point
V_3xN   =  veh_OBJ.get_G_XYZ_3xN();          % get current vehicle data
new_XYZ = arot_OBJ.apply_active_R1(V_3xN); % apply the rotation
veh_OBJ =  veh_OBJ.set_G_XYZ(new_XYZ(1,:)', new_XYZ(2,:)', new_XYZ(3,:)');
gRb     = arot_OBJ.get_active_R1();          % get and store the DCM
veh_OBJ.gRb = gRb;
% update the vehilcle's PLOT
hax(2)  = subplot(2,2,2);  veh_OBJ.plot_3D(hax(2));
str = sprintf('VEHICLE after yaw R1Z(\\phi = %d^o)',degs_yaw);
title(hax(2),str)

% apply the 2nd active multiplication
clear veh_OBJ
veh_OBJ = bh_vehicle_CLS();                     % ORIG pose is starting point
V_3xN   =  veh_OBJ.get_G_XYZ_3xN();            % get current vehicle data
new_XYZ = arot_OBJ.apply_active_R1R2(V_3xN); % apply the rotation
veh_OBJ =  veh_OBJ.set_G_XYZ(new_XYZ(1,:)', new_XYZ(2,:)', new_XYZ(3,:)');
gRb   = arot_OBJ.get_active_R1R2();
veh_OBJ.gRb = gRb;
% update the vehilcle's PLOT
hax(3)  = subplot(2,2,3);  veh_OBJ.plot_3D(hax(3));
str = sprintf('VEHICLE after pitch R2Y(\\theta = %d^o)',degs_pitch);
title(hax(3),str)

% apply the 3rd active multiplication
clear veh_OBJ
veh_OBJ = bh_vehicle_CLS();                       % ORIG pose is starting point
V_3xN   =  veh_OBJ.get_G_XYZ_3xN();              % get current vehicle data
new_XYZ = arot_OBJ.apply_active_R1R2R3(V_3xN); % apply the rotation
veh_OBJ =  veh_OBJ.set_G_XYZ(new_XYZ(1,:)', new_XYZ(2,:)', new_XYZ(3,:)');
gRb    = arot_OBJ.get_active_R1R2R3();
veh_OBJ.gRb = gRb;
% update the vehilcle's PLOT
hax(4)  = subplot(2,2,4);
          veh_OBJ.plot_3D(hax(4));
str = sprintf('VEHICLE after roll R3X(\\psi = %d^o)',degs_roll);
```

```
title(hax(4),str)
```



Published with MATLAB® R2015b