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

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 to 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

Contents

Create a passive rotation object

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

```
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
R2 = OBJ_B.get_R2
R3 = OBJ_B.get_R3
```

```
R1 =

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


R2 =

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


R3 =

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

Here are some compound PASSIVE rotation matrices - part 1

```
R2R1        = OBJ_B.get_R2R1

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

```
R2R1 =

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


diff_mat =

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

Here are some compound PASSIVE rotation matrices - part 2

```
R3R2R1      = OBJ_B.get_R3R2R1

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

```
R3R2R1 =

[                              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)]


diff_mat_B =
```

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

Here's the PASSIVE rotation matrix bRg

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

```
bRg =

[                               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)]
```

Transform a vector in G, into its components in B

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

```
vB =

                               cos(phi)*cos(theta)
 cos(phi)*sin(psi)*sin(theta) - cos(psi)*sin(phi)
 sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta)
```

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

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

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

```
diff_vB =

 0
 0
 0
```

Published with MATLAB® R2015b