Table of Contents

0

```
close all;
clear all;
clc
addpath("C:\joshFunctionsMatlab\")
```

P1

this section is not a solution, just work to help me prove my solution works. See paper work.

```
[Cx,Cy,Cz] = joshAxisRotation();
syms theta1 theta2 theta3; % gamma Beta alpha

C21 = Cz(theta3)*Cy(theta2)*Cz(theta1);
C21s = simplify(subs(C21,theta2, 0)) % for singularity, guess based on
 C21(3,3) has one angle


syms dtheta1 dtheta2 dtheta3


w2t = [0;0;dtheta3];
wti = Cz(theta3)*[0;dtheta2;0];
wi1 = Cz(theta3)*Cy(theta2)*[0;0;dtheta1];
w21 = simplify(w2t+wti+wi1)

R = [[0 sin(theta3) -cos(theta3)*sin(theta2)];...
     [0 cos(theta3) sin(theta2)*sin(theta3)];...
     [1 0 cos(theta2)]];

isAlways(w21 == R*[dtheta3;dtheta2;dtheta1]);
```

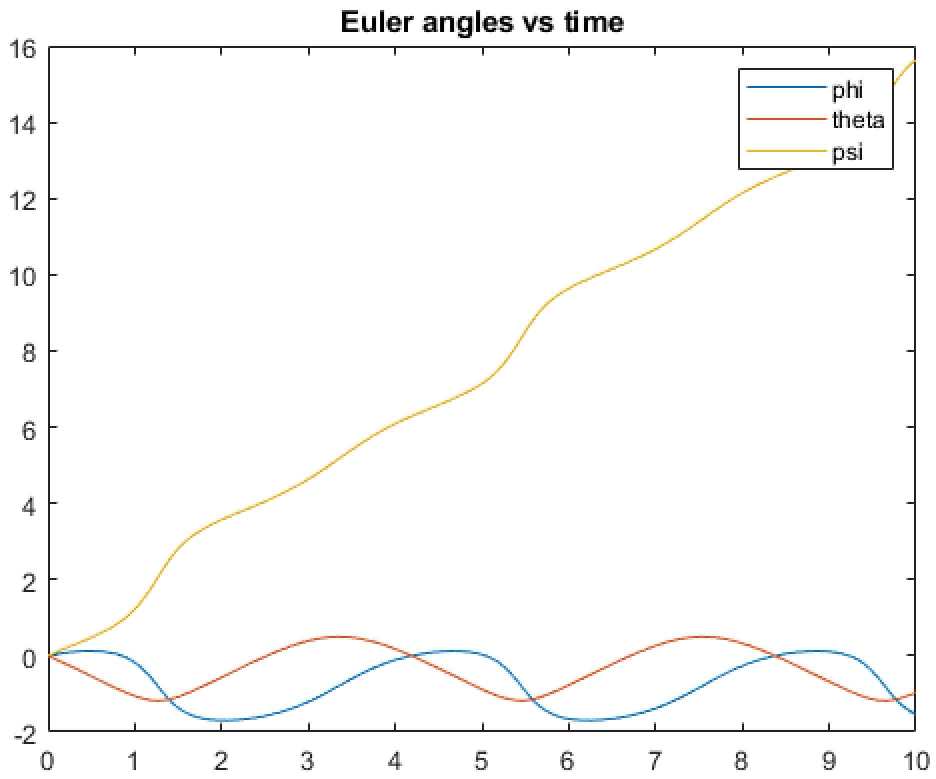
```
C21s =

[ cos(theta1 + theta3), sin(theta1 + theta3), 0]
[-sin(theta1 + theta3), cos(theta1 + theta3), 0]
[                    0,                    0, 1]
```

```
w21 =

dtheta2*sin(theta3) - dtheta1*cos(theta3)*sin(theta2)
dtheta2*cos(theta3) + dtheta1*sin(theta2)*sin(theta3)
                        dtheta3 + dtheta1*cos(theta2)
```

P2

```
clear all

[Cx,Cy,Cz] = joshAxisRotation();

C0 = eye(3); % no rotation so Identity matrix
[eta,epsilon] = joshRotM2Quat(C0);

w21 = [.5;-1;1];
wx = joshCross(w21);

% defines ode for euler angles
dEA =@(t,EA) (1/cos(EA(2)))*([[cos(EA(2)), sin(EA(1))*sin(EA(2)),
 cos(EA(1))*sin(EA(2))];...
                              [0         , cos(EA(1))*cos(EA(2)),-
sin(EA(1))*cos(EA(2))];...
                              [0         , sin(EA(1))           , cos(EA(1))
         ]]) * w21

[t, EA] = ode45(dEA,[0 10],[0;0;0]);

figure
plot(t,EA(:,1),t,EA(:,2),t,EA(:,3))
title("Euler angles vs time")
legend("phi","theta","psi")

% [t,quat] = ode45(dquatfun,[0,10],[epsilon;eta])
disp("I wasn't able to get the quaternion to run correctly but it would be
 accomplished similar to the commented line")
```

```
dEA =

  function_handle with value:

    @(t,EA)(1/
cos(EA(2)))*([[cos(EA(2)),sin(EA(1))*sin(EA(2)),cos(EA(1))*sin(EA(2))];
[0,cos(EA(1))*cos(EA(2)),-sin(EA(1))*cos(EA(2))];
[0,sin(EA(1)),cos(EA(1))]])*w21

I wasn't able to get the quaternion to run correctly but it would be
 accomplished similar to the commented line
```



P3

```
clear all;
[Cx,Cy,Cz] = joshAxisRotation();
syms a0 w t

assume(t>0)
assume(w,'real')
assume(a0>0)

w21 = [0;0;w];
theta = w*t;
r2 = [.5*a0*t^2;0;0];
dr2 = simplify(diff(r2,t));
ddr2 = simplify(diff(dr2,t));
C12 = Cz(-theta);
r1 = C12*r2;
dr1 = simplify(diff(r1,t));
ddr1 = simplify(diff(dr1,t));

wx = joshCross(w21);
dwx = joshCross(diff(w21,t));
```

```
disp ("These logical matrices show that dr1 == C12*(dr2+wx*r2) and ddr1 ==
 C12*(ddr2 + 2*wx*dr2 + dwx*r2 + wx*wx*r2)")
isAlways(dr1 == C12*(dr2+wx*r2))
isAlways(ddr1 == C12*(ddr2 + 2*wx*dr2 + dwx*r2 + wx*wx*r2))
```

```
These logical matrices show that dr1 == C12*(dr2+wx*r2) and ddr1 == C12*(ddr2
 + 2*wx*dr2 + dwx*r2 + wx*wx*r2)

ans =

  3×1 logical array

   1
   1
   1


ans =

  3×1 logical array

   1
   1
   1
```

functions

```
function dquat =  dquatfun(t,quat)
    w = [.5;-1;1];
    deta = -.5*(quat(1:3)')*w;
    depsilon = .5*(quat(4)*eye(3)+joshCross(quat(1:3)))*w;
    dquat = [epsilon;deta];
end
```

Published with MATLAB® R2022a