Table of Contents

passivityCtrl

Creates the ODE function for a two link planar arm tracking a cubic polynomial trajectory by passivity-based control.

```
function [ dx ] = passivityCtrl( t, x, a1, a2)
```

Constants and Variables

Sets *A* as the global *A* variable (the past joint accelerations):

```
global A
```

Set the parameters for the arm:

```
I1=10;  I2 = 10; m1=5; r1=.5; m2=5; r2=.5; l1=1; l2=1;
g=9.8;
```

Calculate the parameters in the dynamic model:

```
a = I1+I2+m1*r1^2+ m2*(l1^2+ r2^2);
b = m2*l1*r2;
d = I2+ m2*r2^2;
```

Trajectory Generation

Note *x* is in the form of *q1*, *q2*, *q1_dot*, *q2_dot*:

Cubic polynomials:

```
vec_t = [1; t; t^2; t^3];
theta_d = [a1'*vec_t; a2'*vec_t];
```

Calculate the velocity and acceleration in both *theta 1* and *theta 2*:

```
a1_vel = [a1(2), 2*a1(3), 3*a1(4), 0];
a1_acc = [2*a1(3), 6*a1(4),0,0 ];
a2_vel = [a2(2), 2*a2(3), 3*a2(4), 0];
a2_acc = [2*a2(3), 6*a2(4),0,0 ];
```

Calculate the desired trajectory (assuming 3rd order polynomials for trajectories):

```
dtheta_d =[a1_vel*vec_t; a2_vel* vec_t];
ddtheta_d =[a1_acc*vec_t; a2_acc* vec_t];
```

Set the current joint values:

```
theta = x(1:2,1);
theta_dot = x(3:4,1);
theta_dot_dot = A;
```

Planar Arm Dynamics

Calculate the parameters in the dynamic model:

```
a = I1+I2*t+m1*r1^2+ m2*(l1^2+ r2^2);
b = m2*l1*r2;
d = I2+ m2*r2^2;
```

Calculate the actual dynamic model of the system:

```
Mmat = [a+2*b*cos(x(2)), d+b*cos(x(2));  d+b*cos(x(2)), d];
Cmat = [-b*sin(x(2))*x(4), -b*sin(x(2))*(x(3)+x(4));
 b*sin(x(2))*x(3),0];
Gmat =  [m1*g*r1*cos(x(1))+m2*g*(l1*cos(x(1))+r2*cos(x(1)+x(2)));
    m2*g*r2*cos(x(1)+x(2))];
invM = inv(Mmat);
invMC = invM*Cmat;
```

Passivity-Based Controller

Set the *kv* gain constant (positive definite matrix):

```
kv = [25 0; ...
      0 25];
```

Set the *capital_lambda* constant (positive definite square matrices):

```
capital_lambda = [10 0; ...
                  0 10];
```

Calculate the tracking errors, *e*, *e_dot*, and *e_dot_dot*:

```
e = theta - theta_d;
e_dot = theta_dot - dtheta_d;
e_dot_dot = theta_dot_dot - ddtheta_d;
```

Calculate *r* and *r_dot*:

```
I = eye(2,2);
r = e_dot + capital_lambda*e;
r_dot = e_dot_dot + capital_lambda*e_dot;
```

Calculate *v*:

```
% v = q_dot - r
v = theta_dot - r;
```

Calculate *a*:

```
%a = q_dot_dot - r
a = theta_dot_dot - r_dot;
```

Calculate the controller, *u*:

```
u = zeros(2,1);
u = Mmat*a + Cmat*v + Gmat - kv*r;
```

Calculate the acceleration values:

```
theta_dot_dot = zeros(2,1);
% theta_dot_dot = sigma_dot - si_dot_dot
theta_dot_dot = invM*(u - Cmat*theta_dot - Gmat);
```

Update the acceleration values:

```
A = theta_dot_dot;
```

Outputs

Initialize the output of the function, *dx*:

```
dx = zeros(4,1);
```

Set the final outputs:

```
dx(1) = x(3,1);
dx(2) = x(4,1);
dx(3) = theta_dot_dot(1);
dx(4) = theta_dot_dot(2);

end
```

Published with MATLAB® R2018a