

10.3. MATLAB - Partial Feedback Linearization

C:\brachiation\PFL\brachiation PFL main.m

1 of 1

```
clear all
close all
clc
global param

%init function
brachiation_PFL_init

%simulation
sol = rk4(@brachiation_PFL,tf,dt,param.ICs);
t=sol.t;
n=length(t);
q1 = sol.y(1,:);
q2 = sol.y(2,:);
q1dot = sol.y(3,:);
q2dot = sol.y(4,:);

%animation
r1 = L1*[sin(q1); -cos(q1)];
r2 = r1 + L2*[sin(q1+q2); -cos(q1+q2)];
figure('name','Animation window')
plt1 = plot([0,r1(1,1)],[0,r1(2,1)],'-ko','linewidth',2); %first segment
hold on
plt2 = plot([r1(1,1),r2(1,1)],[r1(2,1),r2(2,1)],'-ro','linewidth',2); %second segment
axis equal
set(gca,'xlim',[-1.2,1.2],'ylim',[-1,1])

%animation controllers
sld = uicontrol('style','slider','position',[0 50 100 20],'sliderstep',[1/n↵
10/n],'value',1,'min',1,'max',n,'callback','animate');
tinfo = uicontrol('style','text','position',[270 50 100 20],'string','time: ');
```

```
global param flag

%mechanical parameters
m1 = 1.129;
m2 = 0.762;
I1 = 0.042857;
I2 = 0.0406244;
L1 = 0.28;
L2 = 0.28;
xG1 = 0.140526;
xG2 = 0.198104;
yG1 = 0;
yG2 = 0;
g = 9.81;

%store values in a structure
param.m1 = m1;
param.m2 = m2;
param.I1 = I1;
param.I2 = I2;
param.L1 = L1;
param.L2 = L2;
param.xG1 = xG1;
param.xG2 = xG2;
param.yG1 = yG1;
param.yG2 = yG2;
param.g = g;

%simulation parameters
tf = 6; %final simulation time
dt = 0.002; %timestep
q0 = [40/180*pi 100/180*pi]'; %initial conditions
qdot0 = [0 -3]';
ICs = [q0; qdot0];
flag = 0; %error flag for integration

param.tf = tf;
param.dt = dt;
param.ICs = ICs;
```

```

function ydot = dpend_mincoord(t,y)
global param flag

%parameters
m1 = param.m1;
m2 = param.m2;
I1 = param.I1;
I2 = param.I2;
xG1 = param.xG1;
xG2 = param.xG2;
L1 = param.L1;
g = param.g;

q1 = y(1);
q2 = y(2);
q1dot = y(3);
q2dot = y(4);

%Partial Feedback Linearization
u =1;
tau=I2*u + L1*m2*q1dot^2*xG2*sin(q2) + g*m2*xG2*sin(q1 + q2) - ...
    ((I2 + L1*m2*xG2*cos(q2))*(I2*u + L1*m2*u*xG2*cos(q2) + g*L1*m2*sin(q1) + ...
    g*m1*xG1*sin(q1) - 2*L1*m2*q1dot*q2dot*xG2*sin(q2) - ...
    L1*m2*q2dot^2*xG2*sin(q2) + g*m2*xG2*sin(q1 + q2)))/...
    (I1 + I2 + L1^2*m2 + 2*L1*m2*xG2*cos(q2));
param.tau(param.n)=tau;

%equations of motion
m11=I1+I2+L1^2*m2+2*L1*m2*xG2*cos(q2);
m12=I2+L1*m2*xG2*cos(q2);
m22=I2;
c1=-L1*m2*xG2*sin(q2)*q2dot*(2*q1dot+q2dot);
c2=L1*m2*xG2*sin(q2)*q1dot^2;
Q1= -g*((L1*m2 + m1 * xG1)*sin(q1) + m2*xG2*sin(q1 + q2));
Q2=tau - g*m2*xG2*sin(q1 + q2);

M = [[m11,...
      m12];...
      [m12,...
      m22]];
c = [c1;...
      c2];
Q = [ Q1;...
      Q2];

qddot = M\ (Q - c);
qdot = y(3:4);
ydot = [qdot; qddot];
end

```

```
function sol = rk4(fhandle,tf,dt,ICs)
%Runge Kutta
global flag param

%initial values
t = 0;
y = ICs;
n = 1;
N = length(ICs);
sol.t(n) = t;
sol.y(1:N,n) = y;

%simulation
while t<tf && flag<1
    param.n=n+1;
    k1 = dt * feval(fhandle, t, y);
    k2 = dt * feval(fhandle, t + dt/2 , y + k1/2);
    k3 = dt * feval(fhandle, t + dt/2 , y + k2/2);
    k4 = dt * feval(fhandle, t + dt, y + k3);

    t = n*dt;
    y = y + (k1 + 2*k2 + 2*k3 + k4)/6;
    n = n+1;
    sol.t(n) = t;
    sol.y(1:N,n) = y;
end
```

```
sld_value = get(sld,'value');

i = floor(sld_value);

set(plt1,'xdata',[0,r1(1,i)],'ydata',[0,r1(2,i)])
set(plt2,'xdata',[r1(1,i),r2(1,i)],'ydata',[r1(2,i),r2(2,i)])

set(tinfo,'string',['time: ' num2str(i*dt)])

drawnow
```