Term 081

KFUPM

EE 656 – Robotics & Control

HW #5

2DOF Robotic Manipulator

Control Design & Simulation

By

Mohammad Shahab

227598

For

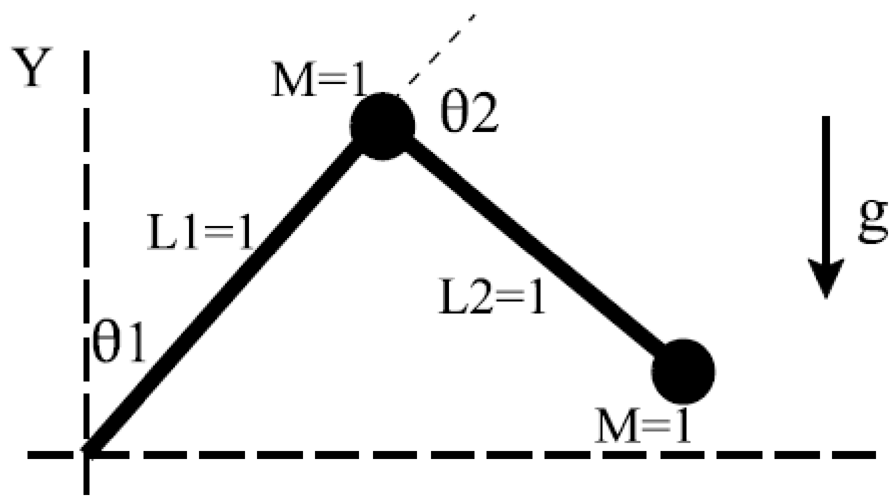
Dr. Ahmad Masoud

20 December 2008

Term 081

EE 656 – Robotics & Control

HW #5



1) Robotic Arm Dynamics

We can put

$$x_1 = L_1 \sin\theta_1$$

$$y_1 = L_1 \cos\theta_1$$

$$x_2 = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

$$y_2 = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$

So, **Kinetic Energy** could be formed as

$$KE = \frac{1}{2}M_1\dot{x}_1^2 + \frac{1}{2}M_1\dot{y}_1^2 + \frac{1}{2}M_2\dot{x}_2^2 + \frac{1}{2}M_2\dot{y}_2^2$$

By simplification,

$\Longrightarrow$

$$KE = \frac{1}{2}(M_1 + M_2)L_1^2\dot{\theta}_1^2 + \frac{1}{2}M_2L_2^2\dot{\theta}_1^2 + M_2L_2^2\dot{\theta}_1\dot{\theta}_2 + \frac{1}{2}M_2L_2^2\dot{\theta}_2^2 + M_2L_1L_2\cos\theta_2\,(\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_1^2)$$

And **Potential Energy** is

$$PE = M_1gL_1\cos\theta_1 + M_2g(L_1\cos\theta_1 + L_2\cos(\theta_1 + \theta_2))$$

So, by Lagrange Dynamics, we form the **Lagrangian**

$$\mathcal{L} = KE - PE$$

$$\mathcal{L} = \frac{1}{2}(M_1 + M_2)L_1^2\dot{\theta}_1^2 + \frac{1}{2}M_2L_2^2\dot{\theta}_1^2 + M_2L_2^2\dot{\theta}_1\dot{\theta}_2 + \frac{1}{2}M_2L_2^2\dot{\theta}_2^2 + M_2L_1L_2\cos\theta_2\left(\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_1^2\right) - M_1gL_1\cos\theta_1 - M_2g(L_1\cos\theta_1 + L_2\cos(\theta_1 + \theta_2))$$

So, forming the dynamics equations to be

$$f_{\theta_{1,2}} = \frac{d}{dt}\left[\frac{\partial\mathcal{L}}{\partial\dot{\theta}_{1,2}}\right] - \frac{\partial\mathcal{L}}{\partial\theta_{1,2}}$$

So, the dynamic equations after simplifications become

$$\left((M_1 + M_2)L_1^2 + M_2L_2^2 + 2M_2L_1L_2\cos\theta_2\right)\ddot{\theta}_1 + (M_2L_2^2 + M_2L_1L_2\cos\theta_2)\ddot{\theta}_2 - M_2L_1L_2\sin\theta_2\left(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right) - (M_1 + M_2)gL_1\sin\theta_1 - M_2gL_2\sin(\theta_1 + \theta_2) = f_{\theta_1}$$

$$(M_2L_2^2 + M_2L_1L_2\cos\theta_2)\ddot{\theta}_1 + M_2L_2^2\ddot{\theta}_2 - M_2L_1L_2\sin\theta_2\,\dot{\theta}_1\dot{\theta}_2 - M_2gL_2\sin(\theta_1 + \theta_2) = f_{\theta_2}$$

So, we can describe the motion of the system by

$$\boxed{\begin{gathered} B(q)\ddot{q} + C(\dot{q}, q) + g(q) = F \\ q = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} \\ B(q) = \begin{bmatrix}((M_1 + M_2)L_1^2 + M_2L_2^2 + 2M_2L_1L_2\cos\theta_2) & (M_2L_2^2 + M_2L_1L_2\cos\theta_2) \\ M_2L_2^2 + M_2L_1L_2\cos\theta_2 & M_2L_2^2\end{bmatrix} \\ C(\dot{q}, q) = \begin{bmatrix}-M_2L_1L_2\sin\theta_2\left(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right) \\ -M_2L_1L_2\sin\theta_2\,\dot{\theta}_1\dot{\theta}_2\end{bmatrix} \\ g(q) = \begin{bmatrix}-(M_1 + M_2)gL_1\sin\theta_1 - M_2gL_2\sin(\theta_1 + \theta_2) \\ -M_2gL_2\sin(\theta_1 + \theta_2)\end{bmatrix} \\ F = \begin{bmatrix}f_{\theta_1} \\ f_{\theta_2}\end{bmatrix} \end{gathered}}$$

Here we have: $M_1 = M_2 = L_1 = L_2 = 1$

2) Control Design

Having the system equation

$$B(q)\ddot{q} + C(\dot{q}, q) + g(q) = F$$

We can have

$$\ddot{q} = B(q)^{-1}[-C(\dot{q}, q) - g(q)] + \hat{F}$$

With

$$\hat{F} = B(q)^{-1}F \Leftrightarrow F = B(q)\hat{F}$$

So, we **decoupled** the system to have the 'new' (non-physical) input

$$\hat{F} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

However, the **physical torque inputs** to the system are

$$\begin{bmatrix} f_{\theta_1} \\ f_{\theta_2} \end{bmatrix} = B(q) \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

The error signals

$$e(\theta_1) = \theta_{1f} - \theta_1$$

$$e(\theta_2) = \theta_{2f} - \theta_2$$

With **final positions**

$$\begin{bmatrix} \theta_{1f} \\ \theta_{2f} \end{bmatrix} = \begin{bmatrix} \frac{\pi}{2} \\ -\frac{\pi}{2} \end{bmatrix}$$

The system has **initial positions** of

$$\boldsymbol{\theta_0} = \begin{bmatrix} -\frac{\pi}{2} \\ \frac{\pi}{2} \end{bmatrix}$$

- **PID Design**

General structure of PID controller for any input would be

$$f = K_p e + K_D \dot{e} + K_I \int e\, dt$$

So, in our case,

$$f_1 = K_{p1}\left(\theta_{1f} - \theta_1\right) - K_{D1}\dot{\theta}_1 + K_{I1} \int e(\theta_1)\, dt$$

$$f_2 = K_{p2}\left(\theta_{2f} - \theta_2\right) - K_{D2}\dot{\theta}_2 + K_{I2} \int e(\theta_2)\, dt$$

So, the complete system equations with control would be

$$\ddot{q} = B(q)^{-1}[-C(\dot{q}, q) - g(q)] + \hat{F}$$

With

$$\hat{F} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} K_{p1}\left(\theta_{1f} - \theta_1\right) - K_{D1}\dot{\theta}_1 + K_{I1} \int e(\theta_1)\, dt \\ K_{p2}\left(\theta_{2f} - \theta_2\right) - K_{D2}\dot{\theta}_2 + K_{I2} \int e(\theta_2)\, dt \end{bmatrix}$$

Recall: with actual physical torques of

$$\begin{bmatrix} f_{\theta_1} \\ f_{\theta_2} \end{bmatrix} = B(q) \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

- **Solution**

In order to apply all controls of Proportional-Derivative-Integral actions, a **'dummy' state** is added for each angle to resemble the ***integration inside the computer***.

$$x_1 = \int e(\theta_1)\, dt \Rightarrow \dot{x}_1 = \theta_{1f} - \theta_1$$

$$x_2 = \int e(\theta_2)\, dt \Rightarrow \dot{x}_2 = \theta_{2f} - \theta_2$$

So, the complete system equations are

$$\begin{cases} \dot{x}_1 = \theta_{1f} - \theta_1 \\ \dot{x}_2 = \theta_{2f} - \theta_2 \\ \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = B(q)^{-1}[-C(\dot{q}, q) - g(q)] + \begin{bmatrix} K_{p1}(\theta_{1f} - \theta_1) - K_{D1}\dot{\theta}_1 + K_{I1}x_1 \\ K_{p2}(\theta_{2f} - \theta_2) - K_{D2}\dot{\theta}_2 + K_{I2}x_2 \end{bmatrix} \end{cases}$$

In MATLAB, "`ode45`" command was used to solve the ODE. (Full program in Appendix)

By trial & error, the 2 controllers' parameters were tuned to have the best performance. The best values for the parameters was found to be

$$K_{p1} = 15$$

$$K_{D1} = 7$$

$$K_{I1} = 10$$

$$K_{p2} = 15$$

$$K_{D2} = 10$$

$$K_{I2} = 10$$

Actually, by observing the structure of above ODE with control, we can see (roughly):

- K_p is related to direct error and to speed of evolution
- K_D is related to speed of interaction with change in states
- K_I is related to overall error cancelation

However, above arguments are rough because of the high nonlinearity of the equation that:

- produces sensitive interaction between controller components
- Small Changes in controller parameters would produce more overshoots and oscillations
- Controller parameters are highly sensitive to initial and final positions!
- So, online tuning of the PID should be considered for global system operation (i.e. non-fixed final positions, trajectory tracking, etc.)

3) States Results

Error forms of θ_1 & θ_2 is shown below

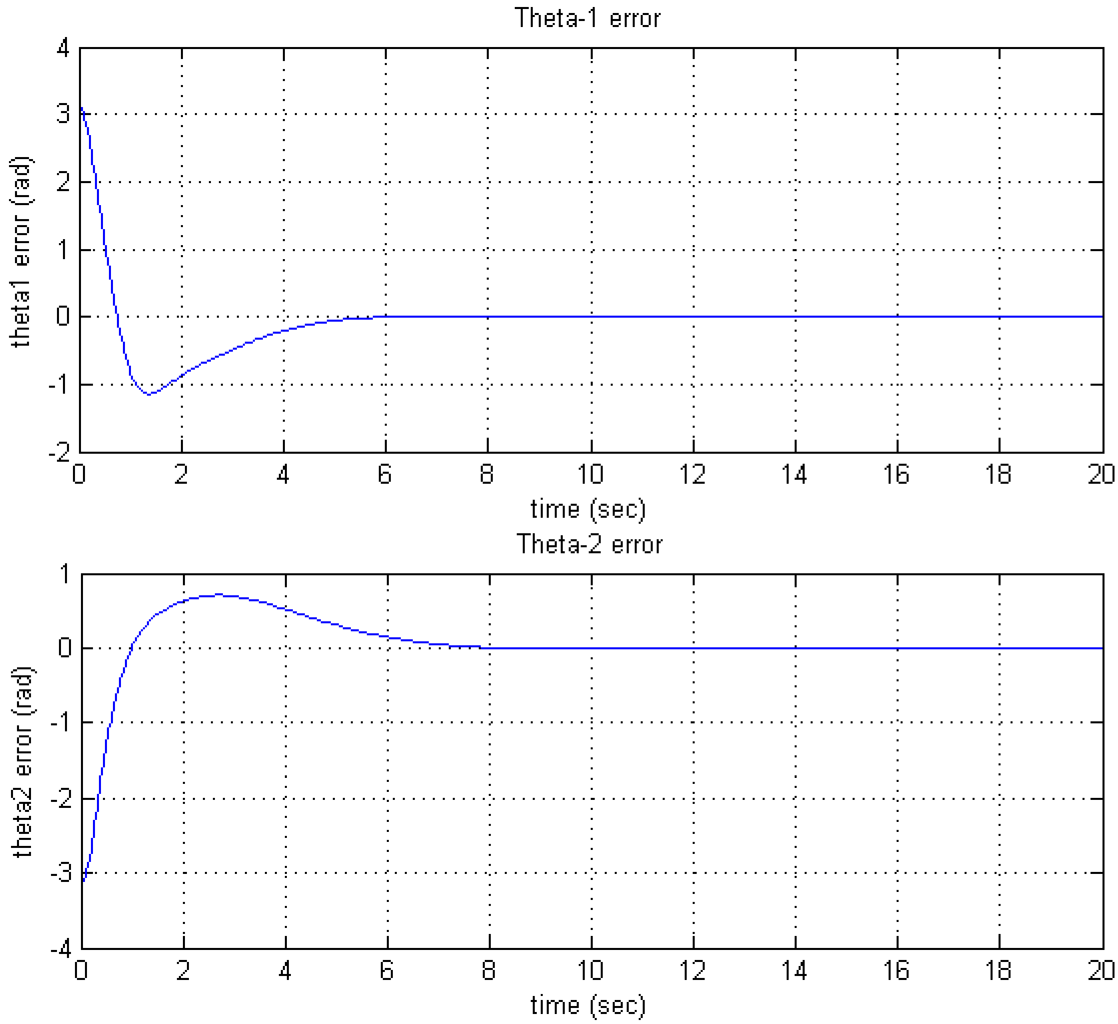


Figure 1: Errors waveforms of theta1 & theta2

Comments: You can see from above waveforms that:

- Acceptable overshoot
- Acceptable settling time
- 'Linear' behavior

NOTE: The same design was tested on other initial and final positions, the result was very different.

4) Torques Results

Here we analyze the torques resulted. The control inputs here are the torques. However, put in mind that the **actual joints torques** are

$$\begin{bmatrix} f_{\theta_1} \\ f_{\theta_2} \end{bmatrix} = B(q) \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

The waveforms of joints torques are

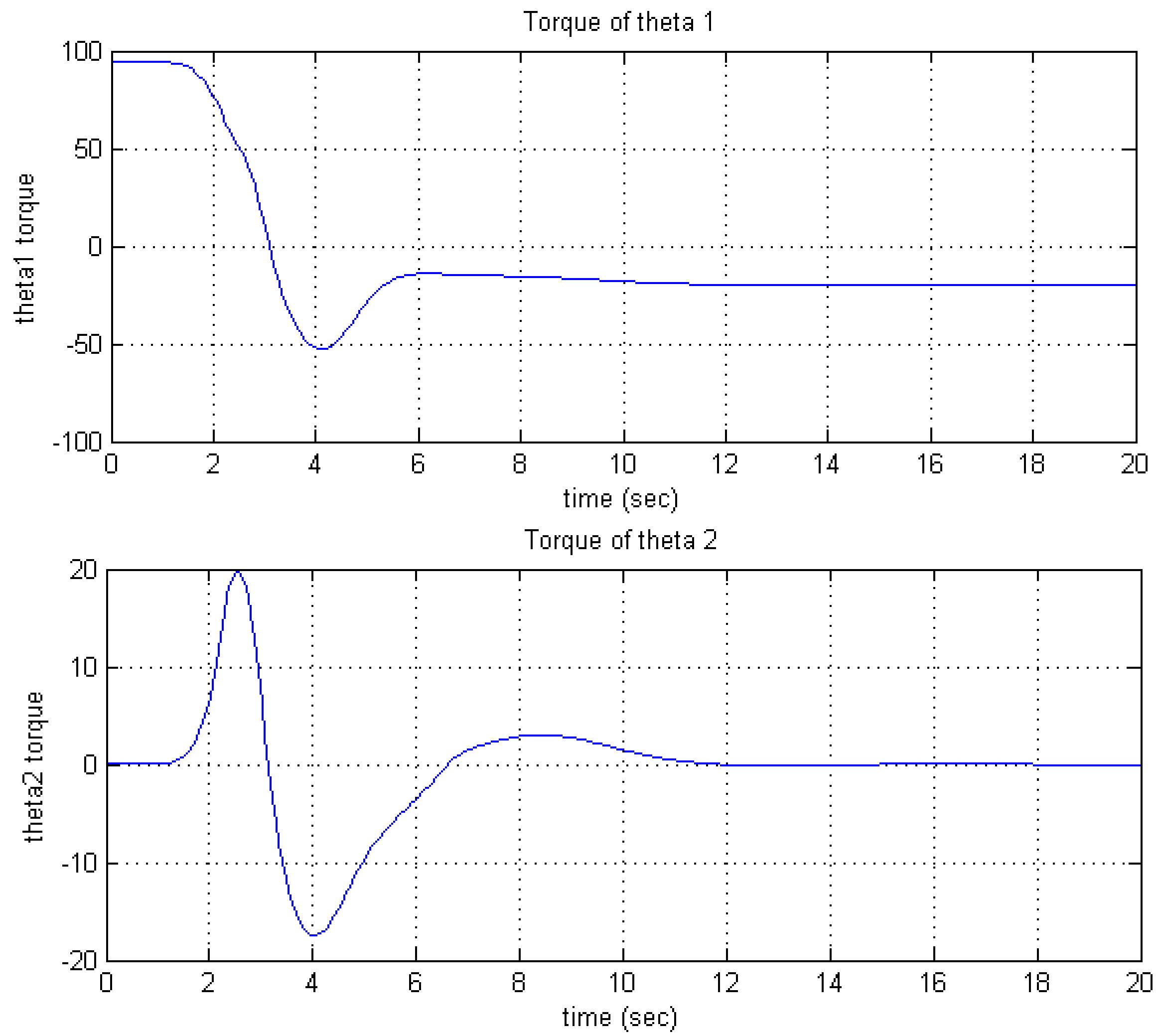


Figure 2: Actual Torques on Joints

Comments: from above plots,

- θ_1-joint somehow encounter high torque in relatively small time
- Overall acceptable performance as relatively energy spent is O.K.

5) Motion Animation

Using MATLAB, the resulting motion is animated in xy-plane to show the actual motion of the robot. An "mpg" file is generated. The file name is "2DOF_rob.mpg". (**MATLAB program is in Appendix**)

Appendix

I) System ODE function file ("r2dof.m")

```
function xdot=r2dof(t,x,ths,spec,Kpid)

xdot=zeros(8,1);
%% set-points
th1s=ths(1);
th2s=ths(2);

%% Robot Specifications
M1=spec(3);
M2=spec(4);
L1=spec(1);
L2=spec(2);
g=9.8;

%% Inertia Matrix
b11=(M1+M2)*L1^2+M2*L2^2+2*M2*L1*L2*cos(x(4));
b12=M2*L2^2+M2*L1*L2*cos(x(4));
b21=M2*L2^2+M2*L1*L2*cos(x(4));
b22=M2*L2^2;
Bq=[b11 b12;b21 b22];

%% C Matrix
c1=-M2*L1*L2*sin(x(4))*(2*x(5)*x(6)+x(6)^2);
c2=-M2*L1*L2*sin(x(4))*x(5)*x(6);
Cq=[c1;c2];

%% Gravity Matrix
g1=-(M1+M2)*g*L1*sin(x(3))-M2*g*L2*sin(x(3)+x(4));
g2=-M2*g*L2*sin(x(3)+x(4));
Gq=[g1;g2];

%% PID Control
% PID parameters for theta 1
Kp1=Kpid(1);
Kd1=Kpid(2);
Ki1=Kpid(3);
% PID parameters for theta 2
Kp2=Kpid(4);
Kd2=Kpid(5);
Ki2=Kpid(6);
%decoupled control input
f1=Kp1*(th1s-x(3))-Kd1*x(5)+Ki1*(x(1));
f2=Kp2*(th2s-x(4))-Kd2*x(6)+Ki2*(x(2));
Fhat=[f1;f2];

F=Bq*Fhat; % actual input to the system

%% System states
xdot(1)=(th1s-x(3)); %dummy state of theta1 integration
xdot(2)=(th2s-x(4)); %dummy state of theta2 integration

xdot(3)=x(5); %theta1-dot
xdot(4)=x(6); %theta2-dot

q2dot=inv(Bq)*(-Cq-Gq+F);
```

```
xdot(5)=q2dot(1); %theta1-2dot
xdot(6)=q2dot(2); %theta1-2dot

%control input function output to outside computer program
xdot(7)=F(1);
xdot(8)=F(2);
```

2) System Solution and Simulation (“r2dof_cntrl.m”)

```
close all
clear all
clc
%% Initilization
th_int=[-pi/2 pi/2]; %initial positions
ths=[pi/2 -pi/2]; %set-points

x0=[0 0 th_int 0 0 0 0]; %states initial values
Ts=[0 20]; %time span

%% Robot Specifications
L1=1; %link 1
L2=1; %link 2
M1=1; %mass 1
M2=1; %mass 2
spec=[L1 L2 M1 M2];

%% PID Parameters
% PID parameters for theta 1
Kp1=15;
Kd1=7;
Ki1=10;
% PID parameters for theta 2
Kp2=15;
Kd2=10;
Ki2=10;

Kpid=[Kp1 Kd1 Ki1 Kp2 Kd2 Ki2];

%% ODE solving
% opt1=odeset('RelTol',1e-10,'AbsTol',1e-20,'NormControl','off');
[T,X] = ode45(@(t,x) r2dof(t,x,ths,spec,Kpid),Ts,x0);

%% Output
th1=X(:,3); %theta1 wavwform
th2=X(:,4); %theta2 wavwform

%torque inputs computation from the 7th,8th states inside ODE
F1=diff(X(:,7))./diff(T);
F2=diff(X(:,8))./diff(T);
tt=0:(T(end)/(length(F1)-1)):T(end);

%xy
x1=L1.*sin(th1); % X1
y1=L1.*cos(th1); % Y1
x2=L1.*sin(th1)+L2.*sin(th1+th2); % X2
y2=L1.*cos(th1)+L2.*cos(th1+th2); % Y2

%theta1 error plot
plot(T,ths(1)-th1)
```

```
grid
title('Theta-1 error')
ylabel('theta1 error (rad)')
xlabel('time (sec)')
%theta2 error plot
figure
plot(T,ths(2)-th2)
grid
title('Theta-2 error')
ylabel('theta2 error (rad)')
xlabel('time (sec)')
%torque1 plot
figure
plot(tt,F1)
grid
title('Torque of theta 1')
ylabel('theta1 torque')
xlabel('time (sec)')
%torque2 plot
figure
plot(tt,F2)
grid
title('Torque of theta 2')
ylabel('theta2 torque')
xlabel('time (sec)')
```

3) Robot Animation (“movieHW5.m”)

```
%% setting frames speed
d=2;
j=1:d:length(T);

%% generating images in 2D
figure
for i=1:length(j)-1
    hold off
    plot([x1(j(i)) x2(j(i))],[y1(j(i)) y2(j(i))],'o',[0 x1(j(i))],[0
y1(j(i))],'k',[x1(j(i)) x2(j(i))],[y1(j(i)) y2(j(i))],'k')
    title('Motion of 2DOF Robotic Arm')
    xlabel('x')
    ylabel('y')
    axis([-3 3 -3 3]);
    grid
    hold on
    MM(i)=getframe(gcf);
end
drawnow;

%% exporting to 'mpg' movie
mpgwrite(MM,'RGB','2DOF_rob.mpg')
```