Assignment 4

By- Aishwary Jagetia

Contents

Notations: For a given variable, x, dx is its time derivative, ddx is

2nd-order derivative.

```
clc
clear all;
close all;
% the following parameters for the arm
I1=10;  I2 = 10; m1=5; r1=.5; m2=5; r2=.5; l1=1; l2=1; g=9.8;

% we compute 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;
```

create symbolic variable for x.

x1 - theta1 x2 - theta2

```
symx= sym('symx',[4,1]);

global M C Gq
M = [a+2*b*cos(symx(2)), d+b*cos(symx(2));
    d+b*cos(symx(2)), d];
C = [-b*sin(symx(2))*symx(4), -b*sin(symx(2))*(symx(3)+symx(4)); b*sin(symx(2))*symx(3),0];
invM = inv(M);
invMC= inv(M)*C;

% Gravity Matrix
g1=-(m1+m2)*g*l1*sin(symx(2))-m2*g*l2*sin(symx(1)+symx(2));
g2=-m2*g*l2*sin(symx(1)+symx(2));
Gq=[g1;g2];
```

specify your initial and final condition.

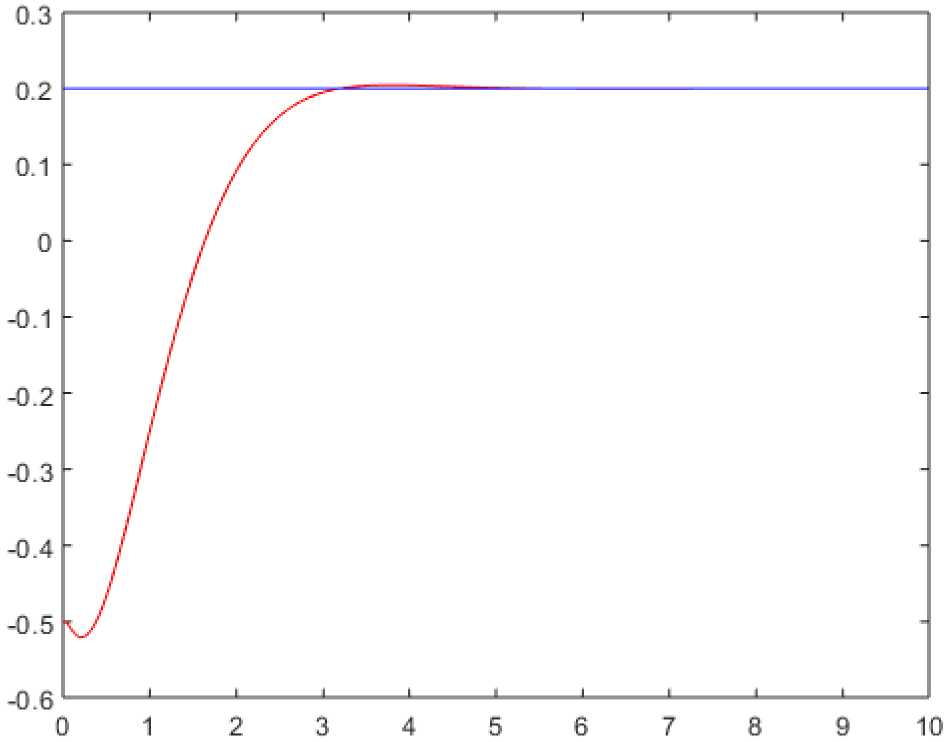
```
x0= [-0.5,0.2,0.1,0.1];
global w torque
w=0.2;
tf=10;
torque = [];
```

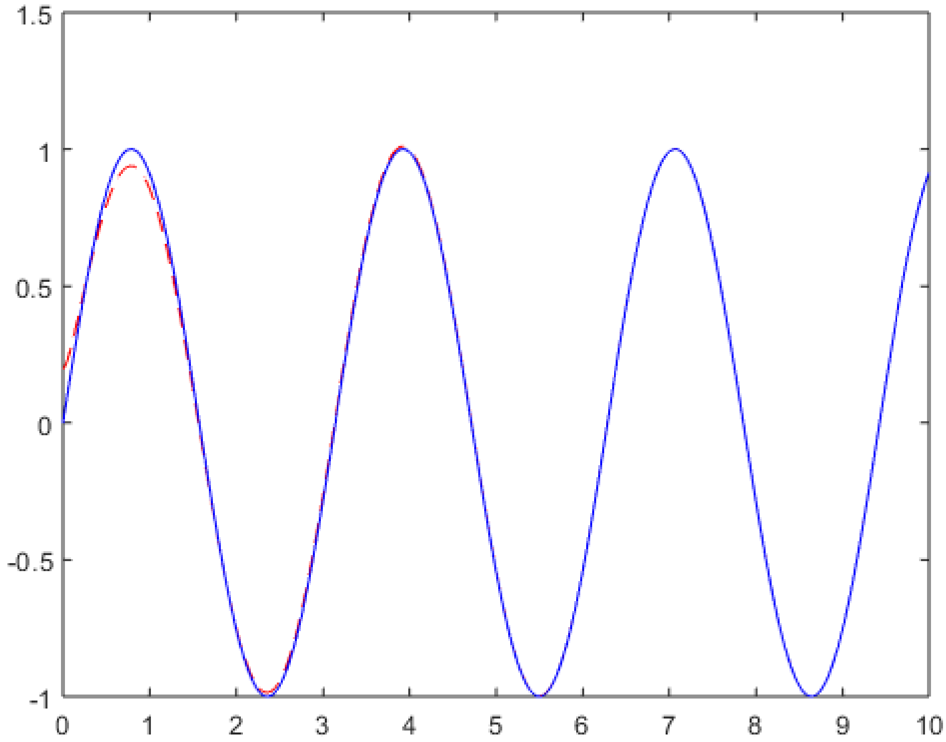
Implement the Augmented PD control for set point tracking.

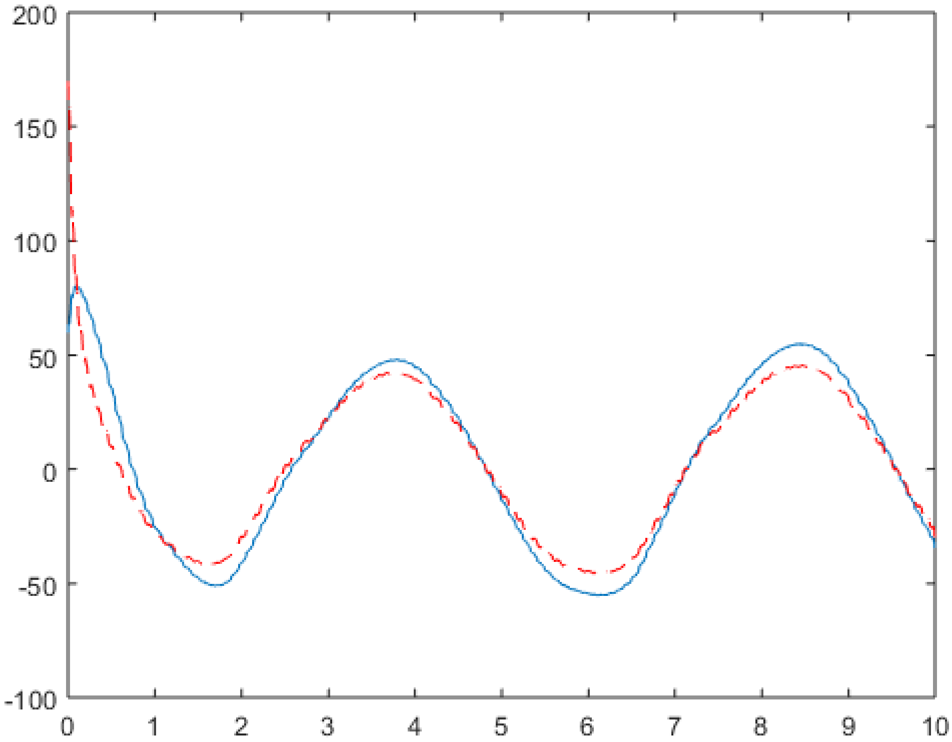
```
xf = [0, 0, 0, 0];
options = odeset('RelTol',1e-4,'AbsTol',[1e-4, 1e-4, 1e-4, 1e-4]);
[T,X] = ode45(@(t,x) AugmentedPDControl(t,x),[0 tf],x0, options);

figure('Name','Theta_1 under PD SetPoint Control');
plot(T, X(:,1),'r-');
hold on
plot(T, w*ones(size(T,1),1),'b-');
figure('Name','Theta_2 under PD SetPoint Control');
plot(T, X(:,2),'r--');
hold on
plot(T, sin(2*T),'b-');

figure('Name','Augmented PD Control');
plot(T, torque(1,1:size(T,1)),'-');
hold on
plot(T, torque(2,1:size(T,1)),'r--');
hold off
torque=[];
```

Augmented PD Control

```
function dx = AugmentedPDControl(t,x)
    w=0.2;
    theta_d=[w;sin(2*t)]; % [x1d;x2d] Desired trajectory
    dtheta_d=[0;2*cos(2*t)]; % [x1d_dot;x2d_dot]
    ddtheta_d=[0;-4*sin(2*t)]; % [x1d_ddot;x2d_ddot]
    theta=x(1:2,1); % [x1;x2]=[x(1);x(2)]
    dtheta=x(3:4,1); % [x1_dot;x2_dot]=[x(3);x(4)]

    global M C Gq M_d C_d Gq_d
    symx= sym('symx',[4,1]);
    M = subs(M, [symx(1);symx(2);symx(3);symx(4)], [x(1);x(2);x(3);x(4)]);
    C = subs(C, [symx(1);symx(2);symx(3);symx(4)], [x(1);x(2);x(3);x(4)]);
    Gq = subs(Gq, [symx(1);symx(2);symx(3);symx(4)], [x(1);x(2);x(3);x(4)]);
    M_d = subs(M, [symx(1);symx(2);symx(3);symx(4)], [theta_d(1);theta_d(2);dtheta_d(1);dtheta_d(2)]);
    C_d = subs(C, [symx(1);symx(2);symx(3);symx(4)], [theta_d(1);theta_d(2);dtheta_d(1);dtheta_d(2)]);
    Gq_d = subs(Gq, [symx(1);symx(2);symx(3);symx(4)], [theta_d(1);theta_d(2);dtheta_d(1);dtheta_d(2)]);
    invM = inv(M);
    invMC= inv(M)*C;

    tau = Controler(theta_d,dtheta_d,ddtheta_d,theta,dtheta);

    global torque
    torque = [torque, tau];

    dx = zeros(4,1);
    dx(1)= x(3); %dtheta1
    dx(2)= x(4); %dtheta2
    dx(3:4) = -invMC* x(3:4) + invM*tau;
end

function tau = Controler(theta_d,dtheta_d,ddtheta_d,theta,dtheta)
   P_e = theta_d - theta;
   V_e = dtheta_d - dtheta;
   Kp = 100*eye(2);
   Kv = 100*eye(2);

   global M C Gq_d
   tau = (Kp*P_e + Kv*V_e) + M*ddtheta_d  + C*dtheta_d ;
end
```


0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
1
2
3
4
5
6
7
8
9
10


1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10



Published with MATLAB® R2017b

Assignment 4

By- Aishwary Jagetia

Contents

Notations: For a given variable, x, dx is its time derivative, ddx is

2nd-order derivative.

```
clc
clear all;
close all;
% the following parameters for the arm
I1=10;  I2 = 10; m1=5; r1=.5; m2=5; r2=.5; l1=1; l2=1; g=9.8;

% we compute 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;
```

create symbolic variable for x.

x1 - theta1 x2 - theta2

```
symx= sym('symx',[4,1]);

global M C Gq dGq
M = [a+2*b*cos(symx(2)), d+b*cos(symx(2));
    d+b*cos(symx(2)), d];
C = [-b*sin(symx(2))*symx(4), -b*sin(symx(2))*(symx(3)+symx(4)); b*sin(symx(2))*symx(3),0];
invM = inv(M);
invMC= inv(M)*C;

% Gravity Matrix
g1=-(m1+m2)*g*l1*sin(symx(2))-m2*g*l2*sin(symx(1)+symx(2));
g2=-m2*g*l2*sin(symx(1)+symx(2));
Gq=[g1;g2];
dGq = [diff(Gq, symx(1)), diff(Gq, symx(2))];
```

specify your initial and final condition.

```
x0= [-0.5,0.2,0.1,0.1];
global i w torque
i = 0;
w = 0.2;
tf=10;
torque = [];
```

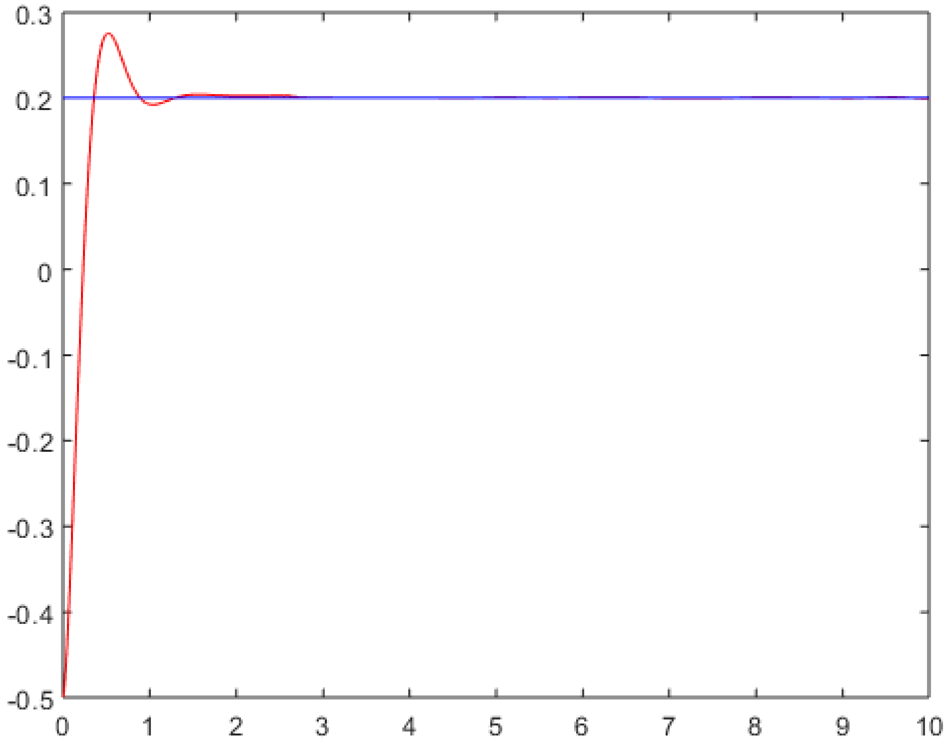
Implement the Iterative Learning control for set point tracking.

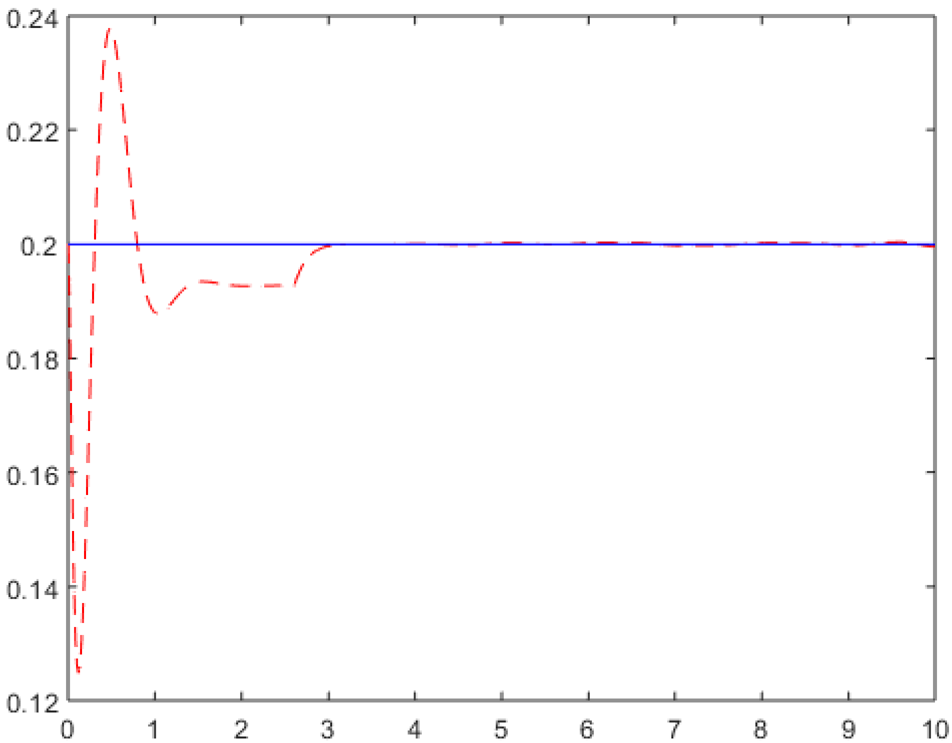
```
xf = [0, 0, 0, 0];
options = odeset('RelTol',1e-4,'AbsTol',[1e-4, 1e-4, 1e-4, 1e-4]);
[T,X] = ode45(@(t,x) IterativeControl(t,x),[0 tf],x0, options);

figure('Name','Theta_1 under PD SetPoint Control');
plot(T, X(:,1),'r-');
hold on
plot(T, w*ones(size(T,1),1),'b-');
figure('Name','Theta_2 under PD SetPoint Control');
plot(T, X(:,2),'r--');
hold on
plot(T, w*ones(size(T,1),1),'b-');

figure('Name','Torque: Augmented PD Control');
plot(T, torque(1,1:size(T,1)),'-');
hold on
plot(T, torque(2,1:size(T,1)),'r--');
hold off
torque=[];
```

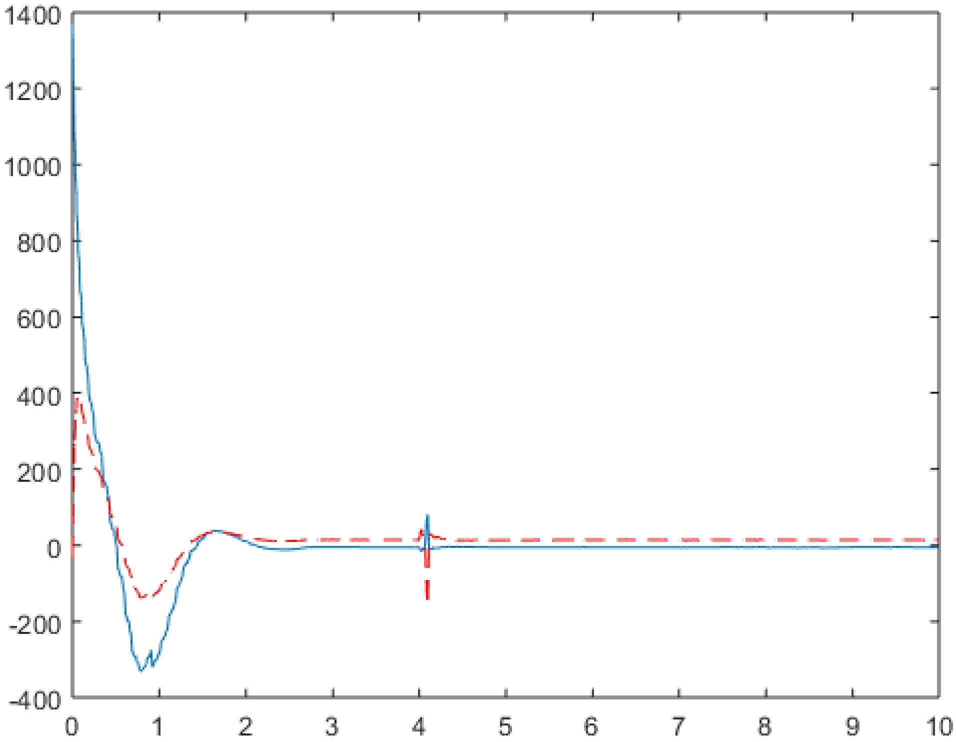
Iterative Learning control

```
function dx = IterativeControl(t,x)
    global i w
    w=0.2;
    theta_d=[w;w]; % [x1d;x2d] Desired trajectory
    dtheta_d=[0;0]; % [x1d_dot;x2d_dot]
    ddtheta_d=[0;0]; % [x1d_ddot;x2d_ddot]
    theta=x(1:2,1); % [x1;x2]=[x(1);x(2)]
    dtheta=x(3:4,1); % [x1_dot;x2_dot]=[x(3);x(4)]

    global M C Gq dGq
    symx= sym('symx',[4,1]);
    M = subs(M, [symx(1);symx(2);symx(3);symx(4)], [x(1);x(2);x(3);x(4)]);
    C = subs(C, [symx(1);symx(2);symx(3);symx(4)], [x(1);x(2);x(3);x(4)]);
    Gq = subs(Gq, [symx(1);symx(2);symx(3);symx(4)], [x(1);x(2);x(3);x(4)]);
    invM = inv(M);
    invMC= inv(M)*C;
    invMG= inv(M)*Gq;
    dGq = subs(dGq, [symx(1);symx(2)], [x(1);x(2)]);

    tau = Controler(theta_d,dtheta_d,ddtheta_d,theta,dtheta,t);

    global torque
    torque = [torque, tau];

    dx = zeros(4,1);
    dx(1)= x(3); %dtheta1
    dx(2)= x(4); %dtheta2
    dx(3:4) = -invMC* x(3:4) + invM*tau -invMG;
end

function tau = Controler(theta_d,dtheta_d,ddtheta_d,theta,dtheta,i)
   global u
   P_e = theta_d - theta;
   V_e = dtheta_d - dtheta; % dtheta_d = 0
%  Kp = double(simplify(dGq*eye(2)))
%  To calculate alpha value I have considered calculating partial
%  differentiation of Gq (Gravity Matrix) with respect to joint angles q,
%  which gives me dGq. The max of dGq is used for selecting the value of Kp
   Kp = 200*eye(2);
   Kv = 30*eye(2);
   beta = 0.1;
```

```
    if (i == 0)
        u = zeros(2,1);
    end
    tau = (1/beta)*(Kp*P_e + Kv*V_e) + u;
    if (norm(dtheta) < 0.0001)
        u = tau;
    end

end
```


0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
1
2
3
4
5
6
7
8
9
10


0.24
0.22
0.2
0.18
0.16
0.14
0.12
0
1
2
3
4
5
6
7
8
9
10



1400
1200
1000
800
600
400
200
0
-200
-400
0
1
2
3
4
5
6
7
8
9
10

Assignment #4

For the dynamic model:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = \tau$$

To find the bound on the gradient of the gravity Term

$$\left\| \frac{\partial N(q)}{\partial q} \right\| \leq \alpha$$

By differentiating gravity matrix w.r.t joint angles.

the max value of $\left\| \frac{\partial N(q)}{\partial q} \right\|$ was coming out to be around 160, which made me choose: $k_p = 200$ to also satisfy

$$\lambda_{min}(k_p) \geq \alpha$$

also,

$\beta = 0.1$ which satisfies $0 \leq \beta \leq \frac{1}{2}$.

Thus, the system converges with global asymptotic stability.