1. Reproduce the simulation in Example 3.4-1 of Lewis et al. That is, for the two link planar elbow arm, simulate in MATLAB the PD computed-torque controller. Use the same arm parameters and desired trajectory used there. Make your MATLAB M-file modular like the one in that example. Plot the same figures plotted there. Turn in your M-file.

2. Repeat the problem 1 simulation using the simplified inertia/gravity control law.

\[ \tau = M(q) \dddot{q} + K_v \dot{e} + K_p e + G(q) \]

which includes \( M(q) \) matrix and gravity but not the coriolis/centripetal terms.

3. Repeat the problem 1 simulation using the simplified control law.

\[ \tau = M(q) (K_v \dot{e} + K_p e) + G(q) \]

which includes \( M(q) \) matrix and gravity but not the acceleration feedforward or the coriolis/centripetal terms.

4. Repeat the problem 3 simulation using the control law.

\[ \tau = M(q) (K_v \dot{e} + K_p e + K_c \epsilon) + G(q) \]

where \( \epsilon(t) \) is the integral of tracking error \( e(t) \).

This is the code used for applying the ode23 function for different control strategies (1-4 problems).

Ctl_Option denotes the desired problem to being solve.

```matlab
function [x,y,t,X] = TLPlana2(Ctl_Option,x0,tf);
% Simulation of a Two-Link Planar robot
% Course: Robotics
% Written by Jose Mireles
% tf is simulation time (final time)
% x0(1) = theta1
% x0(2) = theta2
% x0(3) = (d/dt)theta1
% x0(4) = (d/dt)theta2
% x0(5) = torque input 1 (Initialy zero.- Only for Output purposes)
% x0(6) = torque input 2 (Initialy zero.- Only for Output purposes)
% x0(7) = tracking error 1 (Initialy zero.- Only for Output purposes)
% x0(8) = tracking error 2 (Initialy zero.- Only for Output purposes)
% 1 = Computed Torque Control. PD control
% 2 = Simplified Inertia/Centripetal PD Control (without Nonlinearities)
% 3 = Simplified Inertia/Centripetal PD Control (" & not acceleration feedforward)
% Try with initial conditions, and run it as:
% tf=10; x0=[0.1;0;0;0;0;0;0;0]; [x,y,t,X]=TLPlana2(1,x0,tf);

Amp1=0.1; Amp2=0.1;
L1=1;
L2=1;
t0=0;
tspan=linspace(t0,tf,1000); % A thousand samples from t=0 -> tf
figure(1)
figure(2)
figure(3)
figure(4)
figure(5)
figure(6)
title('TORQUE INPUTS f1 in blue, f2 in red')
ylabel('( time ) ')
xlabel('( Forces )')
hold on
figure(7)
title('TRACKING ERROR e1 in blue, e2 in red')
ylabel('( time ) ')
xlabel('( X )')
hold on

if Ctl_Option == 1
    [t,X]=ode23('Twolink2',tspan,x0);
    % Twolink2 => COMPUTED-TORQUE Control, PD
elseif Ctl_Option == 2
    [t,X]=ode23('Twolink3',tspan,x0);
    % Twolink2 => " without NONLinearities !
elseif Ctl_Option == 3
    [t,X]=ode23('Twolink4',tspan,x0);
    % Twolink2 => " without NonL & accel feedforward
else
    [t,X]=ode23('Twolink5',tspan,x0);
    % Twolink2 => PID Control by integration of tracking error!
end

figure(1)
plot(t,X(:,1),'b',t,X(:,2),'r'), grid on
ylabel('	heta angles')
xlabel('Time (seconds)')
title('Two Link Planar Robot: \theta1 in blue, \theta2 in red')
figure(2)
subplot(2,1,1)
plot(t,X(:,3),'b'), grid on
ylabel('(d/dt) \theta1')
xlabel('time')
title('theta velocities for the Two Link Planar Robot')
subplot(2,1,2)
plot(t,X(:,4),'r'), grid on
ylabel('(d/dt) \theta2')
xlabel('time')
figure(3)
subplot(2,1,1)
plot(X(:,1),X(:,3),'b'), grid on
ylabel('(d/dt) \theta1')
xlabel(' \theta1')
subplot(2,1,2)
plot(X(:,2),X(:,4),'r'), grid on
ylabel('(d/dt) \theta2')
xlabel(' \theta2')

% Plot of the desired trajectory:
figure(4)
qd1 = Amp1*sin(pi*t);
qd2 = Amp2*cos(pi*t);
xd=L1*cos(qd1)+L2*cos(qd1+qd2);
yd=L1*sin(qd1)+L2*sin(qd1+qd2);
plot3(xd,t,yd,'r'), grid on
axis([-2,2,0,10,-1,1])
view([2,-.8,1])
title('DESIRED displacement of the Tool in (X,Y) in blue')
ylabel('( time )')
xlabel('( X )')

% Plot of the REAL trajectory:
figure(5)
x=L1*cos(X(:,1))+L2*cos(X(:,1)+X(:,2));
y=L1*sin(X(:,1))+L2*sin(X(:,1)+X(:,2));
plot3(xd,t,yd,'r'), grid on
axis([-2,2,0,10,-1,1])
hold on
plot3(x,t,y,'b')

The code for the function needed to apply the PD computed-torque controller is the following:

```matlab
function Xdot = TwoLink2(t,X);
% Simulation of the control of a Two-Link Planar robot
% using Computed-Torque Control
%
% Course: Robotics
% Written by Jose Mireles Jr.
% X(1)= theta1
% X(2)= theta2
% X(3) = (d/dt)theta1
% X(4) = (d/dt)theta2
% X(5) = torque input 1 (Only for Output purposes)
% X(6) = torque input 2 (Only for Output purposes)
% X(7) = tracking error 1 (Only for Output purposes)
% X(8) = tracking error 2 (Only for Output purposes)

% Constant values for the desired trajectory:
Amp1=0.1; Amp2=0.1; per=1;

% Constant values of the controller:
kp=100; kv=20;

% Constant values of the Robot:
m1=1; m2=1; a1=1; a2=1; g=9.8;

% PART 1) COMPUTE DESIRED TRAJECTORY qd(t), qdp(t), qdpp(t)
fact=pi/per;
qd(1) = Amp1*sin(fact*t);
qd(2) = Amp2*cos(fact*t);
qd(1) = Amp1*cos(fact*t);
qd(2) = -Amp2*sin(fact*t);
qdpp(1) = -Amp1*(fact^2)*sin(fact*t);
qdpp(2) = -Amp2*(fact^2)*cos(fact*t);

% PART 2) COMPUTED-TORQUE CONTROLLER SUBROUTINE
% COMPUTE TRACKING ERRORS:
e(1) = qd(1)-X(1);
```
\[ e(2) = qd(2) - X(2); \]
\[ ep(1) = qdp(1) - X(3); \]
\[ ep(2) = qdp(2) - X(4); \]

% Adding these two tracking errors to the plot:
figure(7)
hold on
plot(t,e(1),'b',t,e(2),'r')

% COMPUTATION OF \( M(q) \)
\[ m11 = (m1 + m2) * a1^2 + m2*a2^2 + 2*m2*a1*a2*cos(X(2)); \]
\[ m12 = m2*a2^2 + m2*a1*a2*cos(X(2)); \]
\[ m21 = m12; \]
\[ m22 = m2*a2^2; \]

% COMPUTATION OF \( N(q,qp) \)
\[ N1 = -m2*a1*a2*(2*X(3)*X(4) + X(4)^2)*sin(X(2)); \]
\[ N2 = m2*a1*a2*X(3)^2*sin(X(2)); \]
\[ N1 = N1 + (m1 + m2)*g*a1*cos(X(1)) + m2*g*a2*cos(X(1) + X(2)); \]
\[ N2 = N2 + m2*g*a2*cos(X(1) + X(2)); \]

% COMPUTATION OF CONTROL TORQUES
\[ s1 = qdpp(1) + kv*ep(1) + kp*e(1); \]
\[ s2 = qdpp(2) + kv*ep(2) + kp*e(2); \]
\[ f1 = m11*s1 + m12*s2 + N1; \]
\[ f2 = m21*s1 + m22*s2 + N2; \]

% Adding these two forces to the plot:
figure(6)
hold on
plot(t,f1,'b',t,f2,'r')

% PART 3) ROBOT ARM DYNAMICS:
\[ \det = (m11*m22) - (m12*m21); \]

% Inverse Matrix
\[ I_{m11} = m22/\det; \]
\[ I_{m12} = m12/\det; \]
\[ I_{m21} = m21/\det; \]
\[ I_{m22} = m11/\det; \]
\[ Xdot = [X(3); X(4); (I_{m11}*(f1-N1) + I_{m12}*(f2-N2)); (I_{m21}*(f1-N1) + I_{m22}*(f2-N2)); (f1-X(5))/(.01); (f2-X(6))/(.01); (e(1)-X(7))/(.01); (e(2)-X(8))/(.01) ]; \]

% END OF FILE TwoLink2

Six plots were obtained for each problem. Such plots are: 1) Plot of \( \theta_1 \) and \( \theta_2 \) Vs. time, 2) Plot of \( (d/dt \theta_1) \) and \( (d/dt \theta_2) \) Vs. time, 3) Plots of Phase plane of \( \theta_1 \) and \( \theta_2 \), 4) Plot of the real Vs. desired trajectory of the tool in 3D, 5) Plot of torque inputs \( \tau_1 \) and \( \tau_2 \), and 6) Plot of tracking errors (desired minus real trajectory).
1) The plots found for the first problem were obtained by typing:

\[
tf=10; \ x0=[0.1;0;0;0;0;0;0;0]; \ [x,y,t,X]=TLP1ana2(1,x0,tf);
\]

Such plots are the following: (NOTICE THAT INITIAL CONDITIONS ARE NOT ZERO)
Plots for problem 1)
Plots for problem 1)
2) The plots found for the second problem were obtained by typing:
\[
tf=10; \ x0=[0.1;0;0;0;0;0;0;0]; \ [x,y,t,X]=TLPlana2(2,x0,tf);
\]
(NOTICE THAT INITIAL CONDITIONS ARE NOT ZERO)

For this problem, TLPlanar2 uses TwoLink3.m function, which only difference with respect to TwoLink2.m is the Computation of control torques part:

\[
\% \text{ COMPUTATION OF } N(q,qp)
\]
\[
N1c = (m1+m2)*g*a1*cos(X(1)) + m2*g*a2*cos(X(1)+X(2));
N2c = m2*g*a2*cos(X(1)+X(2));
N1 = N1c + -m2*a1*a2*(2*X(3)*X(4)+X(4)^2)*sin(X(2));
N2 = N2c + m2*a1*a2*X(3)^2*sin(X(2));
\]

\[
\% \text{ COMPUTATION OF CONTROL TORQUES}
\]
\[
s1 = qdpp(1) + kv*ep(1) + kp*e(1);
s2 = qdpp(2) + kv*ep(2) + kp*e(2);
f1 = m11*s1 + m12*s2 + N1c; \ % \text{ Only includes Gravity terms!}
f2 = m21*s1 + m22*s2 + N2c; \ % \text{ Only includes Gravity terms!}
\]

The plots for problem 2 were the following:
Plots for problem 2:

![Phase plots](image1)

![Real displacement of the Tool in (X,Y) in blue](image2)
Plots for problem 2)
3) The plots found for the third problem were obtained by typing:
\[
\text{tf} = 10; \quad \text{x0} = [0.1; 0; 0; 0; 0; 0; 0; 0]; \quad [x, y, t, X] = \text{TLPlana2}(3, x0, tf);
\]
(NOTICE THAT INITIAL CONDITIONS ARE NOT ZERO)

For this problem, TLPlanar2 uses TwoLink4.m function, which only difference with respect to TwoLink2.m is the Computation of control torques part:

\[
\begin{align*}
&\text{% COMPUTATION OF } N(q, q) \text{ for the controller and for the plant:} \\
&\quad \text{N1c} = (m_1+m_2)*g*a_1*cos(X(1))+m_2*g*a_2*cos(X(1)+X(2)); \\
&\quad \text{N2c} = m_2*g*a_2*cos(X(1)+X(2)); \\
&\quad \text{N1} = \text{N1c} + -m_2*a_1*a_2*(2*X(3)*X(4)+X(4)^2)*sin(X(2)); \\
&\quad \text{N2} = \text{N2c} + m_2*a_1*a_2*X(3)^2*sin(X(2)); \\

&\text{% COMPUTATION OF CONTROL TORQUES} \\
&\quad \text{s1} = kv*ep(1) + kp*e(1); \\
&\quad \text{s2} = kv*ep(2) + kp*e(2); \\
&\quad \text{f1} = m_{11}*s1 + m_{12}*s2 + \text{N1c}; \\
&\quad \text{f2} = m_{21}*s1 + m_{22}*s2 + \text{N2c};
\end{align*}
\]

The plots for problem 3 were the following:
Plots for problem 3:
Plots for problem 3)
4) The plots found for the fourth problem were obtained by typing:

\[
\begin{align*}
    \text{tf}=10; \ x_0=[0.1;0;0;0;0;0]; \ [x,y,t,X]=\text{TLPlanana2}(4,x_0,\text{tf}); \\
\end{align*}
\]

(NOTICE THAT INITIAL CONDITIONS ARE NOT ZERO)

For this problem, TLPlanar2 uses TwoLink5.m function, which has more differences with respect to the TwoLink2.m function. The entire code for TwoLink5.m is shown here:

```matlab
function Xdot = TwoLink5(t,X);
    % Simulation of the control of a Two-Link Planar robot
    % using PID Computed-Torque Control
    
    % Course: Robotics
    
    % Written by Jose Mireles Jr.
    
    % X(1)= theta1
    % X(2)= theta2
    % X(3) = (d/dt)theta1
    % X(4) = (d/dt)theta2
    % X(5) = torque input 1 (Only for Output purposes)
    % X(6) = torque input 2 (Only for Output purposes)
    % X(7) = tracking error 1
    % X(8) = tracking error 2
    
    % Constant values for the desired trajectory:
    Amp1=0.1; Amp2=0.1; per=1;
    
    % Constant values of the controller:
    kp=100; kv=20; ki=500;    % Ki < Kp*Kv
    
    % Constant values of the Robot:
    m1=1; m2=1; a1=1; a2=1; g=9.8;
    
    % PART 1) COMPUTE DESIRED TRAJECTORY qd(t), qdp(t), qdpp(t)
    fact=pi/per;
    qd(1) = Amp1*sin(fact*t);
    qd(2) = Amp2*cos(fact*t);
    qdp(1) = Amp1*fact*cos(fact*t);
    qdp(2) = -Amp2*fact*sin(fact*t);
    qdpp(1)= -Amp1*(fact^2)*sin(fact*t);
    qdpp(2)= -Amp2*(fact^2)*cos(fact*t);
    
    % PART 2) COMPUTED-TORQUE CONTROLLER SUBROUTINE
    
    % COMPUTE TRACKING ERRORS:
    e(1) = qd(1)-X(1);
    e(2) = qd(2)-X(2);
    ep(1) = qdp(1)-X(3);
    ep(2) = qdp(2)-X(4);
    Epsilon1 = X(7);    % Integral of the error 1!
    Epsilon2 = X(8);    % Integral of the error 2!
    
    % Adding these two tracking errors to the plot:
    figure(7)
    hold on
    plot(t,e(1),'b',t,e(2),'r')
    
    % COMPUTATION OF M(q)
```

University of Texas at Arlington
Jose Mireles Jr.
m11=(m1+m2)*a1^2+m2*a2^2+2*m2*a1*a2*cos(X(2));
m12=m2*a2^2+2*m2*a1*a2*cos(X(2));
m21=m12;
m22=m2*a2^2;

% COMPUTATION OF N(q,qp)
N1c = (m1+m2)*g*a1*cos(X(1))+m2*g*a2*cos(X(1)+X(2));
N2c = m2*g*a2*cos(X(1)+X(2));
N1 = N1c + -m2*a1*a2*(2*X(3)*X(4)+X(4)^2)*sin(X(2));
N2 = N2c + m2*a1*a2*X(3)^2*sin(X(2));

% COMPUTATION OF CONTROL TORQUES
s1 = kv*ep(1) + kp*e(1) + ki*Epsilon1;
s2 = kv*ep(2) + kp*e(2) + ki*Epsilon2;
f1 = m11*s1 + m12*s2 + N1c;   % Not having Non-Linear terms!
f2 = m21*s1 + m22*s2 + N2c;   % Not having Non-Linear terms!

% Adding these two forces to the plot:
figure(6)
hold on
plot(t,f1,'b',t,f2,'r')

% PART 3) ROBOT ARM DYNAMICS:
Det = (m11*m22)-(m12*m21);

% Inverse Matrix
Im11=m22/Det;
Im12=-m12/Det;
Im21=-m21/Det;
Im22=m11/Det;

% STATE EQUATIONS:
% X(1) = theta1
% X(2) = theta2
% X(3) = (d/dt)theta1
% X(4) = (d/dt)theta2
% X(5) = torque input 1 (Only for Output purposes)
% X(6) = torque input 2 (Only for Output purposes)
% X(7) = tracking error 1
% X(8) = tracking error 2

Xdot=[X(3);
      X(4);
      (Im11*(f1-N1)+Im12*(f2-N2));
      (Im21*(f1-N1)+Im22*(f2-N2));
      (f1-X(5))/(.01); % Does not integrates !
      (f2-X(6))/(.01); % Does not integrates !
      e(1) % Integrates !
      e(2) % Integrates !];

% End of File

% End of File

University of Texas at Arlington
Jose Mireles Jr. 14
The plots for problem 4) using $K_p=100$, $K_v=20$, $K_i=500$ were obtained by typing:

$$tf=10; \ x0=[0.1;0;0;0;0;0;0;0]; \ [x,y,t,X]=TLPlana2(4,x0,tf);$$

(Notice initial conditions are not zero)
Plots for problem 4)

(desired displacement in red)
Plots for problem 4)

**TORQUE INPUTS** f1 in blue, f2 in red

**TRACKING ERROR** e1 in blue, e2 in red