SOLUTIONS

Some theoretical parts and explanations. Published Matlab Code (which follows this in pdf) will contain complete answers and plots

a) For this specific problem, i.e., controlling the trajectory of the pelican robot, which is used as the legs of the quadruped robot, I used Computed Torque Control as the control law. Computed Torque Control is the technique of applying feedback linearization to nonlinear systems. It can handle uncertainties.

The following are the equations for Computed Torque Control:
Control Law-

$$\boldsymbol{\tau} = M(\boldsymbol{q})\left[\ddot{\boldsymbol{q}}_d + K_v\dot{\tilde{\boldsymbol{q}}} + K_p\tilde{\boldsymbol{q}}\right] + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$$

M(q) is the Inertia Matrix
C(q,qdot) is the Coriolis Matrix
G(q) is the gravity matrix

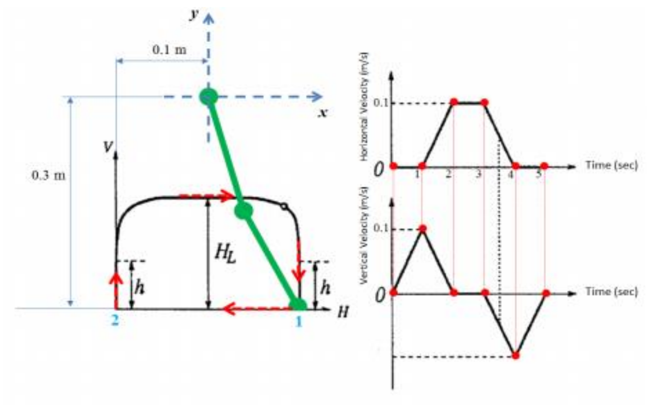
$\ddot{\boldsymbol{q}}_d$ is the acceleration of the joint angles in the desired configuration.

$\dot{\tilde{\boldsymbol{q}}}$ is the error in the joint velocities(angular velocities).

$\tilde{\boldsymbol{q}}$ is the error in joint angles
Here, error means “desired minus the actual”.

I thought this was an appropriate choice because the goal in the problem is to perform motion control, and computed torque control is a good choice for that.

b)-h) To solve these parts, the trajectory of the robot must be specified. The given velocity-time graphs are-



(a) Trajectory Plane (b) Velocity Template

From point 1 to point 2,

→ when $0 \leq t \leq 10$

velocity in x, $V_x = -\frac{0.2}{10} = 0.02$ m/s

velocity in y, $V_y = 0$ m/s.

→ when $10 \leq t \leq 11$

$V_x = 0$

From the graphs, we can say -

$V_y = 0.1(t-10)$

x desired, $x_d = -0.1$

y desired, $y_d = -0.3 + 0.5(t-10)V_y \rightarrow (1)$

→ finding h

substituting $t = 11$ in (1), we get

$y_d = -0.3 + 0.5 \times 1 \times 0.1$

$= -0.3 + 0.05$

$=$ ~~-0.25~~ ~~-0.25~~ -0.25

so, $h = -0.3 - (-0.25)$

$= -0.3 + 0.25$

$= -0.05$

$|h| = 0.05$

$\Rightarrow$ When $11 \le t \le 12$

$$v_x = 0.1(t-10-1)$$

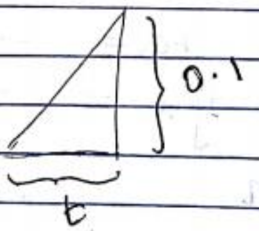
$$v_y = -0.1(t-10-2)$$

$$x_d = -0.1 + \left[0.5(t-11)v_x\right]$$

$$y_d = -0.25 + \left((0.5 \times 0.1) - (0.5 \times (12-t)v_y)\right)$$

Here, the distances travelled are calculated using AREAS under the velocity time graphs.

eg:



distance travelled = Area of triangle

$$= \frac{1}{2} \times t \times 0.1$$

Using the concept of Areas under curves, the remaining trajectory is written down

[NOTE - Trajectory equations are specified in the code]

→ From those equations, we obtain H_L.

$$H_L = -0.3 - -0.2$$
$$= -0.3 + 0.2$$
$$= -0.1$$
$$|H_L| = 0.1$$

g) To plot the end effector velocities, the following formula was used-

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \text{Jacobian} \times \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

The Jacobian marix was calculated.

(Note- The published Code Follows with Complete Solutions and Plots)

```
%RBE 502-Final Exam
%Motion Control of a Walking Robot's Leg
function robot_control_final_exam_Namrita_Madhusoodanan()
clc
clear all
close all
%Parameters of the Pelican Robot
% I1=0.1213;  I2 = 0.0116; m1=6.5225; r1=0.0983; m2=2.0458; r2=0.0229;
 l1=0.26; l2=0.26;
% g=9.81;
% a = I1+I2+m1*r1^2+ m2*(l1^2+ r2^2);
% b = m2*l1*r2;
% d = I2+ m2*r2^2;
%Time Span of Both Phases Together
tS=[0 15];
X0=initRobotConfig(0.1,-0.3);
Kp=[45 0;0 45];
Kv=[10 0;0 10];
[T Y]=ode45(@(t,x) robot(t,x,Kp,Kv),tS,X0);
%b)Generate a plot of the joint angles versus time corresponding to
 the closed-loop system for
%both phases i.e. 15 seconds in a single graph.
figure(1);
plot(T,Y(:,3),T,Y(:,4));
title('Joint Angles over Time')
xlabel('Time(sec)');
ylabel('Joint Angles(rad)');
legend('\q1','\q2');
print('-f1','Angles','-dpng')
%c)Generate a plot of the joint velocities versus time corresponding
 to the closed-loop system for
%both phases i.e. 15 seconds in a single graph.
figure(2);
plot(T,Y(:,5),T,Y(:,6));
title('Joint Velocities over Time')
xlabel('Time(sec)');
ylabel('Angular Velocities(rad/s)');
legend('\omega_1','\omega_2');
print('-f2','Velocities','-dpng')
%d)Generate a plot of the control input as a function of time for both
 phases in a single graph.
T1=diff(Y(:,7))./diff(T);
T2=diff(Y(:,8))./diff(T);
tt=0:(T(end)/(length(T1)-1)):T(end);
figure(3);
plot(tt,T1,tt,T2);
axis([0 20 -100 100]);
title('Control Input as a function of time')
xlabel('Time(sec)');
ylabel('Joint Torques(Nm)');
legend('\tau_1','\tau_2');
print('-f3','ControlInput','-dpng')
```

```
%e)Generate a plot of the resultant end-effector’s path (foot tip’s
 path) in x-y plane. Show this
%graph on top of the above figure (a) i.e. the trajectory plane for
 comparison purpose.
%Solution:
temp_x12=[];
temp_y12=[];
for iterator=1:length(Y)
    ret_x=getForwardKinematics(Y(iterator,3),Y(iterator,4));
    temp_x12=[temp_x12;ret_x(1)];
    temp_y12=[temp_y12;ret_x(2)];
end
XD=[];
YD=[];
VX=[];
VY=[];
for i=1:length(Y)
    if T(i)<=10
        xdes=0.1-0.02*T(i);
        ydes=-0.3;
        vxdes=0.02;
        vydes=0;
    elseif ((T(i)>10)&(T(i)<=11))
        vxdes=0;
        vydes=0.1*(T(i)-10);
        xdes=-0.1;
        ydes=-0.3+0.5*(T(i)-10)*vydes;
    elseif ((T(i)>11)&(T(i)<=12))
        vxdes=0.1*(T(i)-10-1);
        vydes=-0.1*(T(i)-10-2);
        xdes=-0.1+(0.5*(T(i)-11)*vxdes);
        ydes=-0.25+((0.5*0.1)-(0.5*(12-T(i))*vydes));
    elseif ((T(i)>12)&(T(i)<=13))
        ydes=-0.2;
        vxdes=0.1;
        vydes=0;
        xdes=-0.05+(T(i)-12)*vxdes;
    elseif ((T(i)>13)&(T(i)<=14))
        vxdes=-0.1*(T(i)-10-4);
        xdes=0.05+((0.5*0.1)-(0.5*(14-T(i))*vxdes));
        vydes=-0.1*(T(i)-10-3);
        ydes=-0.2+(0.5*(T(i)-13)*vydes);
    elseif (T(i)<=15)
        xdes=0.1;
        vsdes=0;
        vydes=0.1*(T(i)-10-5);
        ydes=-0.25-((0.5*0.1)-(0.5*(15-T(i))*-1*vydes));
    end
    XD=[XD;xdes];
    YD=[YD;ydes];
    VX=[VX;vxdes];
    VY=[VY;vydes];
end
figure(4)
```

```
plot(temp_x12,temp_y12,'--','LineWidth',3);
hold on
plot(XD,YD, 'LineWidth', 2);
title('End-effector’s path (foot tip’s path) in x-y plane and Desired
 Path');
xlabel('x axis (m)') ;
ylabel('y axis(m)') ;
legend('Actual','Desired');
print('-f4','Path(Desired-Actual','-dpng')
%f)Plot end effector position as a function of time
figure(5);
plot(T,temp_x12,T,temp_y12);
title('End effector position as a function of time')
xlabel('Time(sec)');
ylabel('Position(m)');
legend('x','y');
print('-f5','EndPose','-dpng')

%g)Generate a plot of the end-effector velocity (both horizontal and
 vertical velocities) as a
%function of time for both phases i.e. 15 seconds in a single graph.
VEX=[];
VEY=[];
for i=1:length(Y)
    t1=Y(i,3);
    t2=Y(i,4);
    t1dot=Y(i,5);
    t2dot=Y(i,6);
    l1=0.26; l2=0.26;
    J=[-l1*sin(t1)-l2*sin(t1+t2) -l2*sin(t1+t2);
        l1*cos(t1)+l2*cos(t1+t2) l2*cos(t1+t2)];
    velocity=J*[t1dot;t2dot];
    VEX=[VEX;velocity(1)];
    VEY=[VEY;velocity(2)];
end
figure(6)
plot(T,VEX,T,VEY);
title('End-effector velocity (both horizontal and vertical
 velocities)as a function of time')
xlabel('Time(sec)');
ylabel('Velocitie(m/s)');
legend('vx','vy');
print('-f6','EndVelxy','-dpng')



%h)Stick Model of Robot Performing the Task
%Set up first frame of Animation
figure(8);
clf('reset') ;
l1=0.26; l2=0.26;
subplot(2 ,1 ,1) ;
plot(T,temp_x12,T,temp_y12);
```

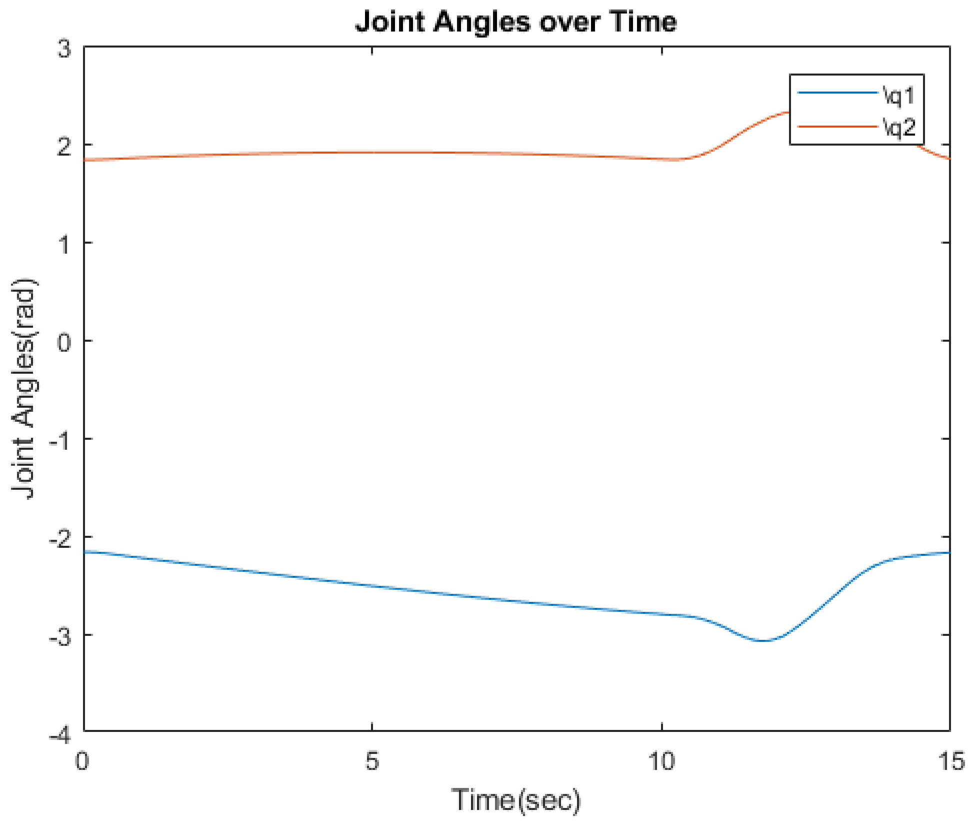
```
hh1(1) =
 line(T(1),temp_x12(1,1),'Marker','.','MarkerSize',20,'Color','b');
hh1(2) =
 line(T(1),temp_y12(1,1),'Marker','.','MarkerSize',20,'Color','y');

xlabel('time(sec)');
ylabel('Position');

subplot(2,1,2);
hh2 = line([0, 1*cos(Y(1,1))],[0, 1*sin(Y(1,1))]);
axis equal;
axis([-0.35 0.35 -0.4 0.15]);

ht = title(sprintf('Time:%0.2f sec\nPosition: %0.2f \nAngle:
%0.2f(Radian)%0.2f(Degree)',T(1),temp_x12(1,1),temp_y12(1,1),...
rad2deg(temp_y12(1,1)))) ;
ylabel('Pelican Robot Computed Torque Control');

for id = 1:length(T)
    set(hh1(1),'XData',T(id),'YData',temp_x12(id,1));
    set(hh1(2),'XData',T(id),'YData',temp_y12(id,1));

    tempx1 = l1*cos(Y(id,3));
    tempy1 = l1*sin(Y(id,3));

    tempx2 = (l1*cos(Y(id ,3))) + (l1*cos(Y(id ,3)+Y(id,4)));
    tempy2 = (l1*sin(Y(id ,3))) + (l1*sin(Y(id ,3)+Y(id,4)));

    set(hh2(1),'XData',[0,tempx1,tempx2],...
        'YData',[0,tempy1,tempy2]);
    set(ht,'String',sprintf('Time:%0.2f sec \n Angle:%0.2f (Radian)
 %0.2f (Degree)’ ,',[T(id),Y(id,3),rad2deg(Y(id,3))])) ;
    f = getframe(gcf);
    if  id == 1
        [mov(:,:,1,id),map] = rgb2ind(f.cdata,256,'nodither');
    else
       mov(:,:,1,id) = rgb2ind(f.cdata,map,'nodither');

    end
end

imwrite(mov,map,'stickAnimation.gif','DelayTime',0,'LoopCount',inf);
end
function[qd] = getInverseKinematics(x,y)
        l1=0.26; l2=0.26;
        q2 = acos((x*x + y*y - l1*l1 - l2*l2)/(2*l1*l2)) ;
        q1 = atan2(y,x) - atan2((l2*sin(q2)),(l1+l2*cos(q2)));
        qd = [q1;q2;] ;
end
function[X] = getForwardKinematics(q1,q2)
        l1=0.26; l2=0.26;
        x = l1*cos(q1) + l2*cos(q1 + q2) ;
        y = l1*sin(q1) + l2*sin(q1 + q2) ;
        X = [x ;y ;] ;
```

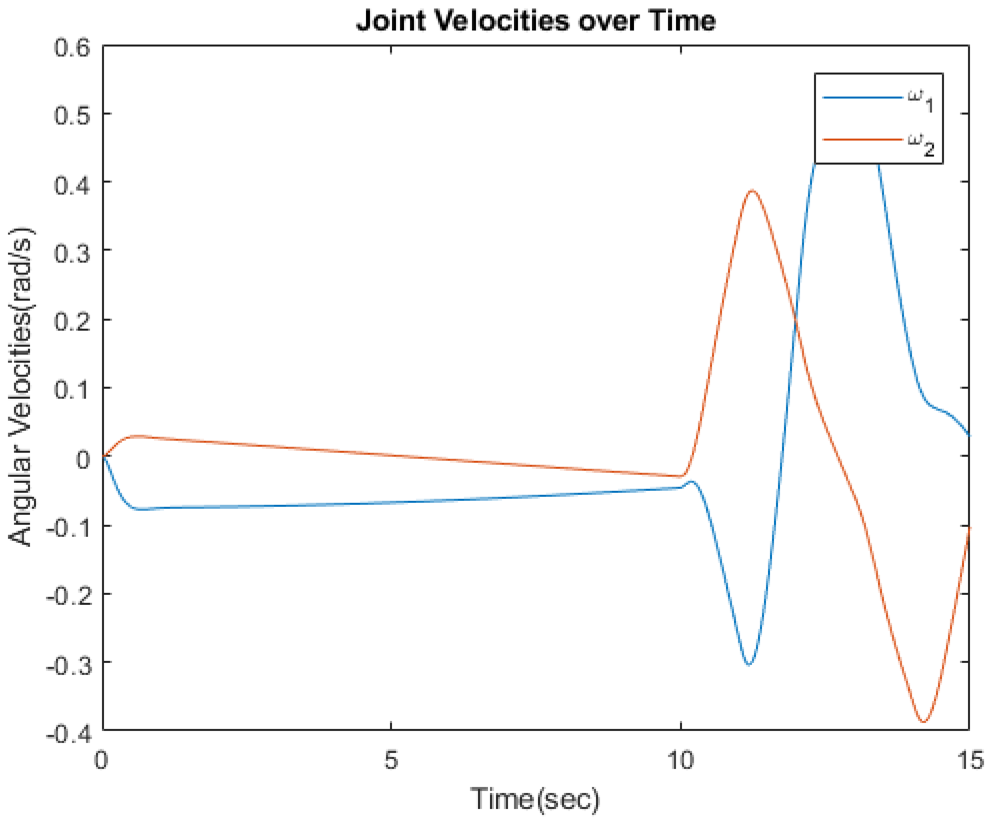
```
end
function xdot=robot(t,x,Kp,Kv)
    I1=0.1213;  I2 = 0.0116; m1=6.5225; r1=0.0983; m2=2.0458;
 r2=0.0229; l1=0.26; l2=0.26;
    g=9.81;
    a = I1+I2+m1*r1^2+ m2*(l1^2+ r2^2);
    b = m2*l1*r2;
    d = I2+ m2*r2^2;
 %State variables
 q1=x(3);%theta1
 q2=x(4);%theta2
 q3=x(5);%theta1dot
 q4=x(6);%theta2dot
 theta=[q1;q2];
 dtheta=[q3;q4];
 global M C G
 M=[a+2*b*cos(q2), d+b*cos(q2);  d+b*cos(q2), d];
 C=[-b*sin(q2)*q4, -b*sin(q2)*(q3+q4); b*sin(q2)*q3,0];
 G=[(m1*r1 + m2*l1)*g*sin(q1) +
 m2*r2*g*sin(q1+q2);m2*r2*g*sin(q1+q2)];
 %Initialize xdot
 xdot=zeros(12,1);
 tau=[0;0;];
 %Desired Trajectory Componenets
 if t<=10
     xd=0.1-0.02*t;
     yd=-0.3;
     qd=getInverseKinematics(xd,yd);
     qdDot=[0;0];
     qdDDot=[0;0];
 elseif ((t>10)&(t<=11))
     vx=0;
     vy=0.1*(t-10);
     xd=-0.1;
     yd=-0.3+0.5*(t-10)*vy;
     qd=getInverseKinematics(xd,yd);
     qdDot=[0;0];
     qdDDot=[0;0];
 elseif ((t>11)&(t<=12))
     vx=0.1*(t-10-1);
     vy=-0.1*(t-10-2);
     xd=-0.1+(0.5*(t-11)*vx);
     yd=-0.25+((0.5*0.1)-(0.5*(12-t)*vy));
     qd=getInverseKinematics(xd,yd);
     qdDot=[0;0];
     qdDDot=[0;0];
 elseif ((t>12)&(t<=13))
     yd=-0.2;
     vx=0.1;
     xd=-0.05+(t-12)*vx;
     qd=getInverseKinematics(xd,yd);
     qdDot=[0;0];
     qdDDot=[0;0];
 elseif ((t>13)&(t<=14))
```

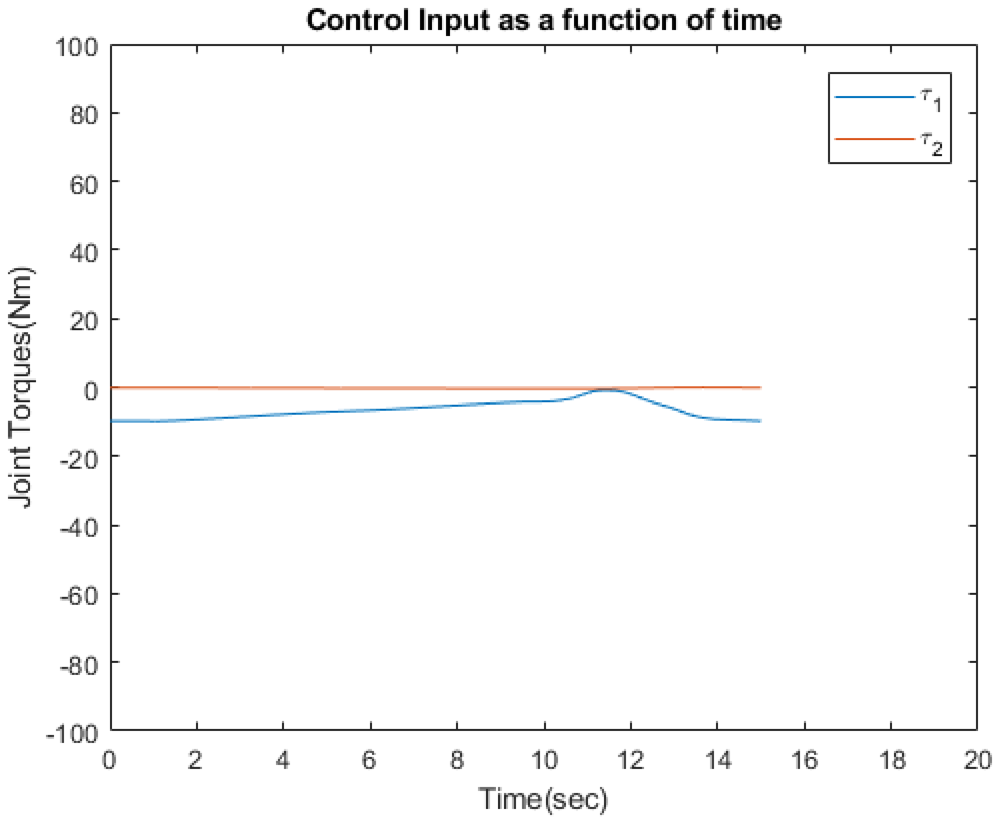
```
        vx=-0.1*(t-10-4);
        xd=0.05+((0.5*0.1)-(0.5*(14-t)*vx));
        vy=-0.1*(t-10-3);
        yd=-0.2+(0.5*(t-13)*vy);
        qd=getInverseKinematics(xd,yd);
        qdDot=[0;0];
        qdDDot=[0;0];
    elseif (t<=15)
        xd=0.1;
        vy=0.1*(t-10-5);
        yd=-0.25-((0.5*0.1)-(0.5*(15-t)*-1*vy));
        qd=getInverseKinematics(xd,yd);
        qdDot=[0;0];
        qdDDot=[0;0];
    end

     e=qd-theta; % position error
     de = qdDot-dtheta; % velocity error
     tau= M*(Kp*e + Kv*de) + C*dtheta + M*qdDDot+G;
     qddot=inv(M)*(tau-C*[q3;q4]-G);

    xdot(1)=qd(1)-x(3);
    xdot(2)=qd(2)-x(4);
    xdot(3)=x(5);
    xdot(4)=x(6);
    xdot(5)=qddot(1);
    xdot(6)=qddot(2);
    xdot(7)=tau(1);
    xdot(8)=tau(2);
    xdot(9)=qd(1)-q1;
    xdot(10)=qd(2)-q2;
    xyErr=getForwardKinematics(qd(1),qd(2))-getForwardKinematics(q1,q2);
    xdot(11)=xyErr(1);
    xdot(12)=xyErr(2);
   end
   function X = initRobotConfig(xi,yi)
           X = zeros(12,1) ;
           X(1) = xi ;
           X(2) = yi ;
           qi = getInverseKinematics(xi,yi);
           X(3) = qi(1) ;
           X(4) = qi(2) ;
           X(5) = 0 ;
           X(6) = 0 ;
           X(7) = 0 ;
           X(8) = 0 ;
           X(9)=0;
           X(10)=0;
           X(11)=0;
           X(12)=0;

    end
```

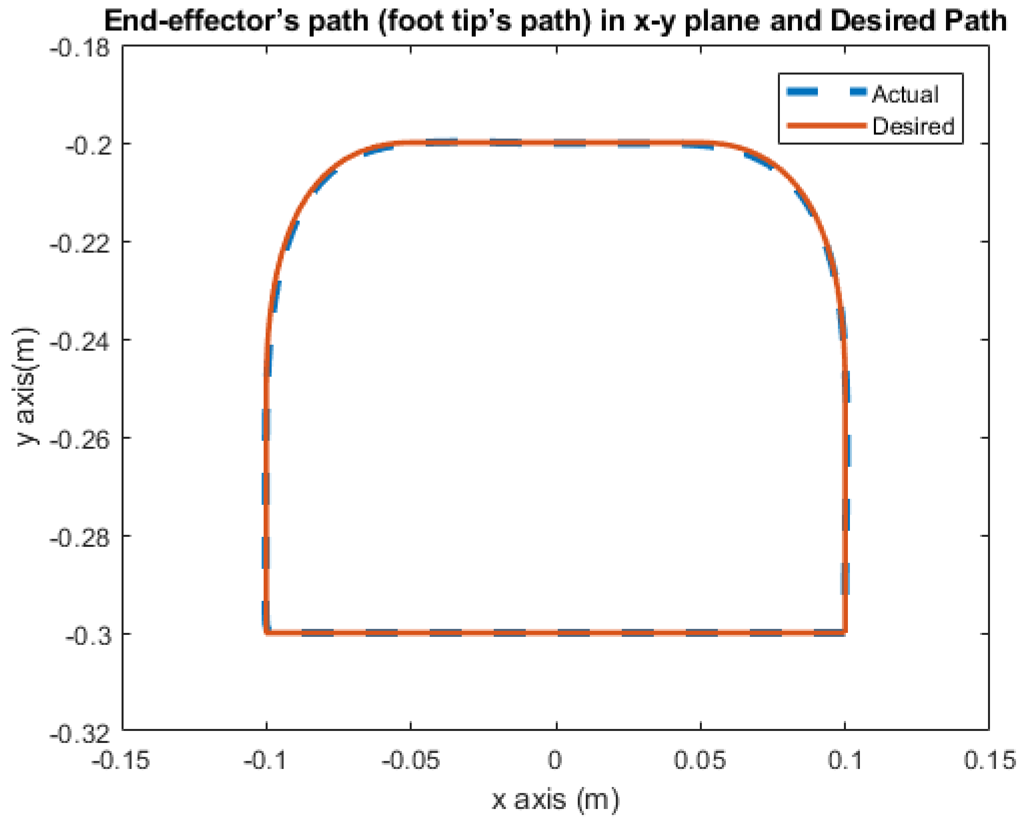
Warning: Error updating Legend.

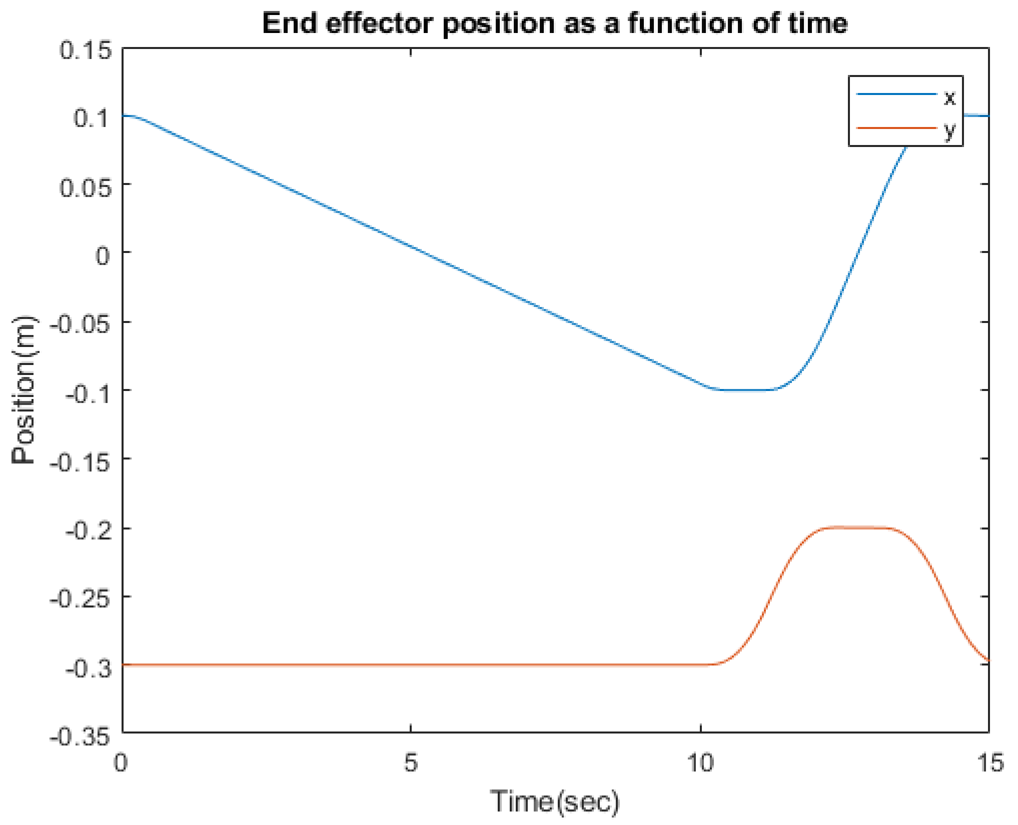
```
 Character vector must have valid interpreter syntax:
\q1

Warning: Error updating Legend.

 Character vector must have valid interpreter syntax:
\q2
```

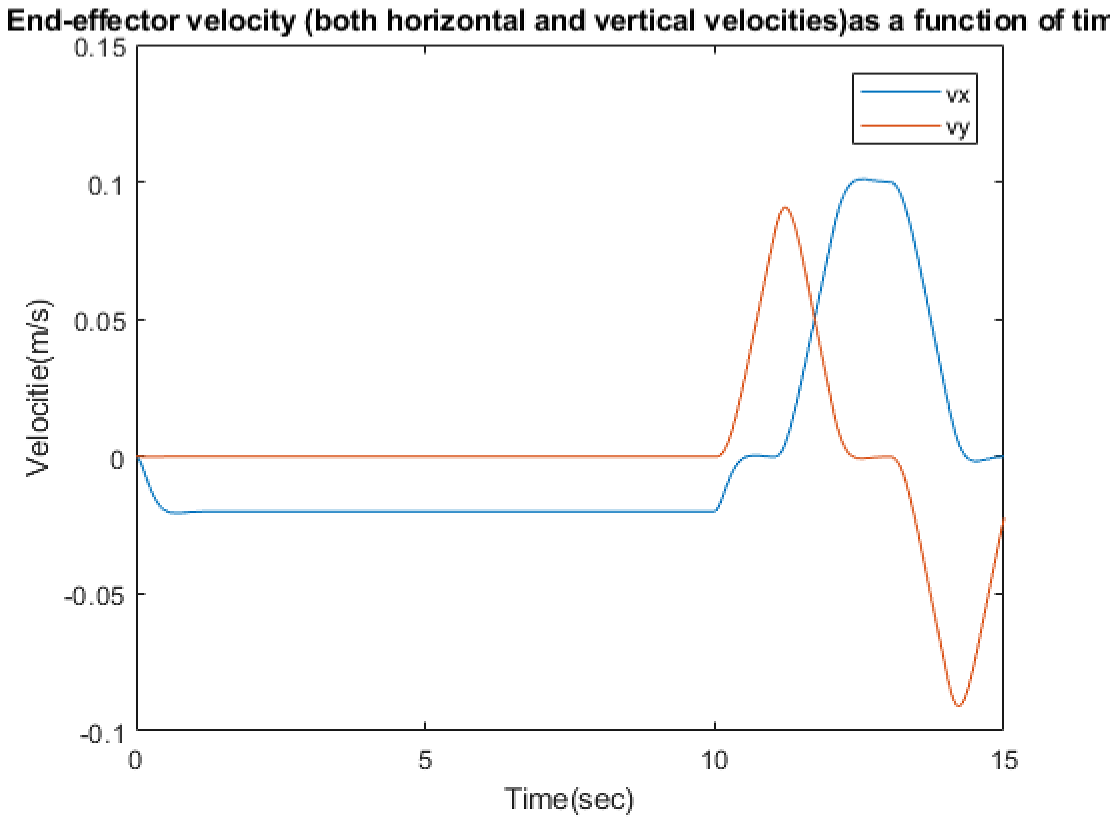


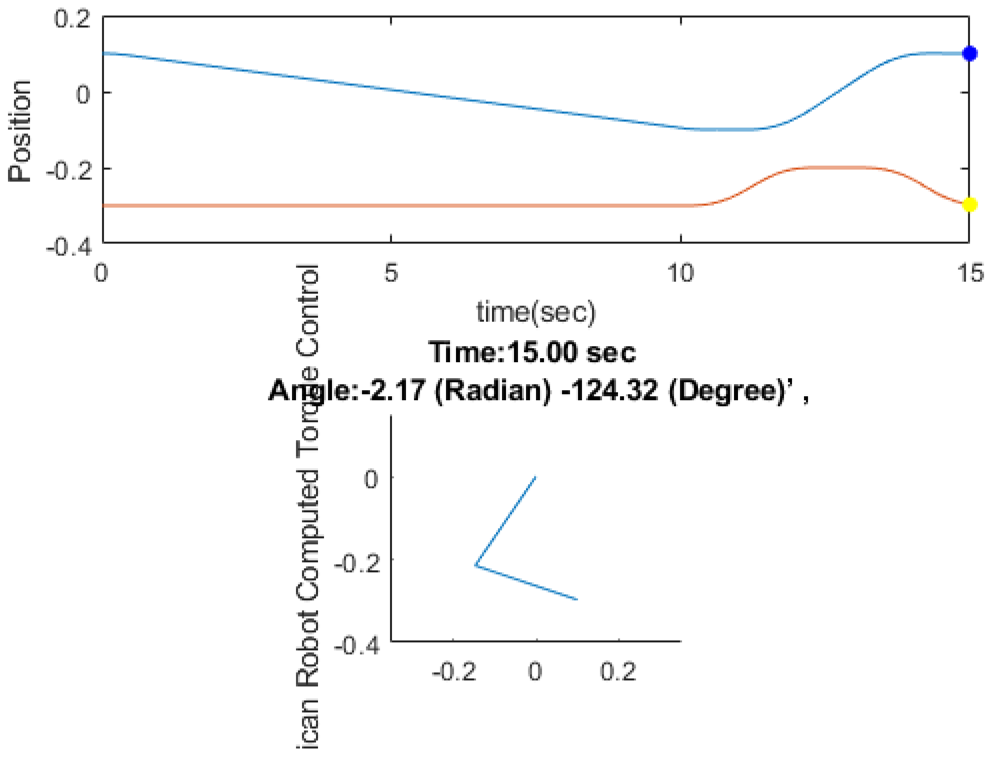

Joint Velocities over Time
Angular Velocities(rad/s)
Time(sec)
ω_1
ω_2
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
5
10
15


Control Input as a function of time
Joint Torques(Nm)
Time(sec)
τ_1
τ_2
100
80
60
40
20
0
-20
-40
-60
-80
-100
0
2
4
6
8
10
12
14
16
18
20


End-effector's path (foot tip's path) in x-y plane and Desired Path
Actual
Desired
y axis(m)
x axis (m)
-0.18
-0.2
-0.22
-0.24
-0.26
-0.28
-0.3
-0.32
-0.15
-0.1
-0.05
0
0.05
0.1
0.15


End effector position as a function of time
x
y
Position(m)
Time(sec)
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
0
5
10
15


End-effector velocity (both horizontal and vertical velocities)as a function of tim
vx
vy
Velocitie(m/s)
Time(sec)
0.15
0.1
0.05
0
-0.05
-0.1
0
5
10
15


Position
0.2
0
-0.2
-0.4
0
5
10
15
time(sec)
Time:15.00 sec
Angle:-2.17 (Radian) -124.32 (Degree)' ,
ican Robot Computed Torque Control
0
-0.2
-0.4
-0.2
0
0.2

Published with MATLAB® R2017b