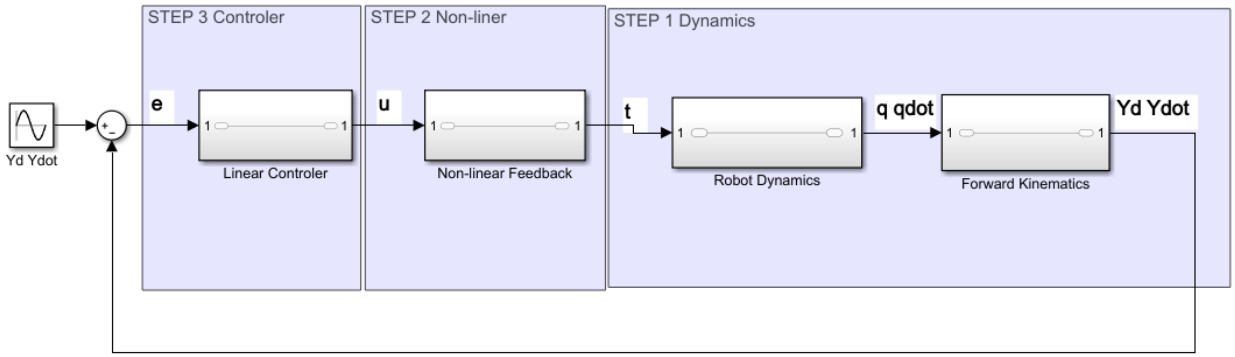
Final Project Report

Name Mikhail Podvalkov

1 Block diagram of the project.



2 Step 1 Symulate Dynamics

The first part of the assignment is to transform dynamics parameters from the task to code (Appendix 1.1) and forward kinematics (Appendix 1.2). The main idea of the step consists of simulating the dynamics of the robot. Vector τ is the input of the robot on this step. And the output is vectors of q and $\dot{q}$. It is a joint's angles and velocity, respectively. Find dynamic equations.

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

Transform it

$$\ddot{q} = D(q)^{-1}(\tau - C(q,\dot{q})\dot{q} + g(q))$$

And also, note that

$$\ddot{q} = \dot{q}$$

To solve the system of differential equations, use ode45. The code is in Appendix (1.3). Stimulate the system on the incoming signal $\tau=[1, 0, 0]^T$. analyze the received graphics of q and position. Based on the input data, the output signal is a mechanical force on the first joint. Therefore, it will change the only angle of the first joint Fig.(1.1).

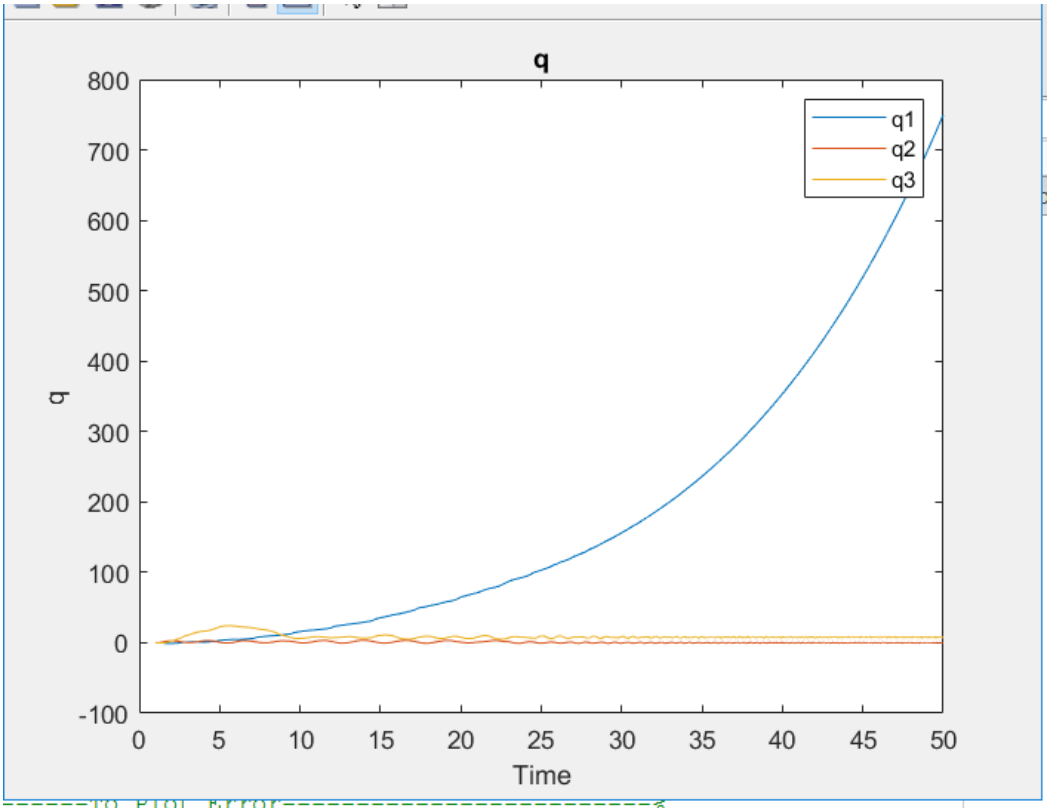


Fig. 1.1 $q_1(t)$, . $q_2(t)$, . $q_3(t)$

The end-efforts movement is in the X Y plane and on the circle because of the one joint design of the PUMA560 Fig(1.2).

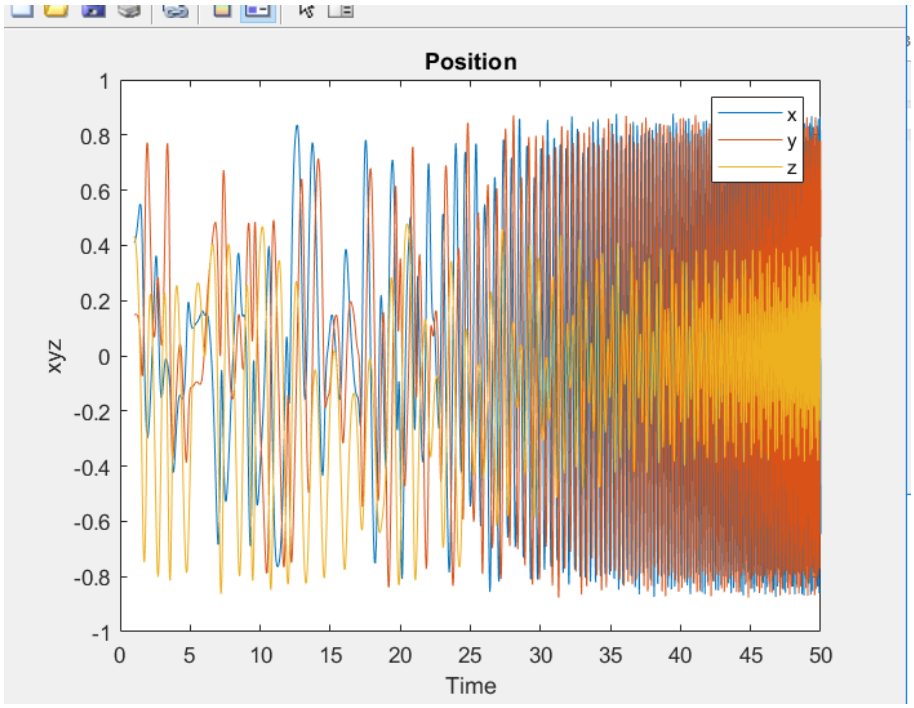


Fig 1.2 position x,y,z

3 Step 2 implement a nonlinear controller

Implement a nonlinear controller in the model for this using formula

$$\tau = D(q)J_h(q)^{-1}(u - J_h\dot{(q)}\dot{q}) + C(q,\dot{q})\dot{q} + g(q)$$

to this formula calculated by using $J_h(q)$ and$J_h\dot{(q)}$.

$$h(q) = \begin{bmatrix} a_3c_1c_{23} + d_4c_1s_{23} + a_2c_1c_2 - d_2s_1 \\ a_3s_1c_{23} + d_4s_1s_{23} + a_2s_1c_2 + d_2c_1 \\ -a_3s_{23} + d_4c_{23} - a_2s_2 \end{bmatrix}$$

$$J_h(q) = \begin{bmatrix} \frac{\partial h_1}{\partial q_1} & \frac{\partial h_1}{\partial q_2} & \frac{\partial h_1}{\partial q_3} \\ \frac{\partial h_2}{\partial q_1} & \frac{\partial h_2}{\partial q_2} & \frac{\partial h_2}{\partial q_3} \\ \frac{\partial h_3}{\partial q_1} & \frac{\partial h_3}{\partial q_2} & \frac{\partial h_3}{\partial q_3} \end{bmatrix};$$

$$J_h\dot{(q)} = \frac{d\,J_h(q)}{dt}$$

The code is presented in Appendix 2.1. After which add the resulting mathematical part to the model. test the resulting part on the input signal u = [1, 0, 0]^T , u = [0, 1, 0]^T , u = [0, 0, 1]^T

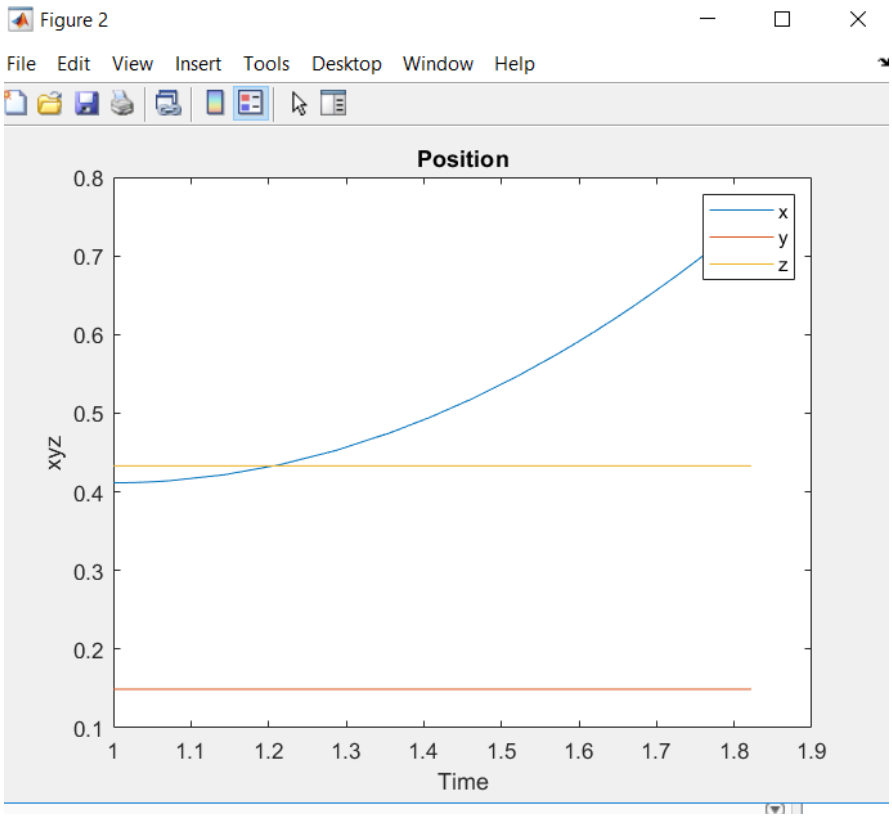


Fig 2. 1 u = [1, 0, 0]

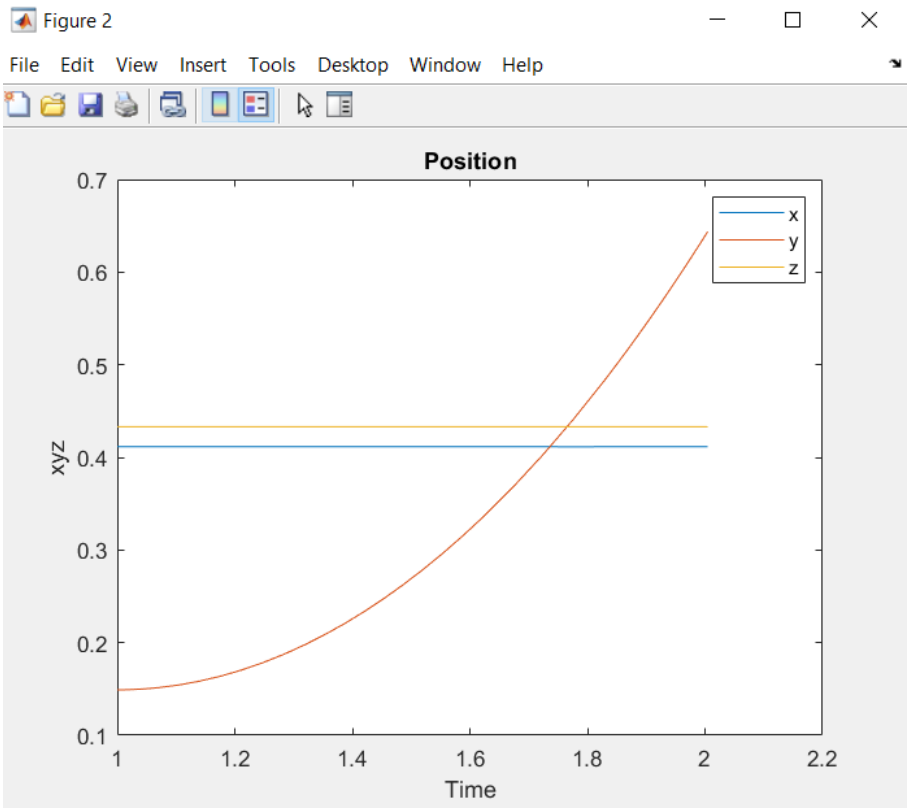


Fig 2.2 Fig 2. 1 u = [0, 2, 0]

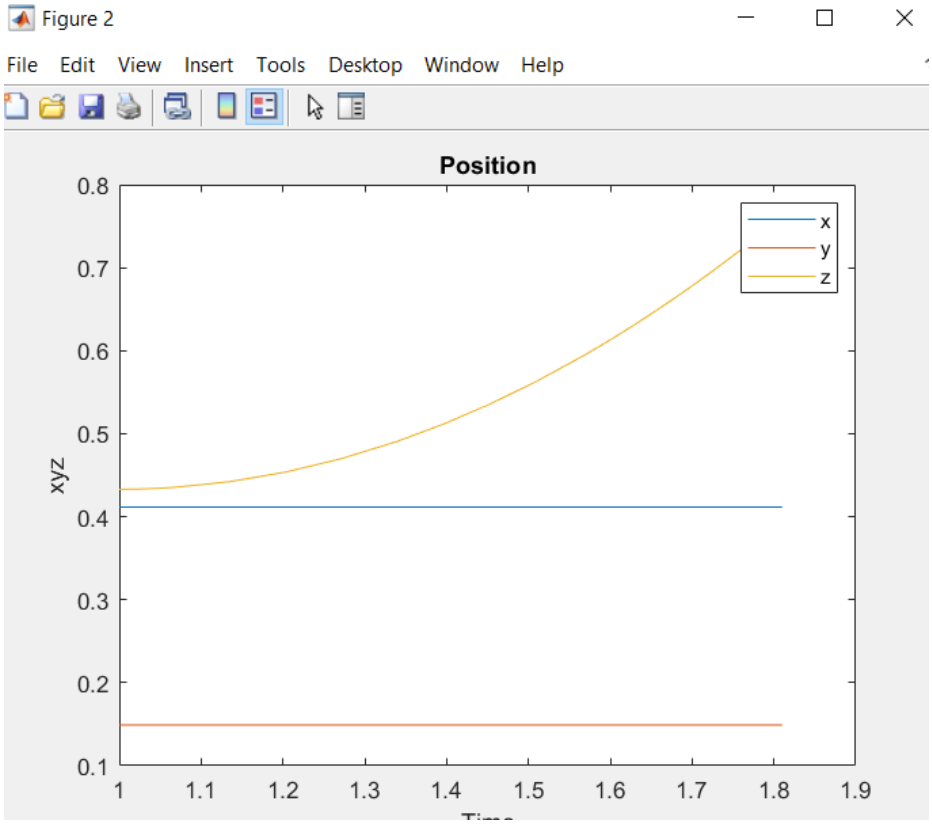


Fig 2.3 u = [0, 0, 1]

Input equations in the already existing dynamical system, we obtain that.

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = D(q)J_h(q)^{-1}(u - J_h\dot{(q)}\dot{q}) + C(q,\dot{q})\dot{q} + g(q)$$

$$J_h(q)\ddot{q} = (u - J_h\dot{(q)}\dot{q})$$

$$\ddot{Y} = u$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = u$$

Therefore, we will get movement only along the axis that does not correspond to zero in the vector u, which we can see in Fig 2.1, 2.2, 2.3

4 Step 3 Apply the linear controller

Formulas from the task develop to the code (Appendix 4.1)

$$Y_d(t) = \begin{bmatrix} -0.866R\cos(\omega t) - 0.56 \\ R\sin(\omega t) \\ 0.5R\cos(\omega t) - 0.08 \end{bmatrix}$$

$$u = \ddot{Y}^d + K_v(\dot{Y}^d - \dot{Y}) + K_p(Y^d - Y)$$

The last part of the model of the robot is the error that is input the same as on the block diagram. Therefore, with infinite system operation, the error will tend to zero. Coefficients can regulate the transient processes of the system, such as the risen time, overshoot, settling time, accuracy, etc. Empirically select coefficients `kd=1.5 kp=1.7`

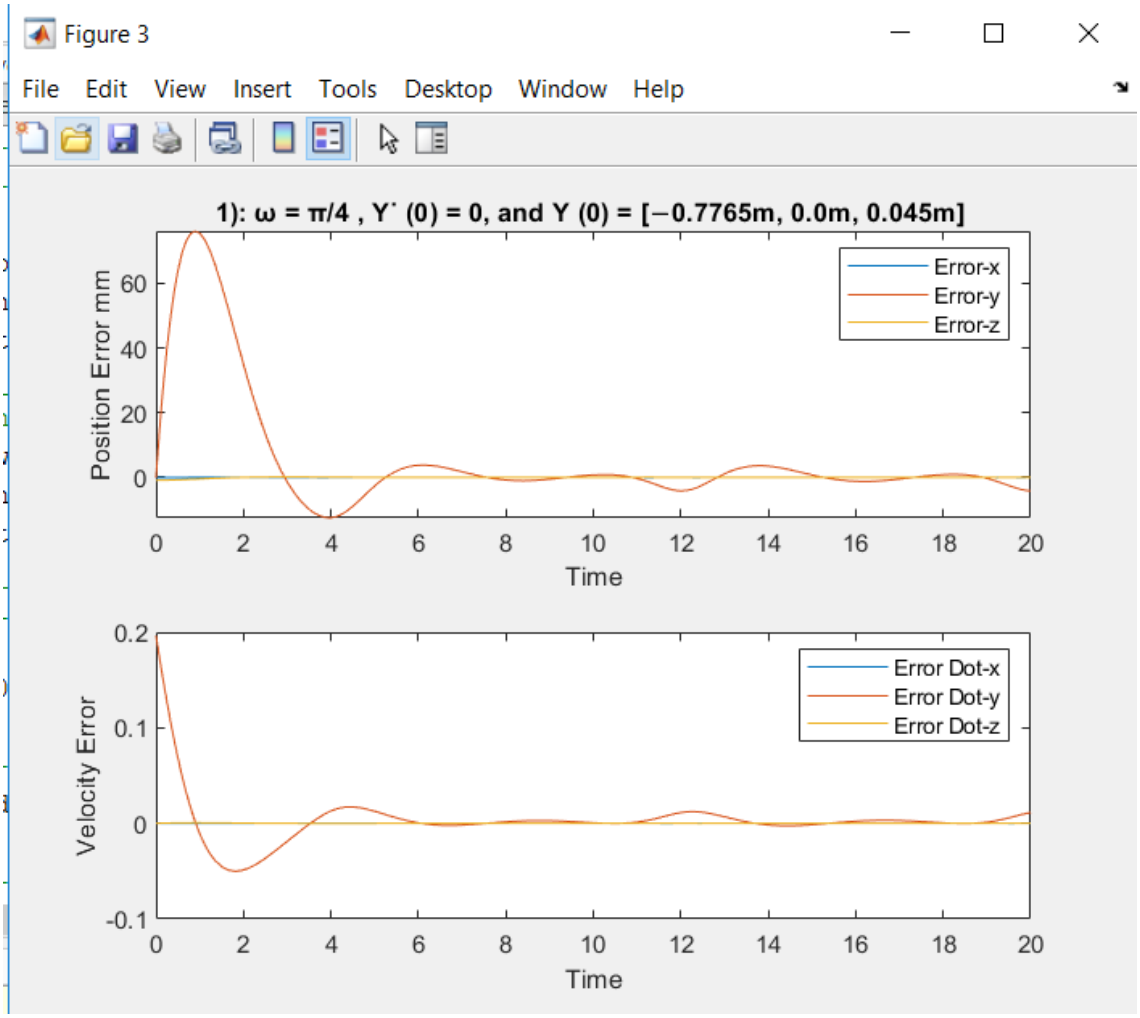


Fig 4. 1 Case 1

First Case. Since the starting point was in the same point as a trajectory is along the X and Z axes (point [-0.7765, 0.25, 0.045]). The error was minimal in them. Therefore, we can see on the graph. Therefore, the robot began to move towards decreasing the error along the Y-axis. Fig(4.1)

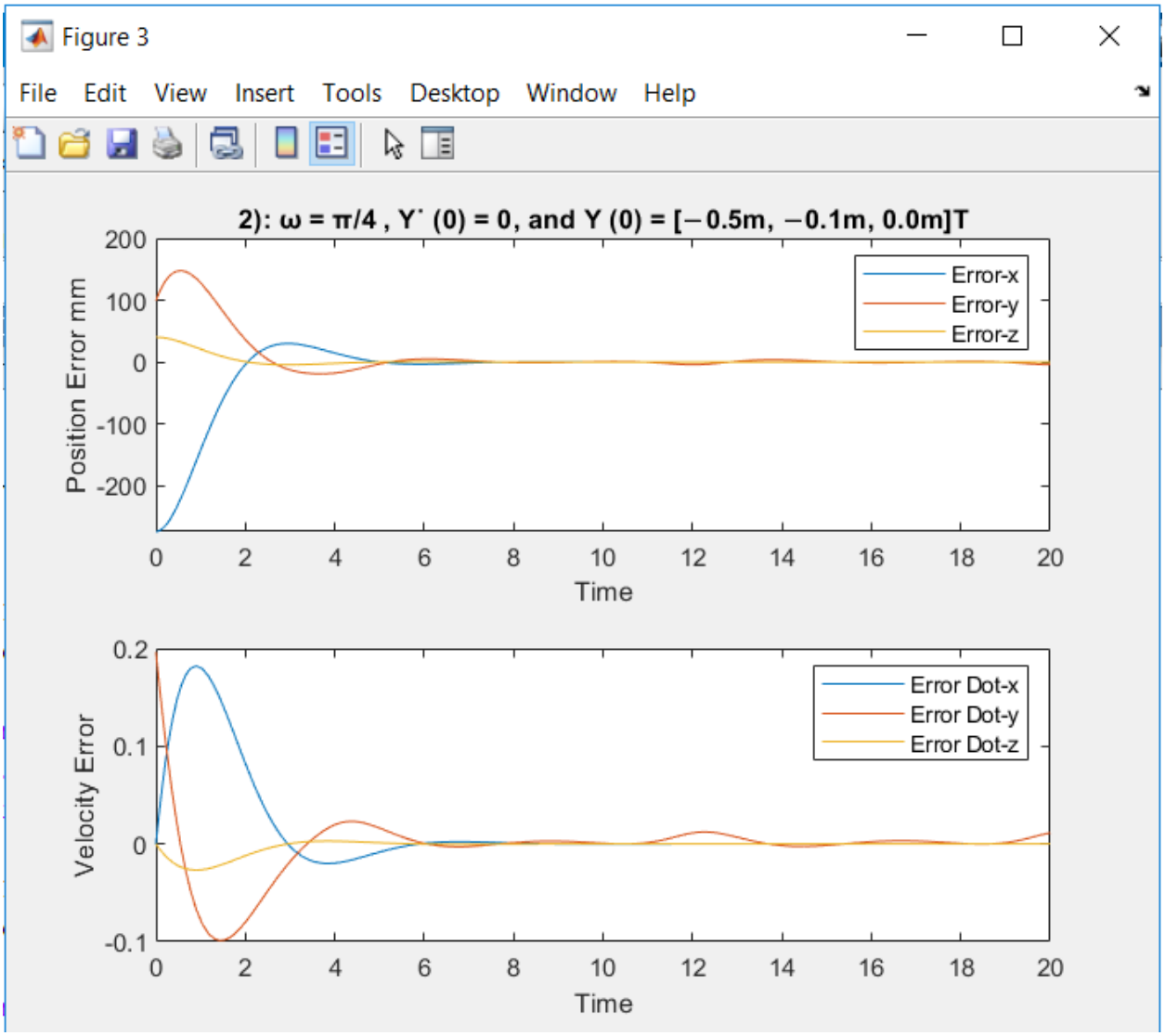


Fig 4.2 Case 2

Second case. The starting point [-0.7765, 0.25, 0.045] is farthest along the axis X. Fig 4.2. Also, if, in the first case, the starting point was catching up with the trajectory, then here point towards it because the velocity error changes sharply.

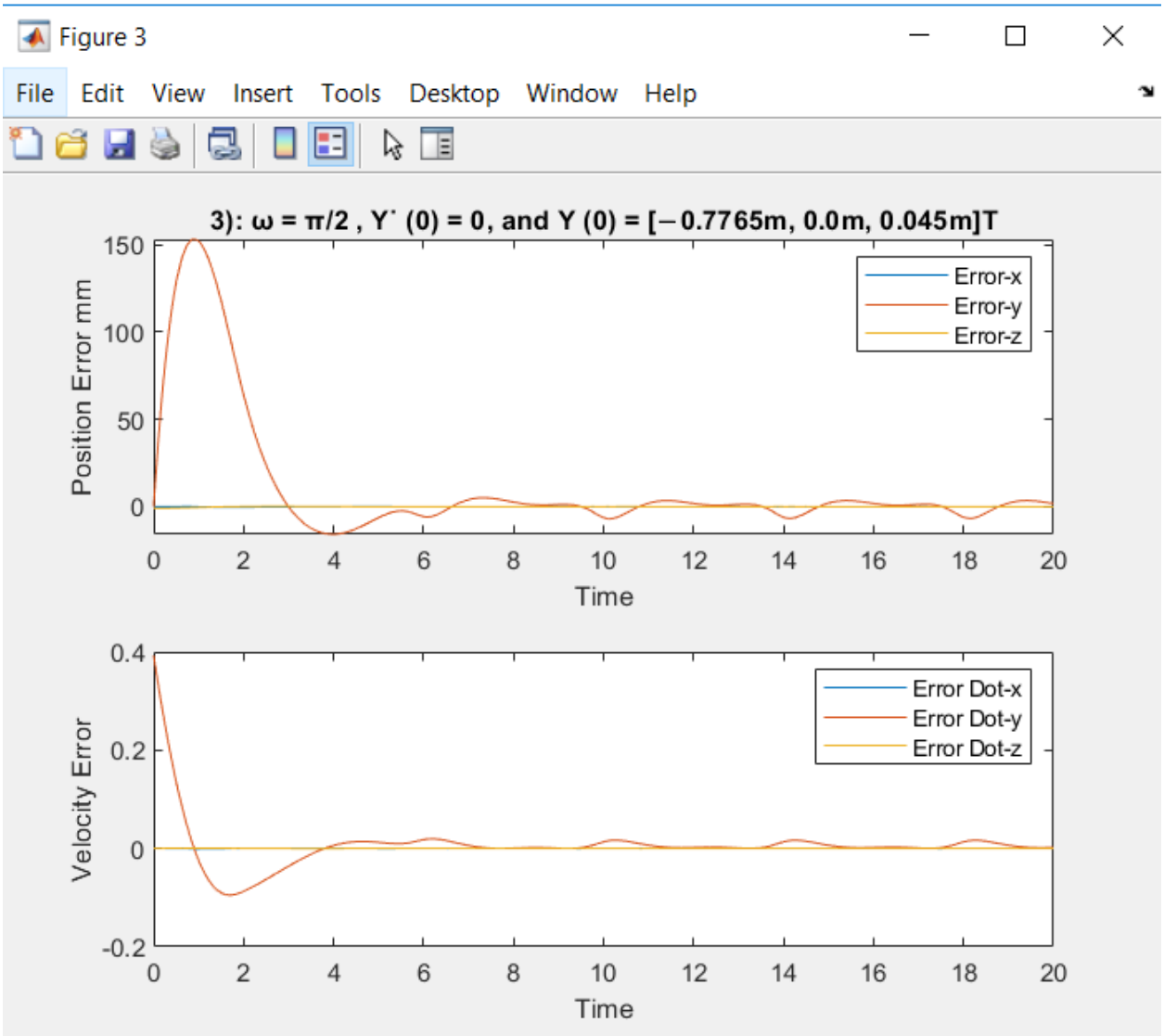


Fig 4.3 Case 3

This case differs from 1 only in frequency, which affects the amplitude on the velocity graph, the frequency is twice as much, and the amplitude of the error velocity graph is twice as large

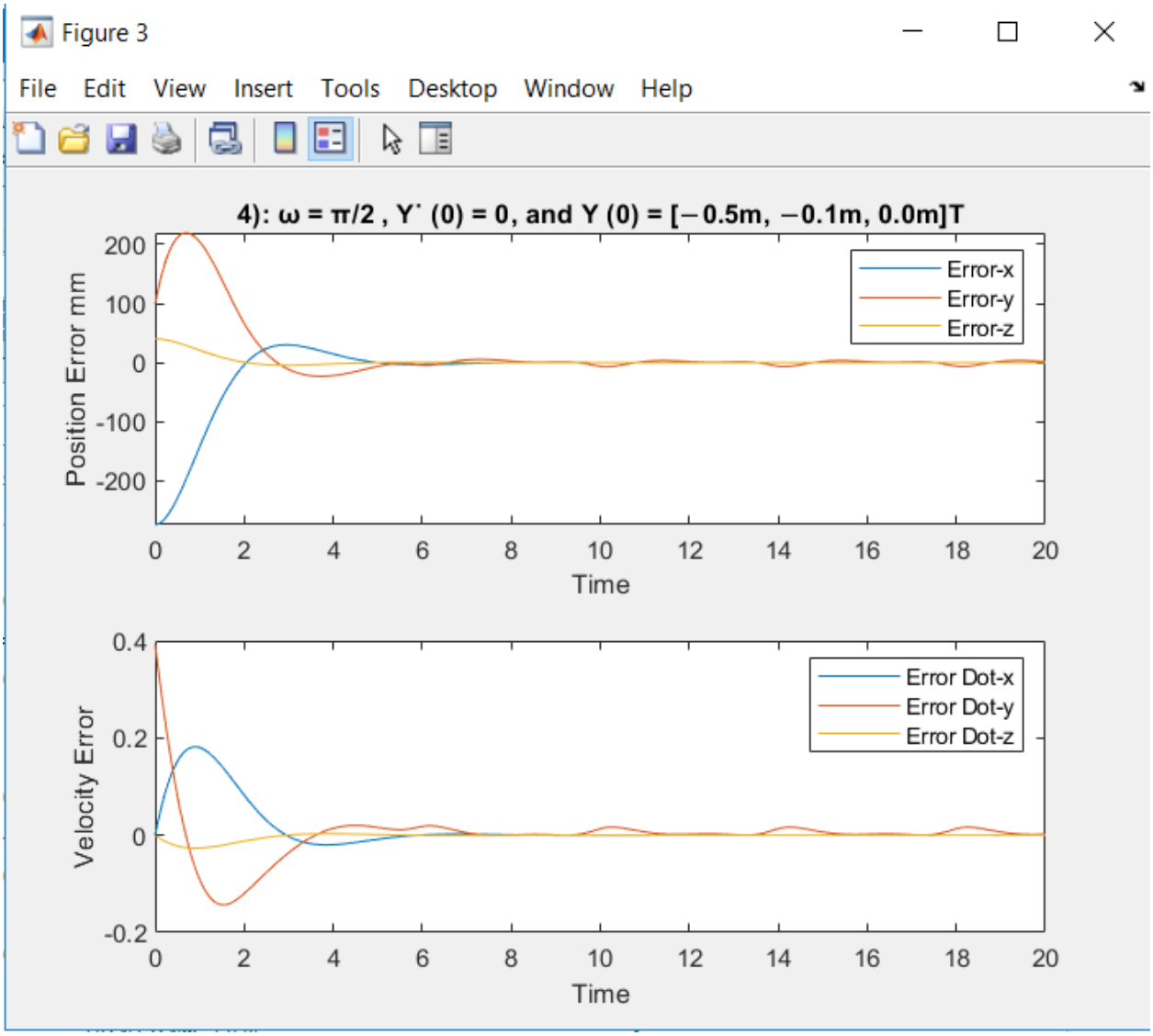


Fig 4.4 Case 4

Similarly, as in the previous case

Appendix

1.1

```
function [D,C,g]=dynamic(y)

y1=y(1);
y2=y(2);
y3=y(3);
y4=y(4);
y5=y(5);
y6=y(6);

%Parameter of D(q)

D(1,1)=2.4574+1.7181*cos(y2)*cos(y2)+0.4430*sin(y2+y3)*sin(
y2+y3)-0.0324*cos(y2)*cos(y2+y3)-
0.0415*cos(y2+y3)*sin(y2+y3)+0.9378*cos(y2)*sin(y2+y3);
D(1,2)=2.2312*sin(y2)+0.0068*sin(y2+y3)-0.1634*cos(y2+y3);
D(1,3)=-0.0068*sin(y2+y3)-0.1634*cos(y2+y3);

D(2,1)=D(1,2);
D(2,2)=5.1285+0.9378*sin(y3)-0.0324*cos(y3);
D(2,3)=0.4424+0.4689*sin(y3)-0.0162*cos(y3);

D(3,1)=D(1,3);
D(3,2)=D(2,3);
D(3,3)=1.0236;

%Parameter of C(q)

c111=0;
c121=0.0207-1.2752*cos(y2)*sin(y2)+0.4429*cos(y3)*sin(y3)-
0.8859*sin(y2)*sin(y3)*sin(y2+y3)+0.0325*cos(y2)*sin(y2+y3)
+ 0.4689*cos(y2)*cos(y2+y3)-0.4689*sin(y2)*sin(y2+y3)-
0.0461*cos(y2+y2)-0.0415*cos(y2+y3)*cos(y2+y3)-
0.0163*sin(y3);
c131=0.0207 + 0.4429*cos(y2)*sin(y2) +
0.4429*cos(y3)*sin(y3)-
0.8859*sin(y2)*sin(y3)*sin(y2+y3)+0.0163*cos(y2)*sin(y2+y3)
+ 0.4689*cos(y2)*cos(y2+y3)-0.0415*cos(y2+y3)*cos(y2+y3);
c211=c121;
c221=1.8181*cos(y2)+0.1634*sin(y2+y3)-0.0068*cos(y2+y3);
c231=0.1634*sin(y2+y3)-0.0068*cos(y2+y3);
c311=c131;
c321=c231;
c331=0.1634*sin(y2+y3)-0.0068*cos(y2+y3);

c112=-c121;
c122=0;
```

```
c132=0;
c212=c122;
c222=0;
c232=0.4689*cos(y3)+0.0162*sin(y3);
c312=0;
c322=c232;
c332=0.4689*cos(y3)+0.0162*sin(y3);

c113=-c131;
c123=-c132;
c133=0;
c213=c123;
c223=-c232;
c233=0;
c313=c133;
c323=c233;
c333=0;

C(1,1)=c111*y4+c211*y5+c311*y6;
C(1,2)=c121*y4+c221*y5+c321*y6;
C(1,3)=c131*y4+c231*y5+c331*y6;
C(2,1)=c112*y4+c212*y5+c312*y6;
C(2,2)=c122*y4+c222*y5+c322*y6;
C(2,3)=c132*y4+c232*y5+c332*y6;
C(3,1)=c113*y4+c213*y5+c313*y6;
C(3,2)=c123*y4+c223*y5+c323*y6;
C(3,3)=c133*y4+c233*y5+c333*y6;

%Parameters of g(q)

g(1,1)=0;
g(2,1)=-48.5564*cos(y2)+1.0462*sin(y2)+0.3683*cos(y2+y3)-
10.6528*sin(y2+y3);
g(3,1)=0.3683*cos(y2+y3)-10.6528*sin(y2+y3);
end
```

1.2

```
function [Yd,Yddot,Yddot_double, Ydot,Y]=kinematic(w,t,q)

    q1=q(1);
    q2=q(2);
    q3=q(3);
    qdot1=q(4)
    qdot2=q(5)
    qdot3=q(6)

Y(1,1)=-0.02032*cos(q1)*cos(q2+q3) +
0.43307*cos(q1)*sin(q2+q3) + 0.4318*cos(q1)*cos(q2) -
0.14909*sin(q1);
```

```
Y(2,1)=-0.02032*sin(q1)*cos(q2+q3) +
0.43307*sin(q1)*sin(q2+q3) + 0.4318*sin(q1)*cos(q2) +
0.14909*cos(q1);
Y(3,1)=0.02032*sin(q2+q3) + 0.43307*cos(q2+q3) -
0.4318*sin(q2);

Ydot(1,1)=0.02032*sin(q1)*cos(q2+q3)*(qdot1)+0.02032*cos(q1
)*sin(q2+q3)*(qdot2+qdot3) -
0.43307*sin(q1)*sin(q2+q3)*qdot1+0.43307*cos(q1)*cos(q2+q3)
*(qdot2+qdot3) - 0.4318*sin(q1)*cos(q2)*qdot1-
0.4318*cos(q1)*sin(q2)*qdot2 -0.14909*cos(q1)*qdot1;
Ydot(2,1)=-
0.02032*cos(q1)*cos(q2+q3)*qdot1+0.02032*sin(q1)*sin(q2+q3)
*(qdot2+qdot3) +
0.43307*cos(q1)*sin(q2+q3)*qdot1+0.43307*sin(q1)*cos(q2+q3)
*(qdot2+qdot3) + 0.4318*cos(q1)*cos(q2)*qdot1 -
0.4318*sin(q1)*sin(q2)*qdot2- 0.14909*sin(q1)*qdot1;
Ydot(3,1)=0.02032*cos(q2+q3)*(qdot2+qdot3) -
0.43307*sin(q2+q3)*(qdot2+qdot3) - 0.4318*cos(q2)*qdot2;

Yd(1,1)=-0.2165*cos(w*t)-0.56;
Yd(2,1)=0.25*sin(w*t);
Yd(3,1)=0.125*cos(w*t)-0.08;

Yddot(1,1)=0.2165*sin(w*t)*w;
Yddot(2,1)=0.25*cos(w*t)*w;
Yddot(3,1)=-0.125*sin(w*t)*w;

Yddot_double(1,1)=0.2165*cos(w*t)*w^2;
Yddot_double(2,1)=-0.25*sin(w*t)*w^2;
Yddot_double(3,1)=-0.125*cos(w*t)*w^2;
end
```

1.3

```
function dydt = odefn(t,y,Y_int,w)


%Compute Dynamics
[D,C,g] = dynamic(y);


% Robot Dynamics
Tou=[1; 0; 0];
qdot_double=inv(D)*(Tou-(C*y(4:6))-g);
dydt(1,1)=y(4);%q1-dot
dydt(2,1)=y(5);%q2-dot
```

```
dydt(3,1)=y(6);%q3-dot

dydt(4,1)=qdot_double(1);
dydt(5,1)=qdot_double(2);
dydt(6,1)=qdot_double(3);

end
```

2.1

```
function dydt = odefn(t,y,Y_int,w)


%Compute Jacobian
[J,Jdot,qdot]=jacobian(y);


%Compute Dynamics
[D,C,g] = dynamic(y);


% Non-linear Feedback
u=[0 0 1]
Tou=D*inv(J)*(u-Jdot*qdot)+(C*qdot)+g;
y(4:6)=qdot.';

% Robot Dynamics
%Tou=[1; 0; 0];
qdot_double=inv(D)*(Tou-(C*y(4:6))-g);
dydt(1,1)=y(4);%q1-dot
dydt(2,1)=y(5);%q2-dot
dydt(3,1)=y(6);%q3-dot

dydt(4,1)=qdot_double(1);
dydt(5,1)=qdot_double(2);
dydt(6,1)=qdot_double(3);

end
```

4.1

```
function dydt = odefn(t,y,Y_int,w)


%Forward Kinematics
```

```
[Yd,Yddot,Yddot_double, Ynew_dot, Ynew]=kinematic(w,t,y);
if t>0
Y_int(1:3)=Ynew';
Y_int(4:6)=Ynew_dot';
end

%Compute Jacobian
[J,Jdot,qdot]=jacobian(y);


%Linear  controller
e=Yd-Y_int(1:3).';
edot=Yddot-Y_int(4:6).';
kd=1.5
kp=1.7
u=Yddot_double+(kd*edot)+(kp*e);


%Compute Dynamics
[D,C,g] = dynamic(y);


% Non-linear Feedback
%u=[0 0 1]
Tou=D*inv(J)*(u-Jdot*qdot)+(C*qdot)+g;
y(4:6)=qdot.';

% Robot Dynamics
%Tou=[1; 0; 0];
qdot_double=inv(D)*(Tou-(C*y(4:6))-g);
dydt(1,1)=y(4);%q1-dot
dydt(2,1)=y(5);%q2-dot
dydt(3,1)=y(6);%q3-dot

dydt(4,1)=qdot_double(1);
dydt(5,1)=qdot_double(2);
dydt(6,1)=qdot_double(3);

end
```

Rest of the code

```
%Main Function
%cases
```

```
% w=(pi/4)
% Y_int=[-0.7765 0 0.045 0 0 0];
%qint=[0.193 -2.59 0.637 0 0 0];

% Y_int=[-0.5 -0.1 0 0 0 0];
%qint=[0.49 -2.16 -0.32 0 0 0];

w=(pi/2)
 %Y_int=[-0.7765 0 0.045 0 0 0];
 %qint=[0.193 -2.59 0.637 0 0 0];

 Y_int=[-0.5 -0.1 0 0 0 0];
 qint=[0.49 -2.16 -0.32 0 0 0];

tspan=[0 20];

%ODE
[t,y]=ode45(@(t,y) odefn(t,y,Y_int,w),tspan,qint);

%Plot Grafics
q=y(:,1:3);
qdot_new=y(:,4:6);
[a,b]=size(q);
Y=zeros(a,b);
Yd=zeros(a,b);
for i=1:a
    q1=q(i,1);
    q2=q(i,2);
    q3=q(i,3);

    Y(i,1)=-0.02032*cos(q1)*cos(q2+q3) +
0.43307*cos(q1)*sin(q2+q3) + 0.4318*cos(q1)*cos(q2) -
0.14909*sin(q1);
    Y(i,2)=-0.02032*sin(q1)*cos(q2+q3) +
0.43307*sin(q1)*sin(q2+q3) + 0.4318*sin(q1)*cos(q2) +
0.14909*cos(q1);
    Y(i,3)=0.02032*sin(q2+q3) + 0.43307*cos(q2+q3) -
0.4318*sin(q2);

    Yd(i,1)=-0.2165*cos(w*t(i))-0.56;
    Yd(i,2)=0.25*sin(w*t(i));
    Yd(i,3)=0.125*cos(w*t(i))-0.08;
end

figure(1)
plot(t,q)
title('q')
xlabel('Time')
ylabel('q')
legend('q1','q2','q3')
```

```
figure(2)
plot(t,Y)
title('Position')
xlabel('Time')
ylabel('xyz')
legend('x','y','z')
error=Yd-Y;

figure(3)
plot(t,error*1000)
title('Position Error ')
xlabel('Time')
ylabel('Error mm')
legend('Error-x','Error-y','Error-z')


for i=1:a
[J]=jacobian_plot(q(i,:));
Ydot(i,:)=(J*qdot_new(i,:).').';


Yddot(i,1)= 0.2165*sin(w*t(i))*w;
Yddot(i,2)= 0.25*cos(w*t(i))*w;
Yddot(i,3)= -0.125*sin(w*t(i))*w;
end

error_dot=Yddot-Ydot;
error=Yd-Y;

figure(3)
subplot(2,1,1);
plot(t,error*1000)
title('4): ? = ?/2 , Y? (0) = 0, and Y (0) = [?0.5m, ?0.1m,
0.0m]T')
xlabel('Time')
ylabel('Position Error mm')
legend('Error-x','Error-y','Error-z')

subplot(2,1,2);
plot(t,error_dot)

xlabel('Time')
ylabel('Velocity Error')
legend('Error Dot-x','Error Dot-y','Error Dot-z')
```

```
function [J,Jdot,qdot]=jacobian(q)


J(1,1)= 0.02032*sin(q(1))*cos(q(2)+q(3)) -
0.43307*sin(q(1))*sin(q(2)+q(3)) -
0.4318*sin(q(1))*cos(q(2)) -0.14909*cos(q(1));
J(1,2)= 0.02032*cos(q(1))*sin(q(2)+q(3)) +
0.43307*cos(q(1))*cos(q(2)+q(3)) -
0.4318*cos(q(1))*sin(q(2));
J(1,3)= 0.02032*cos(q(1))*sin(q(2)+q(3)) +
0.43307*cos(q(1))*cos(q(2)+q(3));

J(2,1)= -0.02032*cos(q(1))*sin(q(2)+q(3)) +
0.43307*cos(q(1))*sin(q(2)+q(3)) +
0.4318*cos(q(1))*cos(q(2)) - 0.14909*sin(q(1));
J(2,2)= 0.02032*sin(q(1))*sin(q(2)+q(3)) +
0.43307*sin(q(1))*cos(q(2)+q(3)) -
0.4318*sin(q(1))*sin(q(2));
J(2,3)= 0.02032*sin(q(1))*sin(q(2)+q(3)) +
0.43307*sin(q(1))*cos(q(2)+q(3));

J(3,1)= 0;
J(3,2)= 0.02032*cos(q(2)+q(3)) - 0.43307*sin(q(2)+q(3)) -
0.4318*cos(q(2));
J(3,3)= 0.02032*cos(q(2)+q(3)) - 0.43307*sin(q(2)+q(3));

qdot= q(4:6);

Jdot(1,1)= 0.02032*cos(q(1))*cos(q(2)+q(3))*qdot(1)-
0.02032*sin(q(1))*sin(q(2)+q(3))*(qdot(2)+qdot(3))-
0.43307*cos(q(1))*sin(q(2)+q(3))*qdot(1)-
0.43307*sin(q(1))*cos(q(2)+q(3))*(qdot(2)+qdot(3)) -
0.4318*cos(q(1))*cos(q(2))*qdot(1)
+0.4318*sin(q(1))*sin(q(2))*qdot(2)+0.14909*sin(q(1))*qdot(
1);
Jdot(1,2)= -
0.02032*sin(q(1))*sin(q(2)+q(3))*qdot(1)+0.02032*cos(q(1))*
cos(q(2)+q(3))*(qdot(2)+qdot(3)) -
0.43307*sin(q(1))*cos(q(2)+q(3))*qdot(1)-
0.43307*cos(q(1))*sin(q(2)+q(3))*(qdot(2)+qdot(3))
+0.4318*sin(q(1))*sin(q(2))*qdot(1)-
0.4318*cos(q(1))*cos(q(2))*qdot(2);
Jdot(1,3)= -
0.02032*sin(q(1))*sin(q(2)+q(3))*qdot(1)+0.02032*cos(q(1))*
cos(q(2)+q(3))*(qdot(2)+qdot(3)) -
0.43307*sin(q(1))*cos(q(2)+q(3))*qdot(1)-
0.43307*cos(q(1))*sin(q(2)+q(3))*(qdot(2)+qdot(3));

Jdot(2,1)= 0.02032*sin(q(1))*sin(q(2)+q(3))*qdot(1) -
0.02032*cos(q(1))*cos(q(2)+q(3))*(qdot(2)+qdot(3)) -
```

```
0.43307*sin(q(1))*sin(q(2)+q(3))*qdot(1)+0.43307*cos(q(1))*
cos(q(2)+q(3))*(qdot(2)+qdot(3)) -
0.4318*sin(q(1))*cos(q(2))*qdot(1)-
0.4318*cos(q(1))*sin(q(2))*qdot(2) -
0.14909*cos(q(1))*qdot(1);
Jdot(2,2)= 0.02032*cos(q(1))*sin(q(2)+q(3))*qdot(1)+
0.02032*sin(q(1))*cos(q(2)+q(3))*(qdot(2)+qdot(3))
+0.43307*cos(q(1))*cos(q(2)+q(3))*qdot(1) -
0.43307*sin(q(1))*sin(q(2)+q(3))*(qdot(2)+qdot(3)) -
0.4318*cos(q(1))*sin(q(2))*qdot(1)-
0.4318*sin(q(1))*cos(q(2))*qdot(2);
Jdot(2,3)=0.02032*cos(q(1))*sin(q(2)+q(3))*qdot(1)+
0.02032*sin(q(1))*cos(q(2)+q(3))*(qdot(2)+qdot(3))+
0.43307*cos(q(1))*cos(q(2)+q(3))*qdot(1) -
0.43307*sin(q(1))*sin(q(2)+q(3))*(qdot(2)+qdot(3));

Jdot(3,1)= 0;
Jdot(3,2)= -0.02032*sin(q(2)+q(3))*(qdot(2)+qdot(3)) -
0.43307*cos(q(2)+q(3))*(qdot(2)+qdot(3)) +
0.4318*sin(q(2))*qdot(2);
Jdot(3,3)= -0.02032*sin(q(2)+q(3))*(qdot(2)+qdot(3)) -
0.43307*cos(q(2)+q(3))*(qdot(2)+qdot(3));


end



function [J]=jacobian_plot(q)
J(1,1)= 0.02032*sin(q(1))*cos(q(2)+q(3)) -
0.43307*sin(q(1))*sin(q(2)+q(3)) -
0.4318*sin(q(1))*cos(q(2)) -0.14909*cos(q(1));
J(1,2)= 0.02032*cos(q(1))*sin(q(2)+q(3)) +
0.43307*cos(q(1))*cos(q(2)+q(3)) -
0.4318*cos(q(1))*sin(q(2));
J(1,3)= 0.02032*cos(q(1))*sin(q(2)+q(3)) +
0.43307*cos(q(1))*cos(q(2)+q(3));

J(2,1)= -0.02032*cos(q(1))*sin(q(2)+q(3)) +
0.43307*cos(q(1))*sin(q(2)+q(3)) +
0.4318*cos(q(1))*cos(q(2)) - 0.14909*sin(q(1));
J(2,2)= 0.02032*sin(q(1))*sin(q(2)+q(3)) +
0.43307*sin(q(1))*cos(q(2)+q(3)) -
0.4318*sin(q(1))*sin(q(2));
J(2,3)= 0.02032*sin(q(1))*sin(q(2)+q(3)) +
0.43307*sin(q(1))*cos(q(2)+q(3));

J(3,1)= 0;
```

```
J(3,2)= 0.02032*cos(q(2)+q(3)) - 0.43307*sin(q(2)+q(3)) -
0.4318*cos(q(2));
J(3,3)= 0.02032*cos(q(2)+q(3)) - 0.43307*sin(q(2)+q(3));
end
```