Mech 564 Final Project Report

Connor Worrell

5/3/2021

Introduciton

For this project I started by inputting the dynamics model for the robot that was given. I then added a function to convert the robot position to the position of the end effector in Cartesian coordinates. Using these two I attempted to visually verify that the dynamics model was correct. Then I implemented a task level controller, and verified that it was working correctly visually. Then I implemented the given controller model, and attempted to tune it to track the circular function.

Dynamics Model of the Robot

I modeled the dynamics of this robot using the given dynamics equations. I used Matlab's symbolic variables to symbolically solve for D, c, g, and Y. I then used the Matlab built in odeToVectorField and matlabFunction to prepare the symbolic functions to be input into the ODE solver. Initially I used ODE45, but switched to ODE23 because it had faster solve times with minimal performance degradation. I then recorded the output of the ODE solver, and used the h function to convert them to Cartesian. I tested this model using $\tau = [1, 0, 0]^T$. The output plots for this model calculated over 10 seconds are shown in figure 1 - 3.

I think that figure 3 shows the best visualization of the end effector. In this figure the line represents the location of the end effector, and the color represents the time between 0

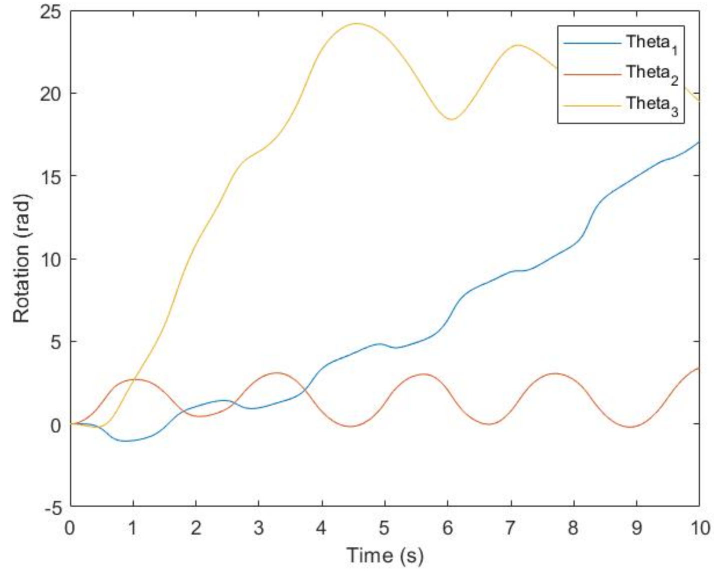

Figure 1: No Control - Joint Angles

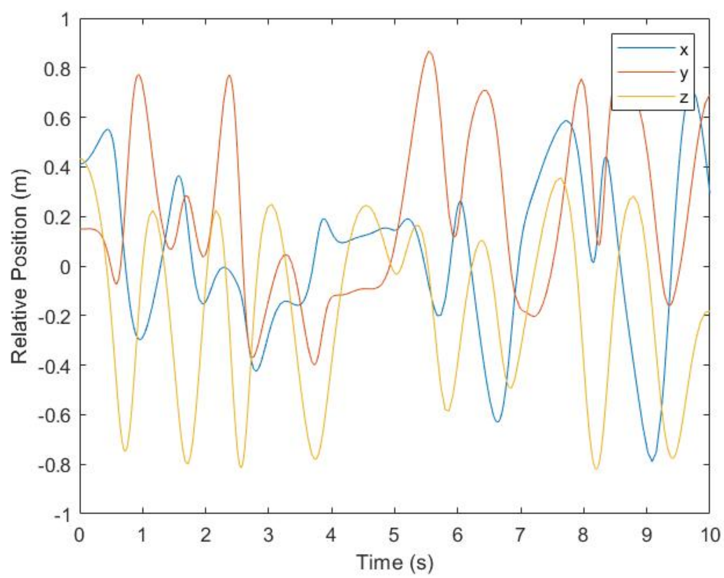

Figure 2: No Control - Cartesian Position of End Effector

seconds (Blue) and 10 seconds (Light Yellow). The end effector appears to drop down, and go through a motion that is very chaotic. I believe that this motion is the correct motion, because the end effector would be expected to crumple under its own weight without any motor power keeping it up, and links 2 and 3 of the robot arm resemble a double pendulum, which is know for its random and chaotic motion. On top of this the motion of joint 1 can be explained by the torque applied by the momentum of links 2 and 3, as well as the applied torque by the input τ. These two effects added together give a sinusoidal-ish wave (momentum) that rises exponentially (τ)

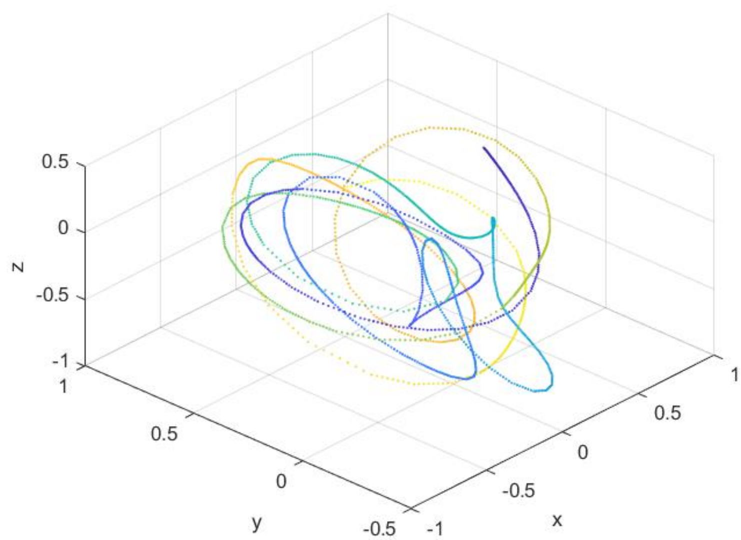

Figure 3: No Control - Cartesian Position of End Effector 3D

Controller Simulation

To simulate the controller, I modeled the nonlinear controller equation using symbolic functions. Initially I attempted to build the controller and robot dynamics simulation into the same ODE, and while Matlab's ODE solver was able to solve it, it took a long time. Because of this I decided to descritize the nonlinear controller, and use the ODE solver to solve many small steps where I numerically recalculate the nonlinear controller equation between each of these. I then tested the non-linear controller by using $u = [1, 0, 0]^T$, $u = [0, 1, 0]^T$,$u = [0, 0, 1]^T$. Teh results for this are shown in figures 4 - 6

In these results I would expect the graph to show the driven axis increasing, and the two other axes remaining unchanged. This is true for the tests with the exception of the very end of the X and Z tests, this is because the robot hit the edge of its work area and the controller became unstable. This section can be ignored.

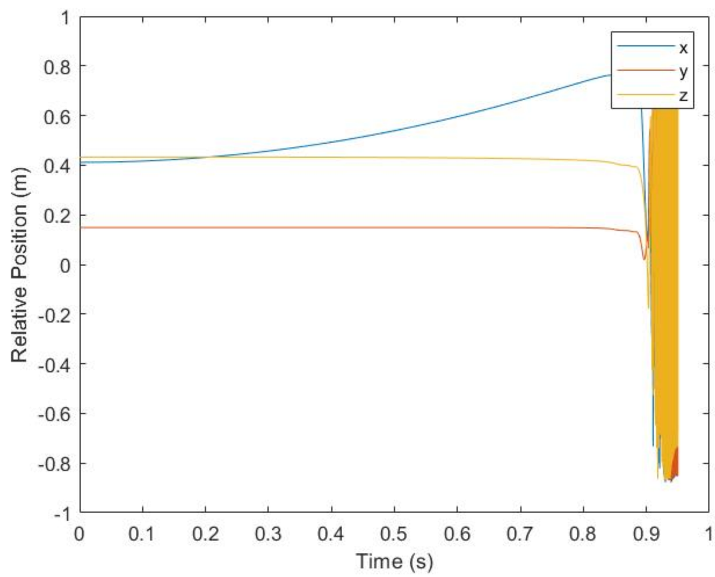

Figure 4: Controller X Position Test

Tracking a Trajectory

In order to track the tilted circle trajectory that was given, I implemented the linear controller from class and tuned the response to be quick. The tuned response parameters are listed in table 1.

For the four test cases I calculated the θ_1-θ_3 required to satisfy the initial conditions given in 4.2.3. I used EES to solve for the initial conditions, and they are listed in table 2.

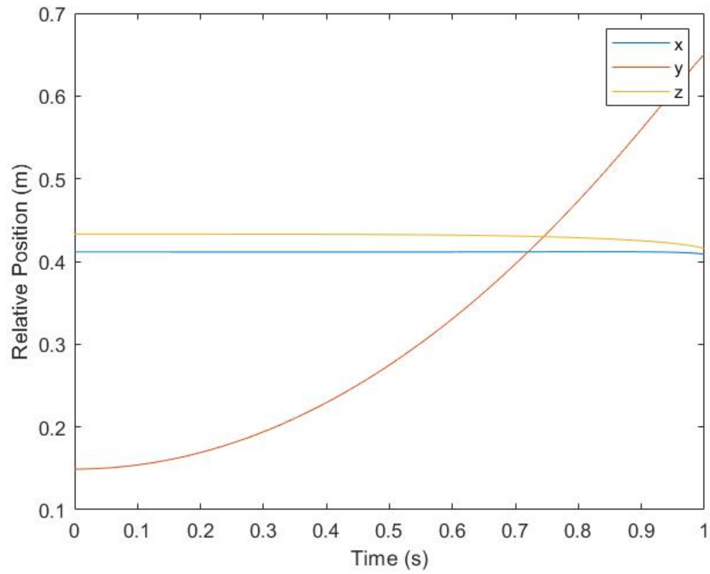

Figure 5: Controller Y Position Test

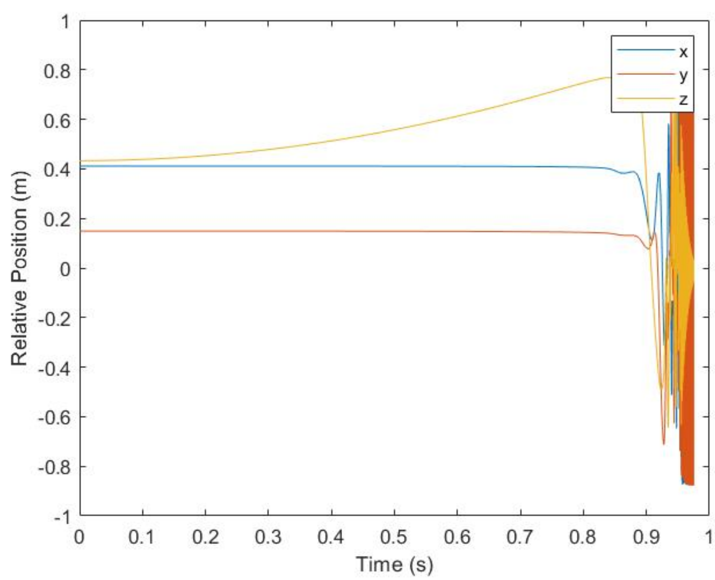

Figure 6: Controller Z Position Test

Table 1: Tuning Parameters

Parameter	Value
K_v	200
K_p	25000

Table 2: Initial Positon

Case	$\theta_1(0)$	$\theta_2(0)$	$\theta_3(0)$	ω
1	2.948	5.733	2.598	$\pi/2$
2	3.041	5.308	3.562	$\pi/2$
3	2.948	5.733	2.598	$\pi/4$
4	3.041	5.308	3.562	$\pi/4$

For each of the cases listed in table 1, I simulated from 0-.2 seconds using a .001 time-step. The graphs for each case can be found in figures 7-10. In these figure blue represents 0 seconds and yellow represents .2 seconds

Discussion

The decoupling was achieved by using the Non-Linear Feedback equation form class. This was verified to be working when testing various inputs for “u” in figures 4 - 5. In these figures it is seen that position in the stationary dimensions was unaffected by the movement of the moving dimension. The linearization of the error was achieved by using the Linear Controller discussed in class. This controller inputs error, and outputs the direction of motion for the end-effector. This was verified to be working by using Cases 1-4. The controller was able to calculate the proper direction to move using the error. The robot’s initial position and velocity have no long term effect on the errors. The starting position has an effect on the velocity error, both Case 1 and 3 have much smaller velocity errors than Cases 2 and 4. A starting position further away from he target has a larger initial error. When the target was spinning faster, the errors in position were larger even after “steady

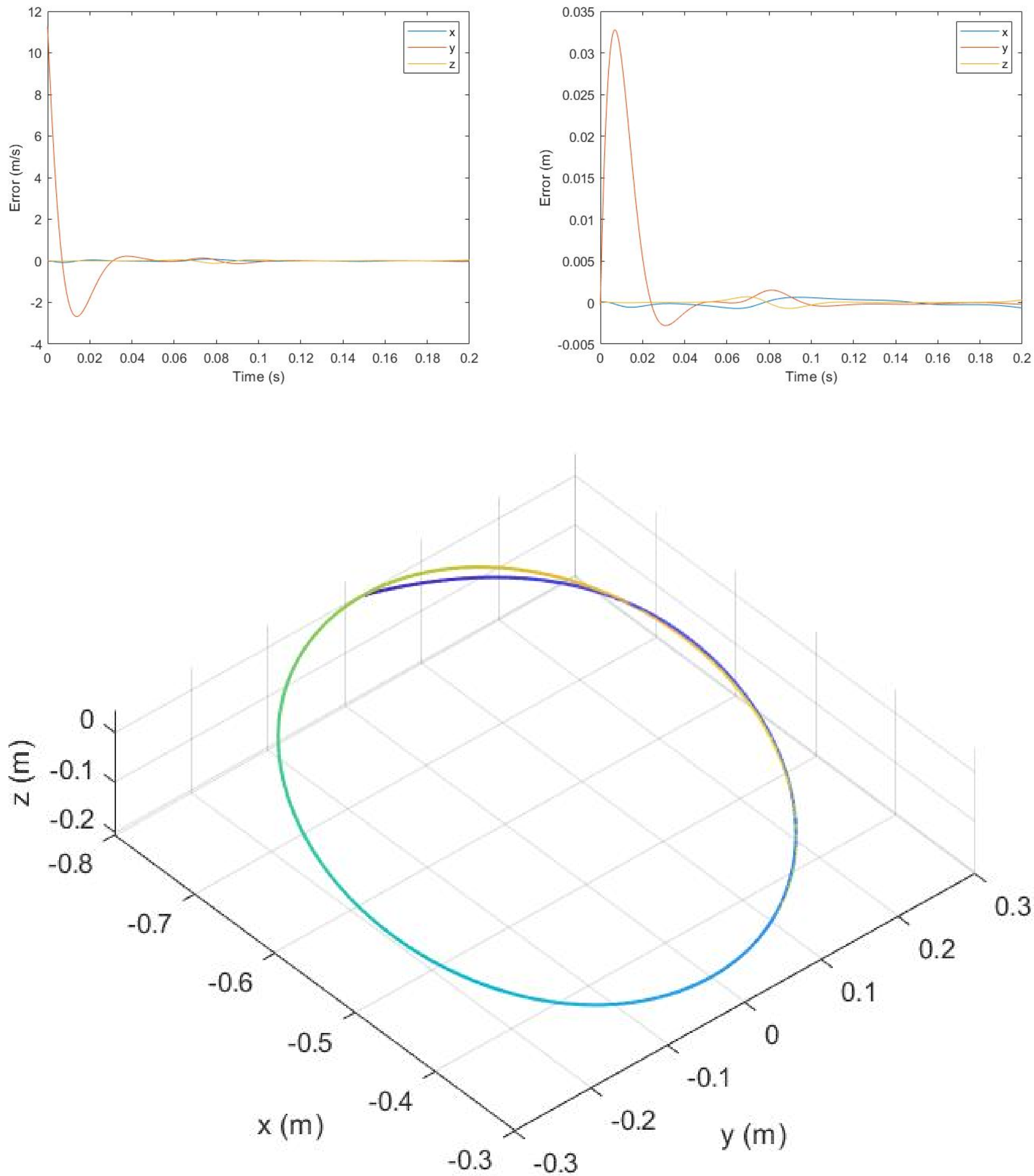

Figure 7: Case 1

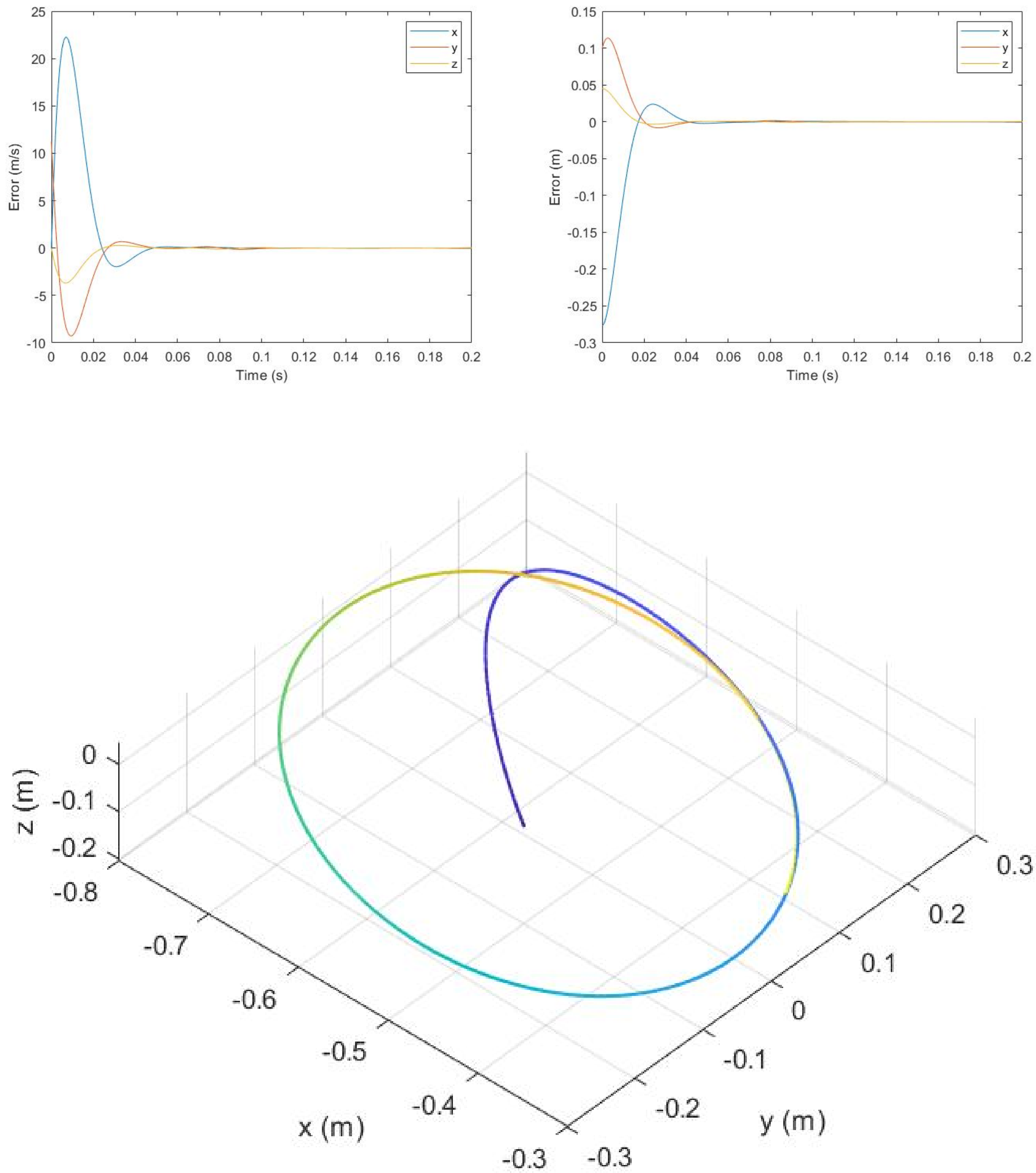

Figure 8: Case 2

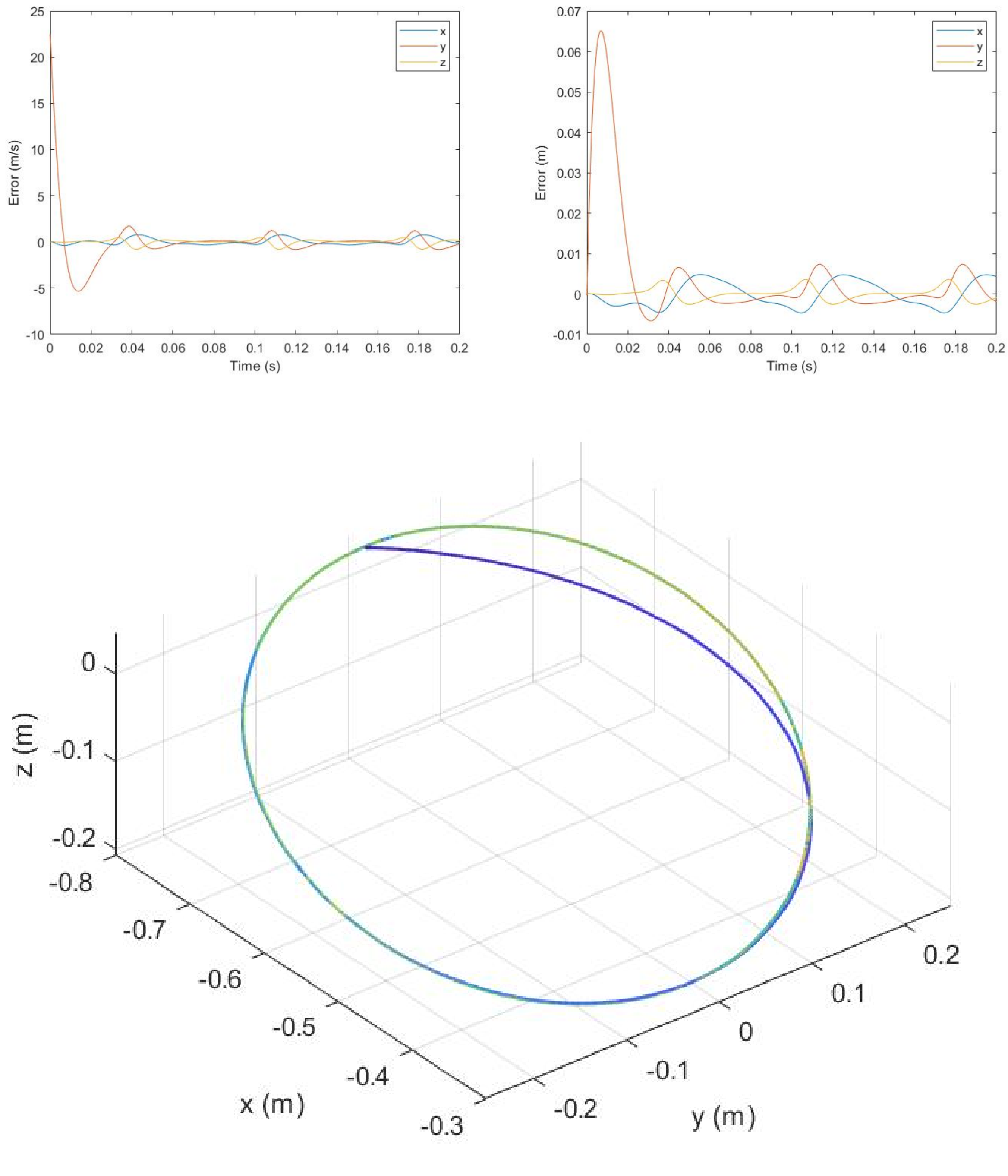

Figure 9: Case 3

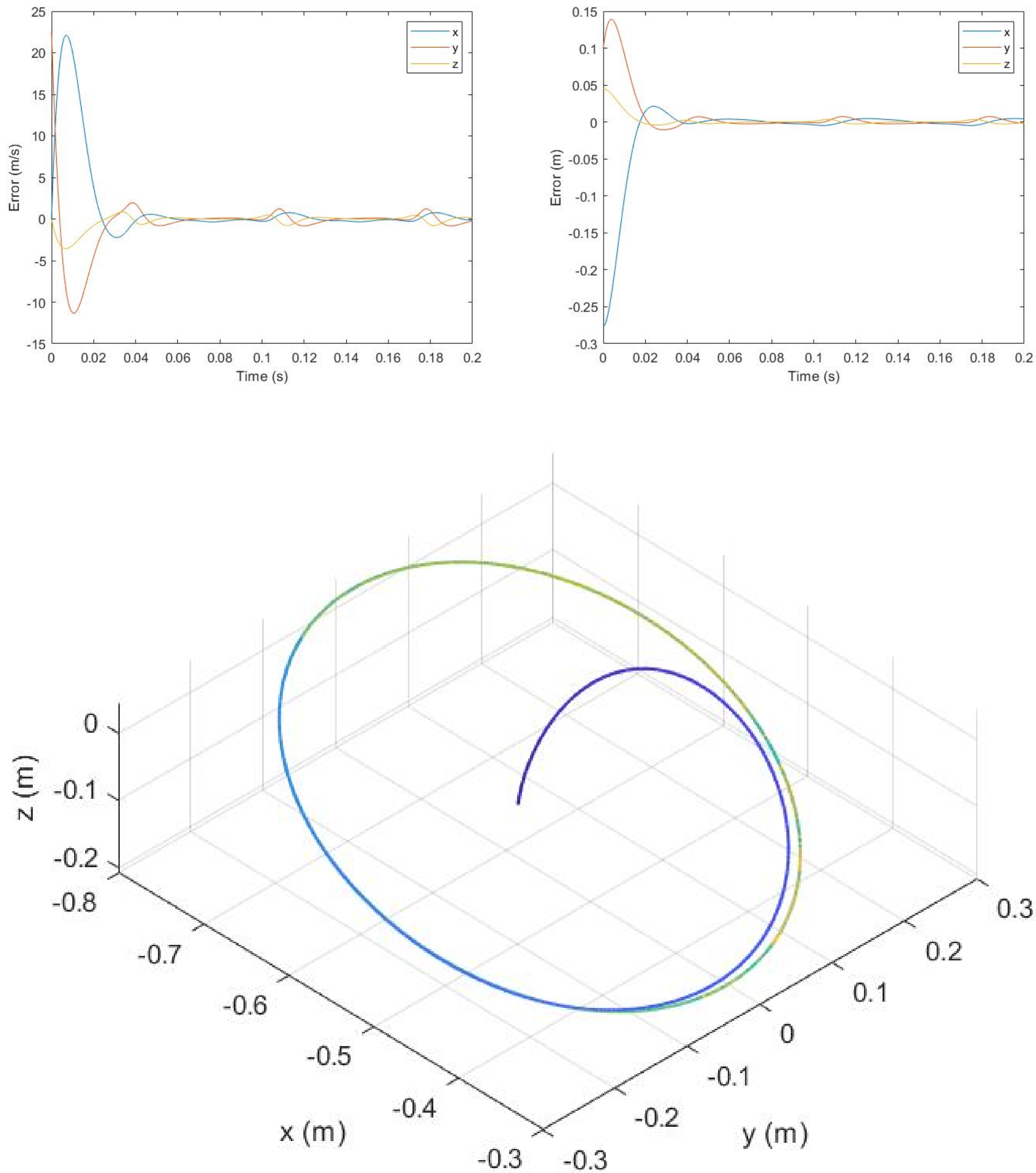

Figure 10: Case 4

state" was reached, this can be seen when comparing the position error of 9 and 7. This simulation is not perfect, and there are a few modeling errors that would cause deviations form real world results. These deviations would be cause by the lack of joint friction in this model, the descretization of the linear controller and non-linear feedback, and errors in the numeric approximation of the ODE solver.

Appendix

```
% ---- Mech564FinalProject ----
close all

syms theta_1(t) theta_2(t) theta_3(t) %real
syms d4 d2 a2 a3 %real
syms t

fig = uifigure;
loadingbar = uiprogressdlg(fig, 'Title', 'Progress', 'Message','1');
drawnow

loadingbar.Message = 'Initilizing Robot';
q = [theta_1(t),theta_2(t),theta_3(t)];
q_2dot = [diff(theta_1(t),2);diff(theta_2(t),2);diff(theta_3(t),2)];
q_dot  = [diff(theta_1(t),1);diff(theta_2(t),1);diff(theta_3(t),1)];

d2 = 149.09/1000; %m
d4 = 433.07/1000; %m
a2 = 431.8/1000; %m
a3 = -20.32/1000; %m

d(1,1) = 2.4574 + 1.7181*cos(theta_2(t))*cos(theta_2(t))+.443*sin(theta_2(t)+theta_3(t))
d(1,2) = 2.2312*sin(theta_2(t)) - .0068*sin(theta_2(t)+theta_3(t))-.1634*cos(theta_2(t)+
d(1,3) = -.0068*sin(theta_2(t)+theta_3(t))-.1634*cos(theta_2(t)+theta_3(t));
d(2,1) = d(1,2);
d(2,2) = 5.1285+.9378*sin(theta_3(t))-.0324*cos(theta_3(t));
d(2,3) = .4424+.4689*sin(theta_3(t))-.0162*cos(theta_3(t));
d(3,1) = d(1,3);
d(3,2) = d(2,3);
d(3,3) = 1.0236;
```

```
c111 = 0;
c121 = 0.0207 - 1.2752*cos(theta_2(t))*sin(theta_2(t)) + 0.4429*cos(theta_3(t))*sin(thet
c131 = 0.0207 + 0.4429*cos(theta_2(t))*sin(theta_2(t)) + 0.4429*cos(theta_3(t))*sin(thet
c211 = c121;
c221 = 1.8181*cos(theta_2(t)) + 0.1634*sin(theta_2(t)+theta_3(t)) - 0.0068*cos(theta_2(t
c231 = 0.1634*sin(theta_2(t)+theta_3(t)) - 0.0068*cos(theta_2(t)+theta_3(t));
c311 = c131;
c321 = c231;
c331 = 0.1634*sin(theta_2(t)+theta_3(t)) - 0.0068*cos(theta_2(t)+theta_3(t));
c112 = -c121;
c122 = 0;
c132 = 0;
c212 = c122;
c222 = 0;
c232 = 0.4689*cos(theta_3(t)) + 0.0162*sin(theta_3(t));
c312 = 0;
c322 = c232;
c332 = 0.4689*cos(theta_3(t)) + 0.0162*sin(theta_3(t));
c113 = -c131;
c123 = -c132;
c133 = 0;
c213 = c123;
c223 = -c232;
c233 = 0;
c313 = c133;
c323 = c233;
c333 = 0;

clearvars c

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

syms g h real

g(1) = 0;
g(2) = -48.5564*cos(theta_2(t)) + 1.0462*sin(theta_2(t))+.3683*cos(theta_2(t)+theta_3(t)
```

```
g(3) = .3683*cos(theta_2(t)+theta_3(t)) - 10.6528*sin(theta_2(t)+theta_3(t));
g = g';

h(1) = a3*cos(theta_1(t))*cos(theta_2(t)+theta_3(t)) + d4*cos(theta_1(t))*sin(theta_2(t)
h(2) = a3*sin(theta_1(t))*cos(theta_2(t)+theta_3(t)) + d4*sin(theta_1(t))*sin(theta_2(t)
h(3) = -a3*sin(theta_2(t)+theta_3(t))+d4*cos(theta_2(t)+theta_3(t))-a2*sin(theta_2(t));
h = h'

tauRight = d*q_2dot+c*q_dot+g;

J = [diff(h(1),q(1)),diff(h(1),q(2)),diff(h(1),q(3))
    diff(h(2),q(1)),diff(h(2),q(2)),diff(h(2),q(3))
    diff(h(3),q(1)),diff(h(3),q(2)),diff(h(3),q(3))
    ]

R = .25 %m

w = pi/4
%w = pi/2

Y_d = [-.866*R*cos(rad2deg(w*t))-0.56
       R*sin(rad2deg(w*t))
       0.5*R*cos(rad2deg(w*t))-.08]

%Y_d = sym([-0.56; 0; -.08])


loadingbar.Message = 'Building u';
%u = diff(Y_d,2)+K_v*(diff(Y_d)-diff(h'))+K_p*(Y_d-h')
%u = [0 1 0]'
loadingbar.Message = 'Building tauLeft';
%tauLeft = (J/d)*(u-diff(J)*q_dot)+c*q_dot+g
%tauLeft = [0,0,0]
syms tauLeft_1 tauLeft_2 tauLeft_3
tauLeft = [tauLeft_1 tauLeft_2 tauLeft_3]'

%tauTest = [0,40,0];
Theta = [0 0 0 0 0 0] % Initial Conditions T1, DT1, T2, DT2, T3, DT3
%Theta = [0,0,1.64247225087442,0,1.46621743544426,0]; %Basically no
% movement, arm fully down

clearvars VelData
%TotalSteps = 1;
%LinerazationTimeStepSize = 1;
```

```
Position = [];

%for i = 1:TotalSteps
% loadingbar.Message = sprintf('Calculating Step %d of %d',i,TotalSteps);
% loadingbar.Value = i/TotalSteps;

%LinearizeSpot = [theta_1(t) == Theta(1); theta_2(t) == Theta(3) ;theta_3(t) == Theta(
tauEqns = [tauLeft(1) == tauRight(1);tauLeft(2) == tauRight(2);tauLeft(3) == tauRight(3)
%[t2add,Theta,Meaning] = plotODESolve(tauTest,tau,LinearizeSpot,Theta(length(Theta(:,1
loadingbar.Message = 'Linearizing';
[SymSys,S] = odeToVectorField(tauEqns)%,LinearizeSpot);
loadingbar.Message = 'Converting Linearization to Function';
Sys = matlabFunction(SymSys,'vars',{'t','Y','tauLeft_1','tauLeft_2','tauLeft_3'});
%tspan = [0 LinerazationTimeStepSize];
%Y0 = Conditions; %[0 0 0 0 0 0]
%Sys1 = @(t,Y) Sys(Y)
%%
loadingbar.Message = 'Solving ODE';

SimulationLength = 0.2;
SimulationStepSize = .001;

% Starting Position T1 DT1 T2 DT2 T3 DT3
%VelData = [deg2rad(180),0,-deg2rad(45),0,deg2rad(90+45),0];

VelData = [2.948,0,5.733,0,2.598,0]; %Case 1,3
%VelData = [3.041,0,5.308,0,3.562,0]; %Case 2,4

TimeData = [0];
Error = [];
Error_Dot = [];

% Remake J,d,c,g,h,J_dot for easy substitution
syms theta_1 theta_1_dot theta_2 theta_2_dot theta_3 theta_3_dot

diffJ = diff(J)
diffY_d = diff(Y_d)
diff2Y_d = diff(Y_d,2)
diffh = diff(h)

% Tuning Parameters
K_v = 200
K_p = 25000
```

```
for i = 0:SimulationStepSize:SimulationLength-SimulationStepSize
    tic
    loadingbar.Value = i/SimulationLength;
    CurrentVelData = VelData(length(VelData(:,1)),:);
    theta_1 = CurrentVelData(1);
    theta_1_dot = CurrentVelData(2);
    theta_2 = CurrentVelData(3);
    theta_2_dot = CurrentVelData(4);
    theta_3 = CurrentVelData(5);
    theta_3_dot = CurrentVelData(6);

    J_ForCalc = double(subs(subs(subs(subs(subs(subs(J,str2sym('diff(theta_1(t), t)'),th
    d_ForCalc = double(subs(subs(subs(subs(subs(subs(d,str2sym('diff(theta_1(t), t)'),th
    diffJ_ForCalc = double(subs(subs(subs(subs(subs(subs(diffJ,str2sym('diff(theta_1(t),
    q_dot_ForCalc = double([theta_1_dot;theta_2_dot;theta_3_dot]);
    c_ForCalc = double(subs(subs(subs(subs(subs(subs(c,str2sym('diff(theta_1(t), t)'),th
    g_ForCalc = double(subs(subs(subs(subs(subs(subs(g,str2sym('diff(theta_1(t), t)'),th
    h_ForCalc = double(subs(subs(subs(subs(subs(subs(h,str2sym('diff(theta_1(t), t)'),th
    Y_d_ForCalc = double(subs(Y_d,str2sym('t'),i));
    diffY_d_ForCalc = double(subs(diffY_d,str2sym('t'),i));
    diff2Y_d_ForCalc = double(subs(diff2Y_d,str2sym('t'),i));
    diffh_ForCalc = double(subs(subs(subs(subs(subs(subs(diffh,str2sym('diff(theta_1(t),

    u = diff2Y_d_ForCalc+K_v*(diffY_d_ForCalc-diffh_ForCalc)+K_p*(Y_d_ForCalc-h_ForCalc)
    [diff2Y_d_ForCalc,K_v*(diffY_d_ForCalc-diffh_ForCalc),K_p*(Y_d_ForCalc-h_ForCalc)]
    %u = normalize(u)
    %u = [0 0 1]';
    tauLeft = (d_ForCalc/J_ForCalc)*(u-diffJ_ForCalc*q_dot_ForCalc)+c_ForCalc*q_dot_ForC
    tauLeft_1 = double(tauLeft(1));
    tauLeft_2 = double(tauLeft(2));
    tauLeft_3 = double(tauLeft(3));

    %options = odeset('RelTol',1e-5,'Stats','on','OutputFcn',@odeplot);
    tspan = [i i+SimulationStepSize];
    [t,y] = ode23(@(t,Y) Sys(t,Y,tauLeft_1,tauLeft_2,tauLeft_3), tspan, CurrentVelData);

    VelData = [VelData;y];%y(length(y(:,1)),[1,3,5])
    TimeData = [TimeData;t];
    toc
end

%Duplicate the first number to be the same size as t

loadingbar.Message = 'Displaying';
```

```
%Data(length(Data(:,1))+1:length(Data(:,1))+length(Theta(:,1)),1:6) = Theta;
%t = [0,t2add()'];
fprintf("T: %d D: %d",size(t,2),size(VelData,1))
%end

close(loadingbar)
close(fig)
%%


figure()
plot(TimeData,VelData(:,[1,3,5]))
legend('Theta_1','Theta_2','Theta_3')
xlabel("Time (s)")
ylabel("Rotation (rad)")




% Proof that forward kinematics function works properly
%Data = [zeros(63,1),zeros(63,1),zeros(63,1),zeros(63,1),(0:.1:2*pi)']

Position = zeros(length(VelData(:,1)),3);
for i = 1:length(VelData(:,1))
    Pos = forwardKinematics(VelData(i,1),VelData(i,3),VelData(i,5),d2,d4,a2,a3);
    Position(i,:) = Pos';
end

figure()
%plot3(Position(:,1),Position(:,2),Position(:,3))
%cla
InterpolationNumber = 1; %for visual clairty on scatter plot add extra dots
% robot plot followed by the target circle
x = [interpn(Position(:,1),InterpolationNumber)];%,double(subs(Y_d(1),str2sym('t'),1:1:
y = [interpn(Position(:,2),InterpolationNumber)];%,double(subs(Y_d(2),str2sym('t'),1:1:
z = [interpn(Position(:,3),InterpolationNumber)];%,double(subs(Y_d(3),str2sym('t'),1:1:
c = [interpn(TimeData,InterpolationNumber)*500];%,ones(360,1)'.*100];
s = [ones(length(interpn(Position(:,1),InterpolationNumber)),1)'];%,ones(360,1)'.*2]
scatter3(x,y,z,s,c)
%patch([x' nan],[y' nan],[z' nan],[t nan],'EdgeColor','interp','FaceColor','none')
xlabel("x (m)")
ylabel("y (m)")
zlabel("z (m)")

figure()
```

```
plot(TimeData,Position(:,1:3))
legend('x','y','z')
xlabel("Time (s)")
ylabel("Relative Position (m)")

Y_d_Data = []
diffY_d_Data = []
h_diff = diff(h)
Velocity = []

for i = 1:length(TimeData)
    Time = TimeData(i);
    Y_d_ForCalc = double(subs(Y_d,str2sym('t'),Time))';
    Y_d_Data = [Y_d_Data;Y_d_ForCalc];
    diffY_d_ForCalc = double(subs(diffY_d,str2sym('t'),Time));
    diffY_d_Data = [diffY_d_Data;diffY_d_ForCalc'];

    theta_1 = VelData(i,1);
    theta_1_dot = VelData(i,2);
    theta_2 = VelData(i,3);
    theta_2_dot = VelData(i,4);
    theta_3 = VelData(i,5);
    theta_3_dot = VelData(i,6);
    diffh_ForCalc = double(subs(subs(subs(subs(subs(subs(diffh,str2sym('diff(theta_1(t),
    Velocity = [Velocity;diffh_ForCalc'];

end

Error = Y_d_Data-Position
figure()
plot(TimeData,Error)
xlabel("Time (s)")
ylabel("Error (m)")
legend('x','y','z')

Error_dot = diffY_d_Data-Velocity
figure()
plot(TimeData,Error_dot)
xlabel("Time (s)")
ylabel("Error (m/s)")
legend('x','y','z')

function [h] = forwardKinematics(theta1,theta2,theta3,d2,d4,a2,a3)
```

```
    h(1) = a3*cos(theta1)*cos(theta2+theta3) + d4*cos(theta1)*sin(theta2+theta3)+a2*cos(
    h(2) = a3*sin(theta1)*cos(theta2+theta3) + d4*sin(theta1)*sin(theta2+theta3)+a2*sin(
    h(3) = -a3*sin(theta2+theta3)+d4*cos(theta2+theta3)-a2*sin(theta2);
    h = h';
end
```