Shaun Pritchard

EEL 5661 - Graduate

11/18/2022

HW3

Professor Roth

COT 4930 COT 5930 EEL 4930 EEL 5661 Robotic Applications (Fall 2022)

Homework 3

Problem 1: Forward and Inverse Kinematics and Resolved-Rate Motion Control (RRMC)

Consider the PUMA 560 robot.

1.1. Obtain a photo of the robot. The DH parameters of the PUMA robot are available. Explain, using the DH modeling convention, how all the PUMA a, d, θ and α DH parameters are found.

Puma 560 [Unimation]:: 6 axis, RRRRRR, stdDH, slowRNE

- viscous friction; params of 8/95;

j	theta	d	a	alpha	offset
1	q1	0	0	1.5708	0
2	q2	0	0.4318	0	0
3	q3	0.15005	0.0203	-1.5708	0
4	q4	0.4318	0	1.5708	0
5	q5	0	0	-1.5708	0
6	q6	0	0	0	0


UNIMATE
PUMA 500
UNIMATE

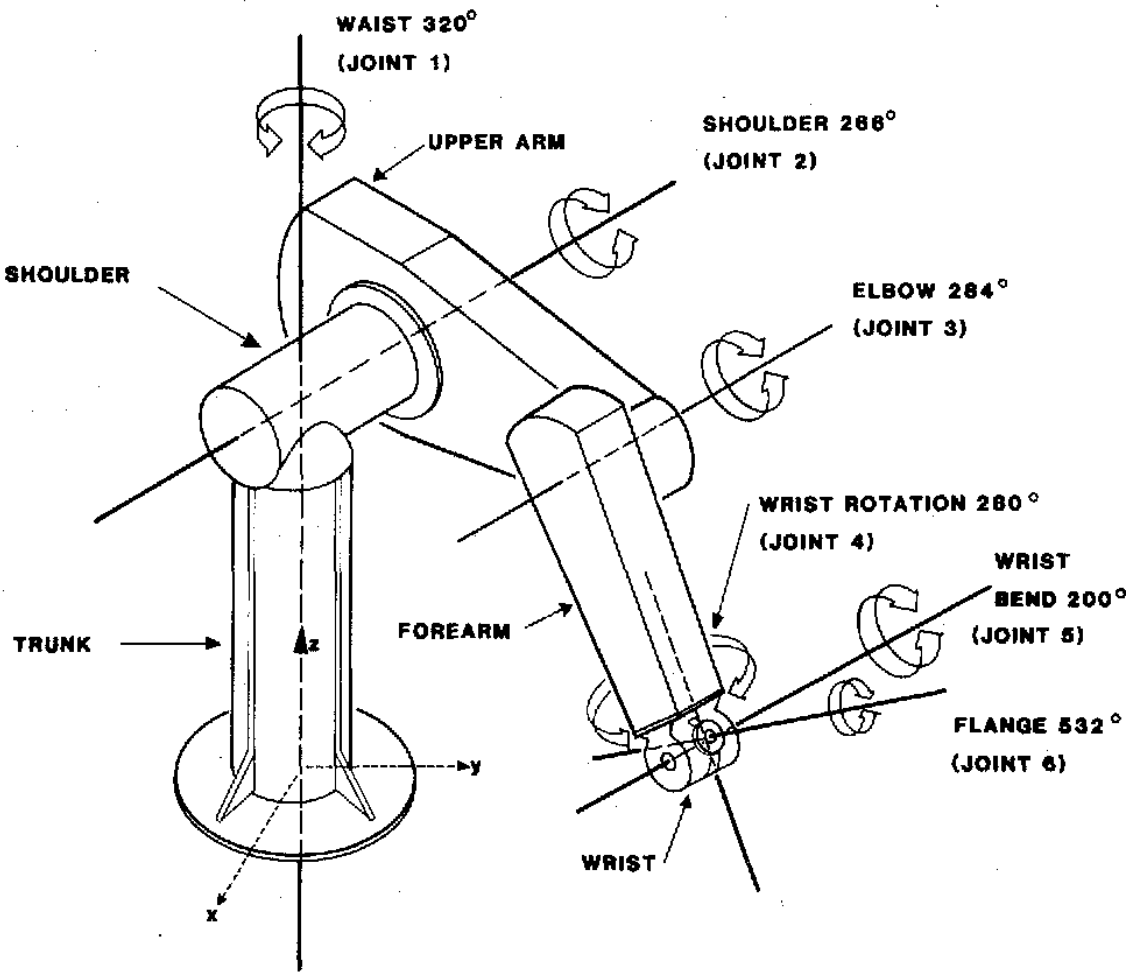
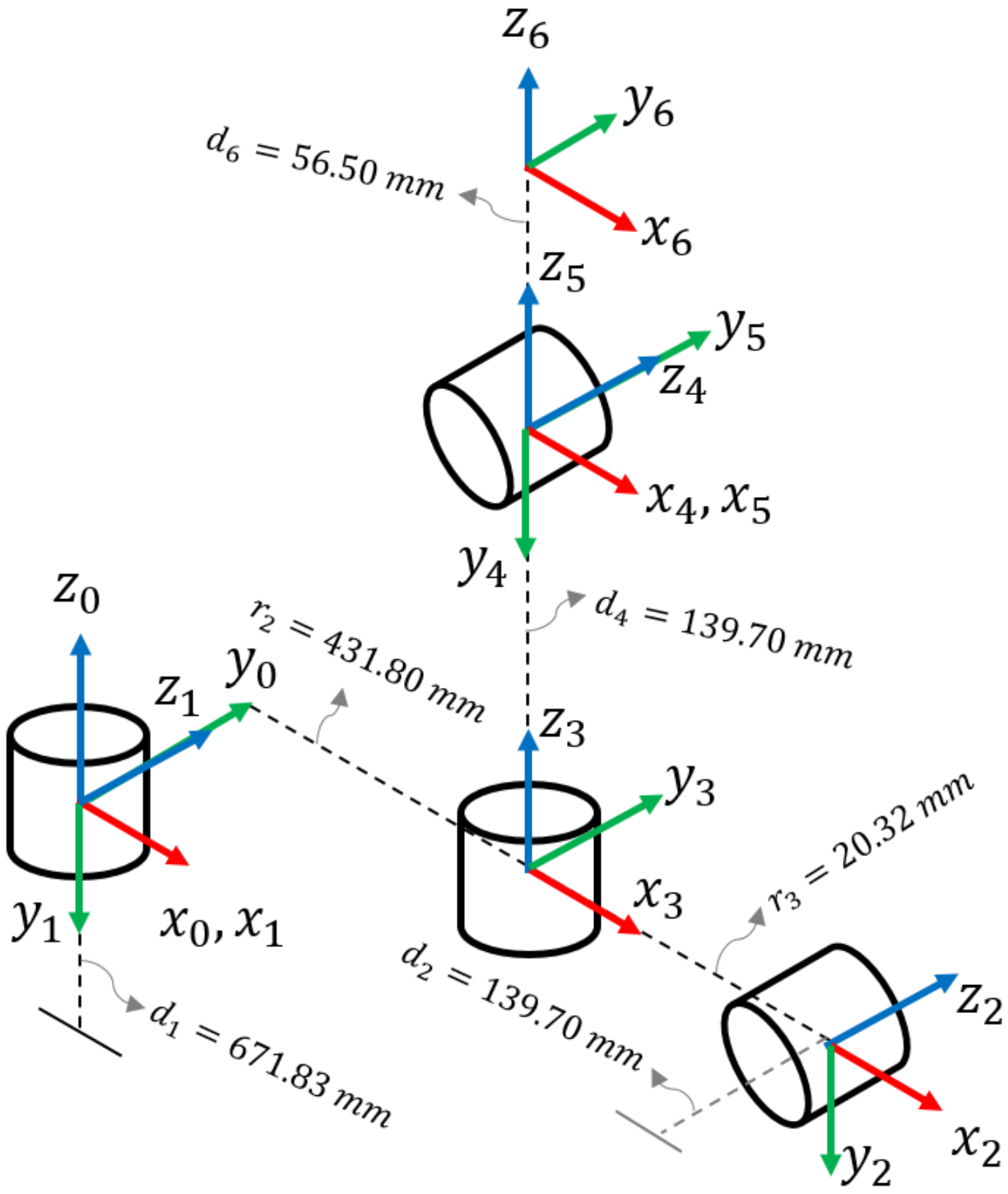


Fig. 1. PUMA 560 robot arm. Degrees of joint rotation and member identification.

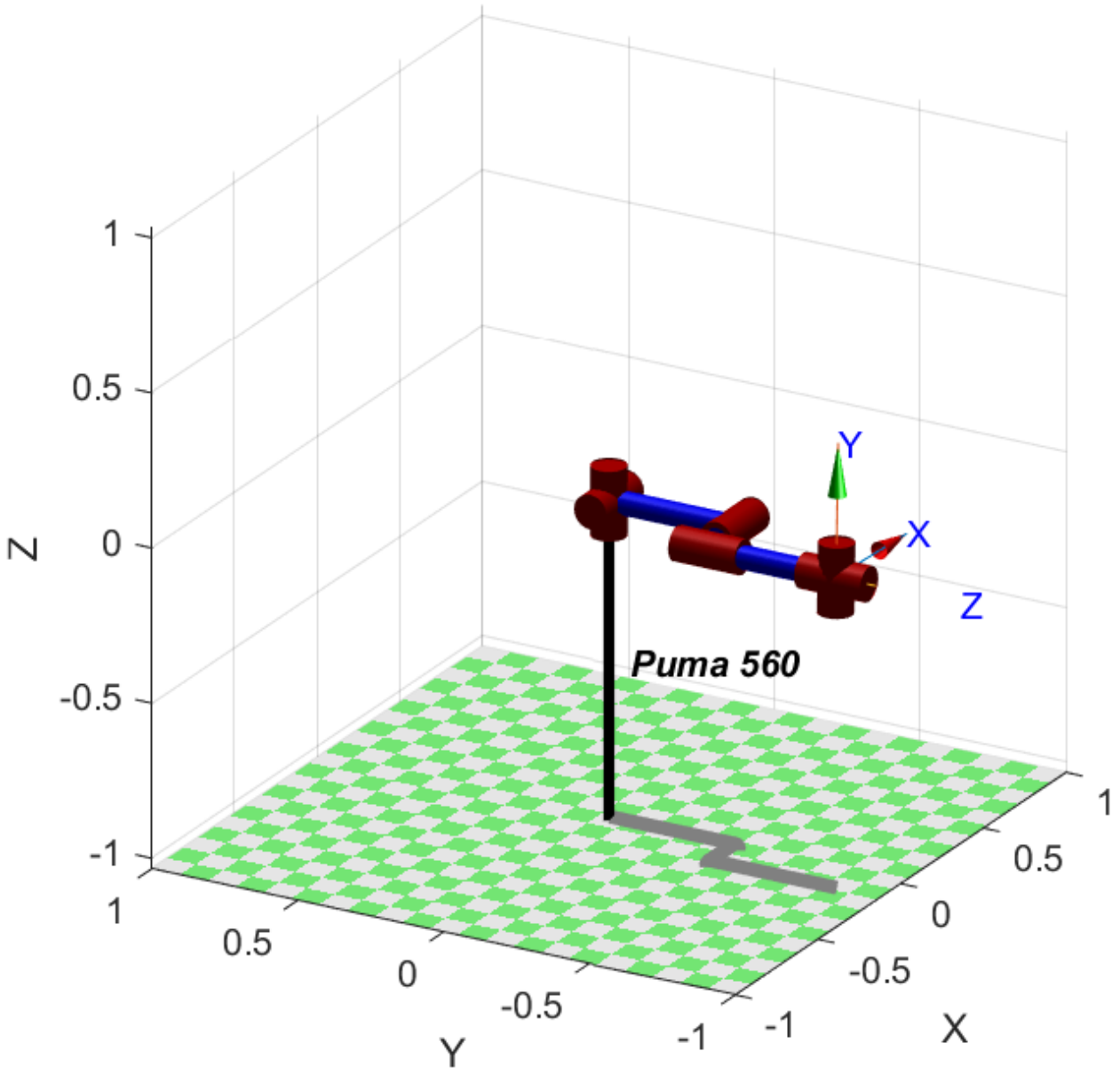


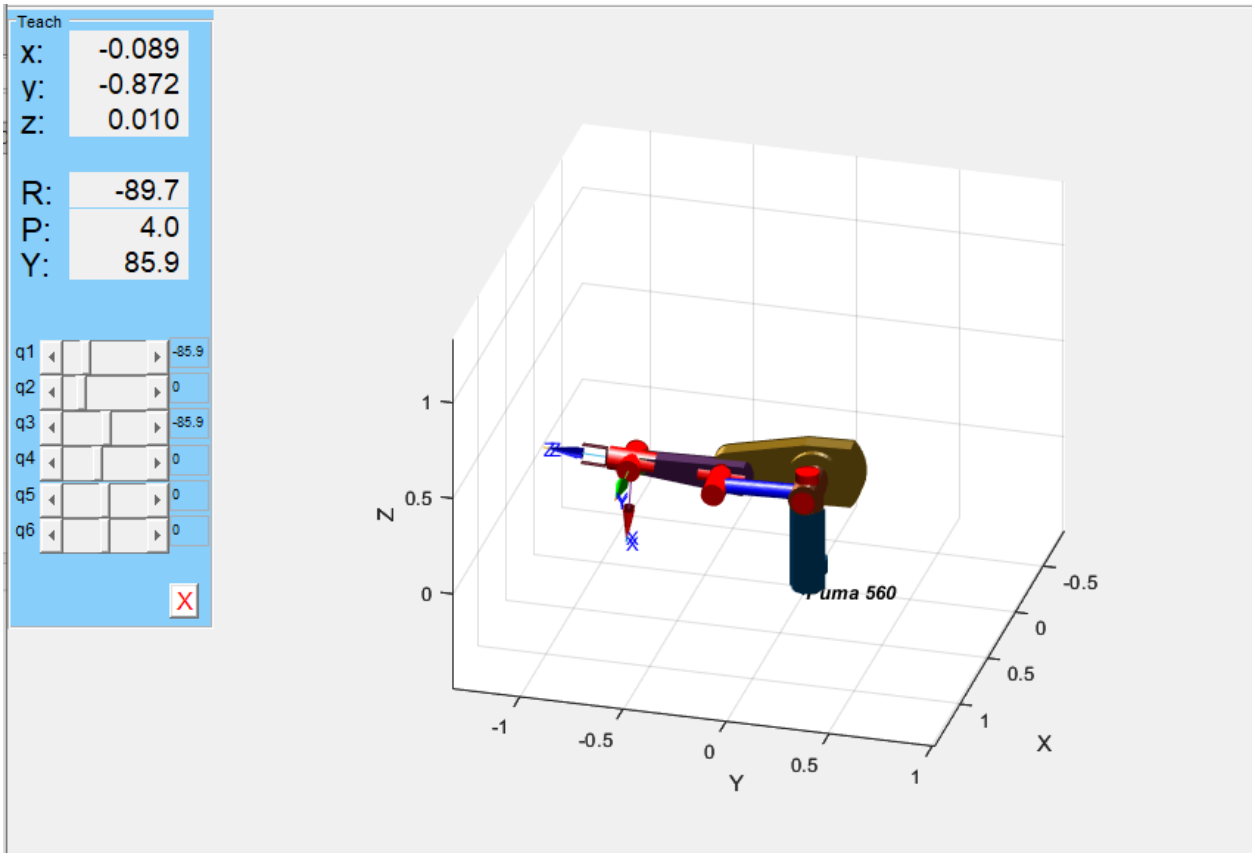
In a kinematic model, Denavit-Hartenberg (DH) relates rotation and displacement of a reference frame B0 to the last link or end effector frame Bn. An orthogonal rotation and displacement matrix is contained in the 4 by 4 homogeneous transformation matrix. An orientation matrix of 3 x 3

indicates the orientation of a joint or an end effector, whereas a displacement matrix of 3 x 1 indicates the position of a joint or end effector.

The DH parameters relate the rotation and displacement of frames Bn to frames Bn−1. The parameters are the joint angles θ, twist angle α, offset r, and link lengths d. The joint angle θ is the rotation at zn−1 to match the xn−1 to xn. The rotation at zn−1 includes all angular displacement at zn−1-axis. The joint angle at frame 1 is simply the angular displacement, θ1. This is true to all of the frames. In contrast, the twist angle α is the rotation at the xn to match the orientation of zn−1 to zn. For example, the twist angle α1 is −90◦ and it means that x1 rotates at −90◦ to match it with z0 to z1. All twist angles at each frame are defined by the earlier process.

1.2. Recall the task (that was discussed in class) in which the robot is moved at high speed, from its zero position P0 to a place P1 above the part that, in later steps, needs to be picked up. Pretend that the robot is a PUMA and also pretend that you (the operator who programs the robot) see the part lying on the conveyor at a repeatable position and orientation P3. Use the teach pendant to move the robot to the desired P1 configuration, right above the part. Then use the “fkine” forward kinematics command to find the tool’s homogenous transformation matrix (with respect to the robot’s base.


Puma 560
Y
X
Z
1
0.5
0
-0.5
-1
Z
Y
X




```
>> p560.fkine(qs)


ans =
    0.0050    0.9975    0.0706   -0.08856
   -0.0706    0.0707   -0.9950    -0.8724
   -0.9975         0    0.0707     0.0103
         0         0         0          1
>>
```

>> position directly above hypothetical part :: qs =[-1.5,0,-1.5,0,0,0]

qs = -1.5000 0 -1.5000 0 0 0

p560.teach(qs)

1.3. Create, using the teach pendant and the "fkine" command, the homogeneous transformation matrix of the part's pose P3, and also create the homogeneous transformation of an intermediate position P2 between P1 and P3. The PUMA is supposed to move slowly and carefully from P1 to P3 (via P2). Use "ikine" (or "ikine6s") inverse kinematics command to check if the inverse kinematics solutions agree with what was obtained during teaching-by-doing.

>> Position robot is in to pick up part from conveyer belt :: T=p560.fkine([1.5,0,1.5,0,0,0])

T =

0.0050 -0.9975 -0.0706 0.1499

0.0706 0.0707 -0.9950 -0.008103

0.9975 0 0.0707 0.05079

0 0 0 1

>> q1=p560.ikine(T)

>> q1 =1.5000 -0.0000 1.5000 0.3256 0.0000 -0.3256

Verify Results with fkine:

q1=p560.fkine(q1)

q1 =

0.0050 -0.9975 -0.0706 0.1499

0.0706 0.0707 -0.9950 -0.008103

0.9975 0 0.0707 0.05079

0 0 0 1

Yes, the Inverse Kinematics does presuppose the initial forward kinematics.

1.4. Let us assume a different scenario (than the one described in 1.2-1.3). Say that the homogeneous transformations for P1 (start of the careful motion) and P3 (end of the careful motion), that you have found earlier, were obtained using a camera and real-time machine vision (rather than by teaching-by-doing as done in 1.2-1.3). Plan, using Peter Corke's Simulink model of, a RRMC controlled motion to descend slowly from P1 to P3 by a straight-line motion along the Z axis. Plot the robot tool's cartesian motion, and also plot the PUMA joint variable angles (as function of t).

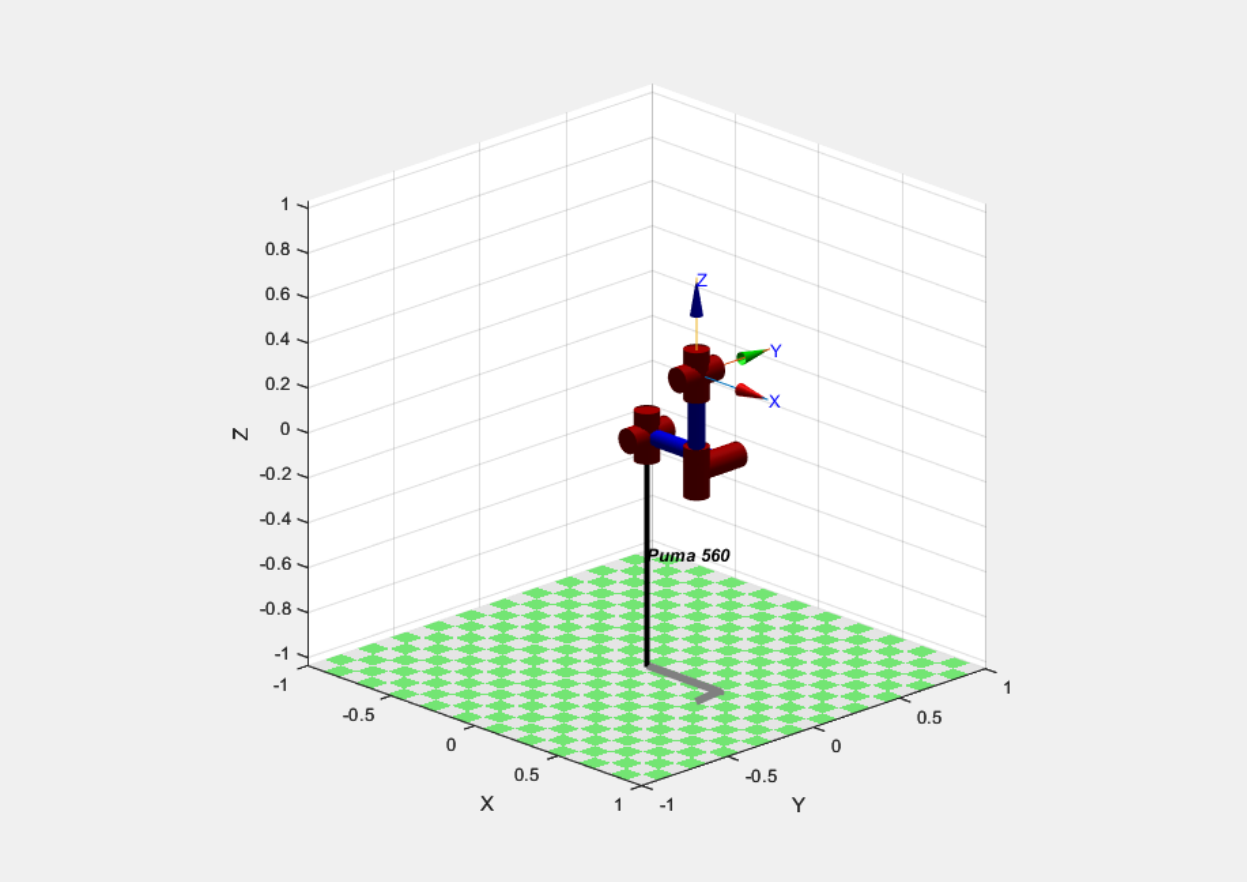
Cartesian velocity is a constant 0.05 m s−1 in the y-direction initially using the sl_rrmc model. Through resolved-rate motion control (RRMC) algorithm lets us generate straight line motion. A Jacobian is the matrix equivalent of the derivative of a vector-valued function of a vector with respect to a vector. If y = f(x) and x ∈ Rn and y ∈ Rm then the Jacobian is the m × n matrix. We use this value to obtain end effector coordinate frame and to map or resolve desired Cartesian velocity to joint velocity without explicitly requiring inverse kinematics. The below Simulink models depict this using the sl_rrmc model. Integrating a camera I would assume the camera pose would be the end effector therefore we can model and the kinematics accordingly to find the end-effector position.

RRMC Cartesian Velocity: [0 0 -0.1 0 0 0] (z-axis motion of -0.1 m/s)

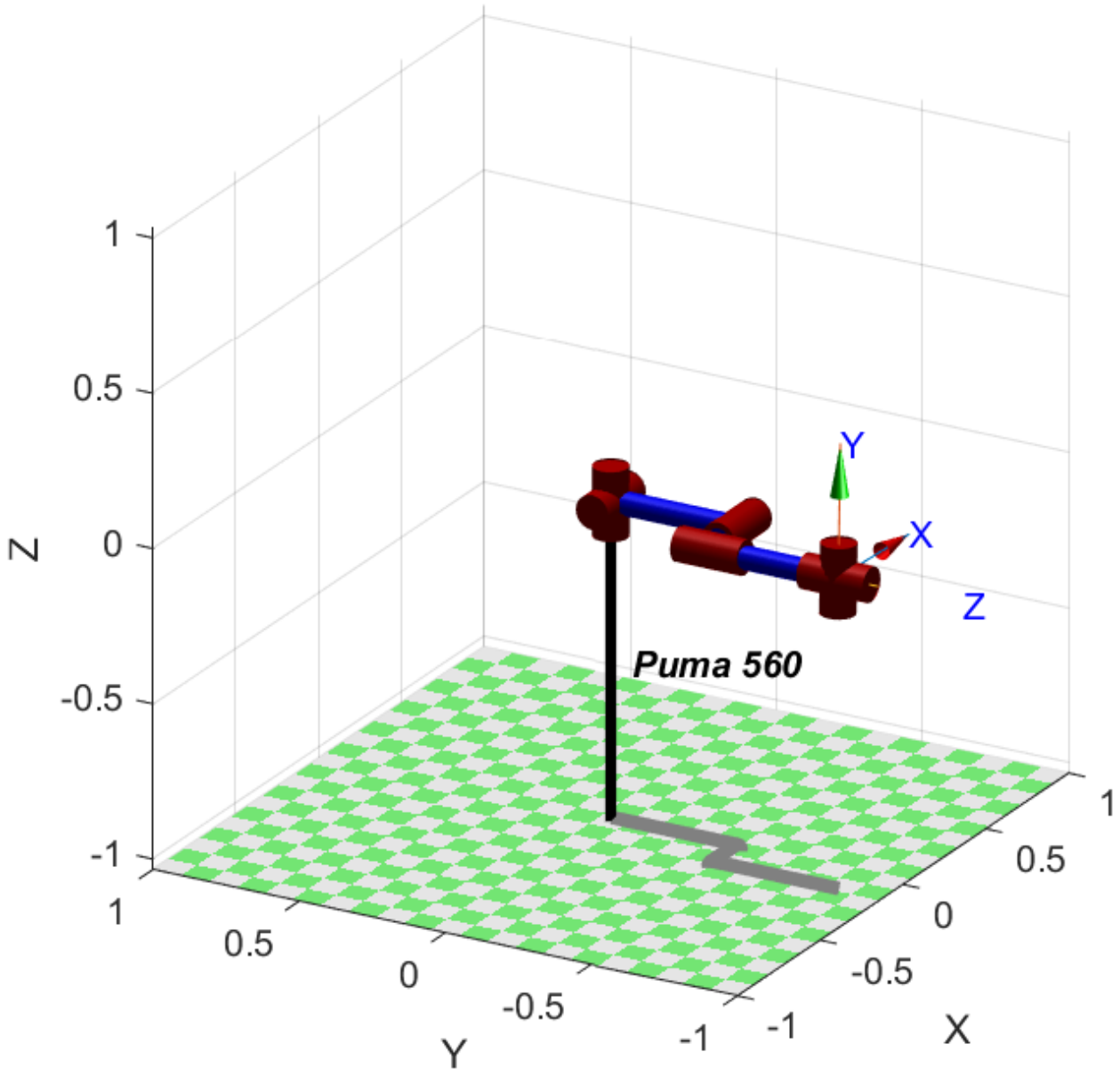
RRMC Initial Joint Angles: [0 -pi/3 pi/8 0 pi/4 0] (upright pose)

Initial K and T values : Gain of 0.5, Sample Time of 0.05

Initial Pose



Final Pose



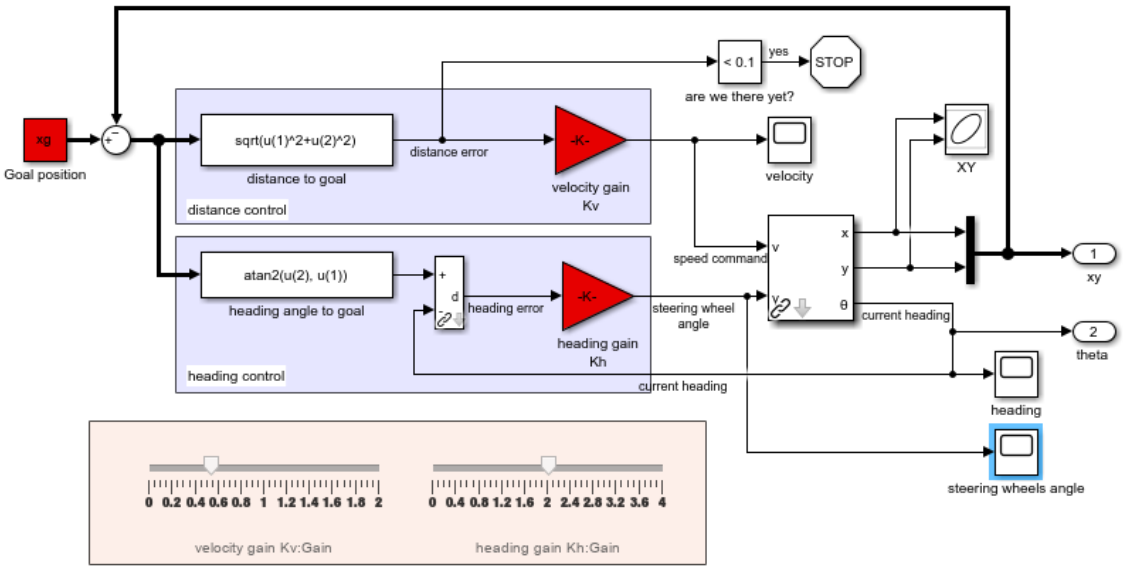
Where Z of end-effector would be the camera pose

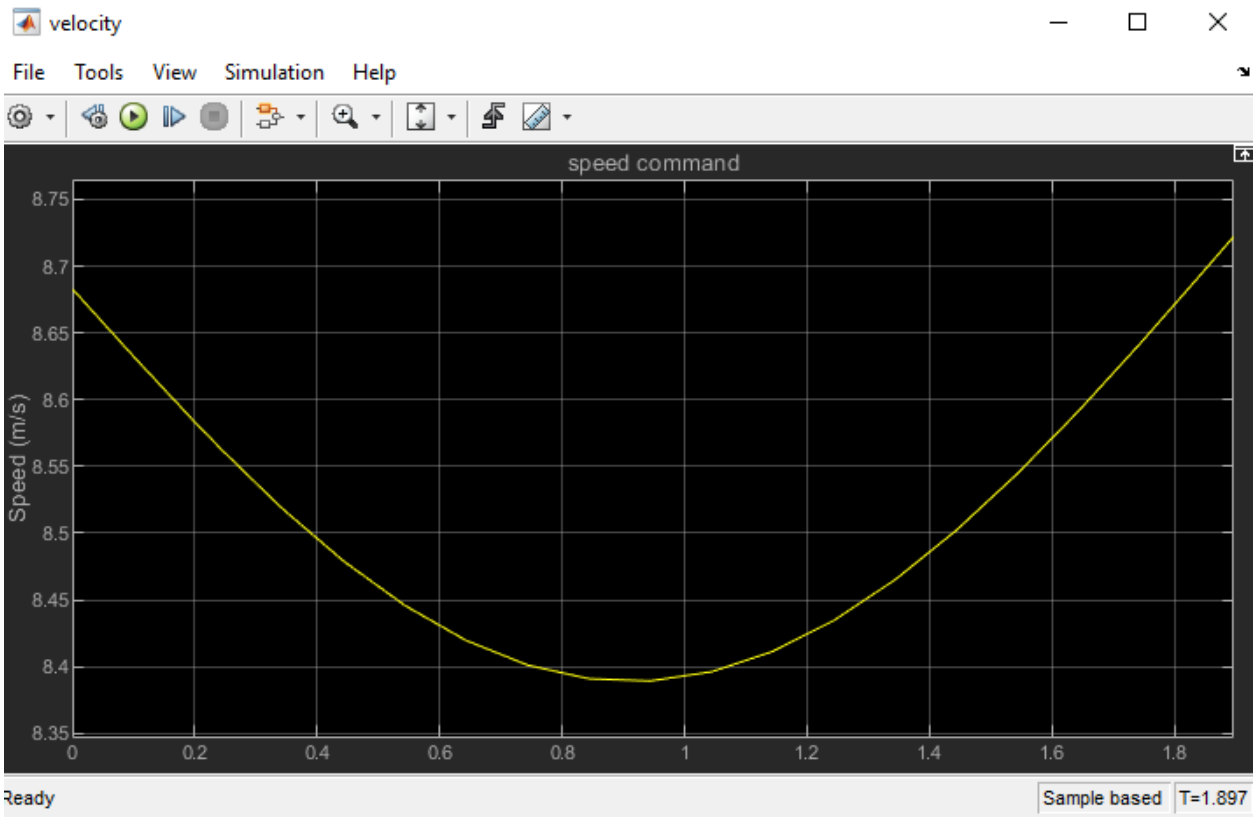
Problem 2: Mobile Robots Closed-Loop Controlled Motion

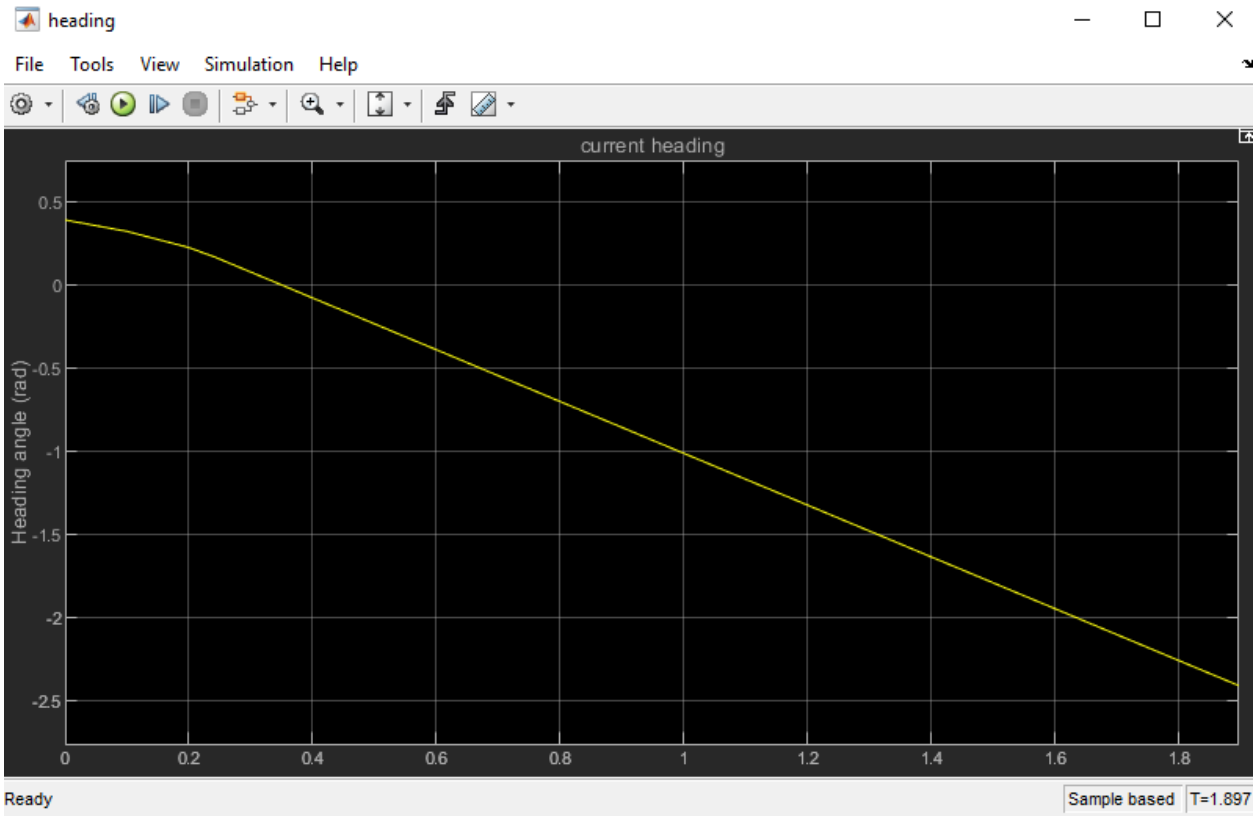
2.1. Demonstrate in Simulink how a differential-drive mobile robot moves, using closed-loop feedback control, from an initial pose [x, y, θ] = [7, 5, π/8] to a final position of [x, y] = [20, 15], in which the vehicle's orientation is unimportant. Plot the vehicle's XY trajectory, and also plot x(t), y(t) and θ(t), $V_l(t)$ and $V_r(t)$. You may use Corke's Simulink model (but replace the bicycle by a differential-drive vehicle) or use your own Simulink model.

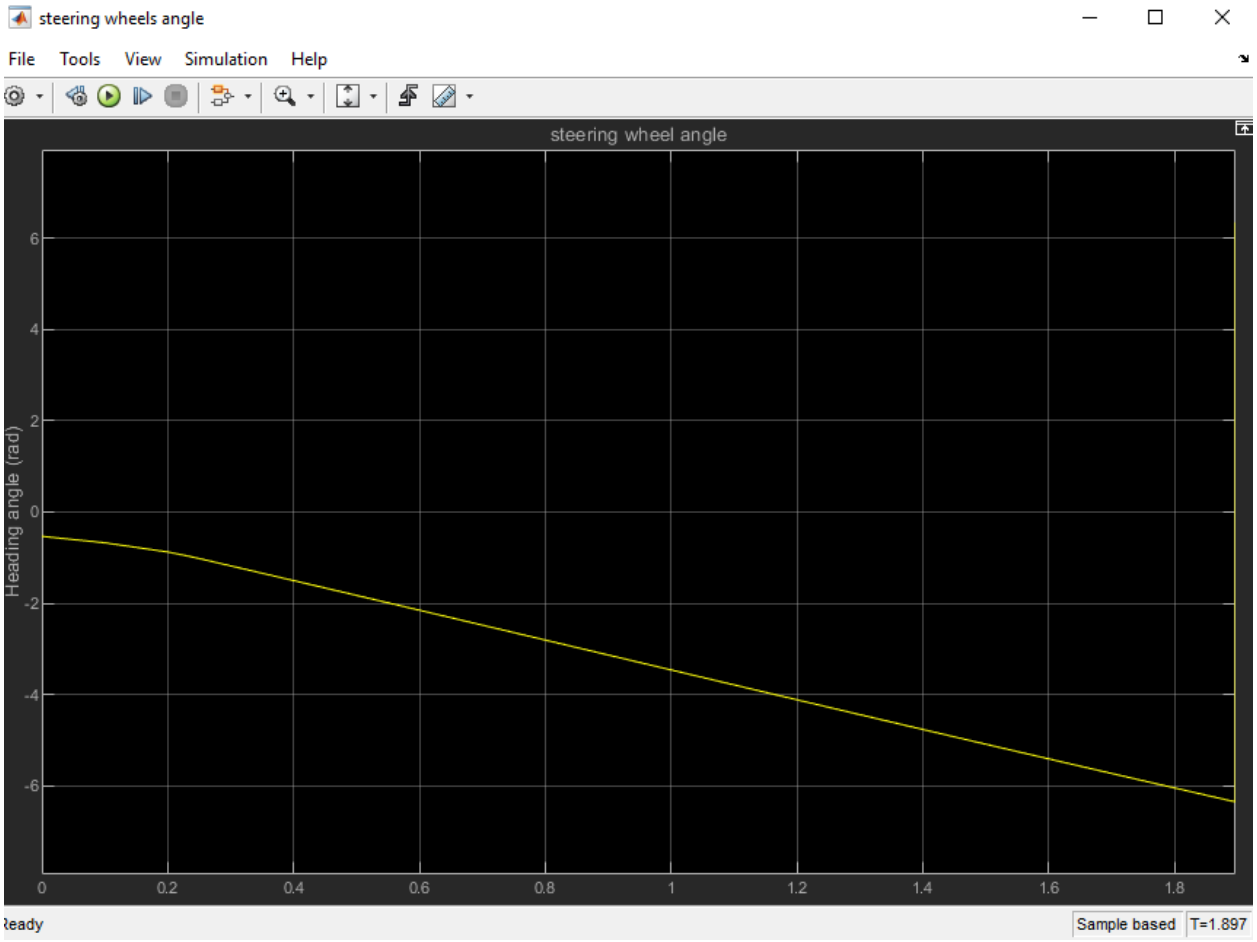
From the initial pose [x, y, θ] = [7, 5, π/8] to a final position of [x, y] = [20, 15] the following x(t), y(t) and θ(t), $V_l(t)$ and $V_r(t)$ are as follows:

```
x0=[7,5,pi/8];
xg=[20,15];
r=sim("sl_drivepoint",5);
q=r.find('y');
t=r.find('t')
mplot(t,q)
plot(q(:,1),q(:,2))
```




velocity
File Tools View Simulation Help
speed command
Speed (m/s)
8.75
8.7
8.65
8.6
8.55
8.5
8.45
8.4
8.35
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8
Ready
Sample based
T=1.897


heading
File Tools View Simulation Help
current heading
Heading angle (rad)
0.5
0
-0.5
-1
-1.5
-2
-2.5
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8
Ready
Sample based
T=1.897



2.2. Demonstrate in Simulink how a synchro mobile robot (also known as unicycle) moves, using closed-loop feedback control, from an initial pose [x, y, θ] = [2, 3, 0] to to follow a line of coefficients [3, -3, 4]. Plot the vehicle's XY trajectory, and also plot x(t), y(t), θ(t), and the two control signals. You may use Corke's Simulink model (but replace the bicycle by a differential drive vehicle) or use your own Simulink model.

From the initial pose [x, y, θ] = [2, 3, 0] to to follow a line of coefficients [3, -3, 4], the following x(t), y(t) and θ(t), $V_l(t)$ and $V_r(t)$ are as follows:

alpha -1.405648, beta 1.405648

going forwards

t =

0

0.0001

0.0008

0.0040

0.0201

0.0701

0.1201

0.1701

0.2201

0.2701

0.3201

0.3701

0.4201

0.4701

0.5201

0.5701

0.6201

0.6701

0.7201

0.7701

0.8201

0.8701

0.9201

0.9701

1.0201

1.0701

1.1201

1.1701

1.2201

1.2701

1.3201

1.3494

1.3494

1.3994

1.4494

1.4994

1.5494

1.5994

1.6494

1.6994

1.7494

1.7994

1.8494

1.8994

1.9494

1.9994

2.0494

2.0994

2.1494

2.1994

2.2494

2.2994

2.3494

2.3994

2.4494

2.4994

2.5494

2.5994

2.6494

2.6994

2.7494

2.7994

2.8494

2.8994

2.9494

2.9994

3.0494

3.0994

3.1494

3.1994

3.2494

3.2994

3.3494

3.3994

3.4494

3.4994

3.5494

3.5994

3.6494

3.6994

3.7494

3.7994

3.8494

3.8994

3.9494

3.9994

4.0494

4.0994

4.1494

4.1994

4.2494

4.2994

4.3494

4.3994

4.4494

4.4994

4.5494

4.5994

4.6494

4.6994

4.7494

4.7994

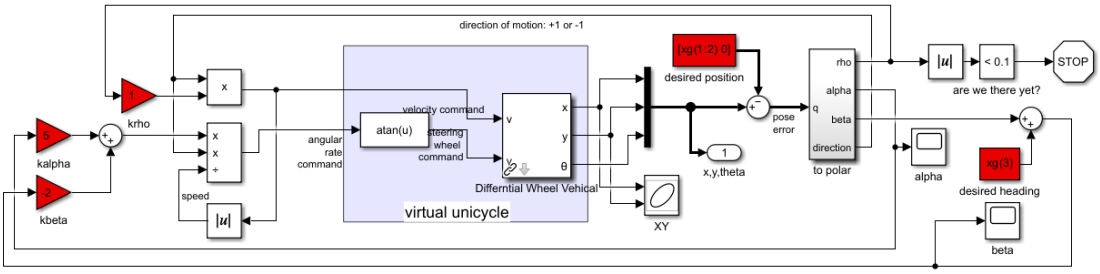
4.8494

4.8994

4.9494

4.9994

5.0000


direction of motion: +1 or -1
1
krho
6
kalpha
-2
kbeta
x
x
x
÷
speed
|u|
angular
rate
command
atan(u)
velocity command
steering
wheel
command
v
ω
x
y
θ
Differntial Wheel Vehical
virtual unicycle
[xg(1:2) 0]
desired position
1
x,y,theta
XY
pose
error
rho
alpha
q
beta
direction
to polar
|u|
< 0.1
are we there yet?
STOP
alpha
xg(3)
desired heading
beta

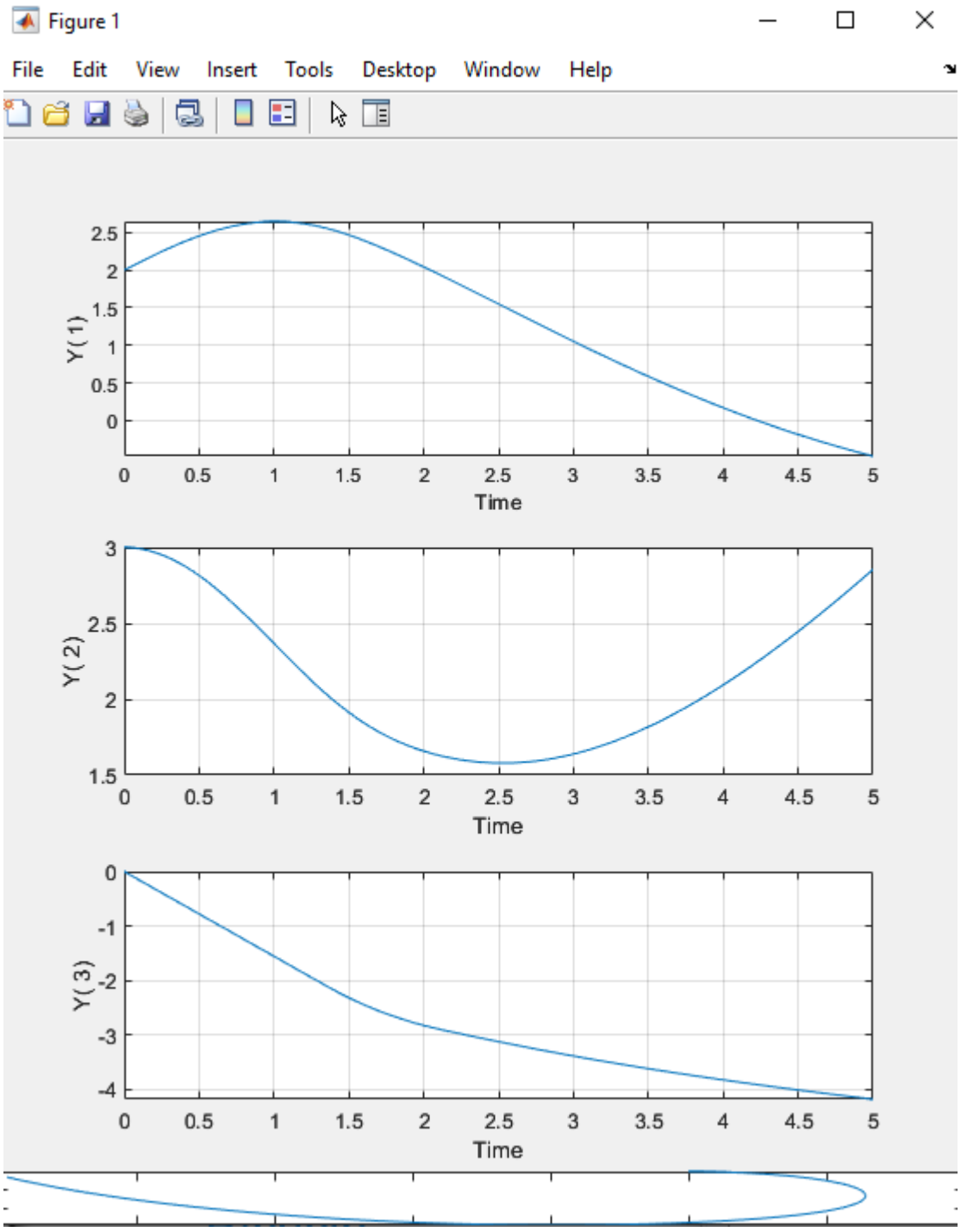
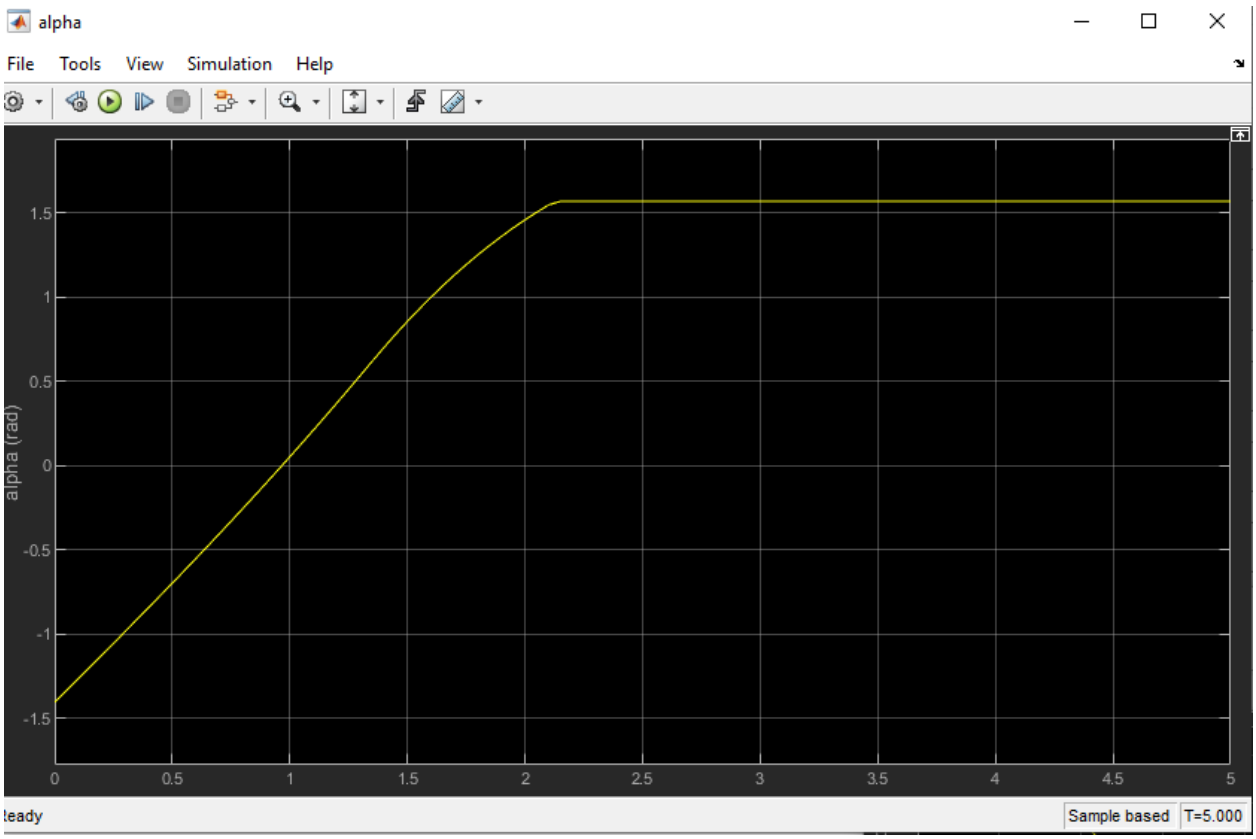
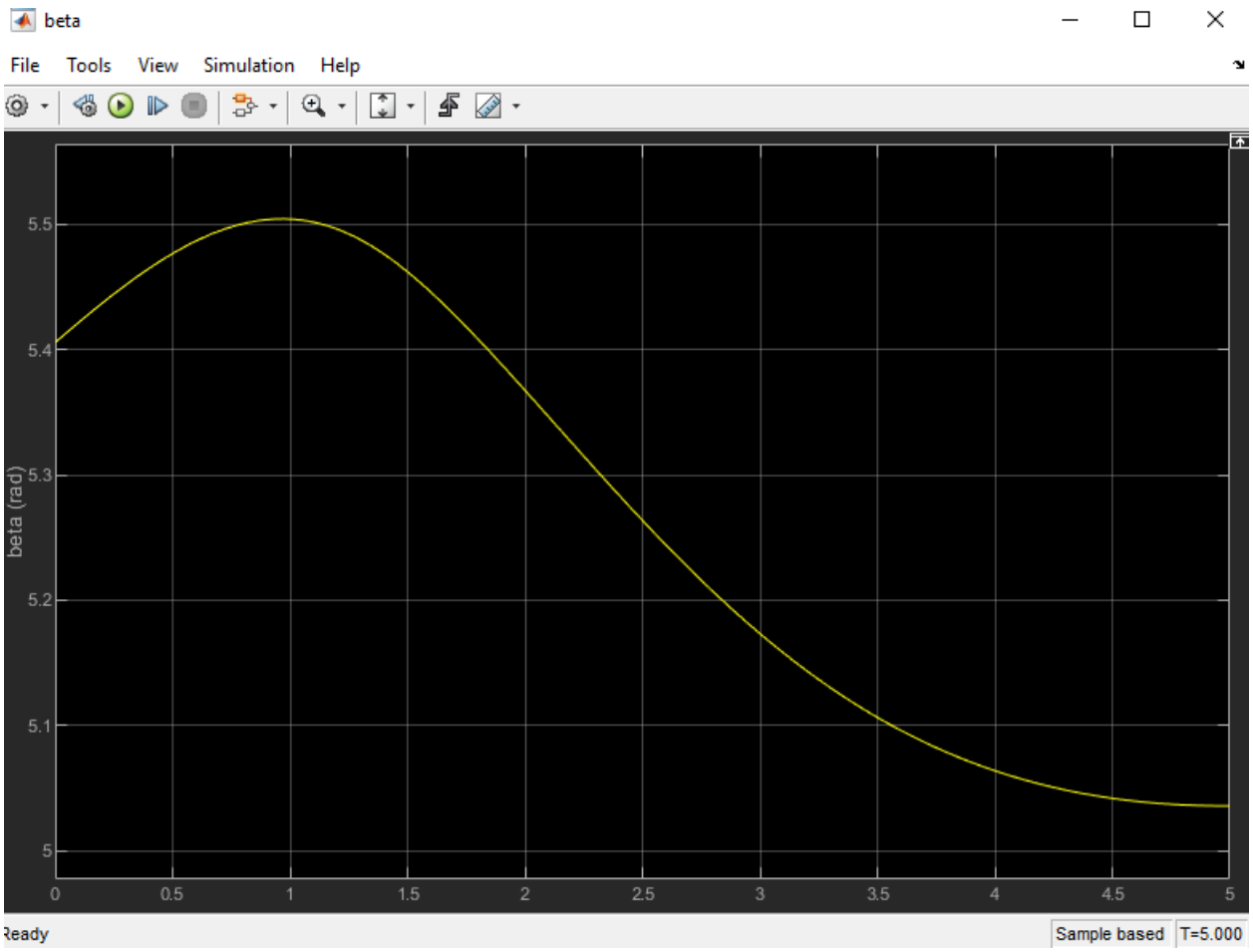


Figure 1
File Edit View Insert Tools Desktop Window Help
Y(1)
Time
Y(2)
Time
Y(3)
Time


alpha
File
Tools
View
Simulation
Help
alpha (rad)
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
eady
Sample based
T=5.000



Problem 3: Joint Space Motion Planning

A 3DOF XYZ manipulator consists of three prismatic joints that move simultaneously for a variety of tool positioning tasks. The maximum velocity and acceleration of joint X are 0.7[m/s] and 3.1[m/s^2], respectively. The maximum velocity and acceleration of joint Y are 0.4[m/s] and 2.1[m/s^2], respectively. The maximum velocity and acceleration of joint Z are 1.0[m/s] and 1.0[m/s^2]. Need to plan trapezoidal velocity profiles for X, Y and Z as the (X,Y,Z) tool coordinates

change from (4 , 1, 1) to (-2 , 3.5, 1) [m] (each).

Shem Pritchard EE1561-G ~~ΔX=~~ ΔY=2.5 ~~ΔZ~~

[4,1,1] [-2, 3.5, 1]

Joint 1 — Joint 2 — Joint 3

Joint 1	Joint 2	Joint 3
$V_{max} = 0.7$	$V_{max} = 0.4$	$V_{max} = 1$
$a_{i\,max} = 3.1$	$a_{max} = 2.1$	$a_{i\,max} = 1$
$\tau_1 = 0.226$	$\tau = 0.190$	$\tau = 1$
$\lvert\Delta q_x\rvert = 6.2$	$\lvert\Delta q_y\rvert = 2.5$	$\lvert\Delta q_z\rvert = 0$

eq: $\sqrt{a_{i\,max}\lvert\Delta q_i\rvert} \le V_{max}$

$\Delta X = 6.2$
$\Delta Y = 2.5$
$\Delta Z = 0$

x = True
$\sqrt{3.1 \cdot 6.2} \le 0.7$

y = ~~true~~
$\sqrt{0.4 \cdot 2.1} \ge 0.19$

Z = ~~true~~
$\sqrt{1 \cdot 1} \ge 0$

$T_1\,opt \quad \frac{0.7}{3.1} + \frac{6.2}{0.8} \quad X^{T1} = 7.77$

$T_2\,opt \quad \frac{0.4}{2.1} + \frac{2.5}{0.4} \quad y^{t2}\; 6.440$

$T_3\,opt \quad \frac{1}{1} + \frac{0}{1} \quad Z^{t3} = 1$

X at max Velocity & acceleration is slower

~~y~~ T2 not acceleration Velocity ~~Ofcom Velocity~~

T3 ?

$V_2\,opt = [a_{2max} \cdot T_m + \sqrt{a_{2max}^2 T_m^2 - 4 a_{max} \lvert\Delta q_z\rvert}] \times 0.5$

$= [2.1 \cdot (7.77) - \sqrt{(2.1^2)(7.77)^2 - 4(2.1)(2.5)}] \times 0.5$

$= (16.317 - 15.66) \times 0.5 = 3.285$

$V_3\,opt = [(1)(0) - \sqrt{(1^2)\cdot(0)^2 - 4(1)(1)}] \times 0.5 = 0$

$T_M = 7.77$

$V_1\,opt = 0.7$	$V_2\,opt = 3285$	$V_3\,opt = 0$
$a_1\,op = 3.1$,	$a_2\,op = 2.1$	$a_3\,opt = 1$

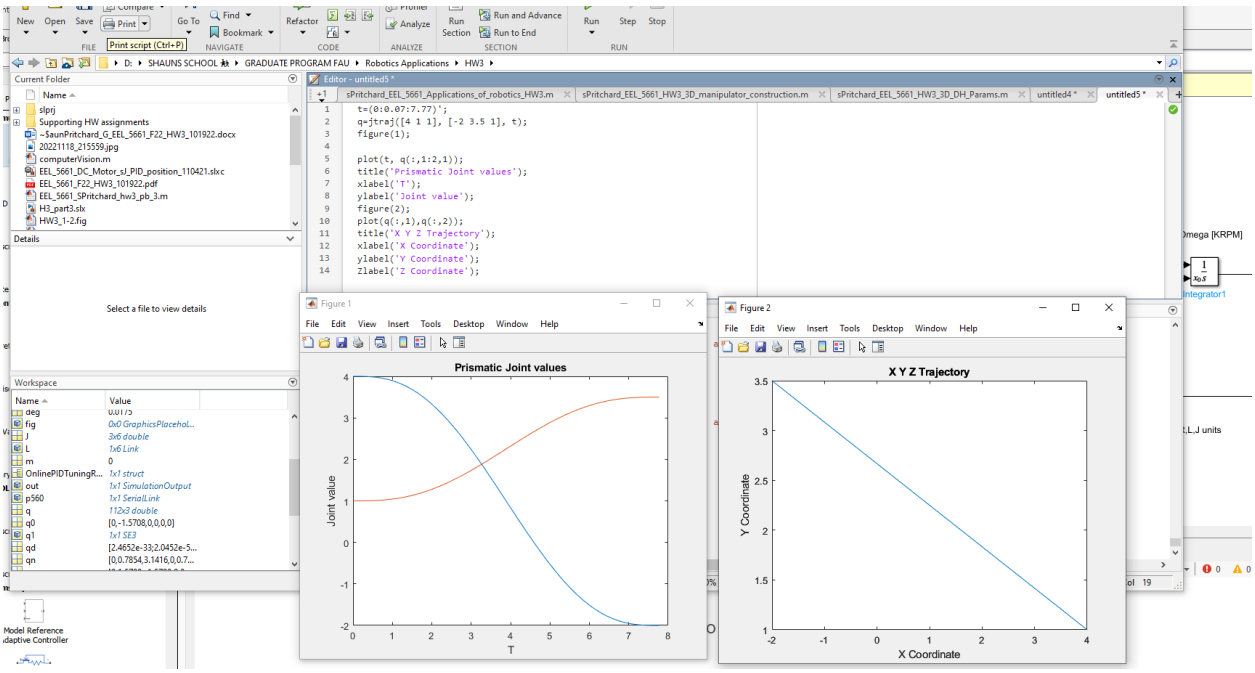
3.1. Decide what should be the synchronized motion time T, whether each velocity profile be trapezoidal or triangular and what should be the acceleration/deceleration times. Also provide information about the chosen velocities and accelerations.

Final Velocities & Accelerations:

- Xv,opt = 0.7 m/s
- Xa,opt = 3.1 m/s2
- Yv,opt = 0.8 m/s (0.3285)
- Ya,opt = 2.9 m/s2
- Zv,opt = 0 m/s (0)
- Za,opt = 1 m/s

Having solved for the parameters for the slowest joint X, it is now possible to calculate the parameters for the faster joints which in order to sync the Tm parameter as a result of solving for the parameters for the slower joint. A trapezoidal velocity profile can be chosen as the velocity profile to be implemented on the prismatic joints.

3.2. Use MATLAB "jtraj" command (or another related command such as "mtraj") to implement the needed motion. See if the RTB command that you choose allows you to integrate all the velocity and acceleration limits, or if it would lead to a solution that is different than the one that is obtained in 3.1.



Problem 4: DC Motor Control

Use lectures notes “DC Motor Simulation” and “Simulation DC Motor PID Tuning” as references and use the posted Simulink models (in Canvas module “Fall 2021 Resource Material”) so that you don’t have to recreate the Simulink diagram, just to slightly modify existing diagrams. No need to use the motor’s simplified transfer function – the provided Simulink diagrams provide exact models of the motor.

Replace the smaller motor E-508A parameters by a larger motor E-510C parameters (based on the data sheets posted in the lecture).

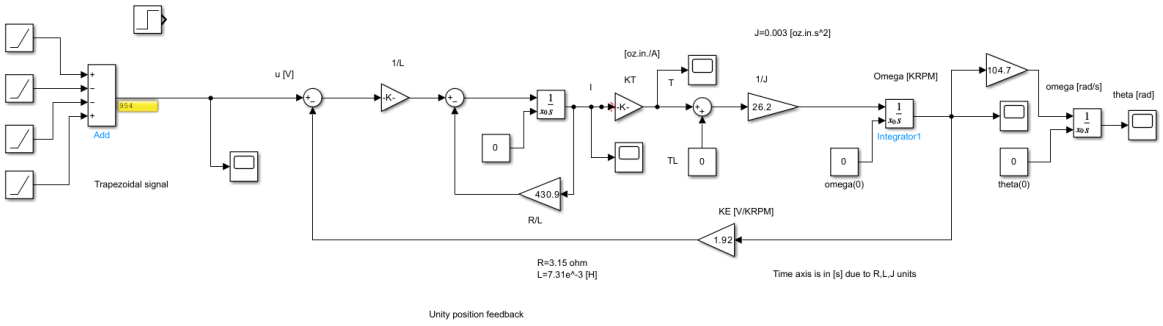
In both assignments (below) **the goal is to advance the motor’s angle θ from an initial angle θ(0) = 0.1[rad] to a final angle θ(t_f) = 0.5 [rad], in t_f = 35ms, or as fast as possible**. Assume that the total moment of inertia (motor plus load) is J = 10J_m, where J_m is listed in the data sheets. In each of the two tasks check that the motor’s peak torque value and motor’s maximum pulse current value are not exceeded. If needed slow the motor down a little. Also be sure that the output angle does not overshoot by more than 10%.

In each of the problems below, show the Simulink diagram, show all the relevant graphical results and add brief explanations.

4.1 Use the DC motor in open loop control. That is, leave the feedback of Ω via K_E, but do not create an additional unity feedback for Ω. No need to use any PID controller block. The task is to create a specific trapezoidal velocity profile input command signal, so that the motor's angle moves by the right amount by the specified timing.

I set all slopes to 185 first staring at 10ms and each other in increments of 4ms after each proceeding slope.

1. L = 0.00137 mH
2. Peak Torque=140 oz.in
3. R = 3.15 ohms
4. Jm = 0.020 oz.in.s2 of (Jm = 10*Jm / Jm = JL = 0.019)
5. JL = 0.020 oz.in.s2
6. KE = 6.91 V/KRPM
7. KT = 9.34 oz.in/A

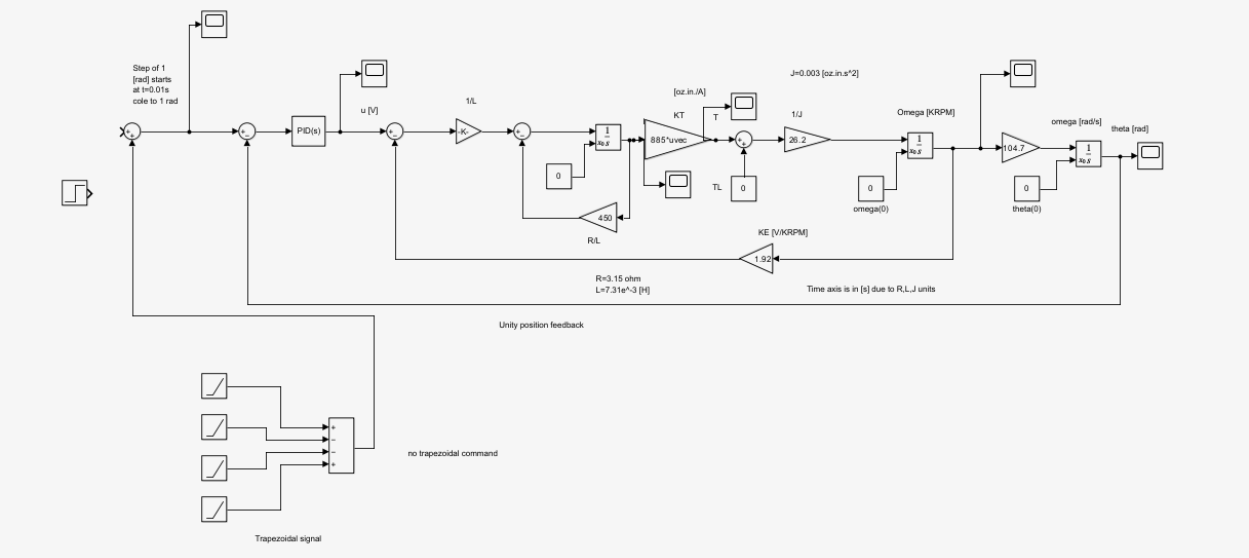


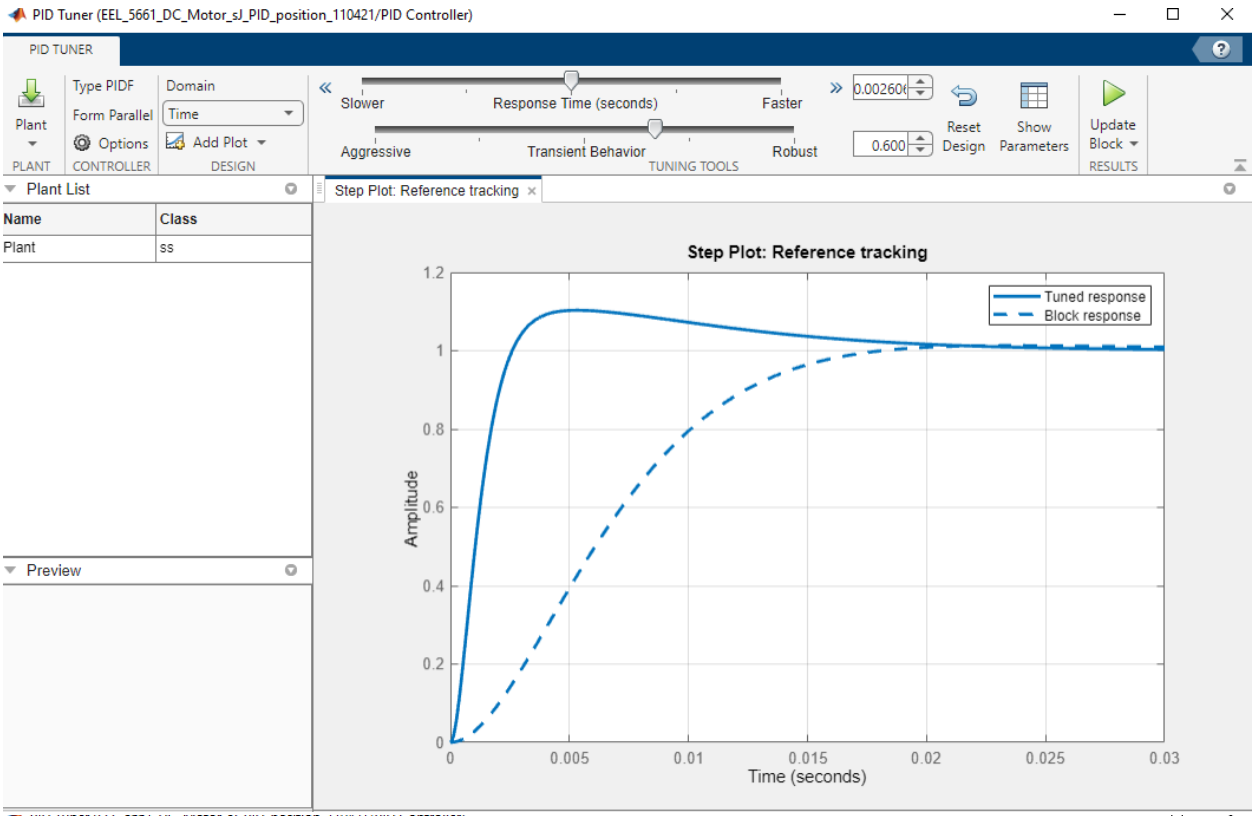
4.2 Use the DC motor in closed loop velocity control. That is, use the given diagram in which you leave the feedback of Ω via K_E, and add unity feedback for Ω. Need to tune a PID controller block. Again, the task is to create a specific trapezoidal velocity profile input command signal, so that the motor's angle moves by the right amount by the specified timing.

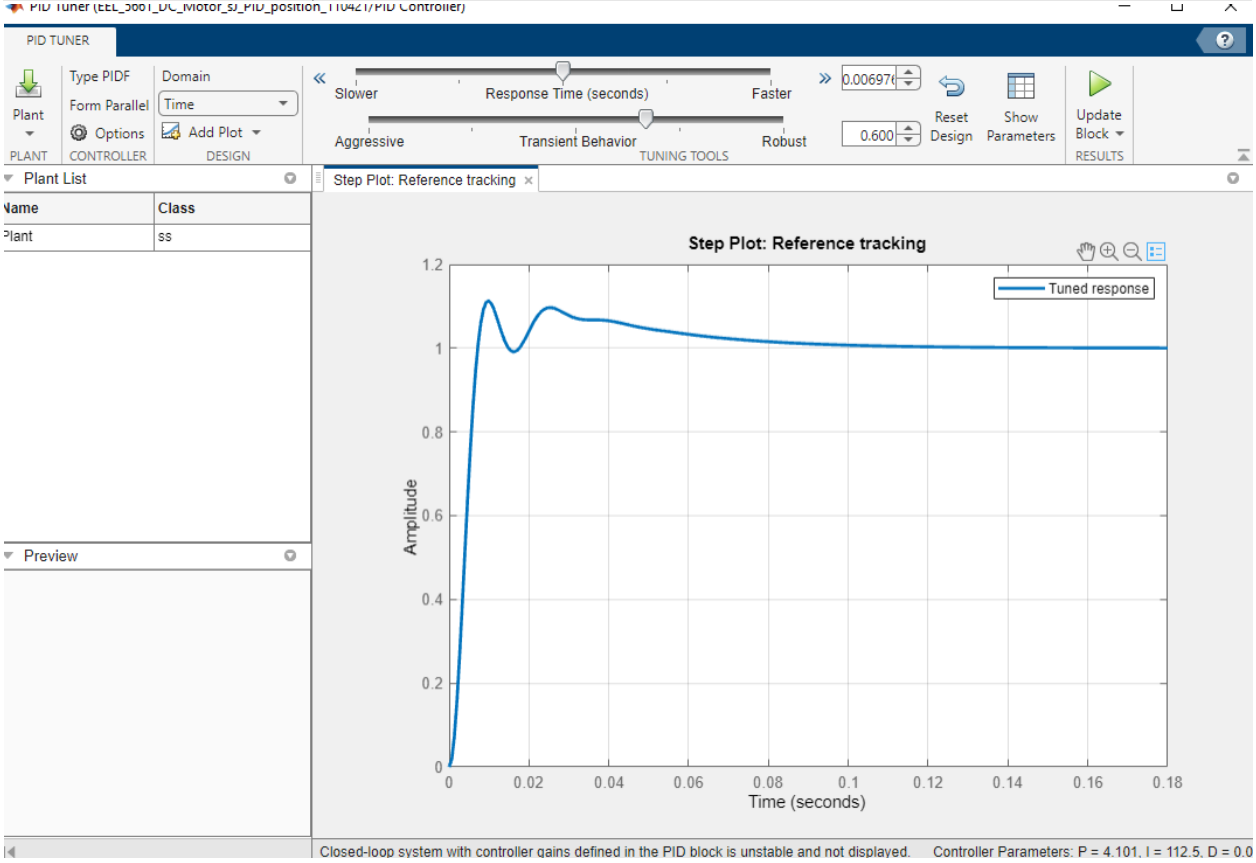
A: It was necessary to remove the step input from the basic PID model in order to solve problem 5.2. As a result of these alterations, I was able to change the motor's angle from 0 to 0.5 rads in 40ms, starting at 10ms and ending at 50ms, as shown below together with the PID tuner.signal. Using these changes, I was able to change the motor's angle from 0 to 0.5 rads in approximately 40ms, and I then set the theta value at the end to 0 to match the initial angle of 0 at the beginning. After that, I changed the slope of all the ramps to 24 percent each, and I made sure they all started 10 milliseconds after the previous ramp, with the first ramp starting immediately after the previous ramp. After editing the ramps, I changed the parameters of the motor in the diagram to those of the E-510C motor once I had edited the ramps. As a last step, I tuned the PID to make sure that the output would not be changed by the looped input. In order to make this closed loop system, all the parameters for this question are the same as those for 5.1, so all I had to do was add the PID control and velocity feedback in order to complete it. In the experiment, I used trapezoidal signals with slope 240 at steps of 0.001 and 0.011 and 0.021 and 0.031 and 0.031 respectively for acceleration, constant speed, deceleration, plateau, respectively. As you can see from the trapezoidal profile below, there is a 10 ms delay between each step, and the accelerating begins at 1 ms. This distance allows us to accelerate, reach constant speed, decelerate and then stop at an angle of 0.5 radians after only 40 ms (trapezoidal profile).

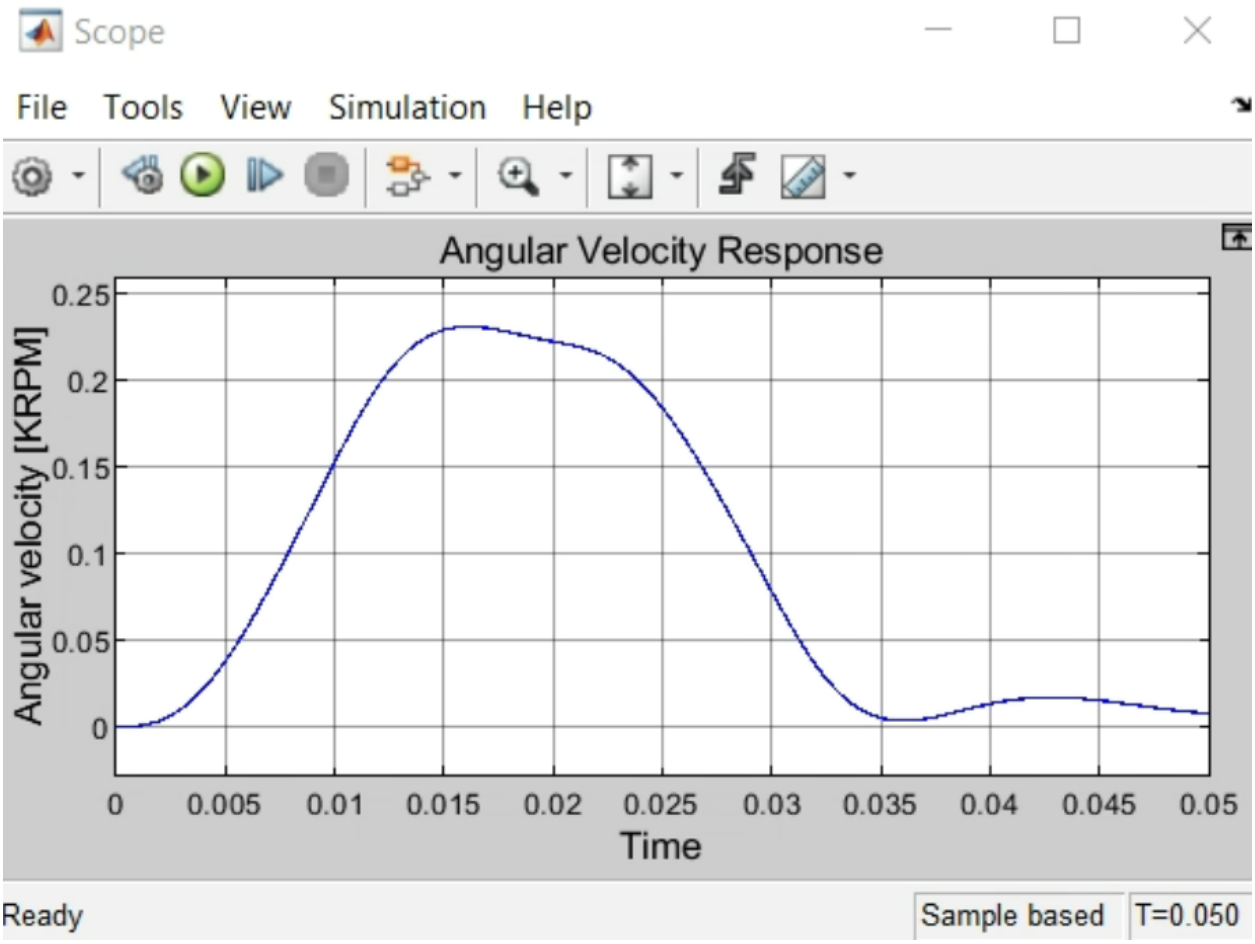
Parameters:

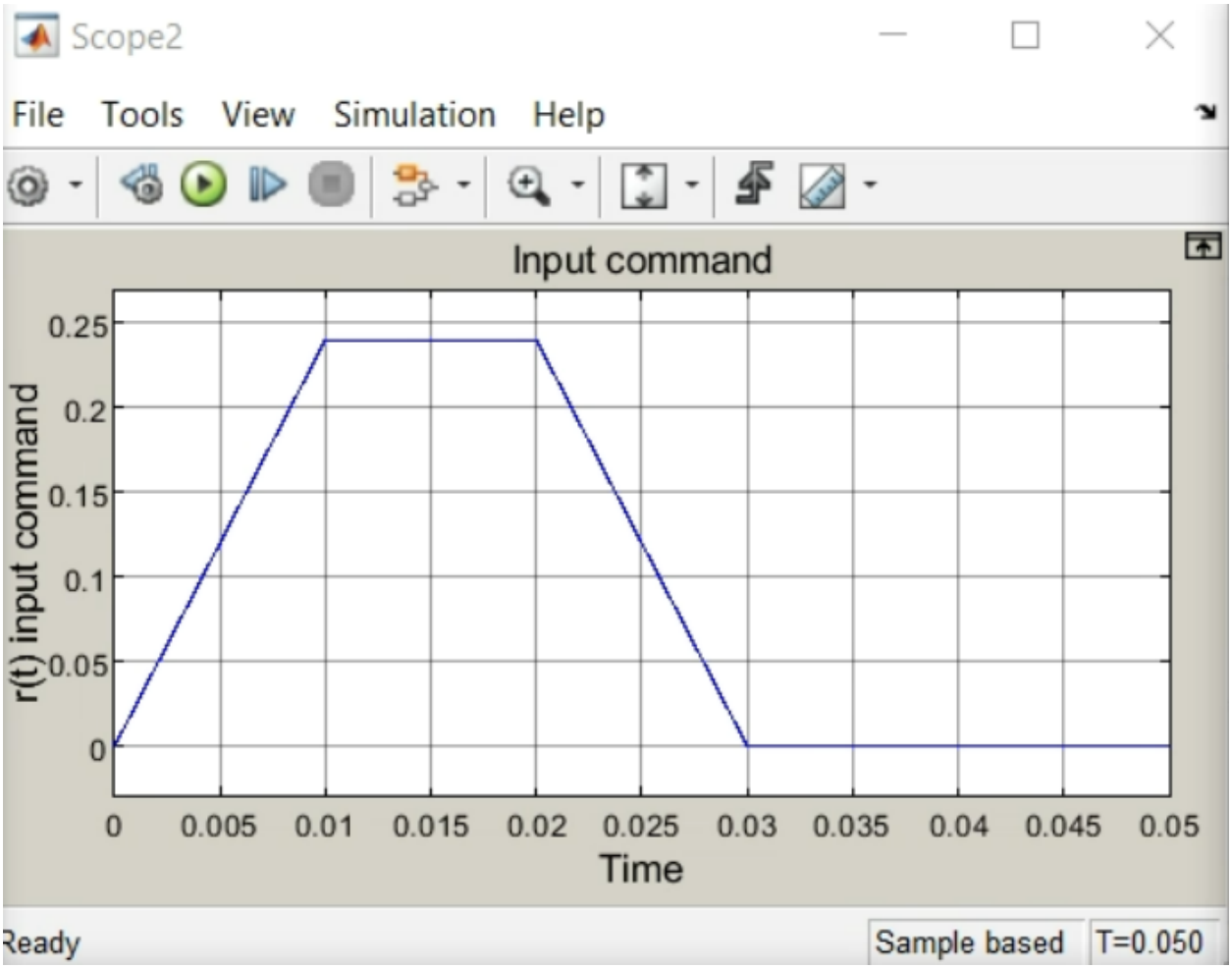
8. L = 0.00731 mH
9. Peak Torque=140 oz.in
10. R = 3.15 ohms
11. Jm = 0.020 oz.in.s2 of (Jm = 10*Jm / Jm = JL = 0.019)
12. JL = 0.020 oz.in.s2
13. KE = 6.91 V/KRPM
14. KT = 9.34 oz.in/A

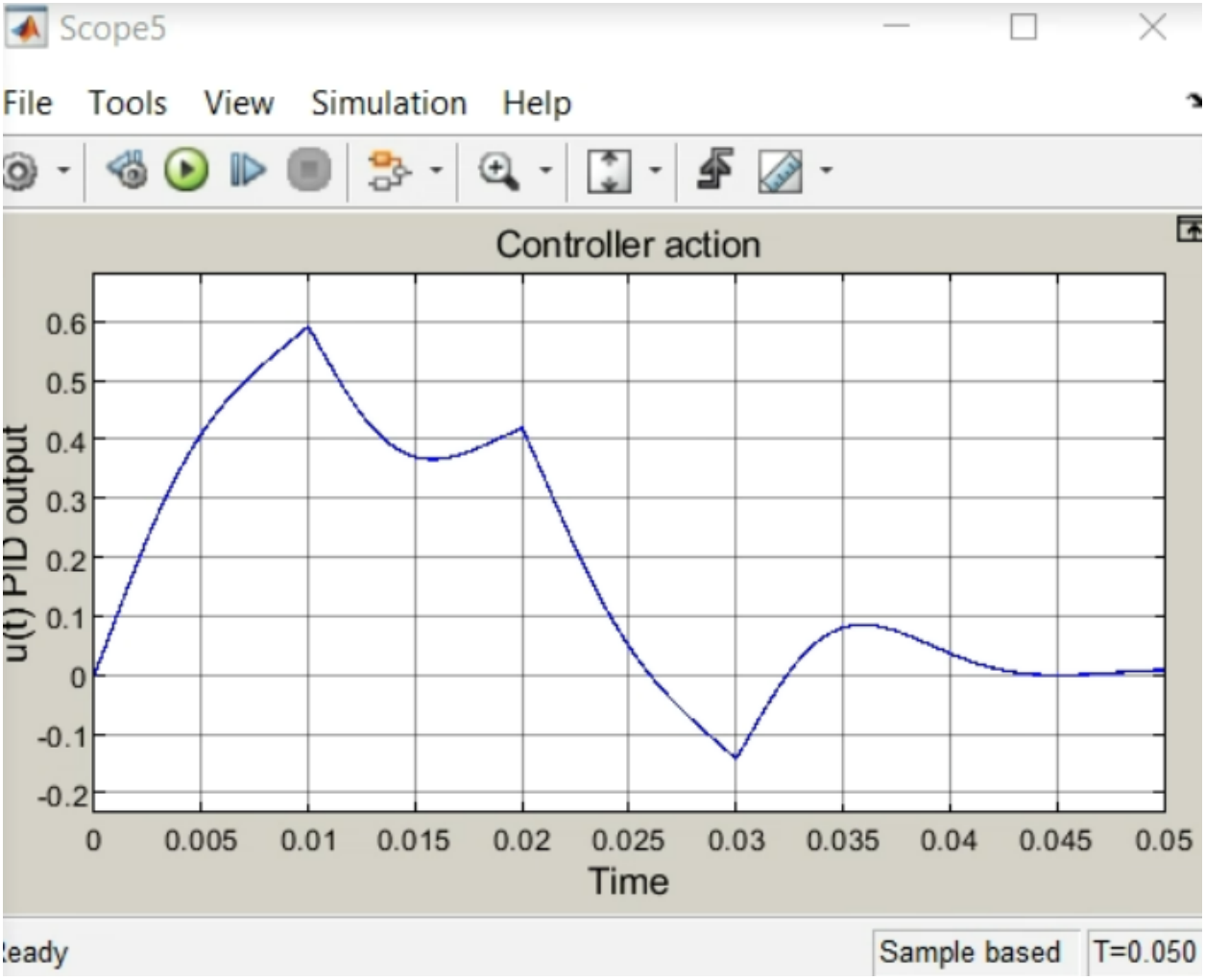


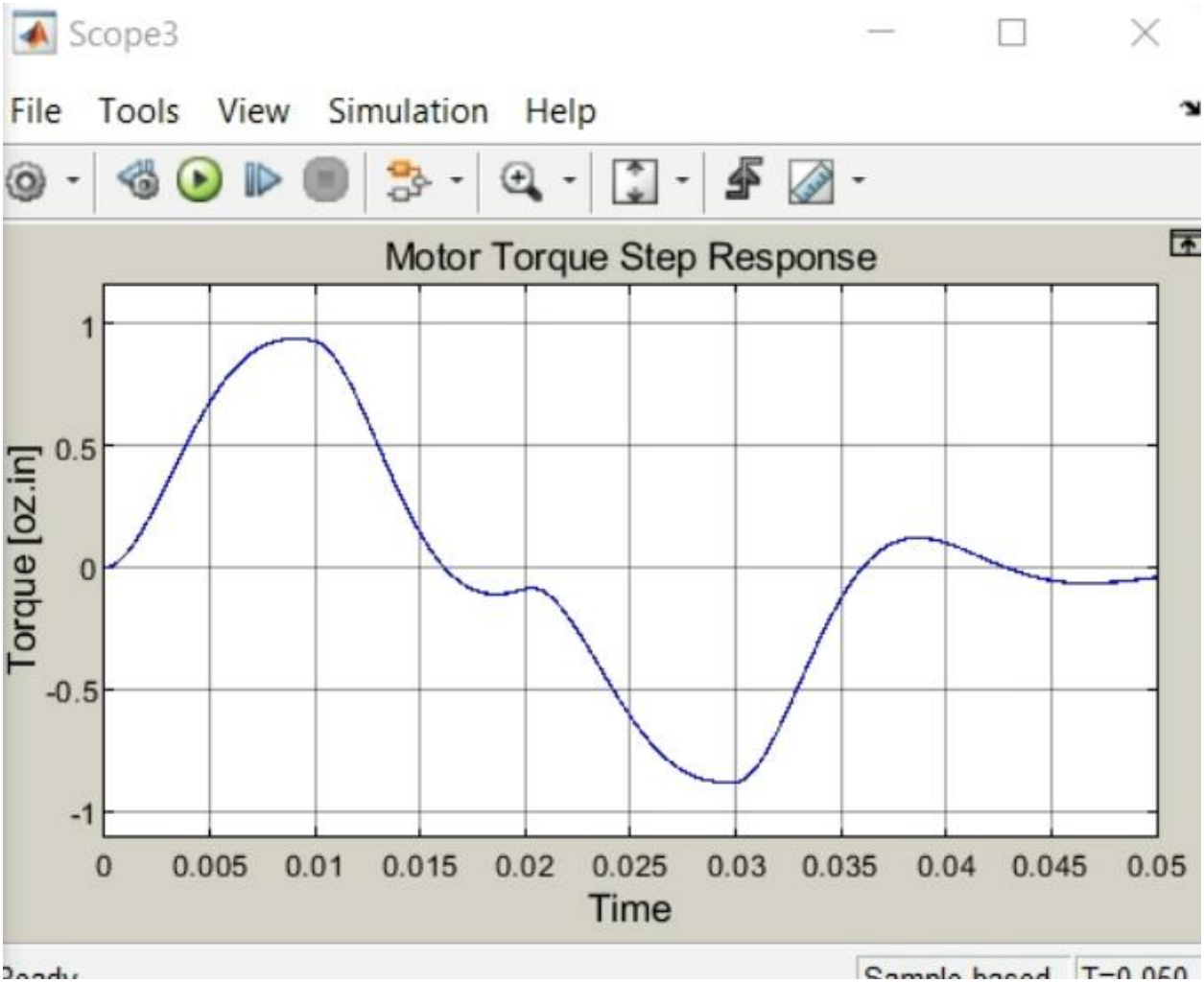

PID Tuner (EEL_5661_DC_Motor_sJ_PID_position_110421/PID Controller)
PID TUNER
Plant
Type PIDF
Form Parallel
Options
Domain
Time
Add Plot
PLANT
CONTROLLER
DESIGN
Slower
Response Time (seconds)
Faster
Aggressive
Transient Behavior
Robust
0.600
Reset Design
Show Parameters
Update Block
TUNING TOOLS
RESULTS
Plant List
Name
Class
Plant
ss
Preview
Step Plot: Reference tracking
Step Plot: Reference tracking
Tuned response
Block response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
0.005
0.01
0.015
0.02
0.025
0.03


PID TUNER
Plant
Type PIDF
Form Parallel
Options
Domain
Time
Add Plot
PLANT
CONTROLLER
DESIGN
Slower
Response Time (seconds)
Faster
Aggressive
Transient Behavior
Robust
0.600
Reset Design
Show Parameters
Update Block
TUNING TOOLS
RESULTS
Plant List
Name
Class
Plant
ss
Preview
Step Plot: Reference tracking
Step Plot: Reference tracking
Tuned response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
Closed-loop system with controller gains defined in the PID block is unstable and not displayed.
Controller Parameters: P = 4.101, I = 112.5, D = 0.0


Scope
File Tools View Simulation Help
Angular Velocity Response
Angular velocity [KRPM]
0.25
0.2
0.15
0.1
0.05
0
0 0.005 0.01 0.015 0.02 0.025 0.03 0.035 0.04 0.045 0.05
Time
Ready
Sample based
T=0.050


Scope2
File Tools View Simulation Help
Input command
r(t) input command
0.25
0.2
0.15
0.1
0.05
0
0 0.005 0.01 0.015 0.02 0.025 0.03 0.035 0.04 0.045 0.05
Time
Ready
Sample based
T=0.050


Scope5
File Tools View Simulation Help
Controller action
u(t) PID output
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0 0.005 0.01 0.015 0.02 0.025 0.03 0.035 0.04 0.045 0.05
Time
Sample based
T=0.050



Submission Deadline: Friday 11/18/2021 by 11:59 PM.