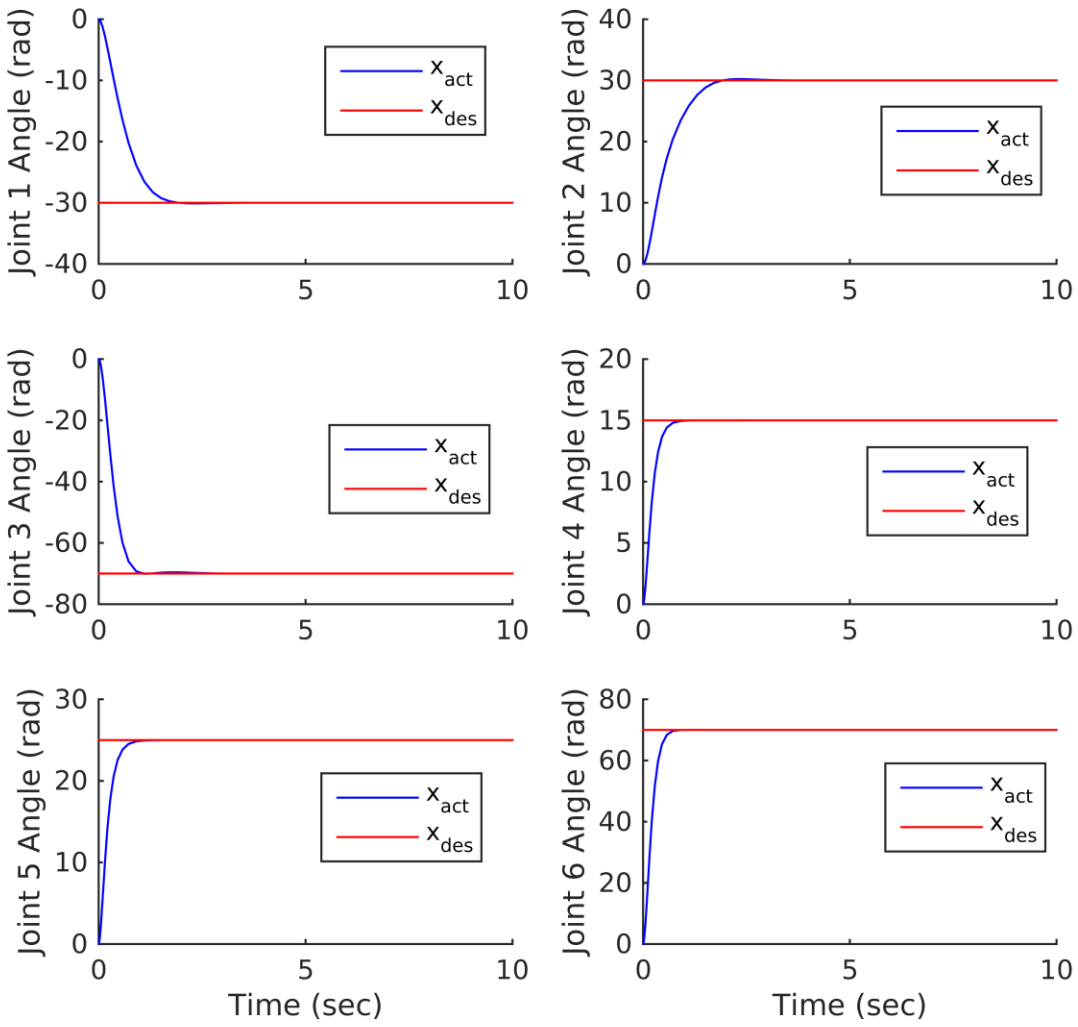
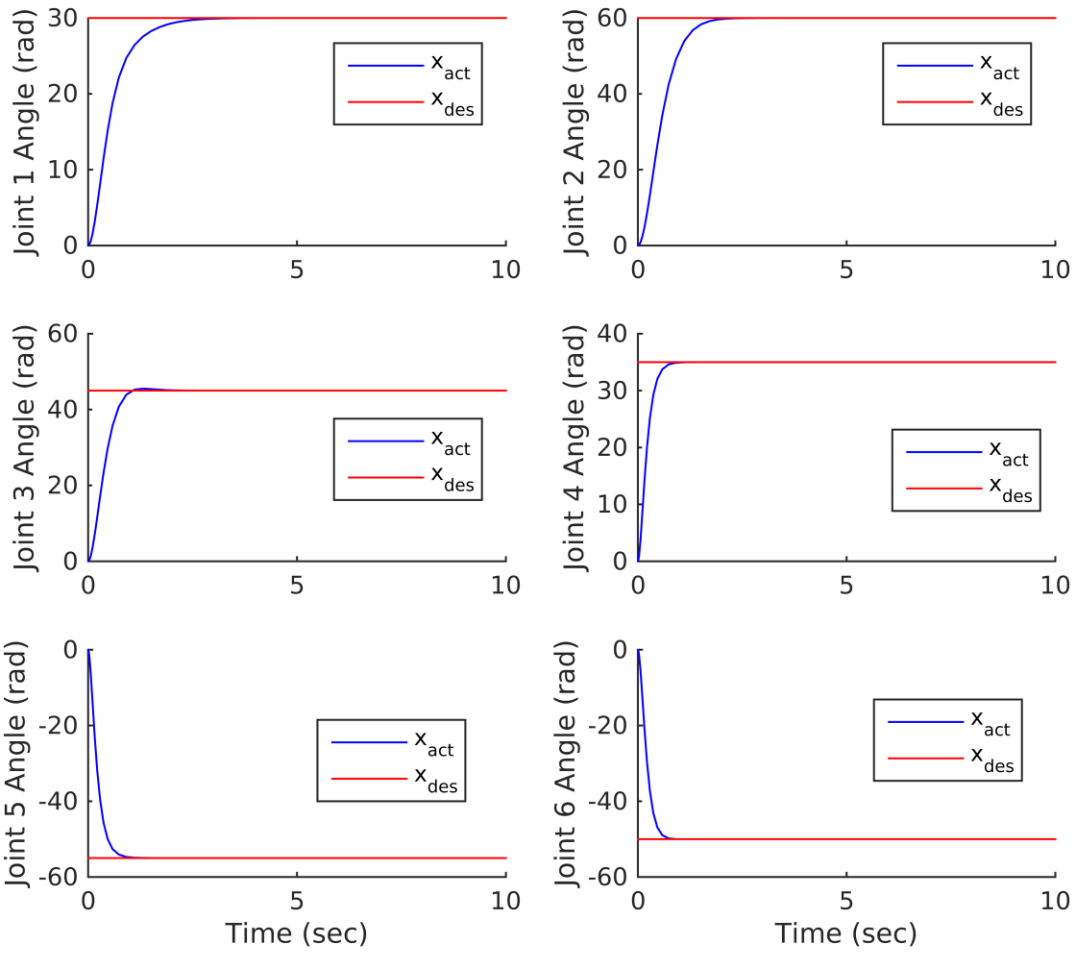
Dustan Kraus
Advanced Mechanisms Robotics

Homework 7

Problem 1

I designed a PD controller with gravity compensation. Below are the plots of the performance for two different configurations.



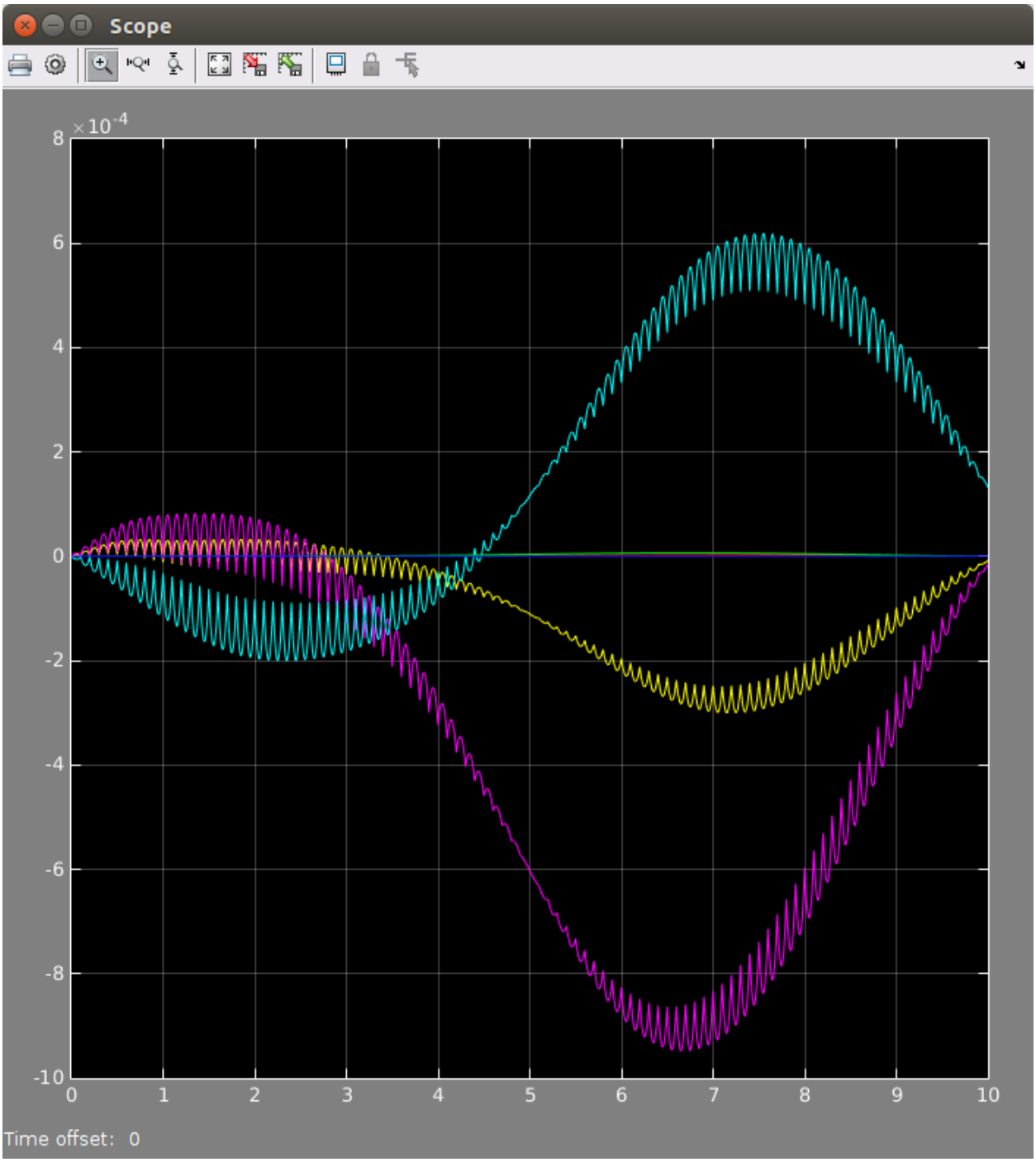


The Kp and Kd gain matrices I used are below:

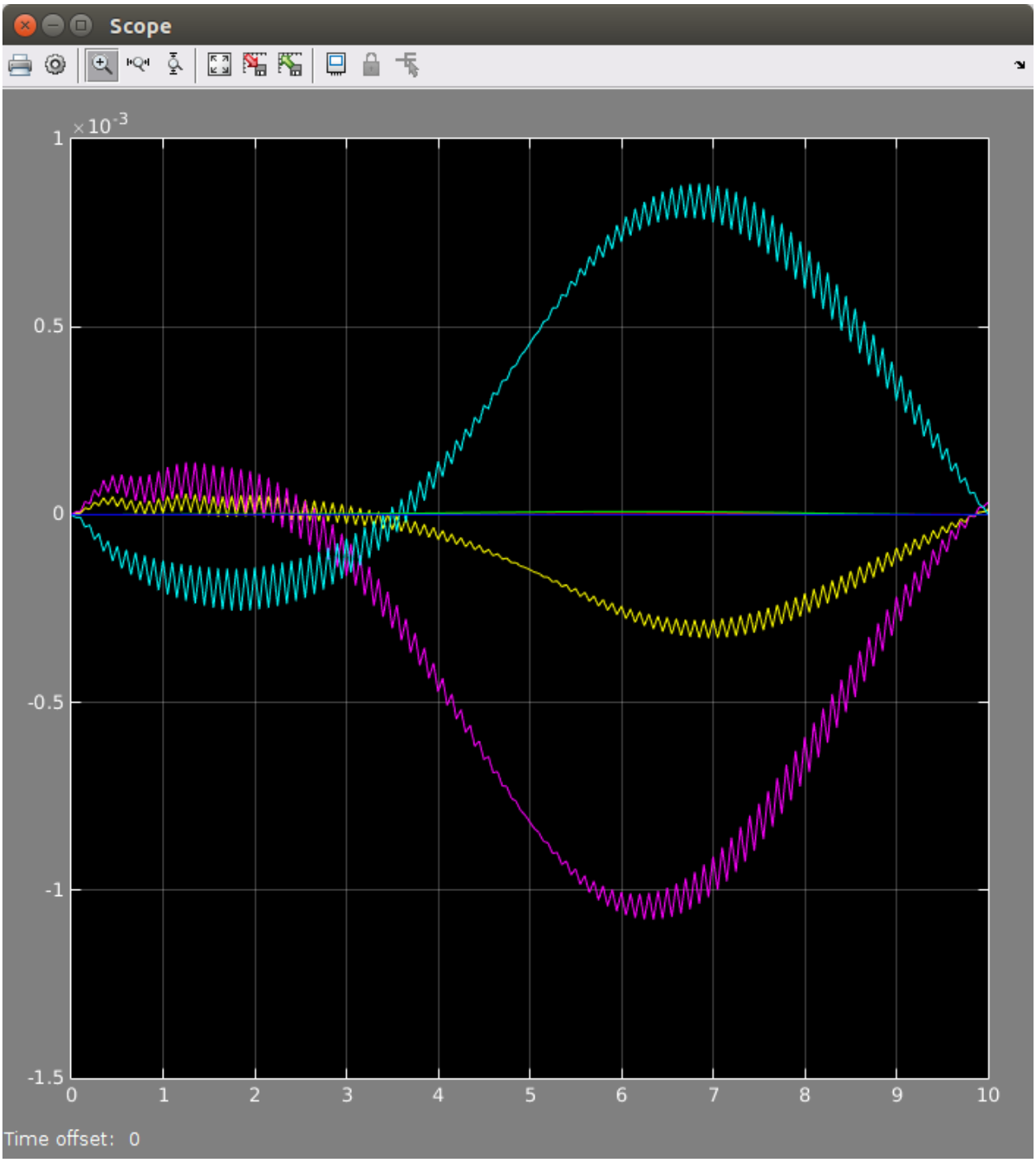
```
P.kp = diag([40 40 20 15 15 15])*eye(6);
P.kd = diag([18 14 3 3 3 3])*eye(6);
```

Problem 2

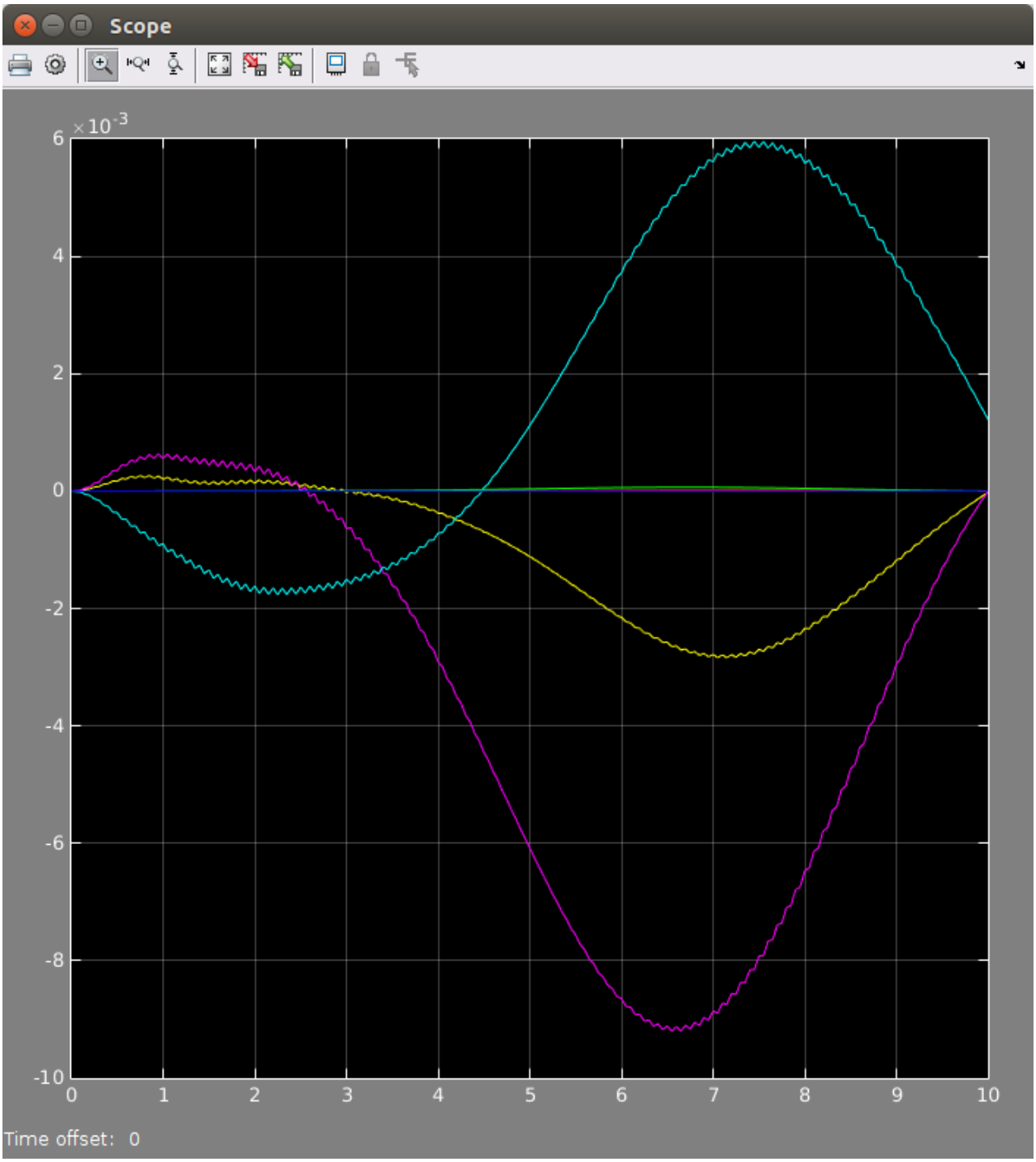
Computed Torque:



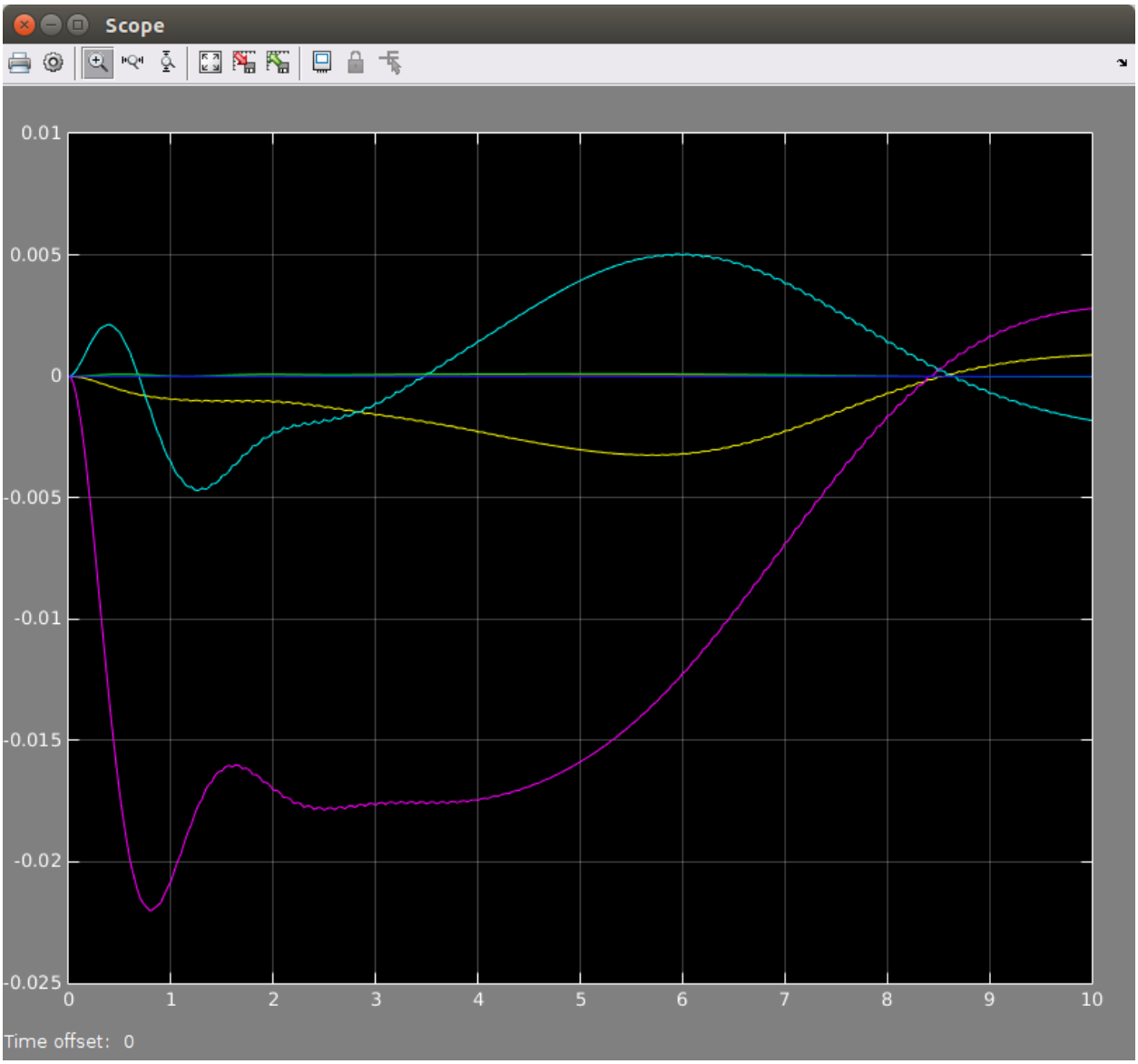
Computed Torque No Change



Computed Torque with Higher kp

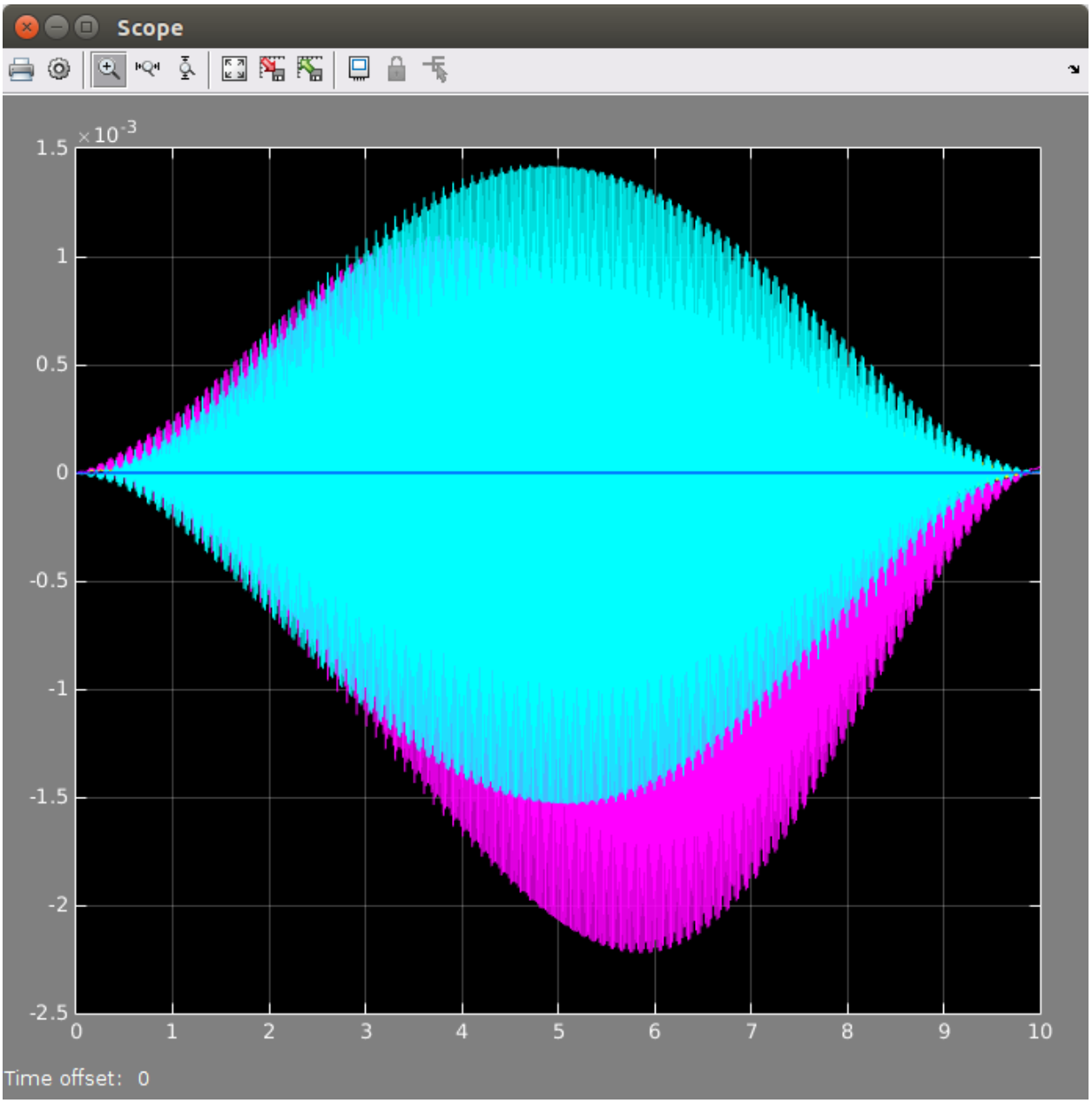


Computed Torque with a slower sample rate

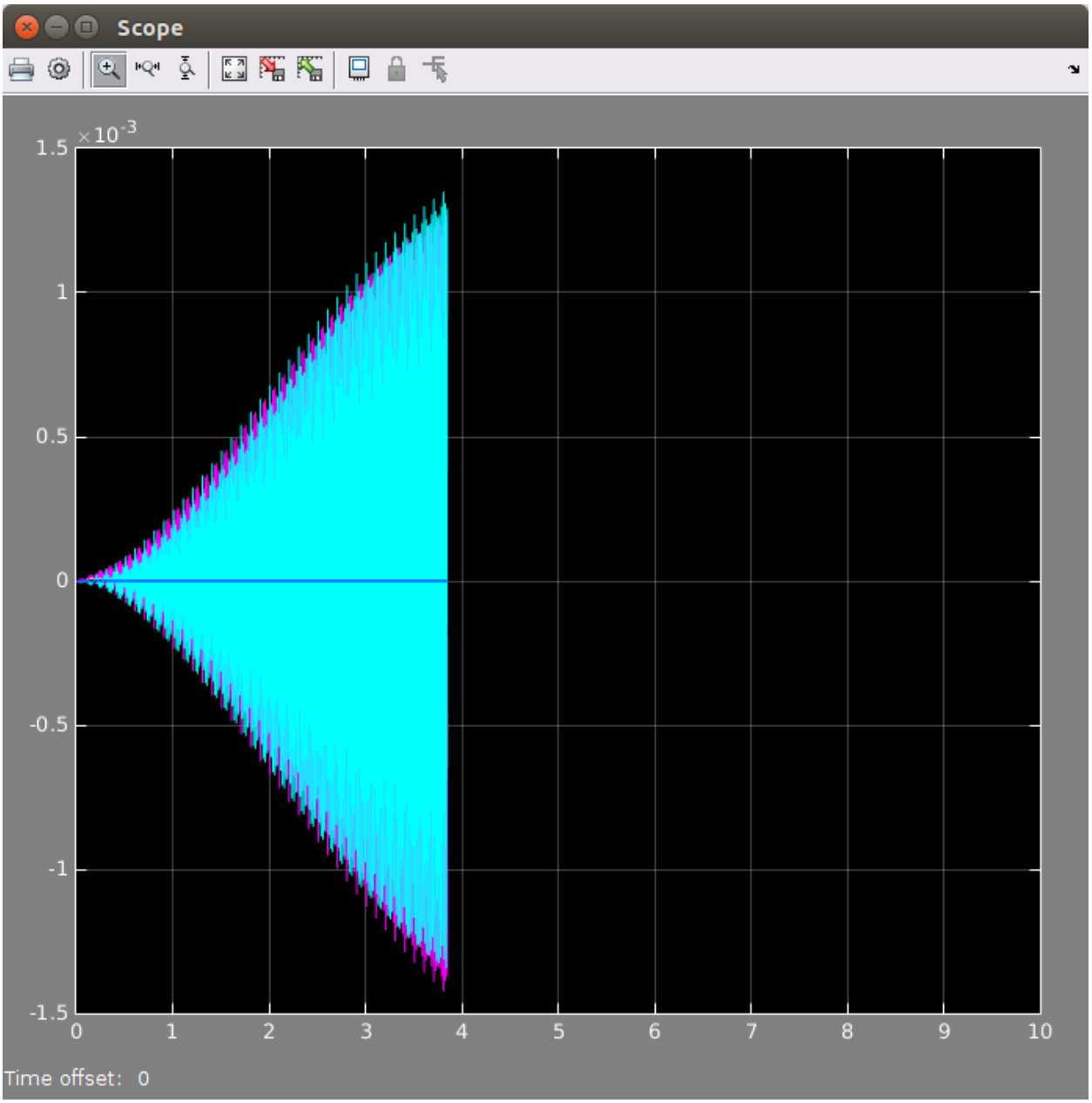


Computed Torque with Perturbation added

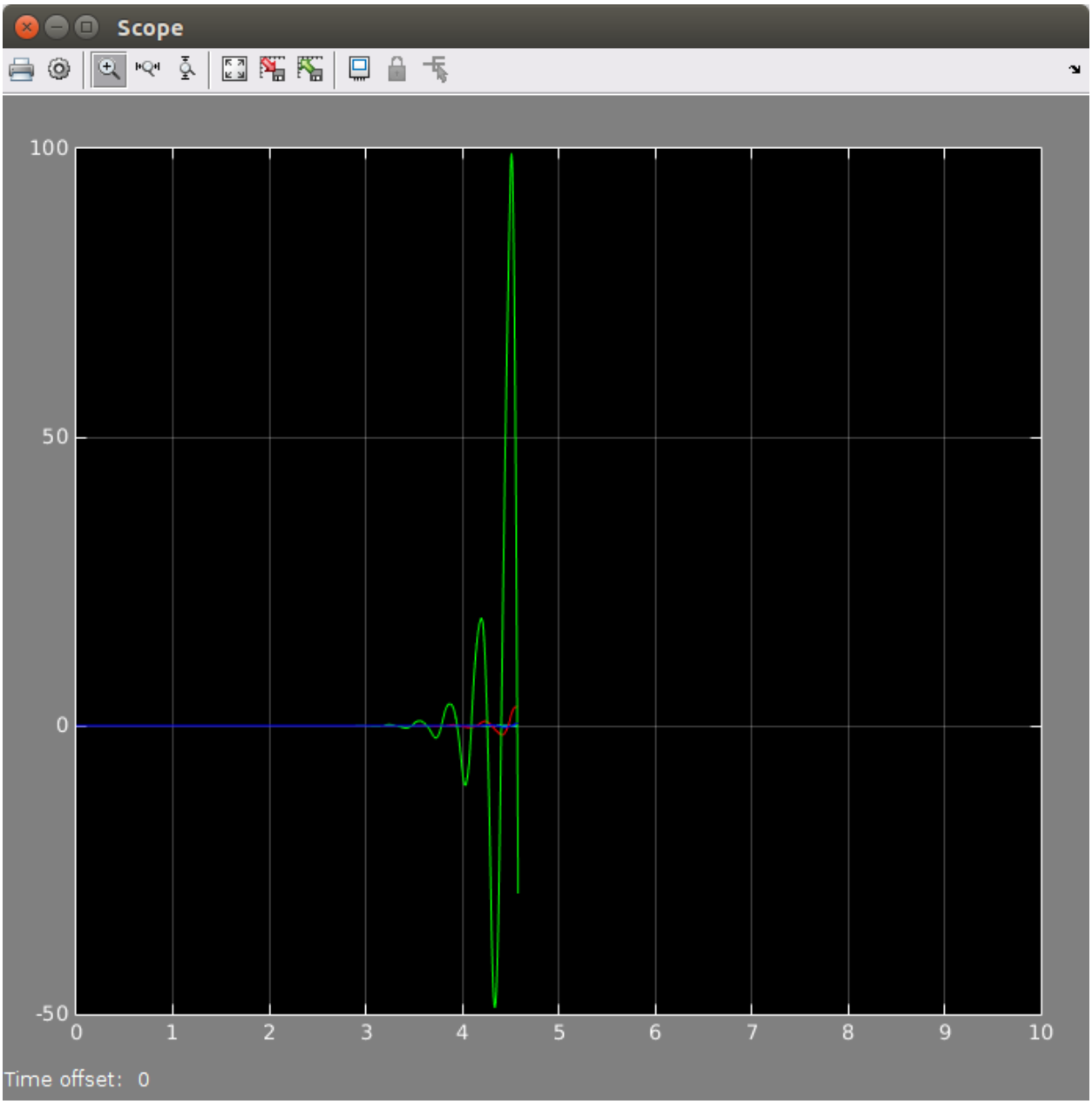
Feed Forward Torque:



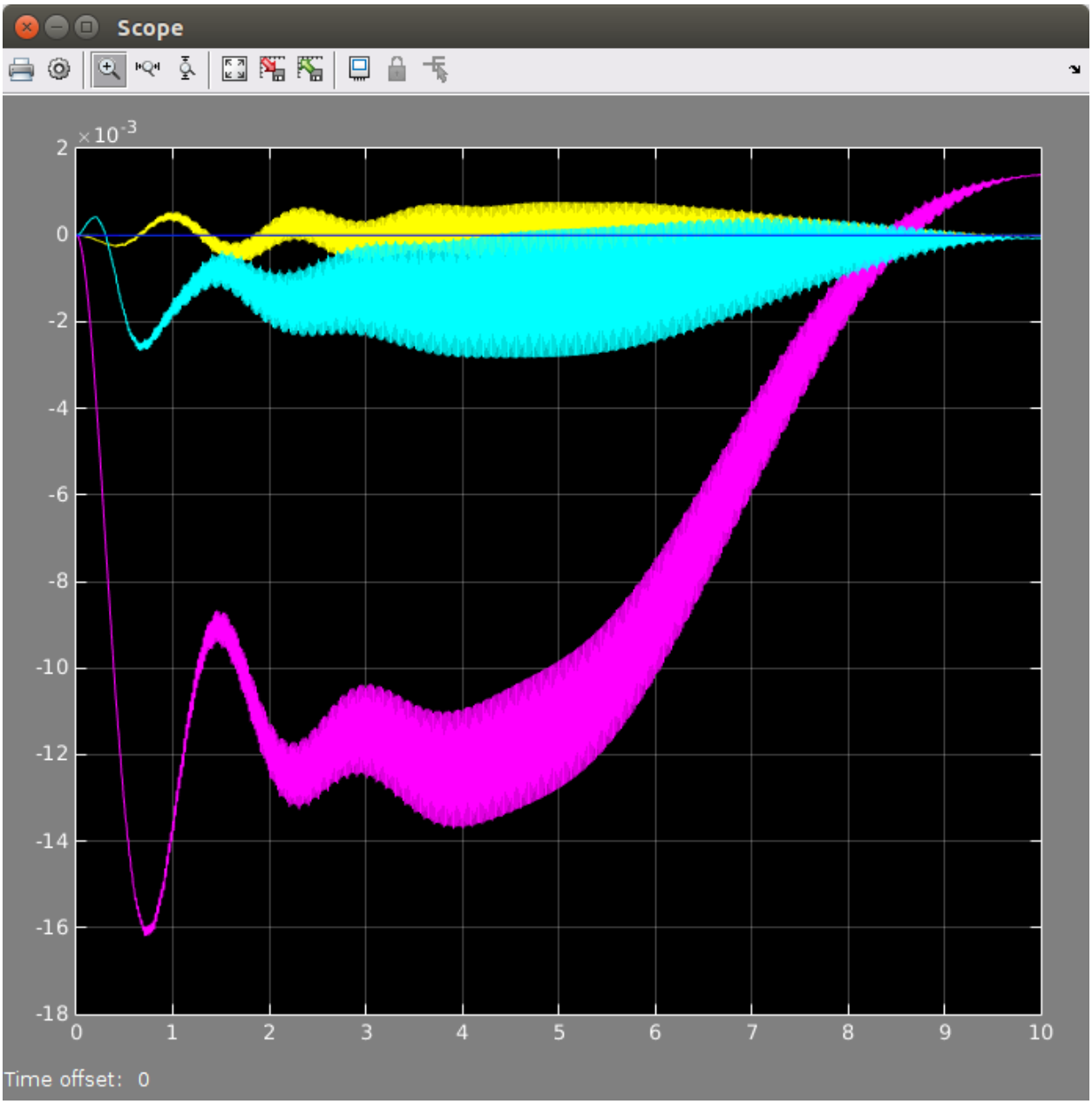
Feed Forward Torque with No change



Feed forward torque with bigger gains



Feed Forward Torque with Slower Sampling (clearly unstable)



Feed Forward Torque with a perturbation

Problem 3:

For this problem, I plotted the points and the projected points in pixel space to verify that the transformation matrices I found were accurate.

(a): T1_est =

0.9801	-0.0000	0.1987	-0.0000
-0.0198	0.9950	0.0978	-0.0000
-0.1977	-0.0998	0.9752	0.5000
0	0	0	1.0000

(b)
T2_est =

```
   1.0000   -0.0000   -0.0000    0.0000
   0.0000    1.0000   -0.0000    0.1000
   0.0000    0.0000    1.0000   -0.0000
        0         0         0    1.0000
```

(c)
T3_est =

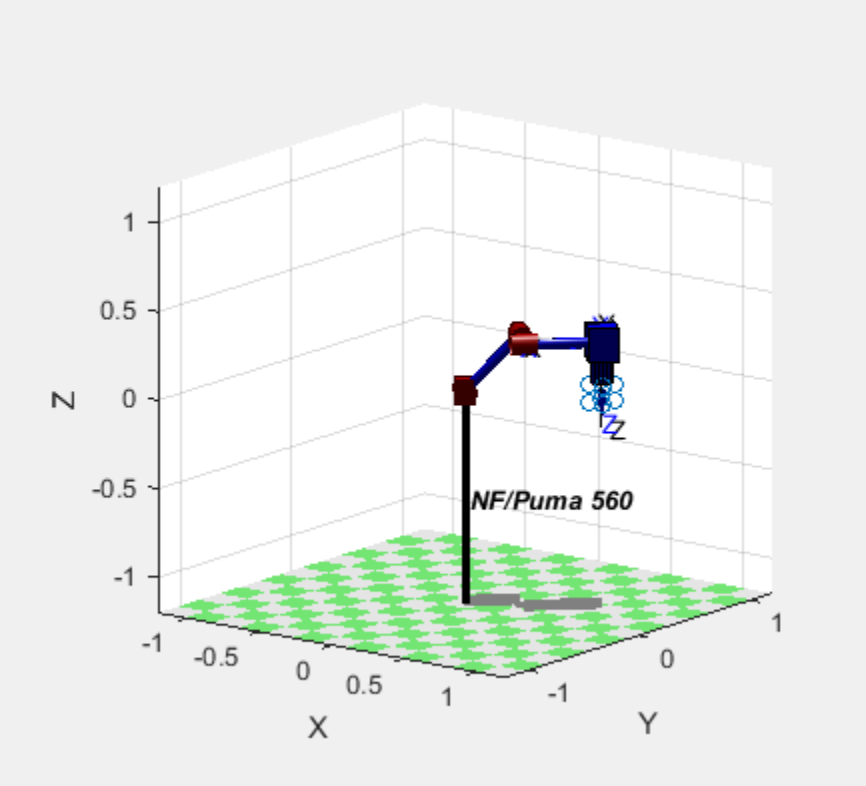
```
   0.9801   -0.1987   -0.0000    0.1000
   0.1977    0.9752    0.0998    0.0000
  -0.0198   -0.0978    0.9950    0.0000
        0         0         0    1.0000
```

Problem 4:

```
clear all
clc

% Make sure that you source the two startup files for the robotics toolbox
% and for the machine vision toolbox

% run ~/Desktop/vision-3.4/startup_rvc.m
% run ~/Desktop/rvctools/startup_rvc.m

%define the robotics toolbox Puma 560 arm
mdl_puma560;

%set the Coulomb friction terms to zero to help with numerical simulation
p560 = p560.nofriction;

%define desired robot pose
q_des = pi/180*[45,45,-135,0,-90,0];
Tcdes_0 = p560.fkine(q_des); %camera is at end effector
Tp_0 = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];

% Define object position in base frame
obj_pos = Tcdes_0(1:3,4);
obj_pos(3) = 0;
Tp_0(1:3,4) = obj_pos;
obj =    [...
    -0.1000   -0.1000    0.1000    0.1000   -0.1000   -0.1000    0.1000
0.1000;...
    -0.1000    0.1000    0.1000   -0.1000   -0.1000    0.1000    0.1000   -
0.1000;...
    -0.1000   -0.1000   -0.1000   -0.1000    0.1000    0.1000    0.1000
0.1000];
obj = obj*0.5;
obj(1,:) = obj(1,:) + obj_pos(1);
obj(2,:) = obj(2,:) + obj_pos(2);
obj(3,:) = obj(3,:) + obj_pos(3);
```

```
scatter3(obj(1,:),obj(2,:), obj(3,:))
p560.plot(q_des)

cam = CentralCamera('default');
cam = cam.move(Tcdes_0);
pixels = cam.project(obj);
%cam.plot(obj);
%cam.plot(pixels);

Tp_c = cam.estpose(obj, pixels)

Tcdes_p = inv(Tp_0)*Tcdes_0;
inv(Tcdes_p)
cam.plot_camera
%P = [0.3, 0.4, 3.0]';
%cam.project(P);

%cam.project(P, 'Tcam', transl(-0.5, 0, 0));
```

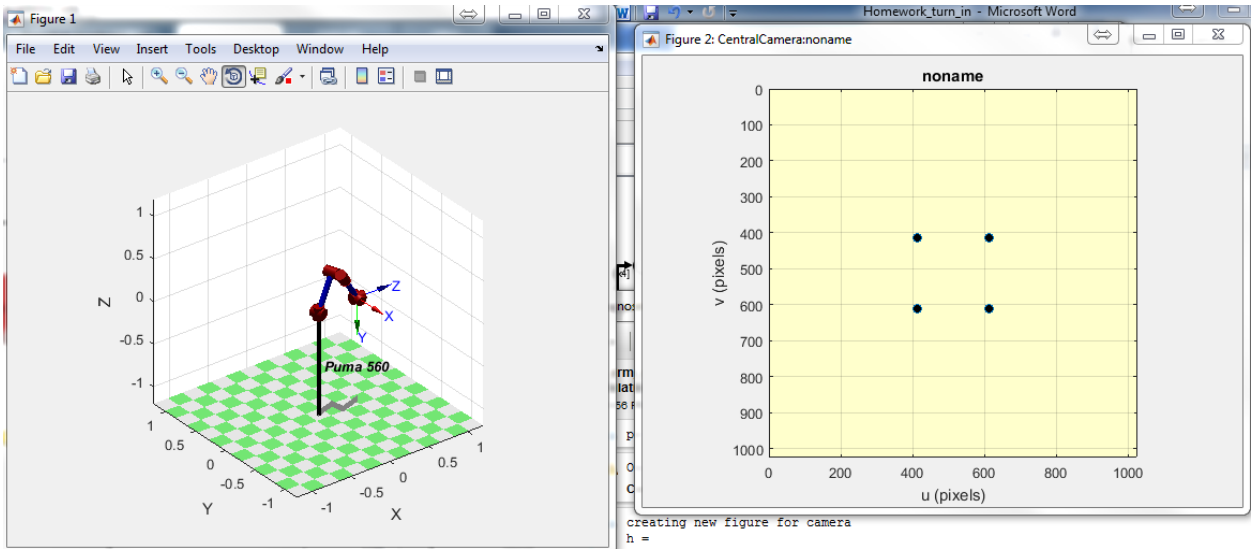
I never could quite get problem 4 figured out. I spent a good amount of time trying to figure out why the cam.estpose() function wasn't working for me, but couldn't quite get it. So I never even got to the servoing part because I spent a good amount of time trying to figure this out.



This shows the camera and shape I generated.

Problem 5:

Because I didn't finish problem 4, I couldn't compare my results with problem 4. But I did open and run sl_arm_ibvs. It was very cool to see servoing work properly.



Problem 6:

Again, because I didn't finish problem 4, I can't use my implementation from it. But If I had finished it, I would use the Simulink roblocks to create the robot model.