Objectives

The objective of this lab is to design a full-state feedback controller using the Linear Quadratic Regulator (LQR) design technique and to understand the effect of varying the penalty matrices P and Q in the cost functional on the performance of the closed-loop system.

Theory

Pole placement for controller design relies on specification of the desired closed-loop poles of the system. This is usually difficult to specify, especially for systems with a large number of states. Furthermore, with pole placement design there is hard to take the "amount" of actuation (called actuation or control effort) that gets used during closed-loop operation into account.

Good regulation of the system can usually be achieved by using high amounts of actuation (for example in a P-controller, higher Kp, and thus greater actuation effort, gives faster rise time). But in reality, we are often limited by power and energy constraints. Ideally, we would like to achieve good system performance while at the same time minimizing the amount of actuation used in achieving the desired performance. One way of expressing this mathematically is through a cost functional of the form:

$$J = \int_0^\infty x^T Q x + u^T R u \, dt \tag{1}$$

where Q and R are weighting matrices (these are the design parameters).

The LQR design problem is to design a state-feedback controller K (i.e. for u = −Kx) such that the cost functional J is minimized1 . The cost functional (2) consists of two terms, the first of which you can think of as being the cost of regulating the state x (regulatory term) and the second being the cost of actuation u (actuation term). Both of these terms depend on a weighting matrix, Q and R, respectively. These matrices are the design parameters, assumed positive semidefinite. The regulatory term will "penalize" deviations from the desired state (here x = 0), while the actuation term will "penalize" you for any actuation effort u =/= 0.

For simplicity we assume in this lab that the matrices Q and R are diagonal: Q = diag(q1, . . . , qn) and R = diag(r1, . . . , rm). Thus, the objective J reduces to

$$J = \int_0^\infty \left(\sum_{i=1}^{n} q_i x_i^2 + \sum_{j=1}^{m} r_j u_j^2 \right) dt \tag{2}$$

The scalars q1, . . . , qn and r1, . . . , rm can be seen as relative weights between different performance terms in the objective J. For Q and R to be positive semidefinite, we need qi ≥ 0 and ri ≥ 0 for all i. The key 1 In fact, one can show that even when optimizing over a larger class of controllers, it turns out that the optimal controller is a linear time-invariant state-feedback controller of the form u = −Kx. 1 of 3 ME C134 / EE C128 Lab 6c UC Berkeley design problem of LQR is to translate performance specifications in terms of the rise time, overshoot, bandwidth, etc. into relative weights of the above form. There is no straightforward way of doing this and it is usually done through an iterative process either in simulations or on an experimental setup. Once the matrices Q and R are completely specified, the controller gain K is found by solving the so-called Algebraic Riccati Equation (ARE), which can be done numerically in MATLAB.

Lab

Implement the controllers you designed in the Pre-Lab on the hardware. Don't forget to include the observer, as you did in the Pre-Lab, and use the state estimate ˆx to control the system. Use a step input of the form r = $[0.3\ 0\ 0\ 0]^T$. Make sure to set the observer initial state to zero.

Consider the following controllers:

1. nominal weights
2. a higher relative weight q_1, other weights nominal
3. a higher relative weight q_3, other weights nominal
4. a higher relative weight of r, other weights nominal

In each case, observe the output response of the system, note the variation of the position of the cart and the pendulum with time and the control input. In addition, plot output y and control u and discuss the effect of the weights on the system behavior. Make sure that the differences are noticeable on your plots. For each case,

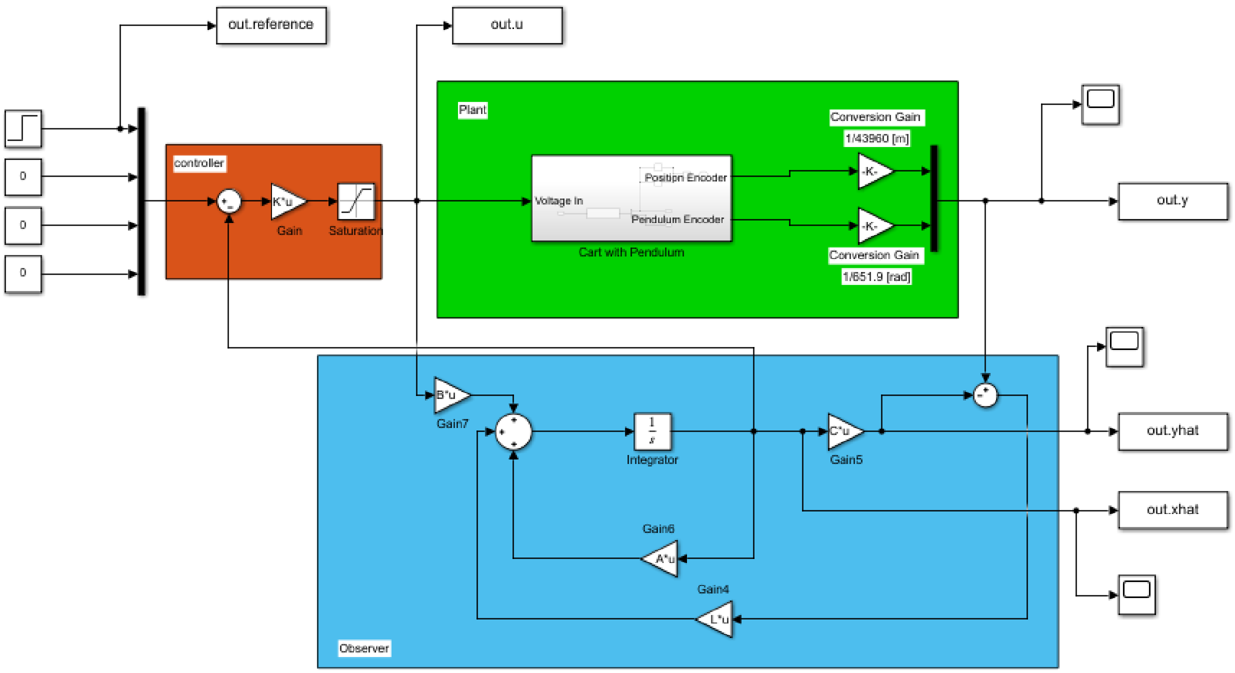
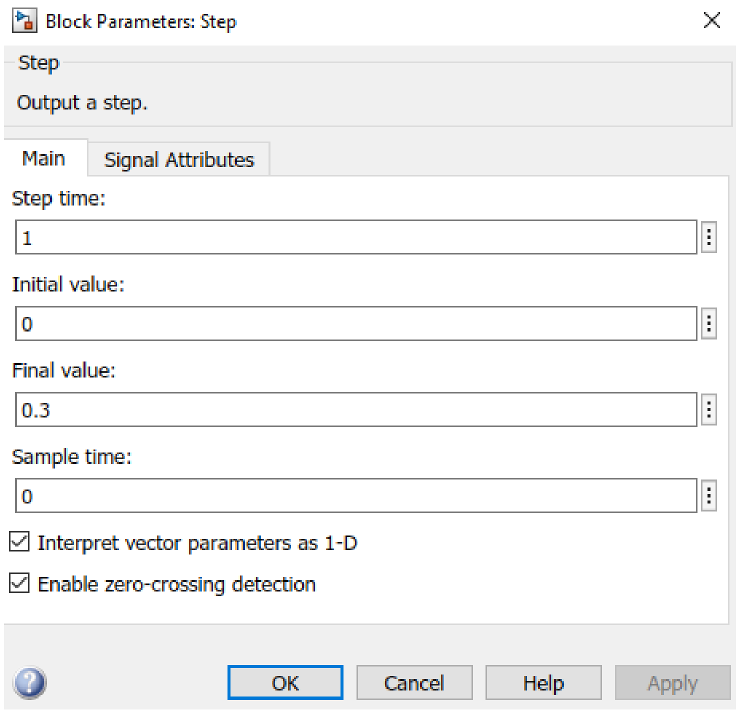


Figure 1: Simulink model used for the step input simulations.



Block Parameters: Step

Step

Output a step.

Main | Signal Attributes

Step time:
1

Initial value:
0

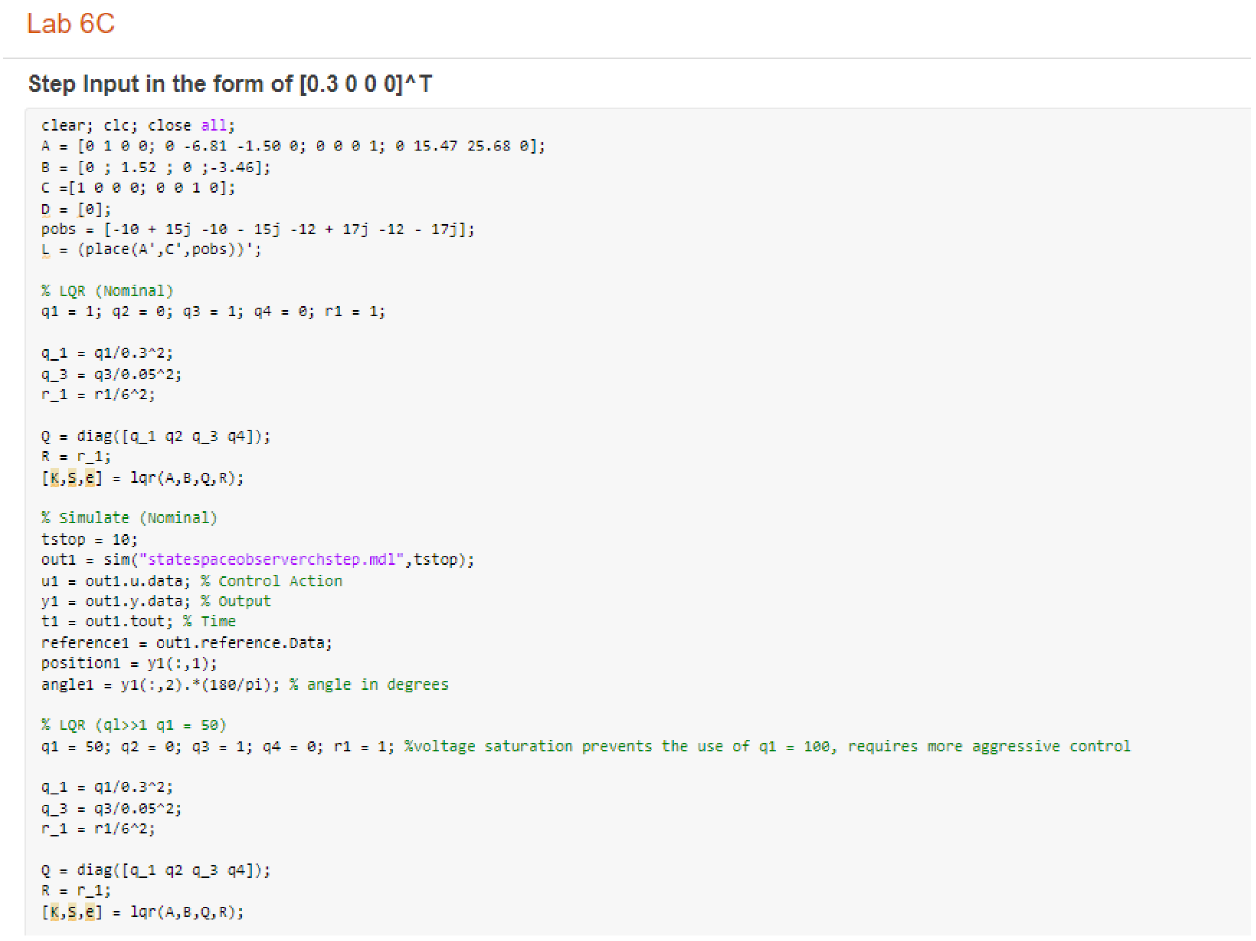
Final value:
0.3

Sample time:
0

☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

Figure 2: Parameters for step function used in the step input simulations.

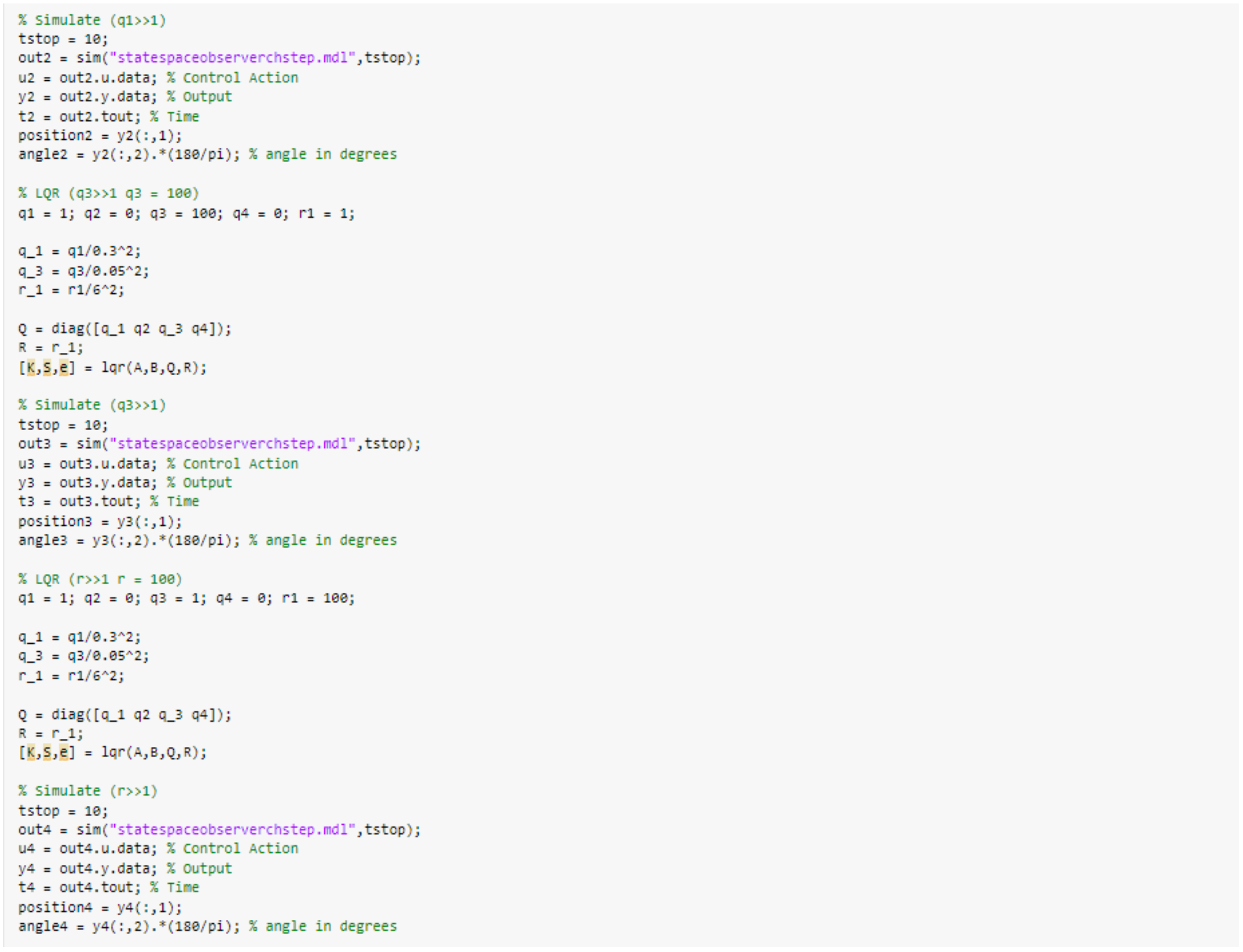


Lab 6C

Step Input in the form of [0.3 0 0 0]^T

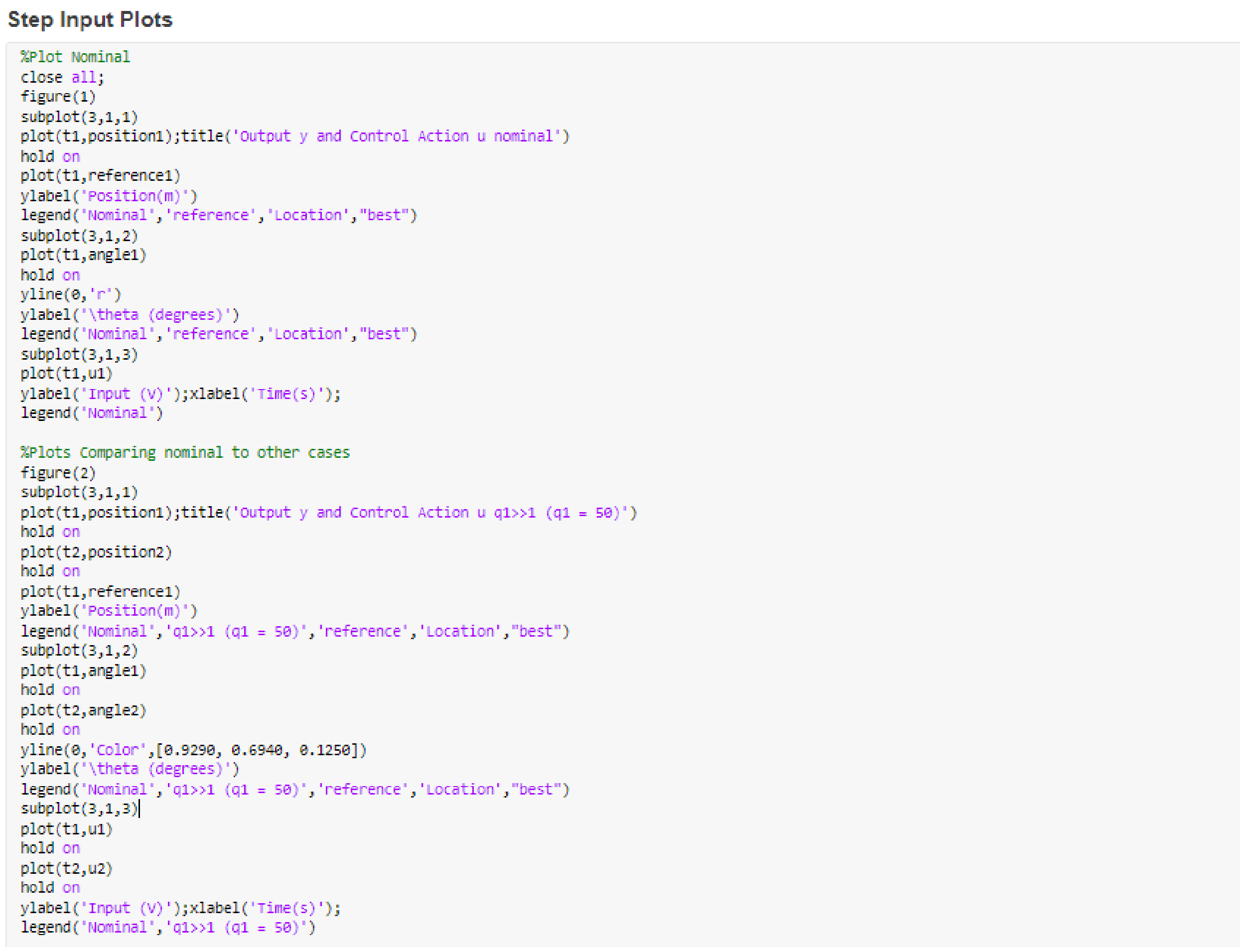
```
clear; clc; close all;
A = [0 1 0 0; 0 -6.81 -1.50 0; 0 0 0 1; 0 15.47 25.68 0];
B = [0 ; 1.52 ; 0 ;-3.46];
C =[1 0 0 0; 0 0 1 0];
D = [0];
pobs = [-10 + 15j -10 - 15j -12 + 17j -12 - 17j];
L = (place(A',C',pobs))';

% LQR (Nominal)
q1 = 1; q2 = 0; q3 = 1; q4 = 0; r1 = 1;

q_1 = q1/0.3^2;
q_3 = q3/0.05^2;
r_1 = r1/6^2;

Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (Nominal)
tstop = 10;
out1 = sim("statespaceobserverchstep.mdl",tstop);
u1 = out1.u.data; % Control Action
y1 = out1.y.data; % Output
t1 = out1.tout; % Time
reference1 = out1.reference.Data;
position1 = y1(:,1);
angle1 = y1(:,2).*(180/pi); % angle in degrees

% LQR (ql>>1 q1 = 50)
q1 = 50; q2 = 0; q3 = 1; q4 = 0; r1 = 1; %voltage saturation prevents the use of q1 = 100, requires more aggressive control

q_1 = q1/0.3^2;
q_3 = q3/0.05^2;
r_1 = r1/6^2;

Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);
```

Figure 3: MATLAB script used for simulating step function cases (part 1).



```
% Simulate (q1>>1)
tstop = 10;
out2 = sim("statespaceobserverchstep.mdl",tstop);
u2 = out2.u.data; % Control Action
y2 = out2.y.data; % Output
t2 = out2.tout; % Time
position2 = y2(:,1);
angle2 = y2(:,2).*(180/pi); % angle in degrees

% LQR (q3>>1 q3 = 100)
q1 = 1; q2 = 0; q3 = 100; q4 = 0; r1 = 1;

q_1 = q1/0.3^2;
q_3 = q3/0.05^2;
r_1 = r1/6^2;

Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (q3>>1)
tstop = 10;
out3 = sim("statespaceobserverchstep.mdl",tstop);
u3 = out3.u.data; % Control Action
y3 = out3.y.data; % Output
t3 = out3.tout; % Time
position3 = y3(:,1);
angle3 = y3(:,2).*(180/pi); % angle in degrees

% LQR (r>>1 r = 100)
q1 = 1; q2 = 0; q3 = 1; q4 = 0; r1 = 100;

q_1 = q1/0.3^2;
q_3 = q3/0.05^2;
r_1 = r1/6^2;

Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (r>>1)
tstop = 10;
out4 = sim("statespaceobserverchstep.mdl",tstop);
u4 = out4.u.data; % Control Action
y4 = out4.y.data; % Output
t4 = out4.tout; % Time
position4 = y4(:,1);
angle4 = y4(:,2).*(180/pi); % angle in degrees
```

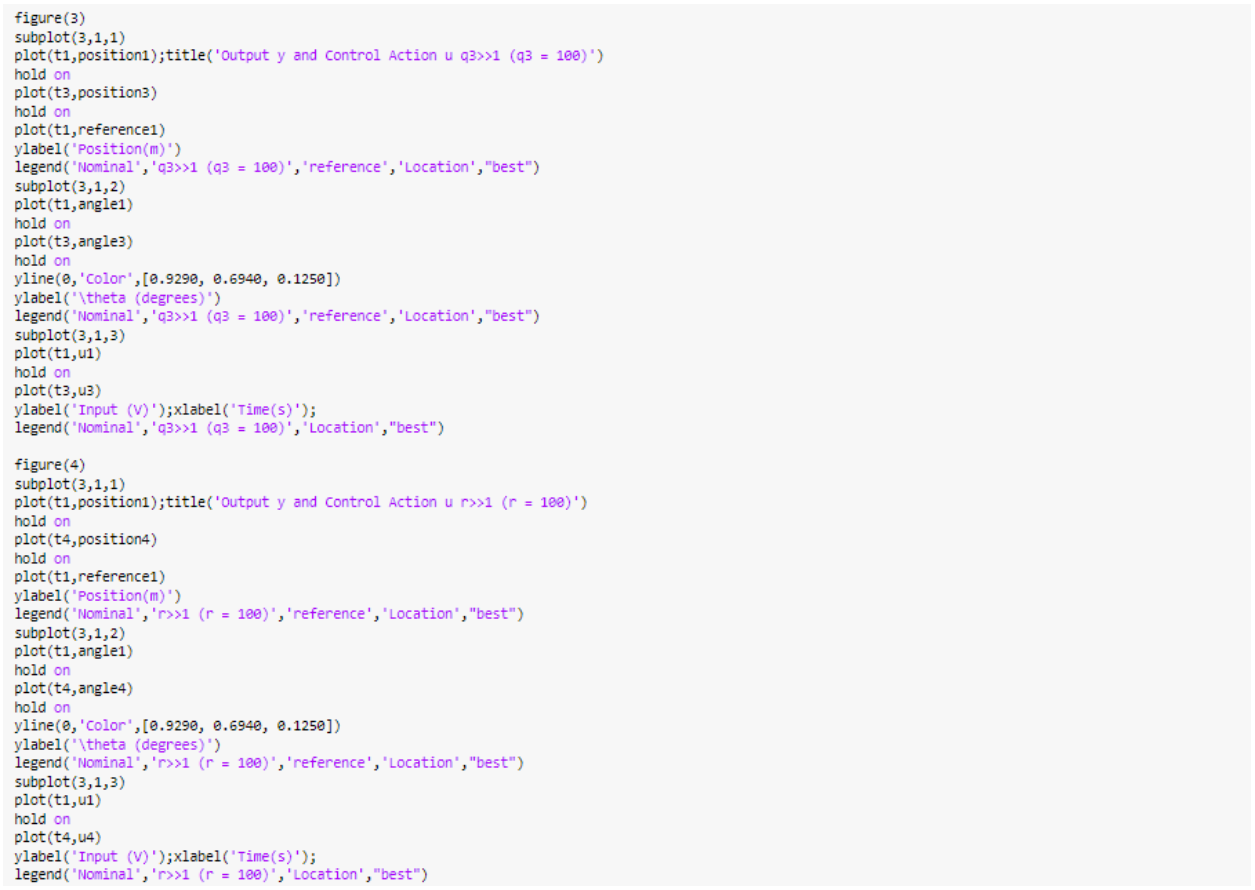
Figure 4: MATLAB script used for simulating step function cases (part 2).



Step Input Plots

```matlab
%Plot Nominal
close all;
figure(1)
subplot(3,1,1)
plot(t1,position1);title('Output y and Control Action u nominal')
hold on
plot(t1,reference1)
ylabel('Position(m)')
legend('Nominal','reference','Location',"best")
subplot(3,1,2)
plot(t1,angle1)
hold on
yline(0,'r')
ylabel('\theta (degrees)')
legend('Nominal','reference','Location',"best")
subplot(3,1,3)
plot(t1,u1)
ylabel('Input (V)');xlabel('Time(s)');
legend('Nominal')

%Plots Comparing nominal to other cases
figure(2)
subplot(3,1,1)
plot(t1,position1);title('Output y and Control Action u q1>>1 (q1 = 50)')
hold on
plot(t2,position2)
hold on
plot(t1,reference1)
ylabel('Position(m)')
legend('Nominal','q1>>1 (q1 = 50)','reference','Location',"best")
subplot(3,1,2)
plot(t1,angle1)
hold on
plot(t2,angle2)
hold on
yline(0,'Color',[0.9290, 0.6940, 0.1250])
ylabel('\theta (degrees)')
legend('Nominal','q1>>1 (q1 = 50)','reference','Location',"best")
subplot(3,1,3)
plot(t1,u1)
hold on
plot(t2,u2)
hold on
ylabel('Input (V)');xlabel('Time(s)');
legend('Nominal','q1>>1 (q1 = 50)')
```

Figure 5: MATLAB script used for plotting step input simulation data (part 1).



```matlab
figure(3)
subplot(3,1,1)
plot(t1,position1);title('Output y and Control Action u q3>>1 (q3 = 100)')
hold on
plot(t3,position3)
hold on
plot(t1,reference1)
ylabel('Position(m)')
legend('Nominal','q3>>1 (q3 = 100)','reference','Location',"best")
subplot(3,1,2)
plot(t1,angle1)
hold on
plot(t3,angle3)
hold on
yline(0,'Color',[0.9290, 0.6940, 0.1250])
ylabel('\theta (degrees)')
legend('Nominal','q3>>1 (q3 = 100)','reference','Location',"best")
subplot(3,1,3)
plot(t1,u1)
hold on
plot(t3,u3)
ylabel('Input (V)');xlabel('Time(s)');
legend('Nominal','q3>>1 (q3 = 100)','Location',"best")

figure(4)
subplot(3,1,1)
plot(t1,position1);title('Output y and Control Action u r>>1 (r = 100)')
hold on
plot(t4,position4)
hold on
plot(t1,reference1)
ylabel('Position(m)')
legend('Nominal','r>>1 (r = 100)','reference','Location',"best")
subplot(3,1,2)
plot(t1,angle1)
hold on
plot(t4,angle4)
hold on
yline(0,'Color',[0.9290, 0.6940, 0.1250])
ylabel('\theta (degrees)')
legend('Nominal','r>>1 (r = 100)','reference','Location',"best")
subplot(3,1,3)
plot(t1,u1)
hold on
plot(t4,u4)
ylabel('Input (V)');xlabel('Time(s)');
legend('Nominal','r>>1 (r = 100)','Location',"best")
```

Figure 6: MATLAB script used for plotting step input simulation data (part 2).

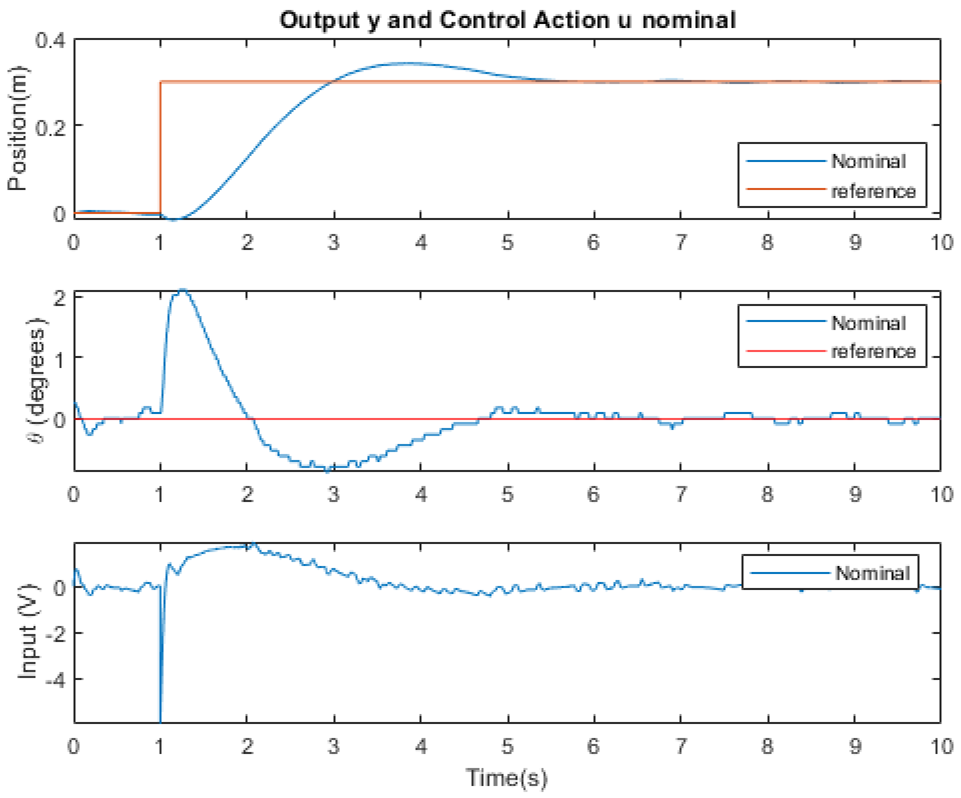


Figure 7: Outputs for cart position, pendulum position, and control input over time for the nominal weights case.

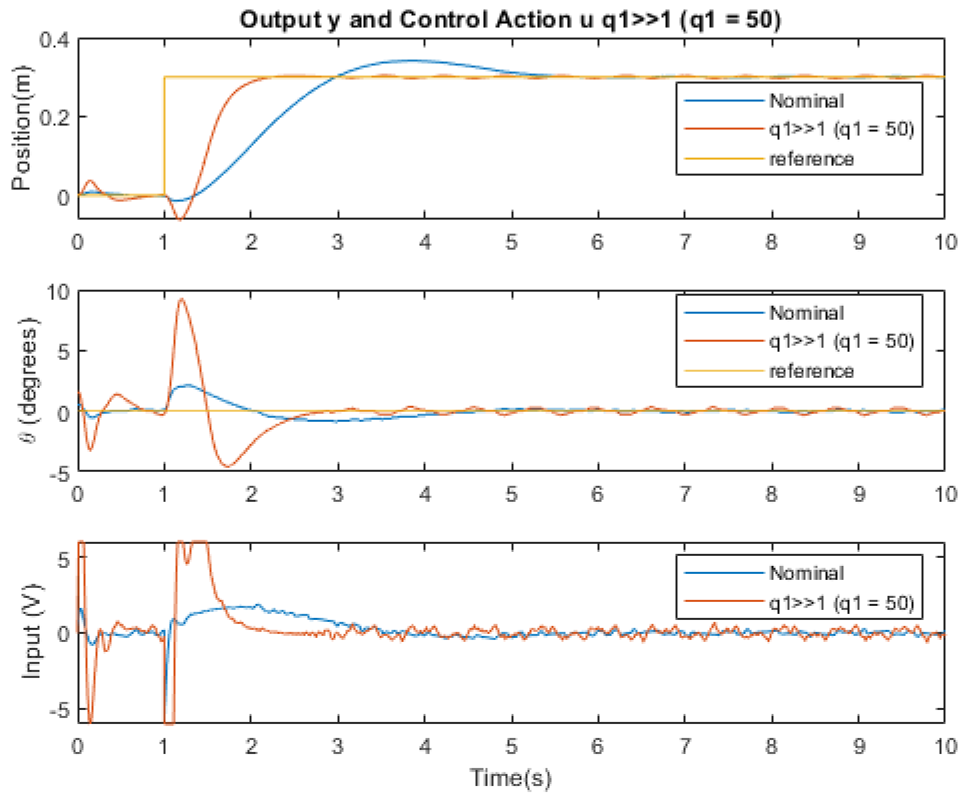


Figure 8: Outputs for cart position, pendulum position, and control input over time for the boosted q1 weight case (other weights nominal).

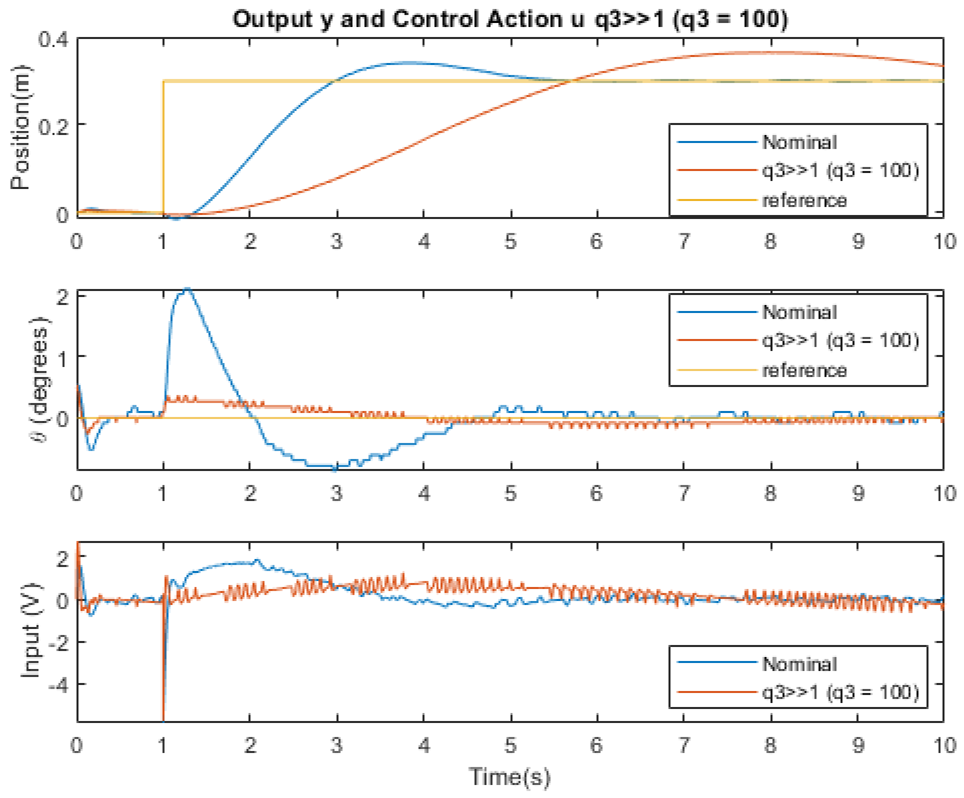


Figure 9: Outputs for cart position, pendulum position, and control input over time for the boosted q3 weight case (other weights nominal).

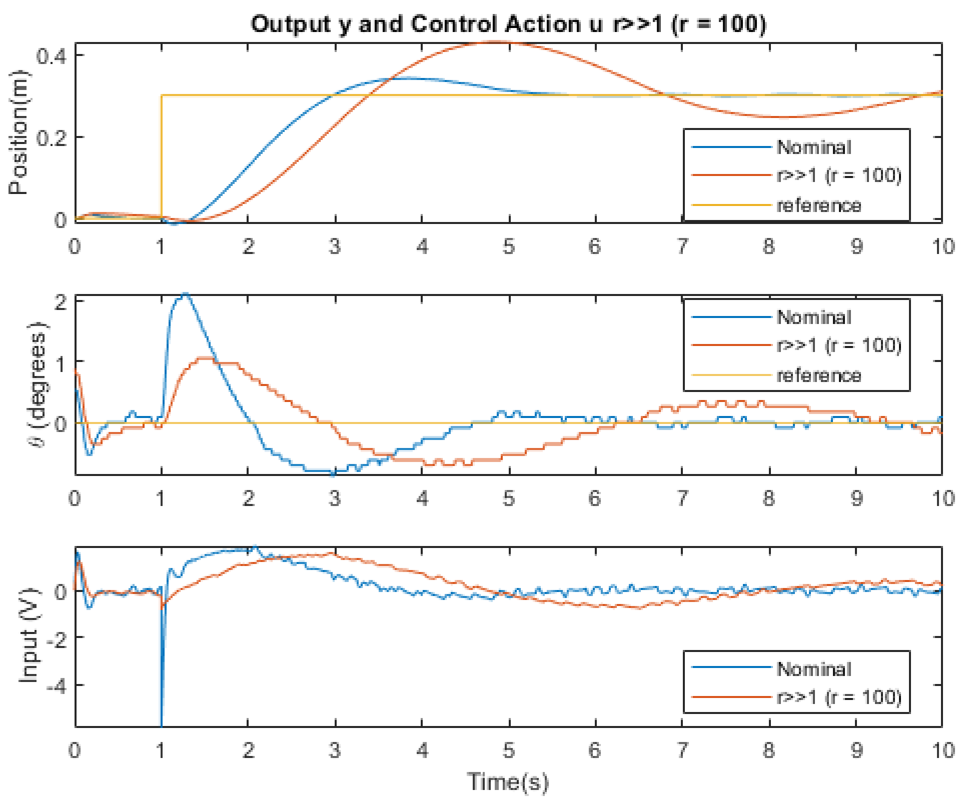


Figure 10: Outputs for cart position, pendulum position, and control input over time for the boosted r weight case (other weights nominal).

- **discuss how the closed-loop system behavior changes w.r.t. the nominal case**

  Case 1: (q1>>1 or q1 = 50)

  Something to note for this case is that we were unable to set the q_1 to 100 due to the saturation of the input and it seems that the cart would go off rail when q_1 is set to 100. This means for this step input in order to use higher weights for q1 we need to incorporate more aggressive control. Observing the set of plots from Figure 8, we can see that the q_1=50 case tracks the step input faster than the nominal case. In other words, the q_1=50 case has a smaller phase lag than the nominal case. One can observe that high peaks in the position, angle and input plots, where there is initially a high peak for the input and then later on when the step occurs there are high peaks for both angle and input.

  Case 2: (q3>>1 or q3 = 100)

  In adjusting the weight so that q_3 >>1 or more specifically we set the weight of q_3 = 100, we can observe through the angle plot in Figure 9, that there is not as much of an extreme peak as the nominal case when the step occurs. In other words, increasing the value of q_3 would increase the damping of the system. We observe on the same set of plots that for position the q3 = 100 case takes longer to converge to (or track) the step input than the nominal case. The input plot does not seem to be affected as much, but appears to look “noisier”.

Case 3: (r >>1 or r = 100)
In adjusting the weight so that r >>1 or more specifically we set the weight of r = 100, we can observe through the input plot in Figure 10, that there is not as much of an extreme drop as the nominal case when the step occurs. We observe on the same set of plots that for position and angle the r = 100 case takes longer to converge to (or track) the step input than the nominal case. In other words, the transient responses for position and angle take longer to die out. Also, the phase lags of the responses seem to be larger with a larger value of r.

- **discuss if and how your results differ from the ones obtained in the simulations in the Pre-Lab**
  Case 1: (q3>>1 or q3 = 100)
  Comparing the plots in Figure 9 with the plots in the Pre-Lab when we set the weight of $q_3 = 100$, we can observe that the plots in Figure 9 show that there are more oscillations and noises as well as taking longer to reach stray states, and this is expected as the hardware is nonlinear and the simulation in the Pre-lab is linearized. There is large upward overshooting for the angle in Figure 9 while there is large undershoot in the angle plot from the Pre-Lab.

  Case 2: (r >>1 or r = 100)
  Comparing the plots in Figure 10 with the plots in the Pre-Lab when we set the weight of $r = 100$, we can also observe that the plots in Figure 10 show that there are more oscillations and noises as well as taking longer to reach stray states. There is large upward overshooting for the angle in Figure 10 while there is large undershoot in the angle plot from the Pre-lab. Nevertheless, the rise time when $r = 100$ for the position is really close to the nominal case in Figure 10 while the rise time when $r = 100$ for the position is much longer than the nominal case in the Pre-Lab. Another thing to be noticed is that input voltage drop is larger and sharper for the input voltage in Figure 10 than the input plot in the Pre-Lab.

Run the sinusoidal reference from lab 6b ($r_1(t) = .05\sin(t)$) using the weights that you think are best. How does the performance compare to Lab 6b?

We think the best case is when we set the weight of $q_3 = 100$. As seen in Figure 18, increasing the weight of q_3 would give a better angle output as well as less input voltage. Reducing the input voltage results in better efficiency as well as saving energy. Efficiency cost is one of the most important strategies in the industry. Although, it sacrifices the position as a trades-off.

The performance compared to lab 6b is that the $q_3 = 100$ case gives less noisy input voltage and angle output as well as tracking both the input voltage and angle references better than lab 6b; however, the $q_3 = 100$ case tracks the position reference much worse than lab 6b which the phase lag is much larger and the magnitude of the response is also reduced.

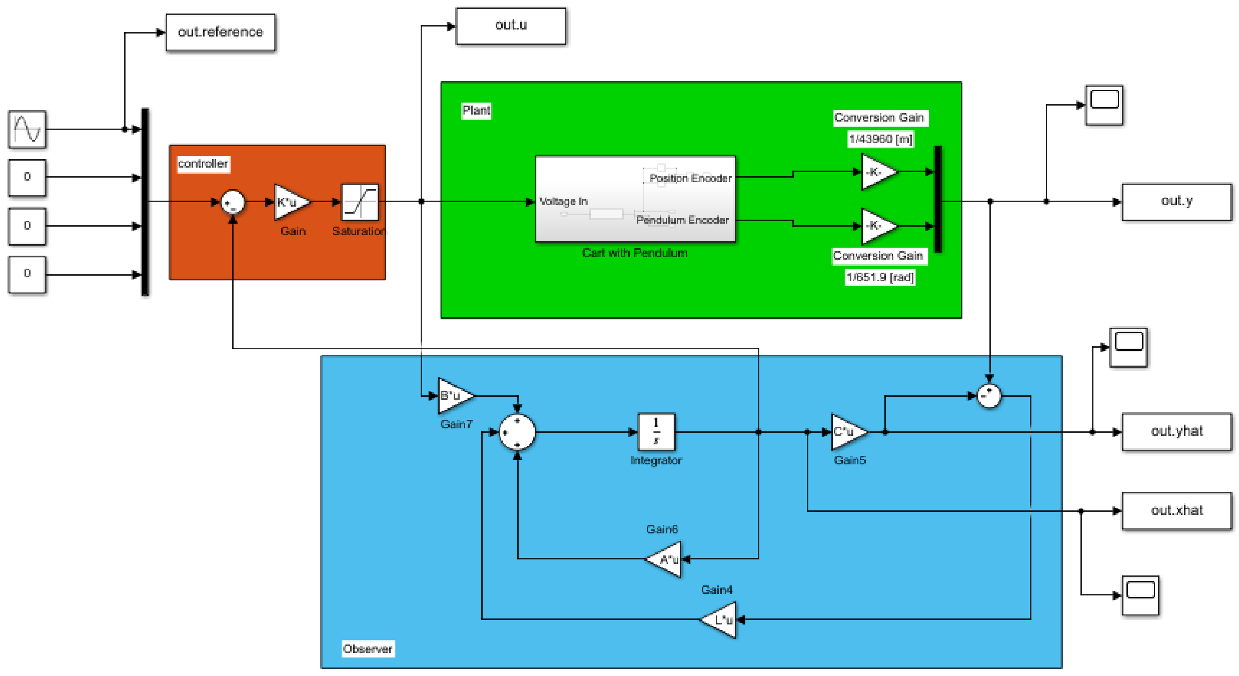
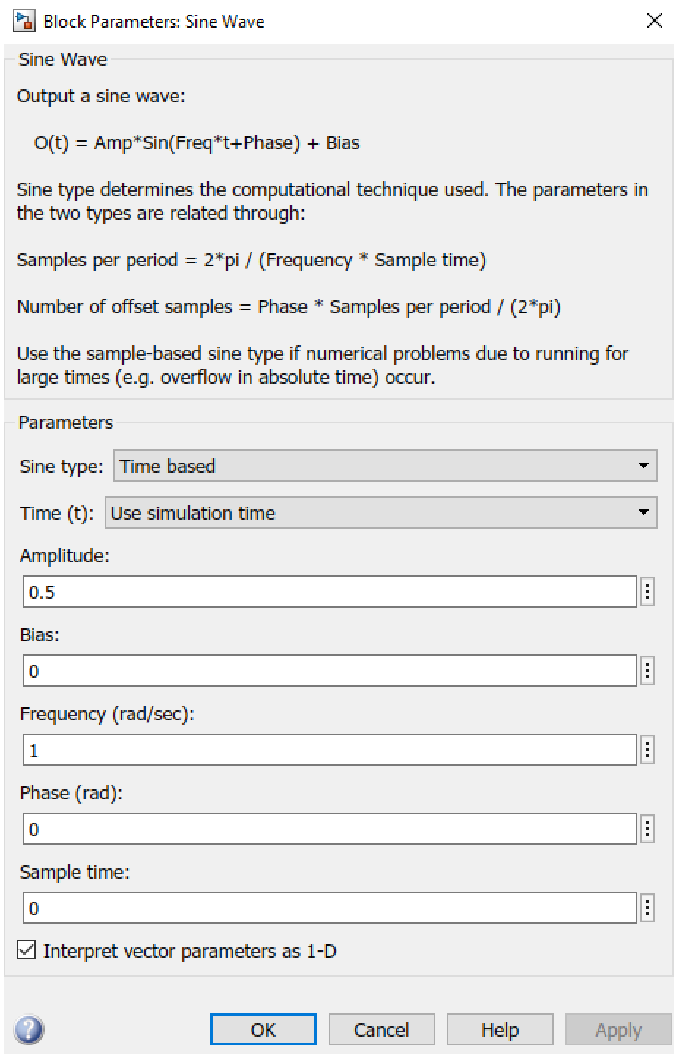


Figure 11: Simulink model used for the sinusoidal reference simulations.


Block Parameters: Sine Wave

Sine Wave

Output a sine wave:

O(t) = Amp*Sin(Freq*t+Phase) + Bias

Sine type determines the computational technique used. The parameters in the two types are related through:

Samples per period = 2*pi / (Frequency * Sample time)

Number of offset samples = Phase * Samples per period / (2*pi)

Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: 0.5

Bias: 0

Frequency (rad/sec): 1

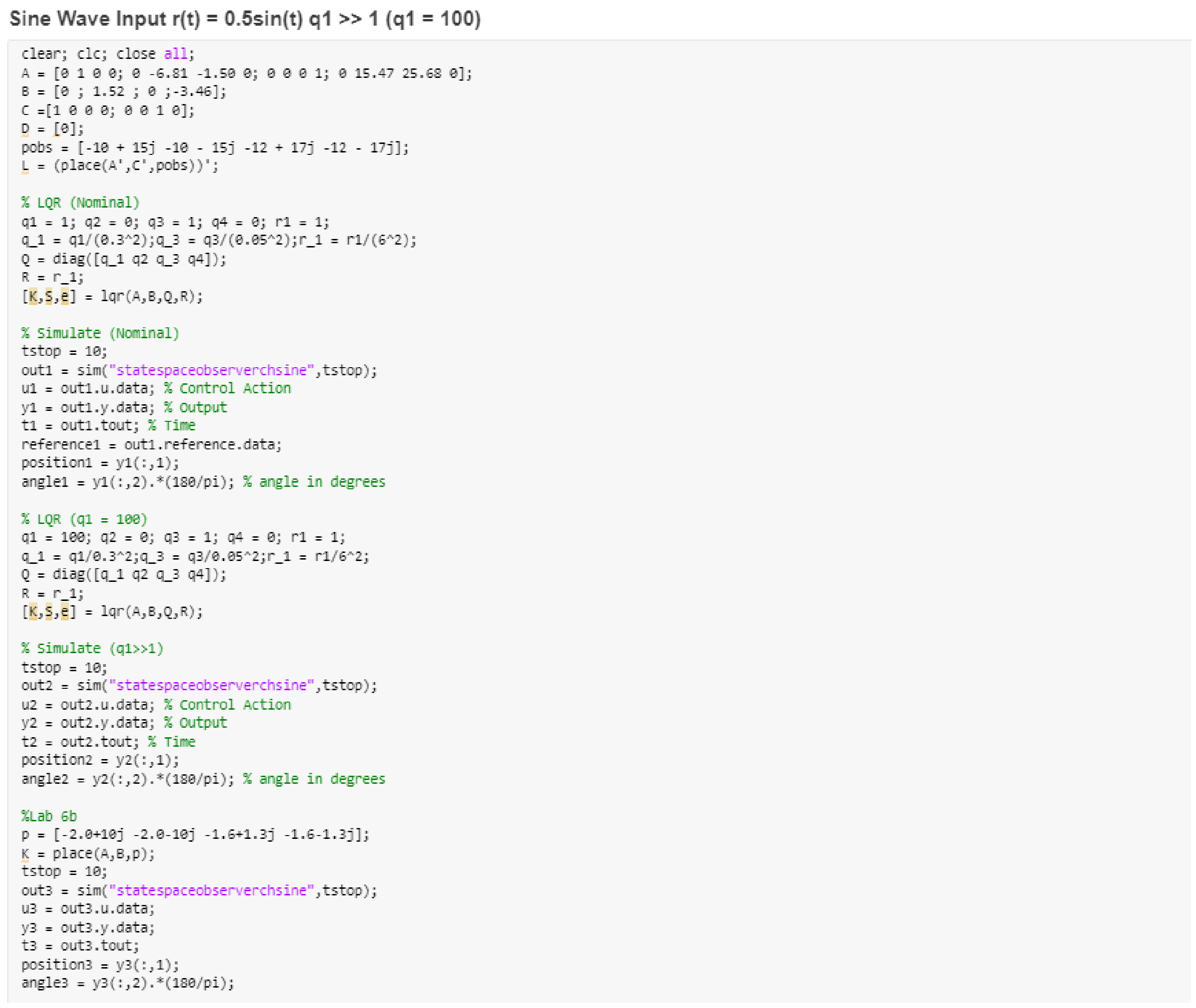
Phase (rad): 0

Sample time: 0

☑ Interpret vector parameters as 1-D

Figure 12: Parameters for sine wave function used in the sinusoidal reference simulations.

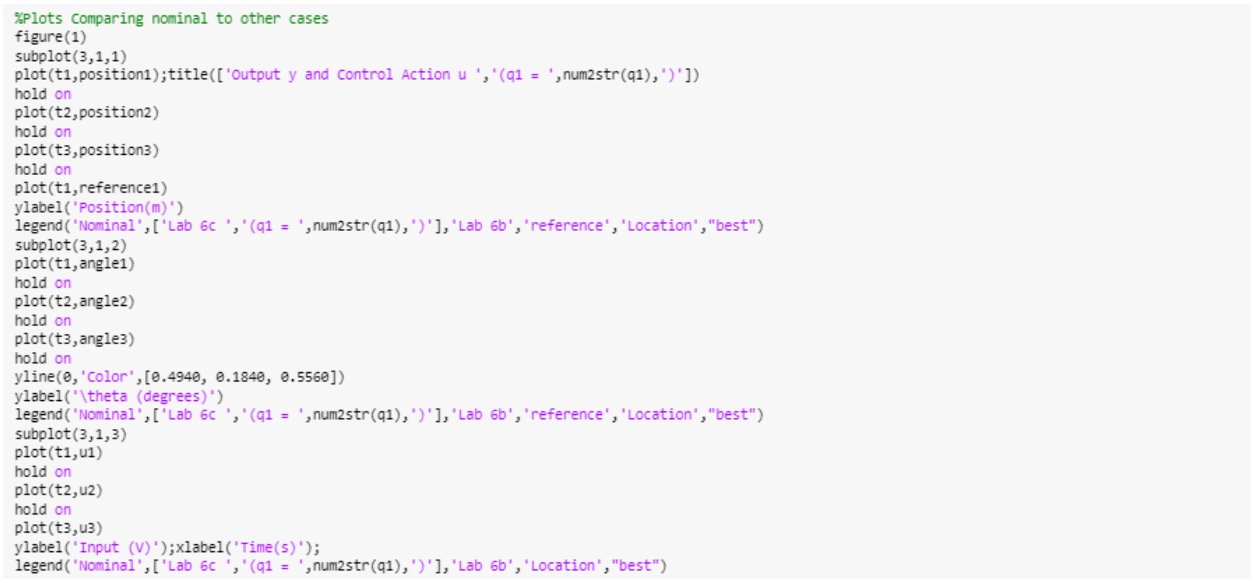
Best performance for Cart Position (q1>>1 or q1 =100)



Sine Wave Input r(t) = 0.5sin(t) q1 >> 1 (q1 = 100)

```
clear; clc; close all;
A = [0 1 0 0; 0 -6.81 -1.50 0; 0 0 0 1; 0 15.47 25.68 0];
B = [0 ; 1.52 ; 0 ;-3.46];
C =[1 0 0 0; 0 0 1 0];
D = [0];
pobs = [-10 + 15j -10 - 15j -12 + 17j -12 - 17j];
L = (place(A',C',pobs))';

% LQR (Nominal)
q1 = 1; q2 = 0; q3 = 1; q4 = 0; r1 = 1;
q_1 = q1/(0.3^2);q_3 = q3/(0.05^2);r_1 = r1/(6^2);
Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (Nominal)
tstop = 10;
out1 = sim("statespaceobserverchsine",tstop);
u1 = out1.u.data; % Control Action
y1 = out1.y.data; % Output
t1 = out1.tout; % Time
reference1 = out1.reference.data;
position1 = y1(:,1);
angle1 = y1(:,2).*(180/pi); % angle in degrees

% LQR (q1 = 100)
q1 = 100; q2 = 0; q3 = 1; q4 = 0; r1 = 1;
q_1 = q1/0.3^2;q_3 = q3/0.05^2;r_1 = r1/6^2;
Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (q1>>1)
tstop = 10;
out2 = sim("statespaceobserverchsine",tstop);
u2 = out2.u.data; % Control Action
y2 = out2.y.data; % Output
t2 = out2.tout; % Time
position2 = y2(:,1);
angle2 = y2(:,2).*(180/pi); % angle in degrees

%Lab 6b
p = [-2.0+10j -2.0-10j -1.6+1.3j -1.6-1.3j];
K = place(A,B,p);
tstop = 10;
out3 = sim("statespaceobserverchsine",tstop);
u3 = out3.u.data;
y3 = out3.y.data;
t3 = out3.tout;
position3 = y3(:,1);
angle3 = y3(:,2).*(180/pi);
```

Figure 13: MATLAB script used for simulating sinusoidal reference (q1 = 100 case) and Lab 6b.



```
%Plots Comparing nominal to other cases
figure(1)
subplot(3,1,1)
plot(t1,position1);title(['Output y and Control Action u ','(q1 = ',num2str(q1),')'])
hold on
plot(t2,position2)
hold on
plot(t3,position3)
hold on
plot(t1,reference1)
ylabel('Position(m)')
legend('Nominal',['Lab 6c ','(q1 = ',num2str(q1),')'],'Lab 6b','reference','Location',"best")
subplot(3,1,2)
plot(t1,angle1)
hold on
plot(t2,angle2)
hold on
plot(t3,angle3)
hold on
yline(0,'Color',[0.4940, 0.1840, 0.5560])
ylabel('\theta (degrees)')
legend('Nominal',['Lab 6c ','(q1 = ',num2str(q1),')'],'Lab 6b','reference','Location',"best")
subplot(3,1,3)
plot(t1,u1)
hold on
plot(t2,u2)
hold on
plot(t3,u3)
ylabel('Input (V)');xlabel('Time(s)');
legend('Nominal',['Lab 6c ','(q1 = ',num2str(q1),')'],'Lab 6b','Location',"best")
```

Figure 14: MATLAB script used for plotting the sinusoidal reference case (q1 = 100) against the Lab 6b simulation.

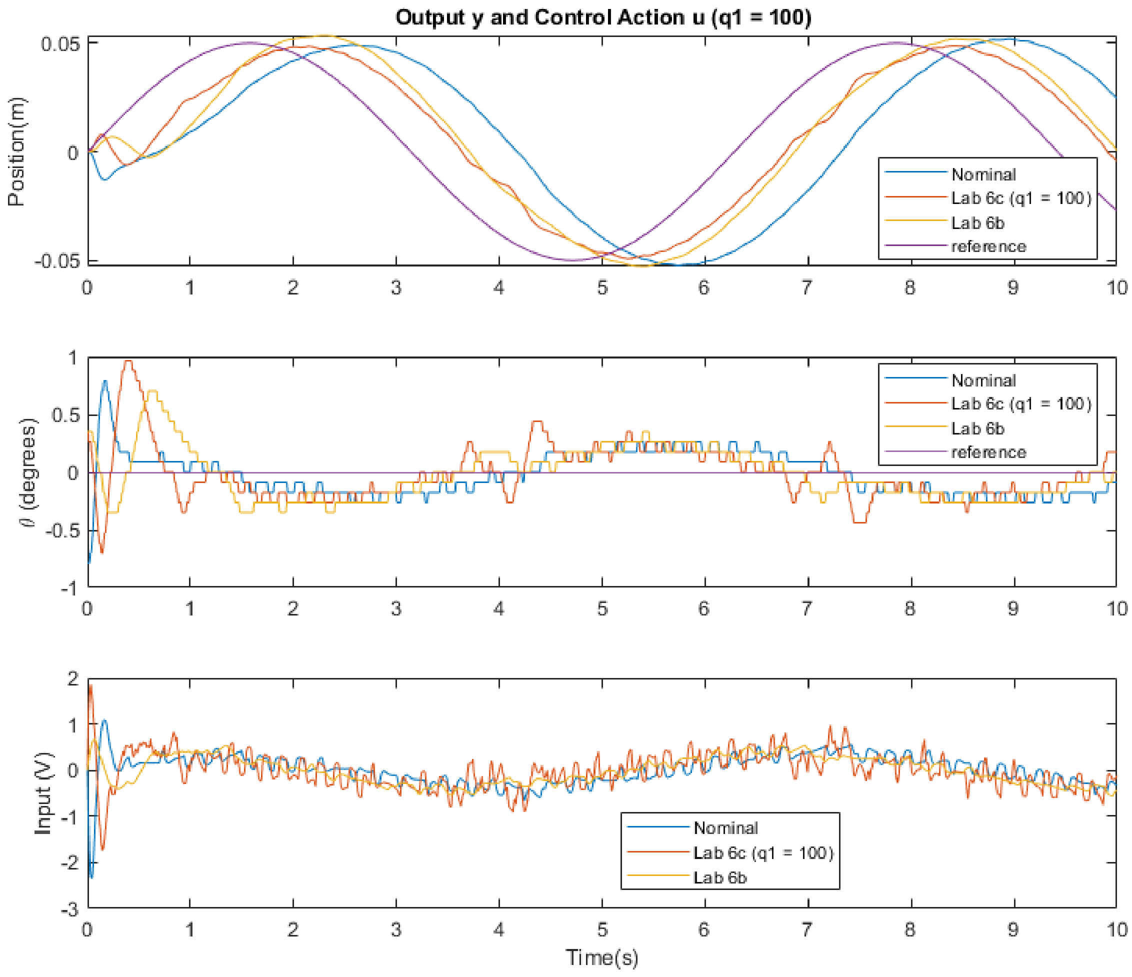
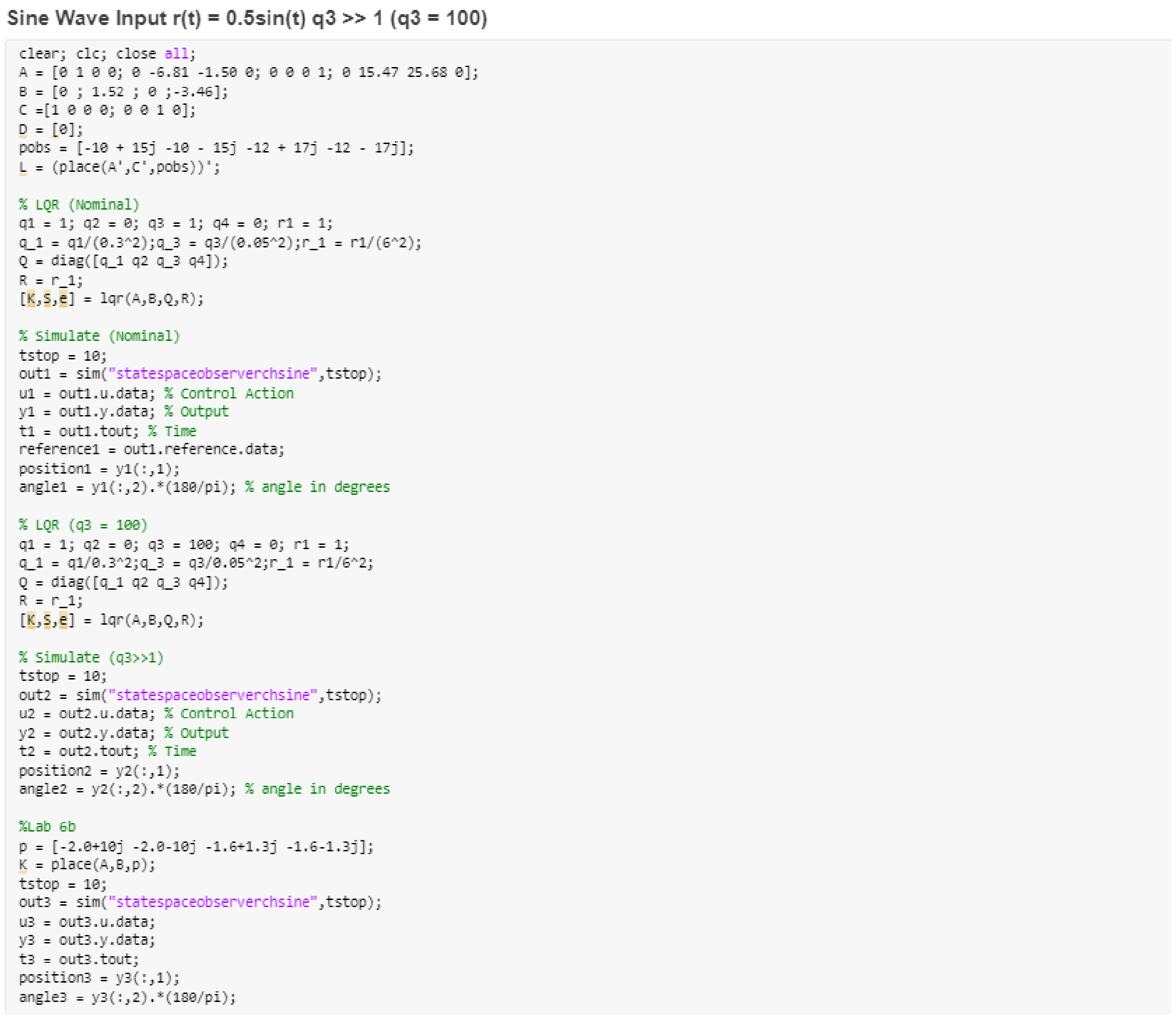


Figure 15: Plots of the sinusoidal reference simulation (q1 = 100) against the Lab 6b simulation.

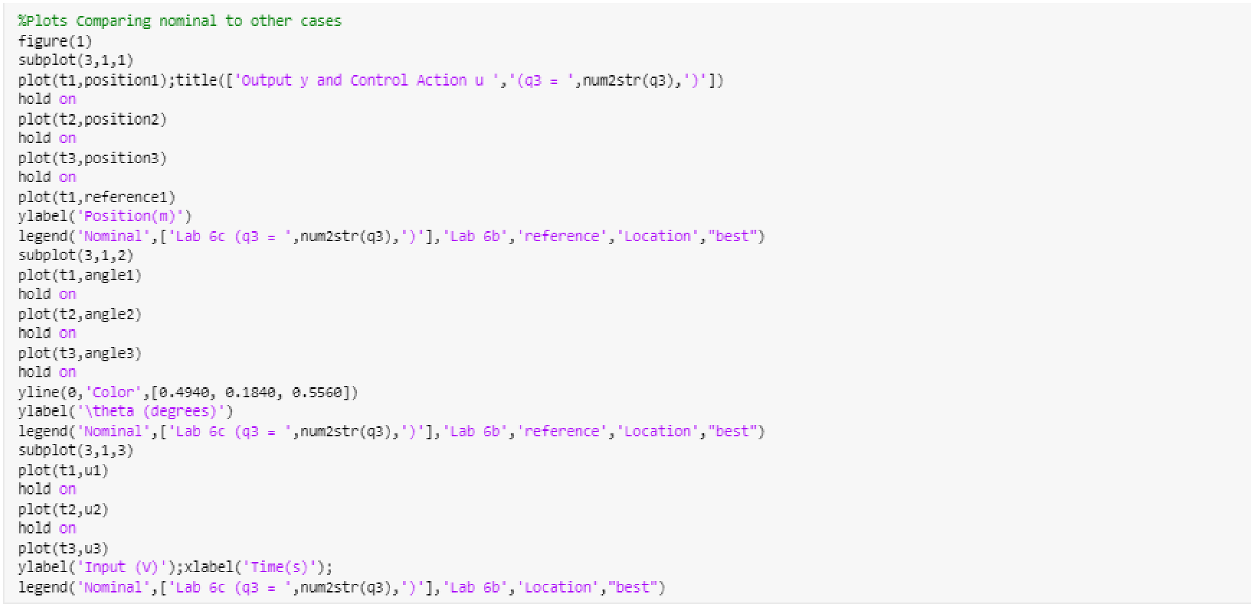
Best Performance for Pendulum Angle (q3>>1 or q3 =100)



Sine Wave Input r(t) = 0.5sin(t) q3 >> 1 (q3 = 100)

```
clear; clc; close all;
A = [0 1 0 0; 0 -6.81 -1.50 0; 0 0 0 1; 0 15.47 25.68 0];
B = [0 ; 1.52 ; 0 ;-3.46];
C =[1 0 0 0; 0 0 1 0];
D = [0];
pobs = [-10 + 15j -10 - 15j -12 + 17j -12 - 17j];
L = (place(A',C',pobs))';

% LQR (Nominal)
q1 = 1; q2 = 0; q3 = 1; q4 = 0; r1 = 1;
q_1 = q1/(0.3^2);q_3 = q3/(0.05^2);r_1 = r1/(6^2);
Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (Nominal)
tstop = 10;
out1 = sim("statespaceobserverchsine",tstop);
u1 = out1.u.data; % Control Action
y1 = out1.y.data; % Output
t1 = out1.tout; % Time
reference1 = out1.reference.data;
position1 = y1(:,1);
angle1 = y1(:,2).*(180/pi); % angle in degrees

% LQR (q3 = 100)
q1 = 1; q2 = 0; q3 = 100; q4 = 0; r1 = 1;
q_1 = q1/0.3^2;q_3 = q3/0.05^2;r_1 = r1/6^2;
Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (q3>>1)
tstop = 10;
out2 = sim("statespaceobserverchsine",tstop);
u2 = out2.u.data; % Control Action
y2 = out2.y.data; % Output
t2 = out2.tout; % Time
position2 = y2(:,1);
angle2 = y2(:,2).*(180/pi); % angle in degrees

%Lab 6b
p = [-2.0+10j -2.0-10j -1.6+1.3j -1.6-1.3j];
K = place(A,B,p);
tstop = 10;
out3 = sim("statespaceobserverchsine",tstop);
u3 = out3.u.data;
y3 = out3.y.data;
t3 = out3.tout;
position3 = y3(:,1);
angle3 = y3(:,2).*(180/pi);
```

Figure 16: MATLAB script used for simulating sinusoidal reference (q3 = 100 case) and Lab 6b.



```
%Plots Comparing nominal to other cases
figure(1)
subplot(3,1,1)
plot(t1,position1);title(['Output y and Control Action u ','(q3 = ',num2str(q3),')'])
hold on
plot(t2,position2)
hold on
plot(t3,position3)
hold on
plot(t1,reference1)
ylabel('Position(m)')
legend('Nominal',['Lab 6c (q3 = ',num2str(q3),')'],'Lab 6b','reference','Location',"best")
subplot(3,1,2)
plot(t1,angle1)
hold on
plot(t2,angle2)
hold on
plot(t3,angle3)
hold on
yline(0,'Color',[0.4940, 0.1840, 0.5560])
ylabel('\theta (degrees)')
legend('Nominal',['Lab 6c (q3 = ',num2str(q3),')'],'Lab 6b','reference','Location',"best")
subplot(3,1,3)
plot(t1,u1)
hold on
plot(t2,u2)
hold on
plot(t3,u3)
ylabel('Input (V)');xlabel('Time(s)');
legend('Nominal',['Lab 6c (q3 = ',num2str(q3),')'],'Lab 6b','Location',"best")
```

Figure 17: MATLAB script used for plotting the sinusoidal reference case (q3 = 100) against the Lab 6b simulation.

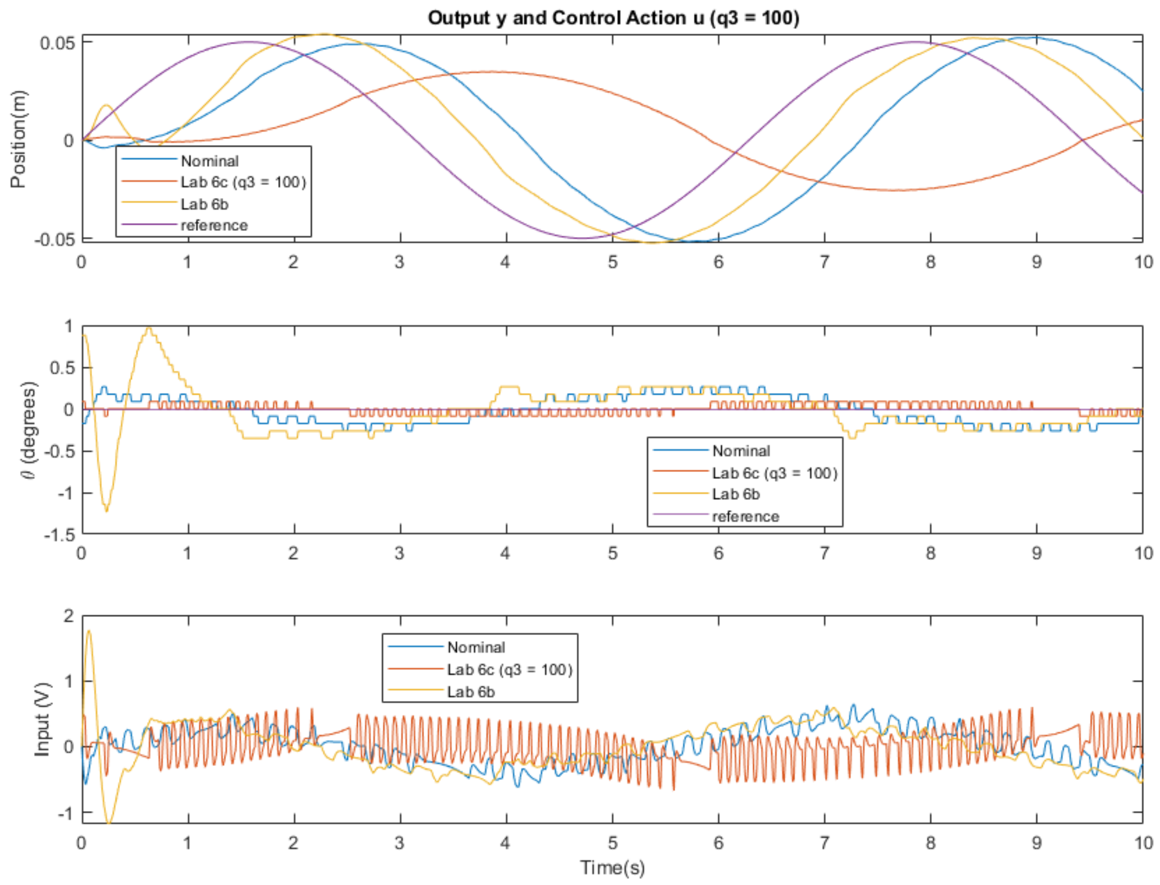
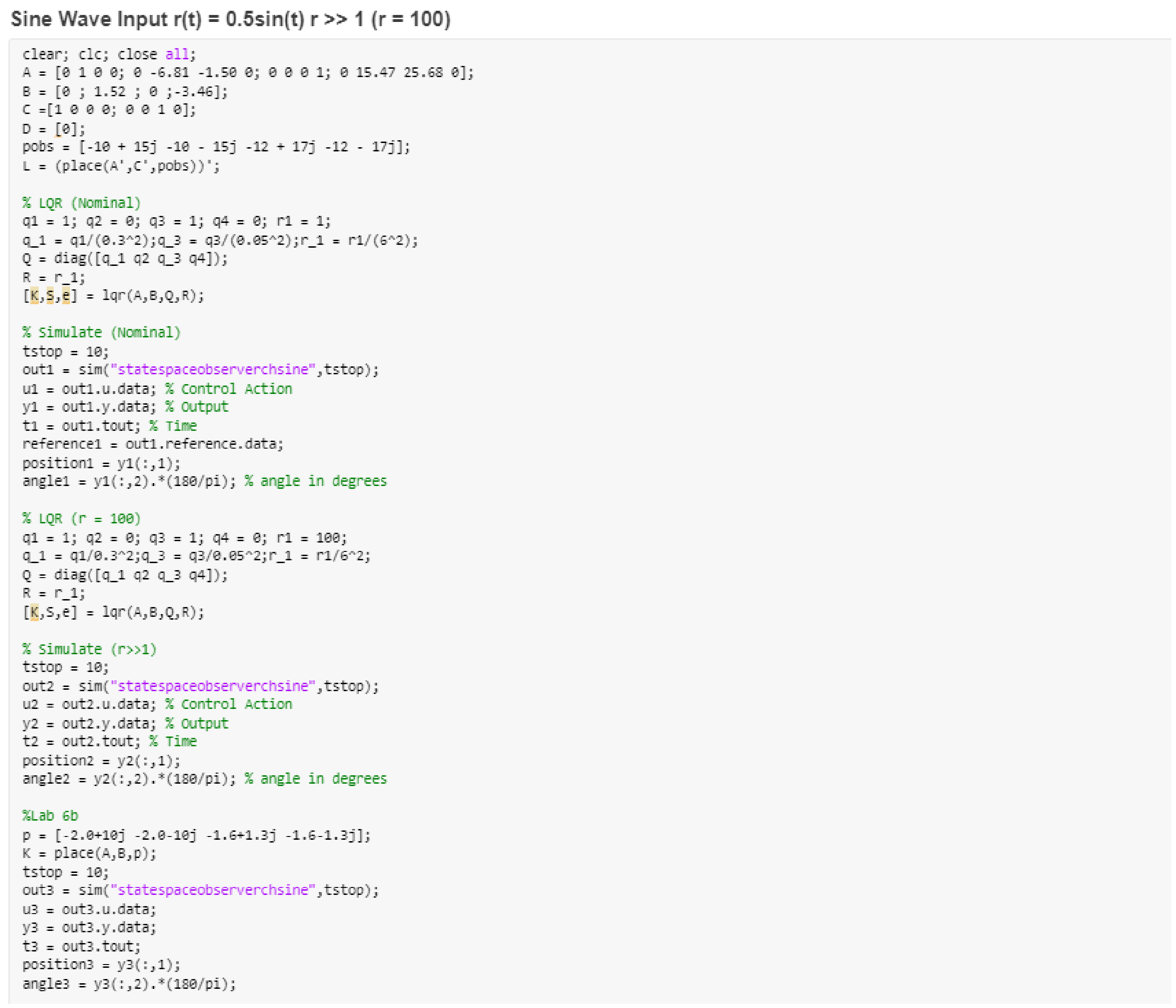


Figure 18: Plots of the sinusoidal reference simulation (q3 = 100) against the Lab 6b simulation.

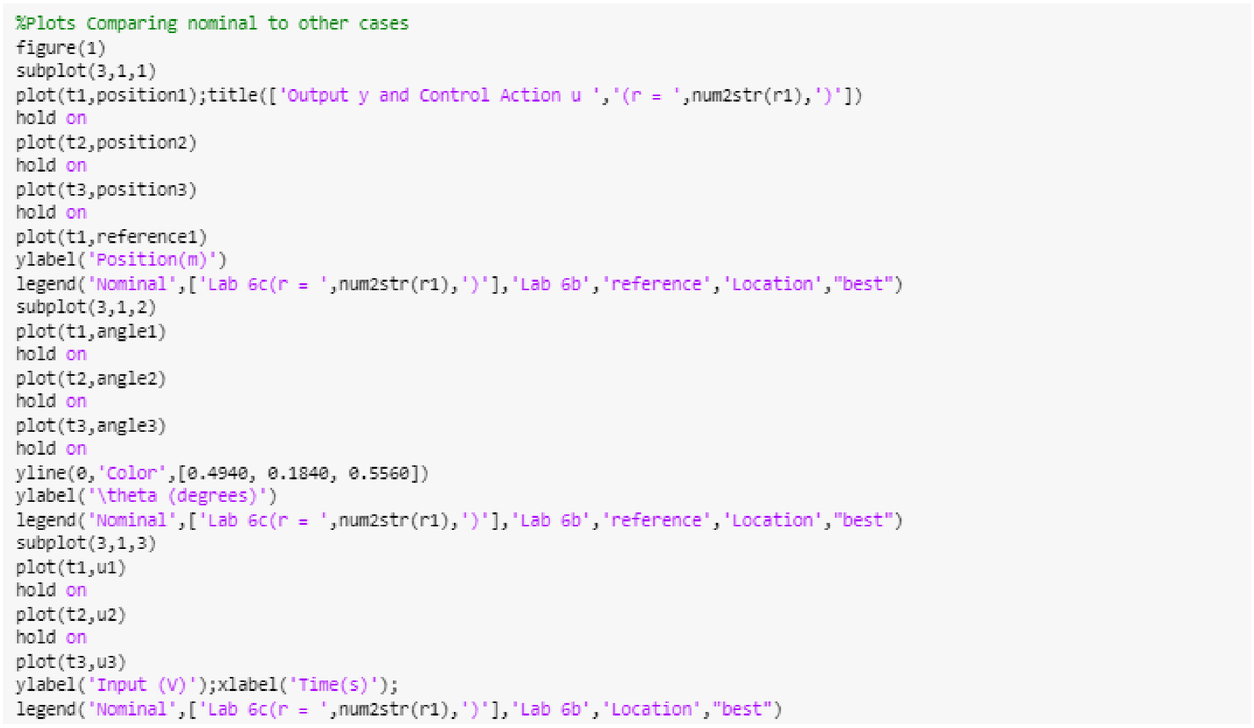
Best Performance for Input (r >> 1 or r = 100)



Sine Wave Input r(t) = 0.5sin(t) r >> 1 (r = 100)

```
clear; clc; close all;
A = [0 1 0 0; 0 -6.81 -1.50 0; 0 0 0 1; 0 15.47 25.68 0];
B = [0 ; 1.52 ; 0 ;-3.46];
C =[1 0 0 0; 0 0 1 0];
D = [0];
pobs = [-10 + 15j -10 - 15j -12 + 17j -12 - 17j];
L = (place(A',C',pobs))';

% LQR (Nominal)
q1 = 1; q2 = 0; q3 = 1; q4 = 0; r1 = 1;
q_1 = q1/(0.3^2);q_3 = q3/(0.05^2);r_1 = r1/(6^2);
Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (Nominal)
tstop = 10;
out1 = sim("statespaceobserverchsine",tstop);
u1 = out1.u.data; % Control Action
y1 = out1.y.data; % Output
t1 = out1.tout; % Time
reference1 = out1.reference.data;
position1 = y1(:,1);
angle1 = y1(:,2).*(180/pi); % angle in degrees

% LQR (r = 100)
q1 = 1; q2 = 0; q3 = 1; q4 = 0; r1 = 100;
q_1 = q1/0.3^2;q_3 = q3/0.05^2;r_1 = r1/6^2;
Q = diag([q_1 q2 q_3 q4]);
R = r_1;
[K,S,e] = lqr(A,B,Q,R);

% Simulate (r>>1)
tstop = 10;
out2 = sim("statespaceobserverchsine",tstop);
u2 = out2.u.data; % Control Action
y2 = out2.y.data; % Output
t2 = out2.tout; % Time
position2 = y2(:,1);
angle2 = y2(:,2).*(180/pi); % angle in degrees

%Lab 6b
p = [-2.0+10j -2.0-10j -1.6+1.3j -1.6-1.3j];
K = place(A,B,p);
tstop = 10;
out3 = sim("statespaceobserverchsine",tstop);
u3 = out3.u.data;
y3 = out3.y.data;
t3 = out3.tout;
position3 = y3(:,1);
angle3 = y3(:,2).*(180/pi);
```

Figure 19: MATLAB script used for simulating sinusoidal reference (r = 100 case) and Lab 6b.



```
%Plots Comparing nominal to other cases
figure(1)
subplot(3,1,1)
plot(t1,position1);title(['Output y and Control Action u ','(r = ',num2str(r1),')'])
hold on
plot(t2,position2)
hold on
plot(t3,position3)
hold on
plot(t1,reference1)
ylabel('Position(m)')
legend('Nominal',['Lab 6c(r = ',num2str(r1),')'],'Lab 6b','reference','Location',"best")
subplot(3,1,2)
plot(t1,angle1)
hold on
plot(t2,angle2)
hold on
plot(t3,angle3)
hold on
yline(0,'Color',[0.4940, 0.1840, 0.5560])
ylabel('\theta (degrees)')
legend('Nominal',['Lab 6c(r = ',num2str(r1),')'],'Lab 6b','reference','Location',"best")
subplot(3,1,3)
plot(t1,u1)
hold on
plot(t2,u2)
hold on
plot(t3,u3)
ylabel('Input (V)');xlabel('Time(s)');
legend('Nominal',['Lab 6c(r = ',num2str(r1),')'],'Lab 6b','Location',"best")
```

Figure 20: MATLAB script used for plotting the sinusoidal reference case (r = 100) against the Lab 6b simulation.

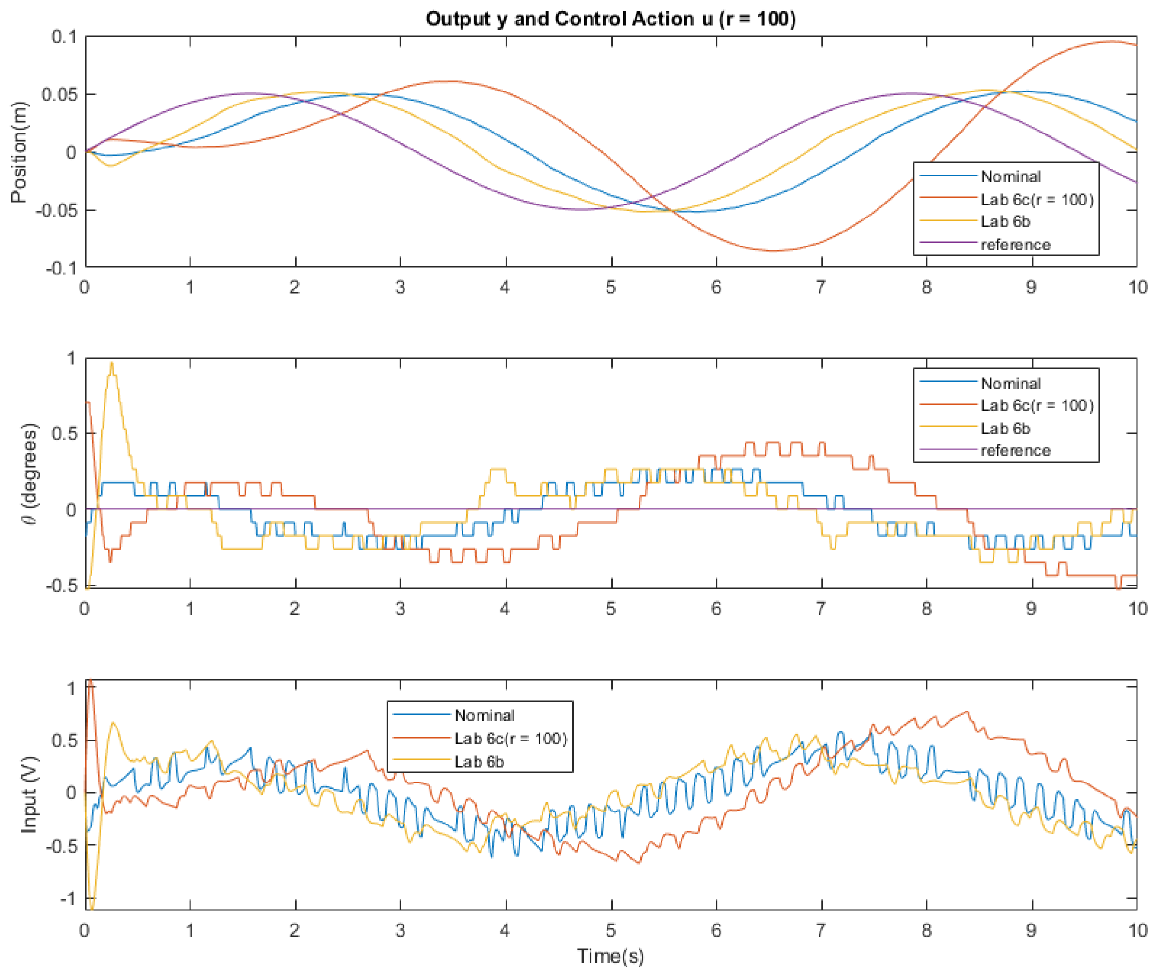


Figure 21: Plots of the sinusoidal reference simulation (r = 100) against the Lab 6b simulation.