ME-5554 Applied Linear System
Final Project

Luan Cong Doan
luandoan@vt.edu

Dec 7, 2015

The design engineer at Precision 3D Measuring Inc. have just developed a prototype 3D Coordinate Measuring Machine (CMM). In order to keep the cost low, the engineers have significantly reduced the amount of structural support in the frame, which unfortunately increase the compliance at the measurement probe. High compliance is generally not acceptable in a precision measurement system.

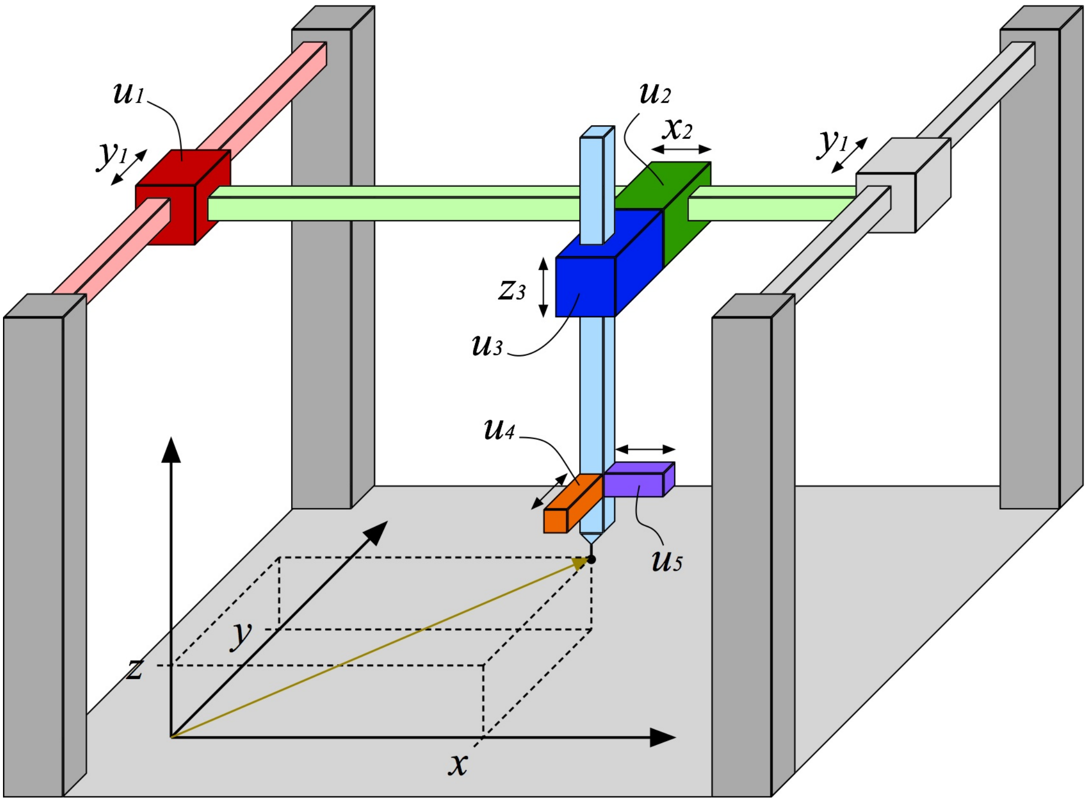

Figure 1: Prototype 3D Coordinate Measurement System

Equation of Motion

The equation of motion for prototype CMM have already been derived and are given by the following equations:

$$\dot{p}_1 = \alpha_1 u_1 - \left(\frac{b_y}{M_y}\right) p_1 - \dot{p}_4 \qquad \dot{p}_5 = \alpha_2 u_2 - \left(\frac{b_x}{M_x}\right) p_5 - \dot{p}_8$$

$$\dot{q}_2 = \left(\frac{1}{M_y}\right) p_1 - \left(\frac{1}{m_4}\right) p_4 - \dot{q}_3 \qquad \dot{q}_6 = \left(\frac{1}{M_x}\right) p_5 - \left(\frac{1}{m_5}\right) p_8 - \dot{q}_7$$

$$\dot{q}_3 = \left(\frac{1}{b_4}\right) (\alpha_4 u_4 + \dot{p}_4 - k_4 q_3) \qquad \dot{q}_7 = \left(\frac{1}{b_5}\right) (\alpha_5 u_5 + \dot{p}_8 - k_5 q_7)$$

$$\dot{p}_4 = k_y q_2 \qquad \dot{p}_8 = k_x q_6$$

$$\dot{y}_1 = \left(\frac{1}{M_y}\right) p_1 \qquad \dot{x}_2 = \left(\frac{1}{M_x}\right) p_5$$

$$y = y_1 - q_2 \qquad x = x_2 - q_6$$

$$M_z \ddot{z} = -b_z \dot{z} + \alpha_3 u_3$$

Because the final project only focus on the y-direction dynamics, states and control signals. So we have:

State Space is defined:

$$\dot{S}_y = A_y.S_y + B_y.u_y$$
$$y = C_y.S_y + D_y.u_y$$

With:

State variables: $S_y = \begin{bmatrix} y_1 & p_1 & q_2 & q_3 & p_4 \end{bmatrix}^T$

$$\dot{S}_y = \begin{bmatrix} \dot{y}_1 & \dot{p}_1 & \dot{q}_2 & \dot{q}_3 & \dot{p}_4 \end{bmatrix}^T$$

Input: $u_y = \begin{bmatrix} u_1 \\ u_4 \end{bmatrix}$

State matrix:

$$A_y = \begin{bmatrix} 0 & \frac{1}{M_y} & 0 & 0 & 0 \\ 0 & -\frac{b_y}{M_y} & -k_y & 0 & 0 \\ 0 & \frac{1}{M_y} & -\frac{k_y}{b_4} & \frac{k_4}{b_4} & -\frac{1}{m_4} \\ 0 & 0 & \frac{k_y}{b_4} & -\frac{k_4}{b_4} & 0 \\ 0 & 0 & k_y & 0 & 0 \end{bmatrix}$$

Input matrix:

$$B_y = \begin{bmatrix} 0 & 0 \\ \alpha_1 & 0 \\ 0 & -\frac{\alpha_4}{b_4} \\ 0 & \frac{\alpha_4}{b_4} \\ 0 & 0 \end{bmatrix}$$

Output matrix: $C_y = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \end{bmatrix}$

Direct Transmission matrix: $D_y = [0]$

Numerical values:

$M_y = 150kg, \quad M_x = 100kg, \quad M_z = 50kg, \quad b_y = 40Ns/m, \quad b_x = 50Ns/m,$

$b_z = 10Ns/m, \quad k_x = 0.2N/m, \quad m_4 = 20kg, \quad b_4 = 2.51Ns/m, \quad k_4 = 7.89N/m,$

$m_5 = 20kg, \quad b_5 = 3.77Ns/m, \quad k-5 = 7.89N/m, \quad k_y = 0.1N/m,$

$\alpha_1 = 0.05N/V, \quad \alpha_2 = 0.1N/V, \quad \alpha_3 = 0.1N/V, \quad \alpha_4 = 0.3N/V, \quad \alpha_5 = 0.5N/V$

So we have numerical results:

State matrix A_y:

$$A_y = \begin{bmatrix} 0 & 0.0067 & 0 & 0 & 0 \\ 0 & -0.2667 & -0.1 & 0 & 0 \\ 0 & 0.0067 & -0.0398 & 3.1434 & -0.05 \\ 0 & 0 & 0.0398 & -3.1434 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \end{bmatrix}$$

Input matrix B_y:

$$B_y = \begin{bmatrix} 0 & 0 \\ 0.05 & 0 \\ 0 & -0.1195 \\ 0 & 0.1195 \\ 0 & 0 \end{bmatrix}$$

Output matrix C_y:

$$C_y = \begin{bmatrix} 10 - 100 \end{bmatrix}$$

Direct Transmission matrix D_y: $D_y = [0]$

Problem 1: Determine the observability of the open-loop plant.

Matlab code:

```
Ob = obsv(A_y,C_y)';    % observability of system check
rOb = rank(Ob);
testO = size(A_y,1);
fprintf('Rank of observability matrix is %d. ', rOb);
if rOb == testO
fprintf('System is observable.\n');
else
fprintf('System is not observable.\n');
end
```

Result:

```
Rank of observable matrix is 5. System is observable.
```

Problem 2: Design the state feedback controller gains with conditions:

$|y| \leq 0.2m$ after 10 seconds

$|u_1| \leq 10000, \qquad |u_4| \leq 1000$

$y_1(0) = 0.2; \qquad p_4(0) = -0.5, \qquad$ all other states are zero.

Matlab code:

```
SF0 = [0.2; 0; 0; 0; -0.5]; % initial condition for problem 2
CPoles = [linspace(-1.1, -1.0,5)];      % poles location
G = place(A_y,B_y,CPoles);
AcF = A_y - B_y*G;
BF = [];
syscF = ss(AcF, BF, C_y,D_y);
```

```
[ycF,tcF,xcF] = initial(syscF,SF0,16);
figure; plot(tcF,ycF(:,1));
grid on;
xlabel('Time'); ylabel('Output y-direction');
title('Output of Y in 16 seconds');
print('OutputYdirection16s','-dpng');

U = -G*xcF';
figure; plot(tcF,U(1,:),'r',tcF,U(2,:),'b');
grid on; legend('u1','u4');
xlabel('Time'); ylabel('Control response');
title('Control input space for U1 and U4 in 16 seconds');
print('ControlInputU1U4_16s','-dpng');
```

Ouput in Y direction:

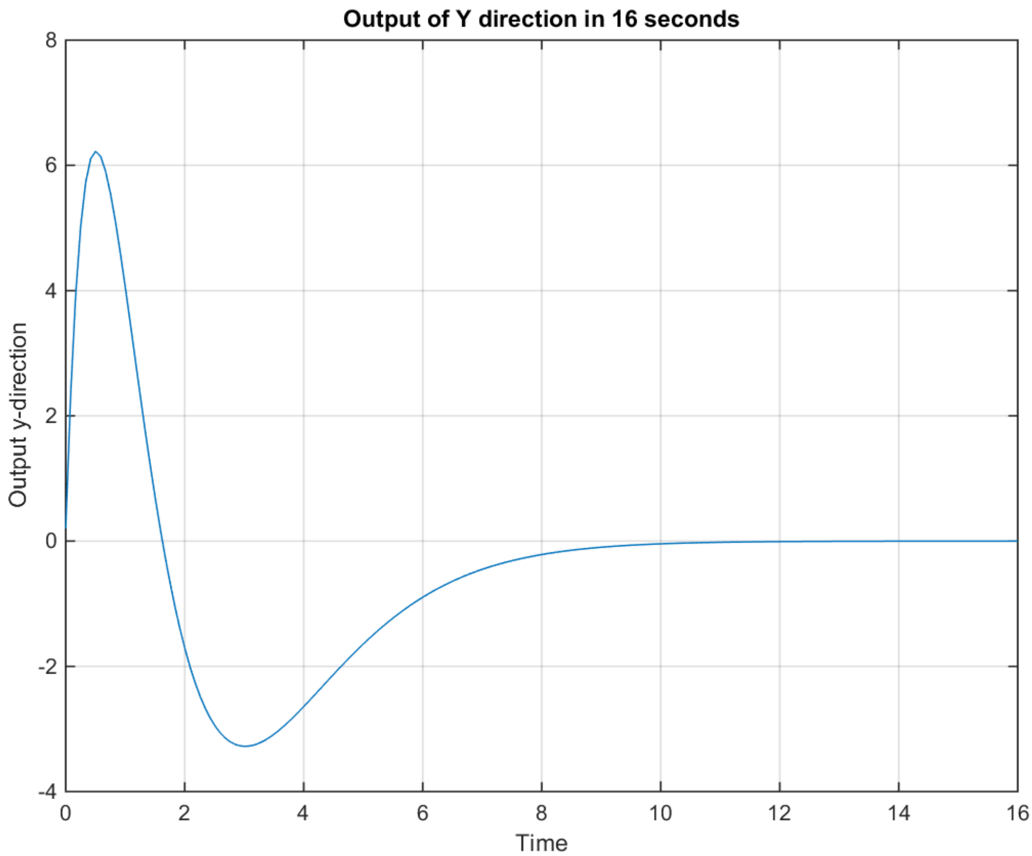

Figure 2: Output in Y direction in 16s

Control input space of U1 and U4:

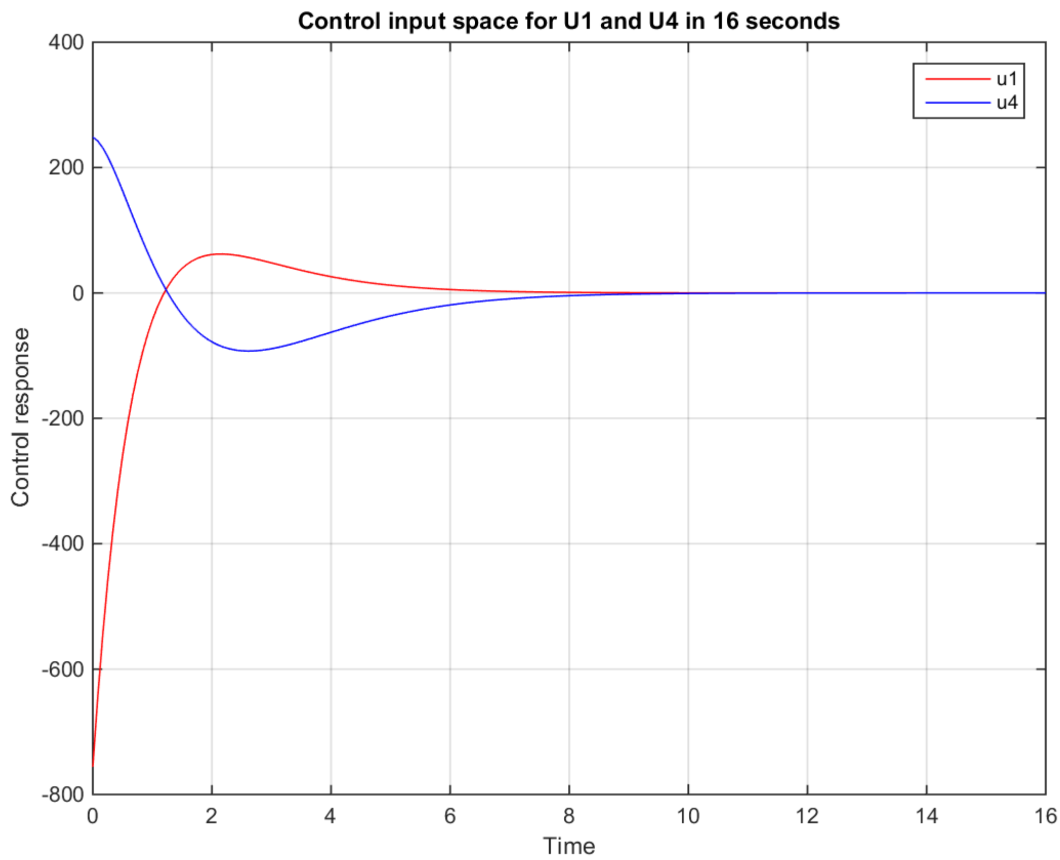

Figure 3: Control input of U1 and U4 in 16s

The state feedback controller gains G is:

```
G

G =

1.0e+03 *

3.9346    0.0367   -0.6439   -0.6449    0.0618
-1.9330   -0.0001    1.9068    1.9065   -0.2761
```

Design closed-loop estimation:

State-space equation: $$\dot{S}_y = A_y.S_y + B_y.u_y$$

$$y = C_y.S_y + D_y.u_y$$

with K is the observer feedback matrix we have Dynamic State Estimator:

$$\dot{\hat{S}}_y = \hat{A}_y\hat{S}_y + \hat{B}_y u_y + K(y - \hat{y})$$

$$\hat{y} = \hat{C}_y\hat{S}_y + \hat{D}_y.u_y$$

with assume that we have a perfect model: $A_y = \hat{A}_y, \quad B_y = \hat{B}_y$

$$C_y = \hat{C}_y, \quad D = \hat{D}$$

Poles for closed-loop observer:

$$OPoles = \begin{bmatrix} -8 & -6.5 & -5 & -0.5 & 0 \end{bmatrix}$$

Open-loop poles, closed-loop poles and the observer poles:

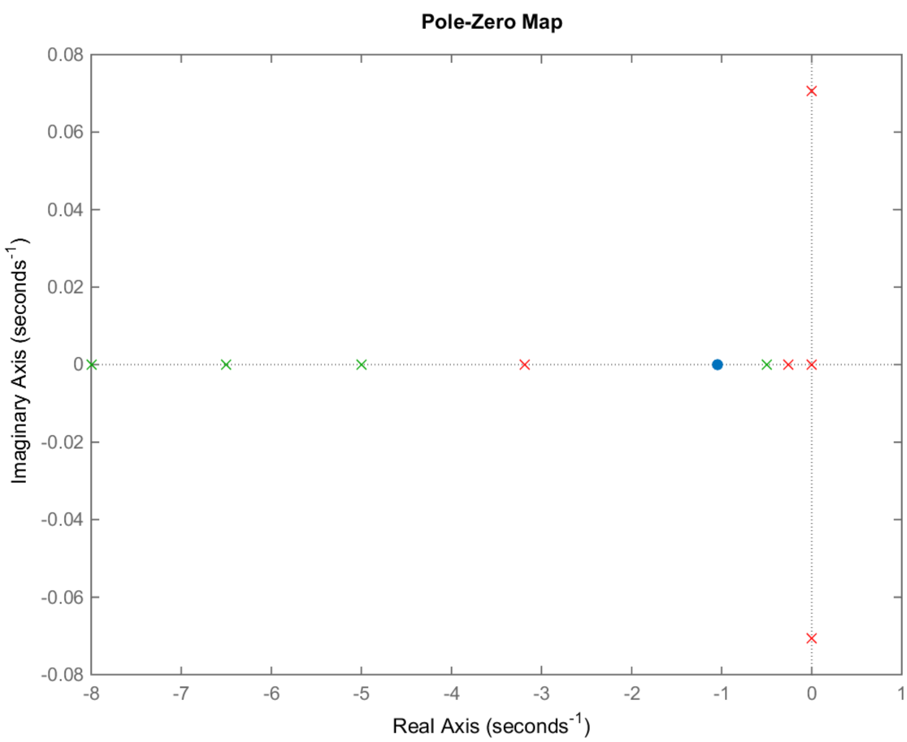

Figure 4: Open-loop poles, closed-loop poles, and observer poles

red x - poles for Open-loop system. green x - poles for observer system.

blue circle - poles for closed-loop (because 5 poles very close together, so it support to be like a circle on pole-zero map).

The state estimation error dynamics for the Luenberger observer is defined:

$$e_y = S_y - \hat{S}_y \Rightarrow \dot{e}_y = \dot{S}_y - \dot{\hat{S}}_y$$

Output estimation error: $\tilde{y} = y - \hat{y}$

We have the state estimation error dynamics:
$$\dot{e}_y(t) = [A_y - KC_y]e_y(t)$$
$$\tilde{y} = C_y e_y(t)$$

The first 4 seconds should be an open loop response of the plant to allow the observer to converge, and the controller cannot be turned on until 4 seconds. It means that observer out put $\tilde{y}$ should reduced to 0 in first 4 seconds.

We have this solved by Matlab:

```
OPoles = [-8 -6.5 -5 -0.5 0];  % poles for closed-loop observer
Ob0 = [0; 0; 0; 0; 0];          % initial condition of observer
e0 = S0 - Ob0;                  % initial for error estimation
K = place(A_y',C_y',OPoles)';
% Check error estimate
    AoE = A_y - K*C_y;
    BoE = [];
    CoE = C_y;
    DoE = [];
sysOE = ss(AoE, BoE, CoE, DoE);      % system oberver error
[yOE, tOE, sOE] = initial(sysOE, e0, 16);
figure;
plot(tOE,yOE); grid on;
xlabel('Time'); ylabel('Error estimation');
title('Error Estimation by Observation');
print('ErrorEstimation','-dpng');
```

Result:

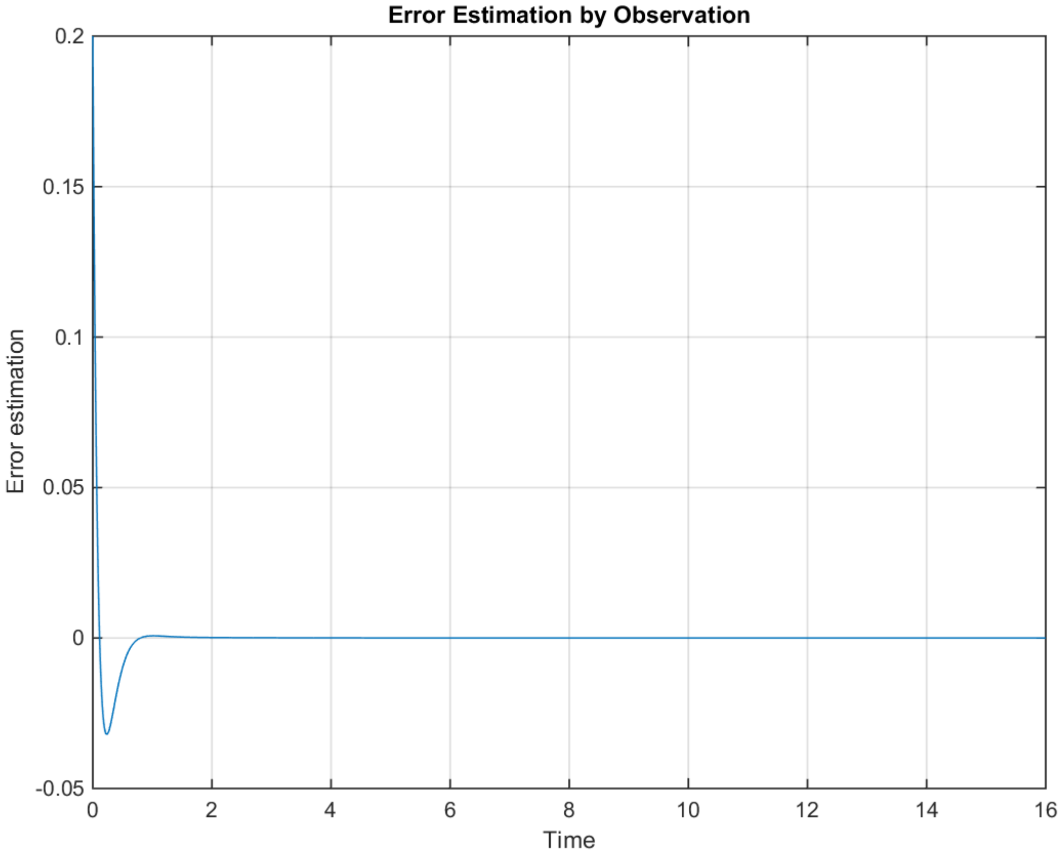

Figure 5: Error estimation in 16 seconds

For first 4 seconds, system should be an open-loop response of the plant to allow the observer to converge. So we have first augmented state-space system for open-loop and observer:

$$\begin{bmatrix} \dot{S}_y \\ \dot{\hat{S}}_y \end{bmatrix} = \begin{bmatrix} A_y & 0 \\ KC_y & A_y - KC_y \end{bmatrix} \begin{bmatrix} S_y \\ \hat{S}_y \end{bmatrix} + \begin{bmatrix} B_y \\ B_y \end{bmatrix} u$$

$$\begin{bmatrix} y \\ \hat{y} \end{bmatrix} = \begin{bmatrix} C_y & 0 \\ 0 & C_y \end{bmatrix} \begin{bmatrix} S_y \\ \hat{S}_y \end{bmatrix}$$

The augmented system has twice as many states and outputs as the plant or observer. So we have:

```
Zero1 = zeros(5,5);  Zero2 = zeros(1,5); % create temp zero matrix
Ag0 = [0.2;0;0;0;-0.5;0;0;0;0;0];  % initial condition for Augmente
G2 = [G,G];                  % gain for Augmented
Ag = [A_y, Zero1; K*C_y, A_y-K*C_y];
Bg = [B_y; B_y];
Cg = [C_y, Zero2; Zero2, C_y];
Dg = [];
sysOp = ss(Ag, Bg, Cg, Dg);
[yOp, tOp, sOp] = initial(sysOp, Ag0, 4);
figure;
plot(tOp, yOp(:,1),'r', tOp, yOp(:,2),'--'); grid on;
xlabel('Time'); ylabel('Output open-loop and observation');
legend('Open-loop','Oservation');
title('Open-loop Plant with Observation');
print('OpenLoopObservation','-dpng');
```

Result:

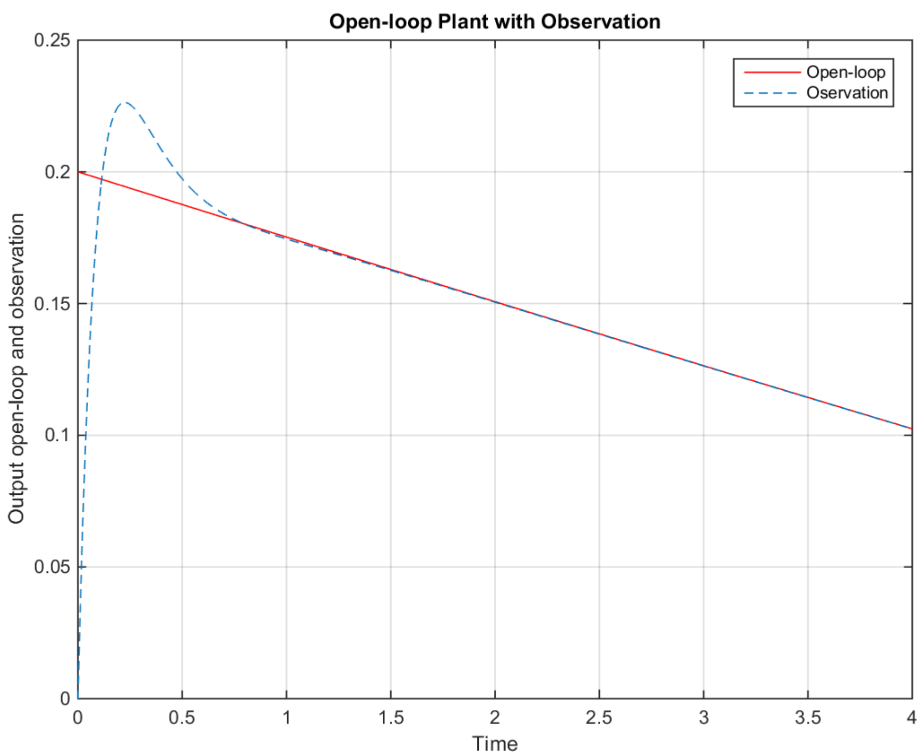

Figure 6: Open-loop Observation in first 4 seconds

After 4 seconds, controller will be turned on, we will have the full state feedback control law with:

- Estimated states: $u_y = -G.\hat{S}_y$
- Initial condition is the latest state of above Open-loop and observation system:

$$int0 = sOp(end, :)$$

The augmented state space of full state feedback system is defined:

$$\begin{bmatrix} \dot{S}_y \\ \dot{\hat{S}}_y \end{bmatrix} = \begin{bmatrix} A_y & -B_y G \\ KC_y & A_y - KC_y - B_y G \end{bmatrix} \begin{bmatrix} S_y \\ \hat{S}_y \end{bmatrix} + [0]u'$$

$$\begin{bmatrix} y \\ \hat{y} \end{bmatrix} = \begin{bmatrix} C_y & 0 \\ 0 & C_y \end{bmatrix} \begin{bmatrix} S_y \\ \hat{S}_y \end{bmatrix} + [0]u'$$

Solving system by Matlab:

```
AgC = [A_y, -B_y*G; K*C_y, A_y-K*C_y-B_y*G];
BgC = [];
CgC = [C_y, Zero2; Zero2, C_y];
DgC = [];
int0 = sOp(end,:);
sysOC = ss(AgC, BgC, CgC, DgC);
[yOC, tOC, sOC] = initial(sysOC, int0, 12);
figure;
plot(tOC, yOC(:,1),'r', tOC, yOC(:,2), '--'); grid on;
xlabel('Time'); ylabel('Observation output');
legend('Closed-loop','Oservation');
title('Closed-loop Plant with Observation');
print('ClosedLoopObservation','-dpng');

UgC = -G2*sOC';
figure; plot(tOC, UgC); grid on;
xlabel('Time'); ylabel('Control input'); legend('u1','u4');
title('Control inputs Close-loop Observation');
print('ClosedLoopControlInput','-dpng');
```

Output in Y-direction and Control Inputs of closed-loop observation simulation:

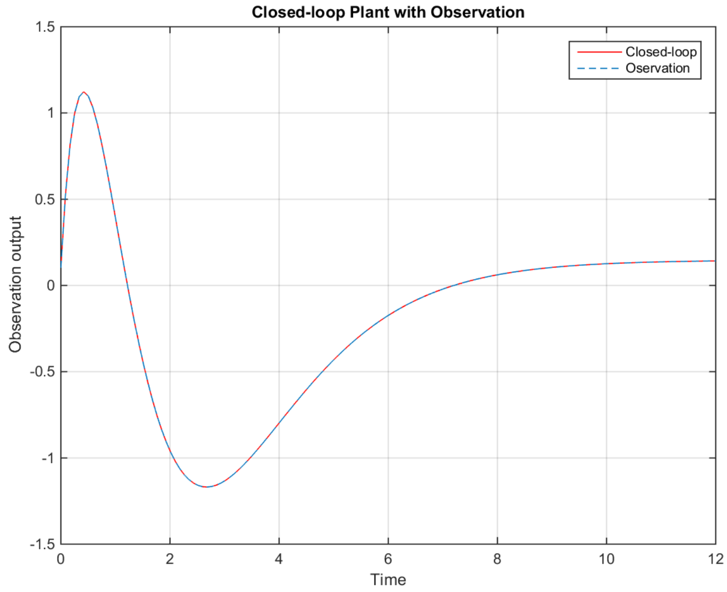

Figure 7: Closed-loop with Observation in next 12 seconds

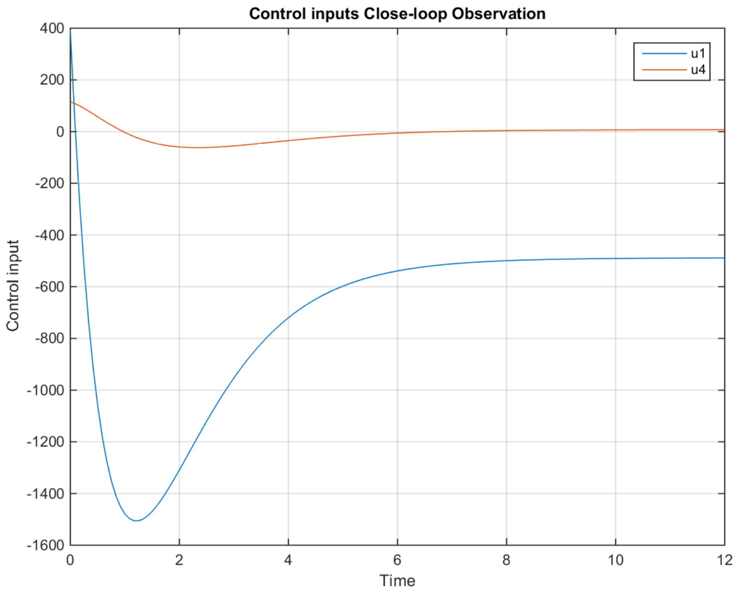

Figure 8: Closed-loop Control inputs with Observation in next 12 seconds

Complete simulation for system in 16 seconds:

```
%% Plot all system output and control input
yt = [yOp;yOC];
tt = [tOp;4+tOC];
figure;
plot(tt,yt(:,1),'r',tt,yt(:,2),'--'); grid on;
xlabel('Time'); ylabel('Output Y');
legend('Output','Observation');
title('Ouput Simulation System in 16 seconds');
print('OutputSimulation16s','-dpng');
```

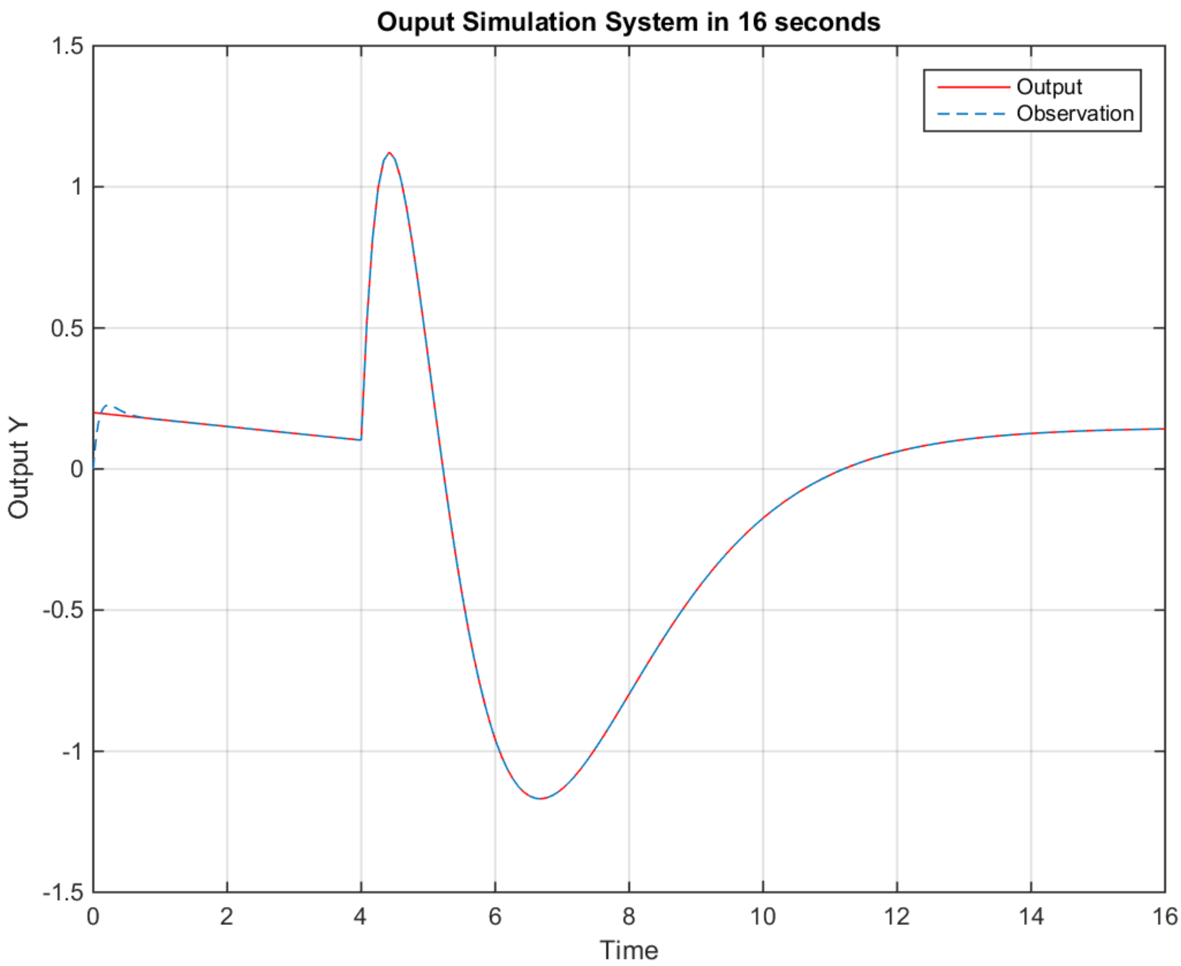

Figure 9: Output Simulation System in 16 seconds

```
u_temp = G2*sOp';
sizeUOp = size(u_temp);
UOp = zeros(sizeUOp);
ut = [UOp, UgC];
figure;
plot(tt,ut); grid on;
xlabel('Time'); ylabel('Control input');
legend('u1','u4');
title('Control inputs in Simulation system in 16 seconds');
print('ControlInputSimulation16s','-dpng');
```

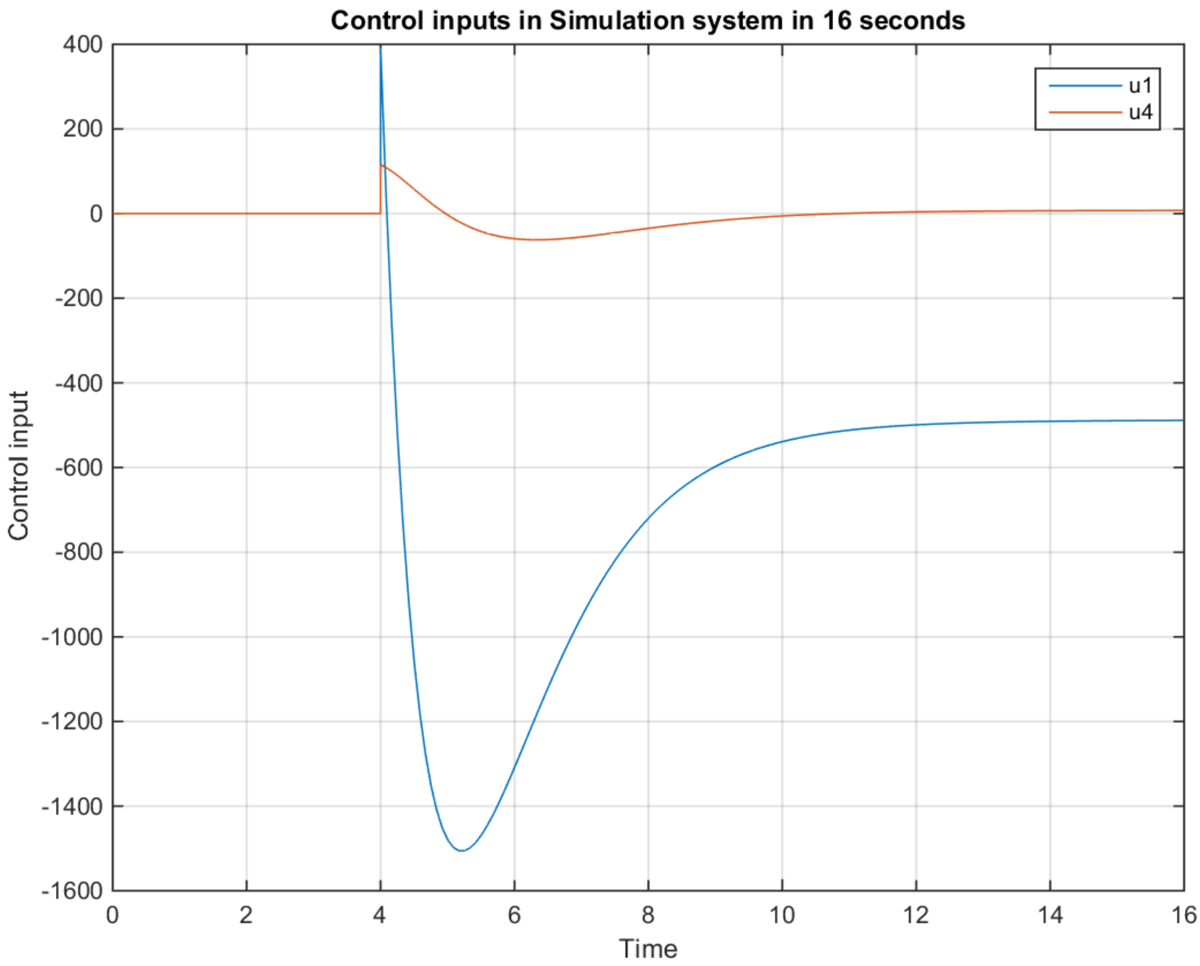

Figure 10: Control inputs in Simulation system in 16 seconds

Plot system:

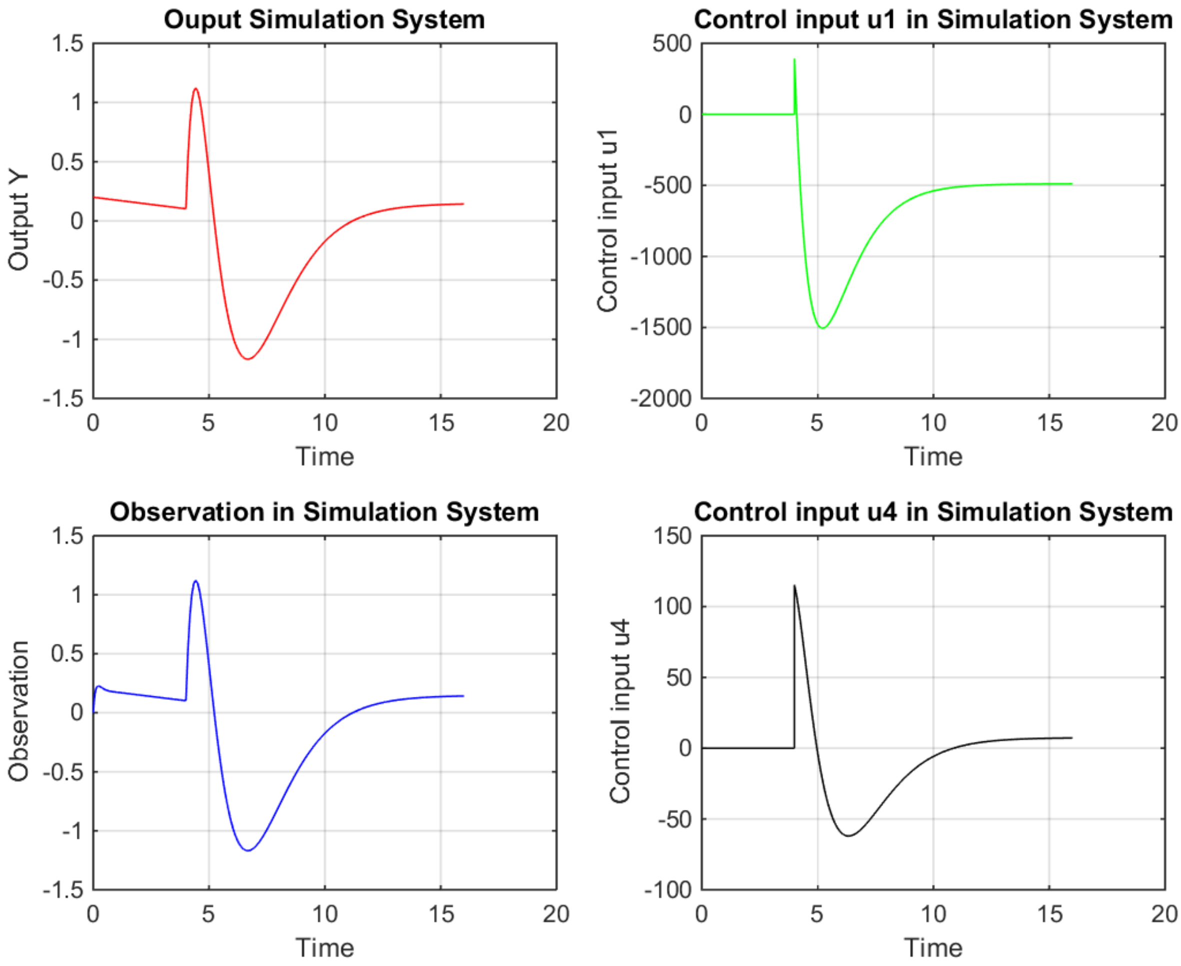

Figure 11: Full Simulation system in 16 seconds