Project 2 - Control Systems Design - Fall 2021

Alan Chacko

11/12/2021

Project Formulation

Fuel cells are electro-chemical-mechanical devices that produce electricity from hydrogen rich fuels without generating carbon dioxide. The process is reversed to water electrolysis. A fuel cell is a triode anode-membrane-cathode. Depending on the type of membrane there are several types of fuel cells. The most developed and used are the proton exchange membrane (PEM) fuel cells.

A Proton Exchange Membrane can be modeled using a 9th order differential equation, with its state space variables given as:

$$x = \begin{bmatrix} m_{O_2} & m_{H_2} & m_{N_2} & \omega_{cp} & p_{sm} & m_{sm} & m_{H_2O_A} & m_{H_2O_C} & p_{rm} \end{bmatrix}^T$$

The subscripts cp, sm, rm stand respectively for the compressor, supply manifold, and return manifold. ω_{cp} is the compressor angular velocity. The m terms refer to the molarity of a molecule in the cell, and p terms refers to the pressure in each manifold.

The output to predict is the cathode air molar flow rate, whose input is dependent on the anode hydrogen molar flow rate (W_{cp}), the fuel cell stack voltage (v_{st}), and supply manifold pressure (p_{sm}).

$$y(t) = \begin{bmatrix} W_{cp} & p_{sm} & v_{st} \end{bmatrix}^T$$

The linearized model can be represented by the following state-space matricies. Note only 8 states are linearized, as the 9th state adds little to the observability of the system.

$$A = \begin{bmatrix} -6.30908 & 0 & -10.9544 & 0 & 83.74458 & 0 & 0 & 24.05866 \\ 0 & -161.083 & 0 & 0 & 51.52923 & 0 & -18.0261 & 0 \\ -18.7858 & 0 & -46.3136 & 0 & 275.6592 & 0 & 0 & 158.3741 \\ 0 & 0 & 0 & -17.3506 & 193.9373 & 0 & 0 & 0 \\ 1.299576 & 0 & 2.969317 & 0.3977 & -38.7024 & 0.105748 & 0 & 0 \\ 16.64244 & 0 & 38.02522 & 5.066579 & -479.384 & 0 & 0 & 0 \\ 0 & -450.386 & 0 & 0 & 142.2084 & 0 & -80.9472 & 0 \\ 2.02257 & 0 & 4.621237 & 0 & 0 & 0 & 0 & -51.2108 \end{bmatrix}$$

$$B^T = \begin{bmatrix} 0 & 0 & 0 & 3.946683 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 5.066579 & -116.446 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 12.96989 & 10.32532 & -0.56926 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Full Order Observer Equations

Knowing the state space form of the system is:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y = Cx(t)$$

We can design a full-order observer by finding an observer gain matrix K and feedback gain matrix F.

The closed loop eigenvalues should be perserved depending on the F matrix used. In our case the eigenvalues of the following must be negative:

$$0 > eig((A - BF)x(t))$$

The observer gain matrix K requires that closed-loop poles are placed farther left of the complex plane, usually 5 times the largest nondominant pole. Placing poles excessively far could introduce sinusoidal noise to the input and output responses.

$$0 > eig((A - KC)x(t))$$

We choose arbitrary desired poles, in my example, I kept the desired poles the same as those given by the eigenvalues A (which are all stable). The poles for the observer are placed farther left. We can then use the `place` function in MATLAB to generate the F and K matrices.

Then we can finally define the system of the observer as:

$$\dot{\hat{x}}(t) = (A - KC)\hat{x}(t) + \begin{bmatrix} B & K \end{bmatrix} \begin{bmatrix} u(t) \\ y(t) \end{bmatrix}$$

To define the initial inputs, $x(0)$ is arbitrarily picked, and we can use least squares estimation to determine the initial $\hat{x}(0)$. Since:

$$y(0) = C * x(0)$$
$$\hat{x}(0) = (C^T C)^{-1} C^T y(0)$$

Reduced Order Observer Equations

For the reduced-order observer, our observer has to be a rank of the number of inputs minus the rank of C. So the observer needs to be rank 5.

We need a reduced-order output gain matrix C_1 such that $p(t) = C_1 x(t)$ where $p(t)$ represents reduced order output, in contrast to $y(t)$ which is full-order. Creating an augmented matrix of $[C \quad C_1]^T$ can be used to define the input in terms of the full-order output and reduced-order put, shown below:

$$x(t) = \begin{bmatrix} C \\ C_1 \end{bmatrix}^{-1} \begin{bmatrix} y(t) \\ p(t) \end{bmatrix} = \begin{bmatrix} L & L_1 \end{bmatrix} \begin{bmatrix} y(t) \\ p(t) \end{bmatrix}$$

We can ultimately find estimate $\hat{x}(t)$ using $\hat{p}(t)$ instead.

To find C_1 and L_1, we can use the following relationships:

$$C \times L = I_C \quad C_1 \times L = 0 \quad C \times L_1 = 0 \quad C_1 \times L = 0$$

C_1 can be found by finding the nullspace of the inverse of L. We can estimate L with the product $C' \times C$. Taking the nullspace of this estimated L gives a C_1 that fullfills all the above relationships. Once we have C and C_1, finding the inverse gives L and L_1.

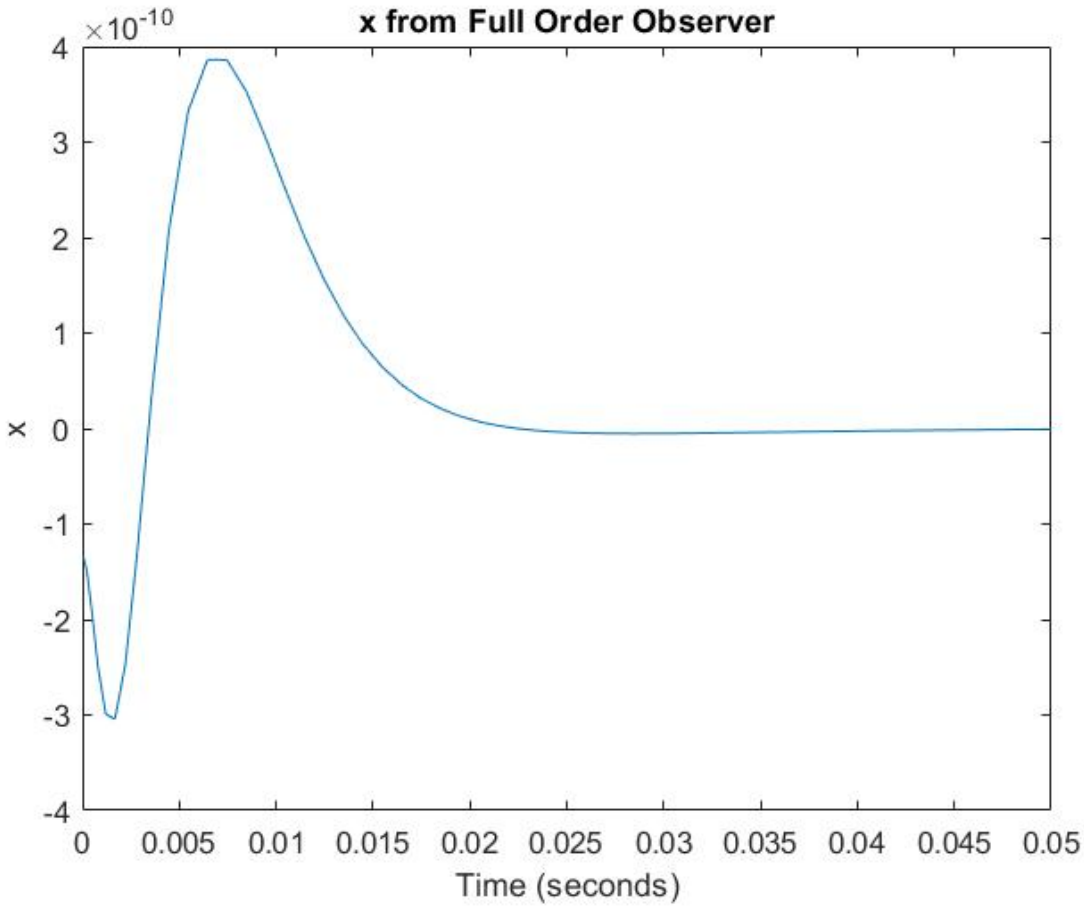
After checking for observability, we define our desired poles for the observer which in this case consist of the 5 non dominant open loop poles of A.

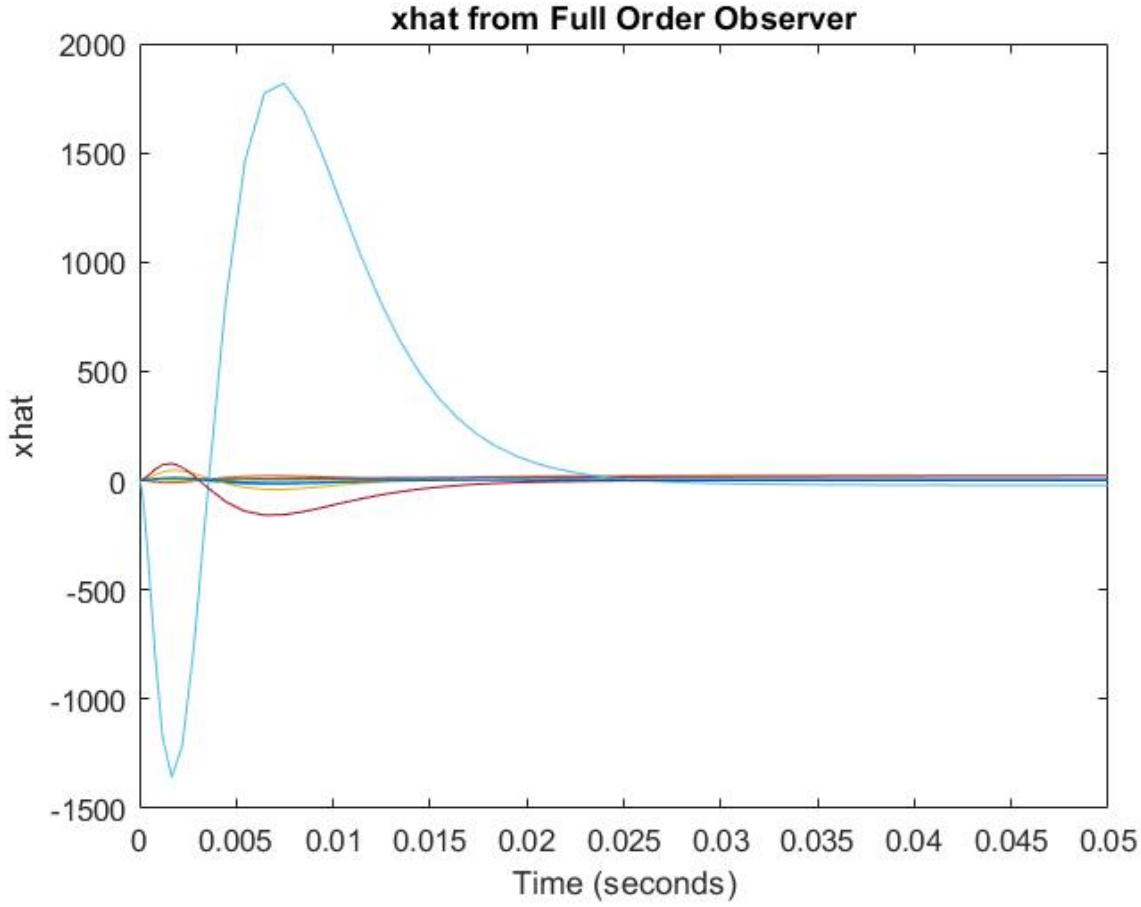
We calculate an observer gain, reduced order A matrix, reduced order B matrix, and reduced order observer gain matrix using $C1$, $L1$, and A. Show in the matlab code in the appendix.

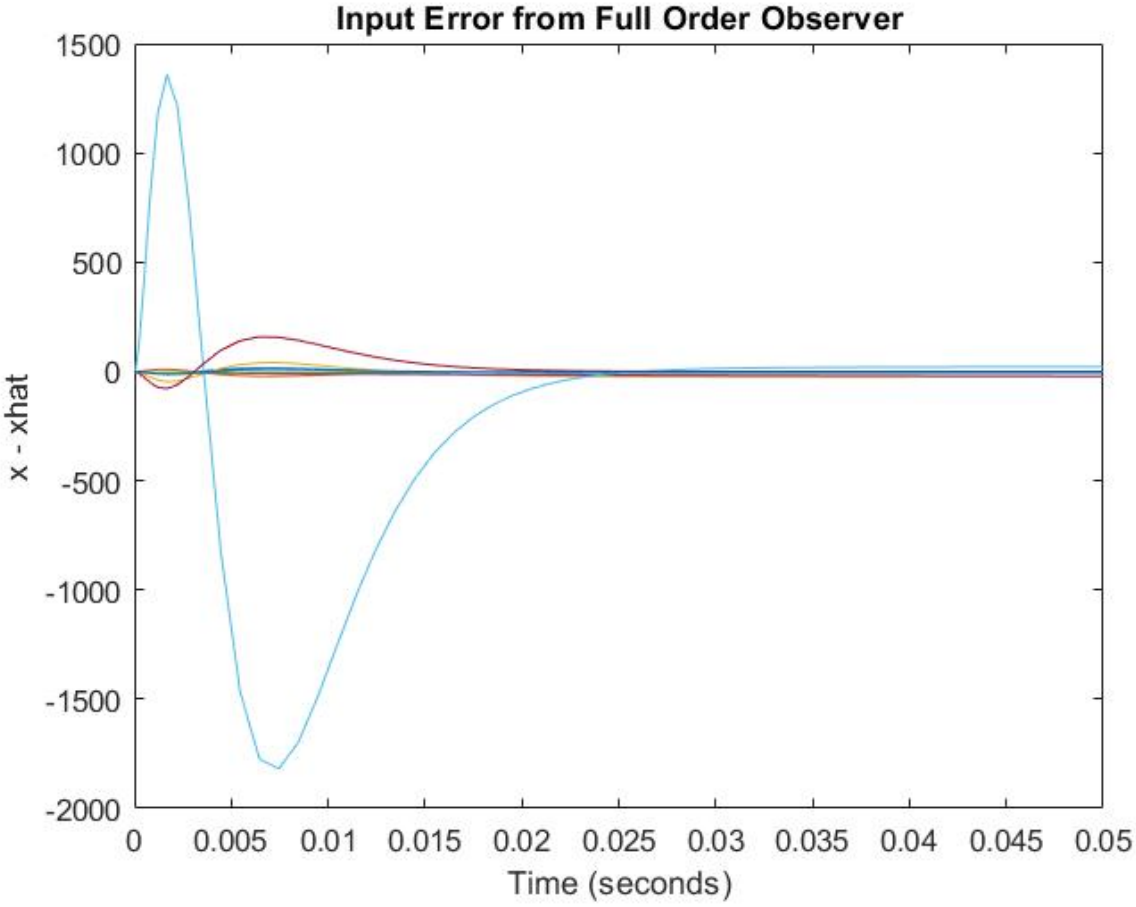
We also calculate a reduced $\hat{x}_{red}(0)$ with the equation:

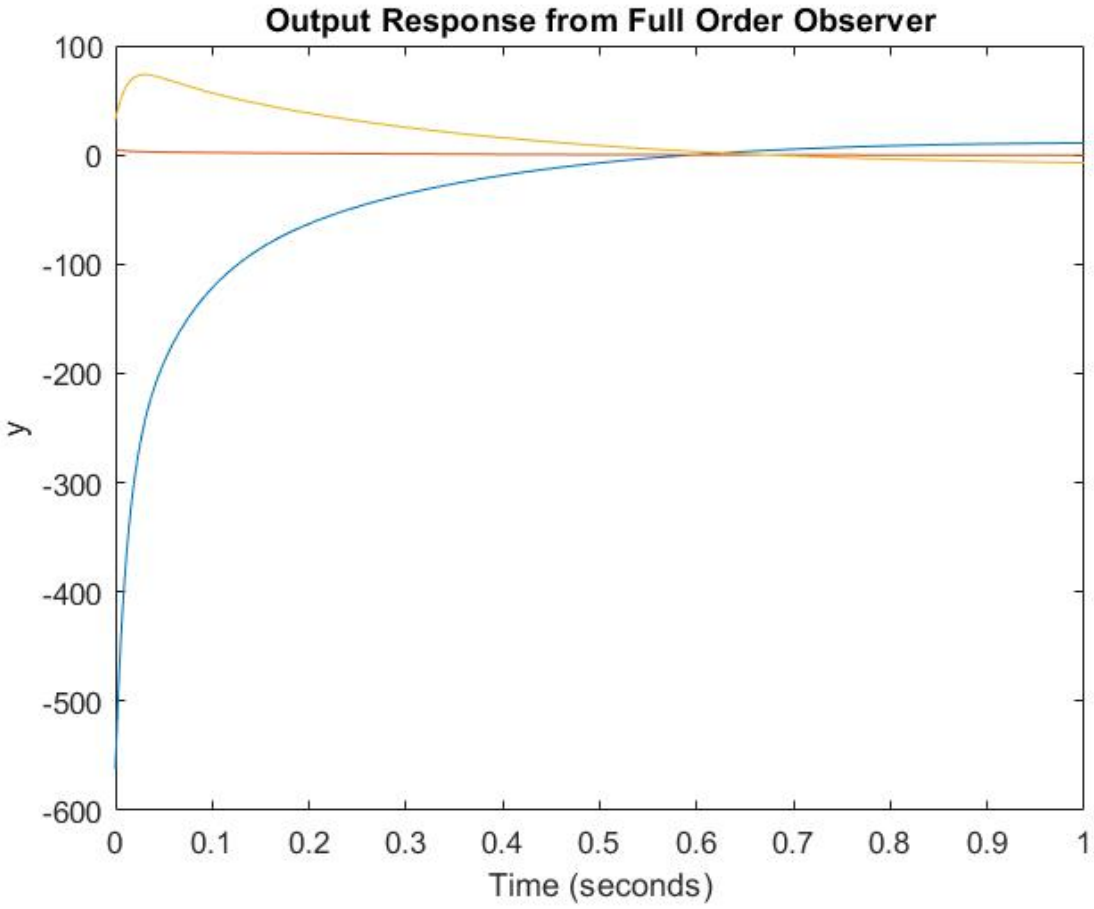
$$\hat{x}_{red}(0) = (L_1^T L_1)^{-1} L_1^T ((C^T C)^{-1} C^T - (L + L_1 K_1)) \times y(0)$$

Full Order Observer Plots



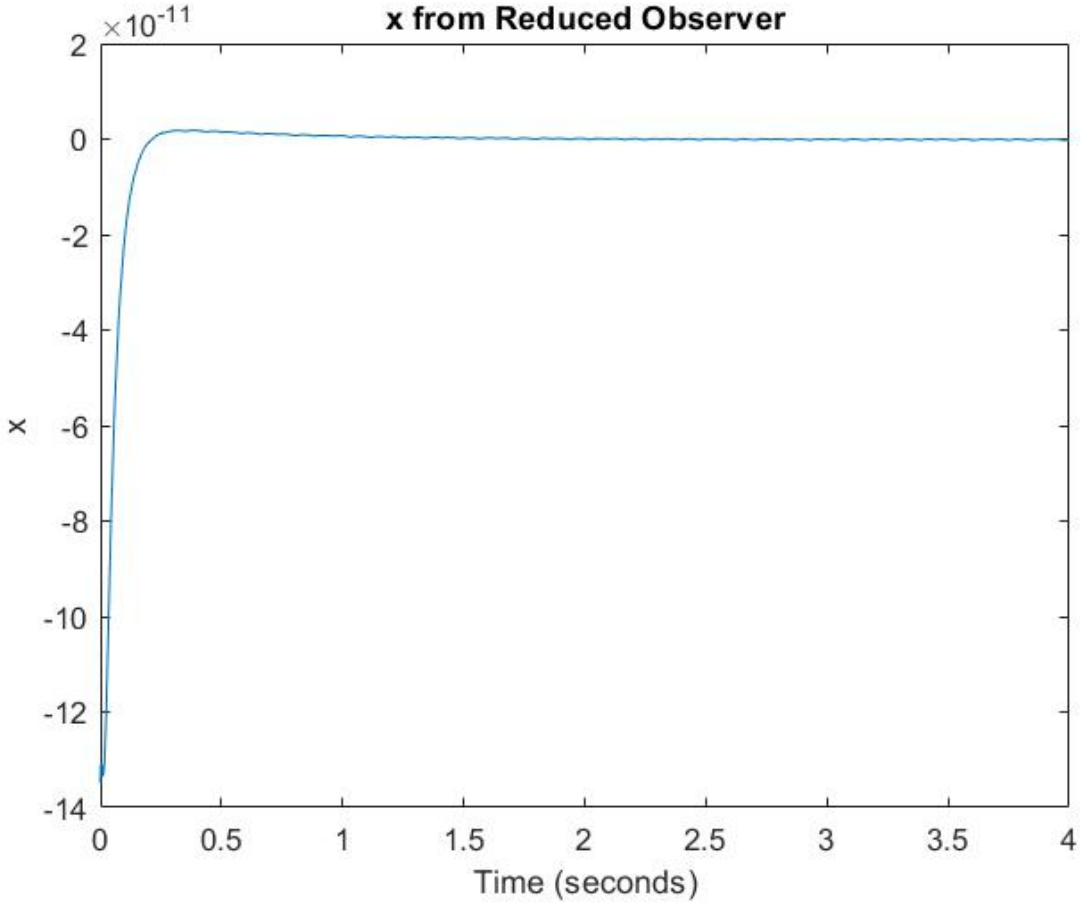

xhat from Full Order Observer
2000
1500
1000
500
0
-500
-1000
-1500
xhat
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
Time (seconds)

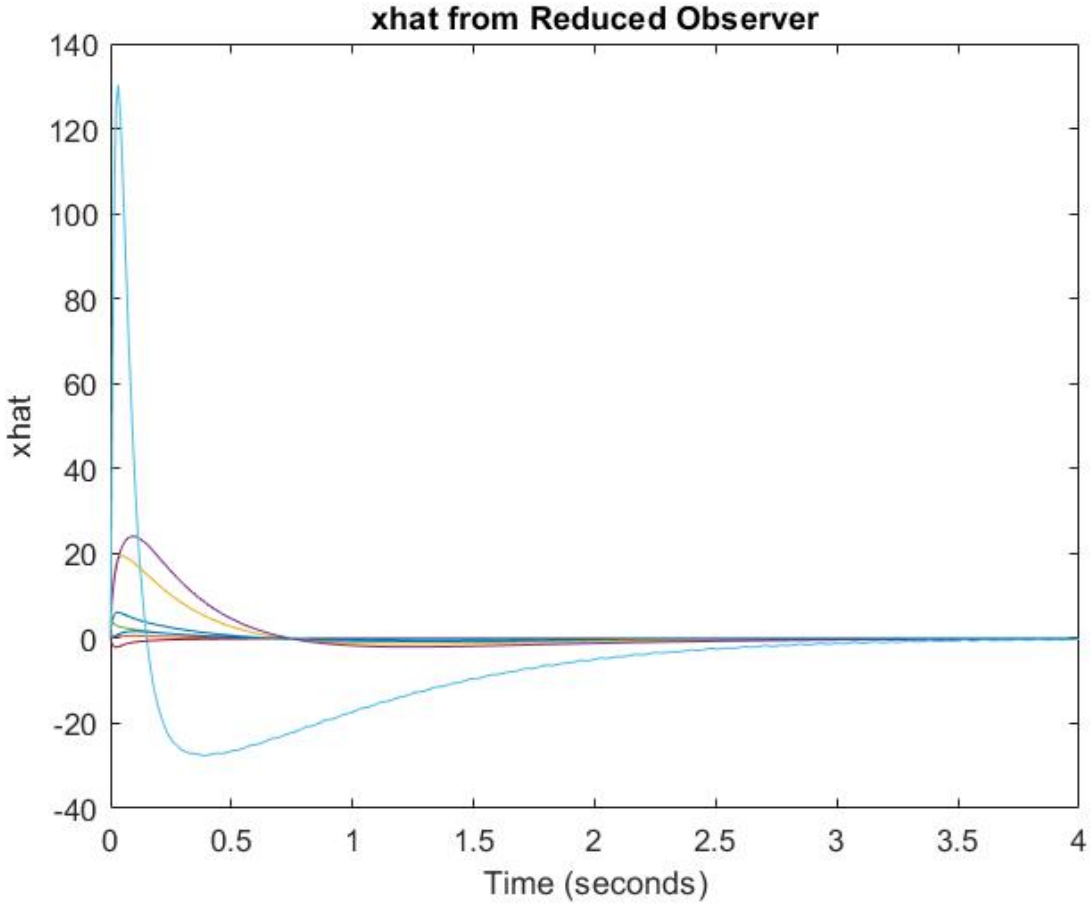

Input Error from Full Order Observer
1500
1000
500
0
-500
-1000
-1500
-2000
x - xhat
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
Time (seconds)

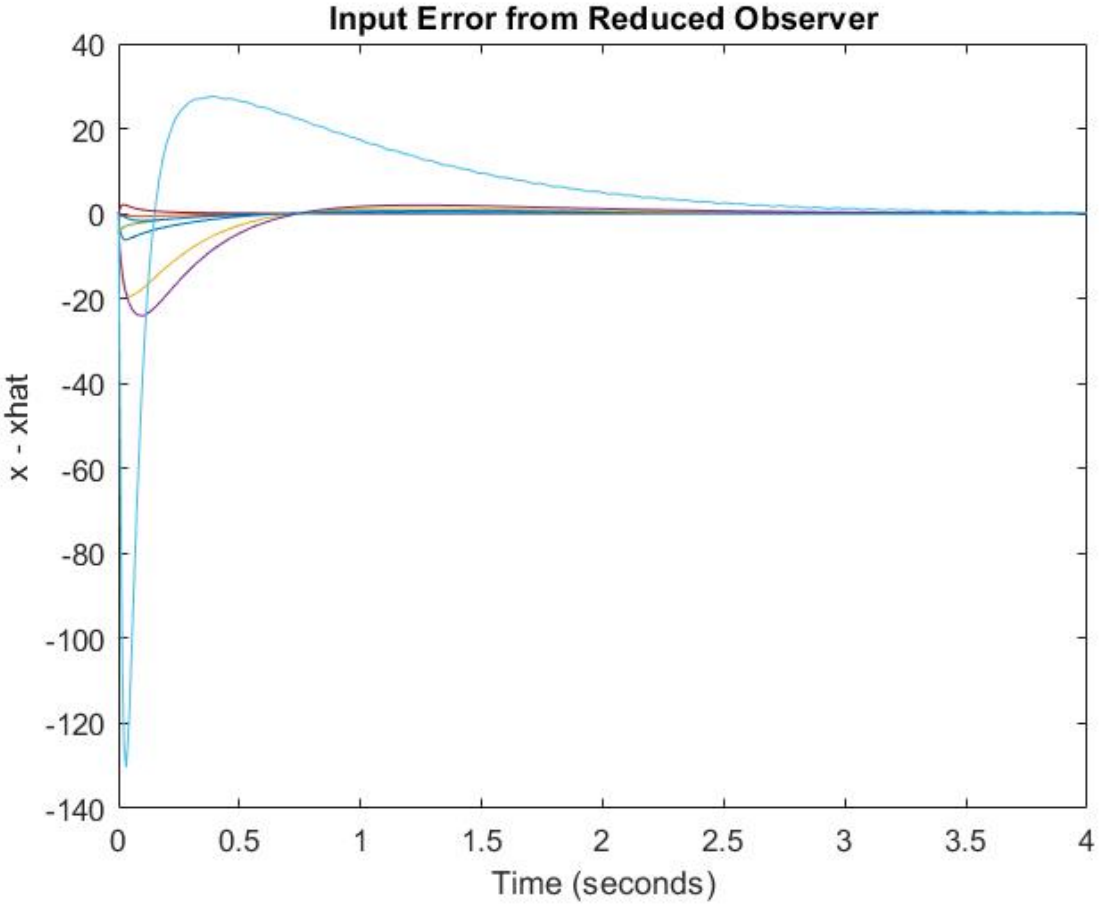

Output Response from Full Order Observer
100
0
-100
-200
-300
-400
-500
-600
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time (seconds)

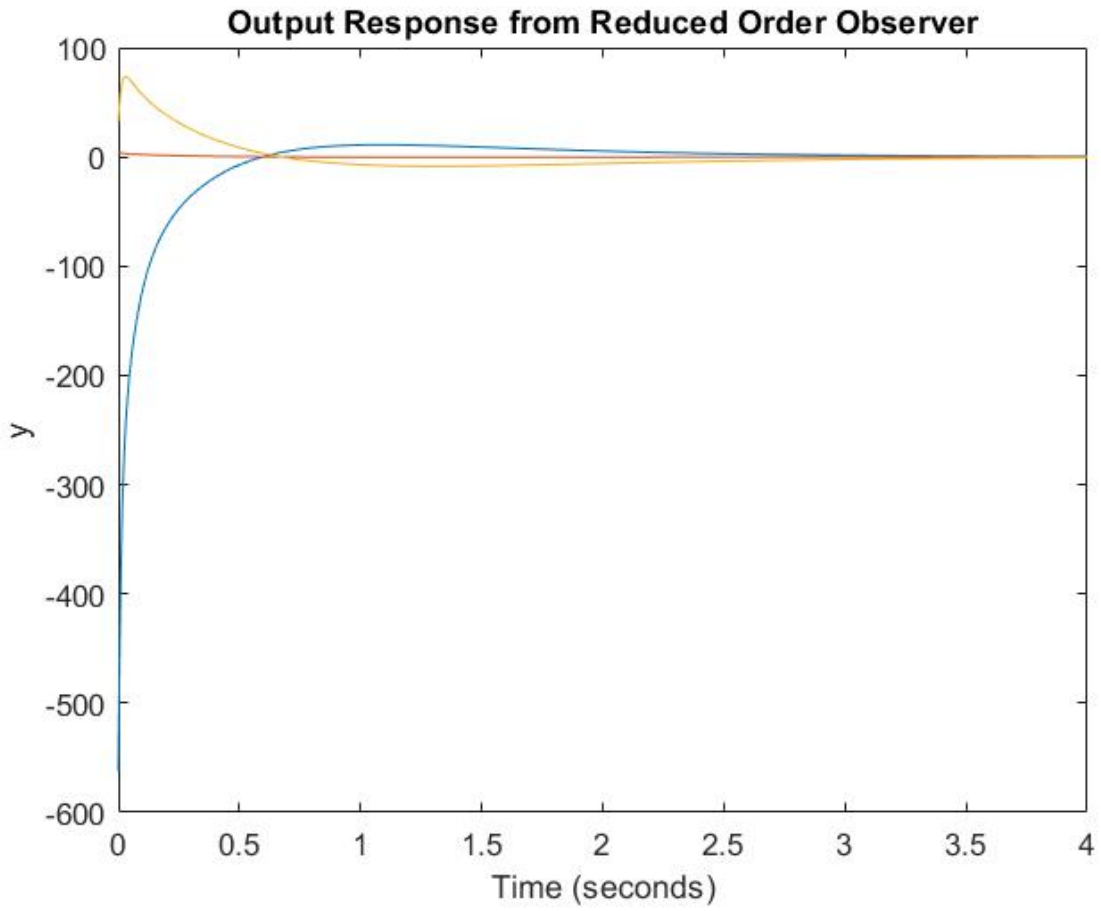
For the full order observer, all states move to 0, with error approaching 0 in $0.04s$. This is fast because the observer poles are 5 times larger than the open loop poles, with the largest pole of the observer at 1000 and the largest pole of the system at 219.

Reduced Order Observer Plots



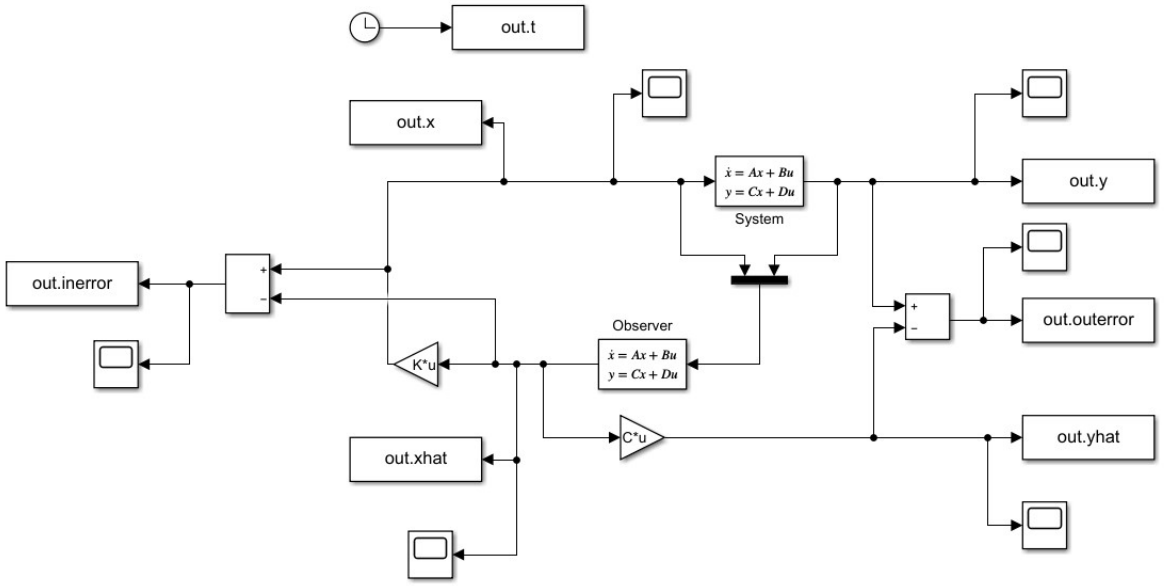

xhat from Reduced Observer
xhat
Time (seconds)
140
120
100
80
60
40
20
0
-20
-40
0
0.5
1
1.5
2
2.5
3
3.5
4


Input Error from Reduced Observer
40
20
0
-20
-40
-60
-80
-100
-120
-140
x - xhat
0
0.5
1
1.5
2
2.5
3
3.5
4
Time (seconds)

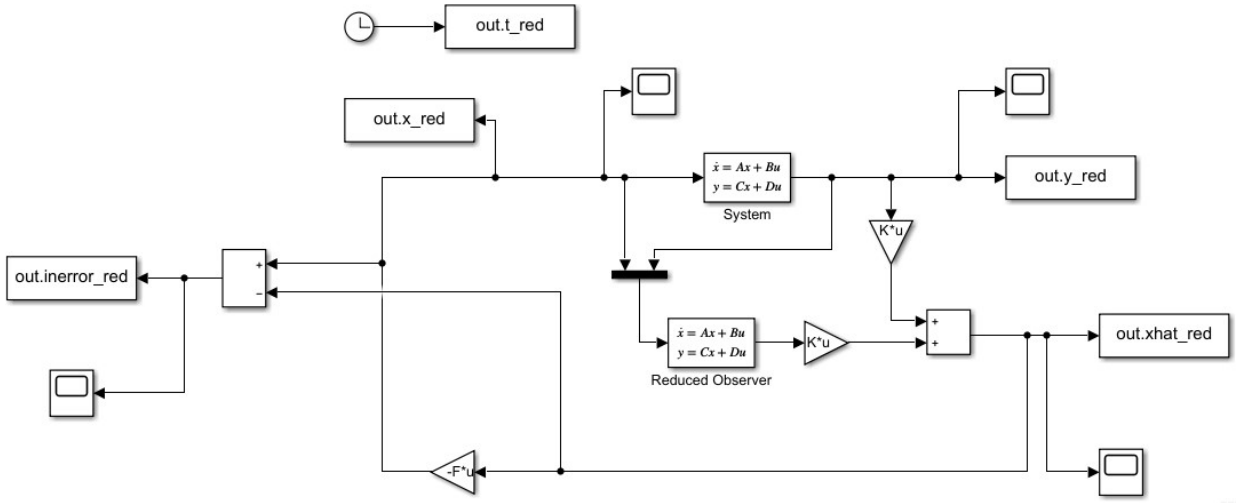


For the reduced order observer, all 8 states move to 0, with error approaching 0 in 4s. This is slower in comparison to the full order observer, because the desired poles used weren't as far from the closed loop poles. This choice reduced potential noise that this observer is more subseptible to in comparison to a full order observer.

Full Order Observer Block Diagram



Reduced Order Observer Block Diagram



Conclusions

In conclusion, both observers are valid choices for observing the stability and behavior of the given 8th order system. The full-order observer can be tuned to use a larger set of poles in comparison to the reduced-order observer and not suffer issues of noise affecting the output and input responses.

Interestingly, the system itself wasn't fully observable due to the rank of $O(A, C)$ being 6, while the reduced order systm was observable with its rank being equal to $r = 5$. With good choices of both feedback and output gains F and K, the full order observer was stable ($A - BF$ and $A - KC$ had negative eigenvalues).

References

[1] N. Nise, *Control Systems Engineering*. Hoboken, NJ: Wiley, 2008

[2] R. Stefani, C. Savant, B. Shahian, and G. Hostetter, *Design of Feedback Control Systems*. Orlando, FL: Saunders College Publishing, 1994.

[3] G. Franklin, J. Powel, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Upper Saddle River, NJ: Prentice Hall, 2002

[4] R. Dorf and R. Bishop, *Modern Control Systems*. Upper Saddle River, NJ: Pearson Education, 2005.

[5] K. Ogata, *Modern Control Engineering*. Upper Saddle River, NJ: Prentice Hall, 2002.

[6] S. Shinners, *Modern Control Systems Theory and Design*. New York: Wiley, 1992.

[7] Z. Gajic and M. Lelic, *Modern Control Systems Engineering*, London, U.K.: Prentice Hall, 1996

Appendix

```
%% Given

A = [-6.30908 0 -10.9544 0 83.74458 0 0 24.05866;
     0 -161.083 0 0 51.52923 0 -18.0261 0;
     -18.7858 0 -46.3136 0 275.6592 0 0 158.3741;
     0 0 0 -17.3506 193.9373 0 0 0;
     1.299576 0 2.969317 0.3977 -38.7024 0.105748 0 0;
     16.64244 0 38.02522 5.066579 -479.384 0 0 0;
     0 -450.386 0 0 142.2084 0 -80.9472 0;
     2.02257 0 4.621237 0 0 0 0 -51.2108];

BT = [0 0 0 3.946683 0 0 0 0];

B = BT';

C = [0 0 0 5.066579 -116.446 0 0 0;
    0 0 0 0 1 0 0 0;
    12.96989 10.32532 -0.56926 0 0 0 0 0];

n = rank(A); % system order
m = rank(B); % # of inputs
c = rank(C); % rank of output
l = 3; % # of outputs

D = zeros(c, m);


%% Full Order Observer

x0 = [1 2 3 4 5 6 7 8]';
y0 = C*x0;
% Least-Squares Estimation of x0hat
x0hat = pinv(C'*C)*C'*y0;

dpoles = eig(A);
```

```
obspoles = linspace(-1000,-300,8);
KT = place(A', C', obspoles);
F = place(A, B, dpoles);
K = KT';

Aobs = A - K*C;
Bobs = [B K];
Cobs = eye(n);
Dobs = zeros(n, m+c);

w = eig(Aobs); % Negative
u = eig(A - B*F); % Negative as well

%% Full Order Observer Plots

% figure(1);
% plot(out.x);
% title("x from Full Order Observer");
% ylabel("x");
%
% figure(2);
% plot(out.xhat);
% title("xhat from Full Order Observer");
% ylabel("xhat");
%
% figure(3);
% plot(out.inerror);
% title("Input Error from Full Order Observer");
% ylabel("x - xhat");
%
% figure(4);
% plot(out.outerror);
% title("Output Error from Full Order Observer");
% ylabel("y - yhat");
%
% figure(5);
% plot(out.y);
% title("Output Response from Full Order Observer");
% ylabel("y");

%% Reduced-Order Observer

r = n - c;
% C*L = Ic, C1*L1=Ir, C*L1 = 0, C1*L=0 -> Lest = C'*C
Lest = C'*C;
C1 = null(Lest)';
Caug = [C;C1];
Laug = inv(Caug);
L = Laug(1:n,1:c);
L1 = Laug(1:n,c+1:n);
O = obsv(C1*A*L1, C*A*L1);
% observable, rank is r = 5
disp(['Rank of Reduced-Order Observer:', num2str(rank(O))]);

lambda_sys = eig(A - B*F);
lambda_red_obs = lambda_sys(1:5); %linspace(-1000,-300,5);
K1T = place((C1*A*L1)', (C*A*L1)', lambda_red_obs);
K1 = K1T';
Aq = C1*A*L1 - K1*C*A*L1;
```

```
Bq = C1*B - K1*C*B;
Kq = C1*A*L1*K1 + C1*A*L - K1*C*A*L - K1*C*A*L1*K1;
% inv(L1'*L1) = eye(5)
x0hat_reduced = eye(5)*L1'*(pinv(C'*C)*C'-(L+L1*K1))*y0;

%% Reduced Order Plots

figure(6);
plot(out.x_red);
title("x from Reduced Observer");
ylabel("x");

figure(7);
plot(out.xhat_red);
title("xhat from Reduced Observer");
ylabel("xhat");

figure(8);
plot(out.inerror_red);
title("Input Error from Reduced Observer");
ylabel("x - xhat");

figure(9);
plot(out.y_red);
title("Output Response from Reduced Order Observer");
ylabel("y");
```