RHp-zeros, LQG, stabilizationn and state estimation

Håkon Dahl-Olsen
NTNU, Trondheim, 20 May 2009

We have a system with two possible measurements and a single input. The transfer functions for these measurements are;

$$g_1 = \frac{1}{-5s+1} \text{ and } g_2 = \frac{-4.8s+1}{5s+1}.$$

This system has a minimal state-space realization

$$\dot{x} = Ax + Bu$$

with

$$A = \begin{bmatrix} 0.2 & 0 \\ 0.5 & -0.2 \end{bmatrix}, B = \begin{bmatrix} -0.2 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ -0.96 & 0.784 \end{bmatrix}, D = 0.$$

The system has open-loop poles at 0.2 and - 0.2 (can read this off the diagonal of A because the system matrix is triagonal).

We consider three cases:

1. We can measure both states directly – excellent performance for load disturbance and process noise.
2. Observer design: because the sensor used to measure x_1 can fail, we design a Kalman filter to estimate x_1 based on measurement of y_2.
3. Sensor failure: feedback control using estimate of x_1.

The control objective is stabilization. We use an LQR controller with minimum input usage; Q=0, R=1. This gives feedback gain of

$$K_{LQR} = \begin{bmatrix} -2 & 0 \end{bmatrix}.$$

Case 1: Full state measurement

The resulting controller is a proportional controller for y_1 with set point $y_1 = 0$ and controller gain $K_c = -2$. The closed-loop poles are now both located at – 0.2 and the system is closed-loop stable. A unit step added to u at time $t = 10$ and a unit step is added to x_1 at time $t = 40$. The state responses are shown in Figure 1 and the measurements and input usage are shown in Figure 2.

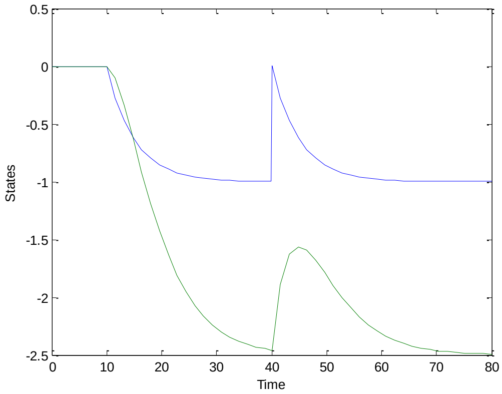

Figure 1: Simulation case (A): blue line is x_1, which is the only state used for feedback control. The minimum input controller clearly stabilizes the solution.

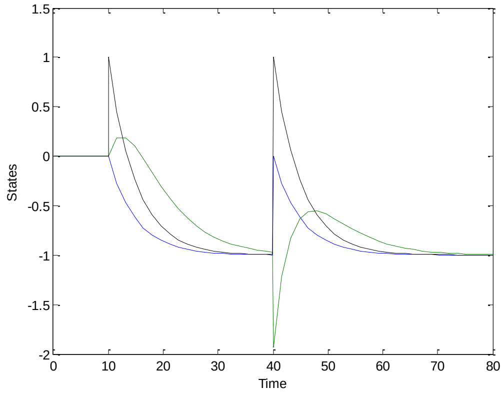

Figure 2: Simulation case (A): blue line is y_1, which is the only state used for feedback control. The green line is y_2; observe the inverse response resulting from the RHP zero in g_2. The black line is the input usage (u).

Case 2: Build an estimator for y_1 based on y_2

We replace the C-matrix above with its second row-only to disallow measurement of y_1. Further, we assume no load disturbances, but significant process noise in the system. Based on this we design a Kalman filter using the **kalman** function in the Matlab control systems toolbox:

```
[Kfilter,ObserverGain,RiccatiMatrix] = kalman(ss(A,B,C,D),1,0.1);
```

The resulting gain matrix is

$$\text{ObserverGain} = 10^4 \times \begin{bmatrix} 0.98 \\ 1.23 \end{bmatrix}.$$

A simulation of the system in closed-loop, but still using direct measurement of y_1 yields the following observer estimates; states are shown in Figure 3 and the measured output in Figure 4. In spite of the big error when the step disturbance in the state occurs, we will try this in closed-loop because we may not have too many good options here if the sensor for y_1 fails.

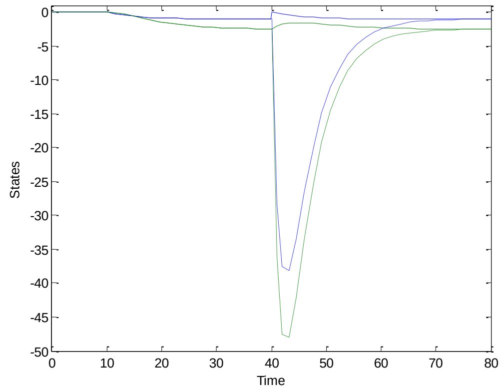

Figure 3: Simulation case (B): solid lines show true state values, whereas dashed lines are estimates. We see that the state estimates do not respond well to process noise.

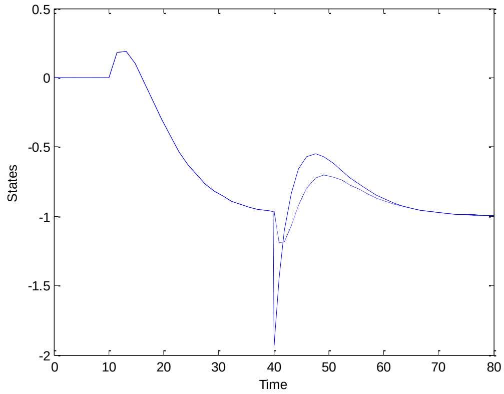

Figure 4: Simulation case (B): even though state estimates are bad, the situation looks good in the output. This is what is visible *online*, which confirms the importance of dynamic simulation.

Case 3: Sensor failure

We now test closed-loop behavior when applying the estimator for y_1. The results are shown in Figure 5 (states) and Figure 6 (output, input). Note the excessive input usage, compare with Figure 2! If the input

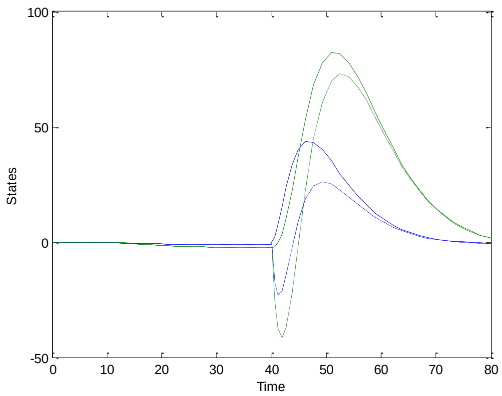

Figure 5: Simulation case (C): solid lines show true state values, whereas dashed lines are estimates. Although the system is stable and the observer seems to work, look at the excessive overshoot; the closed-loop performance is obviously bad.

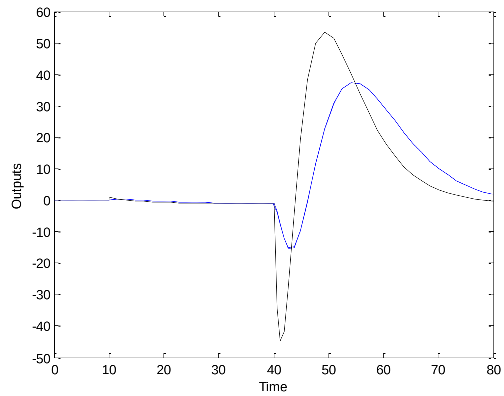

Figure 6: Simulation case (C): the observer fits the output perfectly here. Note the black line (input usage); input usage is excessive!

had been limited, the system would have been closed-loop unstable. To illustrate we simulate the system but with a saturation on the input signal;

$$-5 \le u(t) \le 5.$$

The resulting trajectories are shown in Figure 7 and Figure 8.

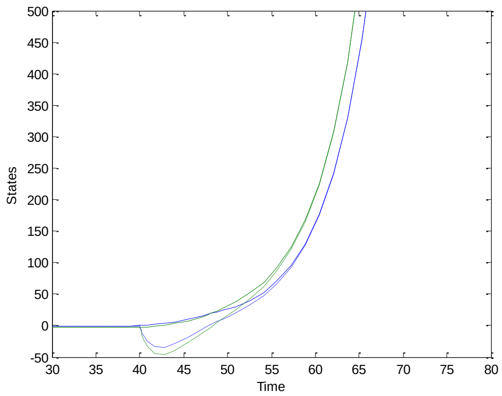

Figure 7: Saturation on *u* gives unstable closed-loop behavior. System blows up on the state disturbance.

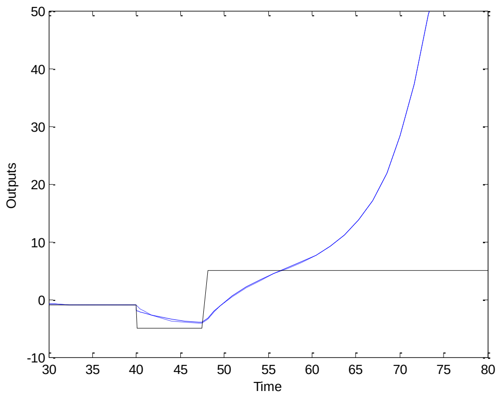

Figure 8: Input and output when system blows up (blue: output, black: input).

The lesson learned from this example is that non-minimum phase behavior can create great difficulties for stabilization. Just because a system is state-observable does not mean that estimating more well-conditioned measurements that for some reason are not available from the measurements we do have, it does not mean it is a good idea to do so, at least not if there are fundamental limitations in the input-output behavior of the available measurements.

Computer simulations

The computer simulations used to generate these plots were done using Matlab/Simulink. There are four Simulink files, corresponding to the four simulations above:

- caseA.mdl: Case 1
- caseB.mdl: Case 2
- caseC.mdl: Case 3
- CaseCb.mdl: Case 3 with input saturation

Before running these files, run the script file **get_lqg.m**
which contains the following:

```
%Define dynamics
A = [0.2 0; 0.5 -0.2];    % open-loop A-matrix has poles in 0.2 and -0.2
B = [-0.2; 0];
C = [1 0; -0.96 0.784];
D = 0;
G=ss(A,B,C,D);

%For Case 2: Kill one measurement
C2=C(2,:);
Gred=ss(A,B,C2,D);

%Create LQR controller
Q=zeros(2); R=1;    %Q=0 gives minimum input usage
Klqr=lqr(A,B,Q,R);

%Set disturbance signal parameters (for use in Simulink)
loadD=1; stateD=1; LTime=10; STime=40;

%Create Kalman filter based on (A,B,C2):
[Kest1,L1,P1]=kalman(Gred,1,0.1);

%Simulate each case for tf=80.
%Simulation variables are stored in workspace, with naming convention
%   VariableName_CASE_#
%To avoid opening simulink for each case, use the syntax
%   sim('caseA',80)
```