

1 Aim

1. To revise and/or acquire and develop skills in using MATLAB simulations.
2. To gain some experience with Kalman filters and LQG design.

2 Noise Reduction

In this section we return to a question posed in CLAB1, namely the removal of WGN from a sinusoidal signal. The signal x_n is defined by

$$x_n = \sin(nT_s\omega_0)$$

where the sample time $T_s = 0.015$ sec and the frequency is $\omega_0 = 7$ rad/sec. The noise v_n is WGN with mean 0 and variance σ_v^2 , where $\sigma_v = 0.2$.

The signal y_n that is measured is given by

$$y_n = x_n + v_n$$

As we saw in CLAB1, the spectra of x_n and v_n are not disjoint, and indeed overlap. Using Kalman filtering, it is possible to reduce the noise provided a suitable model for the signal x_n is used. Signal modelling is in general a complex matter, and the quality of the noise removal depends critically on the suitability of the model. Indeed, modelling of speech and other audio is nontrivial and the subject of substantial work. However, in our case the signal is quite simple, and one way to model it is as follows.

Consider the transfer function

$$\mathcal{H}(z) = \frac{(\sin \omega_0)z^{-1}}{1 - 2(\cos \omega_0)z^{-1} + z^{-2}}$$

QUESTION 2.1 1. Find the poles of the transfer function $\mathcal{H}(z)$.

2. Why might $\mathcal{H}(z)$ be useful in building a signal model for x_n ?
3. Find a 2-d state space realisation (A_0, B_0, C_0) of $\mathcal{H}(z)$.
4. Check that the eigenvalues of A_0 equal the poles of $\mathcal{H}(z)$.
5. Check reachability and observability.

Your results from Question 2.1 can now be used to determine the signal model for x_n (the subscript “sm” means “signal model”):

$$\begin{aligned} x_{sm,n+1} &= A_{sm}x_{sm,n} + B_{sm}u_n + G_{sm}w_n \\ y_n &= C_{sm}x_{sm,n} + D_{sm}u_n + H_{sm}w_n + v_n \end{aligned} \tag{1}$$

where we have used the format used in MATLAB and where we select

$$\begin{aligned} A_{sm} &= A_0 \\ B_{sm} &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ G_{sm} &= B_0 \\ C_{sm} &= C_0 \\ D_{sm} &= 0 \\ H_{sm} &= 0 \\ Qn &= \sigma_w^2 \\ Rn &= \sigma_v^2 \end{aligned}$$

We are modelling x_n as $C_{sm}x_{sm,n}$.

The MATLAB commands

```
sm1=ss(Asm, [Bsm Gsm], Csm, [Dsm Hsm], Ts);
[kf1, L, Y, M] =kalman(sm1, Qn, Rn);
```

produce the following form of the steady state Kalman filter **kf1**:

$$\begin{pmatrix} \hat{x}_{sm,n+1|n} \\ \hat{y}_{n|n} \\ \hat{x}_{sm,n|n} \end{pmatrix} = \begin{pmatrix} A_{sm}(I - MC_{sm}) \\ C_{sm}(I - MC_{sm}) \\ I - MC_{sm} \end{pmatrix} \hat{x}_{sm,n|n-1} + \begin{pmatrix} (I - C_{sm}M)D_{sm} & C_{sm}M \\ -MD_{sm} & M \end{pmatrix} \begin{pmatrix} u_n \\ y_n \end{pmatrix} \quad (2)$$

The matrix Y is the solution of the filter ARE, and the gains L and M are determined from Y .

The output of **kf1** has several components, and we need simply the first component:

$$\hat{x}_{n|n} = \hat{y}_{n|n}$$

for our optimal estimate $\hat{x}_{n|n}$ of x_n . Therefore we use the Kalman filter in the form **kf2**:

$$\begin{aligned} \hat{x}_{sm,n+1|n} &= A_{sm}\hat{x}_{sm,n|n-1} + L(y_n - C_{sm}\hat{x}_{sm,n|n-1}) \\ \hat{x}_{n|n} &= C_{sm}(I - MC_{sm})\hat{x}_{sm,n|n-1} + C_{sm}My_n \end{aligned} \quad (3)$$

where we have used the facts that $B_{sm} = 0$ and $D_{sm} = 0$. The Kalman filter **kf2** can be constructed in MATLAB as follows:

```
Akf2 = Asm-L*Csm;
Bkf=L;
Ckf=Csm*(eye(2)-M*Csm);
Dkf=Csm*M;
kf2=ss(Akf2,Bkf2,Ckf2,Dkf2,Ts);
```

MATLAB EXERCISE 2.2

sigpwgn-kf1.m

1. Read the MATLAB help about the simulation command `lsim` and the Kalman filter command `kalman`.
2. Download the MATLAB m-file `sigpwgn-kf1.m` and save into your working MATLAB folder. Open with the MATLAB editor. You can enter your MATLAB code at the end of this file. The first part of the file sets up the signal plus noise signals.

3. Create your signal model `sm1` as described above, at the end of the m-file.
4. Use the MATLAB simulation command `lsim` to check that your signal model produces the desired signal x_n . Use zero input and a nonzero initial condition:

```
lsim(sm1,zerosig,1)
```

(you will need to create `zerosig`). Compare the signal graphs produced by the earlier part of the m-file and by `lsim`.

5. Add code to the m-file to create the Kalman filters `kf1` and `kf2` as described above.
6. You will need to select a suitable value(s) for σ_w . Start with $\sigma_w = 1$, and then try to see what happens when it is decreased to, say, $\sigma_w = 0.01$, in the simulations to follow.
7. Use `lsim` to filter the signal y_n to produce the filtered estimate $\hat{x}_{n|n}$ (`xf` in the following code):

```
[xf,t,xsmf] = lsim(kf2,y,t);
```

8. Plot graphs of the filtered estimate in the time and frequency domains. Compare with the original signals x_n and y_n . Repeat, using decreasing values for σ_w .
9. Plot the Bode diagram for your Kalman filter `kf2`. What type of filter is it? Obtain plots for decreasing values of σ_w . What do you notice?
10. Comment on the quality of the noise removal. What factors affect the quality?

3 LQG Control

In this section we use LQG methods to design an stabilising output feedback controller for a noisy version of the inverted pendulum model:

$$\begin{aligned} \begin{pmatrix} x_{n+1}^1 \\ x_{n+1}^2 \end{pmatrix} &= \begin{pmatrix} 1 & T_s \\ T_s g & 1 \end{pmatrix} \begin{pmatrix} x_n^1 \\ x_n^2 \end{pmatrix} + \begin{pmatrix} 0 \\ T_s/m \end{pmatrix} u_n + \begin{pmatrix} 1 \\ 0 \end{pmatrix} w_n \\ y_n &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_n^1 \\ x_n^2 \end{pmatrix} + v_n \end{aligned} \quad (4)$$

Use the following parameters:

$$g = 10, \quad m = 0.01, \quad T_s = 0.01.$$

Select your own values of Q and R for LQR design, and your own values Q_n and R_n for your Kalman filter design.

MATLAB EXERCISE 3.1

`pend-output-fb-kf-step.mdl`

1. Download and open the SIMULINK file `pend-output-fb-kf-step.mdl`.
2. Use the command

```
[K,X,E] = dlqr(A,B,Q,R)
```

to determine the feedback gains, the solution of the ARE, and the closed loop eigenvalues.

3. Check directly the eigenvalues of the closed loop matrix

$$A - BK$$

4. Use the command

```
[kf1,L,Y,M] = kalman(pendmod,Qn,Rn)
```

(you will need to create `pendmod`) to determine the filter gain L and the solution to the filter ARE Y .

5. Check directly the eigenvalues of the filter error matrix

$$A - LC$$

6. Enter the gain K into the SIMULINK model, and enter the Kalman filter (as you did for the observer of CLAB3).
 7. Simulate, and observe the (noisy) step response. Describe what you see.
-