E102 Midterm Project

Kyle Lund, Jesse Joseph, and Josh Sealand

April 11, 2016

Introduction

Analysis of the Plant

The plant that we are trying to control is the circuit depicted in figure 1. This is a simple single input, single output system.

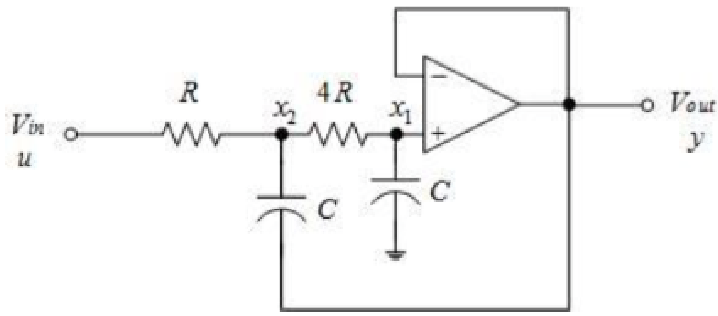


Figure 1: The circuit schematic for our plant. $C = 10\mu F$ and $R = 50k\Omega$

Immediately, we notice that $y = x_1$, because the $+$ and $-$ terminals of the op-amp must be at approximately equal potential. This means that our output equation is:

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{1}$$

We can use Kirchhoff's Current Law at the two nodes x_1 and x_2 to determine the rest of the state space equations for this system.

$$x_1 : \quad \frac{1}{4R}(x_1 - x_2) + C\dot{x}_1 = 0 \tag{2}$$

$$x_2 : \quad \frac{1}{R}(x_2 - u) + \frac{1}{4R}(x_2 - x_1) + C(\dot{x}_2 - \dot{x}_1) = 0 \tag{3}$$

Now if we solve equation 2 for $\dot{x}_1$, we get:

$$\dot{x}_1 = \frac{1}{4RC}(-x_1 + x_2) \tag{4}$$

Solving equation 3 for $\dot{x}_2$ (and plugging in the result above) we get:

$$\dot{x}_2 = \dot{x}_1 + \frac{1}{4RC}(x_1 - x_2) + \frac{1}{RC}(-x_2 + u) = \frac{1}{RC}(-x_2 + u) \tag{5}$$

Putting these equations in matrix form, we obtain:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4RC} & -\frac{1}{4RC} \\ 0 & -\frac{1}{RC} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{RC} \end{bmatrix} u \tag{6}$$

For our system, $\frac{1}{RC} = \frac{1}{50k\Omega * 10\mu F} = 2\text{Hz}$. Using this, our equations become:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} u \tag{7}$$

Designing the Controller

Discrete-Time Representation

To design a digital control system, we need the equivalent discrete-time state space representation of the plant. Matlab can generate this for us with the command `c2d`. The Matlab commands used to do this conversion and their outputs are included below:

```
>> A = [-.5 .5; 0 -2]
>> B = [0; 2]
>> C = [1 0]
>> D = 0;
>> Ts = 0.1;
>> dt_sys = c2d(ss(A,B,C,D),Ts)

dt_sys =

  a =
             x1       x2
   x1    0.9512  0.04417
   x2         0   0.8187

  b =
              u1
   x1  0.004604
   x2    0.1813

  c =
       x1  x2
   y1   1   0

  d =
       u1
   y1   0

Sample time: 0.1 seconds
Discrete-time state-space model.
```

Using this, our discrete-time plant is:

$$\begin{aligned} \mathbf{x}[n+1] &= \begin{bmatrix} 0.951 & 0.0442 \\ 0 & 0.819 \end{bmatrix} \mathbf{x}[n] + \begin{bmatrix} 0.00460 \\ 0.181 \end{bmatrix} u[n] \\ y[n] &= \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}[n] \end{aligned}$$

Full State Feedback

Now that we have a discrete-time state space representation of our plant, we can design a full state feedback controller *in discrete time* to control our system.
To do this, we will use optimal control and minimize the objective function

$$J = \frac{1}{2} \sum_{n=0}^{\infty} \left(20x_1[n]^2 + x_2[n]^2 + 4u[n]^2 \right)$$

This is just a standard quadratic optimization of the form $J = \frac{1}{2}\sum_{n=0}^{\infty}\left(\mathbf{x}[n]^T\mathbf{Q}\mathbf{x}[n] + \mathbf{u}[n]^T\mathbf{R}\mathbf{u}[n]\right)$ where $\mathbf{Q} = \begin{bmatrix} 20 & 0 \\ 0 & 1 \end{bmatrix}$ and $\mathbf{R} = 4$. We can use the Matlab command `dlqr` to solve this optimization problem as follows:

```
>> Ad = dt_sys.a;
>> Bd = dt_sys.b;
>> Q = [20 0; 0 1];
>> R = 4;
>> [K S P_K] = dlqr(Ad,Bd,Q,R)

K =

    1.0821    0.3319


S =

  157.3793   22.9251
   22.9251    7.5598


P_K =

    0.8314
    0.8734
```

From this output, we can see that our feedback gains are:

$$\mathbf{K} = \begin{bmatrix} 1.0821 & 0.3319 \end{bmatrix}$$

And our system poles are:

$$z_1 = 0.8314$$
$$z_2 = 0.8734$$

Observer

In practice, we will not have direct access to the full state of our system. Instead, we will need to approximate the state of the system with an observer of the form:

$$\hat{\mathbf{x}}[n+1] = \mathbf{A}_d\hat{\mathbf{x}}[n] + \mathbf{B}_d u[n] + \mathbf{L}(y[n] - \hat{y[n]})$$
$$\hat{y}[n] = \mathbf{C}\hat{\mathbf{x}}[n]$$

Matlab can design this controller for us (by selecting $\mathbf{L}$) if we provide desired poles for the observer. We need the observer to respond faster than the plant, so that we will be able to estimate the state of the system in real time. To do this, we divide the values of the system poles by five to get the pole locations:

$$z_1 = 0.1663$$
$$z_2 = 0.1747$$

Now, we can use Matlab to determine $\mathbf{L}$:

```
>> P_L = P_K/5;
>> L = acker(Ad',C',P_L).'

L =

    1.4290
    9.5143
```

Therefore, our observer gains are:

$$\mathbf{L} = \begin{bmatrix} 1.4290 \\ 9.5143 \end{bmatrix}$$

Reference Gain

To have a usable system, we also need a reference gain K_r that will be multiplied by the input. We have the following formula for that reference gain in order to have zero steady-state error:

$$K_r = -\left[(\mathbf{C} - \mathbf{D}\mathbf{K})(\mathbf{A}_\mathrm{d} - \mathbf{I} - \mathbf{B}_\mathrm{d}\mathbf{K})^{-1}\mathbf{B}_\mathrm{d} + \mathbf{D}\right]^{-1}$$

We can use Matlab to calculate this for us:

```
>> I = eye(2);
>> Kr = -inv( (C-D*K)*inv(Ad-I-Bd*K)*Bd + D )

Kr =

    2.4140
```

Therefore, our reference gain is:

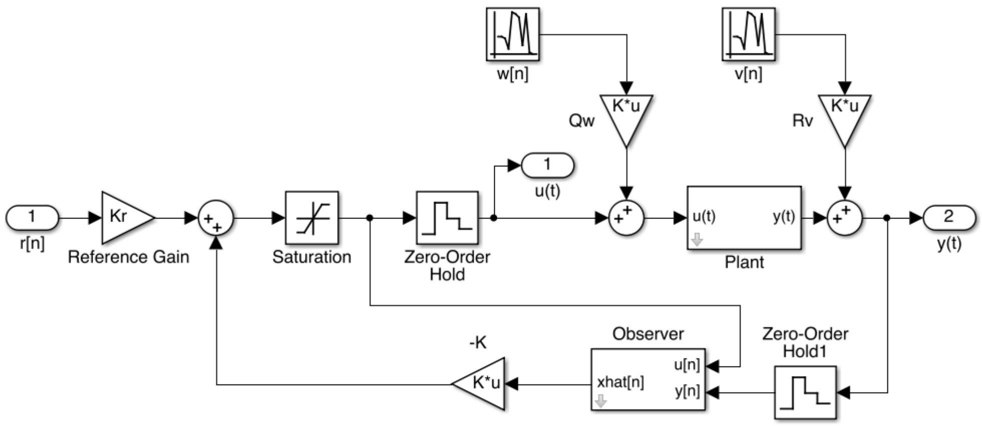
$$K_r = 2.4140$$

Simulation

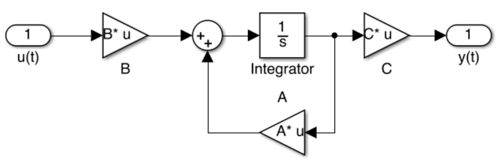
To confirm that our design will meet the given specifications, we simulated the step response of the system in Simulink. Our Simulink model is included as Figure 2.

The results of the Simulation are summarized in Figure 3. The system meets the given specifications of zero overshoot, zero steady state error, and a 2% settling time under 4s. The total control input is also relatively low, as desired.

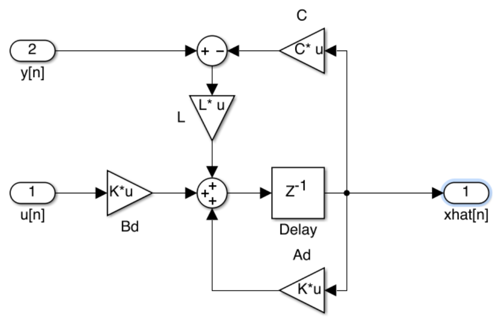
Results



(a) The high level Simulink model for our control system



(b) The Simulink model of our plant



(c) The Simulink model of our observer

Figure 2: Our Simulink model

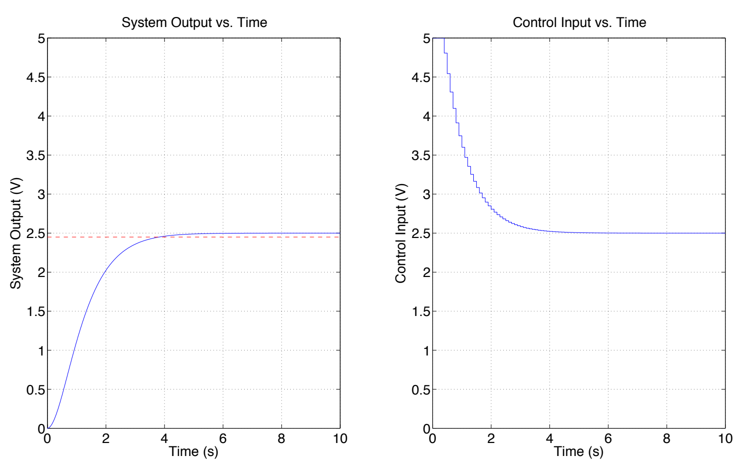


Figure 3: The results of our simulation