E102 Midterm Project

Kyle Lund, Jesse Joseph, and Josh Sealand

April 14, 2016

Introduction

The goal of this project is to design a control system for a simple circuit to meet design specifications for the step response. The design specifications that we were given are:

- zero overshoot
- zero steady state error
- minimize the control input while keeping the 2% settling time under 4s.
- the sample rate is 10 Hz

We will design an observer based controller with full-state feedback to meet these specifications.

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.

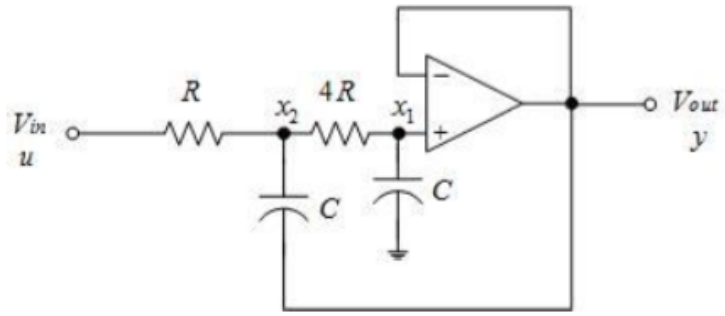


Figure 1: The circuit schematic for our plant. $C = 10\mu F$ and $R = 50k\Omega$

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, $\frac{1}{RC} = \frac{1}{50k\Omega * 10\mu F} = 2$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:

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

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

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

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 $\mathbf{R} = 2$. These values were chosen by trial and error in the simulation. We found that by using Q = I, the system was much too slow, so we chose to increase the relative weight of Q compared to R, the weight on the control input. 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 = 2;
>> [K S P_K] = dlqr(Ad,Bd,Q,R)

K =

    1.6938    0.5148


S =

    142.8152   18.3179
    18.3179    6.1006


P_K =


    0.8344 + 0.0308i
    0.8344 - 0.0308i
```

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

$$\mathbf{K} = \begin{bmatrix}1.6938 & 0.5148\end{bmatrix}$$

And our system poles are:

$$z_1 = 0.8344 + 0.0308i$$
$$z_2 = 0.8344 - 0.0308i$$

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:

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

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:

$$\begin{aligned}z_1 &= 0.1669 + 0.0062i \\ z_2 &= 0.1669 - 0.0062i\end{aligned}$$

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

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

L =

    1.4362
    9.6214
```

Therefore, our observer gains are:

$$\mathbf{L} = \begin{bmatrix}1.4362 \\ 9.6214\end{bmatrix}$$

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:

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

We can use Matlab to calculate this for us:

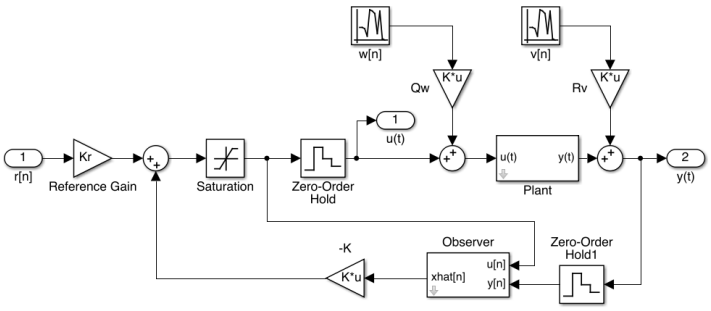
```
>> I = eye(2);
>> Kr = -inv( (C-D*K)*inv(Ad-I-Bd*K)*Bd + D )

Kr =

    3.2086
```

Therefore, our reference gain is:

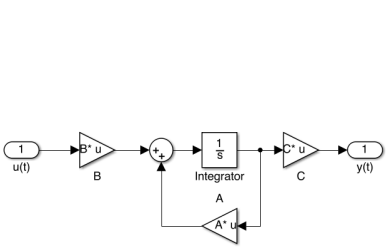
$$K_r = 2.4140$$

Simulation

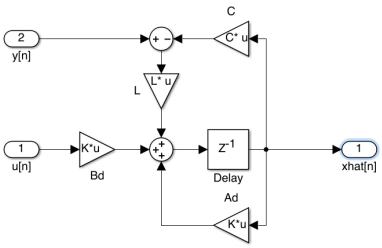
To confirm that our design will meet the given specifications, we simulated the step response of the system in Simulink. Our Simulink model is included as Figure 2.



(a) The high level model for our control system



(b) The model of our plant



(c) The model of our observer

Figure 2: Our Simulink model

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.

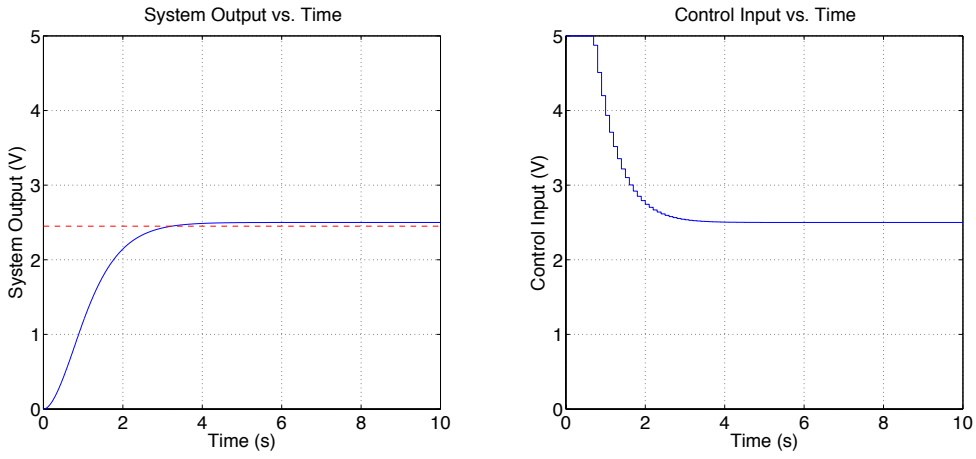


Figure 3: The results of our simulation

Results

We implemented the system using the given circuit schematic. The circuit is depicted in Figure 4.

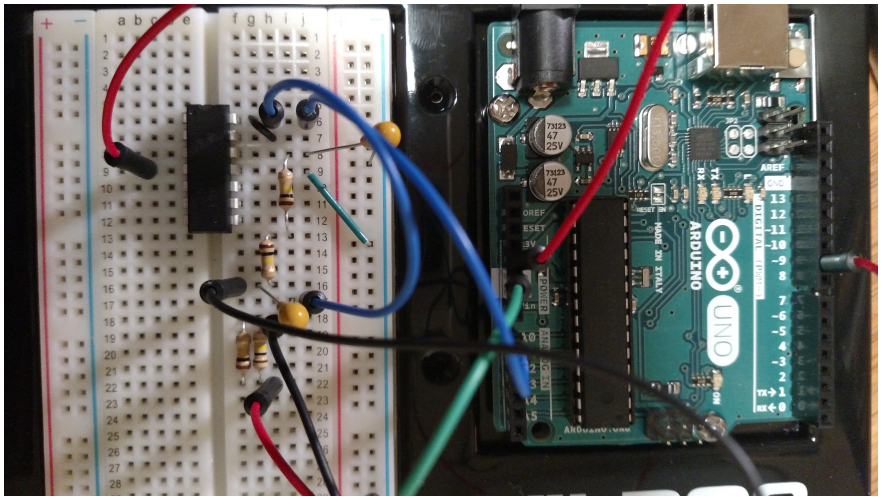


Figure 4: Implementation of the plant being controlled

We implemented the controller on the Arduino (See appendix A for the code), and measured the step response. The results are summarized in Figure 5.

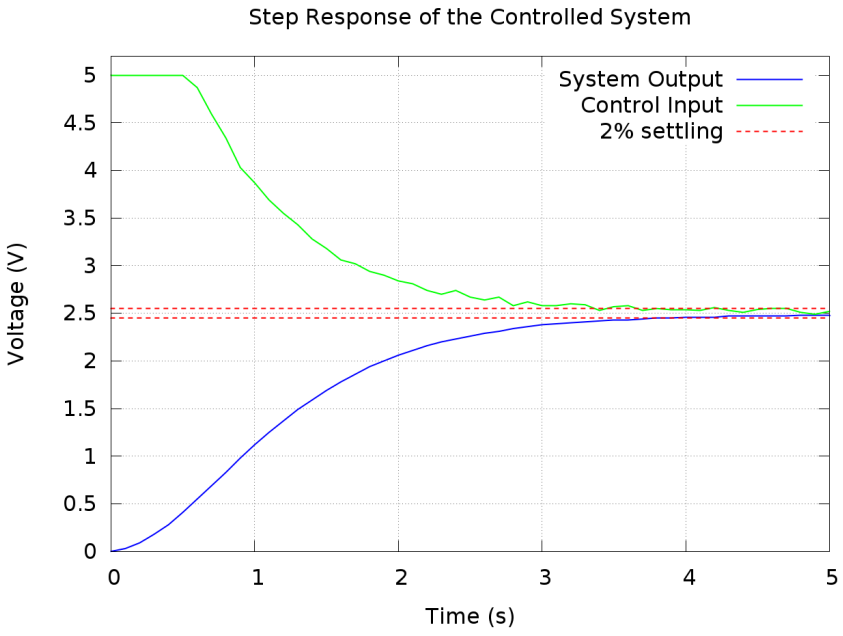


Figure 5: The step response of our controlled system

We found that the actual system achieves a 2% settling time under 4s, as desired. Additionally, the system produces no overshoot and has zero steady state error. The control input is saturated at the start of the step response as expected, because the error is large. We found that our controlled system was slightly slower than the simulation, most likely due to parameter uncertainty in the resistors and capacitors.

Appendix

A. The Arduino code used to implement our controller:

```
#define inputPin A0
#define outputPin 9
#define sample_rate 10 //sample rate, in Hz

float yn = 0;
float un = 0;
float rn = 0;
float xhat[2] = {0,0};
float yhat = 0;

// These values are determined from the simulation
float Ad[2][2] = { {0.9512, 0.0442}, {0.0, 0.8187} };
float Bd[2]    =   {0.0046, 0.1813};
float C[2]     =   {1.0, 0};
float L[2]     =   {1.4362, 9.6214};
float Kr       =    3.2086;
float K[2]     =   {1.6938, 0.5148};

unsigned long loop_start_time;
unsigned long loop_time;
unsigned long start_time;
unsigned int loopNum = 0;

void setup(){
  Serial.begin(9600);
  loop_time = 1000000/sample_rate;
  start_time = micros();
  Serial.println("beginning");
  delay(5000);
  rn = 2.5;
}

void loop(){
  loop_start_time = micros();

  //read the current value from the input
  yn = voltage(analogRead(inputPin));

  //compute the control input and output it to the system
  un = controller(xhat, yhat, yn, un, rn);
  analogWrite(outputPin, digitalValue(un));

  //print the control input and system output
  Serial.print(loopNum * 1.0/sample_rate);
  Serial.print(' ');
  Serial.print(un);
  Serial.print(' ');
  Serial.println(yn);

  //wait so that we sample at 10Hz
  while(micros() - loop_start_time < loop_time);
  ++loopNum;
}
```

```
//This function implements the controller
//Observer, full-state feedback gains, and reference gains;
float controller(float* xhat, float& yhat, float yn, float un, float rn) {
  // yhat[n] = C*xhat[n];
  yhat = C[0]*xhat[0]+C[1]*xhat[1];
  // xhat[n+1] = Ad*xhat[n] + Bd*u[n] + L*(y[n] - yhat[n])
  float new_xhat0 = Ad[0][0]*xhat[0] + Ad[0][1]*xhat[1] + Bd[0]*un + L[0]*(yn-yhat);
  float new_xhat1 = Ad[1][0]*xhat[0] + Ad[1][1]*xhat[1] + Bd[1]*un + L[1]*(yn-yhat);
  xhat[0] = new_xhat0;
  xhat[1] = new_xhat1;

  // control law: u[n] = -K*xhat[n] + Kr*r[n]
  return -K[0]*xhat[0] - K[1]*xhat[1] + Kr*rn;
}

//This function converts the 10-bit value from the ADC to a voltage value
float voltage(int digital_value) {
 return 5.0*digital_value/1023;
}

//This function ensures that the output is in a valid range for analogWrite
int digitalValue(float voltage) {
  if(voltage > 5.0) {
    return 255;
  }
  else if (voltage < 0.0) {
    return 0;
  }
  else {
    return 255* (voltage/5.0);
  }
}
```