Chalmers University of Technology

SSY190: Homework 1

Group 14:
Mamadou Diaby
Raman Haddad
Daniel Pihlquist

April 1, 2015

1 Discretization

This section will describe the main steps of the discretization of the plant and the controller. The discretized models will then be transformed to the discrete time-domain in order to implement it as C-code.

1.1 Plant

The plant model is give by

$$G(s) = \frac{y(s)}{u(s)} = \frac{K}{1+sT}, \tag{1.1}$$

which is, by transforming it to the time-domain, equivalent to

$$\dot{y}(t) = -\frac{1}{T}y(t) + \frac{K}{T}u(t) = Ay(t) + Bu(t). \tag{1.2}$$

By discretizing (1.2) using the Zero-order hold method (ZOH) and a sampling time h

yields the following discrete time state space model

$$y(k+1) = \frac{1}{a}y(k) + \frac{K(a-1)}{a}u(k), \tag{1.3}$$

where $a = \exp\left(\frac{h}{T}\right)$.

1.2 Controller

The controller to be implemented, as per the specifications, is a PID controller with first-order filter on the derivative term and is given by

$$C(s) = \frac{u(s)}{e(s)} = K_p + \frac{K_i}{s} + K_d\frac{s}{1+T_f s}. \tag{1.4}$$

Using the Tustin method, this can be discretized by simply setting $s = \frac{2}{h}\frac{z-1}{z+1}$. Carrying this out results in the following transfer function

$$C(z) = \frac{u(z)}{e(z)} = \frac{c_{e_0}z^2 + c_{e_1}z + c_{e_2}}{c_0 z^2 + c_1 z + c_2}, \tag{1.5}$$

where the constants c_i and $c_{ei}, i \in \{0,\ 1,\ ,2\}$ are defined as

$$\begin{aligned}
c_{e_0} &= 4K_d + (hK_i + 2K_p)(h + 2T_f),\\
c_{e_1} &= -8K_d + 2h^2K_i - 8K_pT_f,\\
c_{e_2} &= 4K_d + (hK_i - 2K_p)(h - 2T_f),\\
c_0 &= 2(h + 2T_f),\\
c_1 &= -8T_f,\\
c_2 &= -2h + 4T_f.
\end{aligned} \tag{1.6}$$

From the above discrete time transfer function the control law becomes

$$\begin{aligned}
u(k) &= \frac{1}{c_0}\left(-c_1u_{k-1} - c_2u_{k-2} + c_{e_0}e_k + c_{e_1}e_{k-1} + c_{e_2}e_{k-2}\right),\\
e(k) &= r(k) - y(k)
\end{aligned} \tag{1.7}$$

Clearly given all the parameters, combining (1.3) and (1.7) the closed loop system can be implemented in C-code using the previous measurements, which can be initialised with zeros to begin with.

2 C - Implementation

It should be noted that we had problem compiling the code in the Virtual Machine. The compiler could not recognise the exponential function, we had to manually link the math-library by using the -lm flag while compiling. This was however not necessary in OSX.

The implementation in C basically consists of two functions, see Listing 1 and 2. These two functions are called from a loop within the `main` function. The `pid` function is called with previous errors set to zero and the output is then passed to the `plant` function with previous output value set to zero.

```
1  double plant(double u, double y)
2  {
3    double a = exp(h / T);  // Constant used in the plant
4    return (1 / a * (y + K * (a - 1) * u));
5  }
```

Listing 1: Function for plant

```
1  double pid(double ek, double ek_1, double ek_2, double uk_1,
       double uk_2)
2  {
3    /**
4    * Numerator and denominator coefficients used
5    * to describe the PID transfer function
6    **/
7    double c1 = (-8) * Tf;
8    double c2 = (-2) * h + 4 * Tf;
9    double c0 = 2 * (h + 2 * Tf);
10   double ce1 = (-8) * Kd + 2 * h * h * Ki + (-8) * Kp * Tf;
11   double ce2 = 4 * Kd + (h * Ki + (-2) * Kp) * (h + (-2) * Tf);
12   double ce0 = 4 * Kd + (h * Ki + 2 * Kp) * (h + 2 * Tf);
13
14   return ((1 / c0) *
15     ((-c1 * uk_1 - c2 * uk_2) +
16      (ce0 * ek + ce1 * ek_1 + ce2 * ek_2)));
17 }
```

Listing 2: Function for PID-controller