# Lab 13
# Discrete Time Control through Emulation

## 13.1 Objective

To implement digital controller through emulation of analog controller designed for regulation of dynamic response of an RC circuit and investigate the effect of sampling time on the performance of a digital control system

## 13.2 Introduction to Digital Controllers

Most of the controllers we have studied so far were described by the Laplace transform or differential equations, which are assumed to be built using analog electronics as done in Lab 11. However, most control systems today use digital computers (usually involving microprocessors) to implement the controllers. Analog electronics can integrate and differentiate signals continuously. In order for a digital computer to accomplish these tasks, the differential equations describing compensation must be approximated by reducing them to algebraic equations involving basic arithmetic operations i.e., addition, subtraction, division and multiplication.

**C**ontrol systems implemented on a digital computer are called ***Digital Control Systems***. Digital control systems differ from their analog counterparts in two important ways.

1. Digital systems operate in discrete time, not in continuous time. That is, control computations do not occur continuously (as done by op-amps for analog systems), but occur at discrete instants in time. These instants are usually regular periodic times, separated by the sampling period T [sec]. The sampling operation introduces artifacts into the signal, which are known as signal aliases. It also introduces delay into the closed loop system, which tends to destabilize the control.

2. Digital systems do not use infinite-precision mathematics. To store the value of a signal—such as a control signal—in computer memory, the value must be quantized. This means that the value is rounded to the nearest number which can be stored. The coefficients of the transfer function of the controller must also be quantized.

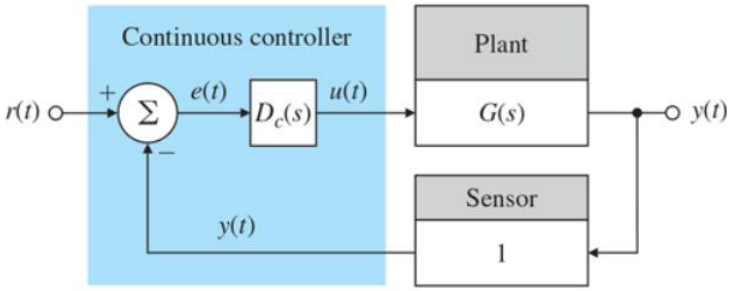
In this lab, you will implement and simulate a discrete-time system in Simulink.

## 13.3 Digitization

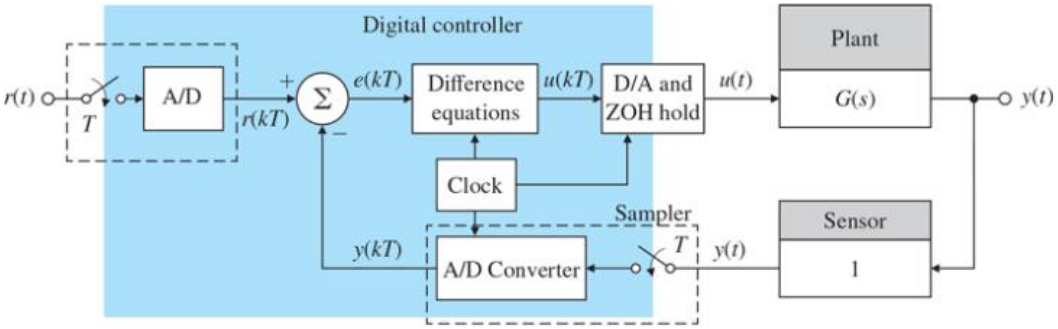
Figure 1(a) shows the topology of the typical continuous system that we have been considering in previous labs. The computation of the error signal e(s) and the dynamic compensation D(s) by controller can all be accomplished in a digital computer as shown in Figure 1(b). The fundamental differences between the two implementations are

(i) The digital system operates on samples of the sensed plant output rather than on the continuous signal

(ii) The control provided by D(s) must be generated by algebraic recursive equations




(a) Feedback control system with a continuous controller




(b) Feedback control system with a digital controller

Figure 1: Block diagram representation of basic control system with comparison of continuous and digital controllers

In the digital control system, the digital controller needs digitized data therefore; digitization i.e. analog to digital conversion of signals is needed. The output of controller (control signal) is initially a digital signal that needs to be converted to analog signal before giving it to the plant as plant needs analog signal for its operation.

## Analog to Digital Conversion and Sampling

Analog to Digital Converter (ADC) samples a physical variable (electrical voltage in our case) and converts into a binary number that usually consists of limited number of bits (say 10 bits). Conversion from the analog signal y(t) to the samples, y(kT), occurs repeatedly at instants of time T seconds apart. T is the sample period, and 1/T is the sample rate or sampling frequency in Hertz. This is represented in Figure 2. To simplify notation, we say that x[k] = x(kT) where x[k] is a discrete-time signal (denoted using square brackets) and x(t)is a continuous-time signal (denoted by parentheses). When you read x[k] you should understand that some sampling period T is implied, which must be somehow specified elsewhere.

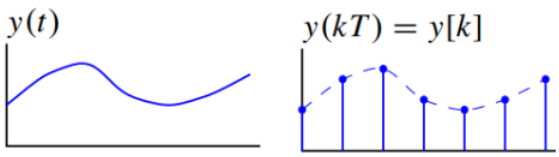



Figure 2: Sampling of a continuous signal *y(t)* where all sample are collected at sampling time of *T* sec

## Digital to Analog Conversion and Zero-Order Hold

The digital control signal produced by the digital controller is first converted into analog signal. Digital to analog converter changes the binary number to an analog voltage, and a zero-order hold maintains that same voltage throughout the sample period. This is shown in Figure 3. The resulting u(t) is then applied to the actuator in precisely the same manner as the continuous implementation.

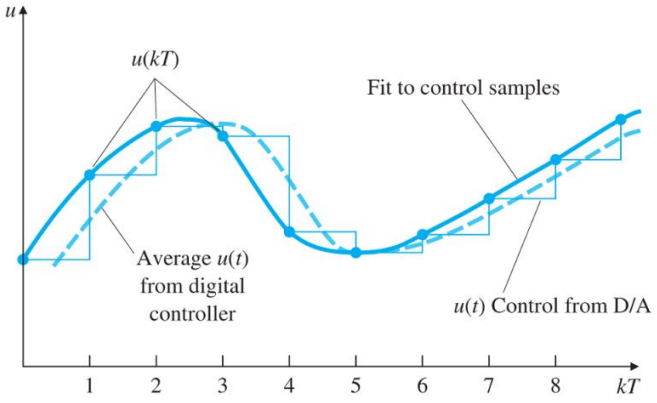



Figure 3: Output of sample and hold

## 13.4 Difference Equation and Discrete Time Controller

The continuous-time systems are defined by differential equations. A discrete time controller would be implemented with difference equation such as:

$$\sum_{i=0}^{n} a_i u[k-i] = \sum_{i=0}^{m} b_i e[k-i]$$

With input e[k] and output u[k] and $a_i$ and $b_i$ which depend upon the transfer function of the controller D(s) in Figure 1. We will cover how cover how to get difference equations of discrete time controller in section 11.6. An example of discrete-time system is a discrete-time integrator. If the sampling period is *T*, then the output of an integrator may be approximated by (rectangular rule).

$$y[k] = T\sum_{i=0}^{k} x[i]$$

$$y[k-1] = T\sum_{i=0}^{k-1} x[i]$$

$$y[k] = y[k-1] + Tx[k]$$

## 13.5 Discrete Time Simulation

When we implement the difference equations, we need two quantities: x[k] and y[k-1]. The input to the system is x[k] and y[k-1] is a delayed version of the system's output. Therefore, we can implement the equations with a feedback loop which has a delay in it.

Simulink has a number of blocks for discrete-time simulations in its Discrete-Time Simulink Library. The two blocks of interest to us now are the "Zero-Order Hold" and "Unit Delay" blocks.


Figure 4: Simulink's unit delay and zero-order hold blocks

For integrator example, consider the difference equation derived for integrator output. The block diagram of Figure 5 implements integrator with sine wave input. A sine-wave generator generates an input of *x(t)*. This signal is sampled to make *x[k]*. The Simulink block which does this function is the Zero-Order Hold block. The *x[k]* is scaled by T to produce *Tx[k]* as per equation. The output of the integrator is *y[k]* which is computed as *x[k]+y[k-1]*. The final remaining signal *y[k-1]* is computed from *y[k]* by passing it through a Unit-Delay block which gives last sample i.e. delayed by one sample time unit

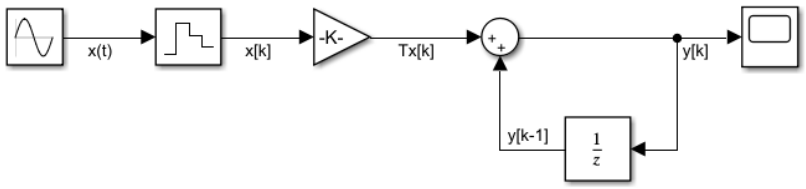
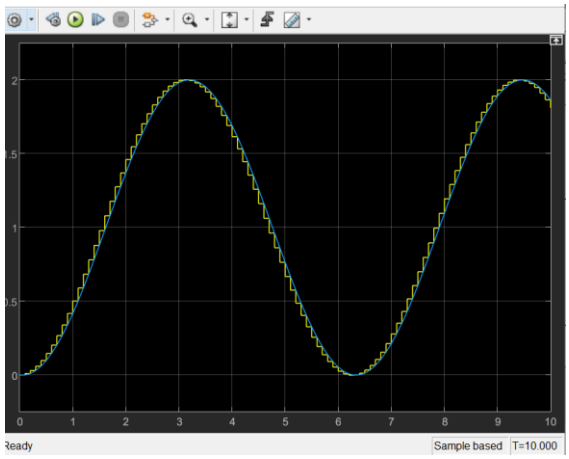
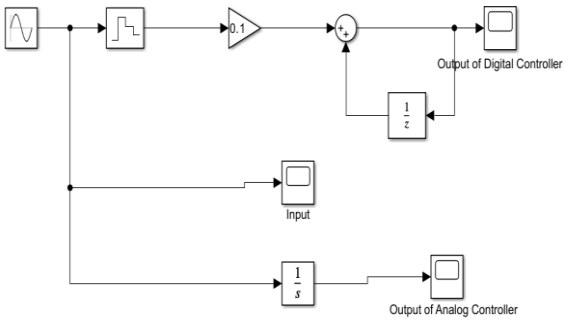



Figure 5: Discrete-time integrator implemented in Simulink

## Task 1: To implement discrete-time integrator in Simulink

1. To practice the discrete-time simulation from difference equation, implement the model shown in Figure 5 to integrate a sine wave of 1 rad/sec with amplitude 1V.
2. For discrete time integrator, take the sampling time *T* of 0.1 sec in your simulation.
3. Compare it with the output of a continuous integrator when it is provided the same sine wave input.
4. Add captures of the obtained plots.



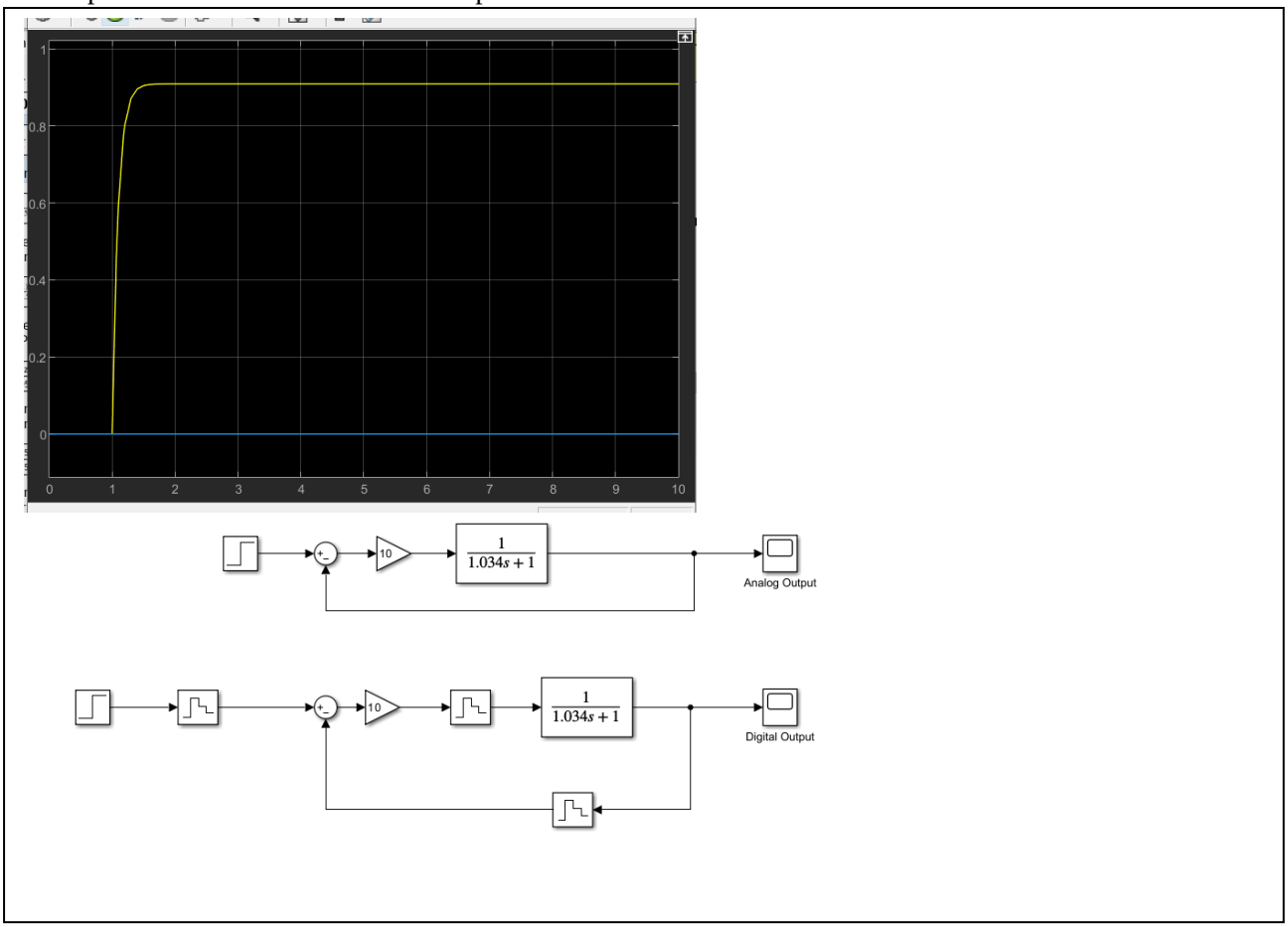

As depicted, the yellow plot represents a discrete-time integrator, while the blue plot corresponds to a continuous-time integrator. It is evident that the output of the discrete-time integrator closely approximates the output of the continuous-time integrator.

## Task 2: To implement discrete proportional controller to control the response of RC circuit

In Lab 11, we implemented analog proportional controller to control the response of an RC circuit (R=2.2kOhms and C=470uF). The plant transfer function was given by; $P(s) = \frac{1}{RCs+1}$.

1. Create a Simulink model to simulate the analog control system for the same plant with P-controller having gain Kp=10. The same was implemented in Lab 11 through op-amps.

2. Now, copy the same system in your model to replace the analog controller representation by a digital controller. The only difference in the two models is the need to convert digital output of controller into analog by using zero-order hold. Use sampling time of 0.01 sec.
3. Add captures of Simulink model and response with suitable labels.




## 13.6 Digital Control Design by Emulation of Analog Controllers

In analog control scheme, we have dynamic compensator Dc(s) which is controller in feedback loop (See Figure 1). By the design of digital controller, we mean obtaining difference equation, which achieves required control specifications. There are two ways to design a digital control system.

*a.* *Discrete equivalent design (also called design by emulation):* In this design, we start from already designed analog controller transfer function D(s) and try to mimic its performance in discrete time using approximate difference equation.
*b.* *Direct digital design (also called discrete design):* In this design approach, we design the controller in discrete time directly without designing first D(s). This approach is beyond the scope of this lab.

In this lab, we will be using method (a) for implementing PI controller that we designed in Lab 09. The purpose of this lab is finding a discrete equivalent to a given analog controller which will approximate the differential equation of the controller. The assumption is that we have the transfer function of an analog controller and wish to replace it with a discrete controller. That controller will accept samples of the controller input *e(kT)* from a sampler and, using past values of the control signal *u(kT)* and present and past samples of the input *e(kT)*, will compute the next control signal to be sent to the actuator.

## Approximation of Derivative

Basis of controller design through emulation fundamentally comes from approximation of derivative operator. A derivative operator is basically defined as;

$$\dot{x}(t) = \underset{\delta t \to 0}{Lim} \frac{\delta x}{\delta t} = \underset{\delta t \to 0}{Lim} \frac{x(t) - x(t - \delta t)}{\delta t}$$

If the sampling rate ***T*** of the discrete-time system is small, we may approximate $\delta t \approx T$

$$\dot{x}(kT) \approx \frac{x(kT) - x((k-1)T)}{T}$$

$$\dot{x}[k] \approx \frac{x[k] - x[k-1]}{T}$$

Above approximation is called backward approximation. Other approximations are given in Table 1. For higher order derivatives, we can repeat the same approximation for example:

$$\ddot{x}[k] \approx \frac{\frac{x[k] - x[k-1]}{T} - \frac{x[k-1] - x[k-2]}{T}}{T}$$

$$\ddot{x}[k] \approx \frac{x[k] - 2x[k-1] + x[k-2]}{T^2}$$

Table 1: Different approximations of derivative

| | |
|---|---|
| **Backward Approximation** | $\dot{x}[k] \approx \frac{x[k] - x[k-1]}{T}$ |
| **Forward Approximation** | $\dot{x}[k] \approx \frac{x[k+1] - x[k]}{T}$ |
| **Tustin's Approximation** | $\frac{\dot{x}[k] + \dot{x}[k-1]}{T} = \frac{x[k] - x[k-1]}{T}$ |

## Putting the Pieces Together

To see how to use approximations to convert an analog controller into a digital controller we will examine a concrete example. Consider general form of PI controller:

$$D(s) = K_P + \frac{K_I}{s} = \frac{K_P s + K_I}{s}$$

Notice that from Figure 1, $D_C(s) = \frac{U(s)}{E(s)}$

$$D_C(s) = \frac{U(s)}{E(s)} = \frac{K_P s + K_I}{s}$$

Re-arranging we get:

$$sU(s) = (K_P s + K_I)E(s) = K_P sE(s) + K_I E(s)$$

Taking inverse Laplace of above equation and noticing $sF(s) \rightarrow \dot{f}(t)$

$$\dot{u}(t) = K_P \dot{e}(t) + K_I e(t)$$

Now substituting backward approximation of derivative, we get:

$$\frac{u[k]-u[k-1]}{T} = K_P\left(\frac{e[k]-e[k-1]}{T}\right) + K_I e[k]$$

$$\frac{u[k]}{T} = K_P\left(\frac{e[k]-e[k-1]}{T}\right) + K_I e[k] + \frac{u[k-1]}{T}$$

$$u[k] = K_P(e[k]-e[k-1]) + TK_I e[k] + u[k-1] \quad \text{(A)}$$

This equation says that the control signal at any time index *k* may be computed using the previous control value *u[k-1]*, the current error signal *e[k]* and the previous error signal *e[k-1]*.

## 13.7 Implementation of Difference Equation

Difference equation (A) can easily be implemented inside any digital computer (even inside microcontroller!). Figure 6 shows Simulink implementation of the equation.

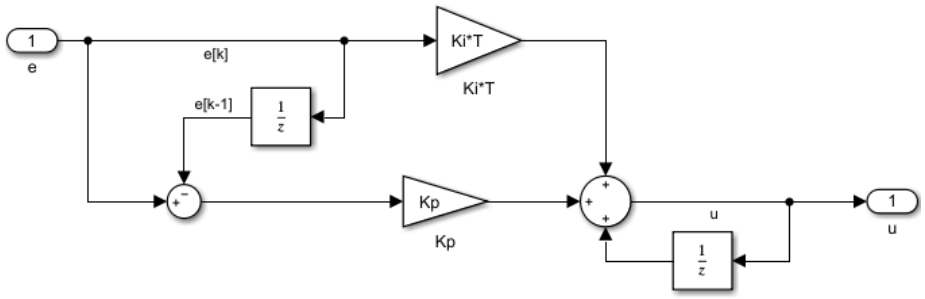


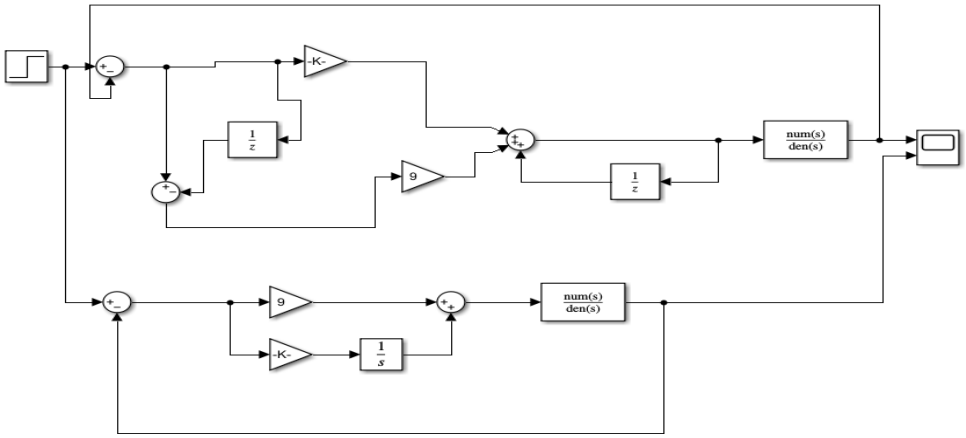

Figure 6: Simulink implementation of PI controller using difference equation

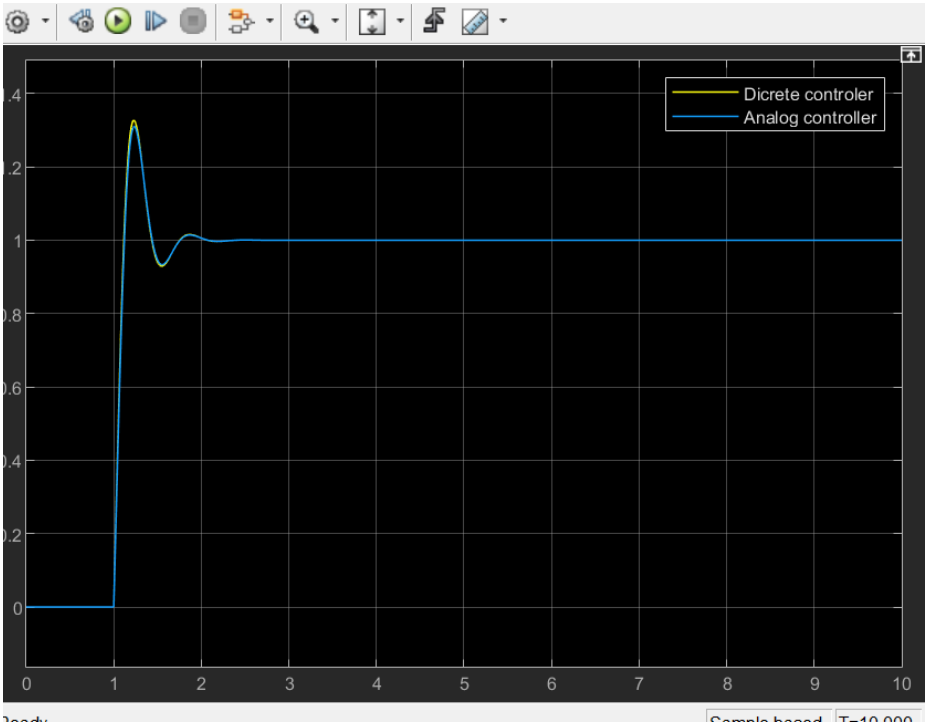
## Task 3: To implement digital PI controller through emulation

The value of controller gains; *Kp* and *Ki*, obtained for the designed PI-controller of lab 09 were 9 and 125, respectively.

1. Modify the Simulink model of Task 2, to replace the P controller by PI controller. Use transfer function block with the given controller gains to model PI controller.
2. Now, in the digital control system counterpart, add the difference equation implementation of PI controller as given in Figure 6. Take sampling time of 0.01 sec.

3. Compare the output of analog and digital control systems by obtaining plots on same window.



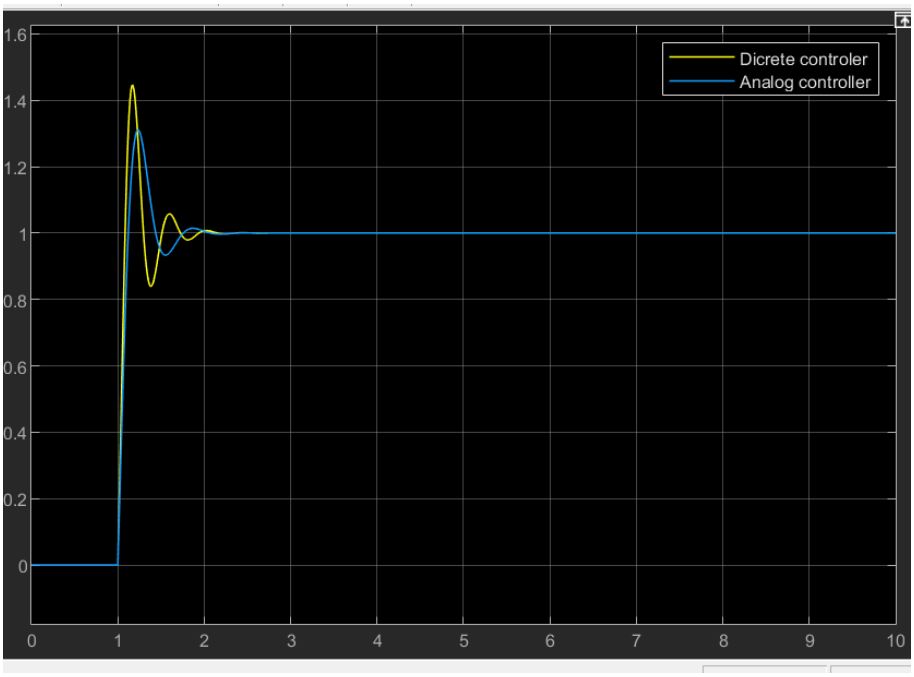




The response of digital control system is same as the analog control system, however overshoot of digital control is bit higher

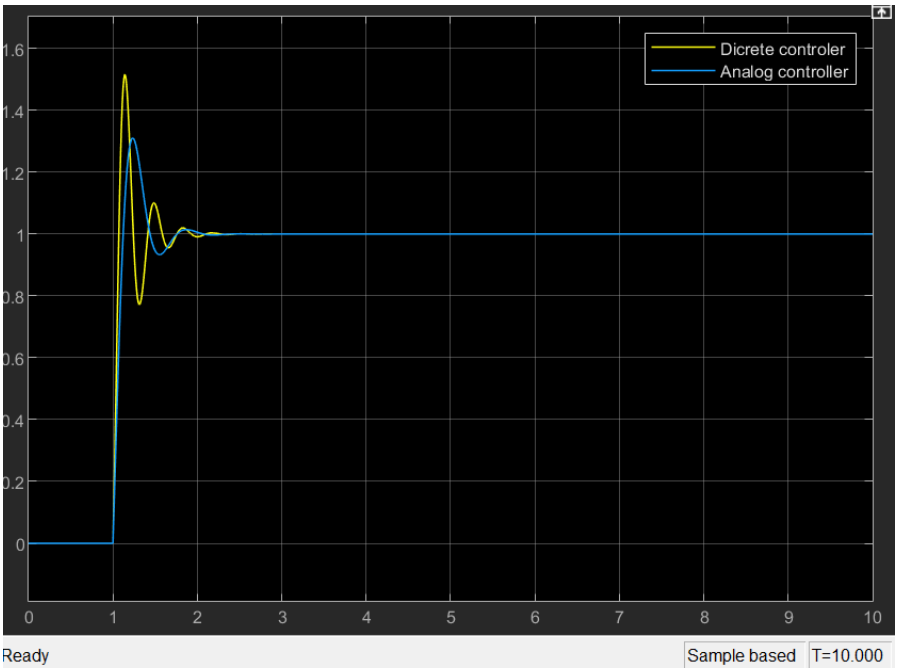
4. Add appropriate labels to your plots. Add capture of plots and implemented Simulink models.

## Task 4: To investigate the effect of sampling time on system response and establish rule of thumb between time-constant and sampling time

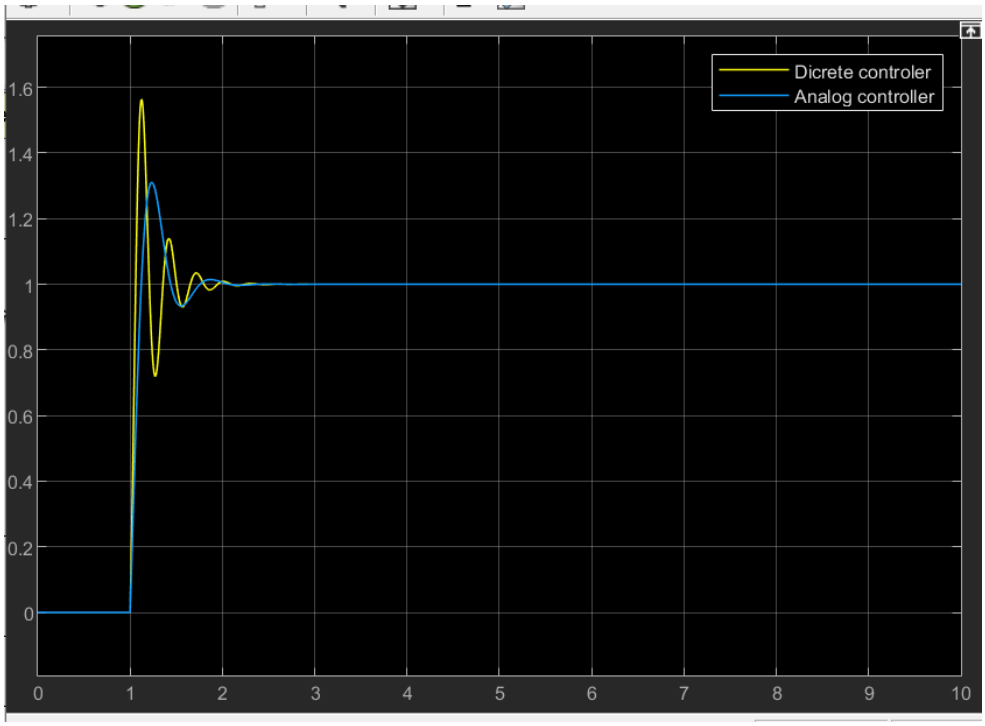
1. In your implemented model, now vary the sampling time ***T*** and observe the changes in response. Increase sampling time from 0.01sec with a step-size of 0.01 sec.
2. Analyze the effect of increasing sampling time on system response. Note that the time constant for the given system is 1.03sec (τ=RC).


Dicrete controler
Analog controller
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0 1 2 3 4 5 6 7 8 9 10

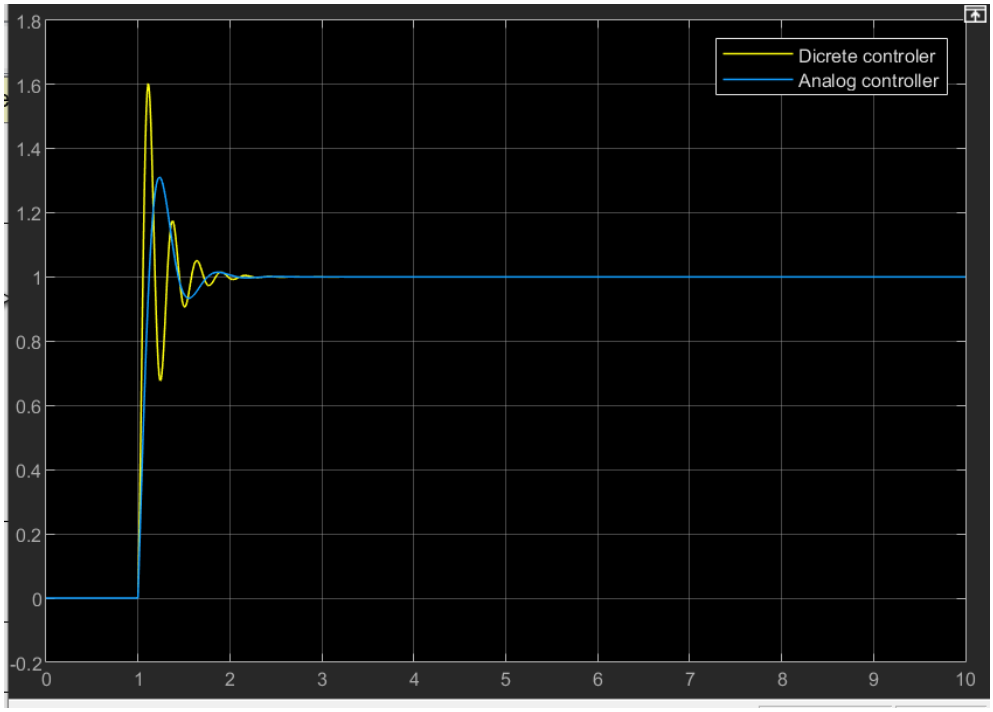
Ts = 0.02


Dicrete controler
Analog controller
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
T=10.000

Ts= 0.03




Ts = 0.04

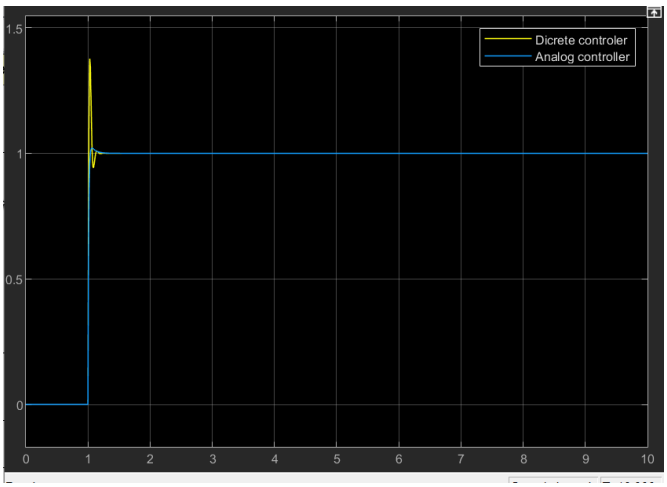



Ts = 0.05

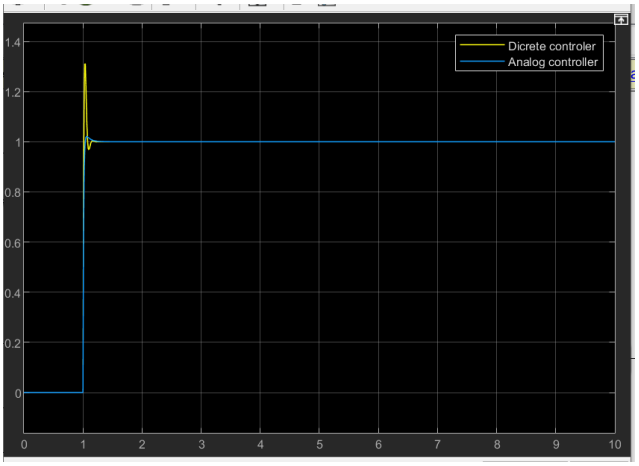
Observation: As Ts increases, the system's output exhibits amplified transient behavior with increasing overshoot and a tendency to oscillate around the desired steady-state value.

3. Modify the plant dynamics such that the time constant becomes 0.1 sec. Repeat the above steps and again observe the impact of increasing sampling time on system response.

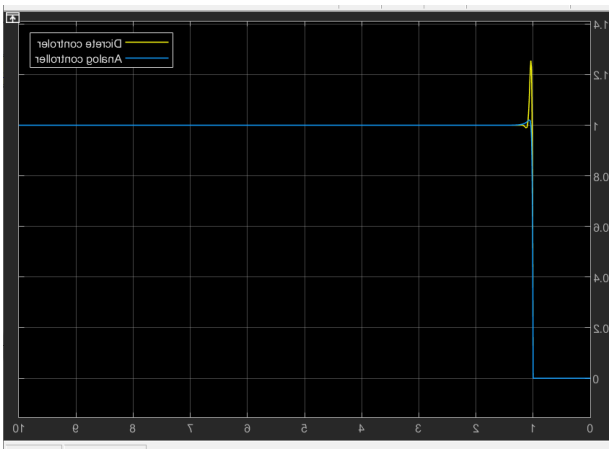
Sample time 0.05



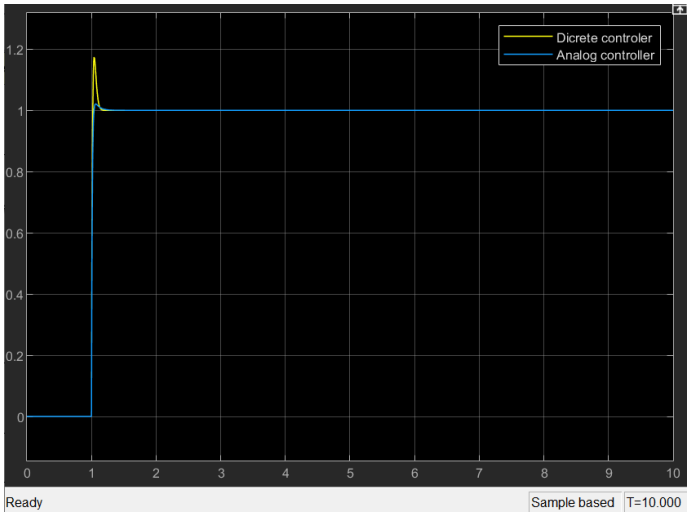

Sample time 0.04



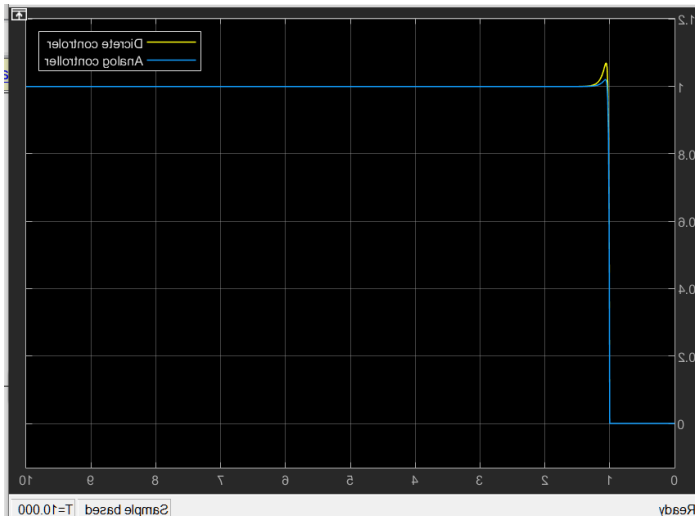

Sample time 0.03



Sample size 0.02




sample size 0.01




The time constant experiences a decline with an increase in sampling time until it reaches 0.03 seconds. However, surpassing a sampling time of 0.03 seconds results in a reversal, causing the time constant to once again rise. Additionally, this extended sampling time induces oscillations in the response of the digital control system..

4. Establish a rule of thumb between plant time constant (τ) and sampling time (T).

| Time constant reduces for small increase in sampling time before oscillation appears |
|---|

## 13.8 Digital Controller Implementation via Arduino Uno using Simulink Support Package

Using Arduino IDE, you can implement controllers through the difference equations on microprocessors like Arduino. Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs; these pins are represented by ~ on the board) and 6 analog input pins (A0 to A5). Analog input pins can be utilized to acquire signals like plant's output that is needed for computation of error. The error then can be used in controller's difference equation to generate controller output which is given to the plant through the output pins.

You can easily find Arduino sketches (codes) that implement PID controllers. Without going into the details of programming Arduino and its IDE, we will be implementing a digital controller in the following task with the help of Simulink® Support Package for Arduino® Hardware. This add-on is already installed on the lab PCs.

With Simulink® Support Package for Arduino® Hardware, you can use Simulink to develop and simulate algorithms that run standalone on your Arduino. It provides an ease of using Simulink blocks for configuring and accessing Arduino inputs and outputs. A quite useful feature of this support package is running simulation in **External mode for interactive parameter tuning and signal monitoring** as your algorithm runs on the device. Model deployment for standalone operation on Arduino boards is also easily possible through **Build, Deploy & Start** option.

## Task 5: To test Arduino Uno with hardware through Simulink interface

In this task, we will be testing the response of the same RC circuit in open loop by giving it input through Arduino and acquiring its output through Arduino to view in Simulink's scope.

1. Create a new Simulink model and don't forget to save it.
2. Go to Modeling > Model Settings (Ctrl+E is the shortcut for this). In Configuration Parameters window, under **Solver** tab, select Type: **Fixed-step** and in step size write **Ts**. This will be used as the sampling time throughout the configuration in all Simulink blocks.
3. Go to the **Hardware Implementation** tab, in **Hardware board** select **Arduino Uno**. You can explore Target Hardware Resources etc. For now, leave everything unchanged. Click OK and close the window.
4. You will see another tab **HARDWARE** in toolstrip at the top of Simulink window. For the hardware in loop simulations, we will be using this tab and its options.
5. From library browser, go to Simulink Support Package for Arduino Hardware > Commons. Add an Analog Input block to your model. The Analog Input block is equivalent to the ***analogread( )*** command. It reads the value from the specified analog pin. The Arduino board contains a 6 channel,10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023.
6. Select the pin number through which you want to read analog input (plant's output i.e., voltage across capacitor in our case) for example pin 4. Write **Ts** in sample time option. To map the integer values in equivalent voltage, add a gain of 5/1023 and connect a Scope to view the analog input to Arduino i.e.,

output of circuit (Voltage across capacitor). In Sample Time parameter of Scope Configuration Properties…, write **Ts**.

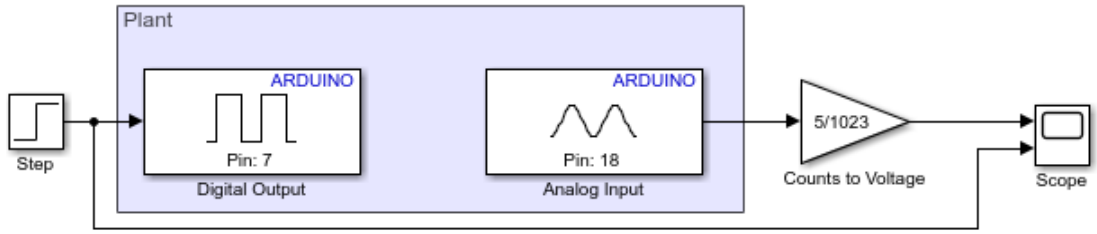
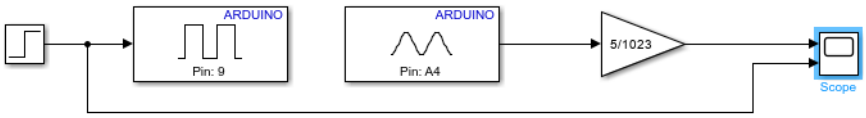
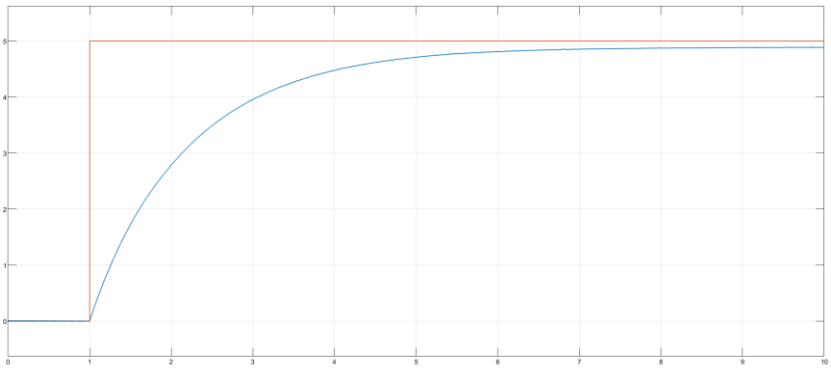
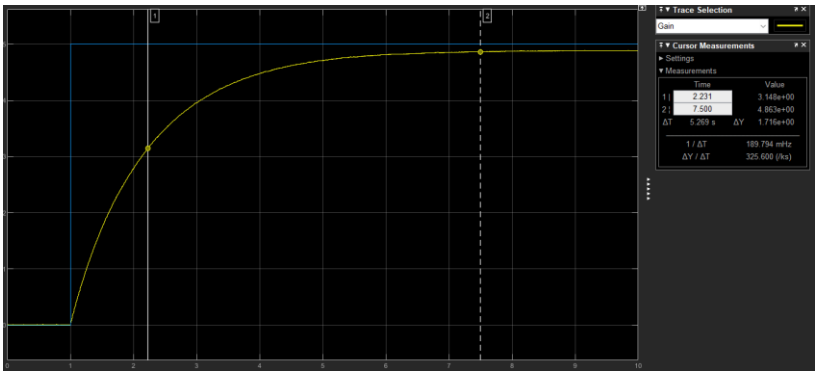



Figure 7: Simulink model for Task 5-open-loop response of RC circuit

7. To give input to the plant, add a Digital Output block and select an appropriate pin number for example pin 7. As a digital output pin, this can give high voltage (5V) or low voltage (0V) at the output. Connect a Step block at the input of this Digital Output block. Set Initial Value 0, Final Value 5, and Step Time of 1sec. Write **Ts** in Sample Time. **Note:** If the Digital Output block is given 1 at its input, it will give high voltage. Giving it **5** at input through step-block is just for viewing the actual input voltage in Scope. Connect it with Scope to view the input and output of plant in same window.
8. Make arrangements for the hardware setup. Connect Arduino Uno through USB port. On a breadboard, connect resistor (2.2k Ohms) and capacitor (470uF) in series. Connect the Ground pin of Arduino board with the ground of your circuit. The specified Digital Output pin (pin 7) should be connected with the RC circuit at input terminal. Analog Input pin (pin 4) should be connected at output terminal of RC circuit i.e., across capacitor.
9. Go to command window and initialize Ts variable with value of 0.001 sec.
10. Now you are ready to run the model in External mode by using Monitor and Tune option at the top with a green run button. Observe the response of RC circuit. Verify the time constant of the system from open loop response. Add captures of the response and model.






Time constant is found out to be twice the value of simulated system i.e. 2.231 sec . This has occurred due to possible issue with physical system.

## Task 6: To implement PI-controller through Arduino Uno

In this task, we will be replacing the analog controller implementation from lab 11 by a digital controller (Arduino Uno implementation of PI controller) to control the same RC circuit. Follow the step-by-step instructions below.

1. Let's convert the open-loop system into a closed-loop system with the PI-controller designed in Lab 09.
2. To do this, first delete the Digital Output block. Add a Sum block to find error by subtracting the plant's output and the reference input (Step). Set final value of step to 0.5V.
3. Once error is computed, add blocks to implement PI controller with proportional controller gain 9 and integral controller gain of 125. The control output should be given through an Arduino pin to the plant's input terminal. We can't use a Digital Output pin as it gives high or low voltage signals. We want a control signal that is dependent on error as calculated by the designed PI-controller. So, the controller's output is in fact a signal with varying magnitude.
4. Unfortunately, Arduino Uno doesn't support analog output. However, it has a few pins that support PWM output and these can be used as output pins for analog write operation. Add a PWM block from Simulink Support Package for Arduino > Commons library. Select pin number 5 which is one of the PWM pins represented by ~ on board. Set frequency of 980.4 Hz. The PWM block accepts the values between 0 and 255. The input controls the duty cycle of the square waveform. An input value of 0 produces a 0 percent duty cycle and an input value of 255 produces a 100 percent duty cycle. Before connecting it with the PI-controller's output, add a gain with value 255/5 since we want maximum 100 % duty cycle for 5V. The gain will convert PI controller's output to a number from 0 to 255 acceptable by the PWM block.

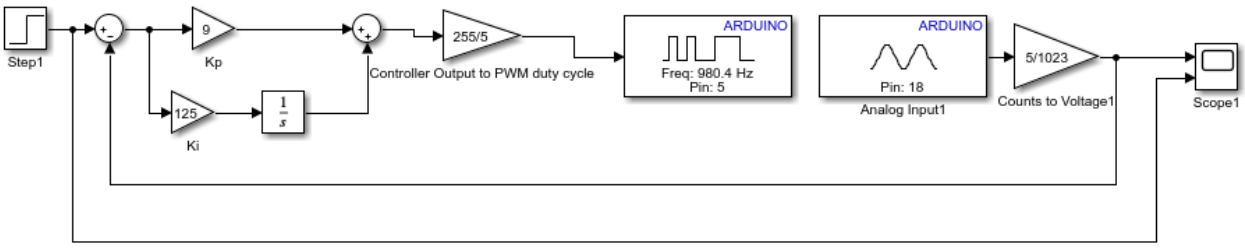
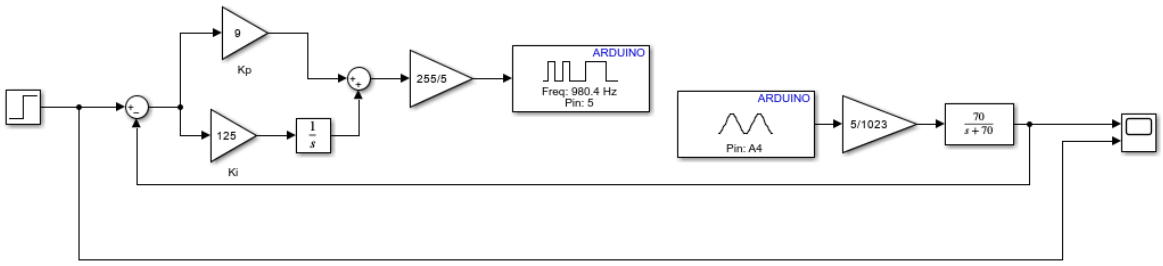
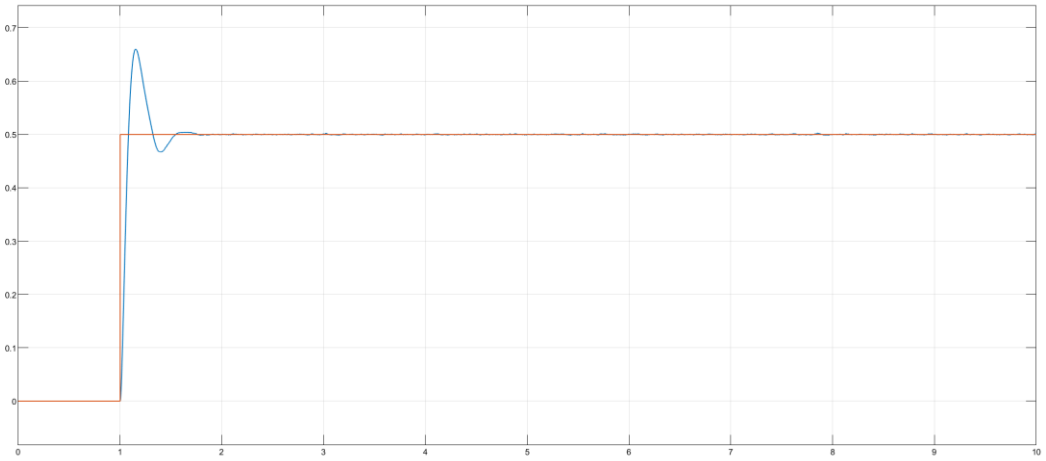
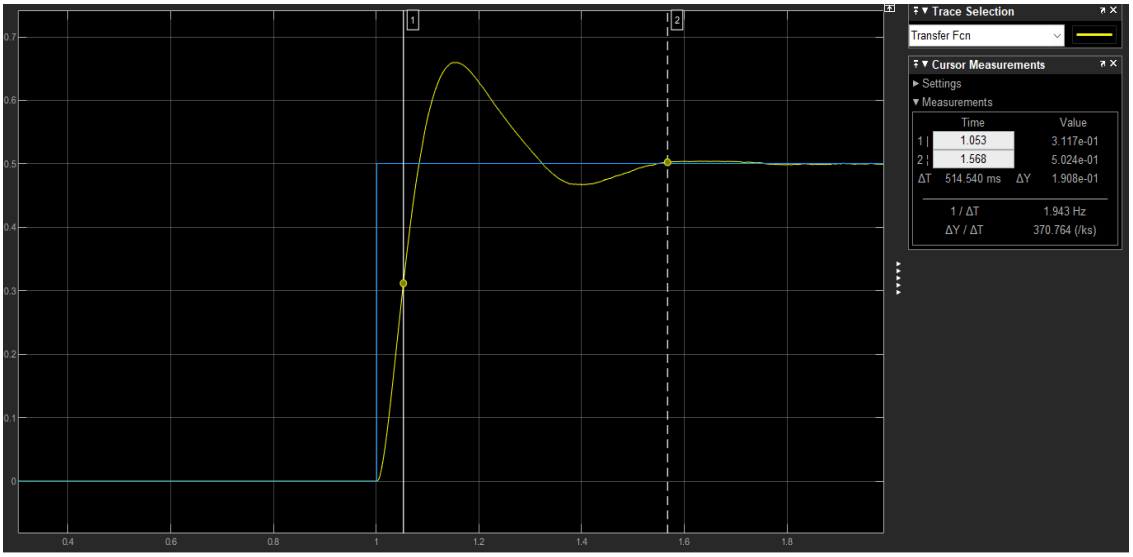



Figure 8: Simulink model for Task 6- closed-loop control of RC-circuit with PI controller

5. Test this controller and the closed-loop response by again running the model in External mode (Monitor & Tune). You will observe a lot of noisy peaks in the response. You can add a low pass filter at the output with transfer function $\frac{70}{s+70}$ after the gain block [5/1023]. For this gain block, set the data type to double in signal attributes tab in its properties.
6. Add capture of the response and comment about the exercise of implementation of digital controller in comparison to analog controller implementation.






Time constant = 1.053 sec

As evident, the system's response is notably faster, and the time constant closely approximates the simulated system's value at 1.053 seconds. However, this acceleration results in an undesirable overshoot.

***Interesting Facts:***

7. Note that at the top right, there is an option of Build, Deploy and Start. Once parameters are finalized through Monitor and Tune option, the standalone application can be deployed on Arduino using this option. You can explore more about this.
8. Arduino pins cannot give a negative voltage but there are ways through which the control signals (PWM signals) can be amplified through additional circuitry or can be utilized to generate voltage signals in both polarities. To control motors through Arduino, motor drivers are often used which solve the problem of power amplification and voltage polarities.

## 13.9 Post-Lab Task

## Task 7: To implement difference equation of given Lead/Lag controller and PID controller

1. Obtain difference equations for the following analog controllers. Show all intermediate steps.
    - Lead/Lag Controller: $D_c(s) = \frac{s+a}{s+b}$
    - PID Controller: $D_c(s) = K_p + \frac{K_i}{s} + K_d s$

1) Difference Equation for lead/lag controller $D_c(s) = \frac{s+a}{s+b}$

$$D_c(s) = \frac{s+a}{s+b}$$

$$\frac{U(s)}{E(s)} = \frac{s+a}{s+b}$$

$$sU(s) + bU(s) = sE(s) + aE(s)$$

$$\dot{u}(t) + bu(t) = \dot{e}(t) + ae(t)$$

$$\dot{u}(t) = \frac{u[k] - u[k-1]}{T}, \quad \dot{e}(t) = \frac{e[k] - e[k-1]}{T}$$

$$e(t) = \frac{e[k] - e[k-1]}{T}$$

$$\frac{u[k] - u[k-1]}{T} + bu[k] = \frac{e[k] - e[k-1]}{T} + ae(k)$$

$$u[k] - u[k-1] + bTu[k] = e[k] - e[k-1] + aTe[k]$$

$$u[k] = \frac{e[k](1+aT) - e[k-1] + u[k-1]}{1+bT}$$

Difference Equation for PID $k_p + \frac{k_i}{s} + k_d s$

$$D_c(s) = k_p + \frac{k_i}{s} + k_d s$$

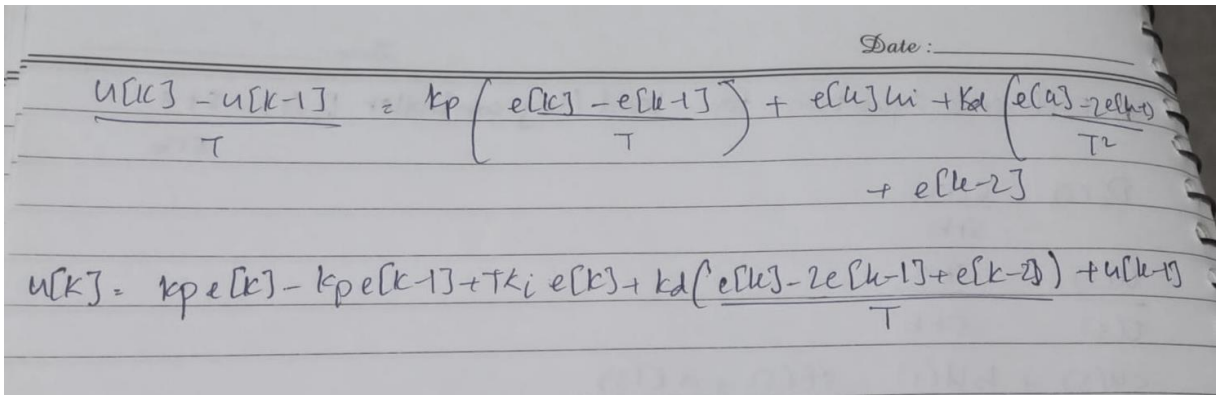
$$\frac{U(s)}{E(s)} = k_p + \frac{k_i}{s} + k_d s$$

$$U(s) = \left[k_p + \frac{k_i}{s} + k_d s\right] E(s)$$

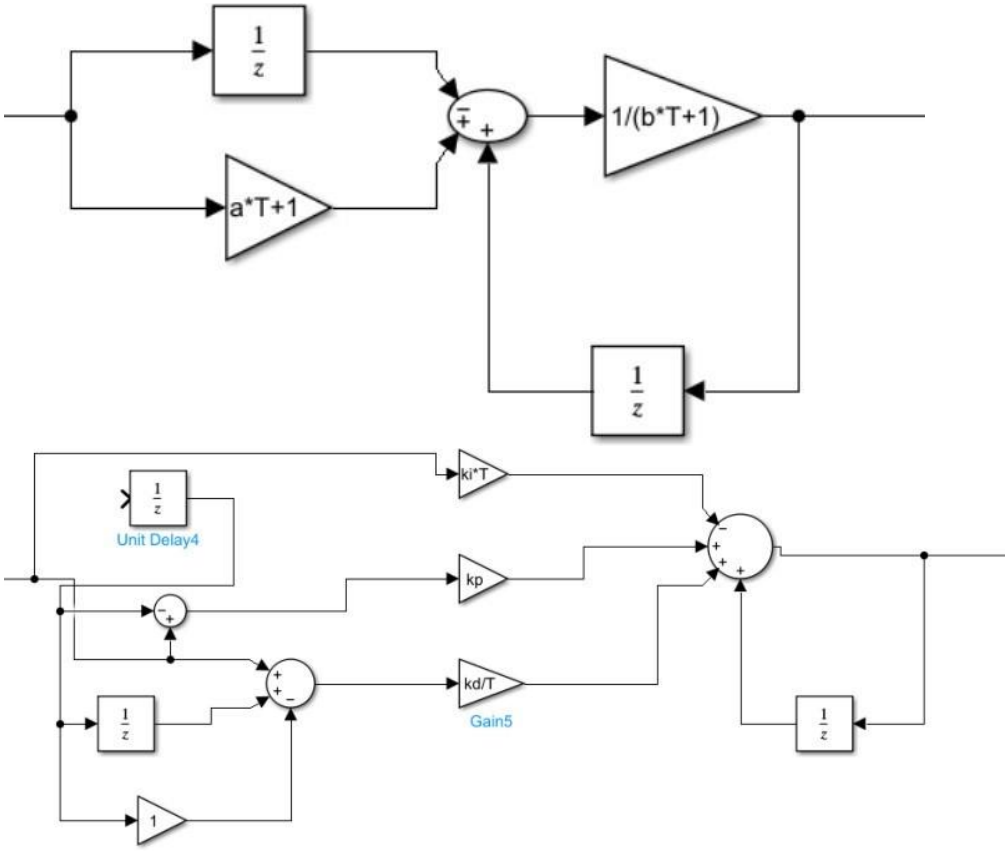
$$sU(s) = \left[sk_p + k_i + k_d s^2\right] E(s)$$

$$\dot{u}(t) = k_p \dot{e}(t) + k_i e(t) + k_d \ddot{e}(t)$$

$$\ddot{e}(t) = \frac{e[k] - 2e[k-1] + e[k-2]}{T^2}$$




2. Implement the obtained difference equations in Simulink environment. Add capture of Simulink models. Response is not needed.



# References

1) Gene F. Franklin, J. David Powell, A.E. Naeini, Feedback Control of Dynamic Systems

2) Dr. Gregory L. Plett; ECE4560 Digital Control Laboratory; University of Colorado at Colorado Springs

# Assessment Rubric
# Lab 13
## Discrete Time Control thought Emulation

| Name: Afsah Hyder | Student ID:07065 |
|---|---|

**Points Distribution**

| Task No. | LR2 Simulation | LR5 Plots | LR 10 Analysis | Class Participation |
|---|---|---|---|---|
| **Task 1** | 4 | 4 | - | - |
| **Task 2** | 4 | 4 | - | - |
| **Task 3** | 8 | 8 | 4 | - |
| **Task 4** | 8 | 8 | 8 | - |
| **Task 5** | 8 | 4 | - | - |
| **Task 6** | 8 | 4 | - | - |
| **Task 7** | 8 | - | 8 | - |
| **SEL** | | | | 20 |
| **Course Learning Outcomes** | CLO 3 | | | CLO 4 |
| **Total Points** | 100 | | | 20 |
| | 120 | | | |

For details on rubrics, please refer to *Lab Evaluation Assessment Rubrics*.