Controlling Single Board Heater System by PID controller

AIM:

1. Controlling Single Board Heater System by using PID controller.

Target group:

- Anyone who has basic knowledge of Control Engineering.

About this Experiment

- Figure1 shows the single board heater system on which this experiment will be performed.
- The setup consists of a heater assembly, fan, temperature sensor, microcontroller and associated circuitry.
- Heater assembly consists of an iron plate placed at a distance of about 3.5 mm from the nichrome coil.
- A 12 V computer fan positioned below this heater assembly is meant for cooling the assembly.
- The temperature sensed by the temperature sensor, AD 590, after suitable processing, is fed to the microcontroller.
- The microcontroller ATmega16 is the heart of the setup. It provides an interface between the process and the computer.
- The LCD display mounted above the microcontroller displays the heated plate temperature, heater and fan inputs and also the commands communicated via serial port.

Figure 1: Single board heater system

- The setup is powered by 12 V, 8 A SMPS.
- We have used LabVIEW as an interface for sending and receiving data. This interface is shown in Fig.2.
- Heater current and fan speed are the two inputs for this system. They are given in PWM units.

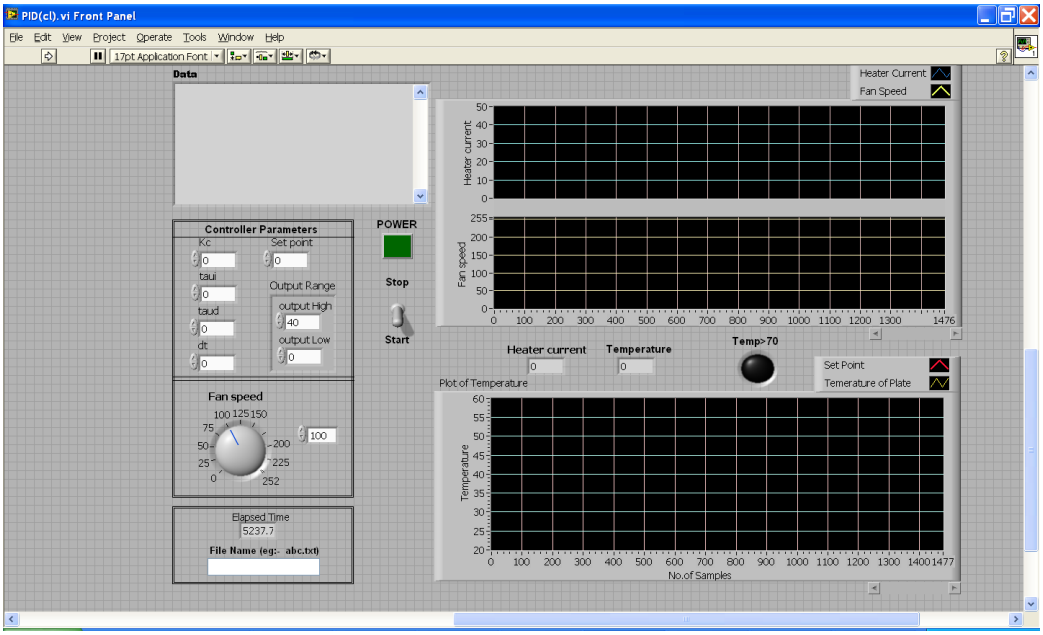

Figure 2: Lab VIEW interface for this experiment

- The plots of their amplitude versus no. of collected samples are also available on the panel.
- The output temperature profile, as read by the sensor, is also plotted.
- The data acquired in the process is stored and simultaneously displayed at the upper left corner of the front panel.
- The whole data could be highlighted, copied and pasted anywhere else on your computer.
- Before beginning with the experiment, one has to input a unique name to the data file which would be created (at bottom left corner). Else, the program will not run and you cannot perform the experiment. A new file name is expected everytime you run the experiment, everytime you get an error message.
- Make sure that the start button is at the start position before you run the program.

Theory

A PID controller is one which tries to minimize the error between measured variable and the set point by calculating the error and then putting a suitable corrective action. For revision purpose, the output of interest of a process is called as the measured variable or process variable, the difference between the set point and the measured variable is called as error and the control action taken to adjust the process is called as manipulated variable. A PID controller does not simply adds or subtracts the control action but instead it manipulates it using three distinct control features, namely, Proportional, Integral and Derivative. Thus, a PID controller has three separate parameters.

1. **Proportional**
   This parameter generates a control action based on the current value of the error. In more simplified sense, if the error is +2, the control action is -2. The proportion action can be generated by multiplying the error with a Proportional constant K_P. Mathematical representation of the same is given below,

$$P_o = K_P e(t) \tag{1}$$

   where,
   P_o is the proportional output
   K_p is the proportional gain
   $e(t)$ is the error signal

   The value of K_p is very important. A large value of K_P may lead to instability of the system. In contrast, a smaller value of K_P may decrease the controllers sensitivity towards error. The problem involved in using only proportion action is that, the control action will never settle down to its target value but will always retain a steady-state error.

2. **Integral**
   This parameter generates a control action depending on the history of errors. It means that the action is based on the sum of the recent errors. It is proportional to both the magnitude as well as duration of the error. The summation of the error over a period of time gives a value of the offset that should have been corrected previously. The integral action can thus be generated by multiplying this accumulated error with an integral gain K_i. Mathematical representation of the same is given below,

$$I_o = K_i \int_0^t e(t)dt \tag{2}$$

   where,
   I_o is the integral output
   K_i is the integral gain ($K_i = K_p/\tau_i$....where, τ_i is the integral time)

   The integral action tends to accelerate the control action. However, since it looks only at the past values of the error, there is always a possibility it causing the present values to overshoot the set point values.

3. **Derivative**
   As the name suggests, a derivative parameter generates a control action by calculating the rate of change of error. A derivative action is thus generated by multiplying the value of rate of change of error with a derivative gain K_d. Mathematical representation of the same is given below,

$$D_o = K_d \frac{d}{dt} e(t) \tag{3}$$

   where,
   D_o is the derivative output
   K_d is the derivative gain ($K_d = K_p/\tau_d$....where, τ_d is the derivative time)

   The derivative action slows the rate of change of controllers output. A derivative controller is quite useful when the error is continuously changing with time. One should however avoid using it alone. This is because there is no output if error is zero and when the rate of change of error is constant.

When all the above control actions are summed up and used together, the final equation becomes,

$$PID_o = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{4}$$

The above equation represents an ideal form of PID controller. This means that the integral controller was used independently. However, it is not a good decision since, the integral action begins only after the error exits for some amount of time. The proportional controller however begins as soon as the error starts existing. Hence, the integral controller is seldom used in conjunction with a proportional controller popularly known as PI controller. Therefore the equation for Proportional Integral action becomes,

$$PI_o = K_p e(t) + \left(K_p/\tau_i\right) \int_0^t e(t)dt \tag{5}$$

$$= K_p \left\{ e(t) + (1/\tau_i) \int_0^t e(t)dt \right\} \tag{6}$$

Similarly, as discussed before, independent use of derivative controller is also not desirable. Moreover, if the process contains high frequency noise then the derivative action will tend to amplify the noise. Hence, derivative controller are also used in conjunction with Proportional or Proportional Integral controller popularly known as PD or PID, respectively. Therefore the equation for Proportional Derivative action becomes,

$$PD_o = K_p e(t) + K_p \tau_d \frac{d}{dt} e(t) \tag{7}$$

$$= K_p \left\{ e(t) + \tau_d \frac{d}{dt} e(t) \right\} \tag{8}$$

Finally, writing the equation for PID controller gives,

$$PID_o = K_p \left\{ e(t) + \frac{1}{\tau_i} \int_0^t e(t)dt + \tau_d \frac{d}{dt} e(t) \right\} \tag{9}$$

PID algorithm in LabVIEW

The control law stated in equation 9 is however not exactly implemented in LabVIEW [1]. The PID expression used by LabVIEW is

$$PID_o = K_p\left\{(SP - PV) + \frac{1}{\tau_i}\int_0^t (SP - PV)dt + \tau_d \frac{d}{dt} PV_f\right\} \tag{10}$$

Integral action

LabVIEW uses Trapezoidal Integration method to implement integral mode as shown in equation 11. It is used to avoid any sharp changes in the integral action for large value of error . It could be seen that the larger the value of error, smaller is the integral action.

$$PI_o(k) = \frac{K_p}{\tau_i}\sum_{i=1}^{k}\left[\frac{e(i) + e(i-1)}{2}\right]\Delta t \tag{11}$$

Derivative action

LabVIEW uses partial derivative approach to implement the derivative action. Here, the the derivation is done on the Process Variable (PV) and not on the error e. This ensures that for large values of error there are no derivative kicks. The same is illustrared in the equation 12

$$PD_o(k) = -K_p\frac{\tau_d}{\Delta_t}(PV_f(K) - PV_f(K-1)) \tag{12}$$

Ziegler-Nichols Rule for Tuning PID Controllers [2]

1. **First Method**
   Ziegler-Nichols give rule for determining values of proportional gain K_p , integral time τ_i , and derivative time τ_d ,based on the step response characteristics of a given plant.In this method one can obtain experimentally the response of a plant to a step input, as shown in figure 3. This method is applicable only when the response to the step input exhibits S-shaped curve.

   As shown in figure 3 by drawing tangent line at inflection point and determining the intersection of the tangent line with the time axis and line $c(t) = u$,we get two constants, delay time L and time constant T.

   Ziegler and Nichols suggested to set the values of K_p, τ_i, τ_d according the formula shown in table 1.

Type of controller	K_p	τ_i	τ_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Table 1: Ziegler-Nichols tuning rule based on step response of plant

Notice that the PID controller tuned by the Ziegler and Nichols rule gives,

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{13}$$

$$= 1.2\frac{T}{L}\left(1 + \frac{1}{2Ls} + 0.5Ls\right) \tag{14}$$

$$= 0.6T\frac{\left(s + \frac{1}{L}\right)^2}{s} \tag{15}$$

Thus the PID controller has a pole at the origin and double zeros at $s = -1/L$.

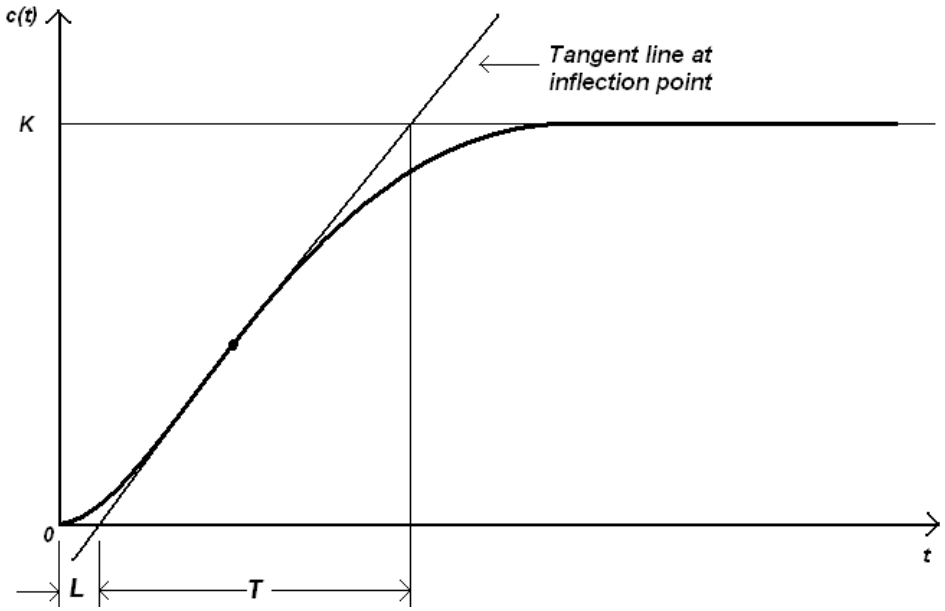

Figure 3: Reaction Curve

2. **Second Method**
The second method is also known as instability Ziegler-Nichols tuning method. This is a closed loop method in which the Integral and Derivative gains of a PID controller are made zero with a unity value of proportional gain. A setpoint change is made and the temperature profile is observed for some time. The temperature would likely maintain a steady-state with some off-set. The gain is increased to a next distinct value (say 2) with a change in the setpoint. The procedure is repeated until the temperature first varies with sustained osscillations. It is necessary that the output (temperature) should have neither under damped nor over damped oscillations. At this particular frequency of sustained oscillations, the corresponding value of K_p is noted and is called as the critical gain K_{cr}. The corresponding period of oscillation is known as P_{cr}. The various P, PI and PID parameters are then calculated with the help of table 2.

Type of controller	K_p	τ_i	τ_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{0.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Table 2: Ziegler-Nichols tuning rule for instability tuning method

NOTE:- While using LabVIEW to tune PID using instability method, put the value of integral time τ_i to be 0, not ∞.

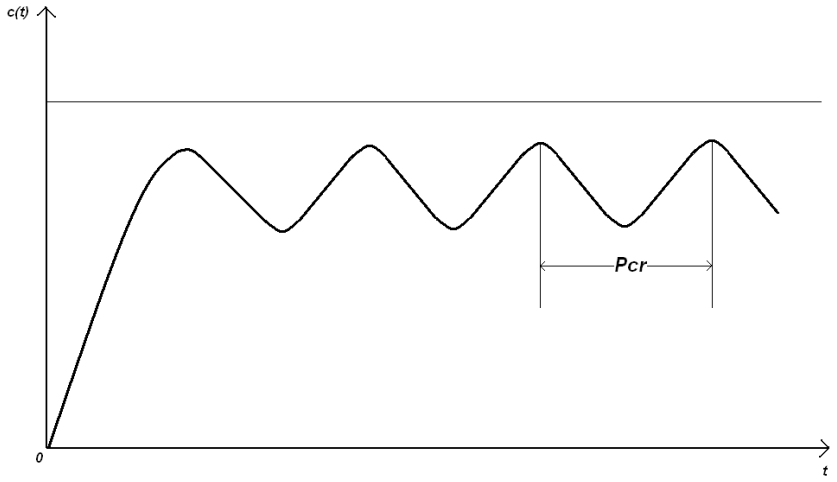

Figure 4: Ziegler-Nichols instability tuning method

Procedure to perform PID experiment

Using the Ziegler-Nichols first method explained earlier, the following values were obtained

$$L = 3.42 \ \ seconds$$
$$T = 50 \ \ seconds$$

For PI

$$K_p = 14.61$$
$$\tau_i = 0.19 \ \ minutes$$

For PID

$$K_p = 17.54$$
$$\tau_i = 0.114 \ \ minutes$$
$$\tau_d = 0.0285 \ \ minutes$$

It should be noted that LabVIEW understands the values of τ_i and τ_d in minutes.

- Implement the PI controller on LabVIEW using the designed values.

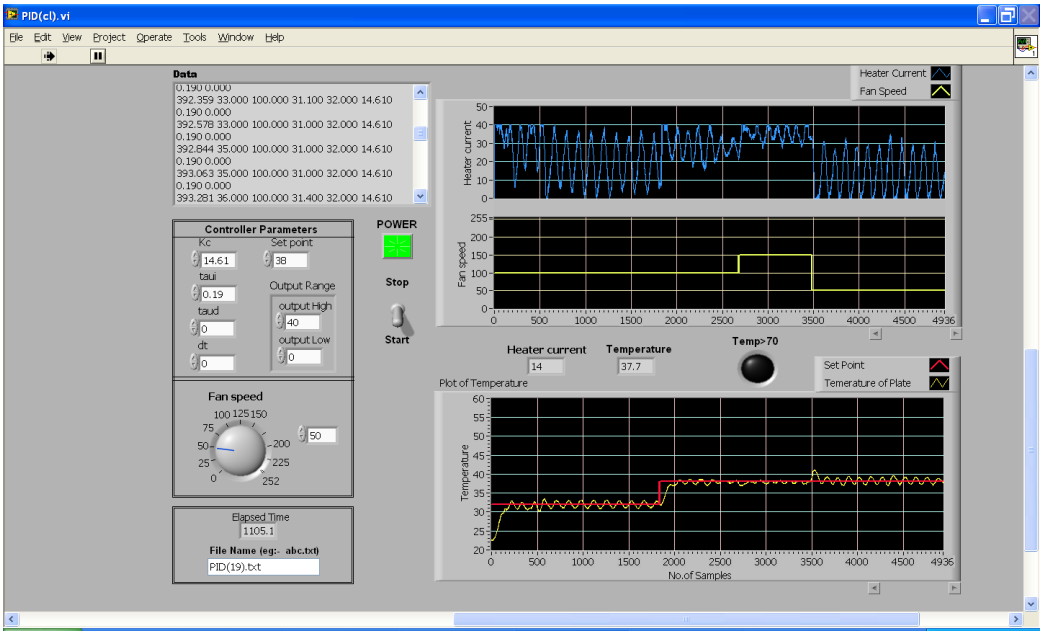

Figure 5: Implementation of a PI controller using Ziegler-Nichols first method

- Implement the PID controller on LabVIEW using the designed values.

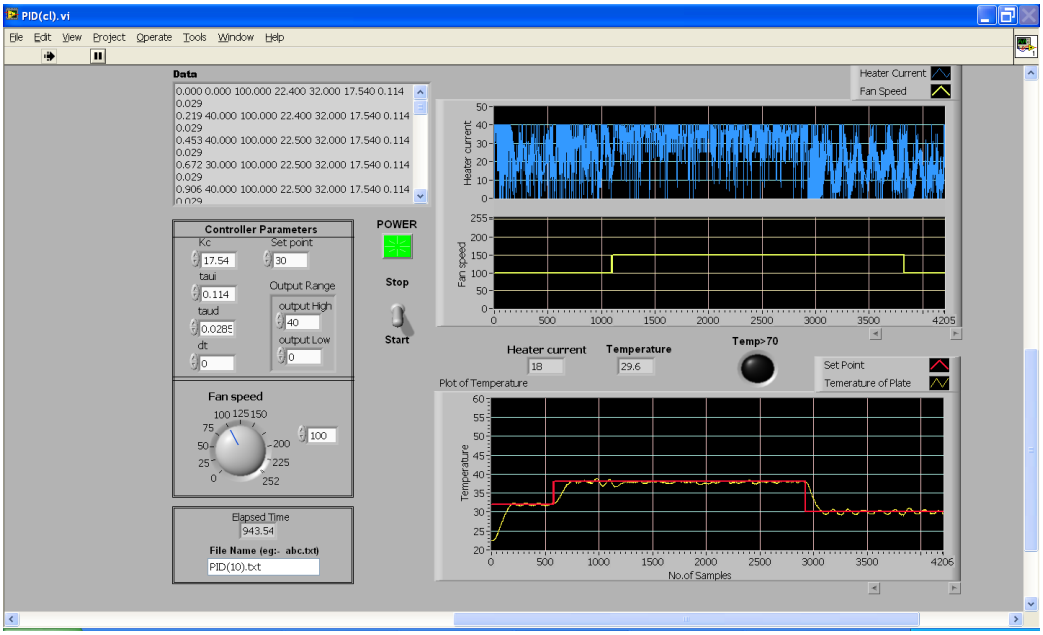

Figure 6: Implementation of a PID controller using Ziegler-Nichols first method

Using the Ziegler-Nichols second method explained earlier, the following values were obtained

$$K_{cr} = 30$$
$$P_{cr} = 19 \ \ seconds$$

For PI

$$K_p = 13.5$$
$$\tau_i = 0.2638 \ \ minutes$$

For PID

$$K_p = 18$$
$$\tau_i = 0.158 \ \ minutes$$
$$\tau_d = 0.0395 \ \ minutes$$

It should be noted that LabVIEW understands the values of τ_i and τ_d in minutes.

- Implement the PI controller on LabVIEW using the designed values.

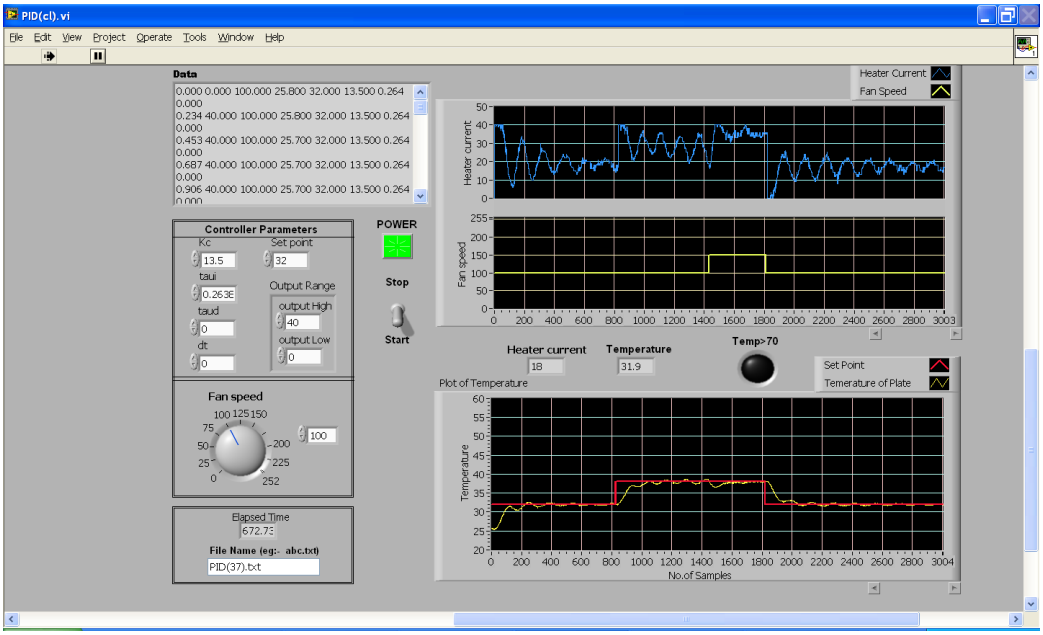

Figure 7: Implementation of a PI controller using Ziegler-Nichols second method

- Implement the PID controller on LabVIEW using the designed values.

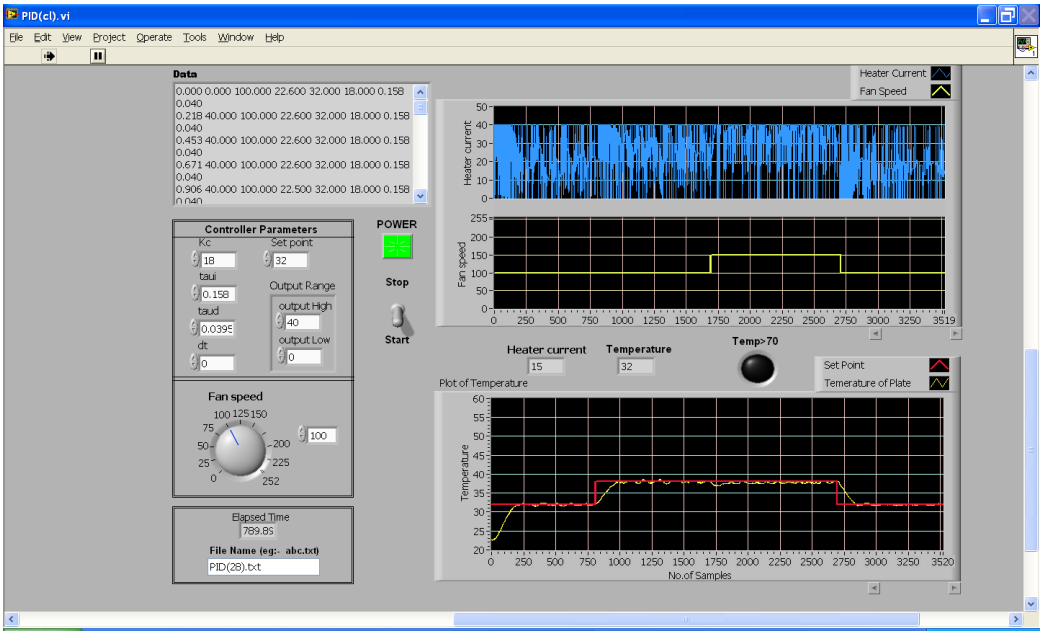

Figure 8: Implementation of PID controller using Ziegler-Nichols second method

References

[1] National Instruments. *PID Control Toolset, User Manual*, 2001. www.ni.com/pdf/manuals/322192a.pdf.

[2] Katsuhiko Ogata. *Modern Control Engineering*. Prentice-Hall of India, 2005.