Demonstrating PID Control Principles using an Air Heater and LabVIEW

Finn Haugen[*], Eivind Fjelddalen[†], Ricardo Dunia[‡], and Thomas F. Edgar[§]

December 6, 2007

Abstract

At Telemark University College in Norway a number (eventually eight) small air heater lab stations have developed for the purpose of effectively demonstrating and/or learning basic PID control skills. This paper describes a number of practical control topics that can be highlighted by this lab station, namely control system performance indexes, control principles, and controller tuning. The control systems are implemented on a PC which runs the graphical programming tool LabVIEW. The analog voltage I/O between the lab station and the PC is implemented with an inexpensive USB-based I/O device. The experiences with this system have been positive.

1 Introduction

Suggestions to renovate the undergraduate process control courses have been recently presented in the literature [2]. Among the suggested actions is "Introduce a number of short laboratory experiences". The suggestions in [2] are based on experiences in the USA, but it can be assumed that the experiences are quite similar and that the suggestions therefore are valid also in other countries.

Along these lines, at Telemark University College in Norway a number of small air heater lab stations have been constructed to provide demonstration and/or learning of basic PID control topics. (Eventually eight lab stations will be constructed, so that eight student groups can be active simultaneously.) The lab station can be controlled with e.g. a PC with NI LabVIEW[1] and NI USB-6008 I/O device for analog (voltage) I/O, see Figure 1. Its dimensions are 40 cm x 45 cm, so it can be used as a "desktop lab station". The components costs about 900 USD (in Norway).

[*]Telemark University College, Norway. E-mail: finn.haugen@hit.no
[†]Telemark University College, Norway. E-mail: eivind.fjelddalen@hit.no
[‡]University of Texas, Austin. E-mail: rdunia@gazoo.che.utexas.edu
[§]University of Texas, Austin. E-mail: tfedgar@austin.utexas.edu
[1]NI = National Instruments

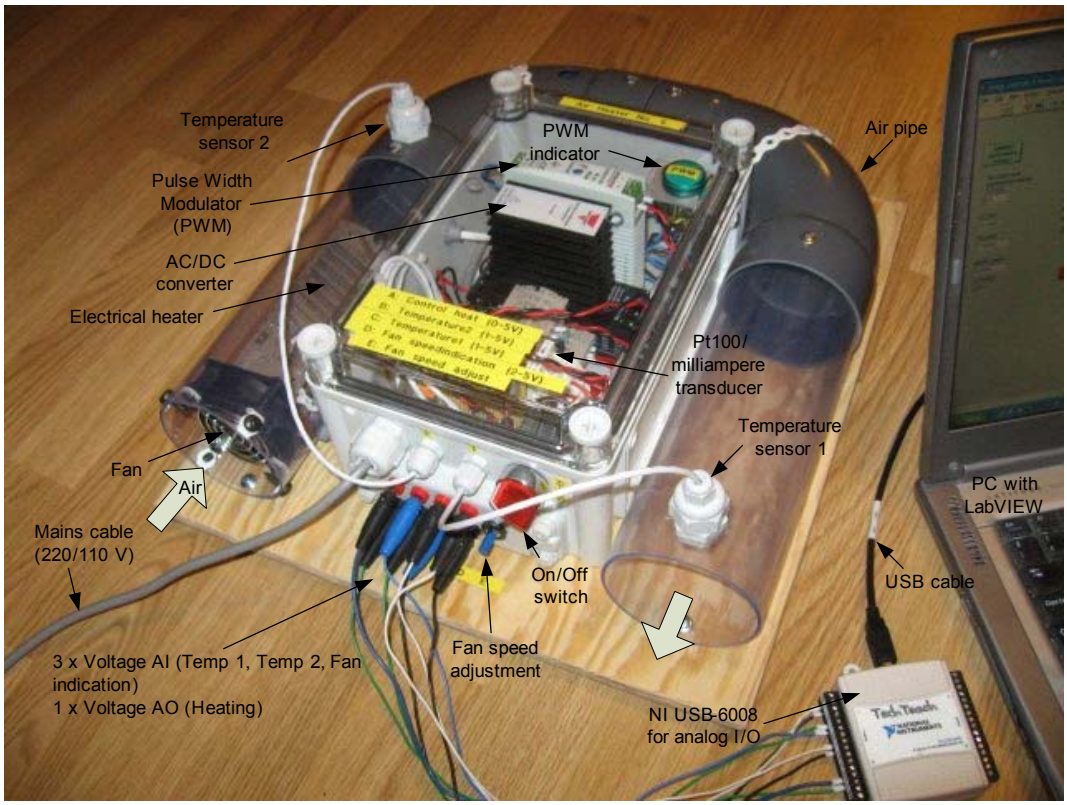

Figure 1: The air heater lab station with NI USB-6008 analog I/O device

The system has been in use in several automation oriented courses in the Fall semester 2007. The students work on the assignments in groups of two or three students. The experiences have been positive.

It is assumed that the students have basic skills in LabVIEW programming before they start working on the assignments, but it is fully possible that students run premade LabVIEW applications. At Telemark University College LabVIEW skills are taught by requiring the students to work through parts of a LabVIEW tutorial [3].

The purpose of this paper is to describe topics that are focused in present student assignments based on the air heater together with the flexible and powerful LabVIEW environment. Of course there is a large number of other interesting topics that may be focused. The final section of this paper indicates some topics that will be focused in future assignments (these have not been implemented yet). Additional interesting topics that may be focused may be found in e.g. [4].

This paper contains the following sections:

- Section 1: Introduction
- Section 2: System Description
- Section 3: Block Diagram of Control System
- Section 4: Performance Criteria
- Section 5: Where to Measure for Control
- Section 6: Measurement Noise and Filtering
- Section 7: Control Principles
- Section 8: Controller Tuning
- Section 9: Cascade Control
- Section 10: Feedforward Control
- Section 11: Summary
- Section 12: Further Use of the Lab Station

2 System Description

Air Tube: The air pipe is made of plastic. The effective length of the pipe from the inlet to the outlet is approximately 1 m.

Fan: A fan makes air flow through the pipe. The fan is operated manually with a knob. The fan position is indicated or measured by a voltage signal which is in the range [2 V, 5 V] (min, max fan speed). This voltage signal can be measured between two terminals. The normal fan speed is defined to be the maximum speed. (We do not know the actual volumetric flow, but this information is not necessary in our applications.)

Heater: The air is heated by an electrical heater. The supplied power is controlled by an external voltage signal in the range [0 V, 5 V] applied to a Pulse Width Modulator (PWM) which connects/disconnects the mains voltage to the heater. The PWM signal is indicated by a lamp on the lab station. The PWM device requires 24 VDC power supply, which is produced by an AC/DC converter.

Temperature Sensors: Two Pt100 temperature elements are available. They have been calibrated equally. The sensor signals are available as voltage signals in the range 1 – 5 V at their respective terminals. (The original sensor signal is a current signal in the range 4 – 20 mA which is sent through a 250 Ω resistor, causing a voltage signal 1 – 5 V.) This voltage range corresponds to the temperature range 20 – 60 °C with a linear relation between the ranges. The default sensor position is defined to be the outermost position in the pipe.

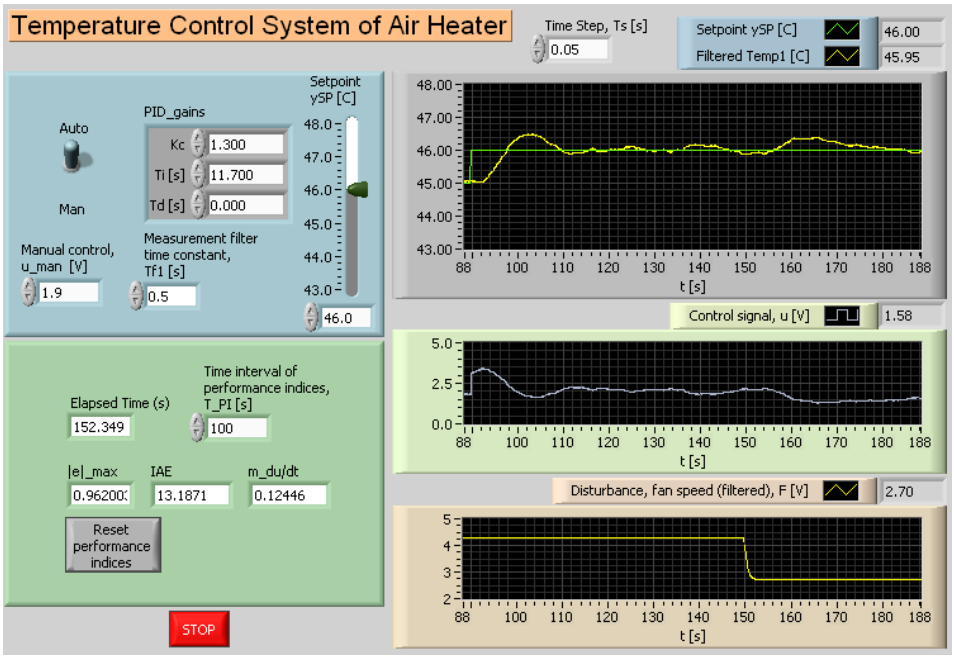

Figure 2: A typical front panel of a LabVIEW program that implements the temperature control system

Analog I/O: Any analog I/O device supporting the voltage ranges defined above can be used in measurement and control applications, e.g. the NI USB-6008 device, see Figure 1.

Controllers: As controller function the **PID Advanced** function in LabVIEW is used. Of course any other controller device can be used as long as it complies with the analog I/O range defined above.

Software: In the experiments reported in this paper the analog I/O functions, control, and plotting was implemented in LabVIEW 8.5. The sampling time (cycle time of the While Loop that makes the program run continuously) was set to 0.05 sec. Figure 2 shows a typical front panel of a LabVIEW program that implements the temperature control system.

Additional Information: See http://home.hit.no/~finnh/air_heater.

3 Block Diagram of Control System

Figure 3 shows a block diagram of the temperature control system as it may be implemented on a PC with LabVIEW and the NI USB-6008 I/O device.

Below are comments to some of the blocks in the block diagram:

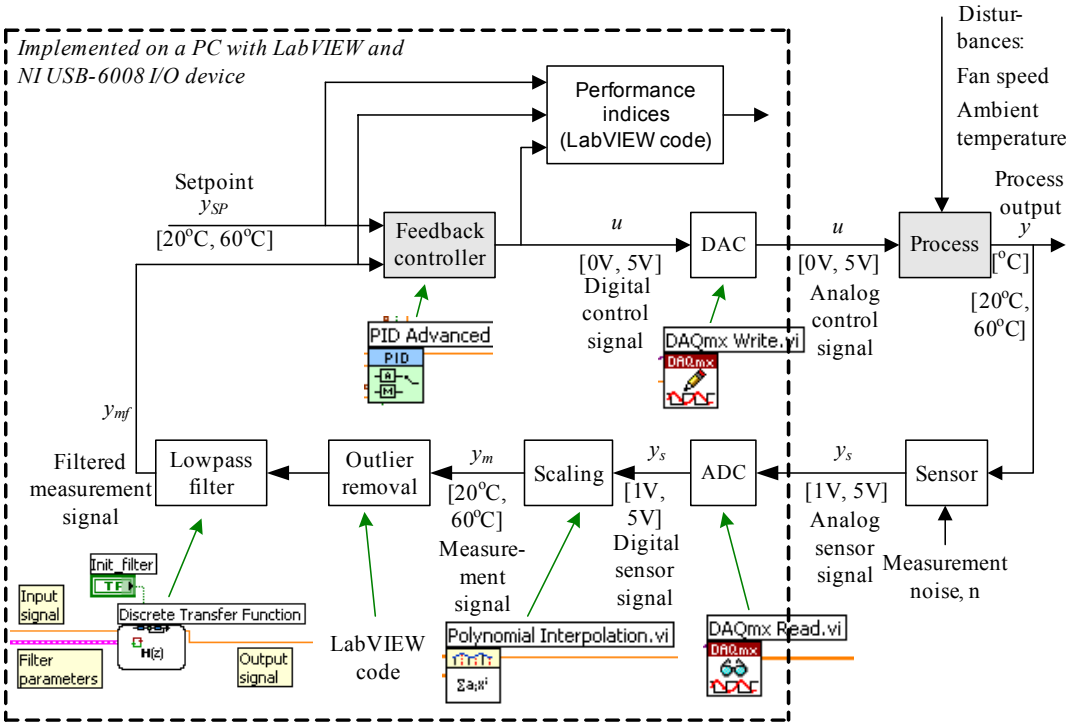

Figure 3: Block diagram of the temperature control system

- **PID Advanced function**: This function is on the PID Palette in LabVIEW. It implements a PID controller function on the so-called ideal (summation) form where the P, I, and D terms are summed and with controller parameters K_c (controller gain), T_i (integral time in minutes) and T_d (derivative time in minutes). It implements anti-windup, and it can be set in automatic/manual mode. Options are nonlinear gain and reduced setpoint weight in the proportional term. The PID Advanced function has no inbuilt lowpass filter.

- **Scaling**: There are scaling functions on the PID Palette in LabVIEW, but these functions assume scaling between engineering units and *percent*. However, in our application we do not use the percent as unit. Therefore we use the **Polynomial Interpolation** function on the Mathematics Palette in LabVIEW in stead.

- **Removing outliers**: It turns out that the temperature measurements contains outliers in the form of spikes.[2] These outliers must be removed so that they do not propagate through the control loop. Although the measurement lowpass filter (se next item) will attenuate the effect of the spikes, they will

[2]The outliers are related to the temperature transmitter/transducer, not LabVIEW or the NI USB-6008 I/O device.

still give some effect. It is better to remove the spikes.[3] In our application this has been implemented as follows:

1. Calculate continuously the mean value, m_T, of the temperature of the 20 most recent samples using the **Mean PtByPt** function.[4]
2. If the present temperature measurement, $T(t_k)$, deviates from m_T by more than a specified value, D_T, then substitute $T(t_k)$ by m_T. We have set D_T to 0.2 °C.

Figure 4 shows how we have implemented this method in LabVIEW.

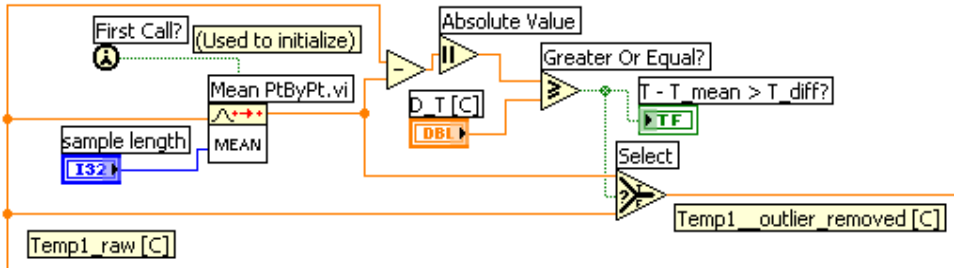

Figure 4: Implementation of removal of outliers in LabVIEW

Figure 5 shows how the method works. The lower plot in Figure 5 shows when outliers have been detected.

- **Measurement filter,** which is in the form of the following digital (or discrete-time) filter

$$y_{m_f}(t_k) = (1-a)y_{m_f}(t_{k-1}) + ay_m(t_k) \tag{1}$$

This filter is denoted the *exponentially weighted moving average (EWMA) filter* [5]. It is an IIR[5] filter.

Students learning control are typically familiar with first order systems, here represented by the standard Laplace transform based transfer function,

$$H_f(s) = \frac{1}{T_f s + 1} = \frac{\mathcal{L}\left\{y_{m_f}(t)\right\}}{\mathcal{L}\left\{y_m(t)\right\}} \tag{2}$$

where T_f [s] is the time-constant. (2) represents a continuous-time lowpass filter. It is informative to regard the EWMA filter as a discretized version of

[3] Measurement outliers can be observed in other control systems, of course. One example is dynamic positioning (DP) systems which implements position control of ships. The outliers origin from e.g. faulty GPS signals.

[4] PtByPt means Point By Point. On the Signal Processing palette of LabVIEW there are a number of Point By Point function that implement "batch" functions, as the **mean** function, as an online or real-time function.

[5] IIR = Infinite Impulse Response

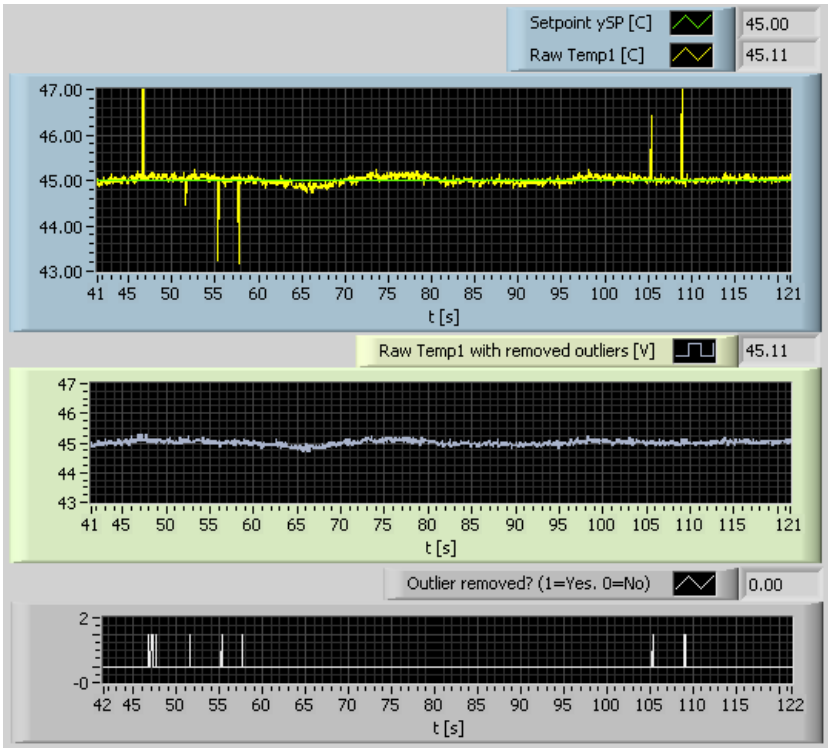

Figure 5: The effect of removing outliers from the temperature measurement signal

this continuous-time filter. It can be shown [5] that discretizing (2) with the Euler backward difference method with sampling time T_s yields (1) with the following relation between the filter parameter a and the time-constant and the sampling time:

$$a = \frac{1}{\frac{T_f}{T_s} + 1} \tag{3}$$

Hence, by specifying T_f, or alternatively the filter bandwidth f_b [Hz] since

$$T_f = \frac{1}{2\pi f_b} \tag{4}$$

the EWMA filter parameter a is given. One benefit of using T_f in stead of a as a filter specification, is that T_f represents the filter dynamics in a familiar way. Furthermore, specifying the filter time constant seems to be the most common way to specifiy the filtering property of measurement filters in industrial automation equipment.[6]

In our application the EWMA measurement filter is implemented with the **Discrete Transfer Function** function on the Control Design / Implemen-

[6] This is the experience of the first author of this paper.

tation palette of LabVIEW. The user (student) can adjust T_f on the front panel of the LabVIEW program, and the EWMA filter parameter a is then calculated according to (3).

4 Performance Indices

4.1 Introduction

[5] presents a number of performance criteria for control systems. These criterias are basically as follows:

1. Control error is within an acceptable limit for any reasonable setpoint value and any disturbance value. In other words, the setpoint tracking and disturbance compensation must be acceptable.
2. The control signal is sufficiently smooth.
3. The closed-loop control system has acceptable stability.
4. The above criteria must be satisfied despite changes in process conditions. In other words, the control system must be robust.

The next section defines *performance indices* which can be used to quantify the above criterias. (The robustness criterion is not quantified here, but it is focused in the Gain Scheduling Section of this paper.) The performance indices presented are all *model-free* and can be calculated online in the LabVIEW program.

4.2 Selected Performance Indices

Below are performance indices which quantify the first three performance criteria listed in the previous section. All of these indices can be used to compare different experimental conditions, as different controller functions, tunings or process conditions. Some of them also provide useful information in themselves, for example the $|e|_{\max}$ index.

- **Maximum of absolute value of control error**, $|e|_{\max}$, cf. criterion no. 1. The control error is defined as the deviation between the setpoint and the measured process output:

$$e = y_{SP} - y_m \tag{5}$$

  $|e|_{\max}$ is perhaps the most important of the performance indices because typically product quality is directly related to the maximum deviation from the setpoint. The variation of the control error is not important as long as the error is within the acceptable limit. Figure 6 illustrates reading off $|e|_{\max}$.

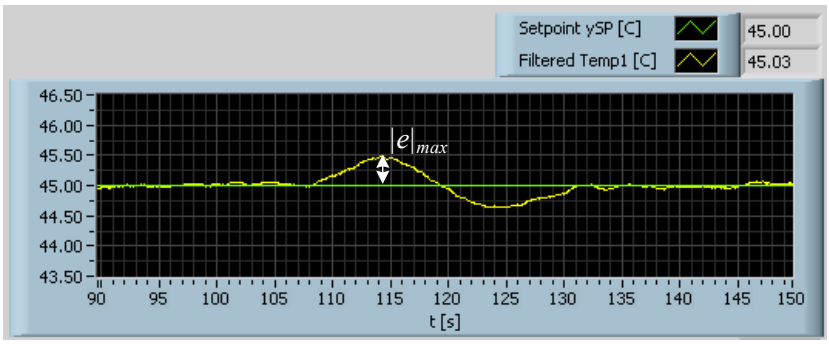

Figure 6: Reading off $|e|_{\max}$

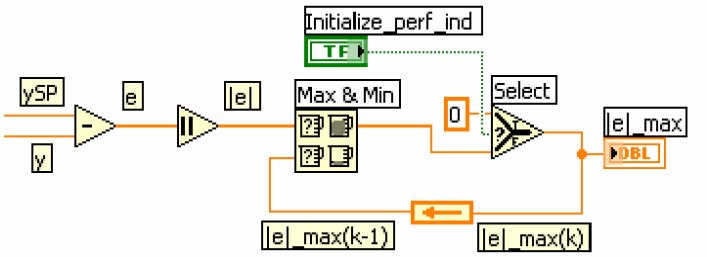

Figure 7: Calculating $|e|_{\max}$ continuously in LabVIEW

Figure 7 shows how $|e|_{\max}$ can be calculated on-line in LabVIEW.

The T_{PI} variable shown in Figure 7 is the time interval of the averaging. N_{PI} is the number of samples that corresponds to T_{PI}. Note that Initialize input to the **Mean PtByPt** function very conveniently sets the data set used internally to calculate the mean value to the present input value of the function.

- **Integrated Absolute value of control Error**, IAE, cf. criterion no. 1. It is frequently used in the litterature to compare different control functions. The IAE is

$$IAE = \int_{t_i}^{t_f} |e|\, dt \tag{6}$$

where t_i is the initial (or start) time and t_f is the final time. In the ideal IAE index t_f is infinity but in practical implementations t_f must of course have a finite value. The less IAE value, the better control (assuming that the behaviour of the control signal has no weight).

Figure 8 shows how (6) can be implemented in LabVIEW. The T_PI variable is the period or duration of the IAE calculation.

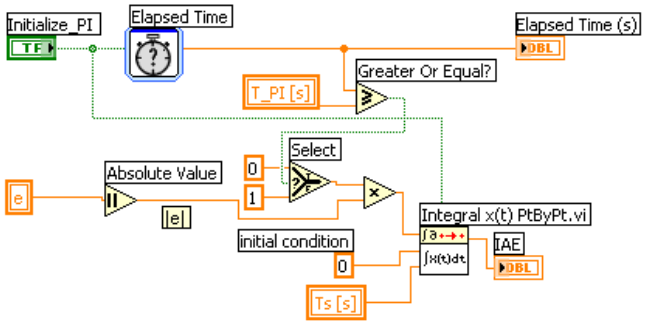

Figure 8: Implementation of (6) in LabVIEW

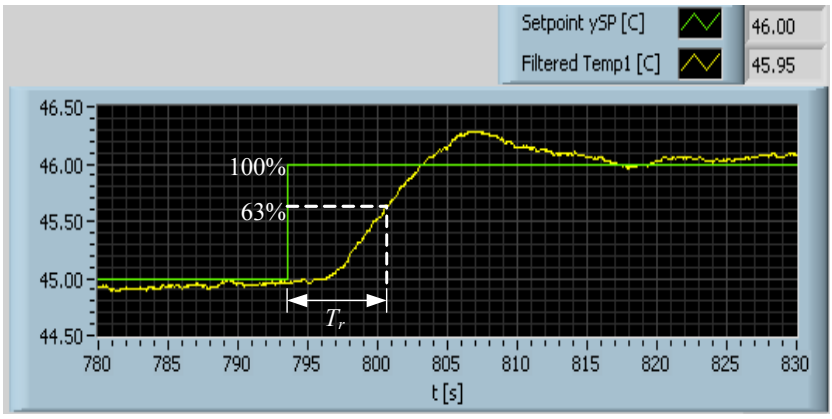

Figure 9: Response-time T_r is the 63% rise time of the response in the process output variable after a setpoint step change

- **The response-time**,T_r, expresses the quickness of the control system, cf. criterion no. 1. T_r is the 63% rise time of the response in the process output variable after a setpoint step change, see Figure 9. T_r can also be denoted the *time-constant* of the system.[7]

- **The settling-time**,T_s, also expresses the quickness of the control system. It can be used to express how quick the control system brings the control error within specified limits after a specified change of the setpoint or the disturbance.

  Using the air heater as an example, the settling time can be the time that elapses from a sudden change of the fan speed (disturbance) from maximum speed to minimum speed – this is a maximum change of the disturbance – until

[7] The time-constant originally refers to the parameter T of this first order transfer function $K/(Ts+1)$. For this model T is the exact 63 % rise time.

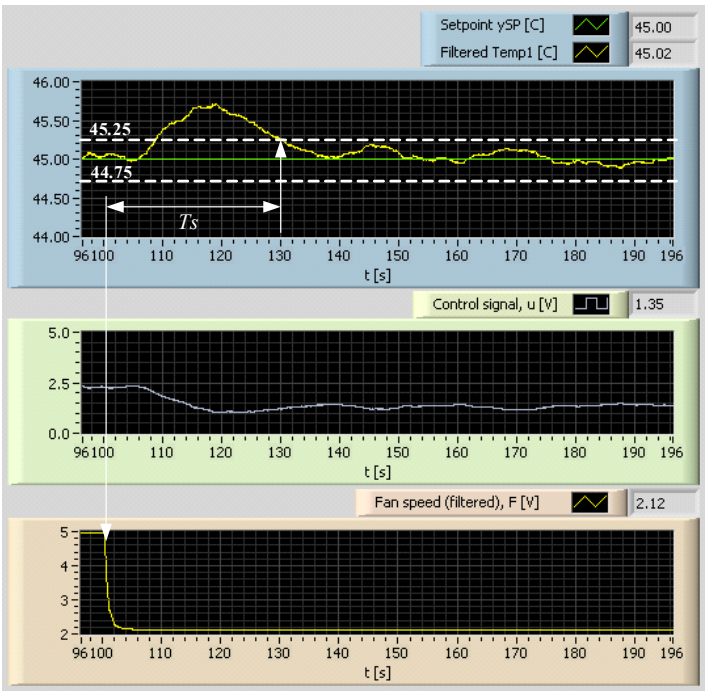

Figure 10: Reading off the settling time T_s for the control system of the air heater

the temperature control error is back within 0.25 °C, see Figure 10. From the figure we see that the settling time is $T_s = 30$ s.

- **Mean of absolute value of time-derivative of control signal**, $m_{du/dt}$, defined by

$$m_{du/dt} = \left| \frac{du}{dt} \right|_{\text{mean}} \tag{7}$$

where the mean value in practical applications must be calculated over some finite time interval. This index expresses the time-variations of the control signal, cf. criterion no. 2 presented in Section 4.2. The smaller value, the smoother control signal which is benefical since it may indicate reduced equipment wear. The time-derivative may be calculated using the Euler backward difference method:

$$\frac{du(t_k)}{dt} \approx \frac{u(t_k) - u(t_{k-1})}{T_s} \tag{8}$$

Figure 11 shows how (7) including (8) can be implemented in LabVIEW.

Figure 12 shows plots of the control signal with a PI controller ($K_p = 1.2$, $T_i = 15.0$ s, $T_d = 0$ s) and a PID controller ($K_p = 1.2$, $T_i = 15.0$ s, $T_d = 2$ s). $m_{du/dt}$ was calculated from the most recent period of 5 seconds of samples.

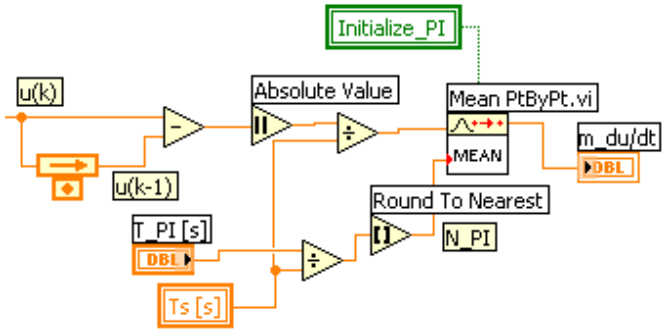

Figure 11: Implementation of (7) in LabVIEW

The results are as follows:

$$\text{PI controller: } m_{du/dt} = 0.05 \tag{9}$$

$$\text{PID controller: } m_{du/dt} = 2.5 \tag{10}$$

(Hence the derivative term causes a tremdous increase in the variations of the control signal.)

- **The amplitude decay ratio,** AR, expresses the stability of the control system, cf. criterion no. 3. AR is the ratio between subsequent peaks (in the same direction) of the oscillations in e.g. the process output variable after a setpoint or disturbance step change, see Figure 13. Thus,

$$AR = \frac{A_2}{A_1} \tag{11}$$

The less amplitude decay ratio, the better stability. Ziegler and Nichols [7] who published famous tuning rules in the 1940s claimed that satisfactory damping corresponds to a decay ratio of approximately 1/4.

5 Where to Measure for Control?

Feedback control makes the process measurement signal become equal to the setpoint (assuming that the setpoint and the disturbances are constant and that the controller has integral action). It is crucial to mount the sensor at a position at a position so that it represents accurately the process variable to be controlled. This may sound obvious, but it is nonetheless very important.

The air heater has two temperature sensors, and they can be placed at different positions in the air pipe, see Figure 1. Assume that the temperature at the outer

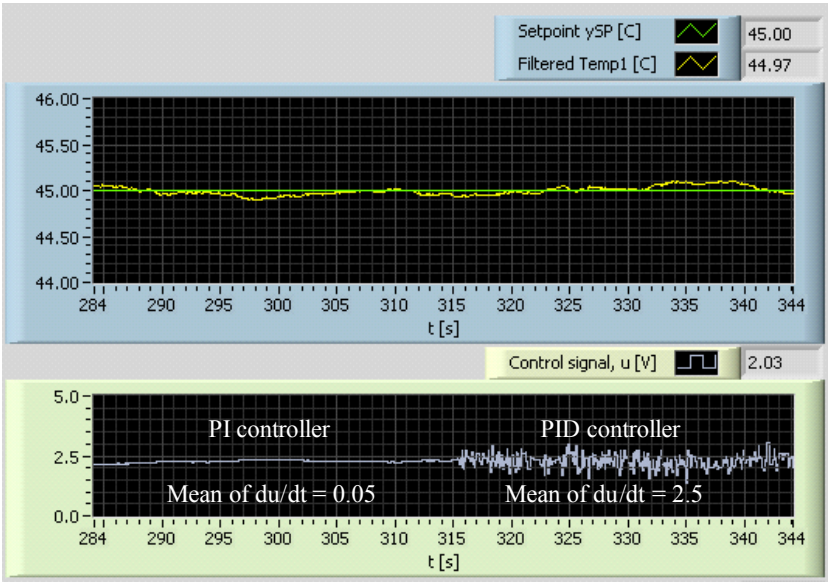

Figure 12: Reponse in the control signal with PI controller and with PID controller

position is to be controlled, and that the sensor – sensor no. 2 – which delivers the measurement signal to the controller, is placed at the inner position of the pipe. Sensor no. 1 is placed at the outer position, so that we can monitor the difference between the two temperatures.

Figure 14 shows plots of the two temperatures together with the setpoint which was 45 °C. (The controller is a PI controller.) From the plots we see that the temperature at the inner position is at the setpoint, while the temperature at the outer position is approximately *1.3* °C *lower*, which is due to cooling and air leakage through the pipe. The lesson to learn is that, if the temperature at the outer position is to be controlled, it is not wise to place the sensor at the inner position.

6 Measurement Noise and Filtering

In a feedback control system measurement noise is propagated via the controller to the control signal, causing variations in the control signal. The derivative term of the controller amplifies these variations. These variations can be reduced with a measurement lowpass filter. The PID Control Palette in LabVIEW contains the **PID Control Input Filter** function which implements a fifth order FIR[8] filter. This filter is inflexible since the order and the filter parameters can not be changed. In other words, it can not be adjusted to the particular need of each application. Lab-

[8] Finite Impulse Response

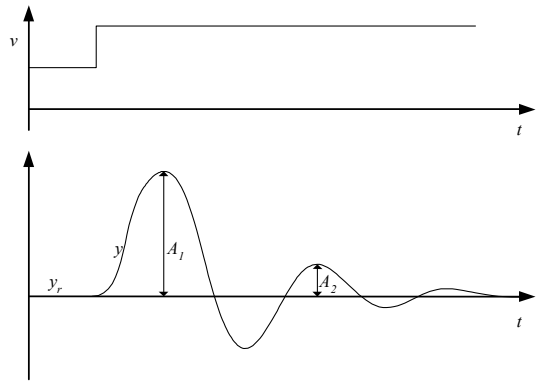

Figure 13: The amplitude decay ratio A_2/A_1 can be used to characterize the stability of a control loop

VIEW contains several other filters that can be used. We have chosen to implement the first order discrete-time EWMA filter given by (1) using the **Discrete Transfer Function** function on the Control Design / Implementation palette of LabVIEW. This function is flexible, and makes it possible to adjust the filter parameters online without the filter states being reset to zero.

Figure 15 illustrates the impact of the measurement noise on the control signal (from the PID controller) for the following three cases. The $|du/dt|_{\text{mean}}$ performance index is calculated in each case.

1. **PID controller. No filter.** PID parameters:

$$K_p = 1.3,\ T_i = 11.7 \text{ s},\ T_d = 1.0 \text{ s} \tag{12}$$

   Performance index was

$$\left|\frac{du}{dt}\right|_{\text{mean}} = 42 \tag{13}$$

2. **PID controller. Lowpass filter.** PID parameters as above. Filter time constant:

$$T_f = 0.5 \text{ s} \tag{14}$$

   Performance index was

$$\left|\frac{du}{dt}\right|_{\text{mean}} = 1.5 \tag{15}$$

3. **PI controller. Lowpass filter.** PI parameters:

$$K_p = 1.3,\ T_i = 11.7 \text{ s} \tag{16}$$

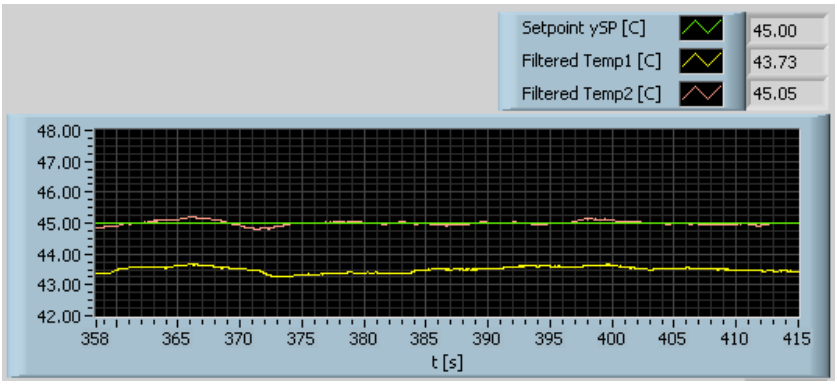

Figure 14: Plots of Temperature 1 at the outer position and Temperature 2 at inner position of the air pipe. Temperature 1 is used by the controller.

Filter time constant:

$$T_f = 0.5 \text{ s} \tag{17}$$

Performance index was

$$\left|\frac{du}{dt}\right|_{\text{mean}} = 0.12 \tag{18}$$

The conclusion from these experiments is that the variations of the control signal are reduced

- if the derivative term of the controller is removed, and
- if a measurement lowpass filter is introduced.

These observations are general. According to [6]: "PI controllers are particularly common, since derivative action is very sensitive to measurement noise".[9]

7 Control Principles

The basic control principles described below were implemented and compared in terms of performance indices. (Cascade control and Feedforward control could have been presented in the present section, too, but in stead they are presented in their respective sections.) In all cases the setpoint was kept constant at 45 °C. The fan speed was reduced from maximum to minimum at time $t = 100$ s, causing a serious disturbance on the process. The performance indices were calculated over an interval

[9] According to Ingvar N. Westengen (MSc) at the Yara industrial plant in Norway more than 90 % of their control loops run with PI controllers, due to the measurement noise problem of the PID controller.

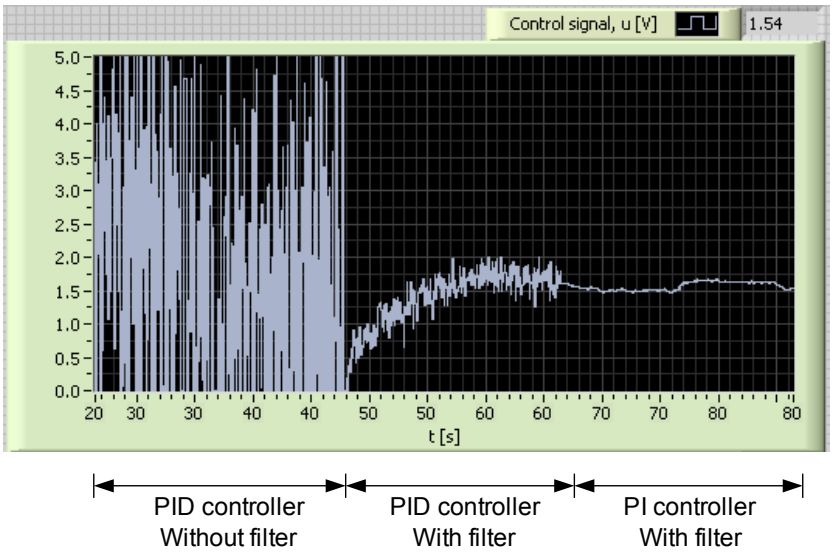

Figure 15: The impact of the measurement noise on the control signal (from the PID controller) for three cases

of 150 sec. The responses are shown in the subsequent figures, and the performance indices are shown in Table 1.

- **Blind control**: The control signal is kept constant regardless of the control error. This is the same as open-loop control. See Figure 16.
- **Manual feedback control**: The control signal is adjusted manually by the operator[10] who tries to keep the control error as small as possible while observing the temperature reponse continuously. See Figure 17.
- **Automatic feedback control**: The control signal is adjusted automatically by a PI controller implemented in the LabVIEW program. The controller parameters were $K_p = 1.3$ and $T_i = 11.7$ s. See Figure 18.

The performance indices are shown in Table 1. Automatic feedback control with PI controller has the best performance indices in this test. The results are quite general. However, a trained operator may perform excellently, exploiting his or her knowledge about the process. On the other hand using an operator as a controller may be an expensive solution.

[10] Finn Haugen. He was not trained for this task. It was his second trial. A trained operator will probably perform better.

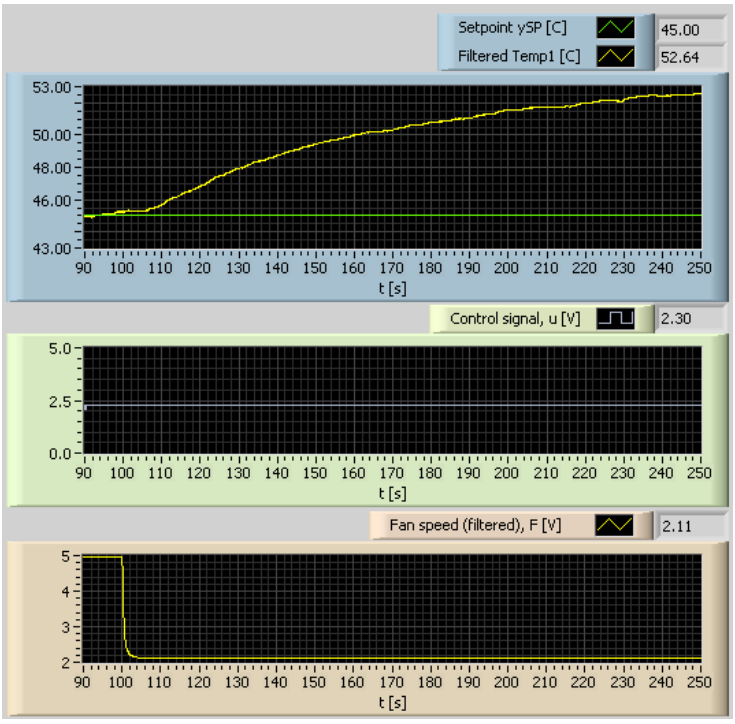

Figure 16: Blind control

	Blind control	**Manual feedback**	**Automatic feedback**
$\|e\|_{\max}$	7.6	1.3	0.8
IAE	696	71	18
$m_{du/dt}$	0	0.15	0.08

Table 1: Performance indices

8 Controller Tuning

8.1 What is Good Tuning?

For a PID controller "good tuning" is usually synonymous with "satisfactory stability" of the control loop. Ziegler and Nichols [7] defined this as one quarter decay ratio between subsequent response peaks in the same direction. Ziegler and Nichols used this as a stability criterion when they derived their PID tuning rules. If you want better stability than expressed by the one quarter decay ratio, then you must tune so that the decay ratio becomes less than one quarter.

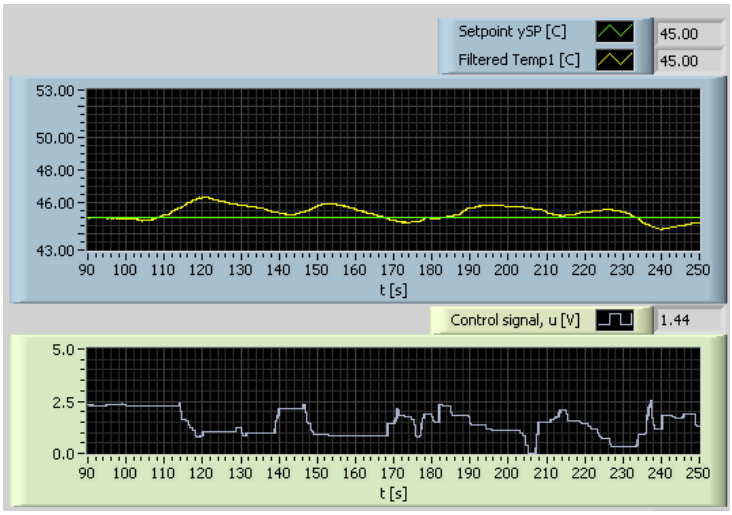

Figure 17: Manual feedback control

8.2 P-I-Tuning Method

Here an assumeably intuitive and simple tuning method is described. For reference it will be denoted the P-I-tuning method here. In this method the control system will not be driven into sustained oscillations in the tuning phase, as in the Ziegler-Nichols' closed loop method, cf. Section 8.3. The method is based on experiments on the closed loop system, see Figure 19. The procedure described below assumes a PI controller, which is the most commonly used controller function.

1. Bring the process to or close to the normal or specified operation point by adjusting the nominal control signal u_0 (with the controller in manual mode).

2. Ensure that the controller is a P controller with $K_p = 0$ (set $T_i = \infty$ and $T_d = 0$). Increase K_p until the control loop gets satisfactory stability as seen in the response in the measurement signal after e.g. a step in the setpoint or in the disturbance (exciting with a step in the disturbance may be impossible on a real system, but it possible in a simulator). If you do not want to start with $K_p = 0$, you can try $K_p = 1$ (which is a good initial guess in many cases) and then increase or decrease the K_p value until you observe a slight overshoot but a well damped response.

3. Set the integral time T_i equal to

$$T_i = 1.5T_{ou} \tag{19}$$

where T_{ou} is the time between the first overshoot and the first undershoot of the step response (a step in the setpoint) with the P controller, see Figure 20.

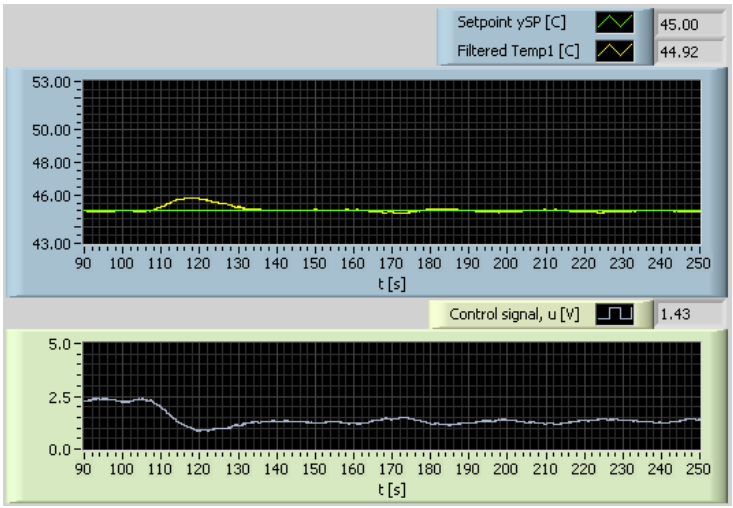

Figure 18: Automatic feedback control (PI controller)

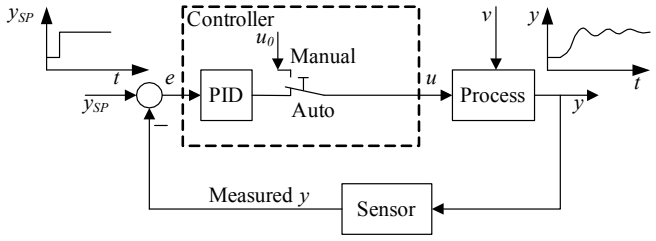

Figure 19: The P-I-Tuning method must be applied to the established control system.

4. Check the stability of the control system by applying a setpoint step. Because of the introduction of the I-term, the loop with the PI controller in action will probably have somewhat reduced stability than with the P controller only. If you think that the stability has become too poor, try reducing K_p somewhat, e.g. reduce it to 80% of the original value.

Figure 21 shows the temperature response due to a setpoint step with a P-controller (cf. item 1 in the procedure above) with $K_p = 1.6$. From this response we find

$$T_{ou} = 10.0 \text{ sec} \tag{20}$$

from which we get

$$T_i = 1.5 \cdot 10 = 15.0 \text{ s} \tag{21}$$

For K_p we try

$$K_p = 0.8 \cdot 1.6 = 1.2 \tag{22}$$

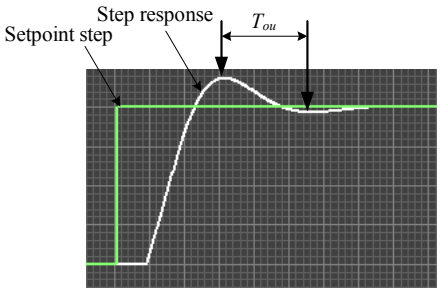

Figure 20: The P-I tuning method: Reading off the time between the first overshoot and the first undershoot of the step response with P controller

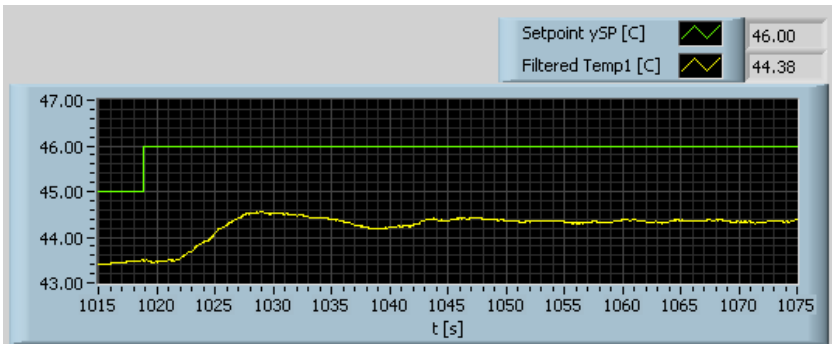

Figure 21: The P-I tuning method: Response in the temperature with P controller with $K_p = 1.5$.

Figure 22 shows the response with the tuned PI-controller, i.e. with

$$K_p = 1.2 \text{ and } T_i = 15 \text{ s} \tag{23}$$

The response indicates that the stability of the control system is ok.

8.3 Ziegler-Nichols' Ultimate Gain Method

The famous *Ziegler-Nichols' Ultimate Gain tuning method* [7] is based on experiments executed on the control loop – real or simulated. The principle of the method is to find the controller gain – denoted the ulitmate gain K_{p_u}, of a *P controller* that keeps the control system at the stability limit, where any signal in the loop shows *sustained oscillations*. It is important that K_{p_u} is found *without the actuator being driven into any saturation limit* (maximum or minimum value) during the oscillations. If such limits are reached, you will find that there will be sustained oscillations for any (large) value of K_p, e.g. 1000000, and the resulting K_p-value (as calculated from the Ziegler-Nichols' formulas, cf. Table 2) is useless (the control system will

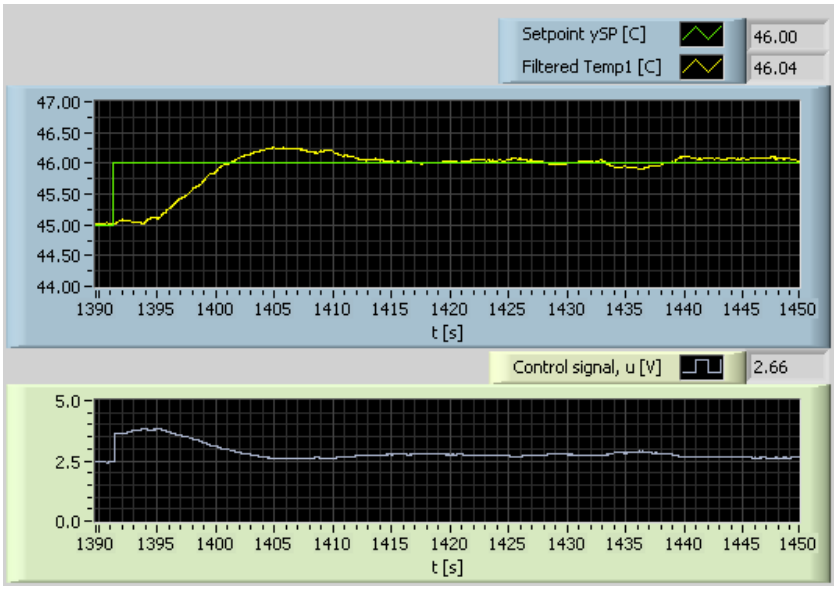

Figure 22: The P-I tuning method: Response in the temperature with a PI controller with $K_p = 1.2$ and $T_i = 15$ s.

probably be unstable). One way to state this is that K_{p_u} must be the *smallest* K_p value that drives the control loop into sustained oscillations.

In addition to noting the ultimate gain, you must measure the *ultimate (or critical) period* P_u of the sustained oscillations.

The controller parameter are calculated according to Table 2.

	K_p	T_i	T_d
P controller	$0.5K_{p_u}$	∞	0
PI controller	$0.45K_{p_u}$	$\frac{P_u}{1.2}$	0
PID controller	$0.6K_{p_u}$	$\frac{P_u}{2}$	$\frac{P_u}{8} = \frac{T_i}{4}$

Table 2: Formulas for the controller parameters in the Ziegler-Nichols' closed loop method.

How does the Ziegler-Nichols' method work with the temperature system? Figure 23 shows the oscillations in the tuning phase with the (ultimate) gain

$$K_{p_u} = 3.0 \tag{24}$$

From 23 we can find the ultimate period

$$P_u = 14 \text{ s} \tag{25}$$

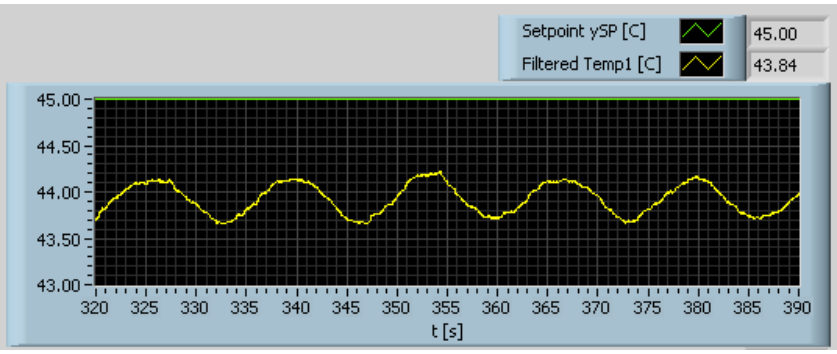

Figure 23: Ziegler-Nichols' Ultimate Gain method: The oscillations in the tuning phase.

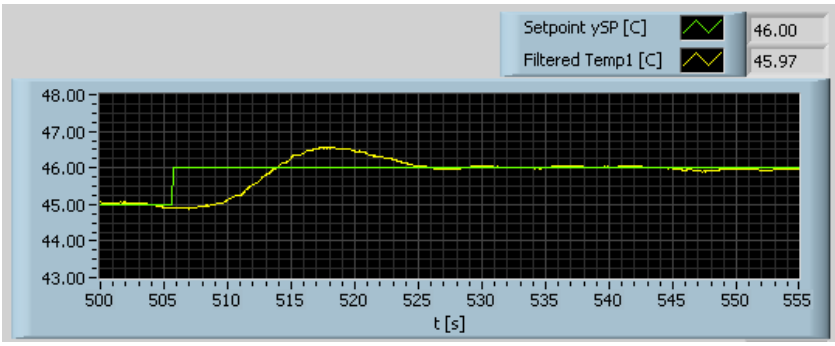

Figure 24: After Ziegler-Nichols' Ulitmate Gain method: Temperature response due to a setpoint step with a PI controller with $K_p = 1.35$ and $T_i = 11.7$ s.

Let us select a PI controller. From Table 2 we get the following PI parameters:

$$K_p = 0.45 \cdot 3.0 = 1.35 \tag{26}$$

$$T_i = \frac{14 \text{ s}}{1.2} = 11.7 \text{ s} \tag{27}$$

Figure 24 shows temperature response due to a setpoint step with the above PID parameter values. The control system shows satisfactory stability.

8.4 Relay-based Tuning

The Åstrøm-Hägglund's Relay-based method [1] can be regarded as a smart, practical implementation of the Ziegler-Nichols' closed loop method described in Chapter 8.3. In the Ziegler-Nichols' method it may be time-consuming to find the smallest controller gain K_p which gives sustained oscillations. You also have to avoid the control signal to reach a maximum or minimum value during the tuning. These problems are eliminated with the Relay-method.

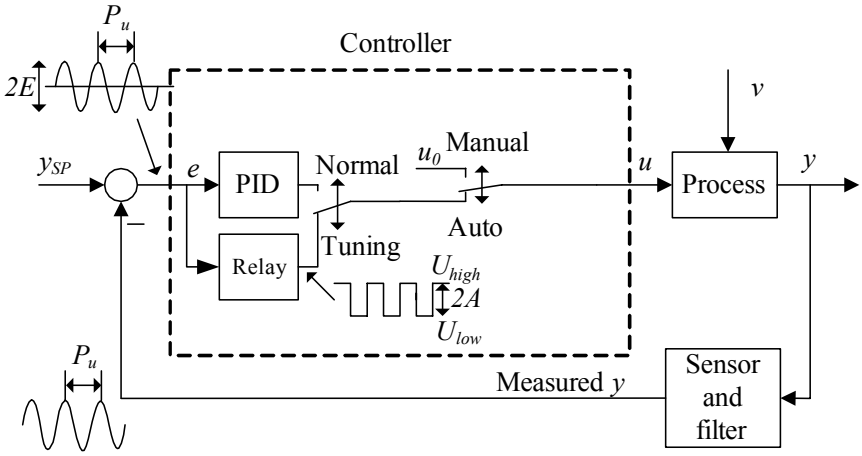

Figure 25: Configuration of the control loop in Åstrøm-Hägglund's On/off method for tuning a PID controller

The method is based on using a relay controller or on/off controller in the place of the PID controller to be tuned during the tuning, see Figure 25. Due to the relay controller the sustained oscillations in control loop will come *automatically*. These oscillations will have approximately the same period as if the Ziegler-Nichols' Ultimate Gain method were used, and the ultimate gain K_{p_u} can be easily calculated. The method is as follows.

1. Bring the process to (or as close to as possible) the specified *operating point* of the control system to ensure that the controller during the tuning is "feeling" representative process dynamic[11] and to ensure that the signals during the tuning can vary without meeting limits due to being in a non-representative operating point. You bring the process to the operating point by manually adjusting the control variable, with the controller in manual mode, until the process variable is approximately equal to the setpoint.

2. Set the amplitude A of the relay controller. A is given by

$$A = \frac{U_{\text{high}} - U_{\text{low}}}{2} \tag{28}$$

   In principle A should be as small as possible so that the oscillations on the loop becomes as small as possible. But if A is too small, there may not be any oscillations at all since the controller may just go into saturation at U_{high} or U_{low}. Since it is crucial to have the oscillations, you may take the chance to set

$$U_{\text{high}} = U_{\text{max}} = 100\% \text{ (typically)} \tag{29}$$

[11] This may be important for nonlinear processes.

and

$$U_{\text{low}} = U_{\min} = 0\% \text{ (typically)} \tag{30}$$

With this choice of relay output levels you are certain to have oscillations in the loop. However, since the control signal is then switching between maximum and minimum, you should make the duration of the relay tuning experiment as short as possible.

You may claim that there is no relay controller in the control system! But you can turn the PID controller into a relay controller with the following settings:

$$K_p = \text{very large, e.g. } 10000 \tag{31}$$

$$T_i = \infty \tag{32}$$

$$T_d = 0 \tag{33}$$

3. Switch the relay controller into the loop. This causes sustained oscillations to appear automatically in the control loop. It is not necessary to excite the control loop externally for the oscillations to come (thus, the setpoint can be constant).

4. Read off the amplitude A_e of the oscillations of the input signal to the relay controller, which is the control error, and calculate the *equivalent gain* as follows:

$$K_e = \frac{\text{Amplitude of relay output}}{\text{Amplitude of relay input}} = \frac{A_u}{A_e} \tag{34}$$

where A_u is[12]

$$A_u = \frac{4A}{\pi} \tag{35}$$

5. Read off the *ultimate period* P_u as the period of the sustained oscillations (the oscillations are not neccessary symmetric, but just read off the period). P_u can be read off from either the measurement signal or from the control signal.

6. Calculate the controller parameters of a P, PI or PID controller according to the Ziegler-Nichols' closed loop method, cf. Table 2, using

$$K_{p_u} = K_e = \frac{\frac{4A}{\pi}}{A_e} = \frac{4A}{\pi A_e} \tag{36}$$

and P_u.

[12] A_u is the amplitude of the first harmonic of a Fourier series expansion of the square pulse train at the output of the Relay controller.

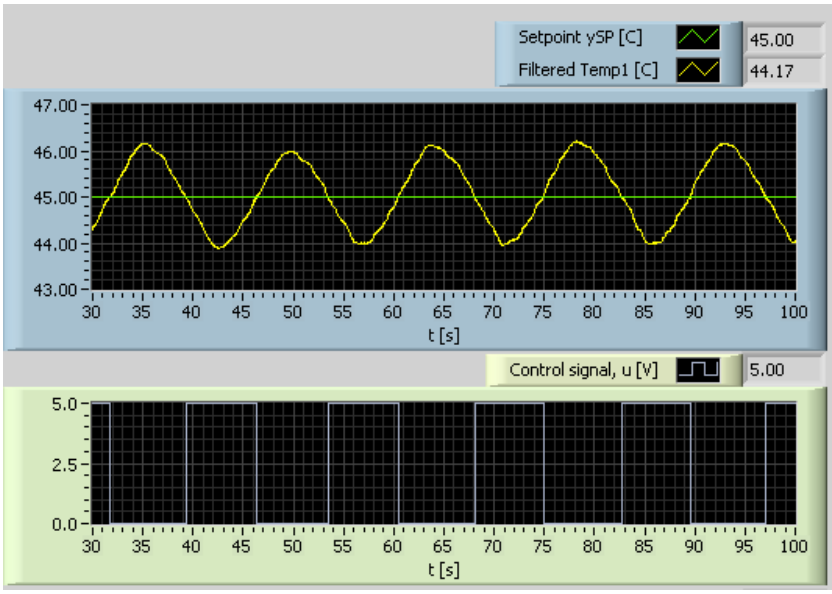

Figure 26: Relay tuning: The oscillations in the tuning phase.

How does the Relay method work with the temperature system? In an experiment the relay control signal high and low values were set to

$$U_{\text{high}} = 5 \text{ V} \tag{37}$$

and

$$U_{\text{low}} = 0 \text{ V} \tag{38}$$

respectively. Figure 26 shows the oscillations in the tuning phase. From 26 we find the ultimate period

$$P_u = 14.0 \text{ s} \tag{39}$$

The amplitude of the control error is appoximately

$$A_e = 1.1 \text{ °C} \tag{40}$$

The ultimate gain becomes, cf. (36),

$$K_{p_u} = \frac{4A}{\pi A_e} = \frac{4 \cdot 2.5 \text{ V}}{\pi \cdot 1.1 \text{ °C}} = 2.9 \text{ V/°C} \tag{41}$$

Assuming a PI controller, the controller parameters are calculated from Table 2 as

$$K_p = 0.45 \cdot K_{p_u} = 0.45 \cdot 2.9 = 1.31 \text{ V/°C} \tag{42}$$

$$T_i = \frac{P_u}{1.2} = \frac{14.0 \text{ s}}{1.2} = 11.7 \text{ s} \tag{43}$$

T_i is the same as found with the Ziegler-Nichols' method, while K_p differs just a little (with Ziegler-Nichols' method $K_p = 1.35$). The setpoint step response will therefore be very similar to the response shown in Figure 24.

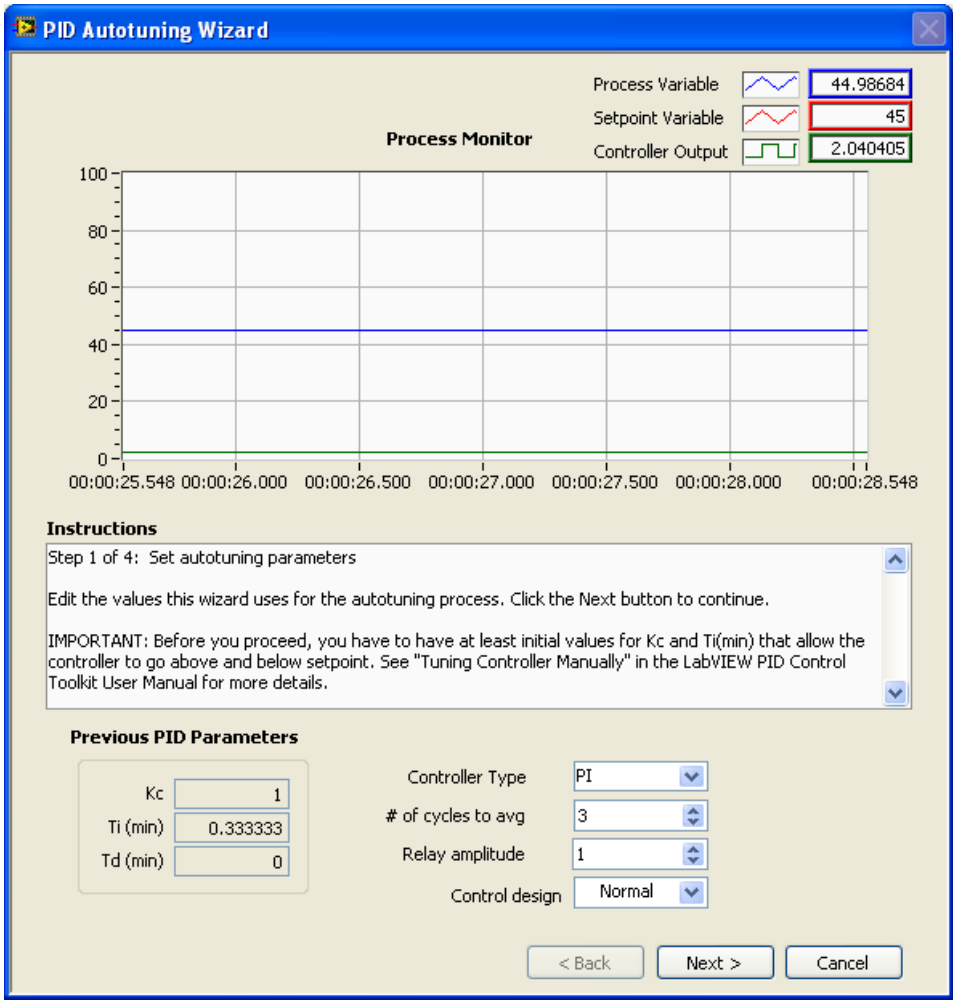

Figure 27: One of the steps of the PID Autotune Wizard

8.5 The Auto-tuner in LabVIEW

The **PID Autotuning** function on the PID Control palette in LabVIEW performs autotuning. The autotuner requires that the control loop is stable initially (with a P, PI or PID controller). During the tuning the autotuner automatically changes the setpoint stepwise, causing oscillations to occur. Parameters of a second order model is estimated, and the controller parameters are calculated from the model. A wizard, see Figure 27, guides the user through a number of steps where the following parameters must be set by the user:

- *Type of controller*: P, PI, PID, and Fast, Normal and Slow tuning.
- *Measurement noise level* (the user can decide to use an estimate that is calcu-

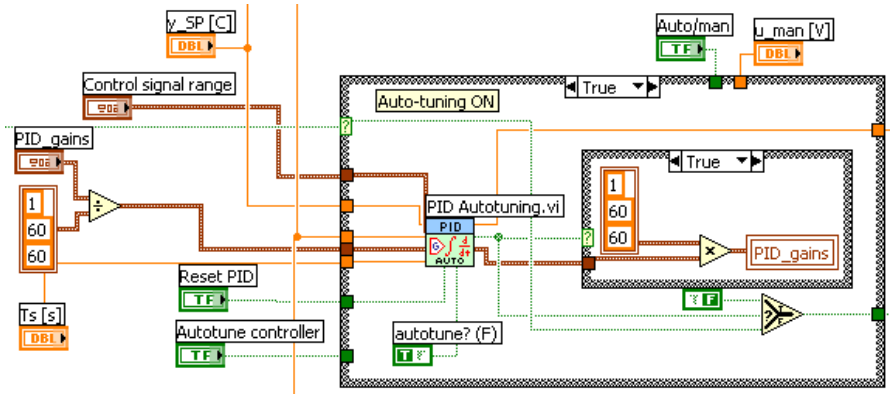

Figure 28: One example of how to include the **PID Autotuning** function in a LabVIEW block diagram

lated by the autotuner)

- *Number of oscillations* before the controller parameters are calculated and the wizard ends.

Figure 28 shows how the **PID Autotuning** function may appear in a LabVIEW block diagram. The wizard is opened when the **autotune? input** to the **PID Autotuning** function is TRUE. When the tuning is finished, the new PID settings are written to the `PID_gains` local variable. The FALSE case of the Case structure shown in Figure 28 (this case is active when the tuning is finished), contains the **PID Advanced** function which is used in normal operation.

In one experiment, I set the initial PI parameters to be applied before the tuning was started as

$$K_p = 1,\ T_i = 20 \text{ s} \tag{44}$$

The result of the autotuning was (the Normal speed option was selected)

$$K_p = 0.54,\ T_i = 21.2 \text{ s} \tag{45}$$

Figure 29 shows temperature response due to a setpoint step with these PI(D) parameter values. The control system shows good stability. The response is more sluggish that with the PI settings from the Ziegler-Nichols' method and the Relay method.

8.6 Gain Scheduling

The problem: It can be shown both experimentally and mathematically (using a simplified model) that the gain and the transport delay of a flow process – from

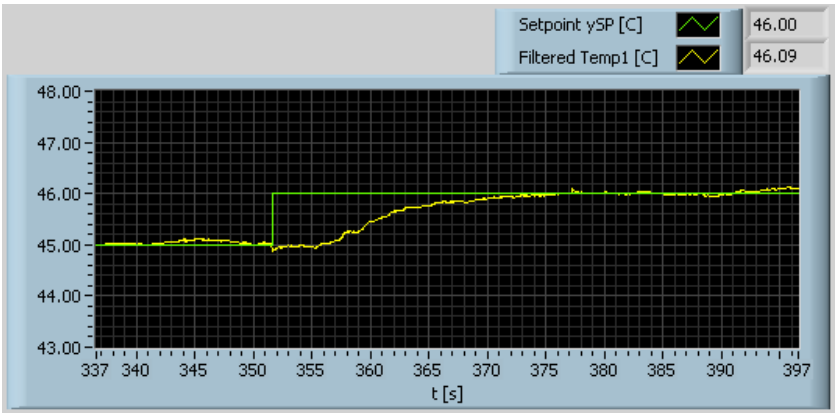

Figure 29: After using the LabVIEW Autotuner: Temperature response due to a setpoint change.

controlled heat to measured temperature – increases as the flow descreases. Consequently, if the temperature controller is tuned for acceptable stability at a high flow rate, the control system may get poor stability – even instability – if the flow rate decreases.[13]

Figure 30 illustrate this for the air heater. The PI controller parameters was set to

$$K_p = 1.6,\ T_i = 8.0 \text{ s} \tag{46}$$

With these PI parameter values the control system is stable at maximum flow rate, but becomes *unstable* at the minimum flow rate (fan speed indication was 2 V), as seen from Figure 30.

One solution to the stability problem as the flow rate is reduced is to make the controller parameters change as functions of the flow rate (fan speed indication voltage), which is denoted the *gain scheduling variable*. In LabVIEW this can be implemented using the **PID Gain Schedule** function. This function contains a user-defined number of PID settings. At any instant of time the PID settings applied to the PID controller is selected by the value of the fan speed voltage. With the **PID Gain Schedule** function the PID settings are held constant during the different intervals of the gain scheduling variable. In this experiment, three PID settings were used, each found by some simple tuning.

Figure 31 shows how the **PID Gain Schedule** function was included in the LabVIEW block diagram. The shedule or table of PID gains is simply a cluster of arrays of three elements (the array contains K_p, T_i and T_d). The number of arrays is equal to the number of PID parameter sets. These three PID parameter settings are:

[13] This is analog to reducing the production rate in the production line of a plant with continuous operation.

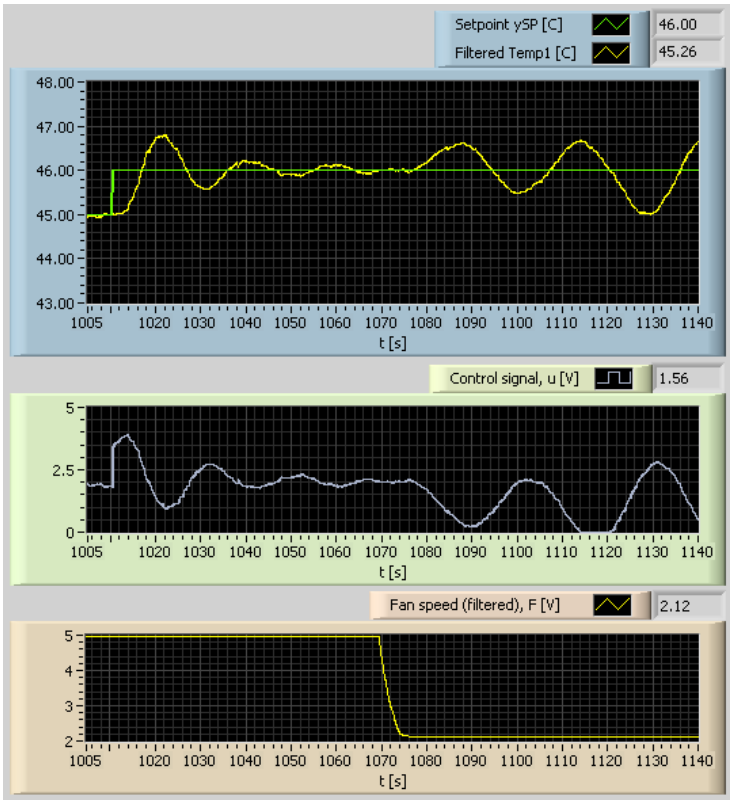

Figure 30: Temperature response when the fan speed (air flow) was *reduced*.

1. Flow indication $F = 5$V:

$$K_p = 1.6,\ T_i = 8 \text{ s} \tag{47}$$

2. Flow indication $F = 4$V:

$$K_p = 1.4,\ T_i = 10 \text{ s} \tag{48}$$

3. Flow indication $F = 3$V:

$$K_p = 1.2,\ T_i = 12 \text{ s} \tag{49}$$

Figure 32 shows that *with gain scheduling* the control system is stable for any flow.

9 Cascade Control

Cascade control can improve the disturbance compensation compared to using ordinary single loop control. Figure 33 shows a block diagram of cascade control as

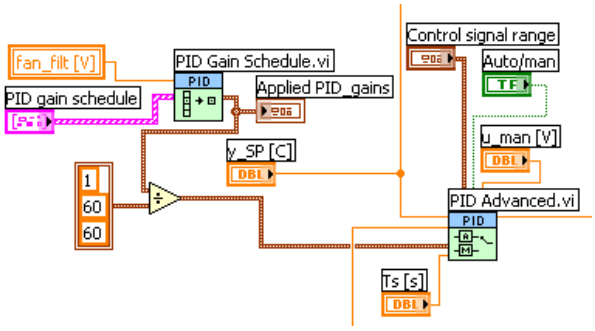

Figure 31: **PID Gain Schedule** function in the LabVIEW block diagram

applied to the air heater to improve the control of the primary process variable, Temperature 1, which is the temperature at the outer sensor location of the pipe, cf. Figure 1. In the LabVIEW program the controllers are implemented with the **PID Advanced** functions.

The experiments for comparing single loop control with cascade control were as desribed below. The temperature setpoint was

$$y_{SP} = 50 \text{ °C} \tag{50}$$

Initially the air fan speed was at minimum. Both control systems were excited by an increase of the fan speed from minimum to maximum speed and then back again to minimum speed. The duration of the increased speed was 15 sec. This change represents a disturbance change on the process.

- **Single loop control**: Temperature 1 was fed back to the controller. The controller was tuned as a PI controller with

  $$K_p = 1.4,\ T_i = 16.0 \text{ s} \tag{51}$$

  The measurement filter has time constant $T_f = 0.5$ s. Figure 34 shows the responses of the single loop control system.

- **Cascade control**: Temperature 1 was fed back to the primary controller, while Temperature 2 at the inner sensor location was fed back to the secondary controller. The secondary controller will try to maintain Temperature 2 at its setpoint (which is the output from the primary controller) by compensating relatively quickly for variations in Temperature 2. If Temperature 2 is kept more constant, the primary controller will get an easier job as it tries to keep Temperature 1 at its setpoint.

  The secondary controller was tuned as a PI controller with

  $$K_p = 1.3,\ T_i = 15.0 \text{ s} \tag{52}$$

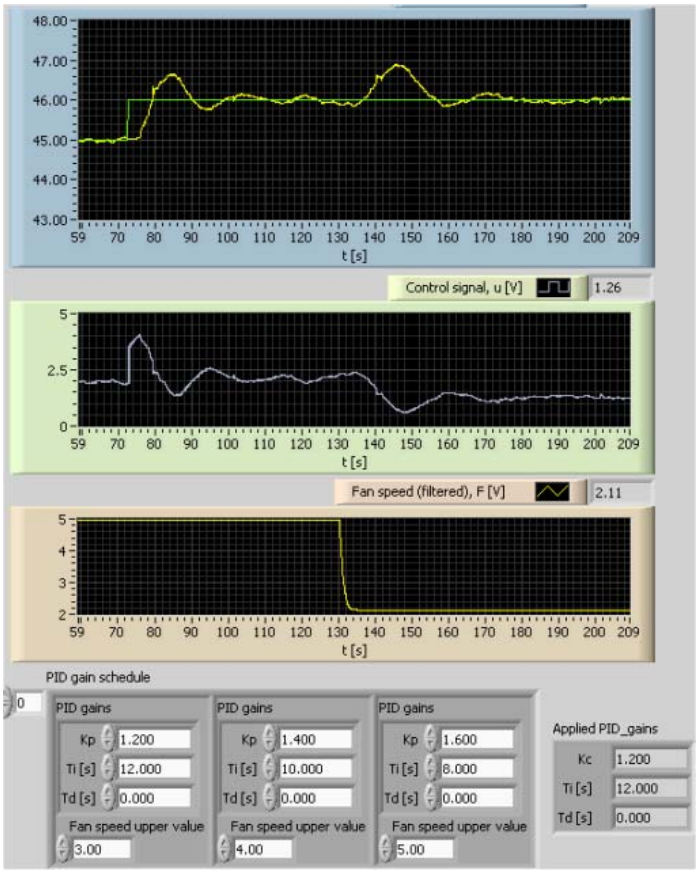

Figure 32: Response in temperature with gain scheduling based on measured fan speed

The primary controller was tuned as a PI controller with

$$K_p = 0.36,\ T_i = 15.0 \text{ s} \tag{53}$$

For each of the sensors the measurement filter has time constant $T_f = 0.5$ s. Figure 35 shows the responses of the cascade control control system.

Table 3 shows the performance indices for each of the control systems.

As the table shows, the control is better with cascade control, but the improvements are not large. And the control signal usage is actually worse with cascade control since it varies more. In other systems the improvements may be more profound. The reason why there is not large improvements in the present system is that there is not much dynamics from the inner sensor location to the outer location. Therefore the secondary loop is not very much quicker than the primary loop, and hence there are not large improvements. Still there is some improvement.

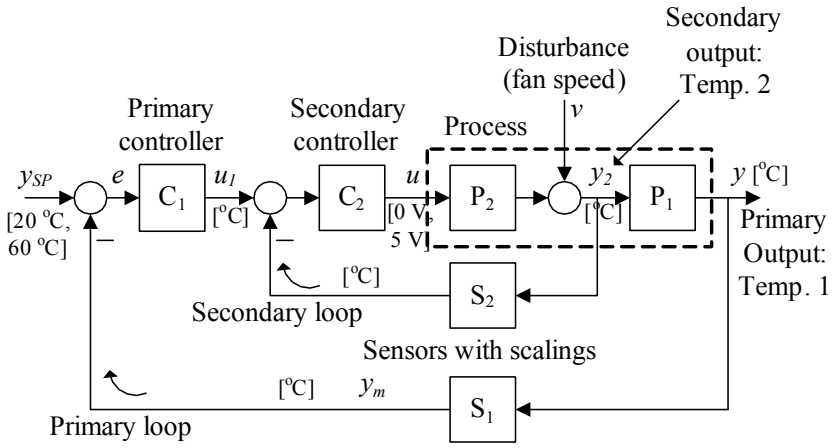

Figure 33: Cascade control of the primary process variable, Temperature 1

	Single loop control	**Cascade control**
$\|e\|_{\max}$	0.98	0.78
IAE	21.9	12.9
$m_{du/dt}$	0.12	0.16

Table 3: Performance indexes of single loop control system and cascade control system

10 Feedforward Control

Feedforward control from a known or measured process disturbance can be designed from a mathematical model of the process to be controlled [5]. The model can be a differential equations model or a transfer function model (the latter may result in a lead-lag transfer function as feedforward controller). Feedforward control can also be designed from simple experiments, *without* an explicit model, as follows:

- Decide a proper set of N different values of the disturbance, v on which the feedforward control will be based, for example $N = 4$ different values of the disturbance.
- For each of these N distinct disturbance values, find (experimentally or by simulation) the value of the control signal u which corresponds to zero steady state control error, which can be obtained with PI or PID feedback control..
- The set of N corresponding values of v and u can be represented by a table, cf. Table 4.

  or in a coordinate system, cf. Figure 36.

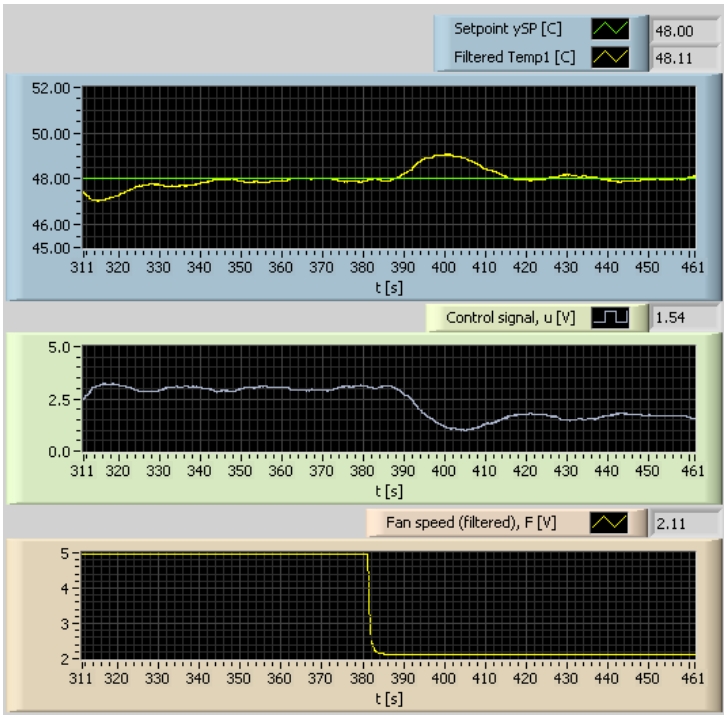

Figure 34: Single loop control of Temperature 1

u	v
u_1	v_1
u_2	v_2
u_3	v_3
u_4	v_4

Table 4: N corresponding values of v and u

- For any given (measured) value of the disturbance, the feedforward control signal u_f is calculated using interpolation, for example linear interpolation as shown in Figure 36. In practice, this linear interpolation can be implemented using a table lookup function.

Note: This feedforward design method is based on *steady state* data. Therefore, the feedforward control will not be ideal or perfect. However, it is easy to implement and it may give substantial better control compared to only feedback control.

How does this feedforward work when applied to the air heater? The disturbance is here the air flow, which is represented by the fan indication voltage, F. Four sets of corresponding values of fan voltage and control signal, u, were recorded, see the table below.

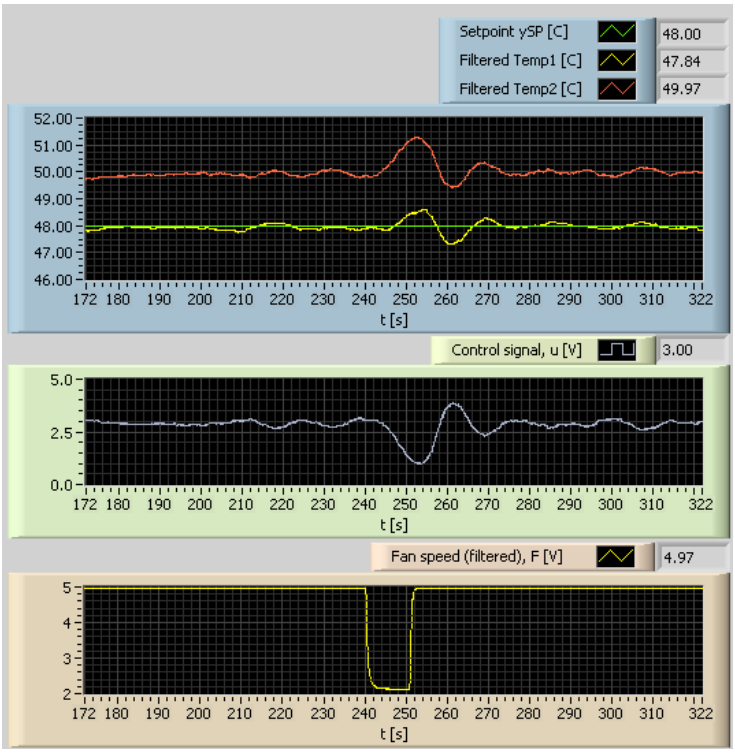

Figure 35: Cascade control of Temperature 1

The feedback controller was a PI controller with parameter values found with the Ziegler-Nichols' Ultimate Gain method. The following two experiments were performed – without and with feedforward, but in both cases with PI feedback control with the above parameters:

- **Without feedforward**: Figure 37 shows the temperature response due to a disturbance change in the form of a reduction of the fan speed from maximum value to minimum value. The minimum fan speed value was held for about 40 sec before it was increased. From the figure we read off the maximum control

u	F
$u_1 = 1.8$ V	$F_1 = 5.0$ V
$u_2 = 1.6$ V	$F_2 = 4.0$ V
$u_3 = 1.3$ V	$F_3 = 3.0$ V
$u_4 = 1.1$ V	$F_4 = 2.1$ V

Table 5: Four corresponding values of F (fan speed) and u

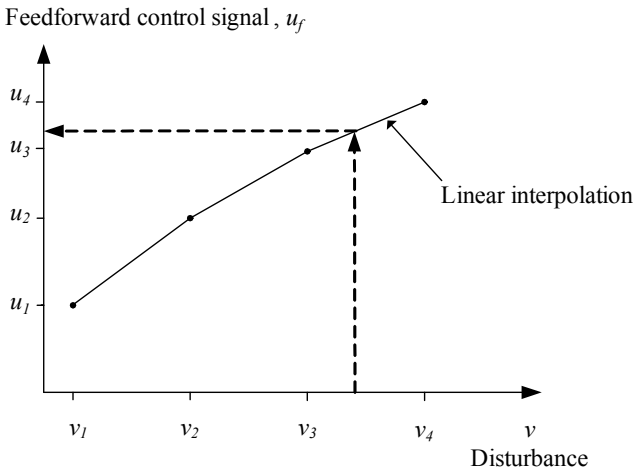

Figure 36: Calculation of feedforward control signal from known disturbance value

error – both after the decrease and after the increase of the fan speed – to be

$$|e|_{\max} = 0.7\ ^\circ\mathrm{C} \tag{54}$$

- **With feedforward**: Figure 38 shows the temperature response due to a disturbance change in the form of a reduction of the fan speed from maximum value to minimum value. The minimum fan speed value was held for about 40 sec before it was increased. From the figure we read off the maximum control error – both after the decrease and after the increase of the fan speed – to be

$$|e|_{\max} = 0.2\ ^\circ\mathrm{C} \tag{55}$$

which is a clear improvement compared to using no feedforward control![14]

Figure 39 shows how the feedforward control signal, u_f, was calculated by linear interpolation with **Interpolate 1D Array** function in LabVIEW and added to the PID control signal to make up the total control signal:

$$u = u_{PID} + u_f \tag{56}$$

It was necessary to change the control output signal limits of the PID controller when the feedforward control was used. The lower output limit was lowered from 0 V to –5 V because the feedforward control signal u_f is positive.

[14] Since $0.7/0.2 = 3.5$ you can say that the inclusion of feedforward control "improves the control more than three times".

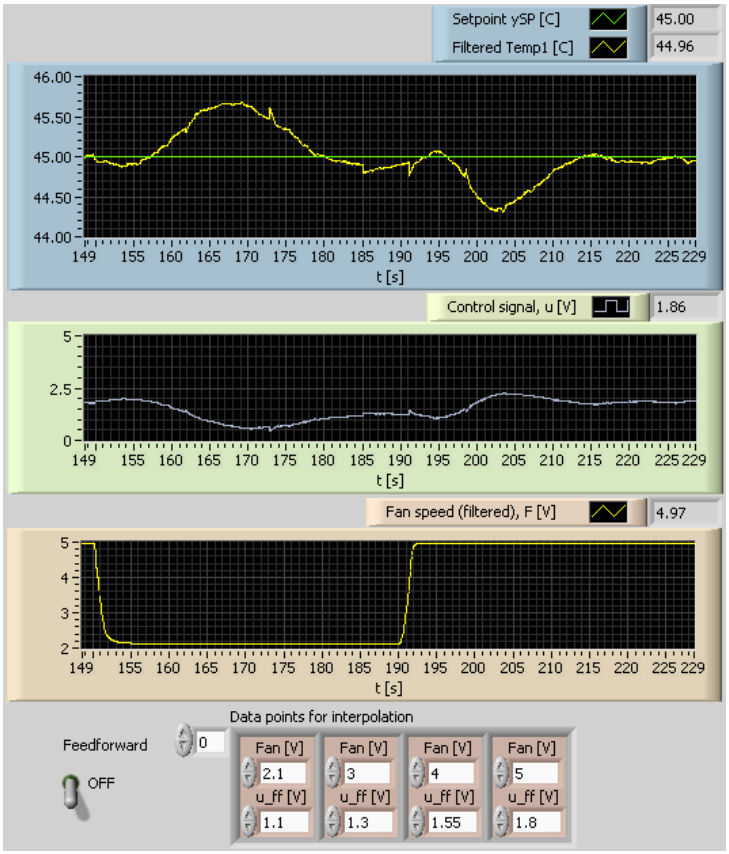

Figure 37: *Without feedforward*: The temperature response due to a disturbance change in the form of a reduction followed by an increase of the fan speed

11 Summary

This paper has shown that several basic control principles and controller tuning methods can be demonstrated with a temperature controlled heated air pipe. LabVIEW provides a convenient environment for implementing the control system due to its block diagram based (graphical) programming environment and the large number of proper functions. Experiences from teaching at Telemark University College in Norway shows that this lab station can be exploited to provide instructive and practical short laboratory assignments in courses in process control.

12 Further Use of the Lab Station

While in the present paper no mathematical process model was used, and hence only model-free controller tuning methods were demonstrated, at Telemark Uni-

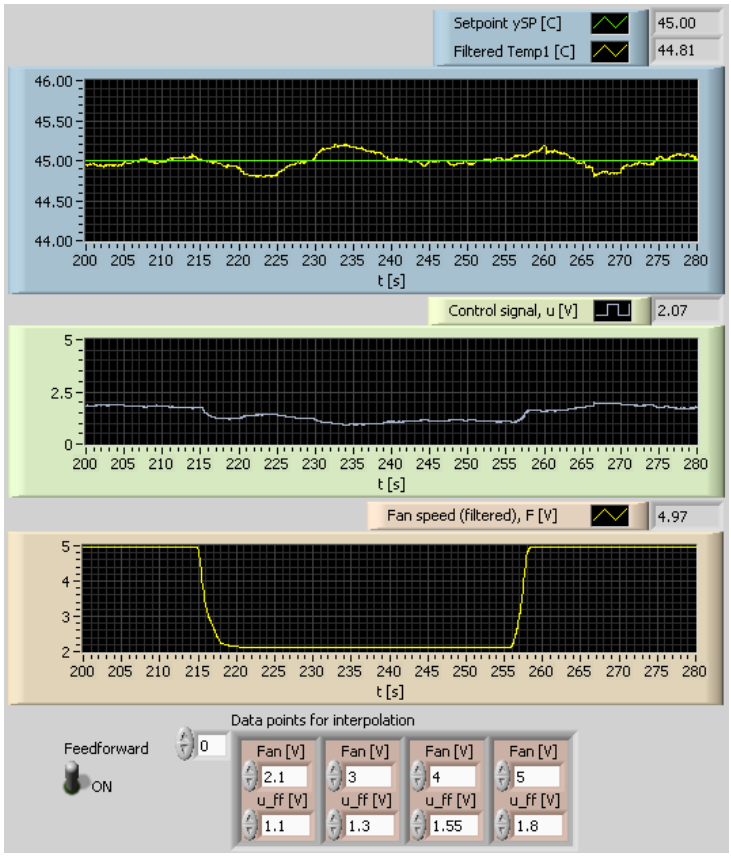

Figure 38: *With feedforward*: The temperature response due to a disturbance change in the form of a reduction followed by an increase of the fan speed

versity College we plan to develop student assignments based on various types of mathematical process models. Some possible future assignments are as follows:

- **System identification**: Developing discrete-time transfer function models of the process
- **Model-based controller tuning**: E.g. tuning based on frequency-domain specifications as stability margins.
- **State estimation**: Estimation of states, including disturbances, using e.g. the Kalman Filter
- **Model-based control functions**, as Model-based Predictive Control together with a state estimator, Smith Predictor with variable delay, and feedforward from setpoint and/or disturbance (air flow).

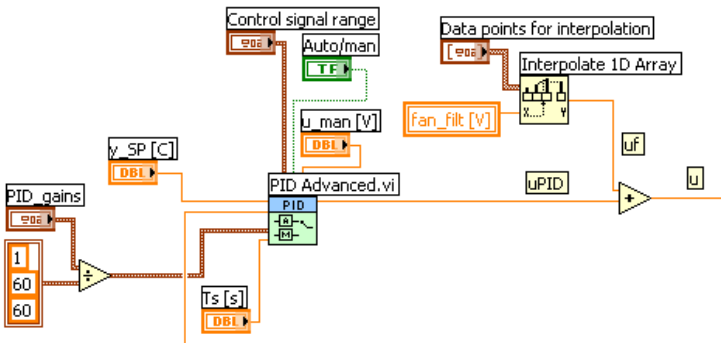

Figure 39: Implementation of feedforward control using the **Interpolate 1D Array** function

In addition to student assignments, the lab station will be used in industrial courses on basis process control.

References

[1] Åstrøm, K. J. and T. Hägglund, *Automatic Tuning of PID Controllers*, Instrument Society of America, 1988

[2] Edgar, Th. F., *Student to engineer. Should the teaching of process control be changed?* InTech, Instruments Society of America (ISA) magazine, October 2006, http://www.isa.org/intech.

[3] Haugen, F., *Introduction to LabVIEW 8.2*, http://techteach.no.

[4] Kiam Heong Ang, Gregory Chong, Yun Li, *PID Control System Analysis, Design, and Technology*, IEEE Transactions on Control Systems Technology, Vol. 13, No. 4, July 2005

[5] Seborg, D. E., Th. F Edgar, and D. A. Mellichamp, *Process Dynamics and Control, Second Edition*, John Wiley & Sons, 2004.

[6] *PID controller*. Wikipedia. http://en.wikipedia.org/wiki/PID_controller

[7] Ziegler, J. G. and N. B. Nichols: *Optimum Settings for Automatic Controllers*, Trans. ASME, Vol. 64, page 759-768, 1942