Chapter 2:
Control Algorithms

2.1 Basic Control Concepts

In engineering and mathematics, **control theory** deals with the behavior of dynamical systems. The desired output of a system is called the **reference** or **set-point**, *r*. The value or process variable needed to be controlled is called the **controlled variable,** *c*. The difference between the set-point and the controlled variable is called the **error**, e. The signal which is affected directly into the process and outputted directly by the controller is called the **manipulated variable**, *m*. It is the process variable that can be adjusted in order to keep the controlled variables at or near its set point. The **measured value**, *m,* may be slightly different to the controlled variable due to noise or signal conditioning. When one or more output variables of a system need to follow a certain reference over time, a controller manipulates the inputs to a system to obtain the desired effect on the output of the system.

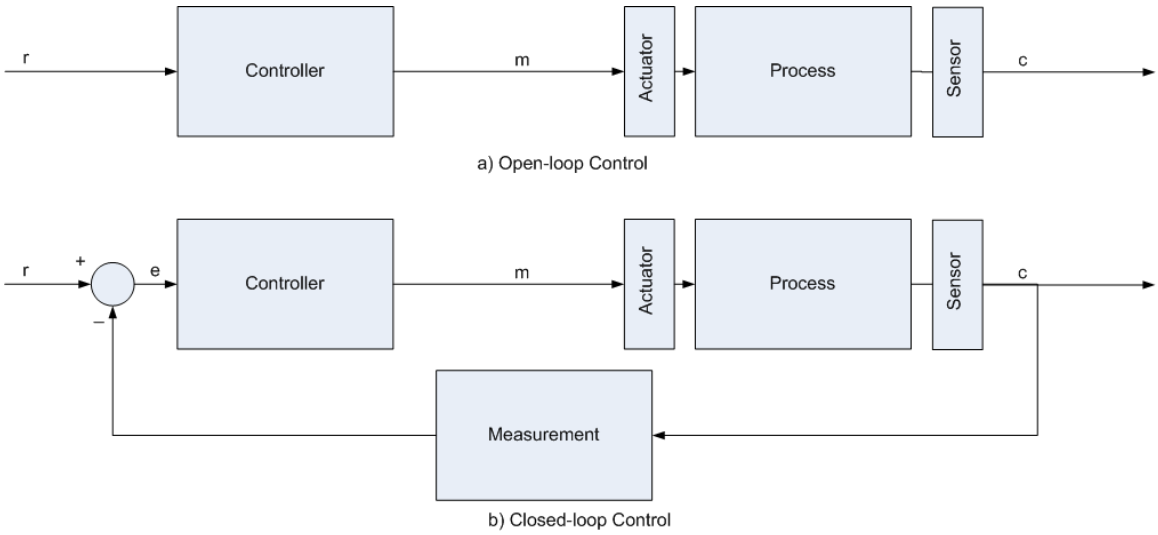


Figure 2.1 Comparing open-loop and closed-loop control.

Common parts found in a typical control system:

- **Process** or **Plant:** the physical system needed to be controlled.
- **Measurement Device:** To control a process accurately, the control loop needs to be able to measure a particular value (such as temperature) on a regular basis. This is usually done by:
    - a **sensor** which takes the measurement,
    - and then a **transmitter** that converts the sensor reading into a standard control signal

- **Transducer:** fgfg…..
- **Actuator** or **Regulator**: This controls the throughput of the process. The type of regulator depends on the need of the process. Examples include: a control valve or variable speed drive that adjusts the flow of a fluid and electronic circuit that supply power to a device (e.g. switch on a heater)
- **Controller:** compares the measured value (e.g. current temperature) with its set-point (the required temperature) and, where there is a difference, it adjusts the regulator for the required process, e.g. a drop in temperature could result in an increase in fuel to the burners in order to bring the temperature back up to the required set-point.

There are mainly two types of control: open-loop and closed-loop. Open-loop control, which is rarely used, simply applies a pre-defined manipulated variable into the process to achieve a certain output. This manipulated variable is a function of the required set point and is either calculated or found by trial-and-error. However, if any external disturbance (i.e. a sudden change in the environment) occurs or any internal disturbance (i.e. the process parameters change due to ageing) occurs or if any noise occurs, the controlled variable will change. Therefore, we need feedback to reduce any undesired change in the controlled variable. This is introduces us to **negative feedback**. Open-loop control can be used in systems sufficiently well-characterized as to predict what outputs will necessarily achieve the desired states. For example, the rotational velocity of an electric motor may be well enough characterized for the supplied voltage to make feedback unnecessary. Drawbacks of open-loop control is that it requires perfect knowledge of the system (i.e. one knows exactly what inputs to give in order to get the desired output), and it assumes there are no disturbances to the system.

Closed-loop control, which is what is implemented in our project, achieves the required feedback. Closed-loop control has another advantage over open-loop control in that we may modify the controller to achieve the **transient response** (i.e. how the controlled variable reaches its steady-state value) we want. The controlled variable is always compared with the required set-point to obtain the error. According to this error, the controller takes the correcting action to exert onto the actuator. How this correcting action is obtained is what differs different types of controllers from each other. The task of a control loop is to hold a particular process variable (e.g. mixer speed, or oven temperature) at the required set-point or value.

The **single-loop controller**, if it is correctly chosen and well tuned, is able to handle about 90% of control tasks found within the process industries. In our project, we have implemented the single-loop controller.

Linear systems, or non-linear systems which are linearized by considering a small portion of its range of operation, have their responses classified into two types of responses: **first-order response** and **second-** (or **higher-**) **order response**:

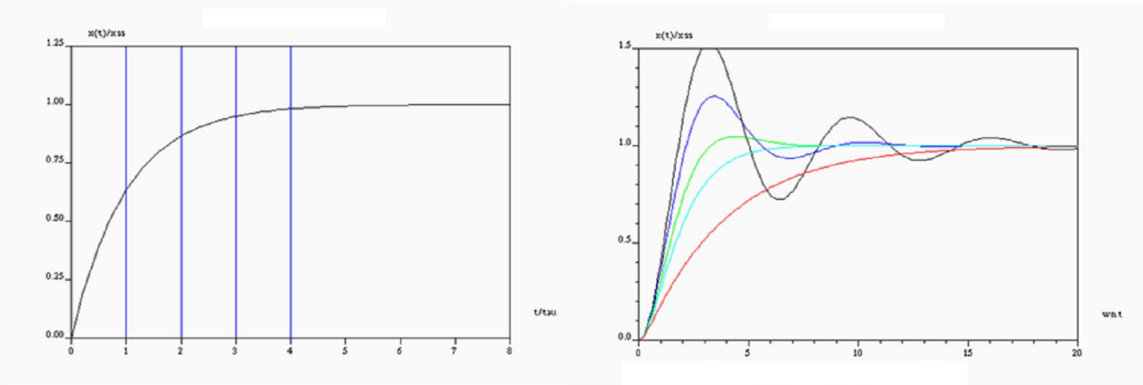


Figure 2.2 Comparing first-order response and second-order response.

- First order response is characterized by the **time-constant,** τ or T_0, which is the time required for the system to reach 63% of its final value. Usually, we would like to have the time-constant as low as possible to make the system response faster. Its transfer function in the s-domain is represented by:

$$H(s) = \frac{1}{1 + sT_0}$$

- Second order response has the following transfer function in the s- domain:

$$H(s) = \frac{{\omega_0}^2}{{\omega_0}^2 + 2\eta\omega_0 + s^2}$$

It is characterized by:

  - **Rise-time,** t_r : refers to the time required for a signal to change from a specified low value to a specified high value. Typically, these values are 10% and 90% of the step height. Usually, we want to minimize this value.
  - **Settling-time,** t_s : refers to the time required for a signal to remain within 2% of its steady-state value. Usually, we also want to minimize this value.
  - **Maximum Overshoot,** *M*: is the maximum positive difference between a signal and its steady-state value. Usually, we also want to minimize this value.

Using closed-loop control we may change the above parameters of a response of a system to virtually any required value.

2.2 Digital Control

For a very long time, feedback control was implemented in continuous-time using analog systems; namely pneumatic system or electronic (op-amp) circuits. However, the rise of

computers paved the way for **digital control**. Since our project is using microcontrollers, we are going to implement digital control.

The term "digital control" usually implies two facts. First, control is taken place in the discrete-time domain not the continuous-time domain. This means that the microcontroller should sample the controlled variable every time sample, T_s , and the control action is determined and taken every sample time. Second, the values of the controlled variable and the manipulated variable are quantized; therefore we have a limited number of values to read and write to.

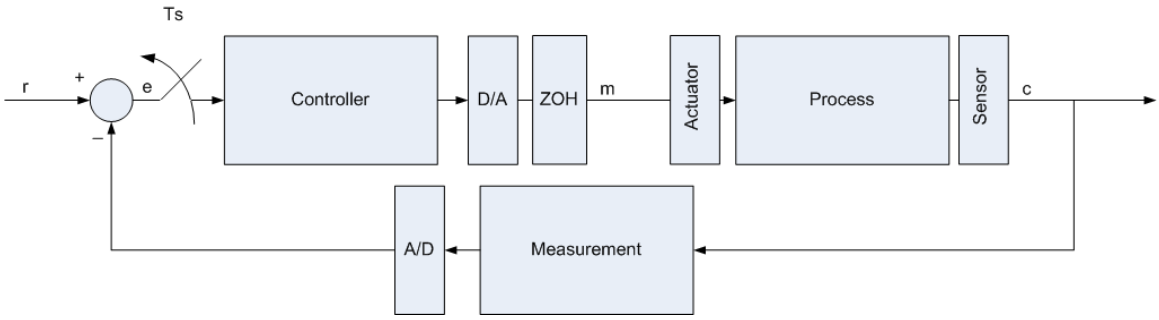
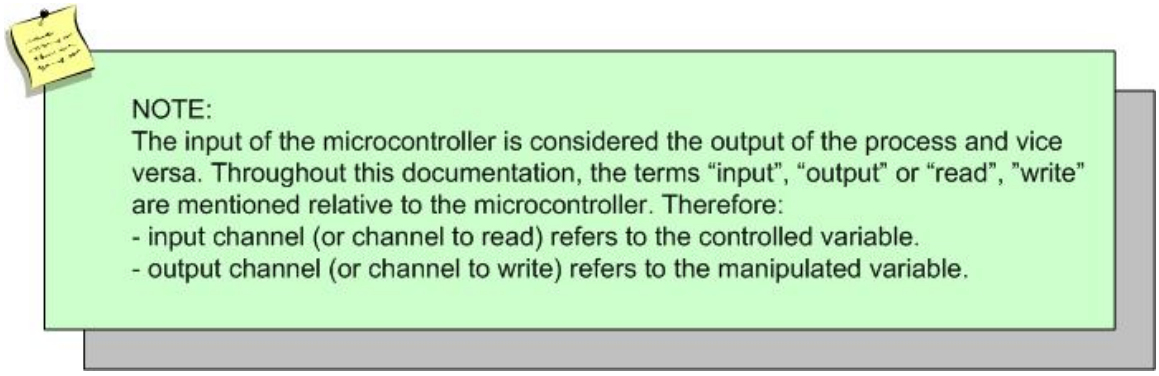


Figure 2.3 Block diagram of digital control closed-loop system.

Digital control systems have introduced 2 more components: **digital-to-analog converter** (**D/A** or **DAC**) and **analog-to-digital converter** (**A/D** or **ADC** or **ATD**). Digital data is represented as a group of bits to be stored in the microcontroller's memory and allow calculation and manipulation.



There are several advantages to digital control over analog control:

- The whole world is moving towards the digital control era.
- Digital components are less susceptible to ageing and environmental variations.
- They are less sensitive to noise.
- Changing a controller does not require an alteration in the hardware (flexibility).
- Reduce the design time.
- Improve the system reliability.

- Easier system integration.
- Low implementation costs.

Considering the quantization of data, a question has emerged: should the data be dealt with as integers or floating numbers and should they be scaled? We have agreed in the project that the user should enter all of his values (for example: set points) as percentage values, and they should be scaled by the monitoring software to a byte value from 0-255. The microcontroller should perform all of its calculations as bytes or integers and return the values as bytes to the monitoring software to scale it back to the percentage range.

There was no need to perform calculations in the microcontroller as floating numbers for 2 reasons:

- The high inefficiency in storage, power and execution time.
- Any calculated values would be casted back to integer numbers (since the analog output should also be discritized), therefore there was no advantage of having high precision values since their precision would be lost again.

Another implementation consideration was the sampling time: would it be large or small?

- A very small sample time means more control actions per unit time which therefore means more power used by the microcontroller.
- A very large sample time means that control actions will be taken too late to perform the desired feedback, causing the system to be unstable. Moreover, the original signal may not be reconstructed

The minimum required sampling time should satisfy Nyquist criterion:

$$f_s > 2f_{max}$$

Where f_{max} is the maximum frequency component in the system, usually equal to its bandwidth.

- For first order systems, we choose $\frac{T_0}{4} < T_s < T_0$, where T_0 is the time constant.
- For second order systems, we choose $0.25 \leq \omega_0 T_s \leq 1.5; \quad 0.7 \leq \eta \leq 1$

We have designed our project so that the sample time is chosen by the user.

2.3 On/Off Control

On/Off controllers are very famous industrial controllers, they can even be found everywhere in our daily life, for example On/Off controllers can be found in air conditioners and refrigerators, heaters and most temperature systems, also in water level regulating systems.

On/Off controllers are famous for the following reasons:

- On/Off controllers are very simple to design and implement.
- They perform very well when chosen correctly to operate on a suitable process (first order process).
- Can be implemented digitally or in analog circuits.
- Are not severely affected with noise, or disturbances.
- Reliable in operation for long times.

On/Off controllers are characterized by a **set point**, which we want the controlled variable to vary around, and the **neutral zone**, which determines the greatest offset from the set point. We need to make a compromise between a small neutral zone value, which will cause high switching rate for actuators which will therefore cause its depreciation, and a large neutral zone value which will cause a large maximum error.

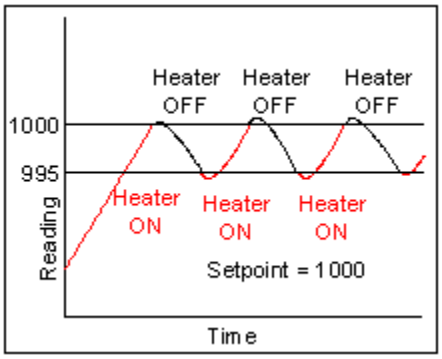


Figure 2.4 An example of On/Off control action.

The output of an On/Off controller is digital since it can be either high or low. On/Off control may be implemented digitally using an if-else statement:

```
if (Analog Input <= (SP + NZ/2)) Digital Output = HIGH

else if (Analog Input >= (SP - NZ/2)) Digital Output = LOW
```

2.4 PID Control and PID Tuning

A **proportional-integral-derivative controller (PID controller)** is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly, based upon three parameters.

The PID controller calculation (algorithm) involves three separate parameters; the Proportional, the Integral and Derivative values. The proportional value determines the reaction to the current error, the integral determines the reaction based on recent errors and the

derivative determines the reaction based on the rate by which the error has been changing. The weighted sum of these three actions is outputted to a control element such as the position of a control valve or power into a heating element.

By "tuning" the three constants in the PID controller algorithm the PID can provide individualized control specific to process requirements including error responsiveness, overshoot of set-point and system oscillation. Note that the general nature of PID control does not guarantee optimal control of the system.

Some applications may require only using one or two modes (by setting undesired control values to zero) to provide the appropriate system control. A PID controller will be called a PI, PD, P or I controller in the absence of respective control actions. PI controllers are particularly common, since derivative action is very sensitive to measurement noise, and the absence of an integral value prevents the system from ever reaching its target value due to control action.

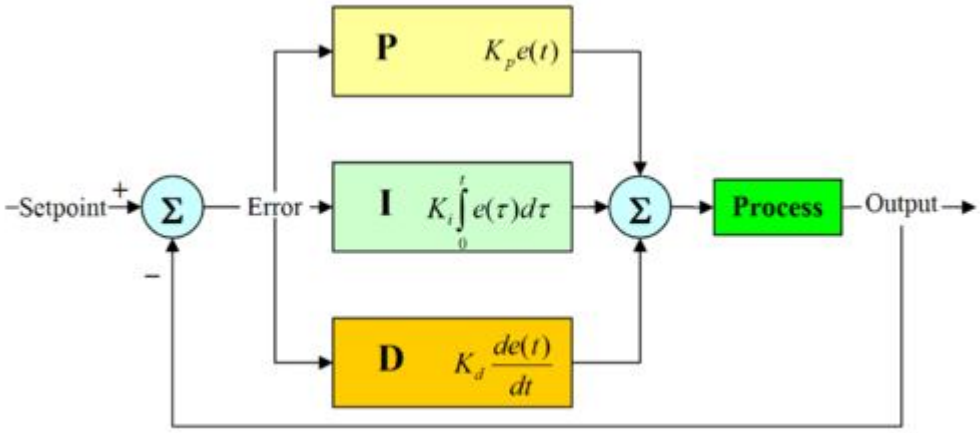


Figure 2.5 A block diagram of a PID controller.

The PID control scheme is named after its three correcting terms, whose sum constitutes the output. The three terms are

Proportional term

The proportional term responds to a change in the process variable proportional to the current measured error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain or proportional sensitivity. The gain is also frequently expressed as a percentage of the proportional band. The proportional term is written as:

$$m = K_p e = \frac{100}{PB} e$$

m: **Output Signal**

K_p: **Proportional Gain**

e: **Error** equal to (set point value - process variable)

PB: **Proportional Band**

A high gain results in a large response to a small error, a more sensitive system (Also called a narrow proportional band). Note that by setting the proportional gain too high, the system can become unstable. In contrast, a small gain results in a small response to a large error, and less sensitive system (Also called a wide proportional band), which is undesirable as the control action may be insufficient to respond to system disturbances.

Finally pure proportional control will never theoretically settle at its target value, but will rather approach the target with a steady state error that is a function of the proportional gain, this is known as a steady state error.

Integral term

The contribution from the integral term is proportional to the past and current values **and** duration of the error signal. The integral term algorithm calculates the accumulated proportional offset over time that should have been corrected previously (finding the offset's *integral*). While this will force the signal to approach the set point quicker than a proportional controller alone and eliminate steady state error, it may also contribute to system instability as the controller will always be responding to past values. This instability causes the process to overshoot the set point since the integral value will continue to be added to the output value, even after the process variable has reached the desired set point.

The responsiveness of the integral function can be calibrated to the specific process by adjusting the constant T_i, called the integral time.

The equation is written as:

$$m = \frac{1}{T_i} \int_0^t e(\tau)\, d\tau$$

m: **Output Signal**

T_i: **Integral Time**

e: **Error** equal to (setpoint value - process variable)

Although mathematically the integral starts at $t = 0$, it is possible to modify the integral to such that it does not "record" all historical values of the error signal. There are many possible schemes for performing such modification, such as windowing the signal or applying a decay term to the integral value itself.

Derivative term

The derivative term provides a braking action to the controller response as the process variable approaches the set point. To accomplish this the process error is predicted at a time in the future T_d, calculated by analyzing the slope of error vs. time (i.e. the rate of change of error, which is its first derivative with respect to time) and adding the anticipated proportional term to the current correction.

Derivative control is used to reduce the magnitude of the overshoot produced by the integral component, but the controller will be a bit slower to reach the set point initially. As differentiation of a signal amplifies the noise levels, this mode of control is highly sensitive to noise in the error term, and can cause a noisy controlled process to become unstable.

By adjusting the constant, T_d, the derivative time, the braking action sensitivity is controlled.

The derivative term is written as:

$$m = T_d \frac{de}{dt}$$

m: **Output Signal**

T_d: **Derivative Time**

e: **Error** equal to (set point value − process variable)

2.4.1 PID Algorithm Implementation

Parallel / "non-interacting" Form

The PID algorithm can be implemented in several ways. The easiest form to introduce is the parallel or "non-interacting" form, where the P, I and D elements are given the same error input in parallel. The output of the controller (i.e. the input to the process) is given by

$$\mathrm{Output(t)} = P_{\mathrm{contrib}} + I_{\mathrm{contrib}} + D_{\mathrm{contrib}}$$

where P_{contrib}, I_{contrib}, and D_{contrib} are the feedback contributions from the PID controller, defined below:

$$P_{\mathrm{contrib}} = K_p\, e(t)$$

$$I_{\mathrm{contrib}} = \frac{1}{T_i} \int_0^t e(\tau)\, d\tau$$

$$D_{\text{contrib}} = T_d \frac{de}{dt}$$

Where $e(\tau) = set\ point - measurement(\tau)$ is the error signal, τ is the time in the past contributing to the integral response and K_p, T_i, T_d are constants that are used to tune the PID control loop:

K_p: Proportional Gain - Larger K_p typically means faster response since the larger the error, the larger the feedback to compensate.

T_i: Integral Time - Smaller T_i implies steady state errors are eliminated quicker. The trade-off is larger overshoot: any negative error integrated during transient response must be integrated away by positive error before we reach steady state.

T_d: Derivative Time - Larger T_d decreases overshoot, but slows down transient response.

Normally the controller is implemented with the K_p gain applied to the I_{contrib}, and D_{contrib} terms as well in the following form, also called the standard form;

$$\text{Output(t)} = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{de}{dt} \right)$$

In the ideal parallel form, the standard parameters T_i and T_d are replaced with (K_i and K_d).

$$\text{Output(t)} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt}$$

In this notation the gain parameters are related to the parameters of the standard form through $K_i = \frac{K_p}{T_i}$ and Kd = KpTd. This parallel form where the parameters are treated as simple gains is the most general and flexible form. However, it is also the form where the parameters have little physical interpretation.

Series / interacting form

Another representation of the PID controller is the series, or "interacting" form. This form essentially consists of a PD and PI controller in series, and it made early (analog) controllers easier to build. When the controllers later became digital, many kept using the interacting form.

Often, one deals with discrete time intervals instead of the continuity. Thus, the PID controller may also be dealt with recursively:

$$\text{Output}_n = \text{Output}_{n-1} + (K_p + K_i + K_d)\, e_n - (K_p + 2K_d)\, e_{n-1} + K_d\, e_{n-2}$$

2.4.2 Loop Tuning

If the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen incorrectly, the controlled process input can be unstable, i.e. its output diverges, with or without oscillations, and is limited only by saturations or breakage. **Tuning** a control loop is the adjustment of its control parameters (gain/proportional band, integral gain/reset, derivative gain/rate) to the optimum values for the desired control response. The optimum behavior on a process change or set point change varies depending on the application. Some processes must not allow an overshoot of the process variable from the set point. Other processes must minimize the energy expended in reaching a new set point. Generally stability of response is required and the process must not oscillate for any combination of process conditions and set points. Tuning of loops is made more complicated by the response time of the process; it may take minutes or hours for a set point change to produce a stable effect. Some processes have a degree of non-linearity and so parameters that work well at full-load conditions don't work when the process is starting up from no-load. This section describes some traditional manual methods for loop tuning.

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual "tune by feel" methods have proven time and again to be inefficient, inaccurate, and often dangerous.

The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and the response speed of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Choosing a Tuning Method		
Method	**Advantages**	**Disadvantages**
Ziegler-Nichols	Proven Method. Online method.	Process upset, some trial-and-error, very aggressive tuning
Tune By Feel	No math required. Online method.	Erratic, not repeatable
Software Tools	Consistent tuning. Online or offline method. May include valve and sensor analysis. Allow simulation before downloading.	Some cost and training involved.
Cohen-Coon	Good process models.	Some math. Offline method. Only good for first-order processes.

Table 2.1 Advantages and disadvantages of different tuning methods.

If the system must remain online, one tuning method is to first set the *I* and *D* values to zero. Increase the *P* until the output of the loop oscillates, then the *P* should be left set to be approximately half of that value for a "quarter amplitude decay" type response. Then increase *I* until any offset is correct in sufficient time for the process. However too much *I* will cause instability. Finally, increase *D*, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However too much *D* will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case a "critically damped" tune is required, which will require a *P* setting significantly less than half that of the *P* setting causing oscillation.

Effects of *increasing* parameters				
Parameter	**Rise Time**	**Overshoot**	**Settling Time**	**S.S. Error**
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	None

Table 2.2 Effects of increasing PID parameters.

Ziegler-Nichols method

Another tuning method is formally known as the "Ziegler-Nichols method", introduced by John G. Ziegler and Nathaniel B. Nichols. As in the method above, the *I* and *D* gains are first set to zero. The "P" gain is increased until it reaches the "critical gain" K_c at which the output of the loop starts to oscillate. K_c and the oscillation period P_c are used to set the gains as shown:

Ziegler-Nichols method			
Control Type	K_p	K_i	K_d
P	$0.5 \cdot K_c$	-	-
PI	$0.45 \cdot K_c$	$1.2K_p / P_c$	-
PID	$0.6 \cdot K_c$	$2K_p / P_c$	$K_p P_c / 8$

Table 2.3 Ziegler-Nichols method.

PID Tuning Software

Most modern industrial facilities no longer tune loops using the manual calculation methods shown above. Instead, PID tuning and loop optimization software are used to ensure consistent results. These software packages will gather the data, develop process models, and suggest

optimal tuning. Some software packages can even develop tuning by gathering data from reference changes.

Mathematical PID loop tuning induces an impulse in the system, and then uses the controlled system's frequency response to design the PID loop values. In loops with response times of several minutes, mathematical loop tuning is recommended, because trial and error can literally take days just to find a stable set of loop values. Optimal values are harder to find. Some digital loop controllers offer a self-tuning feature in which very small set point changes are sent to the process, allowing the controller itself to calculate optimal tuning values.

Other formulas are available to tune the loop according to different performance criteria.