A SunCam online continuing education course

Proportional, Integral, and Derivative Controller Design Part 2

by

Peter J. Kennedy

Proportional, Integral, and Derivative Controller Design Part 2

1.0 Introduction: The second part of this course further describes topics in the design and applications of Proportional -plus- Integral -plus- Derivative (PID) controllers. Part 2 briefly reviews PID Controller Design Part 1, discusses the digital implementation of the PID controller, and finally the use of PID control with adaptive and fuzzy logic control.

PID control is a technique used extensively in feedback control systems. It is simple in structure; being the sum of three terms as illustrated in Figure 1.0. Although simple, this structure provides for a fairly-wide range of tuning adjustment in a feedback control loop—especially for low order dynamic processes. As with Part 1, some background with feedback control theory and design will be helpful when taking the course as provided in [1, 2, and 3].

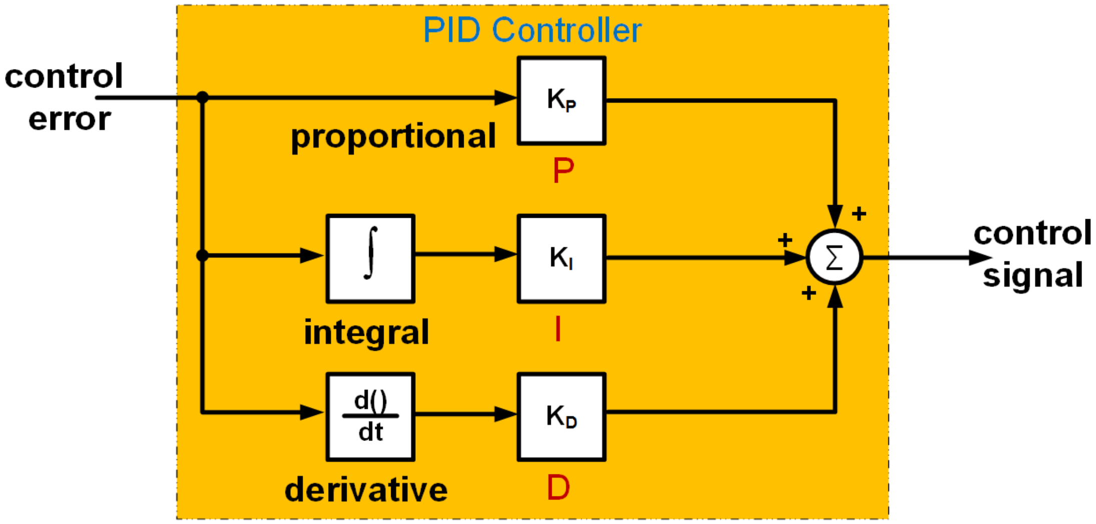


Figure 1.0 Functional Block Diagram PID Controller

PID controllers are used in many control applications; possibly the most common form of feedback control compensation. The versatility of the PID may reside in a fairly-simple control structure; easy to implement in software or hardware, offering loop gain adjustment, an integrator to reduce or null servo error, and the phase lead of a derivative improve loop stability or act as a predictive element. An example of a feedback control system is a building's central heating and air conditioning system. A thermostat, or temperature sensor, is the feedback sensor that measures the room temperature and compares it to the desired temperature or set point, calculating a difference or error. If the temperature is less than the set point, the error could be used by a PID controller to force more heat into the room. The proportional and integral gains will set the response time. As the set point is reached, the integrator will help null the error, and the derivative term reduce any overshoot—shutting down the heating or cooling source to the room.

As with PID Controller Design Part 1, this course will be presented from the perspective of classical control design techniques. Differential or difference equations describe the plant or process dynamics and these equations are transformed into frequency dependent transfer functions. The controller or compensator, such as the PID controller discussed in this course, shapes the closed feedback loop response for a given the plant response, to achieve the control performance objectives. Before focusing on the PID controller design, the next section will provide an overview of general feedback control system architecture, relationships that govern any controller design.

2.0 Feedback Control Block Diagram Algebra: The feedback control system block diagram provides both a visual as well as a mathematical basis for representing of the feedback control system. It is an important part of the control system design and analysis. In general, the feedback control block diagram will be complex—incorporating many elements and several feedback loops. To describe just the basic elements, a very simple block diagram is shown in Figure 2.0.

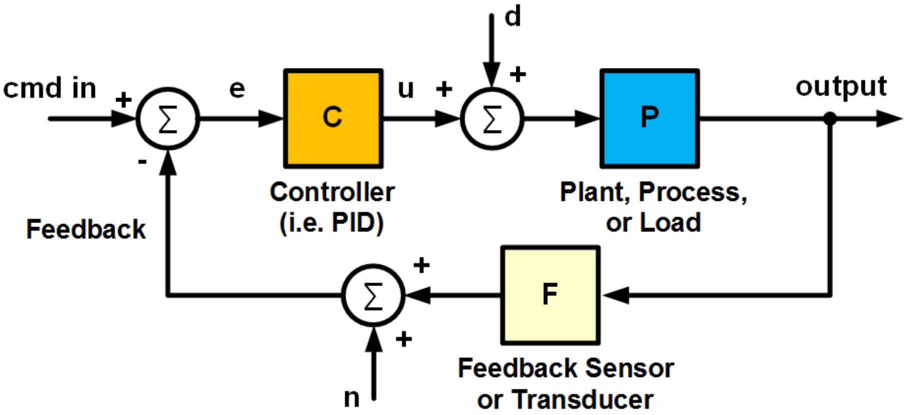


Figure 2.0 Basic Feedback Control Block Diagram

This simple loop includes a controller (C), plant (P), and feedback sensor (H). The inputs are the command input (cmd in), disturbances (d) and sensor noise (n). The command, which is the desired plant output, is the reference variable and sometimes denoted as set-point in process control applications. It is applied to the input summing junction and compared with the measured plant output. The servo error between the command and measured output is calculated and fed to the controller (C); a PID (C -> PID) as described in this course. The PID controller generates the drive signal to the plant that reduces the error until the command input equals the actual output or remains within a desired offset. In an actual control loop design, the PID could be the primary shaping function for the control loop however other compensators such as lead, lag, or lead-lag may be required to obtain the stability margins desired; compensating for the lag of specific control loop components.

Each block in the feedback control loop block diagram can be expressed in the frequency domain using the Laplace transform described in PID Controller Design Part 1 of the course. The gain of each element can be characterized as a function of frequency; effectively making each element a frequency dependent gain. The frequency domain is excellent for linear analysis since the blocks can then be treated algebraically. Referring to the block diagram of Figure 2.0, an expression for the control signal, denoted as u(s) and the output can be obtained algebraically as:

$$Control{:}u(s) = C(s)\cdot(cmd\ in(s) - (n(s) + H(s)\cdot output(s))$$

$$defining\ servo\ error\ as{:}e(s) = cmd\ in(s) - H(s)\cdot output(s)$$

$$then{:}\ u(s) = C(s)\cdot(e(s) - n(s))$$

The output is then given as:

$$output(s) = P(s)\cdot[d(s) + u(s)] = P(s)\cdot(d(s) + C(s)\cdot(e(s) - n(s)))$$

$$or,\quad output(s) = P(s)\cdot(d(s) + C(s)\cdot(cmd\ in(s) - H(s)\cdot output(s) - n(s)))$$

Solving for the output:

$$output(s) = \left[\frac{P(s)\cdot C(s)}{1 + P(s)\cdot C(s)\cdot H(s)}\right]\cdot cmd\ in(s) + \left[\frac{P(s)}{1 + P(s)\cdot C(s)\cdot H(s)}\right]\cdot d(s) - \left[\frac{P(s)\cdot C(s)}{1 + P(s)\cdot C(s)\cdot H(s)}\right]\cdot n(s)$$

The first term in brackets describes the response from the command input to the output and is termed the closed loop transfer function (CLTF) whose magnitude is called the closed loop gain (CLG). The product of all transfer functions in the loop is termed the open loop transfer function (OLTF) and its magnitude called the open loop gain (OLG). The control compensator gain, |C(s)|, is normally very high—such that magnitude of the product of the terms |P(s)*C(s)*H(s)|>> 1 over the effective operating bandwidth of the loop. The feedback term is often a unity gain (or scaled to provide unity gain in the feedback path) as it is simply sensing the plant output. This being the case, the closed loop gain, CLG is unity over the operating bandwidth of the loop and rolls off towards zero at high frequency. If the feedback gain is not unity, it will change the closed loop gain from one to the inverse of the feedback gain magnitude (i.e. 1/H). The second expression accounts for the impact of disturbances on the response. The transfer function in brackets multiplying the disturbance is often termed the compliance, as in how compliant the loop is to a disturbance. This term attenuates the disturbance primarily due to the high gain control term (C(s) = PID(s)) in the open loop gain. Therefore, the higher the controller gain, the more disturbance rejection and the less impact the disturbances will have on the output. However, this gain will be limited by the stability of the loop. The error will be a function of the attenuation provided the transfer function between the cmd in and error or [e(s)/cmd in(s) = 1/(1+P(s)*C(s)*H(s))], termed the loop sensitivity function, as well as the magnitude of the command, disturbance, and noise. Finally, as modeled, the last term is sensor noise which as shown effectively adds directly to the output since it is multiplied by the closed loop gain. As shown in Figure 2.0, it is basically input to the same summing junction as the command input, so this would be expected. The key block diagram relationships are summarized in Table 1.0.

Table 1.0 Key Block Diagram Relationships

output/input	nomenclature	transfer function	metrics	
output(s)/cmd in(s)	CLTF	$\frac{P(s)\cdot C(s)}{1+P(s)\cdot C(s)\cdot H(s)}$	CLG, Φ(CLTF)	Φ(*)-phase
feedback(s)/e(s)	OLTF	$P(s)\cdot C(s)\cdot H(s)$	OLG, Φ(OLTF)	Φ(*)-phase
e(s)/cmd in(s)	sensitivity	$\frac{1}{1+P(s)\cdot C(s)}$	magnitude	H(s)=1; n, d=0
output(s)/d(s)	compliance	$\frac{P(s)}{1+P(s)\cdot C(s)\cdot H(s)}$	magnitude	

Control specific specifications can be defined in either the time or frequency domain. In the time domain, response can be specified in terms of: rise time, overshoot, settling time, and steady state error—all of which are a function of bandwidth and the control compensator structure. In the frequency domain, the designer can specify the: resonant peak, peak frequency, gain crossover frequency, bandwidth (which in some cases are directly related), and the minimum allowable phase and gain margins. The resonant peak is a maximum of the gain at the peak frequency. Control specific specifications need to be derived from system performance specifications. In the time domain, they may relate directly; however, in the frequency domain they are often derived from disturbance rejection and maximum allowable servo error requirements—as they relate to system performance. Another important specification is the control loop type which refers to the number of integrators within the loop; either due to the controller or the plant dynamics. The loop type or number of integrators will determine the accuracy (maximum error) with which the output follows an input command or rejects a disturbance in the long term. The integrator also provides very high gain at low frequency which is good for disturbance rejection and improved accuracy; however, each pole at s=0 also provides a -90° phase shift which can impact stability. The robustness of the controller design must also be considered. If the plant parameters change for whatever reason, what will happen to the loop stability margins? The controller design must account for these variations. This can be accomplished by designing for the worst-case plant variation and ensuring the system meets the stability margins over the full range of plant variation; measuring the plant response (i.e. plant identification) and changing control parameters to account for the plant changes (either manually or on a scheduled basis), or a using an adaptive controller that automatically adjusts the control parameters as a function of the plant variations.

2.1 Plant Modeling: The transfer function for the plant can vary significantly based upon the application. As will be discussed, the PID works best with simpler first or second order plants. For process control applications, delays or dead time must often be accounted for and a simple three parameter plant process model can sometimes be used to describe its response as:

$$P(s) = (\frac{K}{1 + s \cdot \tau}) \cdot e^{-sL}$$

$$K - \text{process scaling}\,; \tau - \text{response time constant}\,; L - \text{dead or delay time}$$

Compensation for the delay is not always possible with a simple controller. If the delay is small relative to the sampling period, a simple first order Taylor Series ($e^{-sL} \approx [1\text{-}sL]$) or Pade' approximation ($e^{-sL} \approx [1\text{-}sL]/[1\text{+}sL]$) may be sufficient. However, if the delay is significant relative to the sampling period, the PID would need to be integrated with a more advanced control algorithm such as the Smith Predictor [1, 2] described in the next section of the course.

3.0 The PID Control Algorithm: As described in PID Controller Design Part 1, the PID controller structure parameterized in terms of gain, and referenced frequently in the literature, is:

$$u(t) = K_P \cdot e(t) + K_I \int_0^t e(\tau) \cdot d\tau + K_D \cdot \dot{e}(t)$$

The control signal variable 'u(t)' was defined section 4.0 and is the sum of three terms:

- A term proportional to the error with gain K_P
- A term proportional to the integral of the error with gain K_I
- A term proportional to the derivative of the error with gain K_D

This time domain representation can be converted to the frequency domain using the Laplace Transform relationships defined in PID Controller Design Part 1, as:

$$u(s) = (K_P + \frac{K_I}{s} + K_D \cdot s) \cdot e(s)$$

There are several versions of the controller that do not include all three terms; proportional-plus-integral (PI), proportional -plus- derivative (PD) or have a slightly different structure such as proportional -plus- integral -and- proportional -plus- derivative (PIPD). The controller's versatility can be observed from the impact it has on key closed loop system time response characteristics that include:

- Rise Time: time required for the plant output to reach 90% of the desired level for first time
- Overshoot: amount the peak level exceeds the steady state value; normalized to this value
- Settling Time: time required for system to reach the steady state
- Steady-state Error: difference between the steady state output and the desired output

The effect that increasing each controller parameter has on the response is summarized in Table 2.0.

Table 2.0 Effect of Increasing Controller Parameter on System Response

Parameter	Rise Time	Overshoot	Settling Time	Steady State Error
K_P	decreases	increases	indeterminate	decreases
K_I	decreases	increases	increases	eliminates for step
K_D	indeterminate	decreases	decreases	indeterminate

Therefore, an increase in K_P will decrease rise time and steady state error but increase overshoot and tendency towards oscillation. The integral term will make the system at least a Type 1 loop with the steady state error as described in Table 1.0. In general, increasing K_I reduces stability margins increasing overshoot and settling time but will decrease rise time. Finally, an increase in K_D decreases both overshoot and settling time. However, it should also be noted that increasing the term too much can lead to instability as well as amplification of noise. This last characteristic of the derivative term often dominates its performance so that in many applications it is not used. One final note regarding the use of the PID: given it only has three control parameters, it does have limitations. It can be highly effective controlling simple stable plants but, in general, not those that have higher order dynamics with high frequency modes subject to oscillation, or requirements with high precision. The PID can usually be used if the plant's response to a step input is similar to that of a first order system.

Another standard PID structure, which is actually the International Society of Automation (ISA) standard form, is parameterized by times associated with the integral and derivative. This form is often used in process control applications. The gains are parameterized relative to the time periods associated with integration and differentiation as: $K_I = K_P / T_I$ where T_I is the integral time T_I, and $K_D = K_P * T_D$ where T_D is the derivative time. With this parameterization, the preceding structure and its frequency domain equivalent can be expressed as:

$$u(t) = K_P \cdot (e(t) + \frac{1}{T_I} \cdot \int_0^t e(\tau) \cdot d\tau + T_D \cdot \dot{e}(t)) \quad \Rightarrow \quad u(s) = K_P \cdot (1 + \frac{1}{s \cdot T_I} + T_D \cdot s) \cdot e(s)$$

For either form, the proportional, integral, and derivative control terms compensate for the past, present and future error response. The integral term will always try to null the steady state error. The derivative term can improve stability with proper design but issues with this term are that it amplifies noise, especially at high frequency and if it is too large can lead to oscillation and instability in a PID controller. Noise can be attenuated by adding a pole to roll-off the frequency response or effectively filter the signal at high frequency. Effectively this is the implementation of the derivative term as a high pass filter (HPF).

There are several other forms of the PID, as listed in Table 3.0, which were described in more detail in Part 1. Different forms of the PID may apply better to different applications. For example, for process control applications tend to be slower and temporal representations are more important for tuning performance. A higher bandwidth target tracking application design may rely more on placement and tuning for a desired frequency response so that frequency parameters are more relevant.

The two standard PID algorithm representations were discussed previously: one is parameterized in terms of a gain, integral time, and derivative time while the other uses absolute controller gains. The standard forms are listed in Table 3.0 as forms 1, 2; with the derivative approximated by a HPF. A slightly different version of the controller is a PIPD in the forward path, Form 3 in the table, with a representation well suited for frequency domain analysis as both zeroes, the pole at zero, and DC gain are easily delineated. One interpretation of this form is a PI controller operating on the predicted value of the error signal. The factored form can also use the HPF as a derivative for noise attenuation.

Table 3.0 PID Controller Forms

	Parameterization	Equation
1	Standard Form Time w/Deriv Filter	$u(s) = K_P \cdot (1 + \frac{1}{s \cdot T_I} + T_D \cdot \frac{a \cdot s}{s+a}) \cdot e(s)$
2	Standard Form Gain w/Deriv Filter	$u(s) = (K_P + \frac{K_I}{s} + K_D \cdot \frac{a \cdot s}{s+a}) \cdot e(s)$
3	PIPD Forward Path	$u(s) = K'_P \cdot (1 + \frac{1}{s \cdot T'_I}) \cdot (1 + T'_D \cdot s) \cdot e(s)$
4	PI Forward Path, PD Feedback Path	$u(s) = (K'_P + \frac{K'_I}{s}) \cdot \left(cmd\ in(s) - (1 + K'_D \cdot s) \cdot output(s)\right)$

Form 4, the last implementation described, a proportional -plus- integral in the forward path operating on the error with a proportional plus derivative (PIPD) controller in the output feedback path. The PD controller proportional gain must be one for zero steady state error. Also as the derivative operates on the output rather than the error which is often a smoother function, this form generates a less noise in its response.

<u>3.1 Plant Delays and Smith Predictor:</u> As discussed, many plants, especially in process control, have a delay that impacts the desired performance. A delay is effectively dead time between the application of the control signal and the time it takes the plant respond. Process dead time is generally due to transport delays in the time it takes material to move from the actuator location to the sensor location that senses the response, which can be quite long. Delays in applications, such as target tracking, are often due to the processing associated with a frame of data relative to the camera frame rate. The delay will have a detrimental impact on controller response. If the delay is small relative to the plant time constant, derivative prediction or phase compensation using approximations for the delay function may improve response. An indicator of the significance of the delay can be obtained by the relative delay ratio:

$$\eta = \frac{L}{L+T}$$

It provides an estimate of the temporal response driver; the response time lag, T, or the processing delay, L. PI control often suffices for processes that are delay dominated, i.e. when η is close to one while derivative action is typically beneficial for processes with small relative delay η. In general, a delay will limit performance by limiting gain or bandwidth such that the gain crossover frequency in bounded by 1/L. As an example, for the plant model discussed in section 2, assume a proportional only controller is used in a unity feedback system so the closed loop response is given by (assume K and K_P >0):

$$CLTF(s) = \frac{K_P \cdot (\frac{K}{1+s\cdot\tau}) \cdot e^{-sL}}{1 + K_P \cdot (\frac{K}{1+s\cdot\tau}) \cdot e^{-sL}} = \frac{K_P \cdot K \cdot e^{-sL}}{1 + s\cdot\tau + K_P \cdot K \cdot e^{-sL}}$$

If the delay is small enough for a first order approximation $e^{-sL} \approx 1 - Ls$ then:

$$CLTF(s) \cong \frac{K_P \cdot K \cdot (1 - s\cdot L)}{1 + s\cdot\tau + K_P \cdot K \cdot (1 - s \cdot L)} = \frac{K_P \cdot K \cdot (1 - s\cdot L)}{s\cdot(\tau - K_P \cdot K \cdot L) + 1 + K_P \cdot K}$$

The pole must negative for a stable response so examining the coefficient of the ‘s’ term $K_P*K < \tau/L$ is required to meet this criterion. This limits the proportional gain significantly if L>τ. One final example is if a PI controller were used such that:

$$CLTF(s) = \frac{K_P \cdot (1 + \frac{1}{T_I \cdot s}) \cdot (\frac{K}{1+s\cdot\tau}) \cdot e^{-sL}}{1 + K_P \cdot (1 + \frac{1}{T_I \cdot s}) \cdot (\frac{K}{1+s\cdot\tau}) \cdot e^{-sL}} = \frac{K_P \cdot K \cdot (T_I \cdot s + 1) e^{-sL}}{(1 + s\cdot\tau)\cdot T_I \cdot s + K_P \cdot K \cdot (T_I \cdot s + 1)\cdot e^{-sL}}$$

Using the first order approximation:

$$\begin{aligned} CLTF(s) &= \frac{K_P \cdot K \cdot (T_I \cdot s + 1)\cdot(1 - s\cdot L)}{(1 + s\cdot\tau)\cdot T_I \cdot s + K_P \cdot K \cdot (T_I \cdot s + 1)\cdot(1 - s \cdot L)} \\ &= \frac{K_P \cdot K \cdot (T_I \cdot s + 1)\cdot(1 - s\cdot L)}{(\tau \cdot T_I - K_P \cdot K \cdot L \cdot T_I)\cdot s^2 + (T_I + K_P \cdot K \cdot (T_I - L))\cdot s + K_P \cdot K} \end{aligned}$$

The first term provides the same bound as derived previously; $K_P*K < \tau/L$ so that τ > L is the desired condition. The second term provides a lower bound on the integrator $T_I > L*(K_P*K/(1+K_P*K)$. This last result indicates the integrator time must be greater than the dead given the assumed delay time. This approach will not work with long delays, as integrating over longer delay periods can lead to integrator ramp-up, saturation, and potential instability since the controller will try and over-compensate as it appears to be applying a control signal without response.

To improve performance for longer delays, an algorithm termed the Smith Predictor (O.J.M. Smith, U. of California at Berkeley, in 1957) is often used. The Smith predictor is a model-based controller which can be effective for processes with long delays. The basic feedback loop with the Smith predictor is shown in Figure 3.0. The plant P(s) includes a delay as with the model described previously. The block termed $P_M(s)$ is a model of the plant without the delay while the block e-^{LmS} is the estimate of the delay. The inner loop contains the nominal controller that can be designed without accounting for the dead time. It uses the process model without dead time to predict the output, feeding it back to the main controller—which generates a control signal such that the output tracks the input. The controller gain can be selected to be to achieve fast and well-damped set point responses. The effects of load disturbance and small modeling error are corrected through an outer loop by feeding back the predictor error e_p.

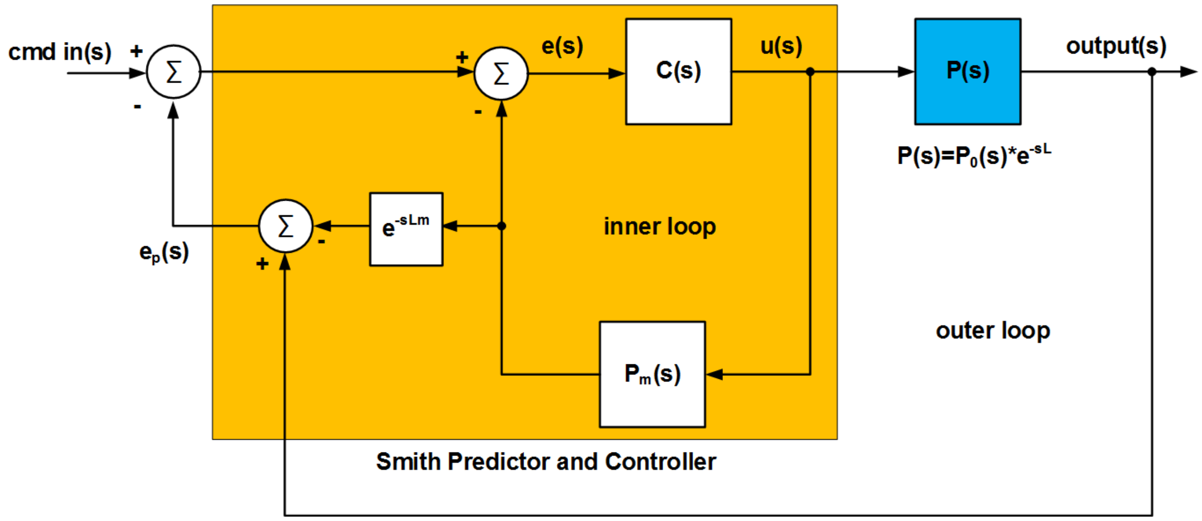


Figure 3.0 Functional Block Diagram for Smith Predictor

Referring to Figure 3.0, the CLTF of this loop is:

$$CLTF(s) = \frac{C(s)\cdot P_0(s)\cdot e^{-sL}}{1 + C(s)\cdot P_M(s)\cdot\left(1 - e^{-sL_M}\right) + C(s)\cdot P_0(s)\cdot e^{-sL}}$$

$$if \quad P_M(s)\cdot e^{-sL_M} = P(s) = P_0(s)\cdot e^{-sL} \Rightarrow P_M(s) = P_0(s) \quad ; \quad e^{-sL_M} = e^{-sL}$$

$$CLTF(s) = \frac{C(s)\cdot P_0(s)\cdot e^{-sL}}{1 + C(s)\cdot P_M(s)}$$

The predictor removes the time delay from the denominator of the closed loop transfer function, improving system performance. However, it will still respond to input variations with a time delay since it in the numerator of the closed loop response. Removing the delay from the denominator is significant however—since it no longer impacts loop stability as discussed previously. This allows for the conventional controller C(s) (PI, PD, or PID) to be tuned for non-delay operation based upon system performance objectives because the effect of the time delay in the feedback loop has been minimized. All this assumes that the model response and delay exactly matches the actual plant response and delay. If this is not the case, the effect of the delay may still be present in the response, even leading to instability. A thorough analysis of the plant response is required and robustness of design to response and delay variations completely analyzed. For example, assume the desired model response without delay is given by:

$$CLTF_M(s) = \frac{1}{1+s\cdot\tau_M}$$

From equation for the CLTF:

$$\frac{CLTF(s)}{e^{-L\cdot s}} = \frac{C(s)\cdot P_0(s)}{1+C(s)\cdot P_0(s)} \quad ; \quad P_M(s) = P_0(s)$$

It is desired that:

$$\frac{CLTF(s)}{e^{-L\cdot s}} = CLTF_M(s) \quad or \quad \frac{C(s)\cdot P_0(s)}{1+C(s)\cdot P_0(s)} = \frac{1}{1+s\cdot\tau_M}$$

The PI controller and plant without delay are given by:

$$\text{controller}: C(s) = PI(s) = K_P\cdot(1+\frac{1}{T_I\cdot s}) = K_P\cdot\frac{(T_I\cdot s+1)}{T_I\cdot s} \quad ; \quad \text{plant}: P_0(s) = \frac{K}{1+s\cdot\tau}$$

So that from the above desired model equality:

$$\frac{K_P\cdot\frac{(1+T_I\cdot s)}{T_I\cdot s}\cdot\frac{K}{1+s\cdot\tau}}{1+K_P\cdot\frac{(1+T_I\cdot s)}{T_I\cdot s}\cdot\frac{K}{1+s\cdot\tau}} = \frac{1}{1+s\cdot\tau_M}$$

Letting the integrator time equal the plant response time or, $T_I = \tau$:

$$\frac{1}{1+s\cdot\tau_M} = \frac{K_P\cdot K\cdot}{T_I\cdot s+K_P\cdot K} \quad for \quad K_P = \frac{T_I}{K\cdot\tau_M}$$

<u>3.2 Digital Implementation and Design of PID Controller:</u> Most of control algorithms today are implemented in an embedded or digital signal processor. Digital control and sample-data control are different ways of expressing the same process with the digital terminology relating more to the algorithm implementation in HW/SW and sample data the mathematical process. Signals to be measured are sampled at a defined rate and commands to control the plant or load are transmitted at the same rate (although multi-rate systems are also used). The sampling functions are realized with analog to digital converters (ADC) or digital to analog converters (DAC). These components sample signals or generate commands at a specified sample time interval (Δ). Mathematically these devices can be modeled as zero order hold (ZOH) functions which sample an input or apply an output at a specified time increment; holding that value until the next time increment. Inherent with this approach is internal sampling within the processor. The sampling has a negative impact on response since the sample period is in effect a delay or dead time producing a phase delay in the control loop. A typical operational sequence is shown in Figure 4.0.

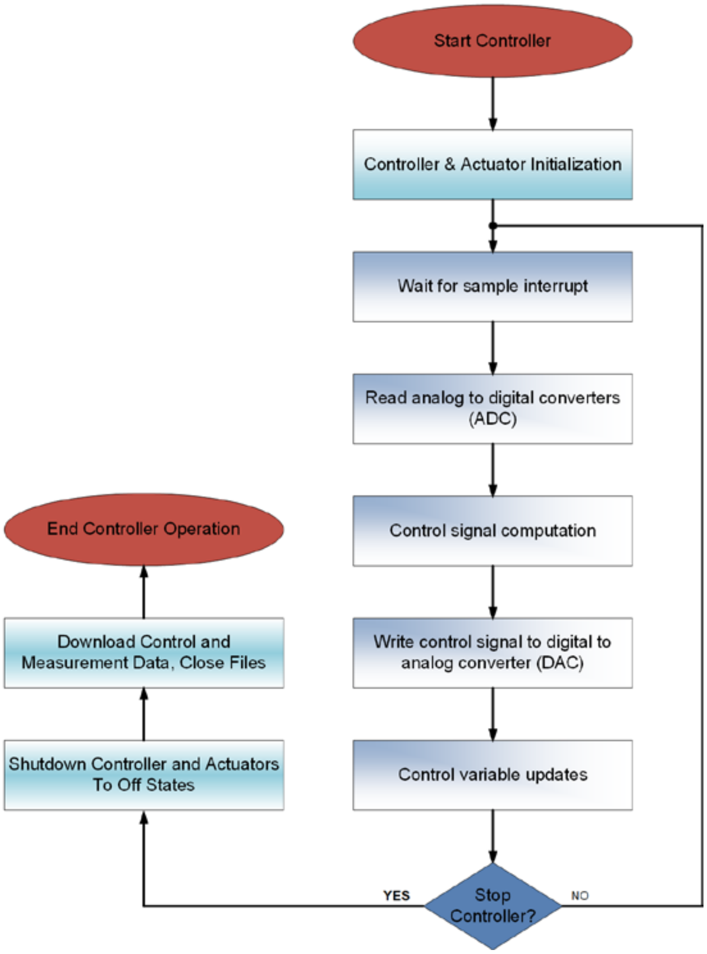

Start Controller
Controller & Actuator Initialization
Wait for sample interrupt
Read analog to digital converters (ADC)
Control signal computation
Write control signal to digital to analog converter (DAC)
Control variable updates
Stop Controller?
YES
NO
Shutdown Controller and Actuators To Off States
Download Control and Measurement Data, Close Files
End Controller Operation

Figure 4.0 Typical Digital Controller Operational Sequence

A primary objective of a digital implementation is to ensure all computations and I/O can be performed within one sampling period or less. Often the most significant portion of the processing time is with the I/O, algorithm processing is generally much faster although video image processing algorithms or tracking and surveillance applications can be computationally intensive.

3.2.1 Sample Data Domain: As with continuous time control system design and analysis; with the sampled data time control system, there are also mathematics related to representations in both the time domain and frequency domain. Sampling limits the maximum measurable frequency and therefore the bandwidth of the control loop. Nyquist's theorem states '*the sampling rate must be at least twice the frequency of the highest frequency component in a signal being sampled*'. With sampling, the infinite continuous frequency spectrum is mapped to a spectrum whose maximum frequency is half the sampling frequency. So, the compensator C(s) must be converted to a digital format; transforming it from a frequency dependent function the infinite continuous frequency spectrum into the sample spectrum limited at half the sampling frequency. Many textbooks are available on the theory of sample-data systems as is an excellent Suncam Course EE 060

"Converting Feedback Systems from Analog to Digital Control". The Z-transform is often used to describe sampled performance in the frequency domain. The definition of the transform can be derived from taking the Laplace transform of a sampled function $f_s(t)$ given by:

$$\mathrm{f_S}(t) = \sum_{n=0}^{\infty} f(n \cdot \Delta) \cdot \delta(t - n \cdot \Delta)$$

The delta function defines the sampling instants at Δ intervals. The Laplace transform is obtained as:

$$\mathrm{L}[\mathrm{f_S}(t)] = L[\sum_{n=0}^{\infty} f(n \cdot \Delta) \cdot \delta(t - n \cdot \Delta)] = \sum_{n=0}^{\infty} f(n \cdot \Delta) \cdot e^{-n \cdot \Delta \cdot s}$$

The exponential term effectively defines the phase of the signal at sampling instants. The z-transform operator is defined as:

$$\mathrm{z} = e^{\Delta \cdot s} = \cos(\Delta \cdot s) + j \cdot \sin(\Delta \cdot s)$$
$$s - Laplace\ operator$$

An important consequence of this definition is that the infinite continuous frequency spectrum (s) is mapped to a variable whose maximum amplitude is one. So, the mapping of the Laplace domain to the Z-domain is effectively a mapping from an infinite frequency spectrum into a unit circle. Using this definition, the Laplace transform of the sampled function is related to the z-transform as:

$$\mathrm{L}[\mathrm{f_S}(t)] = \sum_{n=0}^{\infty} f(n \cdot \Delta) \cdot z^{-n}$$

The general relationship between Laplace (L) and Z (Z) transform is then:

$$\mathrm{L}[\mathrm{f_S}(t)] = Z[f(t)]$$

One of the key properties of the z-transform is the shifting property (shifting in time) which allows for an easy transition between time domain sampled functions described by difference equations to representations in the sampled frequency domain. From the definition of the z-transform:

$$Z[f_s(t - \tau)] = \sum_{n=0}^{\infty} f((n-k) \cdot \Delta) \cdot z^{-n}$$

For sampling time Δ*n with Δ*k delay at sampling instants. Letting m=n-k this expression can be written as:

$$Z[f_s(t - \tau)] = \sum_{k=0}^{\infty} f(m \cdot \Delta) \cdot z^{-m-k} = [\sum_{k=0}^{\infty} f(m \cdot \Delta) \cdot z^{-m}] \cdot z^{-k} = z^{-k} \cdot F(z)$$

The general shifting property to convert between sampled time difference equations and the frequency domain z-transform is:

$$\mathrm{Z}[\mathrm{f_s(k - n)}] = z^{-k} \cdot F(z) \quad ; \quad \mathrm{f_s(k - n)} = Z^{-1}[z^{-k} \cdot F(z)]$$

The Z-transform theory parallels that of the Laplace transform since they are directly related and is described in detail in most automatic control textbooks. Numerous references provide tables with Laplace <-> Z conversions. It is important to realize that when converting from a continuous to sampled-data domain, a direct conversion can be made only if the continuous function has a sampler at the input and output. Three examples are shown in Figure 5.0.

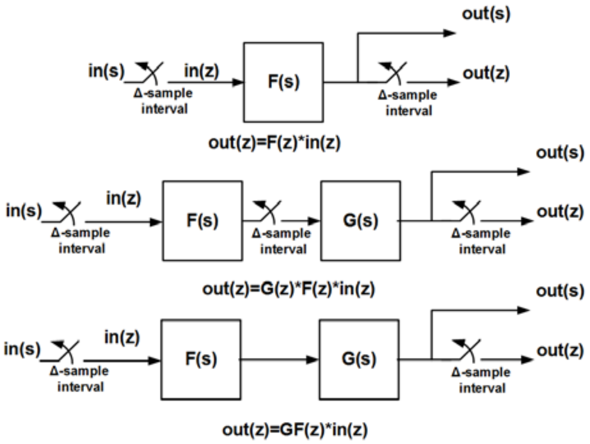


Figure 5.0 Block Diagram Implementations of Sampled Configurations

The first case is the baseline sampling configuration with the equation for the z-transform below. The middle diagram shows two function blocks, F and G, with samplers between each with the z-transform below. Each block can be transformed as separate entities from the Laplace to z-domain. The last diagram is similar to the middle, except there is not a sampler between F and G. Now the z-transform must be taken of the composite F*G function and they cannot be converted as separate entities. In general, if a continuous transfer function has sampled input, it has a direct conversion to the sampled data version, however, if several transfer functions are cascaded then the z transform must be taken of all cascaded blocks. This will primarily impact a representation of the physical process. The digital controller, even if several functional elements, can normally be represented as sampled between each. Applying these sampling rules and considerations to the basic feedback control block diagram in Figure 2.0, a sample data version with all blocks represented with input samplers is shown in Figure 6.0a. For this case, the relationships provided in Table 1.0 apply directly with substitution of ‘z’ for ‘s’ as shown with the z transform equivalent in Figure 6.0b. The noise and disturbance inputs are not included as they are applied directly to the process model without samplers—so would need to be cascaded with the plant model before conversion to the z-domain.

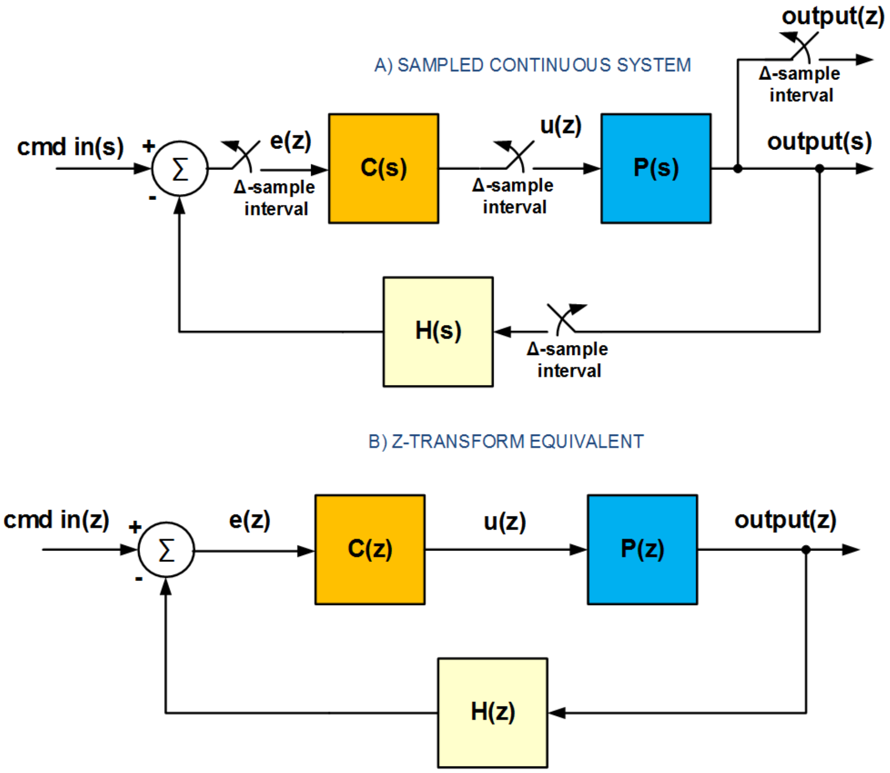


Figure 6.0 Basic Sample Data Feedback Control Block Diagram

The sampler is represented mathematically by a hold function, must often the zero-order hold (ZOH) is used which has the Laplace transform given by:

$$G_{ZOH}(s) = \frac{1 - e^{-\Delta \cdot s}}{s}$$

This function converts an impulse of arbitrary amplitude and generates a fixed constant pulse of the same amplitude between samplings. For Δ*s <<1 the Pade approximation for $e^{-\Delta s}$ can be used as so that $G_{ZOH}(s)$ can be approximated as:

$$G_{ZOH}(s) = \frac{1 - e^{-\Delta \cdot s}}{s} \approx \frac{1}{s} \cdot \left(1 - \frac{1 - \frac{\Delta}{2} \cdot s}{1 + \frac{\Delta}{2} \cdot s} \right) = \frac{\Delta}{1 + \frac{\Delta}{2} \cdot s} \quad ; \quad e^{-\Delta \cdot s} \approx \frac{1 - \frac{\Delta}{2} \cdot s}{1 + \frac{\Delta}{2} \cdot s}$$

3.2.2 PID Digital Implementation: To convert from the continuous to digital domain in a readily implementable form, often approximations are used. Pole zero matching is one technique the directly maps poles and zeroes of a continuous transfer function to poles and zeroes of a sampled data transfer function: p_i, z_i -> $e^{\Delta pi}$, $e^{\Delta zi}$. It can be seen that negative poles and zeroes will map inside a unit circle. With this approach, the sample data transfer function must be scaled to achieve an equivalent amplitude as the continuous function at a critical frequency. Another s-domain to one in the z-domain conversion method is the bilinear transformation with frequency pre-warping is one method. It is derived from the definition of $z=e^{\Delta s}$, solving for $s = (1/\Delta)*\ln(z)$ and taking Taylor Series expansion of the log. It provides good results for converting continuous to sampled controller format, assuming a high enough sample rate. For a specified continuous transfer function G(s), the discrete equivalent can be determined from:

$$\mathrm{G(z)} = G(s)\,|_{s=\frac{2}{\Delta}\frac{z-1}{z+1}}$$

$$s = \frac{2}{\Delta}\frac{z-1}{z+1} \text{ is the bilinear transformation}$$

Frequency pre-warping compensates for the frequency distortion that occurs when mapping the infinite continuous frequency spectrum to a sample limited frequency spectrum using this transformation (i.e. f-continuous 0-> ∞ maps to f-sampled 0->$f_s/2$). An advantage of using this approach is that it maps stable poles and zeroes (roots in left half s-plane) inside the unit circle guaranteeing the stability of the digital filter from a stable analog filter. Critical frequencies in the continuous domain are modified as:

$$\mathrm{f_S} = \frac{1}{\pi\cdot\Delta}\tan(\pi\cdot\mathrm{f_C}\cdot\Delta)$$

Using this transform, a digital representation of the time parameterized standard PID controller form with a roll-off filtered derivative term, as described in Table 2.0 Form 1, is obtained. The controller is given by:

$$u(s) = PID(s)\cdot e(s)$$

A standard form of the PID is used:

$$PID(s) = K_P\cdot(1+\frac{1}{T_I}\cdot\frac{1}{s}+a'\cdot T_D\cdot\frac{s}{s+a'})$$

This is converted to a sample data controller using the bilinear transformation with frequency pre-warping as:

$$PID(z) = K_P\cdot(1+\frac{\Delta}{2\cdot T_I}\cdot\frac{z+1}{z-1}+\frac{a\cdot T_D}{1+\frac{\Delta\cdot a}{2}}\cdot\frac{z-1}{z-\beta})$$

$$\beta = \frac{1-\frac{\Delta\cdot a}{2}}{1+\frac{\Delta\cdot a}{2}} \quad ; \quad a = \frac{1}{\pi\cdot\Delta}\tan(\pi\cdot a'\cdot\Delta)$$

These expressions are then converted to the time domain so they can be used in a digital controller. The control signal and error are sampled signals with the control signal given by:

$$u(z) = PID(z) \cdot e(z)$$

The three terms of the PID can be defined as:

$$\begin{aligned}
&u(z) = u_P(z) + u_I(z) + u_D(z) \\
&u_P(z) = K_P \cdot e(z) \\
&u_I(z) = K_P \cdot \frac{\Delta}{2 \cdot T_I} \cdot \frac{z+1}{z-1} \cdot e(z) \\
&u_D(z) = K_P \cdot \frac{a \cdot T_D}{1 + \frac{\Delta \cdot a}{2}} \cdot \frac{z-1}{z-\beta} \cdot e(z)
\end{aligned}$$

Each term can then be converted to a digital time format using the time shifting property of the z-transform discussed previously, or u(k-1)=Z^{-1}[u(z)z^{-1}]:

$$\begin{aligned}
&u(k) = u_P(k) + u_I(k) + u_D(k) \\
&u_P(k) = K_P \cdot e(k) \\
&u_I(k) = u_I(k-1) + K_P \cdot \frac{\Delta}{2 \cdot T_I} \cdot \left(e(k) + e(k-1)\right) \\
&u_D(k) = \beta \cdot u_D(k-1) + K_P \cdot \frac{a \cdot T_D}{1 + \frac{\Delta \cdot a}{2}} \cdot \left(e(k) - e(k-1)\right)
\end{aligned}$$

With a digital implementation, another form of the controller that can be used is termed the incremental implementation, given by:

$$\begin{aligned}
&\delta u(k) = u(k) - u(k-1) = \delta u_P(k) + \delta u_I(k) + \delta u_D(k) \\
&\delta u_P(k) = u_P(k) - u_P(k-1) = K_P \cdot \delta e(k) = K_P \cdot (e(k) - e(k-1)) \\
&\delta u_I(k) = \delta u_I(k-1) + K_P \cdot \frac{\Delta}{2 \cdot T_I} \cdot \left(e(k) - e(k-2)\right) \\
&\delta u_D(k) = \beta \cdot \delta u_D(k-1) + K_P \cdot \frac{a \cdot T_D}{1 + \frac{\Delta \cdot a}{2}} \cdot \left(e(k) - 2 \cdot e(k-1) + e(k-2)\right)
\end{aligned}$$

The controller output is the sum of the individual increments for each term. Being increments of the controller terms, their magnitudes are in general smaller which computationally can be an advantage. The final controller output is just the integral or sum of the increments:

$$u(n\Delta) = \sum_{k=0}^{n} \delta u(k\Delta) = u((n-1)\Delta) + \delta u(n\Delta)$$

3.3 Digital Control Loop Performance Specifications and Design: As discussed in PID Controller Design Part 1, the control loop performance specification defines the desired feedback loop performance. Most of the design techniques described previously have an equivalent in the sample-data domain. Key digital implementation issues are:

- sample rate limits bandwidth to < half the sampling frequency per the Nyquist theorem
- sampling causes phase delay; roughly the time delay relative to the system bandwidth
- Most digital implementations require a real-time operating system and/or a dedicated digital signal processor or FPGA.
- Aliasing refers to a condition when there are high signal frequencies greater than half the sampling frequency which get represented as low frequency components. An anti-aliasing filter (low pass filter at or near half sample rate) implemented before sampling the signal resolves this issue.

The design process, as discussed, still applies beginning with understanding user requirements, operating environment and limitations; to evaluating design tradeoffs between the user requirements and control loop specifications. Design methods specific to a continuous PID design can also be used with a digital design. In general, the higher the sample rate relative to a desired bandwidth, the better the sampled system will emulate its continuous equivalent. Greater than a 10:1 ratio is a good goal.

The manual tuning techniques based on process measurements developed by Ziegler and Nichols are applicable regardless a continuous or digital design, simply the implementation changes subject to the sample frequency constraints. Analytical design approaches such as model matching or pole-zero placement have digital equivalents. There are, however, alternative design approaches; direct digital design or converting a continuous design to a digital design. If the design can sustain high sample rates relative to the desired system and component bandwidths, then continuous to digital conversion can work, especially using a Bilinear Transform. If this is not the case, then digital analysis techniques similar to those for a continuous design are best used for loop shaping with algebraic pole-zero placement or model matching. As with the continuous design, with model matching the excess number of poles of the model must be >=excess number of poles of the plant and should not have more than one simple pole on the unit circle for stability.

Design Issues, described in Part 1, are still a factor regardless of a continuous or digital system, and their impact on PID controller performance must be considered. Simple controllers like the PI and PID controller are not suitable for all processes. The derivative term will almost always be a noise amplifier, implementation of the derivative as a high pass filter with a roll-off frequency reduces this problem. Integrator windup remains an issue with a digital design and methods described in Part 1 to mitigate it are still effective.

4.0 Digital Home Heating System: This example was analyzed as a continuous time system in PID Controller Design Part 1. It is now considered as a sample data control system for digital controller implementation on a processor. The physical configuration is illustrated in Figure 7.0. The house is idealized as a box filled with air at a uniform temperature T_C. The walls of the house are considered as pure resistance to heat transfer with no energy storage capacity. The overall coefficient of heat transfer is U and the heat transfer area is A. The outdoor environment temperature is T_e, varies with time thereby acting as a disturbance to the control system. The temperature T_C is measured by a temperature sensor in a thermostat or temperature controller mounted inside the house. The desired temperature can also be set by this device, or in today's environment may even interface to a smart phone with an APP that lets it be set remotely. It is assumed that the temperature is converted to a voltage (or current) with a scaling constant K_{TV} (with units Volt/°F) and processed by a digital micro-controller.

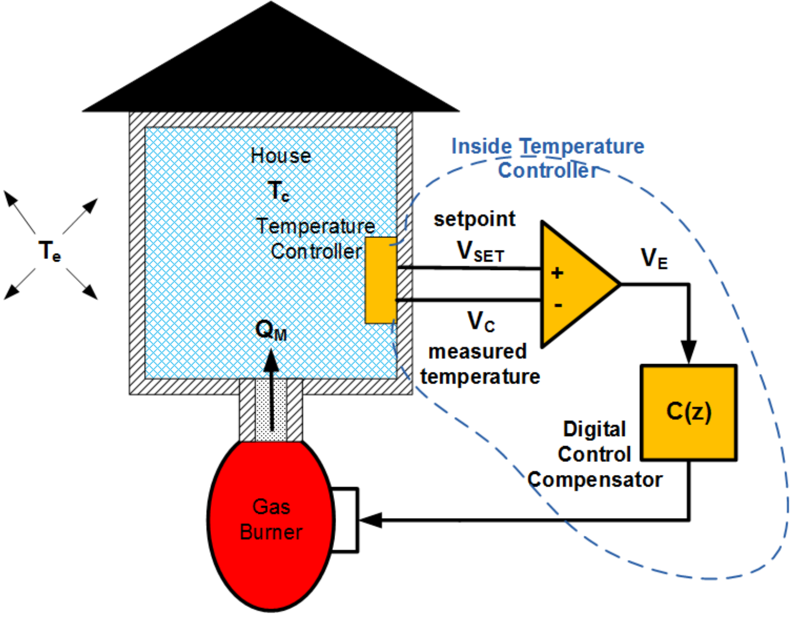


Figure 7.0 Concept Feedback Control Configuration for Heating a House

The digital controller will take the difference between the desired and actual temperature and generates a control signal to the furnace. This signal controls an actuator that increases or decreases the gas flow and effectively the amount of heat into the room. For this simple example, the actuation and heat generation are modeled as a linear process; in reality, it is a complex process. In developing a model of the system, it is assumed that initially the house is in equilibrium with constant values of T_C and T_e. The furnace will then be supplying sufficient heat to balance the heat loss to the environment. Any disturbance or change in the desired temperature will result in an increase or decrease in heat input from this original value. All variables should be considered deviations from their initial equilibrium condition. The simple first order thermal dynamic model for the heat balance in the house (the control loop plant) in the time domain is:

$$\textit{thermal energy stored} = \textit{thermal energy in} - \textit{thermal energy leaving the house}$$

$$or \qquad M \cdot C_P \cdot \dot{T}_C(t) = Q_M(t) - U \cdot A \cdot (T_C(t) - T_e(t))$$

$$where \quad Q_M - thermal\ energy\ in\ \ \frac{BTU}{hr}$$

$$M - mass\ of\ air\ in\ house$$

$$C_P - specific\ heat\ of\ air\ at\ cons\tan t\ pressure$$

$$assume \quad U \cdot A = 150\,\frac{BTU}{Hr \cdot {}^\circ F} \quad ; \quad M \cdot C_P = 180\,\frac{BTU}{{}^\circ F}$$

Converting to the frequency domain and rearranging terms:

$$T_C(s) = \frac{Q_M(s)}{U \cdot A \cdot (\tau \cdot s + 1)} + \frac{T_e(s)}{\tau \cdot s + 1}$$

$$where \quad \tau = \frac{M \cdot C_P}{U \cdot A} = 1.2 Hr = 4320\ \text{sec}$$

The plant model is then given by:

$$P(s) = \frac{1}{U \cdot A \cdot (\tau \cdot s + 1)} = \frac{1}{150} \cdot (4320 \cdot s + 1)\ \frac{{}^\circ F}{BTU / Hr}$$

The temperature T_C is then given by:

$$T_C(s) = P(s) \cdot (Q_M(s) + U \cdot A \cdot T_e(s))$$

The control loop in a sample data format is required for the digital controller design. The plant model is converted to sample format by weighing it by the zero order hold function, $G_{ZOH}(s)$, and then taking the Z-transform, or:

$$P(z) = Z\left\{\frac{1 - e^{-\Delta \cdot s}}{s} \cdot P(s)\right\} = Z\left\{\frac{1 - e^{-\Delta \cdot s}}{s} \cdot \frac{1}{U \cdot A} \cdot \frac{1}{(\tau \cdot s + 1)}\right\}$$

$$= \frac{1}{U \cdot A} \cdot (1 - z^{-1}) \cdot Z\left\{\frac{1}{s} \cdot \frac{1}{(\tau \cdot s + 1)}\right\}$$

Using standard La Place to Z transform conversion tables (effectively use pole-zero mapping), the transform of the last term is:

$$Z\left\{\frac{1}{s} \cdot \frac{1}{(\tau \cdot s + 1)}\right\} = \frac{z \cdot \left(1 - e^{-\frac{\Delta}{\tau}}\right)}{(z - 1) \cdot \left(z - e^{-\frac{\Delta}{\tau}}\right)}$$

Substituting into the expression for P(z) results in:

$$P(z)=\frac{1}{U\cdot A}\cdot\frac{\left(1-e^{-\frac{\Delta}{\tau}}\right)}{\left(z-e^{-\frac{\Delta}{\tau}}\right)}=\frac{1}{U\cdot A}\cdot\frac{1-\beta}{z-\beta}=\frac{2.36\cdot10^{-6}}{150}\cdot\frac{1}{(z-9.9999764\cdot10^{-1})}\quad\frac{{}^\circ F}{BTU/Hr}$$

$$where\quad \Delta=0.01\sec\quad;\quad \beta=e^{-\frac{\Delta}{\tau}}\sim 9.9999764\cdot10^{-1}$$

The 100 Hz sample rate is much higher than required for the loop response time, however it is a rate that should be achievable even with low end real time control processors and this rate should mitigate the need for any frequency pre-warping. Conversion of the expression for temperature T_C follows the rules described in section 3.0. The temperature control is the heat input, assumed proportional to the sampled temperature difference scaled by K_{VQ}, and therefore also a sampled quantity. This is a simplification for the example. The external temperature is not sampled however so that the Z-transform must be taken as the product of the continuous external and plant temperatures (PT_e). The sampled expression for temperature T_C is:

$$T_C(z)=P(z)\cdot Q_M(z)+U\cdot A\cdot PT_e(z)$$

$$PT_e(z)=Z\{P(s)\cdot T_e(s)\}$$

As with the continuous time system, the control loop shown in Figure 8.0 measures the house temperature T_C , compares it to a desired temperature T_S or the set point and adjust the heat input using the digital controller C(z).

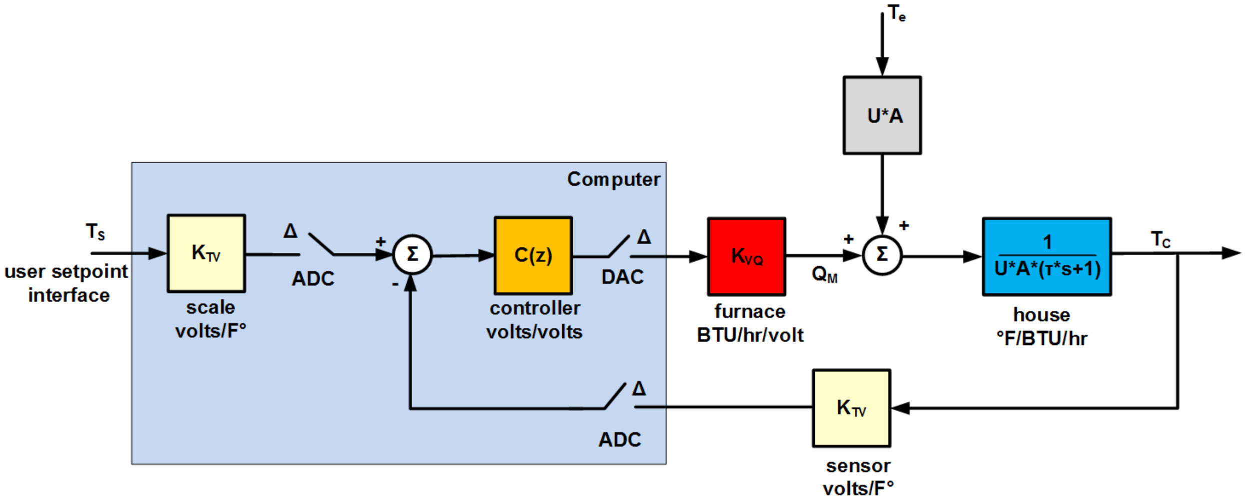


Figure 8.0 Block diagram of home heating control servo loop

From the block diagram, Q_M is given by:

$$Q_M(z)=K_{VQ}\cdot C(z)\cdot K_{TV}\cdot(T_S(z)-T_C(z)\quad BTU/HR$$

Substituting into the expression for temperature T_C the loop dynamics can be expressed as:

$$T_C(z) = U \cdot A \cdot PT_e(z) + K_{VQ} \cdot P(z) \cdot C(z) \cdot (K_{TV} \cdot (T_S(z) - T_C(z))$$

$$assume \quad K_{VQ} \cdot K_{TV} = 0.067 \frac{BTU/HR}{°F} \quad \Rightarrow \left(\frac{BTU/HR}{volt}\right) \cdot \left(\frac{volt}{°F}\right)$$

Solving for T_C:

$$T_C(z) = \frac{U \cdot A \cdot PT_e(z)}{1 + K_{VQ} \cdot K_{TV} \cdot P(z) \cdot C(z)} + \frac{K_{VQ} \cdot K_{TV} \cdot P(z) \cdot C(z) \cdot T_S(z)}{1 + K_{VQ} \cdot K_{TV} \cdot P(z) \cdot C(z)}$$

The error between the set point temperature and the actual temperature is given by:

$$T_S(z) - T_C(z) = -\frac{U \cdot A \cdot PT_e(z)}{1 + K_{VQ} \cdot K_{TV} \cdot P(z) \cdot C(z)} + \frac{T_S(z)}{1 + K_{VQ} \cdot K_{TV} \cdot P(z) \cdot C(z)}$$

As discussed previously, the error is a function of the magnitude of the outside and set point temperatures and the error attenuation provided by the controller gain. The simplest controller is a proportional gain or C(z) = K_P with units of units of volts /volts. Substituting into the previous expression and using the plant definition:

$$T_S(z) - T_C(z) = \left(\frac{z - \beta}{z - \varphi}\right) \cdot \left[-U \cdot A \cdot PT_e(z) + T_S(z)\right]$$

$$where \quad \varphi = \left(\beta - \frac{K_{VQ} \cdot K_{TV} \cdot K_P \cdot (1 - \beta)}{U \cdot A}\right)$$

As this is a Type 0 system, there will be a steady state error to a step change. For a step change in δT_e and δT_S, the steady state error is given by:

$$E_{SS} = Lim_{z \to 1} \left\{(z - 1) \cdot (T_S(z) - T_C(z))\right\}$$

$$E_{SS} = Lim_{z \to 1} \left\{(z - 1) \cdot \left(\left(\frac{z - \beta}{z - \varphi}\right) \cdot \left[-U \cdot A \cdot \delta PT_e(z) + \frac{\delta T_S \cdot z}{z - 1}\right]\right)\right\}$$

For a step in the external temperature, but without sampling the transfer function δPT_e is given by:

$$\delta PT_e(z) = \frac{\delta T_e}{U \cdot A} \frac{z \cdot (1 - \beta)}{(z - 1) \cdot (z - \beta)}$$

The steady state error is then:

$$E_{SS} = Lim_{z\to 1}\left\{(z-1)\cdot\left(\left(\frac{z-\beta}{z-\varphi}\right)\cdot\left[-U\cdot A\cdot\frac{1}{U\cdot A}\frac{\delta T_e\cdot z\cdot(1-\beta)}{(z-1)\cdot(z-\beta)}+\frac{\delta T_S\cdot z}{z-1}\right]\right)\right\}$$

$$E_{SS} = Lim_{z\to 1}\left\{-\frac{\delta T_e\cdot z\cdot(1-\beta)}{(z-\varphi)}+\frac{\delta T_S\cdot z\cdot(z-\beta)}{(z-\varphi)}\right\} = \left(\frac{1-\beta}{1-\varphi}\right)\cdot(-\delta T_e+\delta T_S)$$

$$E_{SS} = \left(\frac{U\cdot A}{U\cdot A+K_{VQ}\cdot K_{TV}\cdot K_P}\right)\cdot(-\delta T_e+\delta T_S) = \lambda_s\cdot(-\delta T_e+\delta T_S)$$

Note that this result is the same obtained for the continuous case. The step change in the input will most likely come from a new set point, as the temperature of the environment would not be expected to change instantaneously so: $E_{SS} = \lambda_s\cdot\delta T_S$ to a step change. The gain K_P is chosen to provide an attenuation factor λ_s that is consistent with the allowable difference between the new set point and actual temperature. For example, with the gain chosen as: $K_P = \dfrac{100\cdot U\cdot A}{K_{VQ}\cdot K_{TV}}$ the resulting attenuation would be a factor of 100 or -40 dB. The achievable gain will also depend upon the actual drive characteristics of the furnace. The gain crossover can be determined by a Bode plot or estimated directly from the OLTF as f_{GC}~0.004 Hz.

Given the baseline P-only results, a proportional plus integral controller (PI) design is examined next. This implementation results in a Type 1 loop with the steady state offset to a step equal to zero. Using the bilinear transform as discussed in section 3.0, the PI compensator has the form:

$$C(z) = K_P\cdot(1+\frac{\alpha\cdot\Delta}{2}\frac{z+1}{z-1})\quad or\quad C(z) = K_P'\cdot\frac{z-\gamma}{z-1}\quad for$$

$$K_P = \frac{420.0\cdot U\cdot A}{K_{VQ}\cdot K_{TV}}\quad ;\quad K_P' = K_P\cdot\left(1+\frac{\alpha\cdot\Delta}{2}\right)\quad ;\quad \gamma = \frac{1-\dfrac{\alpha\cdot\Delta}{2}}{1+\dfrac{\alpha\cdot\Delta}{2}}\quad ;\quad \alpha = 0.045$$

The open loop transfer function is now:

$$K_{VQ}\cdot K_{TV}\cdot P(z)\cdot C(z) = K_{VQ}\cdot K_{TV}\cdot(\frac{1}{U\cdot A}\cdot\frac{1-\beta}{z-\beta})\cdot K_P'\cdot(\frac{z-\gamma}{z-1})$$

Closed loop system denominator is now a quadratic polynomial. The stability margins are obtained using a Bode plot. The open loop gain and phase are shown in Figure 9.0. With the high sample rate and slow response time results are similar to the continuous case discussed in PID Controller Design Part 1, with the gain crossover frequency at ~0.0166 Hz, a PM~66.5°. The main difference with the continuous case is the phase response now falls off as frequency approaches the sampling frequency of 100 Hz. Again, care must be taken in an actual design to ensure the higher bandwidth does not put too much demand on the furnace drive.

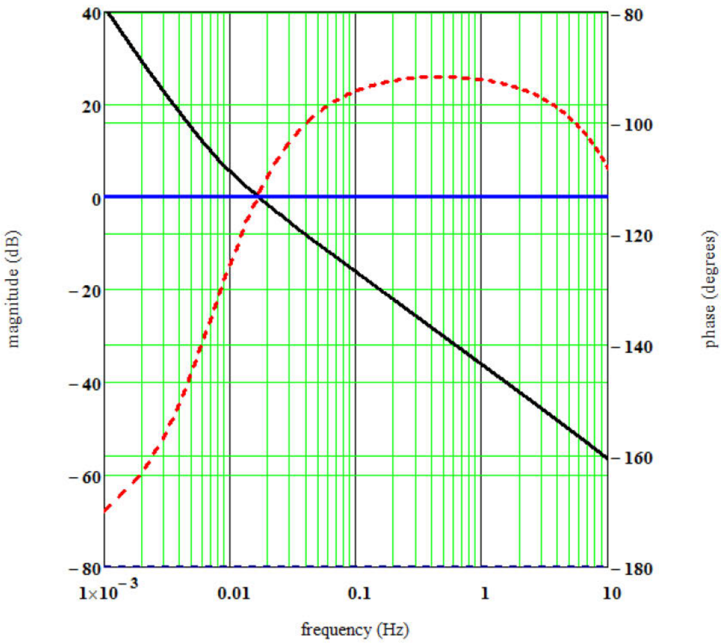


Figure 9.0 Bode plot for Example with PI controller

The expression for the error for this compensator is now:

$$T_S(z)-T_C(z)=\frac{1}{1+K_{VQ}\cdot K_{TV}\cdot\frac{1}{U\cdot A}\cdot\frac{1-\beta}{z-\beta}\cdot K'_P\cdot\frac{z-\gamma}{z-1}}\cdot\left[-U\cdot A\cdot PT_e(z)+T_S(z)\right]$$

$$T_S(z)-T_C(z)=\frac{(z-1)\cdot(z-\beta)}{z^2+b_1\cdot z+b_0}\cdot\left[-U\cdot A\cdot PT_e(z)+T_S(z)\right]$$

$$where\quad b_1=-\left(1+\beta-\frac{K_{VQ}\cdot K_{TV}\cdot K'_P\cdot(1-\beta)}{U\cdot A}\right)\quad;\quad b_0=\left(\beta-\frac{\gamma\cdot K_{VQ}\cdot K_{TV}\cdot K'_P\cdot(1-\beta)}{U\cdot A}\right)$$

As this is a Type 1 system so there will be no steady state error to a step change δT_e and δT_S as shown below:

$$E_{SS}=Lim_{z\to 1}\left\{(z-1)\cdot\left(\frac{(z-1)\cdot(z-\beta)}{z^2+b_1\cdot z+b_0}\cdot\left[-U\cdot A\cdot\frac{\delta T_e}{U\cdot A}\frac{z\cdot(1-\beta)}{(z-1)\cdot(z-\beta)}+\frac{\delta T_S\cdot z}{z-1}\right]\right)\right\}=0$$

For a ramp change to the external temperature:

$$\delta PT_e(z)=\frac{\delta\dot{T}_e}{U\cdot A}\frac{z\cdot(a_1\cdot z+a_0)}{(z-1)^2\cdot(z-\beta)}$$

$$where\quad a_1=\Delta-(1-\beta)\cdot\tau;\quad a_0=\Delta\cdot\beta-(1-\beta)\cdot\tau$$

The steady state error is:

$$E_{SS} = Lim_{z\to 1}\left\{(z-1)\cdot\left(\frac{(z-1)\cdot(z-\beta)}{z^2+b_1\cdot z+b_0}\cdot\left[-U\cdot A\cdot\frac{\delta\dot{T}_e}{U\cdot A}\frac{z\cdot(a_1\cdot z+a_0)}{(z-1)^2\cdot(z-\beta)}+\frac{\Delta\cdot\delta\dot{T}_S\cdot z}{(z-1)^2}\right]\right)\right\}$$

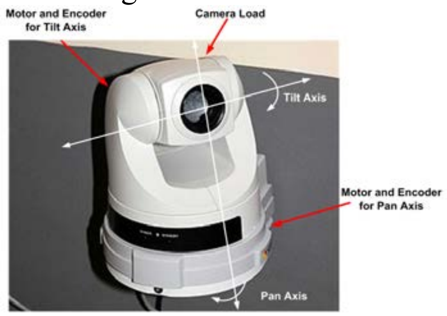
$$E_{SS} = Lim_{z\to 1}\left\{\left(\frac{1}{z^2+b_1\cdot z+b_0}\cdot\left[-z\cdot(a_1\cdot z+a_0)\cdot\delta\dot{T}_e+\Delta\cdot z\cdot(z-\beta)\cdot\delta\dot{T}_S\right]\right)\right\}$$

$$E_{SS} = \frac{U\cdot A}{K_{VQ}\cdot K_{TV}\cdot K_P'\cdot(1-\gamma)\cdot(1-\beta)}\cdot\left[-\Delta\cdot(1-\beta)\cdot\delta\dot{T}_e+\Delta\cdot(1-\beta)\cdot\delta\dot{T}_S\right]$$

$$E_{SS} = \frac{\Delta\cdot U\cdot A}{K_{VQ}\cdot K_{TV}\cdot K_P'\cdot(1-\gamma)}\cdot(-\delta\dot{T}_e+\delta\dot{T}_S) \quad ; \quad \frac{\Delta\cdot U\cdot A}{K_{VQ}\cdot K_{TV}\cdot K_P'\cdot(1-\gamma)}\approx 0.05$$

The set point will normally not change as a ramp, however, the temperature of the environment can so that most likely the steady state error for this case would be $E_{SS} = 0.05\cdot\delta\dot{T}_e$.

<u>5.0 Motion Control Feedback System:</u> Position and rate feedback control systems are used in many applications: cameras for surveillance and in the movie industry, communication and radar antennas, computer disk control, pointing lasers, etc. For this application, the plant, usually termed the load, is mounted on a linear or rotary stage driven by a motor. A servo amplifier converts control commands to a motor drive signal, for example a control voltage to a motor drive current applied to the motor stator windings. This generates a torque causing the load mounted on the rotary stage to rotate. A typical pan-tilt zoom (PTZ) camera assembly, with the basic position control loop elements loop is shown in Figure 10.0.


Motor and Encoder for Tilt Axis
Camera Load
Tilt Axis
Motor and Encoder for Pan Axis
Pan Axis

https://commons.wikimedia.org/wiki/File:Axis_214_PTZ_Camera.jpg

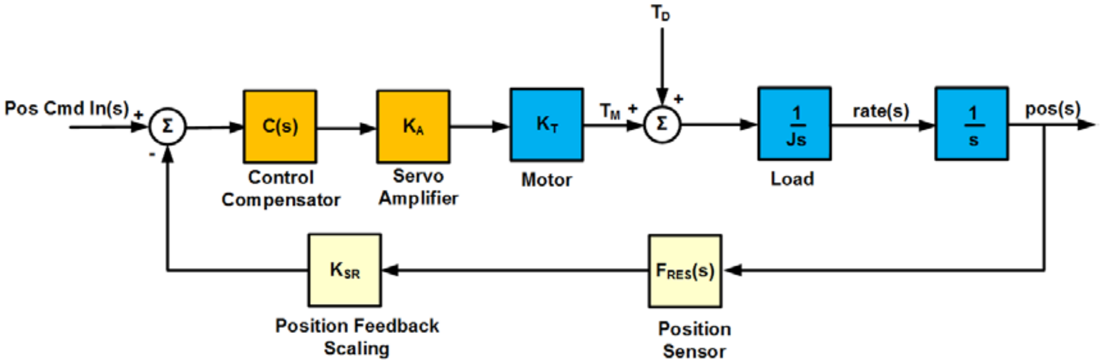


Figure 10.0 PTZ Camera Assembly and Basic Block Diagram for a Motion Control Feedback

The position command input is compared to the measured position, calculating the position error. The control compensator uses the error to generate a drive command to the motor—which is applied through the servo amplifier. The output of the motor is a torque that is applied to the load along with any disturbances. As shown, the load is simply modeled as pure inertia neglecting the effects of friction. The position feedback sensor and scaling constant work in tandem to generate a measured signal in the units used by the servo loop; normally the net gain is unity. This example uses a DC servo motor with a current feedback servo amplifier. The relationship between the control signal and the motor torque reduces to the product of two constants; the servo amp scaling K_A (A/V), and the motor torque constant K_T (in/lb/A) as shown in the figure.

6.1 Continuous Time Motion Control Example: The constants and control compensator transfer function are given as:

- K_T=0.1 in-lb/amp
- K_A=0.5 amp/volt
- C(s)=2*pi*40*K_S*[(2*pi*160*)/(s+2*pi*160)]*[(s+2*pi*16)] volt/deg
- P(s)=1/(J*s^2) ; J=0.0001 in-lb-sec2
- K_{SR} *F_{RES}(s) = 1 (deg/deg) per assumption of ideal sensor
- K_S=J/(K_A*K_T)

The open loop transfer function is given by:

$$OLTF(s) = \frac{K_A \cdot K_T}{J \cdot s^2} \frac{K_S \cdot 2 \cdot \pi \cdot 40 \cdot 2 \cdot \pi \cdot 160 \cdot (s + 2 \cdot \pi \cdot 16)}{(s + 2 \cdot \pi \cdot 160)}$$

$$= \frac{2 \cdot \pi \cdot 40 \cdot (s + 2 \cdot \pi \cdot 16)}{s^2} \cdot \frac{2 \cdot \pi \cdot 160}{s + 2 \cdot \pi \cdot 160}$$

A PD compensator is used with a zero at 16 Hz to shape the loop and a roll off filter at 160 Hz for noise suppression. There are already two integrators in the loop due to the physical dynamics of converting acceleration to position; which also means this is a Type 2 servo loop. It can be seen from the Bode plot in Figure 11.0 that the gain cross over frequency is at ~40 Hz. The phase margin is about 54° with an infinite gain margin. Closed loop response, shown by the solid green line, has unity gain to ~40 Hz and some minor closed loop peaking. The -3 dB BW is 60 Hz. There are two scaling constants, one in the forward path and the other in the feedback path. The scaling constant in the feedback path provides for unity gain feedback or K_{SR}*F_{RES}=1. The scaling constant in the forward path cancels the motor drive and plant scaling parameters, and allows the control compensator to effectively set the loop bandwidth in the analysis.

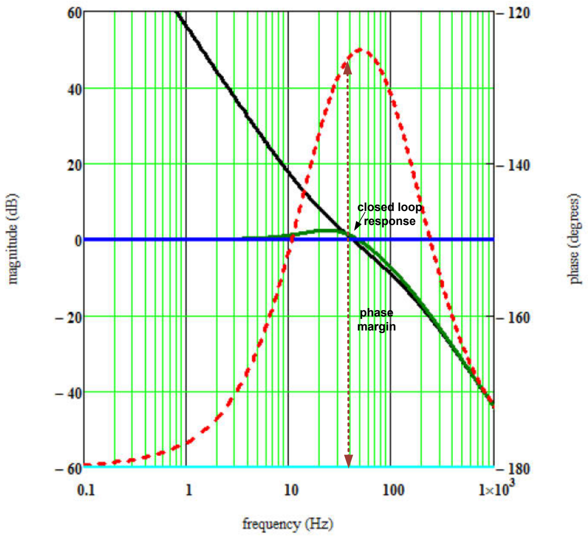


Figure 11.0 Bode plot for Motion Control Feedback Loop Example

The compliance plot of a disturbance acceleration (or torque as A=T/J) input to the control loop acceleration output is shown in Figure 12 and effectively provides -16 dB of rejection at 10 Hz.

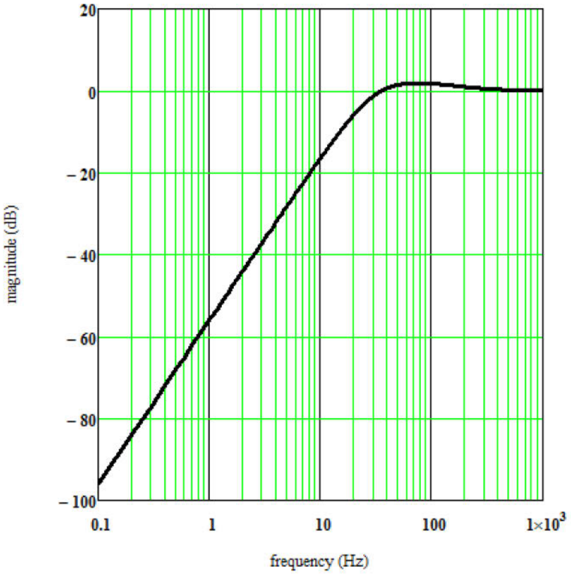


Figure 12. Torque Compliance Plot for Motion Control Example

<u>5.2 Digital PID Motion Controller:</u> A digital controller is designed based upon the continuous analog design—converting the analog controller to a sample data configuration, as discussed in section 3.2.2. The digital controller is implemented in a processor, this configuration illustrated in Figure 13.0. The block diagram is modified to include sample functions which are analog to digital converters (ADC) or digital to analog converters (DAC) that sample signals or generate commands

at a specified Δ time interval. Mathematically, these devices are modeled as zero order hold (ZOH) functions, described in section 3.2.1, which sample an input at a specified time and hold the value until the next sample. The ZOH is applied in the forward path often to the plant although a disturbance, akin to the external temperature in the previous example, is normally not in the sampled path.

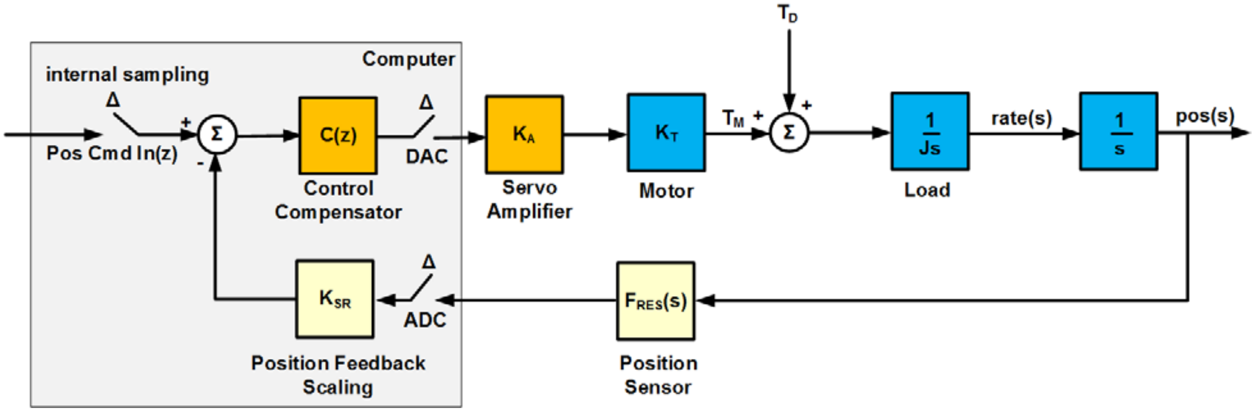


Figure 13.0 Position Control Configuration with a Digital Controller

A digital implementation using the bilinear transformation with frequency pre-warping provides the transfer functions frequency response in a sample data domain. For example, the compensator C(s) was given by:

$$C(s) = K_S \cdot \frac{2\cdot\pi\cdot 40\cdot 2\cdot\pi\cdot 160\cdot(s+2\cdot\pi\cdot 16)}{(s+2\cdot\pi\cdot 160)}$$

The sampling frequency is assumed to be 4000 Hz (ΔT=0.00025 sec). There are three critical frequencies, which when pre-warped are:

$$2\cdot\pi\cdot 40 \Rightarrow \quad \omega_{S1} = 2\cdot\pi\cdot 39.986$$
$$2\cdot\pi\cdot 16 \Rightarrow \quad \omega_{S2} = 2\cdot\pi\cdot 15.999$$
$$2\cdot\pi\cdot 160 \Rightarrow \quad \omega_{S3} = 2\cdot\pi\cdot 159.165$$

Because of the high sample rate relative to the critical frequencies, none really change significantly from their continuous frequency equivalent. In the z-domain the compensator is expressed as:

$$C(z) = C(s)\,|_{s=\frac{2}{\Delta}\frac{z-1}{z+1}} = K_0 \cdot \left[\frac{z-\alpha}{z-\beta}\right]$$

$$with \quad K_0 = K_S\cdot\omega_{S1}\cdot\omega_{S3}\cdot\left[\frac{1+\frac{\Delta\cdot\omega_{S2}}{2}}{1+\frac{\Delta\cdot\omega_{S3}}{2}}\right] \quad ; \quad \alpha = \frac{1-\frac{\Delta\cdot\omega_{S2}}{2}}{1+\frac{\Delta\cdot\omega_{S2}}{2}} \quad ; \quad \beta = \frac{1-\frac{\Delta\cdot\omega_{S3}}{2}}{1+\frac{\Delta\cdot\omega_{S3}}{2}}$$

For sample-data representation can be expressed as:

$$C(z^{-1}) = K_0 \frac{1-\alpha\cdot z^{-1}}{1-\beta\cdot z^{-1}}$$

Finally, the digital implementation of the controller is:

$$u(k) = \alpha\cdot u(k-1) + K_0\cdot(e(k) - \alpha\cdot e(k-1))$$

The Bode plot for the sampled system is shown in Figure 14.0 and can be compared to Figure 11.0 for the continuous time version. Due to the sampling, the phase margin is reduced to 51° while the gain margin is 28 dB; a decrease in both effective stability margins.

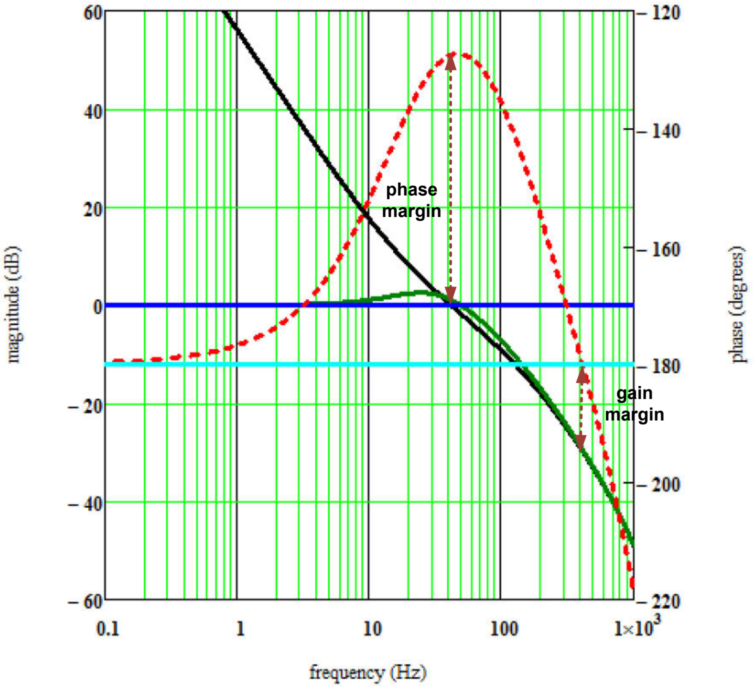


Figure 14.0 Bode plot for Sample Data Version of Motion Control Example

<u>6.0 Advanced PID Control Techniques:</u> The integration of a PID control structure with more complex control methods can sometimes lead to improved performance. However, the degree of complexity is not always justified by the performance improvement so should be evaluated well in a tradeoff study. Gain scheduling is one of the simpler forms of control algorithm adaptation. Gains can be scheduled as a function of operating mode, servo error magnitude, convergence, or convergence rate. The approach is very useful with systems that are non-linear or with operating parameters or conditions that vary with time. Process measurements are used to determine the process state and then the controller is adjusted based upon predetermined control parameters that provide the best performance. Processor performance taking major leaps forward now allows the use of mathematical techniques requiring more processing than previously possible. For example, for a navigation and tracking application, small MEMS inertial navigation systems slightly larger than a quarter come complete with an Extended Kalman Filter to estimate inertial position and

rates. This purpose of this section is to take a very brief look at integrating the PID with two more advanced techniques; adaptive control and fuzzy logic control.

6.1 Adaptive PID Controller: Adaptive controllers were designed to account for changes in the dynamics of a plant, process and/or disturbances. For example, with a precision pointing system, as described previously, variations in friction and cable restraint with temperature can be problematic. A controller that adapts it parameters to account for these changes will improve performance. Adaptive control technology has been evolving since the late 1950s and early 1960s. An overview of some early adaptive control methods can be found in [4]. Adaptive controllers are related to gradient optimization techniques. With linear as well as non-linear systems, gradient optimization algorithms can be used to adjust process control variables based upon their effect on a control loop performance metric or index of performance (IOP); maximizing or minimizing the IOP depending on the performance definition. A simple example of an IOP is the servo or modeling error squared defined as:

$$J(e) = \frac{e(t)^2}{2}$$

This function is minimized by the finding the value of the control parameter, K that drives the error squared to zero. As this is a positive function, if its derivative is negative the objective should be met. A simple approach is to differentiate the function as:

$$\frac{dJ(e)}{dt} = \frac{dJ(e)}{de} \cdot \frac{de}{dt} = e(t) \cdot \dot{e}(t) = e(t) \cdot \frac{de}{dK} \cdot \frac{dK}{dt}$$

Choosing the derivative of the control parameter as:

$$\frac{dK}{dt} = -\lambda \cdot e(t) \cdot \frac{de}{dK}$$

Provides the desired result, or:

$$\frac{dJ(e)}{dt} = -\lambda \cdot e(t)^2 \cdot (\frac{de}{dK})^2 \leq 0$$

The derivative is proportional to the gradient of the error with respect to the control parameter. The parameter, lambda, is used to adjust the convergence rate of the adaptation equation. It will be shown that here is much similarity between this parameter update algorithm those used in present adaptive controllers.

Adaptive controllers are often categorized as either indirect or direct methods. With the indirect approach, controller parameters are obtained by first identifying the parameters of a plant or process model and then determining the controller’s coefficients as one would in a non-adaptive case. A block diagram of the indirect approach is shown in Figure 15.0. This approach is more amenable for use with other conventional algorithms, such as a Smith Predictor if the system has long delay, since the conventional architecture is maintained.

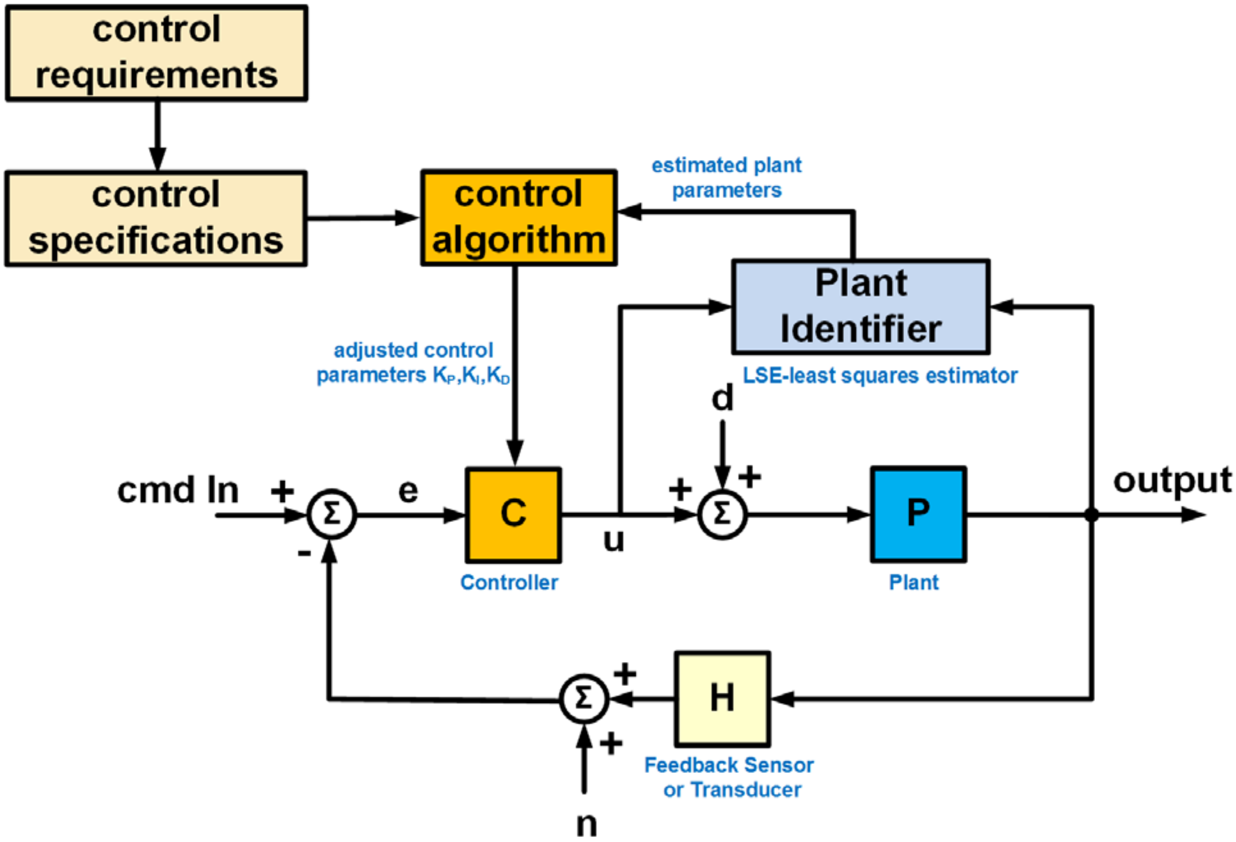

control requirements
control specifications
control algorithm
estimated plant parameters
Plant Identifier
LSE-least squares estimator
adjusted control parameters K_P,K_I,K_D
cmd In
+
-
Σ
e
C
Controller
u
d
+
+
Σ
P
Plant
output
H
Feedback Sensor or Transducer
Σ
+
+
n

Figure 15.0 Indirect Adaptive Control Architecture

The plant model parameters are estimated based upon an equation error derived from the assumed plant transfer function model using some type of least squares or gradient parameter estimation algorithm. The estimated plant parameters are used to update the control parameters. Convergence of the equation error is required for parameter convergence. An excellent reference for reviewing identification techniques is [5].

With a direct approach, controller parameters are adjusted directly to obtain the same response as that of a reference model; a control architecture termed model reference adaptive control (MRAC). It uses gradient type update equation based upon measured data and a model error. The convergence of the model error to zero guarantees model following performance. The model reference adaptive control (MRAC) architecture is often used for this method. Reference [6] includes many papers leading to the present state of the art, now described in texts on the subject, including a seminal paper by Dr. R.V. Monopoli. A block diagram of the direct adaptive architecture is shown in Figure 16.0.

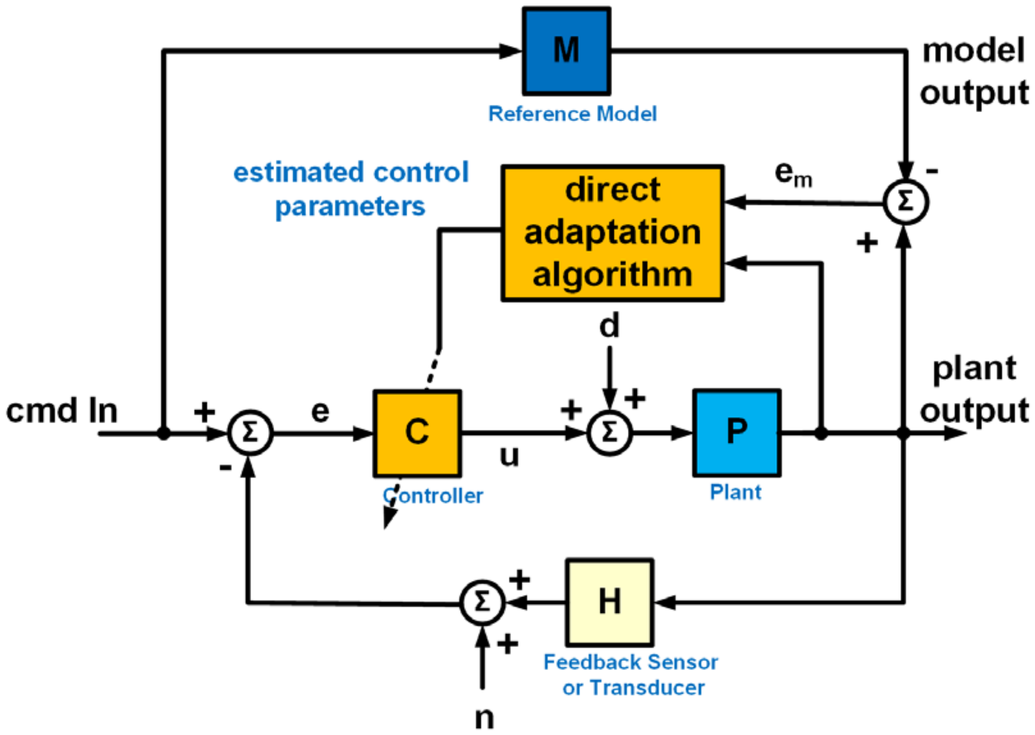


Figure 16.0 Direct Adaptive Control Architecture

Substantial theory has been developed supporting algorithm implementation in recent years. However, it still remains one of those techniques where the projected improvement in performance must be weighed heavily against the increase in design complexity. As stability and algorithm convergence are critical, much research has focused on this topic. Numerous papers and books have been published establishing conditions under which an algorithm implementation is stable and will converge. Mathematically, these algorithms are fairly complex in themselves, and conditions establishing stability and convergence even more so. In addition, as ideal conditions are seldom the case with an actual design, algorithms must be bounded and monitored for instability in order to readjust or reset for continued convergence in the real environment. Key issues to consider with adaptive control are:

- Sufficient excitation is a key requirement; process dynamics can be completely identified or controlled only if all of its dynamic modes (roots of the characteristic equation, dominant frequencies, eigenvalues, etc.) within the bandwidth of the system design are excited; otherwise those not accounted for will be uncompensated by the model.
- Convergence and stability of the adaptation mechanism is the key to the adaptation process. Conditions for stability and convergence, besides being difficult to obtain for a real system operating environment, are also mathematically complex to derive and interpret. Often

supervision or monitoring of the adaptation process is necessary to adjust for divergence or stagnation of the algorithm.

- Relative to an actual design, even the adaptation procedure tends to be fairly complex mathematically, and any adjustment algorithms required to supervise convergence or stability only exacerbates this.
- The limitations of the PID controller relative to plant complexity and model matching constraints discussed previously still exist with the adaptive technique.

An example of a model reference adaptive control (MRAC) application, specific to a PID control structure using a direct adaptive control architecture is provided. The model is chosen to meet the system control objectives. The example provides only a notional design process for a simple plant; issues with higher order plants that lead to further complexity are not addressed. First or second order plants are assumed and a model one order greater than the plant (i.e. plant denominator second order -> model denominator third order). The example describes the basic design concepts, mathematical theory and conditions for stability and convergence, and steps in the design process. The purpose is to provide an overview so that details can be explored further in an actual design. The PID MRAC design is based on the model matching conditions, discussed in Part 1 and section 3.0, for a desired CLTF model T(s) and a given plant defined as:

$$\frac{y_m(s)}{r(s)} = \mathrm{M}(s) = \frac{N_M(s)}{D_M(s)}\ \{\text{model}\} \quad ; \quad \frac{y(s)}{u(s)} = P(s) = \frac{N_P(s)}{D_P(s)}\ \{\text{plant}\}$$

$$\therefore\ N_M(s)\cdot r(s) = D_M(s)\cdot y_m(s) \qquad \therefore\ N_P(s)\cdot u(s) = D_P(s)\cdot y(s)$$

A compensator $C_{PID0}(s)$ is required for the nominal closed feedback loop to match the plant, or:

$$\frac{N_M(s)}{D_M(s)} = \frac{C_{PID0}(s)\cdot P(s)}{1+C_{PID0}(s)\cdot P(s)}$$

The control input u(s) provides the plant control signal consistent with the desired matching condition based upon the compensator $C_{PID0}(s)$. If the plant is unknown, the compensator $C_{PID0}(s)$ cannot be determined directly. The compensator is estimated by $C_{PID}(s)$ which is a combination of $C_{PID0}(s)$ and a transfer function, $\delta C_{PID}(s)$, containing the residual parameter errors that define $C_{PID0}(s)$ or:

$$C_{PID}(s) = C_{PID0}(s) + \delta C_{PID}(s)$$

For this application, it is more efficient algebraically to define the PID relative to a common denominator ‘s’ due to the integrator, or:

$$C_{PID}(s) = \frac{C(s)}{s} = \frac{C_0(s)}{s} + \frac{\delta C(s)}{s}$$

The adaptive controller is designed by making the parameters of $C_{PID}(s)$ adjustable (i.e. time varying) such that they reduce the error between the actual response and the ideal response as defined by an equation error. The algorithm is outlined in Table 4a-4f. A block diagram of the control algorithm is shown in Figure 17.0. Table 4a defines model, plant and controller definitions in functions 1-7. The controller is implemented in a forward path configuration consistent with a standard PID architecture. The controller will provide the desired matching relationship between the control loop and model response if chosen with parameters defined by C_{PID0} as described previously, however these parameters are unknown. The sign of the plant input parameter b_0 is known.

Table 4a Model, Plant, and Controller Definitions for Adaptive PID Design

	Function or Definition	Equation
	Definition of Plant and Model Response	
1	Plant Output	$D_P(s)\cdot y(s) = b_0 \cdot u(s)$
2	Servo error	$e(s) = y(s) - r(s)$
3	Model Output	$D_M(s)\cdot y_m(s) = N_M(s)\cdot r(s)$
4	Control Signal as Implemented	$u(s) = \frac{1}{\hat{b}_{0.}}\cdot\left(\frac{C(s)}{s}\cdot e(s) + \frac{\Delta N_M(s)}{s}\cdot r(s)\right)$
5	Model numerator N_m polynomial	$N_M(s) = C(s) + \Delta N_M(s)$
6	Convert error to (r-y) and factor out 1/s	$u(s) = \frac{1}{s}\cdot\frac{1}{\hat{b}_{0.}}\cdot\left(-C(s)\cdot y(s) + N_M(s)\cdot r(s)\right)$
7	Expand Controller C(s)	$C(s) = K_D\cdot s^2 + K_P\cdot s + K_I = C_0(s) + \delta C(s)$

In Table 4b filtered states are defined in 8-11. The filter 1/F(s) serves its standard purpose, but also chosen such that the F(s) polynomial order is one less than the model denominator polynomial such that the transfer function between the model and equation error, to be defined, is strictly positive real (SPR); or simply put, very stable. The SPR theory is critical to algorithm convergence requirements. When the transfer function is expressed as a vector differential equation, stability theory requires the SPR condition to obtain matrix relationships that are negative definite as required for convergence. In simple terms, for a convergence condition akin to that needed for the simple IOP example discussed earlier. The SPR transfer function is closely related to the concept of a passive system; one that continuously dissipates the initial energy stored without generating its own energy thereby always settling to an equilibrium point. More generally, a I/O map is passive if, on average, increasing the output *y* requires increasing the input *u*; mathematically expressed as integral of their products being > 0 (i.e. both same sign). A passive system results if a system transfer function G(s):

- is stable; no poles in the closed RHP
- Re(G(s) > 0 for |ω|->∞
- G(s) is real for real s; satisfied by any physical system
- Lim{ ω^2G(ω)}>0 as ω->∞

A PID controller is passive because the control signal (the output) moves in the same direction as the error signal (the input). But a PID controller with delay is not passive, because the control signal can move in the opposite direction from the error, a potential cause of instability.

Table 4b Filtered Model, Plant, and Controller

	Function or Definition	Equation
Filtered States and control		
8	Filtered State	$y_f(s)=\dfrac{y(s)}{F(s)} \quad ; \quad r_f(s)=\dfrac{r(s)}{F(s)}$
9	Filtered Plant Output	$D_P(s)\cdot y_f(s)=b_0\cdot u_f(s)$
10	Filtered model output	$D_M(s)\cdot y_{mf}(s)=N_M(s)\cdot r_f(s)$
11	Filtered Control	$u_f(s)=\dfrac{1}{s}\cdot\dfrac{1}{\hat{b}_0}\cdot\left(-C(s)\cdot y_f(s)+N_M(s)\cdot r_f(s)\right)$

The controller design for model matching is expanded in 12-17 of table 4C by substituting the controller equation from 11. The PID feedback and plant polynomials are combined to generate the model denominator polynomial, $D_M(s)$, and residual error polynomials as described by 17.

Table 4c Control Design for Model Matching

	Function or Definition	Equation
Control Design for Model Matching		
12	Substitute 11 -> 9	$D_P(s)\cdot y_f(s)=\dfrac{1}{s}\cdot\dfrac{b_0}{\hat{b}_0}\cdot\left(-C(s)\cdot y_f(s)+N_M(s)\cdot r_f(s)\right)$
13	Expand gain b_0	$\dfrac{b_0}{\hat{b}_0}=1-\dfrac{\delta b_0}{b_0}$
14	Define w(s)	$w(s)=-C(s)\cdot y_f(s)+N_M(s)\cdot r_f(s)$
15	Substitute 14, 13 -> 12	$D_P(s)\cdot y_f(s)=\dfrac{1}{s}\cdot\left(-C(s)\cdot y_f(s)+N_M(s)\cdot r_f(s)\right)-\dfrac{1}{s}\cdot\dfrac{\delta b_0}{b_0}\cdot w(s)$
16	Multiply by s and define $D_M(s)$	$D_M(s)=s\cdot D_P(s)+C_0(s)$
17	Collect terms, substitute 16->15	$D_M(s)\cdot y_f(s)=N_M(s)\cdot r_f(s)-\delta C(s)\cdot y_f(s)-\dfrac{\delta b_0}{b_0}\cdot w(s)$

The model and equation errors are defined in 18-26. The model error is the difference between the plant output (plant output -> y) and the model output (model output -> y_m). The equation error is the sum of the residual parameter error terms as listed in step 20. Step 21 indicates the filtered plant response is driven by the equation error. The equation error residual parameter errors will be time varying when the controller coefficients are adaptive. The general expression for the equation error with time dependent parameters errors with the adaptive system is defined in steps 21-23.

The plant output is now given by the equation in step 24. The equation for the modeling error is obtained by subtracting the equation for the model output in step 3. It can be seen in 25 that it is driven by the equation error. The final expression, 26, is a conversion of the polynomial to a vector state equation. This is where the filter plays a key role by making the transfer function SPR.

Table 4d Model and Equation Errors

	Function or Definition	Equation
Model and Equation Errors		
18	Define modeling error	$e_m(s) = y(s) - y_m(s)$
19	From 17 define equation error	$\varepsilon(s) = -\left(\delta C(s)\cdot y_f(s) + \frac{\delta b_0}{b_0}\cdot w(s)\right)$ $= D_M(s)\cdot y_f(s) - N_M(s)\cdot r_f(s)$
20	Combine terms in 17	$D_M(s)\cdot y_f(s) = N_M(s)\cdot r_f(s) + \varepsilon(s)$
21	Define state vector	$\hat{X}(t) = \left[\ddot{y}_f(t) \quad \dot{y}_f(t) \quad y_f(t) \quad w(t)\right]^T$
22	Define time varying parameter vectors	$\hat{P}(t) = \hat{P}_0 + \delta\hat{P}(t) \quad ; \quad \hat{P}_0 = \left[K_{D0} \quad K_{P0} \quad K_{I0} \quad \frac{1}{b_0}\right]^T$ $\delta\hat{P}(t) = \left[\delta K_D(t) \quad \delta K_P(t) \quad \delta K_I(t) \quad \frac{\delta b_0(t}{b_0}\right]^T$ $\hat{P}(t) = \left[K_D(t) \quad K_P(t) \quad K_I(t) \quad \frac{1}{b_0(t)}\right]^T$
23	Equation error in time domain	$\varepsilon(t) = X(t)^T \cdot \delta P(t)$
24	Multiply 20 by F/D$_m$	$y(s) = \frac{N_M(s)}{D_M(s)}\cdot r(s) + \frac{F(s)}{D_M(s)}\cdot\varepsilon(s)$
25	Determine model error; subtract model output 3 from 24	$e_m(s) = \frac{F(s)}{D_M(s)}\cdot\varepsilon(s)$
26	Convert 25 to time domain vector equation	$\dot{\hat{E}}(t) = \overline{A}_M \cdot \hat{E}(t) + \hat{B}\cdot\varepsilon(t) \quad ; \quad e_m(t) = \hat{D}^T \cdot \hat{E}(t)$

Table 4e contains the final steps, 27-33, defining a system IOP conditions for stability and/or convergence of the adaptation algorithm. The IOP, generally defined as a Lyapunov function, is a positive function of the model and equation errors with symmetric positive definite gain matrices: Γ, Λ, and Q and the matrix transfer function between model and equation errors, D*(sI-A$_M$)$^{-1}$*B, SPR. Lyapunov stability theory defines the criteria for the convergence under SPR conditions. For this simple description, viewing it as positive function can suffice, so that a negative derivative at least indicates stability. The IOP derivative is calculated in 28 and the error vector from 26 substituted to obtain the form in 29. The SPR theory now comes into play stating that if given the model error transfer function is SPR then the matrix sum is equivalent to the negative of Q:

$$\overline{A}_M{}^T \cdot \overline{\Gamma} + \overline{\Gamma}\cdot\overline{A}_M = -Q \quad and \quad \overline{\Gamma}\cdot\hat{B} = \hat{D}$$

The second equality can be used obtain solely the model error from the error vector—otherwise several model error derivatives are required. The derivative in 29 contains all terms which need to be compensated. Choosing the parameter update equation in 31 results in the derivative of the positive function being negative or zero. The model error is sufficient to obtain the desired theoretical result; including equation error, may help an actual implementation. Differentiation, via a HPF, is required to calculate the equation error and its derivative. The advantage of including these terms, if any, needs to be weighed against the associated noise and processing error. As presented, the result 33 would indicate the filtered error would converge and the parameter errors bounded. The equation error should also converge, although as the model error is derived from the equation error via a passive system, this should follow.

Table 4e Performance Metric and Parameter Adaptation Equations

	Function or Definition	Equation
Performance Metric and Parameter Update Adaptation Equations		
27	Define IOP	$V(t)=0.5\cdot\hat{E}(t)\cdot\overline{\Gamma}\cdot\hat{E}(t)+0.5\cdot\varepsilon(t)^2+0.5\cdot\delta\hat{P}(t)^T\cdot\overline{\Lambda}\cdot\delta\hat{P}(t)>0$
28	Differentiate	$\dot{V}(t)=\dot{\hat{E}}(t)^T\cdot\overline{\Gamma}\cdot E(t)+\hat{E}(t)^T\cdot\Gamma\cdot\dot{\hat{E}}(t)+\varepsilon(t)\cdot\dot{\varepsilon}(t)+\delta\hat{P}(t)^T\cdot\overline{\Lambda}\cdot\delta\dot{\hat{P}}(t)$
29	Substitute 26 for $\dot{\hat{E}}(t)$	$\dot{V}(t)=\hat{E}(t)^T\cdot\left(\overline{A}_M{}^T\cdot\overline{\Gamma}+\overline{\Gamma}\cdot\overline{A}_M\right)\cdot\hat{E}(t)+\hat{B}^T\cdot\overline{\Gamma}\cdot\hat{E}(t)\cdot\varepsilon(t)$ $+\varepsilon(t)\cdot\dot{\varepsilon}(t)+\delta\hat{P}(t)^T\cdot\overline{\Lambda}\cdot\delta\dot{\hat{P}}(t)$
30	Define error	$e_m(t)=\hat{B}^T\cdot\overline{\Gamma}\cdot\hat{E}(t)=\hat{D}^T\cdot\hat{E}(t)$
31	Define parameter vector update equation	$\delta\dot{\hat{P}}(t)=-\overline{\Lambda}^{-1}\cdot\hat{X}(t)\cdot(e_m(t)+\dot{\varepsilon}(t)+\lambda\cdot\varepsilon(t))$
32	Substitute 31->29	$\dot{V}(t)=-\hat{E}(t)^T\cdot Q\cdot\hat{E}(t)+e_m(t)\cdot\varepsilon(t)+\varepsilon(t)\cdot\dot{\varepsilon}(t)+$ $\delta P(t)^T\cdot\overline{\Lambda}\cdot(-\overline{\Lambda}^{-1}\cdot X(t)\cdot(e_m(t)+\dot{\varepsilon}(t)+\lambda\cdot\varepsilon(t)))$ $\dot{V}(t)=-E(t)^T\cdot Q\cdot E(t)+e_m(t)\cdot\varepsilon(t)+\varepsilon(t)\cdot\dot{\varepsilon}(t)$ $-e_m(t)\cdot\varepsilon(t)-\varepsilon(t)\cdot\dot{\varepsilon}(t)-\lambda\cdot\varepsilon(t)^2$
33	Final derivative non-positive	$\dot{V}(t)=-E(t)^T\cdot Q\cdot E(t)-\lambda\cdot\varepsilon(t)^2<0$

The algorithm is summarized in Table 4f, 34-40. In the parameter update equation for b_0, the unknown magnitude scales the parameter error however this can be considered as part of the parameter gain matrix and incorporated with λ_4 and sgn(b_0) under assumption it is known.

Table 4f Adaptation Algorithm Summary

	Function or Definition	Equation
Adaptation Algorithm Summary		
34	Model error (18)	$e_m(t) = y(t) - y_m(t)$
35	Equation error (19)	$\varepsilon(t) = X(t)^T \cdot \delta P(t)$
36	Controller (11, 14)	$u_f(t) = \cdot \int_0^t \frac{-1}{b_{0\cdot}(\tau)} \cdot w(\tau) d\tau$
37	Parameter update $K_D(t)$	$\dot{K}_D(t) = -\frac{1}{\lambda_1} \cdot \ddot{y}_f(t) \cdot (e_m(t) + \dot{\varepsilon}(t) + \lambda \cdot \varepsilon(t))$
38	Parameter update $K_P(t)$	$\dot{K}_P(t) = -\frac{1}{\lambda_2} \cdot \dot{y}_f(t) \cdot (e_m(t) + \dot{\varepsilon}(t) + \lambda \cdot \varepsilon(t))$
39	Parameter update $K_I(t)$	$\dot{K}_I(t) = -\frac{1}{\lambda_3} \cdot y_f(t) \cdot (e_m(t) + \dot{\varepsilon}(t) + \lambda \cdot \varepsilon(t))$
40	Parameter update $b_0(t)$	$\dot{b}_0(t) = -\frac{\mathrm{sgn}(b_0)}{\lambda_4} \cdot w(t) \cdot (e_m(t) + \dot{\varepsilon}(t) + \lambda \cdot \varepsilon(t))$

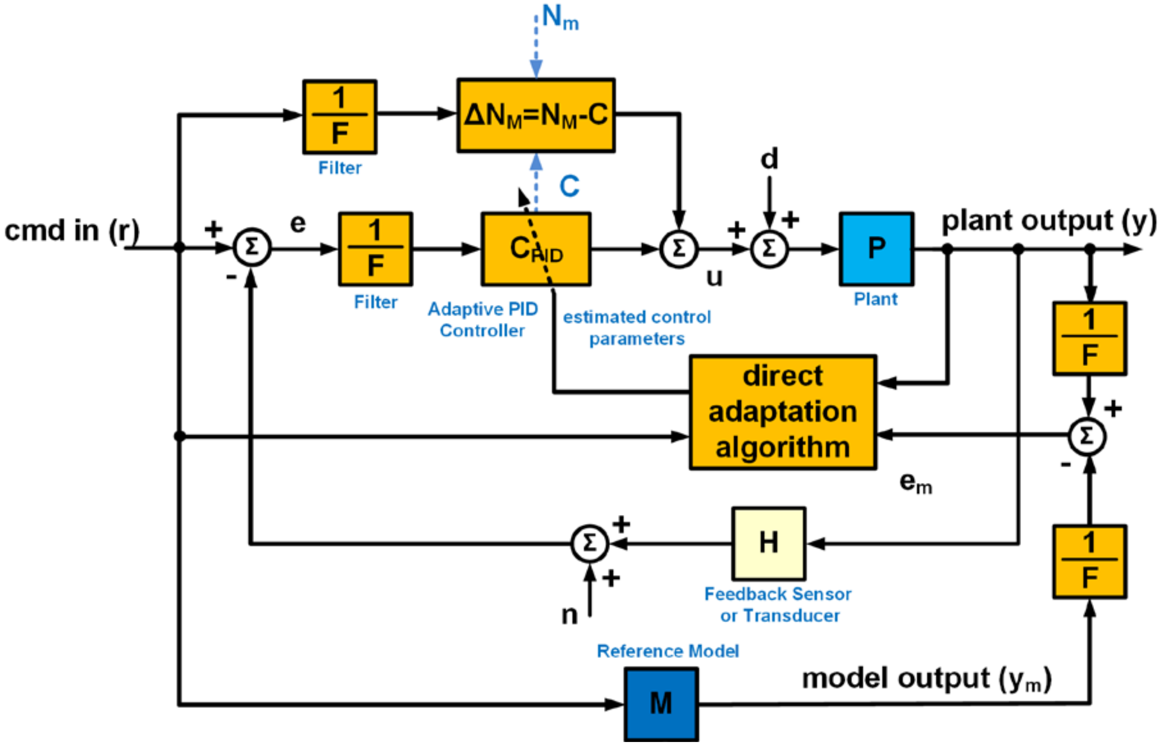


Figure 17.0 PID MRAC Block diagram

This completes the discussion for a PID MRAC; hopefully providing some insight into implementation and at least the groundwork for further study for its application and the theory behind the convergence conditions central to algorithm performance that must be addressed.

6.2 PID Fuzzy Logic Control: PID Controller performance is dependent on many component and environmental disturbances. Nominal designs should include a response margin sufficient to operate within a desired performance envelope given a specified disturbance environment. However, components wear and environments cannot always be controlled—degrading performance. As discussed in the previous section, adaptive controllers can be used but become quit complex, and convergence is not always guaranteed. Fuzzy logic is a method of weighting a set of signal variables define controller performance to create a weighted sum control signal that minimizes the loop error. The PID controller interfaces well with the FLC since it generates three outputs definitive of performance that can be directly used by the FLC logic. The interaction of the PID controller with fuzzy logic can be implemented by a wide variety of configurations; most do not require models of the plant. FLC has the ability to adjust for a changing environment, potentially maintaining better performance over a wider range of component and environmental disturbances or possibly even higher order plant dynamics than a conventional controller. The basic architecture described uses a nominal PID in tandem with the Fuzzy Logic Controller (FLC). The three PID signals are generated but not summed; being individually transmitted to the FLC. The PID design is based upon nominal design procedure used to obtain the system control objective. The FLC then tunes the loop to sustain or improve performance as operating conditions vary.

6.2.1 Fuzzy Logic Controller: The basic control loop architecture considered is shown in Figure 18.0. The loop controller C(s) includes the PID structure in tandem with a Fuzzy Logic Controller (FLC). The FLC algorithm uses measurements of the system error weighted by the control coefficients to generate the variables that drive performance which are scaled and bounded by shaping functions.

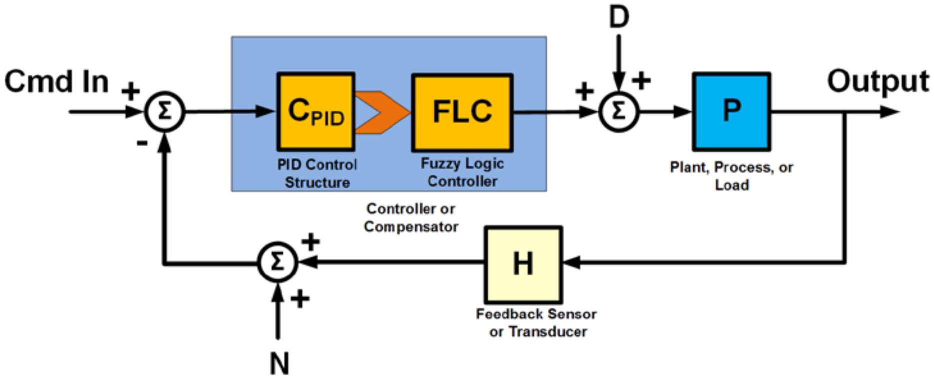
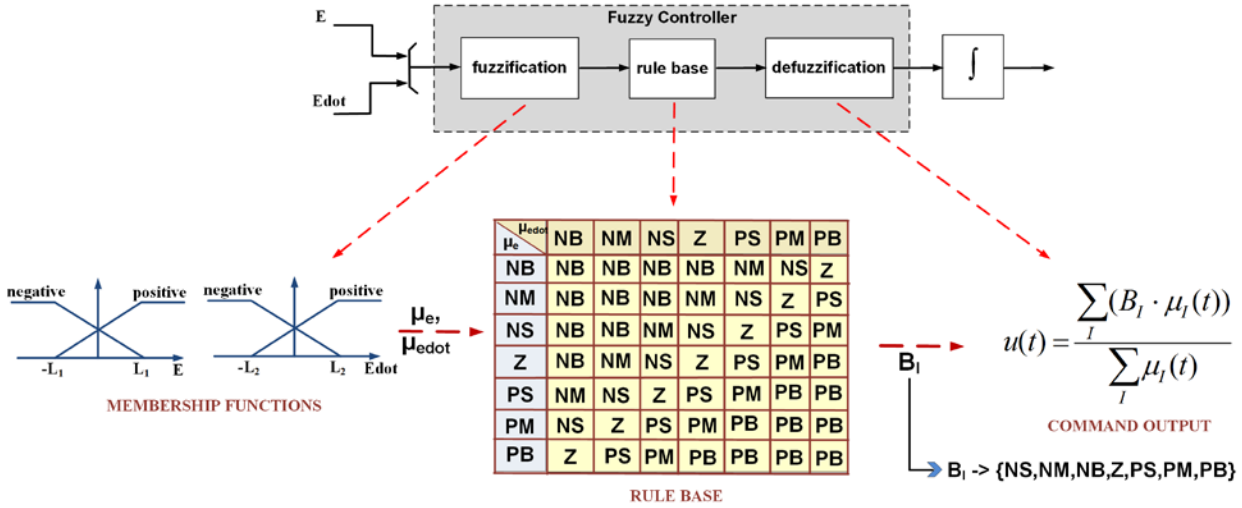


Figure 18.0 Control Loop with PID FLC Architecture

The general FLC algorithm block diagram is shown in Figure 19.0. For this simple case, measurements of the error and its derivative are the inputs to the controller, now termed crisp variables, where the initial process, fuzzification, scales and bounds the inputs or effectively shapes them using what is termed membership functions. In general, there can be as many inputs as required to define loop performance. The PID adapts well to interfacing with the FLC since if it works in a conventional loop, it effectively has the three variables necessary for control. With a full PID, the inputs can be the error, integrated error, and error derivative, or sometimes these signals differentiated as the error and its first and second derivatives. In this case, the FLC output is usually integrated. In addition, the error signals can be input strictly the errors or weighted by the nominal PID gain coefficients. There are three steps in generating a control signal; fuzzification, controller rule inference, defuzzification discussed below.



μ_e \ μ_{edot}	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	Z
NM	NB	NB	NB	NM	NS	Z	PS
NS	NB	NB	NM	NS	Z	PS	PM
Z	NB	NM	NS	Z	PS	PM	PB
PS	NM	NS	Z	PS	PM	PB	PB
PM	NS	Z	PS	PM	PB	PB	PB
PB	Z	PS	PM	PB	PB	PB	PB

$$u(t) = \frac{\sum_{l}(B_l \cdot \mu_l(t))}{\sum_{l} \mu_l(t)}$$

Figure 19.0 Functional Block Diagram of Fuzzy Controller Structure

Fuzzification: Membership functions scale and bound the measured a crisp variable to a subinterval of its actual range. A set of membership functions are defined for each variable of interest. These functions can have several scaling implementations including triangular (illustrated in Figure 19.0), trapezoidal, Gaussian, etc. often symmetrical although this is not required, that map a measured crisp variable to a scaled variable, termed fuzzified variable μ_i. Each function is centered at a value which defines a level of magnitude significant to the variable range (i.e. high, medium, low or percent of range, etc.) and often these functions are designed to overlap by 50%. With the triangular membership function, the crisp variable is linearly scaled as a fuzzified variable. So, for each input variable of interest (e, edot) there is a corresponding fuzzified value (μ_1=μ_e, μ_2=μ_{edot}) mapped by the membership functions.

Rule Base Inference: A set of rules are established that govern the ultimate state of the control output generally designed to reduce the control error. Each rule 'I' evaluates the state of the set of crisp input values as mapped by their respective membership functions to a fuzzy variable. All fuzzy variables at a point in time are evaluated relative to a specific rule base and a max or min selection rule (i.e. for n crisp variable μ_I=min [$\mu_I(X_1)$, $\mu_I(X_2)$ … $\mu_I(X_n)$]) is used to choose the fuzzy variable input to the final process. As a simple rule example is; e and edot of opposite sign implies the error is converging to zero so that no increase or even a reduced control signal is desirable. If they are of the same sign, and then a strong control signal is required to begin reducing the error. The output of the rule evaluation are weighting factors B_I quantified as NL, NM, NS, Z, PL, PM, and PS; where N denotes negative, P positive, L large, M medium, S small, and Z-zero which are mapped to numerical values based upon the control signal operating range.

<u>Defuzzification:</u> The final step in the fuzzy control process is generation of the output command. There are several implementations of this, however the easiest to understand is simply the sum of the control terms normalized to the sum of the membership function values. This implementation provides a centroid or weighted average control signal. There is control term associated with each rule that is a product of its weighting factor B_I multiplied by an associated value of the fuzzy variable, μ_I. The fuzzy control output with I rules is then obtained as:

$$u(t)=\frac{\sum_I (B_I\cdot\mu_I(t))}{\sum_I \mu_I(t)}=\sum_I B_I\cdot\sigma_I(t)\quad;\quad where\quad \sigma_I(t)=\frac{\mu_I(t)}{\sum_I \mu_I(t)}$$

This output may or may not be integrated depending on the definition of the crisp variables measured as mentioned above a more detailed block diagram of a typical control loop using this approach, similar to one implemented, MatLab Simulink, is shown in Figure 20.0. MatLab Simulink has its own algorithm implementation of the FLC. This implementation utilizes the PID structure with each individual differentiated leg of the PID providing inputs to the FLC. The output of the FLC is integrated to obtain the final control signal.

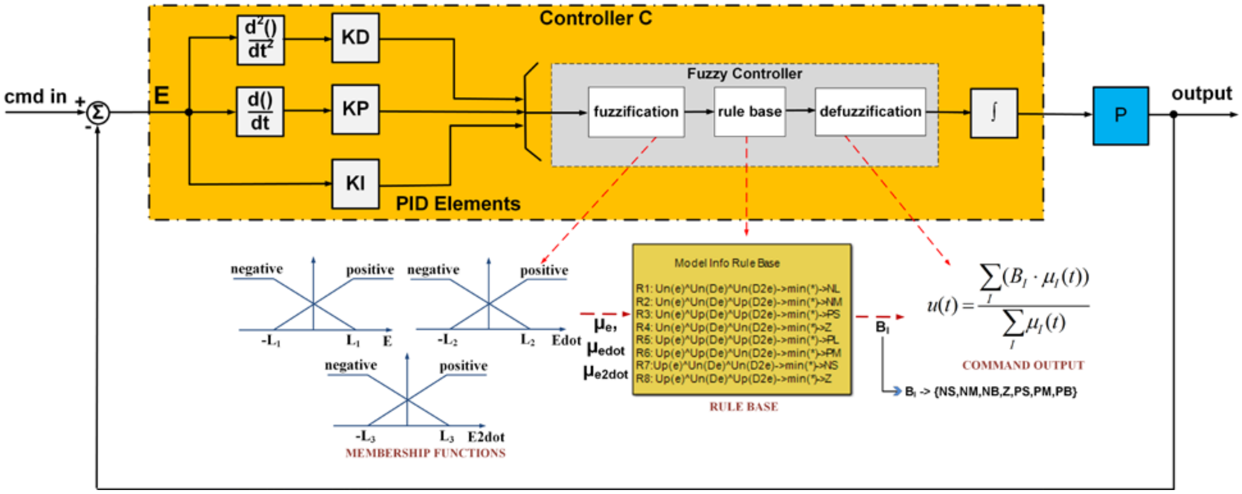

Controller C
cmd in
+
-
Σ
E
$\frac{d^2()}{dt^2}$
$\frac{d()}{dt}$
KD
KP
KI
PID Elements
Fuzzy Controller
fuzzification
rule base
defuzzification
∫
P
output
negative
positive
-L1
L1
E
negative
positive
-L2
L2
Edot
negative
positive
-L3
L3
E2dot
MEMBERSHIP FUNCTIONS
μe,
μedot
μe2dot
Model Info Rule Base
R1: Un(e)^Un(De)^Un(D2e)->min(*)->NL
R2: Un(e)^Un(De)^Up(D2e)->min(*)->NM
R3: Un(e)^Up(De)^Up(D2e)->min(*)->PS
R4: Un(e)^Up(De)^Un(D2e)->min(*)->Z
R5: Up(e)^Up(De)^Up(D2e)->min(*)->PL
R6: Up(e)^Up(De)^Un(D2e)->min(*)->PM
R7:Up(e)^Un(De)^Un(D2e)->min(*)->NS
R8: Up(e)^Un(De)^Up(D2e)->min(*)->Z
RULE BASE
BI
$u(t)=\frac{\sum_I (B_I\cdot\mu_I(t))}{\sum_I \mu_I(t)}$
COMMAND OUTPUT
BI -> {NS,NM,NB,Z,PS,PM,PB}

Figure 20.0 Control Loop using PID FLC

There are many implementations of the FLC, however, this particular architecture seemed to be used frequently and was consistent with the nominal PID design. For every step in the FLC process, there are different algorithms that can be used: membership functions, rule bases, and the output control signal algorithm. It is a widely used control technique, possibly lagging in the U.S., and certainly worth consideration where a controller is subject to changing operating and environmental conditions.

7.0 Summary: This completes the course. The topics covered began with a description of the basic feedback control loop block diagram in section 2. The PID control algorithm was described in section 3 describing different configurations followed by the digital implementation of the PID. The building temperature control example, used in Part 1 was described again in section 4.0 using a digital PI controller. Section 5.0 provided another example for a motion control application using a digital PD controller. Finally, section 6.0 described implementation of the PID within an MRAC architecture and as well as integrated with an FLC.

References;

1. Astrom, K., Hagglund T., PID Controllers: Theory, Design, and Tuning, Instrument Society of America, PO Box 12277, Research Triangle Park, NC 27709, ISBN 10: 155617-516-7, ISBN 13: 987-155617-516-9, 1995
2. Astrom, K., Hagglund T., Advanced PID Control, Instrument Society of America, PO Box 12277, Research Triangle Park, NC 27709, ISBN 10-155617-942-1, ISBN-13: 978-1556179-942-6, 2006
3. Astrom, K., Control System Design Lecture Notes ME 155A UCSB, Chapter 6, https://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom.html
4. Mishkin, E., Braun, L., Adaptive Control Systems, Brooklyn Polytechnic Institute Series, McGraw-Hill Book Company, 1961
5. Ljung, L., System Identification, Theory for the User, ISBN 0-13-881640-9, Prentice Hall, Englewood Cliffs, NJ 07672, 1987
6. M. Gupta (Editor), Adaptive Methods for Control System Design, ISBN 0-87942-207-6, Institute Electrical and Electronic Engineers, 345 East 47th Street, New York