Integral and Derivative Control Modes

As was discussed in a previous section, proportional control will always exhibit some degree of offset. Calibrating or tuning the controller at the point at which the process operates most frequently can minimize this offset. If necessary, one can also manually adjust the controller bias. This is not easily done on-the-fly with a pneumatic controller, but most microprocessor-based controllers allow you to easily change, or manually reset, the controller bias point.

Consider the discharge-air control system in Figure 1. The control action diagram

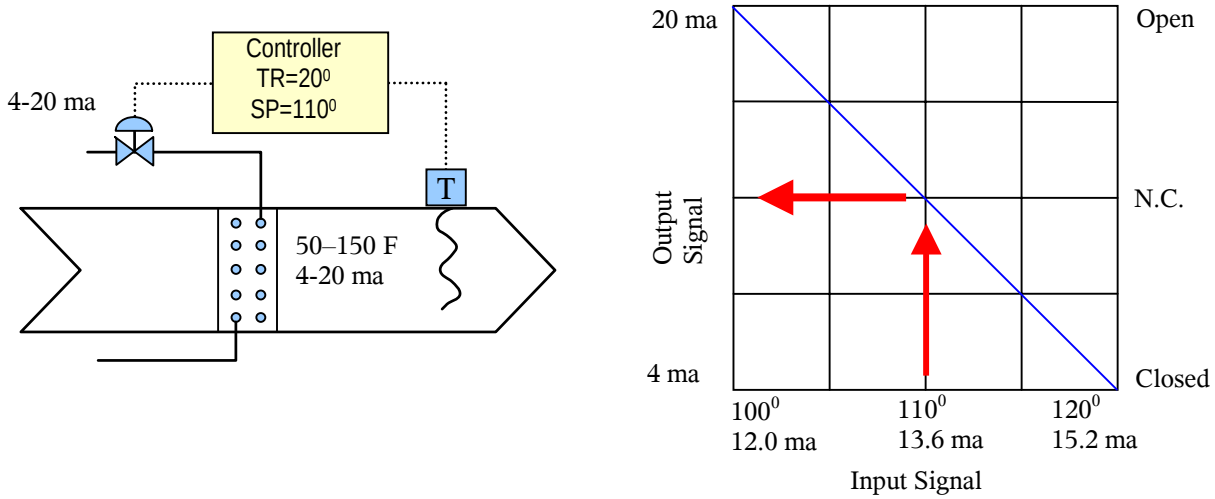

Figure 2 Discharge air control loop with calibration diagram

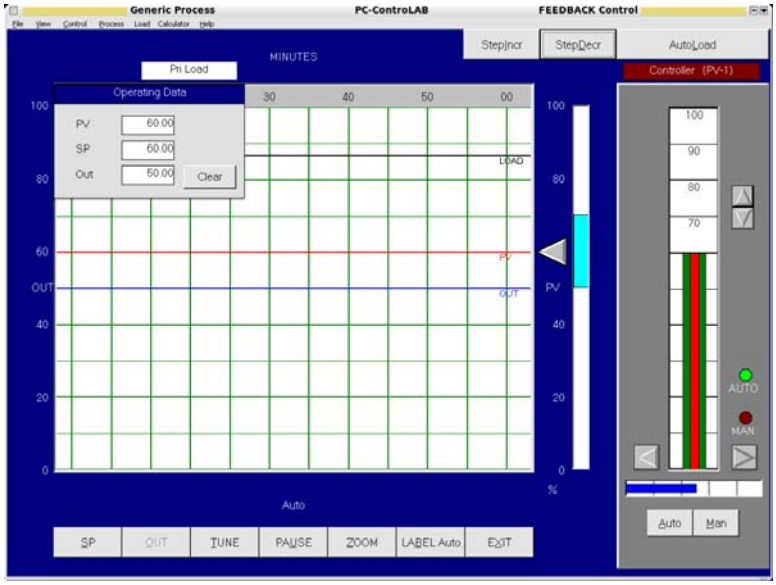

Figure 1 Simulation of system shown in Figure 1 (Screenshot of PC ControLab Simulation)

indicates a 50% bias. That is, the controller output is at midpoint when the controller is controlling at set point. This calibration condition is established under some arbitrary load. Figure 2 shows a simulation of this condition. The process variable and the controller output are expressed in percent of range. The arrow indicates the set point and the bar just right of the set point arrow represents the

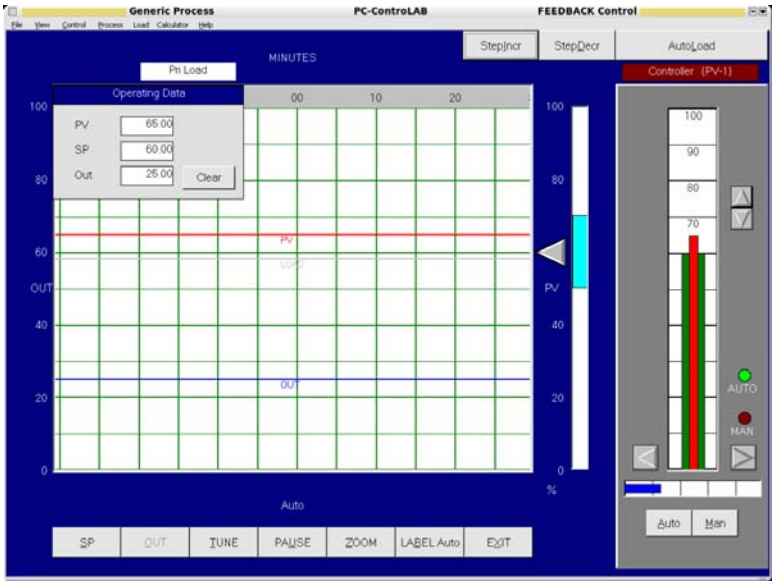

Figure 4 Illustration of offset for a proportional controller (Screenshot of PC ControLab Simulation)

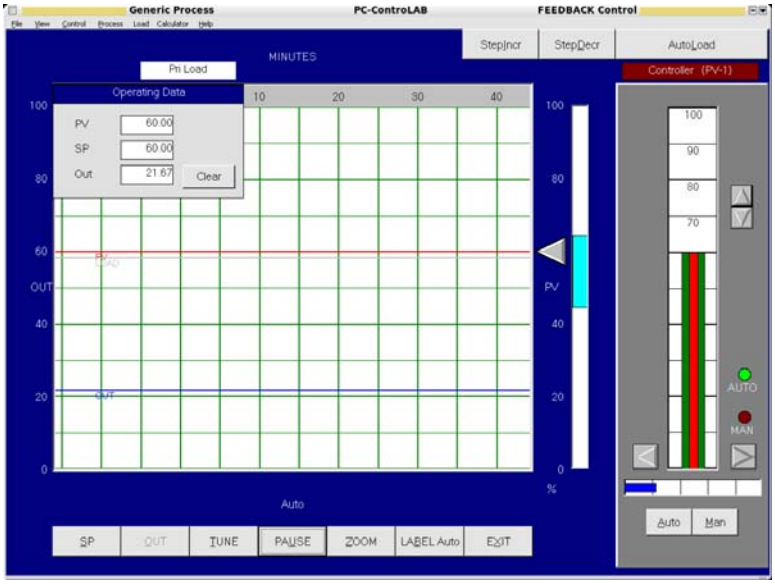

Figure 3 Controller bias adjusted to eliminate offset (Screenshot of PC ControLab Simulation)

proportional band. Note the arrow is at midpoint of the proportional band indicating a 50% bias. The result of this is a 50% controller output. For purposes of discussion, assume the load line shown in Figure 2 represents 87% of design load at this calibration point.

But consider the effect of a load reduction. In Figure 3, the load is reduced to about 58% of design load while the set point remains constant. Notice that although the control system reestablishes equilibrium by reducing output to about 25%, the process variable does not return to set point. In this case, the process variable achieves a value of 65% (115 ^{o}F). If one wanted to reduce this offset, one could do so by adjusting the controller bias setting. In this case, a controller bias of 21.67% eliminates offset forcing the controlled variable to attain the value of set point as illustrated in Figure 4. Note that the percent of bias signal equals the percent of controller output at that point when offset is forced to zero.

Automatic Reset

If it is necessary for a control loop to exhibit zero offset while in operation, it is not practical to continuously reset controller bias manually. Instead, we use automatic reset. Automatic reset is also referred to simply as reset or integral.

In mathematic terms, the integral is a method of finding the area under a curve. For example, consider the following graph of a parabola $y=x^2$. If one wanted to find the area under the curve between the values of 20 and 40, one would solve the following integral.

$$Area = \int_{20}^{40} x^2 = \frac{x^3}{3}\Big|_{20}^{40} = 18{,}667$$

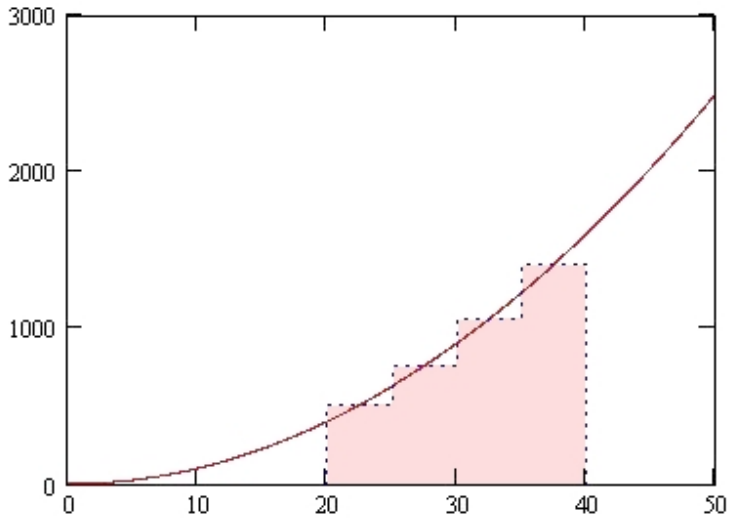

Figure 5 The integral function of a controller estimates the amount of error signal under an error curve by dividing it into finite rectangles. The width of the rectangle is the integral time constant

However, the implementation of a symbolic integral in a controller is impractical. Instead, the integral is estimated numerically in a fashion similar to Simpson's rule. The application of Simpson's rule estimates the area under the curve by dividing the curve into equal width rectangles. The height of each rectangle is such that the curve passes through the center of the rectangle. One then estimates the area by adding the area under each rectangle. In the example shown, each rectangle has a width of 5 units. Using this method, the estimate of the area under the curve between the limits of 20 and 40 is 18,625. Obviously, the wider the width of the rectangle, the less accurate is the estimation of area when compared to a symbolic integration.

So what does this have to do with process control? Consider the ideal controller equation for a PI controller.

$$Output = bias + K_c \times error + \frac{K_c}{T_i}\int error\ dt$$

The term T_i is known as an integral time constant. It is analogous to the width of the rectangle described above. This width will have units of time and each rectangle represents a repeat. Thus the integral time constant has units of hours, minutes or seconds per repeat. Some controllers require the integral time constant to be input as repeats per unit of time. Obviously, this is nothing more than the reciprocal of the time constant.

Using logic similar to that of Simpson's rule, the integrator determines the area under the error curve for a single repeat between the current value of the process signal and the set point. This area represents total error. Using this value, the integrator will increase or decrease controller output to drive the error to zero.

The output of the integrator logic is cumulative. Figure 6 shows a situation where a control system responds to a step change in set point. The error under each curve segment is as shown. However, the error is cumulative over time. Note that as the signal is driven to set point such that the error goes to zero, the cumulative error as seen by the integrator is non-zero. This is what allows the integrator to force the error to zero.

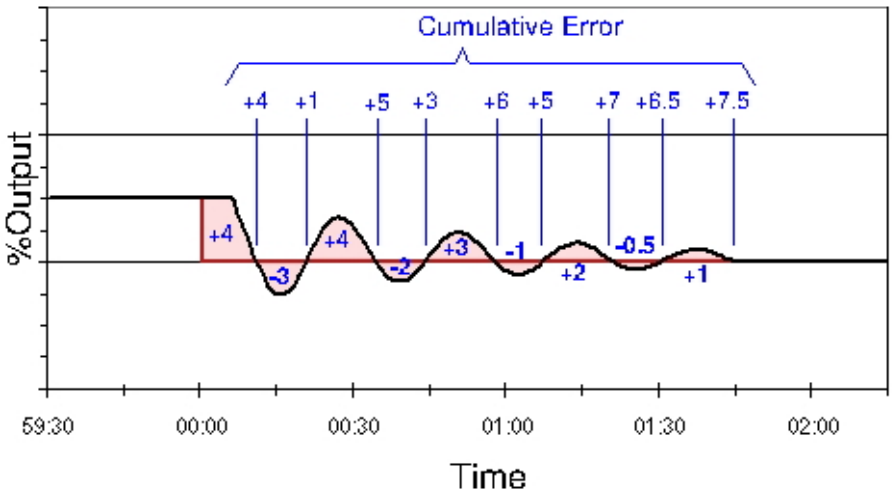

Figure 6 Integrator calculation of error for an oscillating control signal

The integrator cannot determine the error under an error curve until the error has occurred. As such, integral action always lags what is currently happening. For this reason, integral tends to destabilize a control loop. In order to provide some form of stability, one must reduce proportional gain. Unfortunately, the reduction of proportional gain tends to make the control loop less responsive. Despite this drawback, integral mode is recommended for most control loops.

Rate Control

As described, the use of integral in a control loop is a destabilizing factor requiring one to reduce the amount of proportional gain. This, in turn, tends to slow the response of the loop. To account for this drawback, the derivative mode is introduced. Mathematically, a derivative is the slope of a curve at any given point. The slope of a line is determined as the change in the y-variable divided by the change in the x-variable. When talking about control, this definition reduces to a change in the value of the controlled variable over a period of time. For this reason, derivative is often referred to as rate. When applied to a controller, this rate is actually determined over a finite time period. This period of time is referred to as the derivative time constant. The derivative logic of a controller uses this rate of change to determine whether the error signal is increasing or decreasing. As such, the derivative mode of a controller is anticipatory in nature. For this reason, it tends to stabilize a control loop and allows one to increase the amount of proportional gain thus improving control loop response.

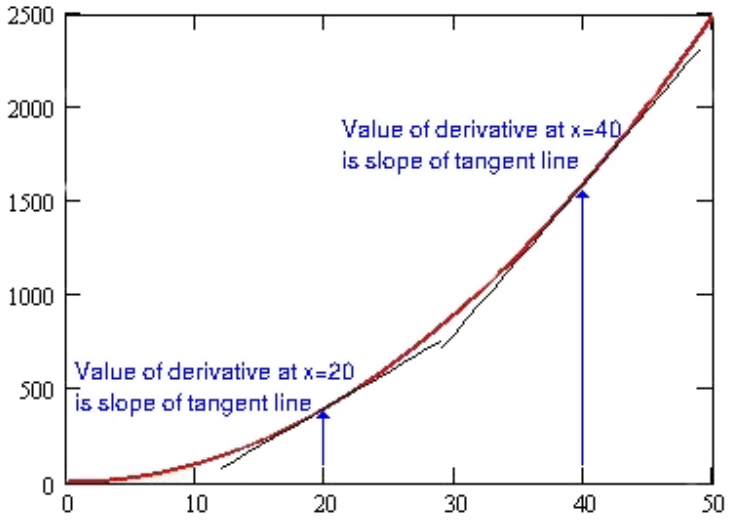

Figure 7 An illustration of the derivative of a function at two discrete points

When adding the derivative mode to a controller, the ideal controller equation becomes:

$$Output = bias + K_c \times error + \frac{K_c}{T_i} \int error\ dt + K_c \times T_d \times \frac{d}{dt}(error)$$

The variable T_d is the derivative time constant.

It is important to note the derivative output of a controller is non-zero only when the error is changing. If an error exists, but is not changing, the derivative logic will not

contribute to controller output. However, if the error signal is increasing, the derivative output of the controller will also increase. Conversely, if the error signal decreases, derivative contribution to controller output will also decrease.

The nature of derivative is such that it cannot be used for fast processes such as flow or pressure control, nor can it be used if the process signal is noisy. Fast processes do not benefit from derivative mode simply because they are already responsive. The addition of derivative will amplify the responsiveness resulting in instability. Noisy processes do not benefit from derivative because the rate of change of error is constantly changing. This results in an unstable derivative contribution to the controller output signal.

To better understand why these two statements, let's compare the response of the proportional, integral and derivative logic blocks of a three-mode controller to various types of input signals.

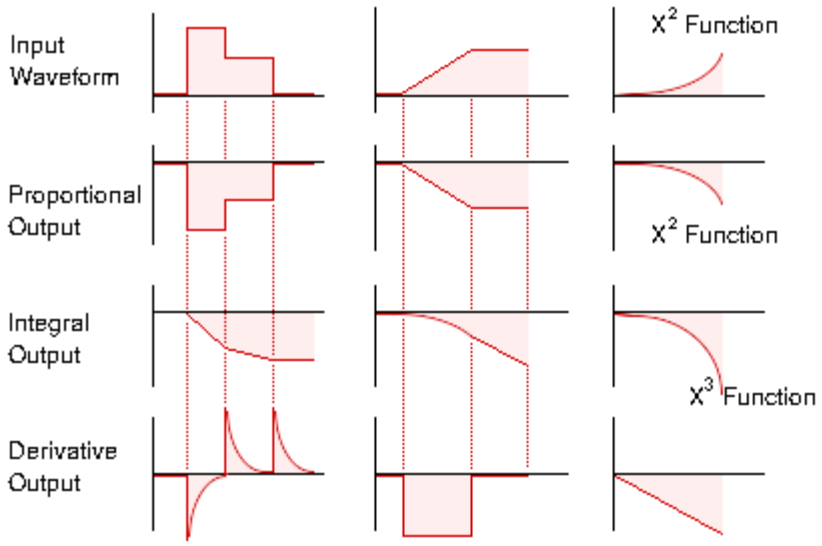

Figure 8 Response of each mode of a three-mode controller for a given input waveform

The first waveform in Figure 8 is a step function. The shape of the proportional output signal will mirror this input waveform. The integrator determines the area under the error curve. Over the first step of the step function, the area under the curve increases rapidly and at a constant rate. When the input function exhibits a step decrease, the rate of the integrator output also decreases, but will still hold constant. As the step input wave goes to zero, the integrator output will not change but will hold constant. On the other hand, the derivative output acts as shown. Remember that derivative output is a function of the rate of change of the error signal. The input function is a step function. The slope (rate of change) of a vertical line is infinity. Thus the derivative logic of the controller will spike

the controller output forcing it to saturation. It will then decay to zero within five derivative time constants.

The second waveform is a signal that ramps from zero to some maximum value, then holds constant. The proportional output will again mirror this input waveform. The integrator will determine the area under this ramping signal curve over time. The resulting output is a signal that increases as square function. When the input signal reaches its constant value, the integrator output will begin to ramp as the error signal continuously increases. It is this phenomenon that causes integral windup. Integral windup usually occurs when the process is shut down but the control system remains active. In such a case, the error signal is constantly increasing, thus the integrator output will continuously increase. Since the process is shut down, no corrective action takes place. As such, the integrator will eventually saturate the controller output. This can prove to be a problem on system start-up. Often, special precautions are taken to zero out the integrator output during the start-up procedure.

The derivative output for this ramping input signal is as shown. Derivative is a function of the rate of change of an error signal. A ramping input means the error increases at a constant rate, thus the derivative logic will provide an output equal to this rate of change. When the ramping input signal reaches a constant value, the amount of error is now constant. Since there is no rate of change in the error signal, the derivative output goes to zero.

The last signal is one that follows a parabolic or square function. In such a case, the proportional output will mirror this input signal. The integrator output will increase quite rapidly, in this case, as a cubic function. The derivative of such a function is a straight line. In this case, the derivative output will increase at a constant rate.

It should be noted the outputs shown in Figure 8 illustrate the shape of the output. The actual magnitude is a function of the proportional gain. In other words, if the proportional gain is equal to 2, then the controller will output a signal with a shape that mirrors the input signal, but will a magnitude twice that of the input signal. Table 1 summarizes the affect each controller mode has on various elements of control loop response.

	Proportional	Integral	Derivative
Stability	High gain destabilizes control loop	Integral has an inherent destabilizing effect. This is especially true if there is to little reset action.	Derivative has an inherent stabilizing effect. However, too much derivative will destabilize a loop.
Response Time	Higher values of gain increase response time.	Integral will increase the response time for low lag processes	Derivative is designed to improve the response time of a loop
Overshoot	A high gain tends to cause more overshoot	Integral tends to reduce overshoot for low lag processes	Derivative will decrease overshoot
Offset	High gain reduces offset	Integral is designed to eliminate offset	Derivative has no effect on offset
Settling Time	A high gain tends to increase settling time	Since integral has a destabilizing effect, it tends to increase settling time	Derivative generally will improve settling time.

Table 1 The effect of each mode of a three-mode controller on various aspects of the control loop